SAS® 9.3 Companion for UNIX Environments

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® 9.3 Companion for UNIX Environments*. Cary, NC: SAS Institute Inc.

SAS® 9.3 Companion for UNIX Environments

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

For a hardcopy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government Restricted Rights Notice: Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, July 2011
1st printing, August 2012
2nd electronic book, May 2015

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at `support.sas.com/publishing` or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contents

About This Book

Syntax Conventions for the SAS Language

Overview of Syntax Conventions for the SAS Language

SAS uses standard conventions in the documentation of syntax for SAS language elements. These conventions enable you to easily identify the components of SAS syntax. The conventions can be divided into these parts:

- syntax components
- style conventions
- special characters
- references to SAS libraries and external files

Syntax Components

The components of the syntax for most language elements include a keyword and arguments. For some language elements, only a keyword is necessary. For other language elements, the keyword is followed by an equal sign (=).

keyword
: specifies the name of the SAS language element that you use when you write your program. Keyword is a literal that is usually the first word in the syntax. In a CALL routine, the first two words are keywords.

In the following examples of SAS syntax, the keywords are the first words in the syntax:

CHAR (*string, position*)

CALL RANBIN (*seed, n, p, x*);

ALTER (*alter-password*)

BEST *w*.

REMOVE <*data-set-name*>

In the following example, the first two words of the CALL routine are the keywords:

CALL RANBIN(*seed*, *n*, *p*, *x*)

The syntax of some SAS statements consists of a single keyword without arguments:

DO;
... SAS code ...

END;

Some system options require that one of two keyword values be specified:

DUPLEX | NODUPLEX

argument
: specifies a numeric or character constant, variable, or expression. Arguments follow the keyword or an equal sign after the keyword. The arguments are used by SAS to process the language element. Arguments can be required or optional. In the syntax, optional arguments are enclosed between angle brackets.

In the following example, *string* and *position* follow the keyword CHAR. These arguments are required arguments for the CHAR function:

CHAR (*string, position*)

Each argument has a value. In the following example of SAS code, the argument *string* has a value of 'summer', and the argument *position* has a value of 4:**`x=char('summer', 4);`**

In the following example, *string* and *substring* are required arguments, while *modifiers* and *startpos* are optional.

FIND(*string, substring* <,*modifiers*> <,*startpos*>

Note: In most cases, example code in SAS documentation is written in lowercase with a monospace font. You can use uppercase, lowercase, or mixed case in the code that you write.

Style Conventions

The style conventions that are used in documenting SAS syntax include uppercase bold, uppercase, and italic:

UPPERCASE BOLD
: identifies SAS keywords such as the names of functions or statements. In the following example, the keyword ERROR is written in uppercase bold:

ERROR<*message*>;

UPPERCASE
: identifies arguments that are literals.

In the following example of the CMPMODEL= system option, the literals include BOTH, CATALOG, and XML:

CMPMODEL = BOTH | CATALOG | XML

italics
: identifies arguments or values that you supply. Items in italics represent user-supplied values that are either one of the following:

- nonliteral arguments In the following example of the LINK statement, the argument *label* is a user-supplied value and is therefore written in italics:

  LINK *label*;

- nonliteral values that are assigned to an argument

  In the following example of the FORMAT statement, the argument DEFAULT is assigned the variable *default-format*:

  FORMAT = *variable-1* <, ..., *variable-nformat*><DEFAULT = *default-format*>;

Items in italics can also be the generic name for a list of arguments from which you can choose (for example, *attribute-list*). If more than one of an item in italics can be used, the items are expressed as *item-1, ..., item-n*.

Special Characters

The syntax of SAS language elements can contain the following special characters:

=

an equal sign identifies a value for a literal in some language elements such as system options.

In the following example of the MAPS system option, the equal sign sets the value of MAPS:

MAPS = *location-of-maps*

< >

angle brackets identify optional arguments. Any argument that is not enclosed in angle brackets is required.

In the following example of the CAT function, at least one item is required:

CAT (*item-1* <*, ..., item-n*>)

|

a vertical bar indicates that you can choose one value from a group of values. Values that are separated by the vertical bar are mutually exclusive.

In the following example of the CMPMODEL= system option, you can choose only one of the arguments:

CMPMODEL = BOTH | CATALOG | XML

...

an ellipsis indicates that the argument or group of arguments following the ellipsis can be repeated. If the ellipsis and the following argument are enclosed in angle brackets, then the argument is optional.

In the following example of the CAT function, the ellipsis indicates that you can have multiple optional items:

CAT (*item-1* <*, ..., item-n*>)

'*value*' or "*value*"

indicates that an argument enclosed in single or double quotation marks must have a value that is also enclosed in single or double quotation marks.

In the following example of the FOOTNOTE statement, the argument *text* is enclosed in quotation marks:

FOOTNOTE <*n*> <*ods-format-options* '*text*' | "*text*">;

;

a semicolon indicates the end of a statement or CALL routine.

In the following example each statement ends with a semicolon: **`data namegame; length color name $8; color = 'black'; name = 'jack'; game = trim(color) || name; run;`**

References to SAS Libraries and External Files

Many SAS statements and other language elements refer to SAS libraries and external files. You can choose whether to make the reference through a logical name (a libref or fileref) or use the physical filename enclosed in quotation marks. If you use a logical name, you usually have a choice of using a SAS statement (LIBNAME or FILENAME) or the operating environment's control language to make the association. Several methods of referring to SAS libraries and external files are available, and some of these methods depend on your operating environment.

In the examples that use external files, SAS documentation uses the italicized phrase *file-specification*. In the examples that use SAS libraries, SAS documentation uses the italicized phrase *SAS-library*. Note that *SAS-library* is enclosed in quotation marks:

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```

What's New in the SAS 9.3 Companion for UNIX Environments

Overview

The following categories list the areas of change for SAS in UNIX environments:

- "Concatenating Files in autoexec.sas" on page xi
- "Deprecated Option" on page xi
- "Documentation Enhancements" on page xii
- "Encoding for Pathnames on Disk" on page xii
- "Identifying Where the Value of a SAS System Option Is Set" on page xii
- "SAS Output" on page xii
- "Processing Files on Tape" on page xiii
- "SAS Statement Option" on page xiii
- "SAS System Options" on page xiii
- "SAS Window Enhancement" on page xiii

In the second maintenance release for SAS 9.3, a section was added that shows how to convert a UNIX datetime value to a SAS datetime value on page xiv.

Concatenating Files in autoexec.sas

You can concatenate your files in an autoexec.sas file by using the APPEND and INSERT system options with the AUTOEXEC system option. For more information, see "Inserting and Appending Autoexec Files" on page 22.

Deprecated Option

The PRODTOC option has been deprecated.

Documentation Enhancements

- The error message in the SYSTASK statement has been updated to provide more information about the error.
- References to TAPE engines have been removed from the documentation, along with references to the FILECLOSE= data set option, which specifies how a tape is positioned when a SAS data set is closed.

Encoding for Pathnames on Disk

SAS normally uses the default session encoding when referencing external files and directories. The PATHENCODING environment variable provides an alternative encoding for external file and directory references. PATHENCODING is valid only for files that are located on disk. When the PATHENCODING environment variable has a valid encoding value, SAS transcodes the pathname in the specified encoding. For a list of valid encoding values on UNIX, see “UNIX Encoding Values” in Chapter 21 of *SAS National Language Support (NLS): Reference Guide*.

Identifying Where the Value of a SAS System Option Is Set

The VALUE option in the PROC OPTIONS statement enables you to identify how the option was set (for example, in a configuration file, on a command line, and so on). For more information, see “Determining How a SAS System Option Was Set” on page 349.

SAS Output

The following changes were made to SAS output:

- When you run SAS interactively, the LISTING destination is closed by default. The HTML destination is open by default, and HTMLBlue is the default style. Also, ODS Graphics is enabled by default.
- When you run SAS in batch mode, the LISTING destination is open and is the default. ODS Graphics is not enabled by default.

Processing Files on Tape

If you have files on tape, use a staging directory so that files can be processed directly from disk. The use of tape drives on UNIX is no longer supported.

SAS Statement Option

The following statement option has been enhanced:

NOSETPERM (p. 335)
: The NOSETPERM LIBNAME option affects assignments to a path. The option specifies that permission settings are not inherited from one library member to another library member when members are open with the same libref.

SAS System Options

The following system options have been enhanced:

ALTLOG (p. 366)
: The ALTLOG system option can use directives to enable you to control when log copies are open and closed, and how they are named, based on real-time events such as time, month, and day of week.

AUTOEXEC (p. 370)
: The AUTOEXEC system option supports the use of multiple files. You can use the APPEND and INSERT options to concatenate multiple files in your autoexec.sas file.

STIMEFMT (p. 427)
: The STIMEFMT system option enables you to customize the format of the output from the STIMER and FULLSTIMER system options.

VERBOSE (p. 434)
: As in SAS 9.2, the output from the VERBOSE system option lists the options and their values. In 9.3, an additional list is created that identifies where the options were set. This list is first written to a journal file, and then it is written to the SAS log. If SAS fails to initialize, output is still available even though a SAS log was not created.

SAS Window Enhancement

The **Results** tab in Preferences dialog box has a new check box, **Use ODS Graphics**. Checking this box enables you to automatically generate graphs when running procedures that support ODS graphics. **Use ODS Graphics** is turned on by default.

Converting a UNIX Datetime Value to a SAS Datetime Value

A UNIX datetime value is stored as the number of seconds since January 1, 1970. A SAS datetime value is stored as the number of seconds since January 1, 1960. To convert a UNIX datetime value to a SAS datetime value, you must add 10 years in seconds to the UNIX datetime value. For more information, see "Converting a UNIX Datetime Value to a SAS Datetime Value" on page 219.

Recommended Reading

- *Base SAS Procedures Guide*
- *SAS Data Set Options: Reference*
- *SAS Formats and Informats: Reference*
- *SAS Functions and CALL Routines: Reference*
- *SAS Language Reference: Concepts*
- *SAS Macro Language: Reference*
- *Moving and Accessing SAS Files*
- *SAS National Language Support (NLS): Reference Guide*
- *SAS Output Delivery System: User's Guide*
- *SAS Statements: Reference*
- *SAS System Options: Reference*

For a complete list of SAS publications, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

Part 1

Running SAS Software Under UNIX

2

Chapter 1

Getting Started with SAS in UNIX Environments

Starting SAS Sessions in UNIX Environments

Invoking SAS

A SAS session is invoked using a link in the **!SASROOT** directory. (The **!SASROOT** directory is a term that represents the name of the directory or folder in which SAS is installed at your site or on your computer.) Your UNIX administrator can add this link to the list of commands for your operating environment. For more information about the **!SASROOT** directory, see “Introduction to the !SASROOT Directory” on page 443.

Ask your system administrator for the command that invokes SAS at your site. At many sites, the command to invoke SAS is **sas**, but a different command might have been defined during the SAS installation process at your site. This documentation assumes that SAS is invoked by the **sas** command.

Note: Before you start your SAS session, review the different techniques for interrupting and terminating your SAS session. For more information, see “Exiting or Interrupting Your SAS Session in UNIX Environments” on page 27. Also, if you cannot stop your SAS session, contact your system administrator.

SAS Invocation Scripts

SAS is invoked by scripts that are located in the **`!SASROOT/bin`** directory. A SAS invocation script is created for each language that is installed. The invocation scripts are named using the language codes of the installed language. For example, **`sas_en`** invokes the English version of SAS. All languages are installed in all locations.

For more information about setting up SAS, refer to the installation documentation for the UNIX environment.

SAS Configuration Files

SAS creates a separate configuration file for each language that is installed. The language-specific configuration files have the form **`!SASROOT/nls/<language>/sasv9.cfg`** for each language. An additional configuration file that is language independent is **`!SASROOT/sasv9.cfg`**. This master configuration file in is used by all languages in addition to the language-specific files in **`!SASROOT/nls/<language>/`**. You can modify these configuration files to meet your needs. For information about how to customize SAS configuration files, see "Customizing Your SAS Session by Using Configuration and Autoexec Files" on page 21.

Syntax of the SAS Command

The general form of the SAS command is as follows:

sas <*-option1*…*-option-n*> <*filename*>

sas –sysin *filename*

You can use these arguments with the SAS command:

-option1 ... *-option-n*
: specifies SAS system options to configure your session or X command line options. For more information, see "SAS System Options under UNIX" on page 348 and "X Command Line Options" on page 13. If you omit any options (either on the command line or in the configuration file), the SAS (or site-specific) default options are in effect.

filename
: specifies the name of the file containing the SAS program to be executed. Specifying a filename on the SAS command invokes a batch SAS session. Omit the filename to begin an interactive session.

: If the file is not in the current directory, specify its full pathname. A .sas extension is inferred if the full pathname is not given.

: *Note:* This command can fail in cases where an option does not recognize *filename*. In this case, **`-sysin`** ***`filename`*** is required.

Example: Invoke an Interactive SAS Session

To invoke an interactive SAS session, without specifying any SAS system options, enter

```
sas
```

The execution mode will depend on your default settings. For more information, see "Selecting a Method of Running SAS in UNIX Environments" on page 7.

To specify the NODATE and LINESIZE system options, you could enter

```
sas -nodate -linesize 80
```

What If SAS Does Not Start?

There are several reasons why SAS might not start. Three reasons are listed here:

- SAS will not start if you specify an autoexec file that does not exist. An error message will appear in the SAS log stating that the physical file does not exist.
- SAS will not start if you specify invalid options, such as a misspelled option: **`./sas –nodms –stimerr (stimer is misspelled)`**.
- SAS will not start if SAS cannot find the configuration file. This error normally indicates an installation problem.

If SAS does not start, the SAS log might contain error messages that explain the failure. However, error messages that SAS issues before the SAS log is initialized are written to the console log. With the addition of better error handling, a failure could cause information to be written to standard output as well.

If the system is not patched correctly, SAS can generate an error such as NLS Extension Failure. This and other types of error messages indicate that the installation did not set up the search rules correctly.

Under UNIX, the STDOUT fileref specifies the location of the console log.

Running SAS in a Foreground or Background Process

UNIX is a multiprocessing operating system, so you can run multiple processes at the same time. For example, you can have one process running in the foreground and three in the background.

A foreground process executes while you wait for the prompt. That is, you cannot execute additional commands while the current command is being executed. After you enter a command, the shell starts a process to execute the command. After the system executes the command, the shell displays the prompt and you can enter additional commands. The following is an example of SAS executing as a foreground process:

```
sas
```

Running in the foreground enables you to access standard input and output.

A background process executes independently of the shell. After you enter a command, the shell starts a process to execute the command, and then issues the system prompt. You can enter other commands or start other background processes without waiting for your initial command to execute. The following is an example of the command that is used to execute a background process:

```
sas&
```

Note: Both the C shell and the Korn shell include commands that enable you to move jobs among three possible states: running in the foreground, running in the background, and suspended. If you run SAS in –nodms mode, the process stops waiting for input. In dms mode, control of standard output and input is retained by the shell.

Selecting a Method of Running SAS in UNIX Environments

You can run SAS in the SAS windowing environment mode, interactive line mode, and batch mode:

- "Invoking SAS in the Windowing Environment" on page 8
- "Interactive Line Mode in UNIX Environments" on page 9
- "Batch Mode in UNIX Environments" on page 10

Ask your UNIX system administrator which interface or mode of operation is the default at your site.

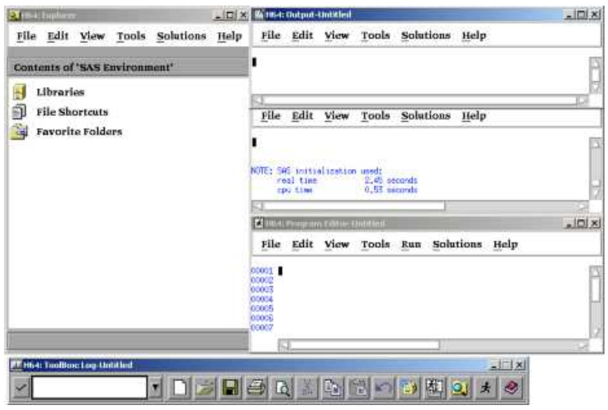
SAS Windowing Environment in UNIX Environments

Introduction to the SAS Windowing Environment

SAS Windows

You interact with SAS through windows using the keyboard, mouse, menus, and icons. The windowing environment includes, but is not limited to, the Explorer, Program Editor, Output, Log, and Results windows. The following display shows the Explorer, Output, Log, and Program Editor windows. The ToolBox window is also displayed.

Display 1.1 *Windows in the SAS Windowing Environment*


H64: Output-Untitled
File Edit View Tools Solutions Help
Contents of 'SAS Environment'
Libraries
File Shortcuts
Favorite Folders
File Edit View Tools Run Solutions Help
H64: ToolBox: Log-Untitled

```
NOTE: SAS initialization used:
      real time           2.45 seconds
      cpu time            0.53 seconds
```

Your SAS session might default to the windowing environment interface. (You can change the default by using the config files.) If you want to use the windowing environment, you can start your SAS session as a foreground process, or as a background process by adding an ampersand (&) to your SAS command line. See "Running SAS in a Foreground or Background Process" on page 6 for an example of these SAS commands.

For more information about using the windowing environment, see "Definition of the SAS Windowing Environment" on page 140.

Note: If you are not using an X display, then you can invoke SAS in interactive line mode by using the NODMS system option. For more information, see "Interactive Line Mode in UNIX Environments" on page 9.

What Is the Explorer Window?

Explorer is a windowing environment for managing basic SAS software tasks such as viewing and managing data sets, libraries, members, applications, and output. The SAS Explorer is a central access point from which you can do the following:

- manipulate SAS data through a graphical interface
- access the Program Editor, Output, and Log windows (as well as other windows)
- view the results of SAS procedure output in the Results window
- import files into SAS

What Are the Program Editor, Output, and Log Windows?

The Program Editor, Output, and Log windows enable you to edit and execute SAS programs and display output. For more information about these windows, see the online SAS Help and Documentation.

Invoking SAS in the Windowing Environment

You can use the following commands to specify which windows open when a SAS session starts.

- You can open the Program Editor, Output, and Log windows by specifying the DMS system option:

  ```
  sas -dms
  ```

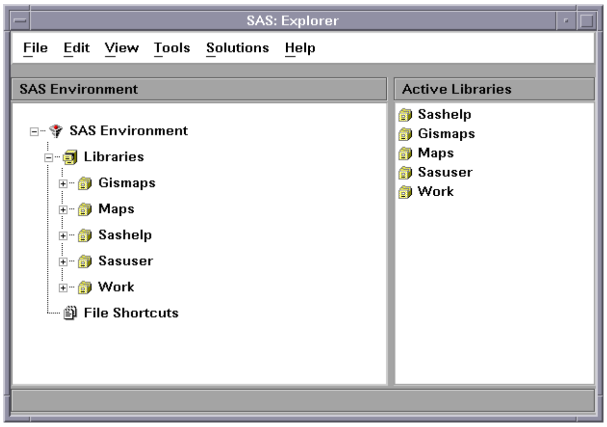
- You can open the Program Editor, Output, Log, and Results windows, as well as the Explorer window, by specifying the DMSEXP system option:

  ```
  sas -dmsexp
  ```

- You can open only the Explorer window by specifying the EXPLORER system option:

  ```
  sas -explorer
  ```

Display 1.2 *SAS Explorer Window*



The default specification for invoking SAS is **`sas -dmsexp`**. This command displays the Program Editor, Output, Log, and Results windows as well as the Explorer window. If you invoke SAS without the **`-dmsexp`** option, the Explorer window does not display.

SAS also opens a tool box from which you can open additional SAS windows. For more information about the tool box, see "Working in the SAS Windowing Environment" on page 140.

Exiting SAS in the Windowing Environment

To end your SAS session, enter the BYE or ENDSAS command in the command window, or select **File** ⇨ **Exit** from the menu of the SAS session that you want to end.

Interactive Line Mode in UNIX Environments

Introduction to Interactive Line Mode

If you are not using an X display, you can invoke SAS in interactive line mode by using the NODMS system option.

You enter SAS statements line by line in response to prompts issued by SAS. SAS reads the source statements from the terminal as you enter them. DATA and PROC steps execute when one of the following occurs:

- a RUN, QUIT, or DATALINES statement is entered
- another DATA or PROC statement is entered
- the ENDSAS statement is entered

To use interactive line mode, you must run SAS in the foreground.

Invoking SAS in Interactive Line Mode

To start an interactive line mode session, invoke SAS with the NODMS or NODMSEXP system option:

```
sas -nodms
sas -nodmsexp
```

By default, SAS log and procedure output (if any) appear on your display as each step executes.

You can also invoke SAS in interactive line mode and pass parameters to it:

```
sas -sysparm 'A B C'
```

The value **`A B C`** is assigned to the SYSPARM macro variable. You can include a program name, such as **`progparm.sas`** in the program editor or from the SAS command prompt if you invoked SAS in line mode by using the –nodms option.

After you invoke SAS, the **`1?`** prompt appears, and you can begin entering SAS statements. After you enter each statement, a line number prompt appears.

Exiting SAS in Interactive Line Mode

You can end the session by pressing the EOF key, usually CTRL-D (see "Using Control Keys" on page 29) or by issuing the ENDSAS statement:

```
endsas;
```

The session ends after all SAS statements have executed.

Batch Mode in UNIX Environments

Introduction to Running SAS in Batch Mode

To run SAS in batch mode, you specify your SAS program name in the SAS invocation command. You can run batch mode in the foreground, in the background by specifying an ampersand at the end of the SAS command, or submit your application to the batch queue by using the **batch**, **at**, **nohup**, or **cron** UNIX commands. (For more information, refer to the UNIX man pages for the **batch**, **at**, **nohup**, or **cron** commands.) If you start your application with one of these UNIX commands and you log off from your system, then your application will complete execution. If your application contains statements that start an interactive procedure such as FSEDIT, then you need to run your batch application in the foreground or you need to specify the –noterminal option.

Invoking SAS in Batch Mode

To invoke SAS in batch mode, you must specify a filename in the SAS command. For example, if **weekly.sas** is the file that contains the SAS statements to be executed, and you want to specify the NODATE and LINESIZE system options, you would enter the following command:

```
sas weekly.sas -nodate -linesize 90
```

The command would run the program in the foreground. If you want to run the program in the background, add the ampersand to the end of the command:

```
sas weekly.sas -nodate -linesize 90 &
```

SAS creates a .log file and a .lst file in the current directory that contains the log and procedure output.

Submitting a Program to the Batch Queue

To submit your program to the batch queue, you can use the **batch**, **at**, **nohup**, or **cron** commands. For example, you could submit **weekly.sas** from your shell prompt as follows:

```
$ at 2am
sas weekly.sas
<control-D>
warning: commands will be executed using /usr/bin/sh
job 8400.a at Wed Mar 16 02:00:00 2011
$
```

If you create a file that contains the SAS command (for example, **cmdfile.sh**) that is necessary to run your program, then you can enter the following command at your shell prompt:

```
at 2am < cmdfile.sh
```

SAS sends the output to a file that has the same name as the program. The output file has an extension of .lst. The log file writes to a file with an extension of .log. Both of these files are written to your current directory. Refer to the UNIX man pages for these commands for more information about submitting jobs to the batch queue. For more information about routing output, see "Printing and Routing Output" on page 92.

If you submit a file in batch mode, then a line that is greater than 256 bytes will be truncated. An explicit message about this truncation is written to the SAS log.

Note: If your program contains statements that start an interactive procedure such as the FSEDIT procedure, CATALOG procedure, or the REPORT procedure, you will need to run your program as a foreground process, or you will need to use the –noterminal option.

Writing Data from an External File Using UNIX Pipes

You can use a UNIX pipe to write data from an external file to a SAS program. For example, suppose that your data resides in the external file **mydata** and your SAS program **myprog.sas** includes this statement:

```
infile stdin;
```

Issue this command to have **myprog.sas** read data from the external file **mydata**:

```
cat mydata | sas myprog.sas
```

For information about using external files, see "Using External Files and Devices" on page 70. For information about another way to have a SAS program read data from an external file, see "File Descriptors in the Bourne and Korn Shells" on page 80.

Running SAS on a Remote Host in UNIX Environments

Introduction to Running SAS on a Remote Host

When you invoke SAS in an interactive mode, you can run SAS on your local host, or you can run SAS on a remote host and interact with the session through an X server running on your workstation. The server provides the display services that are needed for the X Window System.

Most of the time, the server name is derived from the computer's name. For example, if your computer is named **green**, the name of the server is **green:0.0**. In most cases, the X server will already be running when you log in. If you need to start your server manually, consult the documentation that is provided with your X Window System software.

To run SAS on a remote host, you must tell SAS which display to use by either setting the DISPLAY environment variable or specifying the **-display** X command line option.

Steps for Running SAS on a Remote Host

To run SAS on a remote host you must tell SAS which display to use by either setting the DISPLAY environment variable prior to invoking SAS or by specifying the**-display x** as a SAS command line option. Then follow these steps:

1. Make sure that the clients running on the remote host have permission to connect to your server. With most X servers, authorization is controlled by using an .Xauthority file that is located in the user's home directory. In addition, the **xhost** command can be used to circumvent authority. To use the **xhost** client to permit all remote hosts to connect to your server, enter the following command at the system prompt on the system that is running your X server:

   ```
   xhost +
   ```

   If your system does not control access with the **xhost** client, consult your system documentation for information about allowing remote access.

   For information about editing and displaying authorization information, see the UNIX man page for xauth.

2. Log in to the remote system, or use a remote shell.

3. Identify your server as the target display for X clients that are run on the remote host. You can identify your server in one of two ways:

   a. Set the DISPLAY environment variable. In the Bourne and Korn shells, you can set the DISPLAY variable as follows:

      ```
      DISPLAY=green:0.0
      export DISPLAY
      ```

      In the Korn shell, you can combine these two commands:

      ```
      export DISPLAY=green:0.0
      ```

      In the C shell, you must use the UNIX **setenv** command:

      ```
      setenv DISPLAY green:0.0
      ```

      The DISPLAY variable will be used by all X clients on the system.

      Note: To determine the shell for your current system, type **ps** at the UNIX command prompt or check the value of the SHELL environment variable.

   b. Use the DISPLAY system option. For example:

      ```
      sas -display green:0.0
      ```

      If you have trouble establishing a connection, you can try using an IP address instead of a display name, for example:

      ```
      -display 10.22.1.1:0.0
      ```

      Note: This option is a command line option for the X Window System, not for SAS. Specifying this option in a SAS configuration file or in the SASV9_OPTIONS environment variable might cause problems when you are running other interfaces.

Preventing SAS from Attempting to Connect to the X Server

To prevent SAS from attempting to connect to the X server, unset the DISPLAY environment variable and use the **`-noterminal`** SAS option on the command line. The **`-noterminal`** option specifies that you do not want to display the SAS session. You must specify this option to generate a graph in batch mode. You must also specify this option when you use PROC IMPORT and PROC EXPORT. For more information, see "Running SAS/GRAPH Programs" in *SAS/GRAPH: Reference*.

Troubleshooting Connection Problems

If SAS cannot establish a connection to your display, it prints a message that indicates the nature of the problem and then terminates. An example of a message that you might receive is the following:

```
ERROR:  The connection to the X display server could not be made.
        Verify that the X display name is correct, and that you have
        access authorization. See the online Help for more information
        about connecting to an X display server.
```

Make sure that you have brought up the SAS session correctly. You might need to use the **`xhost`** client (enter **`xhost +`**) or some other method to change display permissions. You can also specify the NODMS system option when you invoke SAS to bring your session up in line mode.

If you are unable to invoke SAS, try running another application such as **`xclock`**. If you cannot run the application, you should contact your UNIX system administrator for assistance.

X Command Line Options

How to Specify X Window System Options

When you invoke some X clients, such as SAS, you can use command line options that are passed to the X Window System. In general, you should specify X Window System options after SAS options on the command line.

Supported X Command Line Options

The following list describes the X command line options that are available when you invoke a SAS session from the command prompt.

-display *host*:*server.screen*
: specifies the name or IP address of the terminal on which you want to display the SAS session. For example, if your display node is **`wizard`** whose IP address is 10.22.1.1:0.0, you might enter

```
-display wizard:0.0
```

or

```
-display 10.22.1.1:0.0
```

-name *instance-name*
: reads the resources in your SAS resource file that begin with *instance-name*. For example, **`-name MYSAS`** reads the resources that begin with **`MYSAS`**, such as

```
MYSAS.dmsfont: Cour14

MYSAS.defaultToolbox: True
```

-title *string*
: specifies a title for your SAS session window. Titles can contain up to 64 characters. Window titles are displayed in the case in which they are entered, which can be lower case, mixed case, or upper case. To use multiple words in the title, enclose the words in single or double quotation marks. For example, **`-title MYSAS`** produces **`MYSAS:Explorer`** in the title bar of the Explorer window.

-xrm *string*
: specifies a resource to override any defaults. For example, the following resource turns off the Confirm dialog box when you exit SAS:

```
-xrm 'SAS.confirmSASExit: False'
```

Unsupported X Command Line Options

SAS does not support the following X command line options because their functionality is not applicable to SAS or is provided by SAS resources. For more information about SAS resources, see "Overview of X Resources" on page 164.

-geometry
: Window geometry is specified by the **`SAS.windowHeight`**, **`SAS.windowWidth`**, **`SAS.maxWindowHeight`**, and **`SAS.maxWindowWidth`** resources.

-background, -bg
: These options are ignored.

-bordercolor, -bd
: These options are ignored. Refer to "Defining Colors and Attributes for Window Elements (CPARMS)" on page 200 for a description of specifying the color of window borders.

-borderwidth, -bw
: These options are ignored. The width of window borders is set by SAS.

-foreground, -fg
: These options are ignored.

-font, -fn
: SAS fonts are specified by the **`SAS.DMSFont`**, **`SAS.DMSboldFont`**, and **`SAS.DMSfontPattern`** resources.

-iconic
: This option is ignored.

-reverse, -rv, +rv
: These options are ignored. For more information about a description for specifying reverse video, see "Defining Colors and Attributes for Window Elements (CPARMS)" on page 200.

-selectionTimeout
: Time-out length is specified by the **`SAS.selectTimeout`** resource.

-synchronous, +synchronous
: The XSYNC command toggles synchronous communication between SAS and the X server.

-xn1language
: This option is ignored.

Executing Operating System Commands from Your SAS Session

Deciding Whether to Run an Asynchronous or Synchronous Task

You can execute UNIX commands from your SAS session either asynchronously or synchronously. When you run a command as an asynchronous task, the command executes independently of all other tasks that are currently running. To run a command asynchronously, you must use the SYSTASK statement. See "SYSTASK Statement: UNIX" on page 340 for information about executing commands asynchronously.

When you execute one or more UNIX commands synchronously, you must wait for those commands to finish executing before you can continue working in your SAS session. You can use the CALL SYSTEM routine, %SYSEXEC macro program statement, X statement, and X command to execute UNIX commands synchronously. The CALL SYSTEM routine can be executed with a DATA step. The %SYSEXEC macro statement can be used inside macro definitions, and the X statement can be used outside of DATA steps and macro definitions. You can enter the X command on any SAS command line. For more information, see "CALL SYSTEM Routine: UNIX" on page 263 and "Macro Statements in UNIX Environments" on page 289.

Executing a Single UNIX Command

Single Commands

To execute only one UNIX command, you can enter the X command, X statement, CALL SYSTEM routine, or %SYSEXEC macro statement as follows:

X *command*

X *command*;

CALL SYSTEM ('*command*');

%SYSEXEC *command*;

Note: When you use the %SYSEXEC macro statement, if the UNIX command that you specify includes a semicolon, you must enclose the UNIX command in a macro quoting function. For more information about quoting functions, see *SAS Macro Language: Reference*.

Example 1: Executing a UNIX Command by Using the X Statement

You can use the X statement to execute the **ls** UNIX command (in a child shell) as follows:

```
x ls -l;
```

Example 2: Executing a UNIX Command by Using the CALL SYSTEM Routine

Inside a DATA step, you can use the CALL SYSTEM routine to execute a **`cd`** command, which will change the current directory of your SAS session:

```
data _null_;
call system ('cd /users/smith/report');
run;
```

The search for any relative (partial) filenames during the SAS session will now begin in the **`/users/smith/report`** directory. When you end the session, your current directory will be the directory in which you started your SAS session.

For more information about the CALL SYSTEM routine, see “CALL SYSTEM Routine: UNIX” on page 263.

How SAS Processes a Single UNIX Command

When you specify only one command, SAS checks to see whether the command is **`cd`**, **`pwd`**, **`setenv`**, or **`umask`** and, if so, executes the SAS equivalent of these commands. The SAS **`cd`** and **`pwd`** commands are equivalent to their Bourne shell counterparts. The SAS **`setenv`** command is equivalent to its C shell namesake. The SAS **`umask`** command is equivalent to the numeric mode of the **`umask`** command supported by the Bourne, Korn, and C shells. These four commands are built into SAS because they affect the environment of the current SAS session. When executed by SAS software, they affect only the SAS environment and the environment of any shell programs started by the SAS session. They do not affect the environment of the shell program that began your SAS session.

If the command is not **`cd`**, **`pwd`**, or **`setenv`**, SAS starts a shell in which it executes the command that you specified. The shell that is used depends on the SHELL environment variable. If the command is **`umask`**, but you do not specify a *mask*, then SAS passes the command to the shell in which the current SAS session was started. For more information about the **`umask`** command, see “Changing the File Permissions for Your SAS Session” on page 17.

Executing Several UNIX Commands

Executing UNIX Commands

You can also use the X command, X statement, CALL SYSTEM routine, and %SYSEXEC macro statement to execute several UNIX commands:

X '*command-1*;...*command-n*'

X '*command-1*;...*command-n*';

CALL SYSTEM ('*command-1*;...*command-n*');

%SYSEXEC *quoting-function*(*command-1*;...*command-n*);

Separate each UNIX command with a semicolon (;).

Note: When you use the %SYSEXEC macro statement to execute several UNIX commands, because the list of commands uses semicolons as separators, you must enclose the string of UNIX commands in a macro quoting function. For more information about quoting functions, see *SAS Macro Language: Reference*.

Example: Executing Several Commands Using the %SYSEXEC Macro

The following code defines and executes a macro called **`pwdls`** that executes the **`pwd`** and **`ls -l`** UNIX commands:

```
%macro pwdls;
%sysexec %str(pwd;ls -l);
%mend pwdls;
%pwdls;
```

This example uses **`%str`** as the macro quoting function.

How SAS Processes Several UNIX Commands

When you specify more than one UNIX command (that is, a list of commands separated by semicolons), SAS passes the entire list to the shell and does not check for the **`cd`**, **`pwd`**, **`setenv`**, or **`umask`** commands, as it does when a command is specified by itself (without semicolons).

For more information about how SAS processes the **`cd`**, **`pwd`**, **`setenv`**, or **`umask`** commands, see "How SAS Processes a Single UNIX Command" on page 16.

Changing the File Permissions for Your SAS Session

At invocation, a SAS session inherits the file permissions from the parent shell. Any file that you create will inherit these permissions. If you want to change or remove file permissions from within SAS, issue the following command in the X statement: **`umask`**. The **`umask`** command applies a new "mask" to a file, that is, it sets new file permissions for any new file that you create. In this way, the **`umask`** command can provide file security by restricting access to new files and directories for the current process.

The default value for **`umask`** varies. Some systems, like Secure Linux, use a default of 220. Other systems use 022 as the default. System administrators can set their own default value and you can check your default and change it in your own .kshrc, .cshrc, or .profile files. These values affect all child processes that are executed in the shell. Any subsequent file that you create during the current SAS session will inherit the permissions that you specified. The permissions of a file created under a given mask are calculated in octal representation.

Note: The value of a mask can be either numeric or symbolic. For more information about this command, see the UNIX man page for **`umask`**.

Executing X Statements in Batch Mode

If you run your SAS program in batch mode and if your operating system supports job control, the program will be suspended when an X statement within the program needs input from the terminal.

If you run your SAS program from the batch queue by submitting it with the **`at`** or **`batch`** commands, SAS processes any X statements as follows:

- If the X statement does not specify a command, SAS ignores the statement.
- If any UNIX command in the X statement attempts to get input, it receives an end-of-file (standard input is set to **`/dev/null`**).
- If any UNIX command in the X statement writes to standard output or standard error, the output is mailed to you unless it was previously redirected.

Customizing Your SAS Registry Files

SAS registry files store information about the SAS session. The SAS registry is the central storage area for configuration data for SAS. The following list identifies some of the data that is stored in the registry:

- the libraries and file shortcuts that SAS assigns at start–up. These shortcuts could include secure information, such as your password.
- the printers that are defined for use and their print setup.
- configuration data for various SAS products.

The Sasuser registry file (called regstry.sas7bitm) contains your user defaults. These registry entries can be customized by using the SAS Registry Editor or by using PROC REGISTRY. For more information, see “The SAS Registry” in *SAS Language Reference: Concepts*.

CAUTION:

For experienced users only. Registry customization is generally performed by experienced SAS users and system administrators.

Customizing Your SAS Session by Using System Options

Ways to Customize Your SAS Session

You can customize your SAS environment in several ways. One way is through the use of SAS system options. For information about other ways to customize a SAS session, see “Overview of Customizing SAS in X Environment” on page 164.

Ways to Specify a SAS System Option

SAS options can be specified in one or more ways:

- in a configuration file
- in the SASV9_OPTIONS environment variable
- in the SAS command
- in an OPTIONS statement (either in a SAS program or an autoexec file) (An autoexec file contains SAS statements that are executed automatically when SAS is invoked. The autoexec file can be used to specify some SAS system options, as well as to assign librefs and filerefs to data sources that are used frequently.)
- in the System Options window

See “Summary of All SAS System Options in UNIX Environments” on page 350 to view where each SAS system option can be specified.

Any options that do not affect the initialization of SAS, such as CENTER and NOCENTER, can be specified and changed at any time.

Some options can be specified only in a configuration file, in the SASV9_OPTIONS variable, or in the SAS command. These options determine how SAS initializes its interfaces with the operating system and the hardware; they are often called configuration options. After you start a SAS session, these options cannot be changed. Usually, configuration files specify options that you would not change very often. In those cases when you need to change an option just for one job, specify the change in the SAS command.

Overriding the Default Value for a System Option

The default values for SAS system options will be appropriate for many of your SAS programs. However, you can override a default setting using one or more of the following methods:

configuration file

Modify your current configuration file (see "Order of Precedence for Processing SAS Configuration Files" on page 23) or create a new configuration file. Specify SAS system options in the file by preceding each with a hyphen. For ON or OFF options, just list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword identifying the option followed by the option value. All SAS system options can appear in a configuration file.

For example, a configuration file might contain these option specifications:

```
-nocenter

-verbose

-linesize 64
```

SASV9_OPTIONS environment variable

Specify SAS system options in the SASV9_OPTIONS environment variable before you invoke SAS. See "Defining Environment Variables in UNIX Environments" on page 24.

Settings that you specify in the SASV9_OPTIONS environment variable affect SAS sessions that are started when the variable is defined.

For example, in the Korn shell, you would use:

```
export SASV9_OPTIONS='-fullstimer -nodate'
```

SAS command

Specify SAS system options in the SAS command. Precede each option with a hyphen:

```
sas -option1 -option2...
```

For ON or OFF options, list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword that identifies the option, followed by the option value. For example,

```
sas -nodate -work mywork
```

Settings that you specify in the SAS command last for the duration of the SAS session or, for those options that can be changed within the session, until you change them. All options can be specified in the SAS command.

OPTIONS statement within a SAS session

Specify SAS system options in an OPTIONS statement at any point within a SAS session. The options are set for the duration of the SAS session or until you change them. When you specify an option in the OPTIONS statement, do not precede its name with a hyphen (-). If the option has an argument, use = after the option name.

For example,

```
options nodate linesize=72;
options editcmd='/usr/bin/xterm -e vi';
```

For more information about the OPTIONS statement, see “OPTIONS Statement” in *SAS Statements: Reference*. Not all options can be specified in the OPTIONS statement. To find out about a specific option, look up its name in “Summary of All SAS System Options in UNIX Environments” on page 350.

OPTIONS statement in an autoexec file
Specify SAS system options in an OPTIONS statement in an autoexec file. An autoexec file contains SAS statements that are executed automatically when SAS is invoked. The autoexec file can be used to specify some SAS system options, as well as to assign librefs and filerefs to data sources that are used frequently. For example, your autoexec file could contain the following statements:

```
options nodate pagesize=80;
filename rpt '/users/myid/data/report';
```

System Options window
Change the SAS system options from within the System Options window.

In general, use quotation marks to enclose filenames and pathnames specified in the OPTIONS statement or the System Options window. Do not use quotation marks otherwise. Any exceptions are discussed under the individual option. To shorten filenames and pathnames that you specify, you can use the abbreviations listed in Table 2.6 on page 54.

How SAS Processes System Options Set in One Place

If the same option is set more than once within the SAS command, a configuration file, or the SASV9_OPTIONS environment variable, only the last setting is used; the others are ignored. For example, the DMS option is ignored in the following SAS command:

```
sas -dms -nodms
```

The DMS option is also ignored in the following configuration file:

```
-dms
-linesize 80
-nodms
```

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLOC, SASAUTOS, or SASHELP system options more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system options. For more information, see the “APPEND System Option: UNIX” on page 368 and “INSERT System Option: UNIX” on page 392.

How SAS Processes System Options Set in Multiple Places

System Options Set in Multiple Places

When the same option is set in more than one place, the most recent specification is used. The following places are listed in order of precedence. For example, a setting made in the System Options window or OPTIONS statement will override any other setting. However, if you set a system option using the SASV9_OPTIONS environment variable, then this option will override only the setting for the same system option in your configuration file.

Order of Precedence for Processing System Options

The precedence for processing system options is as follows:

1. System Options window or OPTIONS statement (from a SAS session or job).
2. autoexec file that contains an OPTIONS statement (after SAS initializes). (An autoexec file contains SAS statements that are executed automatically when SAS is invoked. The autoexec file can be used to specify some SAS system options, as well as to assign librefs and filerefs to data sources that are used frequently.)
3. SAS command.
4. SASV9_OPTIONS environment variable.
5. configuration files (before SAS initializes). For more information, see "Order of Precedence for Processing SAS Configuration Files" on page 23.

For example, if a configuration file specifies NOSTIMER, you can override the setting in the SAS command by specifying –FULLSTIMER.

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLOC, SASAUTOS, or SASHELP system option more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system options to add the new pathname. For more information, see the "APPEND System Option: UNIX" on page 368 and "INSERT System Option: UNIX" on page 392.

Customizing Your SAS Session by Using Configuration and Autoexec Files

Customizing Your SAS Session

You can customize your SAS environment in several ways. To customize your SAS environment at the point of invocation, you can use configuration and autoexec files. For information about how to customize a SAS session using the windowing environment, see "Overview of Customizing SAS in X Environment" on page 164.

Introduction to Configuration and Autoexec Files

Defining Configuration and Autoexec Files

You can customize your SAS session by defining configuration and autoexec files. You can use these files to specify system options and to execute SAS statements automatically whenever you start a SAS session. SAS system options control many aspects of your SAS session, including output destinations, the efficiency of program execution, and the attributes of SAS files and libraries. For a complete description of SAS system options, see *SAS System Options: Reference*.

For SAS 9.3, the configuration file is typically named sasv9.cfg, and the autoexec file is named autoexec.sas. These files typically reside in the directory where SAS was installed. By default, this directory is the **!SASROOT** directory.

You can have customized configuration and autoexec files in your user home directory. If you do, then SAS will use the customizations specified in these files when you start a SAS session. For more information about the order of precedence SAS uses when processing configuration files, see "Order of Precedence for Processing SAS Configuration Files" on page 23.

SAS system options can be restricted by a UNIX system administrator, so that once they are set by the administrator, they cannot be changed by a user. A system option can be restricted globally, by group, and by user. For more information, see the configuration guide for the UNIX environment on the `Technical Support Web site`, and see "Restricted Options" in Chapter 1 of *SAS System Options: Reference*.

Using the AUTOEXEC System Option

The AUTOEXEC system option specifies the autoexec file. The autoexec file contains SAS statements that are executed automatically when you invoke SAS, or when you start another SAS process. The autoexec file can contain any SAS statements. For example, your autoexec file can contain LIBNAME statements for SAS libraries that you access routinely in SAS sessions.

SAS looks for the AUTOEXEC system option in the following places. It uses the first AUTOEXEC system option that it finds.

- in the command line
- in the SASV9_OPTIONS environment variable
- in the configuration file

If neither the AUTOEXEC nor NOAUTOEXEC system option is found, SAS looks for the autoexec file in three directories in the following order:

1. your current directory
2. your home directory
3. the `!SASROOT` directory (for more information, see "The !SASROOT Directory" on page 443)

SAS uses the first autoexec file that it finds to initialize the SAS session. If you want to see the contents of the autoexec file for your session, use the ECHOAUTO system option when you invoke SAS.

Inserting and Appending Autoexec Files

You can concatenate files in your autoexec file by using the following system options with the AUTOEXEC system option: "INSERT System Option: UNIX" on page 392 and "APPEND System Option: UNIX" on page 368. The autoexec file is always a UNIX file. If your filename contains embedded blanks or special characters, you must enclose the filename in quotation marks. Otherwise, quotation marks are optional when one or more filenames are specified.

You can use the following syntax to concatenate autoexec files:

```
-autoexec "(/path1/autoexec.sas /path2/autoexec.sas /path3/autoexec.sas)"
```

You can use the following syntax with the INSERT system option:

```
-insert autoexec "a.sas" -insert autoexec "b.sas"
```

You can use the following syntax with the APPEND system option:

```
-append autoexec "a.sas" -append autoexec "b.sas"
```

If any file in a concatenated autoexec list does not exist or cannot be opened (for example, if you are not authorized for Read access), SAS issues error messages to the log. SAS terminates without executing any of the files in the list. The final SAS exit code is 103, which indicates system start-up failure.

Differences between Configuration and Autoexec Files

The differences between configuration files and autoexec files are as follows:

- Configuration files can contain only SAS system option settings, while autoexec files can contain any valid SAS statement. For example, you might want to create an autoexec file that includes an OPTIONS statement to change the default values of various system options and LIBNAME and FILENAME statements for the SAS libraries and external files that you use most often.
- Configuration files are processed before SAS initializes, while autoexec files are processed immediately after SAS initializes but before it processes any source statements. An OPTIONS statement in an autoexec file is equivalent to submitting an OPTIONS statement as the first statement of your SAS session.

Creating a Configuration File

To create a configuration file, follow these steps:

1. Use a text editor to write the SAS system options into a UNIX file. Save the file as either sasv9.cfg or .sasv9.cfg. (For more information, see "Order of Precedence for Processing SAS Configuration Files" on page 23.)
2. Specify one or more system options on each line. Use the same syntax that you would use for specifying system options with the SAS command, but do not include the SAS command itself. For example, a configuration file might contain the following lines:

   ```
   -nocenter
   -verbose
   -linesize 64
   -work /users/myid/tmp
   ```

3. Save and close the configuration file.

Order of Precedence for Processing SAS Configuration Files

SAS is shipped with a default configuration file in the **!SASROOT** directory. Your on-site SAS personnel can edit this configuration file so that it contains whichever options are appropriate to your site.

You can also create one or more of your own configuration files. SAS reads option settings from each of these files in the following order:

1. sasv9.cfg in the **!SASROOT** directory. (See "Contents of the !SASROOT Directory" on page 443.)
2. sasv9_local.cfg in the **!SASROOT** directory. (See "Contents of the !SASROOT Directory" on page 443.)
3. .sasv9.cfg in your home directory. (Notice the leading period.)
4. sasv9.cfg in your home directory.
5. sasv9.cfg in your current directory.
6. any restricted configuration files. Restricted configuration files contain system options that are set by the site administrator and cannot be changed by the user. Options can be restricted globally, by group, or by user. For more information about restricted configuration files, see the configuration guide for the UNIX environment.

For future releases of SAS, the names of these files will change accordingly.

For each system option, SAS uses the last setting that it encounters. Any other settings are ignored. For example, if the WORKPERMS system option is specified in sasv9.cfg in the **!SASROOT** directory and in sasv9.cfg in your current directory, SAS uses the value specified in sasv9.cfg in your current directory.

Specifying a Configuration File for SAS to Use

When you specify a configuration file for SAS to use, you bypass the search of the configuration files listed in "Order of Precedence for Processing SAS Configuration Files" on page 23.

Note: SAS still processes any restricted configuration files that exist. The settings in these files take precedence over the settings in the configuration file that you specify.

If you set both SASV9_OPTIONS and SASV9_CONFIG, SAS always uses SASV9_OPTIONS. SASV9_CONFIG is used only if you do not use –config in the command line.

To specify a configuration file, complete one of the following steps:

- specify a configuration file with the CONFIG system option in the SAS command:

  ```
  sas -config filename
  ```

- specify a configuration file in the SASV9_OPTIONS environment variable. See "Defining Environment Variables in UNIX Environments" on page 24. For example, in the Korn shell, you would use:

  ```
  export SASV9_OPTIONS='-config filename'
  ```

- define the environment variable SASV9_CONFIG. See "Defining Environment Variables in UNIX Environments" on page 24. For example, in the Korn shell, you would use:

  ```
  export SASV9_CONFIG=filename
  ```

filename is the name of a file that contains SAS system options.

If you have specified a configuration file in the SASV9_OPTIONS or SASV9_CONFIG environment variables, you can prevent SAS from using that file by specifying NOCONFIG in the SAS command.

If SAS cannot find SASV9_OPTIONS, the following message is written to the SAS log:

```
ERROR: Cannot open [/fullpath/filename]: No such
       file or directory.
```

Defining Environment Variables in UNIX Environments

What Is a UNIX Environment Variable?

UNIX environment variables are variables that apply to both the current shell and to any subshells that it creates (for example, when you send a job to the background or execute a script). If you change the value of an environment variable, the change is passed forward to subsequent shells but not backward to the parent shell.

In a SAS session, you can use the SASV9_OPTIONS environment variable to specify system options and the SASV9_CONFIG environment variable to specify a configuration file. You can also use environment variables as filerefs and librefs in various statements and commands. Filerefs and librefs consist of uppercase letters, digits, and the underscore character in environment variable names. Other characters are not recognized by SAS.

Note: A SAS/ACCESS product initializes the environment variables that it needs when loading. Any changes that you make to an environment variable after initialization will not be recognized. For more information, see the documentation for your SAS/ACCESS product.

How to Define an Environment Variable for Your Shell

Defining Environmental Variables

The way in which you define an environment variable depends on the shell that you are running. (To determine which shell you are running, type **`ps`** at the command prompt or **`echo $SHELL`** to see the current value of the SHELL environment variable.)

Bourne and Korn Shells

In the Bourne shell and in the Korn shell, use the **`export`** command to export one or more variables to the environment. For example, these commands make the value of the variable **`scname`** available to all subsequent shell scripts:

```
$ scname=phonelist
$ export scname
```

In the Korn shell, you can combine these commands into one command:

```
$ export scname=phonelist
```

If you change the value of **`scname`**, then the new value affects both the shell variable and the environment variable. If you do not export a variable, only the shell script in which you define has access to its value.

C Shell

In the C shell (csh and tcsh), you set (define and export) environment variables with the **`setenv`** (set environment) command. For example, this command is equivalent to the commands shown previously:

```
% setenv scname phonelist
```

Displaying the Value of an Environment Variable

To display the values of individual environment variables, use the **`echo`** command and parameter substitution. An example is: **`echo $SHELL`** which returns the current value of the SHELL environment variable. Use the **`env`** (or **`printenv`**) command to display all environment variables and their current values.

Specifying an Encoding for Pathnames on Disk

SAS uses the default session encoding when referencing external files and directories. The PATHENCODING environment variable provides an alternative encoding for external file and directory references. PATHENCODING is valid only for files that are located on disk. When the PATHENCODING environment variable has a valid encoding value, SAS transcodes the pathname in the specified encoding. For a list of valid encoding values on UNIX, see “UNIX Encoding Values” in Chapter 21 of *SAS National Language Support (NLS): Reference Guide*.

Determining the Completion Status of a SAS Job in UNIX Environments

The exit status for the completion of a SAS job is returned in `$STATUS` for the C shell, and in `$?` for the Bourne and Korn shells. A value of 0 indicates normal termination. You can affect the exit status code by using the ABORT statement. The ABORT statement takes an optional integer argument, *n*, which can range from 0 to 255.

Note: Return codes of 0–6 and return codes greater than 977 are reserved for use by SAS.

The following table summarizes the values of the exit status code.

Table 1.1 *Exit Status Code Values*

Condition	Exit Status Code
All steps terminated normally	0
SAS System issued warnings	1
SAS System issued errors	2
User issued ABORT statement	3
User issued ABORT RETURN statement	4
User issued ABORT ABEND statement	5
SAS could not initialize because of a severe error	6
User issued ABORT RETURN - n statement	n
User issued ABORT ABEND - n statement	n

If you specify the ERRORABEND SAS system option on the command line, and the job has errors, the exit status code is set to 5.

UNIX exit status codes are in the range 0-255. Numbers greater than 255 might not print what you expect because the code is interpreted as a signed byte.

Exiting or Interrupting Your SAS Session in UNIX Environments

Methods for Exiting SAS

Use one of the following methods to exit a SAS session:

- Select **File** ⇨ **Exit** if you are using SAS in the windowing environment.
- Use **`endsas;`**.
- Enter **`BYE`** in the ToolBox if you are using SAS in the windowing environment.
- Use CTRL+D if this control key sequence is your EOF command and if you are using SAS in interactive line mode.

Methods for Interrupting or Terminating SAS

Interrupting or Terminating SAS

In addition to the methods for exiting SAS, SAS provides methods for interrupting or terminating a SAS session. SAS does not recommend that you use these methods until you have tried to exit SAS by one of the methods listed in "Methods for Exiting SAS" on page 27.

You can interrupt or terminate SAS in the following ways:

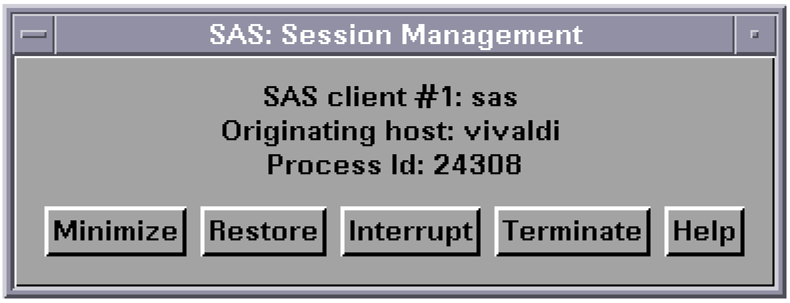
- Press the interrupt or quit control key. Interrupt will display a dialog box, while quit will force a shutdown. Using the quit control key is not recommended.
- Use the SAS: Session Management window.
- Enter the UNIX **`kill`** command. Use this command when all other methods of exiting SAS have failed. By default, the kill command is **`kill -15`** (SIGTERM).

  Using the UNIX **`kill —9`** command on a SAS process that is running might corrupt data sets that are open for Write or Update access.

Interrupting a SAS Process

The method that you use to interrupt a SAS process depends on how you invoke SAS.

- If you are running SAS in interactive line mode or in batch mode in the foreground, then you can use either of the following methods to interrupt SAS:
  - Press the control key sequence that is set to interrupt in the shell that invoked SAS. In most cases, this control key sequence is CTRL+C. See the man page for the **`stty`** command to determine the appropriate control key sequence for your environment.
  - Use the **`-SIGINT`** option in the **`kill`** command. For more information, see "Using the UNIX kill Command" on page 30.
- If you are running the SAS windowing environment in the foreground, then click **Interrupt** in the SAS: Session Management window.

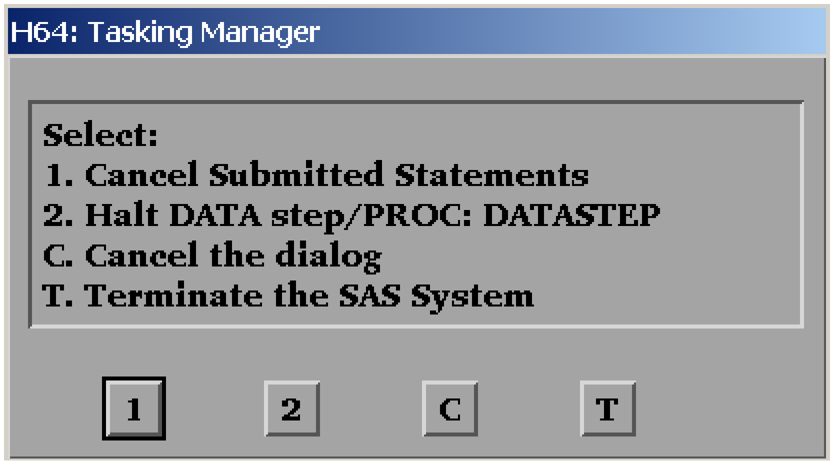


Note: You can access the SAS Session Manager by invoking SAS with the -DMS or -DMSEXP option. Select **SAS: Session Management** from the menu.

- If you are running SAS in batch mode, you must use control keys to interrupt the SAS process. A SAS: Session Management window is not available.

The interrupt signal sends a request to the supervisor to handle an interrupt. The interrupt signal is not handled until a safe point in the code is reached. A safe point is one that allows the interrupt handler to be run safely. The supervisor responds as soon as possible with a prompt or window that requests what type of interrupt action you want to take. During this time, normal processing of a DATA step or PROC step is suspended.

For example, when you interrupt a DATA step or PROC step, a Tasking Manager window, similar to the following, appears:



Select:
1. Cancel Submitted Statements
2. Halt DATA step/PROC: DATASTEP
C. Cancel the dialog
T. Terminate the SAS System

The following table explains each of the options in the window:

Table 1.2 *Options in the Tasking Manager Window*

Option	Description	What This Option Does
1	Cancel Submitted Statements	Selecting this option terminates the current DATA step or PROC step. Outstanding source code that is waiting to execute is flushed from the system. In interactive line mode, you return to the command prompt.

Option	Description	What This Option Does
2	Halt DATA step/ PROC: DATASTEP	Selecting this option sends an interrupt signal to the DATA step or PROC step. The default behavior is for the DATA step or PROC step to terminate, and to execute the next statement. A procedure might specify its own handler to process the interrupt. In this case, the procedure might request more input from you. For example, SAS webAF has a different interrupt menu than PROC SQL. *Note:* If you are using a relational database, the interrupt signal might be handled differently, depending on which relational database you are using.
C	Cancel the dialog	Selecting this option cancels the interrupt, and you return to normal processing.
T	Terminate the SAS System	Selecting this option causes the DATA step or PROC step to be terminated. Outstanding source code that is waiting to execute is flushed from the system. SAS exits as cleanly as possible.

Terminating a SAS Process

If you are running the SAS windowing environment in the foreground, click **Terminate** in the SAS: Session Management window. If you are running a SAS process in interactive line mode in the background, use control keys to terminate the SAS process, or use the **`kill`** command.

If you click **Terminate** in the SAS: Session Management window, a dialog box appears, confirming that you want to end the session. If you click **OK**, then the SAS session and any queries that are currently running are terminated. If you click **Cancel**, you are returned to the SAS session.

Using Control Keys

Control keys enable you to interrupt or terminate your session by pressing the interrupt or quit key sequence. However, control keys can be used only when your SAS program is running in interactive line mode or in batch mode in the foreground. You cannot use control keys to stop a background job.

Note: You cannot use control keys to stop a batch job that has been submitted with the **`batch`**, **`at`**, **`nohup`**, or **`cron`** command.

Because control keys vary from system to system, issue the UNIX **`stty`** command to determine which key sends which signal. The **`stty`** command varies considerably among UNIX operating environments, so check the UNIX man page for **`stty`** before using the command. Usually, one of these forms of the command will print all of the current terminal settings:

```
stty
stty -a
stty everything
```

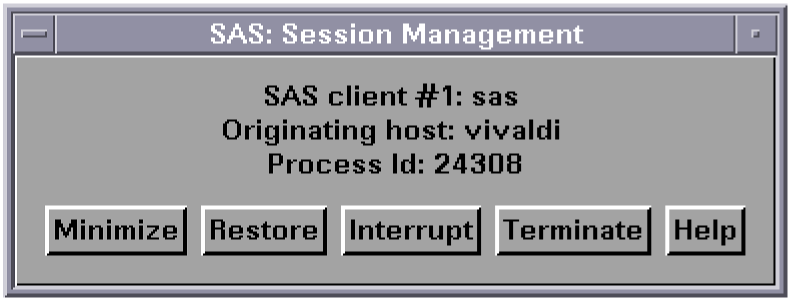
The output should contain lines similar to these:

```
intr = ^C; quit = ^\; erase = ^H;
kill = ^U; eof = ^D; eol = ^@
```

The caret (^) represents the CTRL key. In this example, CTRL+C is the interrupt key and CTRL+\ is the quit key. Quit is a more forceful termination and might result in data corruption. If you use –SIGTERM, SAS attempts to shut down the system correctly.

Using the SAS Session Manager

If you invoke SAS in the windowing environment, you can use the SAS session manager to interrupt or terminate your SAS session. The SAS session manager is automatically minimized when you start SAS. To interrupt or terminate your SAS session, open the SAS: Session Management window and click **Interrupt** or **Terminate**:


SAS: Session Management

SAS client #1: sas
Originating host: vivaldi
Process Id: 24308

If asynchronous SAS/CONNECT tasks are running when you terminate a SAS session, these tasks are terminated and no warning message is displayed. Generally, it is better to exit from the file menu or tool box.

Note: Clicking **Interrupt** is equivalent to specifying the **`-SIGINT`** option on the **`kill`** command. Clicking **Terminate** is equivalent to specifying the **`-SIGTERM`** option on the **`kill`** command.

For more information about the SAS Session Manager, see "The SAS Session Manager (motifxsassm) in UNIX" on page 143.

Using the UNIX kill Command

Note: Use the **`kill`** command only after you have tried all other methods to exit your SAS session.

The **`kill`** command sends an interrupt or terminate signal to SAS, depending on which signal you specify. You can use the **`kill`** command to interrupt or terminate a SAS session running in any mode. The **`kill`** command cannot be issued from within a SAS session. You must issue it from another terminal or from another window (if your terminal permits it).

The format of the **`kill`** command is:

kill <*-signal-name*> *pid*

To send the interrupt signal, specify **`-SIGINT`**. To send the terminate signal, specify **`-SIGTERM`**. Use the **`ps`** command and its options to determine the process identification number (pid) of the SAS session that you want to interrupt or terminate.

The results of using the **`ps`** command differ in different operating environments. See the UNIX man page for your operating environment for specific information about the **`ps`** command and its options. Adding options helps to determine which process you want to kill if you have more than one SAS process running. Also, servers (metadata, OLAP, and so on) leave a process identification number in their start-up directories. You can use this number with the **`kill`** command.

The following table lists some of the important **`kill`** signals.

Table 1.3 *Description of Important kill Signals*

Signal	Option	Description
0	SIGNULL	Checks access to process identifier.
1	SIGHUP	Causes SAS to terminate.
2	SIGINT	Causes SAS to interrupt the session. SIGINT is very similar to SIGQUIT.
3	SIGQUIT	Causes a more forceful shutdown than SIGTERM. It does not cause a core dump.
9	SIGKILL	Brings down SAS. Use this option only after all attempts to exit SAS have failed. Using SIGKILL can cause data corruption.
15	SIGTERM	Causes SAS to terminate.

For more information, see the UNIX man pages for the `ps` and `kill` commands.

Messages in the Console Log (STDOUT)

If SAS encounters an error or warning condition when the SAS log is not available, then any messages that SAS issues are written to the console log. Normally, the SAS log is unavailable only early in SAS initialization and late in SAS termination.

If you are using the -STDIO option, the log is displayed in stderr, and the listing is displayed in stdout.

Ending a Process That Is Running as a SAS Server

If you need to end a process running as a SAS server, use one of the following methods:

- If you are using the SAS Metadata Server, use the SAS Management Console to end a process.
- If you are using another SAS server, use the UNIX scripts that shipped with the servers to stop the process. You can also use these scripts to start (or restart) a server, as well as determine whether the server is already running. For more information about these scripts, contact your site administrator.

  Note: If the server does not respond to the UNIX script, then you can use the `kill` command to end the server process. For more information, see "Using the UNIX kill Command" on page 30.

Interrupting a SAS Process and the Underlying DBMS Process

CAUTION:

Interrupting a SAS process and the underlying DBMS process might kill all jobs that are running on your DBMS. Interrupting your SAS and DBMS processes should be an exception. Use care when you construct your queries. For example, if SAS sends SQL to an RDBMS, there is no way to interrupt the SQL statements because SAS no longer has control of them. The statements are running in the RDBMS.

When you interrupt a SAS process, you might terminate the current query. If you are using the current query to create a new data set, then the data set is still created even if the query is terminated. If you are using the current query to overwrite a data set, the data set is not overwritten if the query is terminated. In most cases you do not receive a warning that the query did not complete.

Note: In this section, SAS process refers to a series of events. It is not the process on the operating system. When you interrupt or terminate a SAS process, the process on the operating system might still be running.

In many cases (such as using Oracle in UNIX environments), when you interrupt or terminate a query on a server, the following processes stop:

- processing of current extractions. For example, if you forgot to include a WHERE clause in your SQL query and are now extracting 1 billion rows into SAS, issuing an interrupt stops the SAS process and the extract step in the DBMS.
- processing of queries that are in progress on the server. For example, you might have a very complex extract query that runs for a long time before producing a result. Issuing an interrupt stops the SAS and DBMS processes. As a result, the complex query running on your DBMS server is interrupted and terminated.
- update, delete, and insert processing. For example, you are updating, deleting, or inserting many rows in your DBMS. An interrupt stops the SAS and DBMS processes.

Chapter 2

Using SAS Files

Introduction to SAS Files, Libraries, and Engines in UNIX Environments

SAS Files

What Is a SAS File?

Your data can reside in different types of files, including SAS files and files that are formatted by other software products, such as database management systems. Under UNIX, a SAS file is a specially structured UNIX file. Although the UNIX operating environment manages the file for SAS by storing it, the operating system cannot process it because of the structure built into the file by SAS. For example, you can list the filename with the **`ls`** command, but you cannot use the **`vi`** editor to edit the file. A SAS file can be permanent or temporary.

Case Sensitivity in Data Set Names

In UNIX operating environments, SAS data set names are written in all lowercase characters. Because of this requirement, SAS reads only data set names that are written in all lowercase characters.

If you use the UNIX utilities **`mv`** or **`cp`** to rename SAS data set names with uppercase or mixed-case characters, SAS can no longer read the data set names.

UNIX is case sensitive. Therefore, a data set named *xxx*.sas7bdat is not the same as a data set named *XXX*.sas7bdat. In fact, both of these data sets can reside in the same directory as completely different data sets.

SAS Libraries

What Are SAS Libraries?

SAS files are stored in SAS libraries. A SAS library is a collection of SAS files within a UNIX directory. Any UNIX directory can be used as a SAS library. (The directory can also contain files called external files that are not managed by SAS. See “Using External Files and Devices” on page 70 for how to access external files.) SAS stores temporary SAS files in a Work library, which is automatically defined for you. You must specify a library for each permanent SAS file. For more information, see “Work Library” on page 60.

What Is a Libref?

SAS libraries can be identified with librefs. A libref is a name by which you reference the directory in your application. For more information about how to assign a libref, see “Referring to SAS Files by Using Librefs in UNIX Environments” on page 50.

Engines

What Is an Engine?

SAS files and SAS libraries are accessed through engines. An engine is a set of routines that SAS must use to access the files in the library. SAS can read from and, in some cases, write to the file by using the engine that is appropriate for that file type. For some file types, you need to tell SAS which engine to use. For others, SAS automatically chooses the appropriate engine. The engine that is used to create a SAS data set determines the format of the file.

Additional Resources

For more information about SAS files, libraries, and engines, see *SAS Language Reference: Concepts*.

Common Types of SAS Files in UNIX Environments

SAS Data Sets

What Are SAS Data Sets?

SAS data sets consist of two types:

- "SAS Data Files (Member Type DATA)" on page 36
- "SAS Views (Member Type VIEW)" on page 37

Descriptor Information and Data Values

Data sets consist of descriptor information and data values organized as a table of rows and columns that can be processed by one of the engines. The descriptor information includes data set type, data set label, the names and labels of the columns in the data set, and so on. A SAS data set can also include indexes for one or more columns.

SAS data sets are implemented in two forms:

- If the data values and the data set's descriptor information are stored in one file, the SAS data set is called a SAS data file.
- If the file contains information about where to obtain a data set's data values and descriptor information, the SAS data set is called a SAS view.

The default engine processes the data set as if the data file or data view and the indexes were a single entity.

For more information, see "SAS Data Files (Member Type DATA)" on page 36 and "SAS Views (Member Type VIEW)" on page 37.

SAS Data Files (Member Type DATA)

The SAS data file is probably the most frequently used type of SAS file. These files have the extension **`.sas7bdat`**. SAS data files are created in the DATA step and by some SAS procedures. There are two types of data files:

- Native data files store data values and their descriptor information in files formatted by SAS. These data files are the traditional SAS data sets from previous releases of SAS.

  Native SAS data files created by the default engine can be indexed. An index is an auxiliary file created in addition to the data file it indexes. The index provides fast access to observations within a SAS data file by a variable or key. Under UNIX, indexes are stored as separate files, but are treated as integral parts of the SAS data file by SAS.

CAUTION:

Do not remove index files using UNIX commands. Removing the index file can damage your SAS data set. Also, do not change its name or move it to a different directory. Use the DATASETS procedure to manage indexes.

- Interface data files store data in files that have been formatted by other software and that SAS can read only. For more information, see "Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments" on page 63.

SAS Views (Member Type VIEW)

A SAS view contains only the information needed to derive the data values and the descriptor information. Depending on how the SAS view is created, the actual data can be located in other SAS data sets or in other vendors' files.

Views can be of two kinds:

- Native SAS views contain information about data in one or more SAS data files or SAS views. This type of view is created with the SQL procedure or DATA step.
- Interface SAS views contain information about data formatted by other software products such as a database management system. For example, the ACCESS procedure in SAS/ACCESS software creates an interface SAS view.

SAS Catalogs

Catalogs are a special type of SAS file that can contain multiple entries. Many different types of entries can be kept in the same SAS catalog. For example, catalogs can contain entries created by SAS/AF and SAS/FSP software, windowing applications, key definitions, SAS/GRAPH graphs, and so on.

Catalogs have the SAS member type of CATALOG.

Stored Program Files

Stored program files are compiled DATA steps. For information about the stored program files, see *SAS Language Reference: Concepts*.

Stored program files have the SAS member type of PROGRAM.

Access Descriptor Files

Access descriptor files describe the data formatted by other software products such as the Oracle or the Sybase database management systems. Descriptor files created by the ACCESS procedure in SAS/ACCESS software have the SAS member type of ACCESS.

Filename Extensions and Member Types in UNIX Environments

Because SAS needs to distinguish between the different file types, it automatically assigns an extension to each file when it creates the file. Also, because each SAS file is a member of a library, SAS assigns each file a member type.

The following table lists the file extensions and their corresponding SAS member types.

CAUTION:

Do not change the file extensions of SAS files. File extensions determine how SAS accesses files; changing them can cause unpredictable results.

Table 2.1 *File Extensions for SAS File Types*

Version 6		Version 8, SAS 9			
Random Access Files	**Sequential Access Files**	**Random Access Files**	**Sequential Access Files**	**SAS Member Type**	**Description**
.sas	.sas	.sas	.sas	.sas	SAS program
.lst	.lst	.lst	.lst	.lst	Procedure output
.log	.log	.log	.log	.log	SAS log file
.ssdnn (all Version 6 files end with a two-character code (nn) that identifies sets of compatible SAS files) See "Sharing SAS Files in a UNIX Environment" on page 40 for more information.	.sdqnn	.sas7bdat	.sas7sdat	DATA	SAS data file
.snxnn	.siqnn	.sas7bndx	.sas7sndx	INDEX	Data file index; not treated by the SAS system as a separate file
.sctnn	.scqnn	.sas7bcat	.sas7scat	CATALOG	SAS catalog
.sspnn	.ssqnn	.sas7bpgm	.sas7spgm	PROGRAM	Stored program (DATA step)
.ssvnn	.svqnn	.sas7bvew	.sas7svew	VIEW	SAS view
.ssann	.saqnn	.sas7bacs	.sas7sacs	ACCESS	Access descriptor file
.sstnn	.stqnn	.sas7baud	.sas7saud	AUDIT	Audit file
.sfdnn	.sfqnn	.sas7bfdb	.sas7sfdb	FDB	Consolidation database
.ssmnn	.smqnn	.sas7bmdb	.sas7smdb	MDDB	Multidimensional database
.sdsnn	.soqnn	.sas7bods	.sas7sods	SASODS	Output delivery system file
.snmnn	.sqnnn	.sas7bdmd	.sas7sdmd	DMDB	Data mining database

Version 6		Version 8, SAS 9			
Random Access Files	**Sequential Access Files**	**Random Access Files**	**Sequential Access Files**	**SAS Member Type**	**Description**
.sitnn	.srqnn	.sas7bitm	.sas7sitm	ITEMSTOR	Item store file
.sutnn	.suqnn	.sas7butl	.sas7sutl	UTILITY	Utility file
.spunn	.spqnn	.sas7bput	.sas7sput	PUTILITY	Permanent utility file
.ssbnn	.sbqnn	.sas7bbak	.sas7sbak	BACKUP	Backup file

A UNIX directory can store a variety of files, but you might find it more practical to store files in separate directories according to their use. Also, you can keep libraries that are accessed by different engines in the same directory, but this is not recommended. For more information, see "Using Multiple Engines for a Library in UNIX Environments" on page 56.

Using Direct I/O

Introduction to Direct I/O

Direct I/O is a method for processing input and output files and is used in file handling. Direct I/O enables SAS to read files from and write files directly to storage devices without first going through the UNIX operating environment's read and write caches. You can use direct I/O for SAS files. Using direct I/O might improve system performance, depending on the number and types of jobs that you are running.

SAS uses three related options that affect direct I/O:

- ENABLEDIRECTIO statement option
- USEDIRECTIO= statement option
- USEDIRECTIO= data set option

The ENABLEDIRECTIO option in the LIBNAME statement makes direct I/O processing available for data sets that are listed in the DATA statement. The libref that points to the data sets must have been defined in a LIBNAME statement that uses the ENABLEDIRECTIO option. Using ENABLEDIRECTIO itself does not turn on direct I/O.

A libref that is assigned to a directory with the ENABLEDIRECTIO option will not match another libref that is assigned to the same directory without the ENABLEDIRECTIO option. The two librefs will point to the same directory, but the files that are opened using the libref with ENABLEDIRECTIO can be read from and written to using direct I/O. Files that are opened using the other libref will be read from and written to using the regular disk I/O calls.

The USEDIRECTIO= data set option in the DATA statement or the USEDIRECTIO= statement option in the LIBNAME statement turns on direct I/O for data sets in which the ENABLEDIRECTIO statement option has been applied. Using USEDIRECTIO=

without first applying the ENABLEDIRECTIO option has no effect on direct I/O in a data set.

Turning On Direct I/O

You can turn on direct I/O in two ways:

- Use both the ENABLEDIRECTIO and USEDIRECTIO= options in the LIBNAME statement.

  This method opens for direct I/O all of the files that are referenced by the libref in the LIBNAME statement.

- Use the ENABLEDIRECTIO option in the LIBNAME statement to render direct I/O available, and use the USEDIRECTIO= data set option in a DATA statement to turn on direct I/O functionality.

  This method opens for direct I/O only the data set where the option is used. The data set must be referenced by the libref in the LIBNAME statement.

For more information about these options and how they are used, see:

- ENABLEDIRECTIO in “Engine and Host Options” on page 337
- USEDIRECTIO= in “Engine and Host Options” on page 337
- “USEDIRECTIO= Data Set Option: UNIX” on page 250

Holding a File in Memory: The SASFILE Statement

You can use the SASFILE statement to open a SAS data set. SAS attempts to allocate enough buffers to hold the entire data set in memory. If enough memory is available, then the entire data set is kept in memory until the data set is closed. If enough memory is not available, then SAS allocates as many buffers as it can. If your file is very large or if SAS is already using a large amount of memory, then using the SASFILE statement will not help.

When the SASFILE statement executes the first time, SAS opens the file. Subsequent DATA and PROC steps use the file without having to open it again because the file remains in memory. The file remains open until a second SASFILE statement closes it, or until the program or session ends. For more information, see “SASFILE Statement” in *SAS Statements: Reference*.

Sharing SAS Files in a UNIX Environment

Sharing SAS Files

If more than one SAS process has Write access to a SAS file (a data set, catalog, library, and so on) at the same time, you would get unpredictable results if the file was updated. SAS locks the file to prevent more than one user from having Write access to a file. When one SAS process opens a file with Write access, other processes are blocked from Write access until the first process closes the file. SAS provides statement and system options to override this file protection. However, in almost all cases, you should leave file protection turned on.

Options to Use for File Locking: SAS Files

You can turn off file locking for SAS files in the following ways:

- Use the FILELOCKS option in the LIBNAME statement.
- Use the FILELOCKS system option.

File Locking for SAS Files: The FILELOCKS Statement Option

By default, SAS restricts Write access to one user. The FILELOCKS option in the LIBNAME statement overrides the default and allows multiple users to have Write access to a file. SAS files that are opened under the libref in the LIBNAME statement are the files that are locked. Multiple users have Read access to files.

The FILELOCKS statement option applies to most (but not all) of the SAS I/O files (for example, data sets and catalogs) that are opened under the libref in the LIBNAME statement.

For more information, see "LIBNAME Statement: UNIX" on page 333.

File Locking for SAS Files: The FILELOCKS System Option

By default, SAS restricts Write access to one user. The FILELOCKS system option overrides this default for both SAS files and external files and allows multiple users to have Write access to a file. The FILELOCKS system option enables you to apply a behavior globally to individual files that are opened.

You can use the FILELOCKS system option at start-up, in the OPTIONS statement, or in the command line. You can specify multiple instances of the FILELOCKS system option. Each instance is added to an internal table of paths and settings. The FILELOCKS system option applies to most (but not all) of the SAS I/O files (for example, data sets and catalogs) that are opened under the libref in the LIBNAME statement. For more information, see "FILELOCKS System Option: UNIX" on page 381 and "LIBNAME Statement: UNIX" on page 333.

Waiting to Use a Locked File

If you want to use a SAS file that is locked by another process, you can use the FILELOCKWAIT option in the LIBNAME statement to specify how long SAS will wait for the locked file to become available to another process. The FILELOCKWAIT statement option affects only those files that are opened under the libref in a LIBNAME statement. For more information, see "LIBNAME Statement: UNIX" on page 333.

Conditions to Check When FILELOCKS=NONE

When file locking is turned off (that is, when the FILELOCKS system option is set to NONE), SAS attempts to open a file without checking for an existing lock on the file. These files are not protected from shared Update access.

CAUTION:

SAS recommends that you do not use the FILELOCKS=NONE option. If multiple users open the same file for Write access, then the file might become corrupted. The

FILELOCKS=NONE option is used primarily to determine whether a job failed because of a locked file.

If the FILELOCKS system option is set to NONE, then you should perform one of the following tasks:

- Make sure that your **sasuser** directory is unique for each SAS session. Typically, the system administrator assigns this directory in the system configuration file. The specification in that file or in your personal configuration file helps ensure that the directory is unique as long as you run only one SAS session at a time.

  If you run two or more SAS sessions simultaneously, you can guarantee unique user files by specifying different **sasuser** directories for each session. In the first session, you can use:

  ```
  -sasuser ~/sasuser
  ```

  In the *n* the session, you can use:

  ```
  -sasuser ~/sasusern
  ```

  For more information, see "Order of Precedence for Processing SAS Configuration Files" on page 23 and "RSASUSER System Option: UNIX" on page 412. The RSASUSER option can be used to control modifications to the Sasuser library when it is shared by several users (see "RSASUSER System Option: UNIX" on page 412.)

- If you run two or more SAS sessions simultaneously (either by using the X statement or by invoking SAS from two different windows) and you use the same Sasuser.Profile catalog, do not perform any actions (such as using the WSAVE command or changing key assignments) within the SAS session to change the Sasuser.Profile catalog because both sessions can use the same catalog.

- Multiple users can read the same data sets at the same time. However, only one user at a time should write to or update a data set so that data is not overwritten or corrupted.

When FILELOCKS=CONTINUE

By default, SAS restricts Write access to one user. When you use the FILELOCKS=CONTINUE option, SAS fails to open a file if that file is locked by another user, and writes an error message to the log. However, if SAS returns a message that identifies some other error, then SAS disregards the lock on the file, opens the file, and continues to execute the job.

Sharing Files over a Network

Introduction to Sharing Files on Multiple Workstations

SAS can be licensed to run on one or more workstations in a network of similar computers. The license specifically lists the workstations on which SAS can run. Unlicensed workstations in the network might have access to the SAS executable files, but they might not be able to run SAS.

If the licensed workstations are connected through NFS mounts so that they share a file system, they can all share a single copy of the SAS executable files, although sharing of a single copy is not necessary. They can also share SAS files. However, if a SAS session opens a data set or catalog for update, it must obtain an exclusive file lock on that file to prevent other SAS sessions from accessing that file. Other SAS sessions will be blocked from access as long as the file is open.

If SAS is installed on different types of workstations that are connected through NFS, then each type of workstation must have its own copy of the SAS executable files. For information about how to move catalogs and data sets between hosts, see "Compatible Computer Types in UNIX Environments" on page 44.

Accessing Files on Different Networks

You can access a file on a different type of workstation if the two computers are connected to the same file system. You can access external files that were created under a different operating environment.

If you create a data set or catalog and save it to a directory, and later want to access the file from another computer on a different network, you have several alternatives for working with that file:

- You can log on to a computer remotely and perform the work there if you rarely use the file.
- You can log on to a computer remotely and perform the work there if the file changes often. This alternative ensures that you are accessing the most current version of the file. If you use PROC CPORT to copy the file to your computer, the original file might have changed between the time it was copied and the time it was read.
- You can log on to a computer remotely rather than transfer the file to your computer if you want to use the file once. It might not be efficient to use PROC CPORT, or you might not have enough disk space for PROC CPORT to run locally.
- You can use the File Transfer Protocol (FTP) or the Routing Control Processor (RCP) to transfer the file from the remote computer to your computer.
- You can do part of your work on your computer and part of your work on a remote computer. One example of this alternative is to run a set of statements on a small test case on the local computer, and then submit the real work to be done on the remote computer. Similarly, you can subset a large data set on another computer and then do local analysis on that subset. You can accomplish this task by using SAS/CONNECT software. For more information about Remote Library Services, see *SAS/CONNECT User's Guide*.

Troubleshooting: Accessing Data over NFS Mounts

SAS might hang when accessing data over NFS mounts if the FILELOCKS option is set to FAIL or CONTINUE. To alleviate the problem, ensure that all NFS file locking daemons are running on both computers (usually statd and lockd). Your UNIX system administrator can assist with starting statd and lockd.

Note: To test whether there is a problem with file locking, you can set the FILELOCKS system option to NONE temporarily. If setting FILELOCKS to NONE resolves the problem, then you know that there probably is a problem with the statd and lockd daemons. It is recommended that you do not set FILELOCKS to NONE permanently because it might cause data corruption or unpredictable results.

Compatible Computer Types in UNIX Environments

Characteristics of Compatible Computer Types

Big endian, little endian, and bi-endian refer to the way an operating system orders integer values. These values are ordered in bytes, and the way in which bytes are ordered is called byte ordering. The ordering of bytes depends on the operating environment on which the integers were produced.

On big endian platforms, the value 1 is stored in binary and is represented here in hexadecimal notation. One byte is stored as 01, two bytes as 00 01, and four bytes as 00 00 00 01. On little endian platforms, the value 1 is stored in one byte as 01 (the same as big endian), in two bytes as 01 00, and in four bytes as 01 00 00 00.

If an integer is negative, the "two's complement" representation is used. The high-order bit of the most significant byte of the integer will be set to on. For example, -2 would be represented in one, two, and four bytes on big endian platforms as FE, FF FE, and FF FF FF FE, respectively. On little endian platforms, the representation would be FE, FE FF, and FE FF FF FF. These representations result from the output of the integer binary value -2 expressed in hexadecimal notation.

Different computers store numeric binary data in different forms. Hewlett-Packard and IBM store data in big-endian format. Linux and Tru64 UNIX store data in little-endian format. Solaris SPARC stores data in big-endian format, and Solaris x64 stores data in little-endian format.

SAS files can be transported between compatible computer types using various methods including NFS, FTP, and CD. For two computer types to be compatible, they must have the following characteristics in common:

- Same word length. Word lengths can be either 32-bit or 64-bit.
- Same ordering of bytes in memory. Computer types differ in whether the most significant byte (MSB) or the least significant byte (LSB) is loaded at the lower memory address. This byte order is often referred to as “big endian” or “little endian.”

Compatible Computer Types for Release 6.12 through SAS 9.3

The tables in this section show the compatible computer types for Release 6.12 through SAS 9.3. After each table, a brief explanation is provided.

Table 2.2 *Compatible Computer Types for Sharing Release 6.12 Files*

Bits	Compatible Computer Types
32	Intel ABI, Linux, HP-UX (version 6), Solaris, AIX, IRIX
64	Tru64 UNIX, HP-UX on PA-RISC (release 9), HP-UX on Itanium (release 9)

You can move a Release 6.12 SAS data set that was created on a 32-bit HP-UX host to a 32-bit AIX host using NFS, FTP, or CD. Because HP-UX and AIX are compatible computer types, you can use the V6 engine to read the HP-UX data set on the AIX host.

The same 32-bit HP-UX data set can be moved to a 32-bit Intel ABI host. However, because these computer types are incompatible, you cannot use the V6 engine to read the HP-UX data set.

You can use the V9 engine (SAS 9.3) to read V6 data sets, but you cannot write to them.

For information about reading Release 6 data sets, see "Reading Version 6 Files" on page 48.

Table 2.3 *Compatible Computer Types for Sharing Release 7 through SAS 9.3 Files*

Bits	Platform	Compatible Computer Types
32	big endian	HP-UX for HP 9000 servers, Solaris for SPARC, AIX, and IRIX
32	little endian	Intel ABI, Linux
64	big endian	HP-UX for HP 9000 servers, Solaris for SPARC, AIX, and HP-UX for HP Integrity Servers
64	little endian	Linux Itanium, Solaris for x64, Linux x64

Note: In Release 8.2, both 32–bit and 64–bit SAS were available for the AIX, HP-UX, and Solaris operating environments. In SAS 9, SAS is 64–bit for these environments.

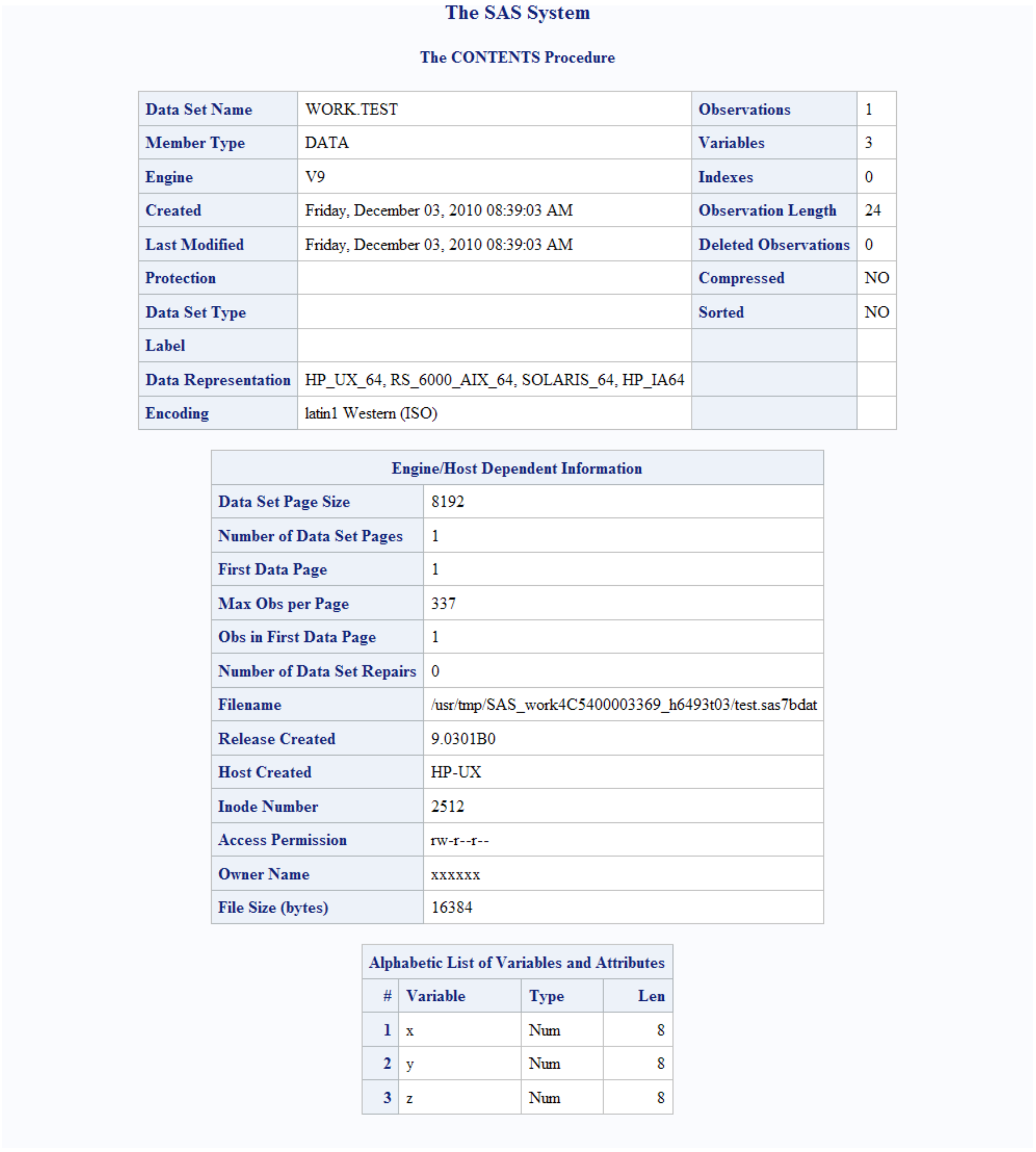
You can move a Version 8 data set that was created on a 32-bit Solaris host to a 32-bit HP-UX host using methods such as NFS, FTP, or CD. Because this data set was created on compatible computer types, you will be able to read this data set in SAS.

The same 32-bit Solaris data set can be moved to a 64-bit HP-UX host. Because these computer types are incompatible, SAS will use Cross-Environment Data Access (CEDA) to read this data set. For more information, see "Reading Version 8 or Later Files from Incompatible Computer Types" on page 49.

Determining Compatible Computer Types in SAS 9.3

In SAS 9.3, the `Data Representation` field of the PROC CONTENTS output shows the compatible computer types for a SAS file. The following is a display of the PROC CONTENTS output.

Display 2.1 *PROC CONTENTS Output Using the V9 Engine*



The SAS System

The CONTENTS Procedure

Data Set Name	WORK.TEST	**Observations**	1
Member Type	DATA	**Variables**	3
Engine	V9	**Indexes**	0
Created	Friday, December 03, 2010 08:39:03 AM	**Observation Length**	24
Last Modified	Friday, December 03, 2010 08:39:03 AM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64		
Encoding	latin1 Western (ISO)		

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	337
Obs in First Data Page	1
Number of Data Set Repairs	0
Filename	/usr/tmp/SAS_work4C5400003369_h6493t03/test.sas7bdat
Release Created	9.0301B0
Host Created	HP-UX
Inode Number	2512
Access Permission	rw-r--r--
Owner Name	xxxxxx
File Size (bytes)	16384

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	x	Num	8
2	y	Num	8
3	z	Num	8

In this example, the **Data Representation** field shows the compatible computer types for this data set. You can use NFS, FTP, or CD to move any data set to any computer and SAS will load the data set. If one data set has the same data representation, then SAS can read the data set natively. Otherwise, SAS must use CEDA. For more information about CEDA, see "Reading Version 8 or Later Files from Incompatible Computer Types" on page 49.

In SAS 9.3 for the Solaris x64 platform, PROC CONTENTS lists the following compatible computer types in the **Data Representation** field of the PROC CONTENTS output: **`SOLARIS_X86_64`**, **`LINUX_X86_64`**, **`ALPHA_TRU64`**, and **`LINUX_IA64`**.

The following table lists the possible values for the **`Data Representation`** field for SAS 9.3, and the corresponding computer types.

Table 2.4 *Data Representation Value for Each Computer Type in SAS 9.3*

Data Representation Value	Corresponding Computer Type
HP_UX_64	HP-UX PA-RISC 64-bit
HP_IA64	HP-UX Itanium processor family
LINUX	Linux on Intel 32-bit hardware
LINUX _X86_X64	Linux on 64-bit hardware
RS_6000_AIX_64	AIX 64-bit
SOLARIS_X64	Solaris for x64_x86
SOLARIS_64	Solaris 64-bit on SPARC

Note: The **`Encoding`** value might affect your ability to move SAS files between compatible computer types. It is important to note this value when you are transporting SAS files between hosts. For more information about encoding, see *SAS National Language Support (NLS): Reference Guide*.

Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments

What Is File Migration?

File migration moves libraries forward to a new release of SAS. In many cases, SAS files from previous releases or from other hosts are compatible with SAS 9.3. If the files are not compatible, you can use PROC MIGRATE, a utility procedure, to migrate your files and libraries. Information to consider when you migrate your files includes the release of SAS in which your data currently resides, what member types exist in your libraries, and whether you must move members from 32–bit libraries to 64–bit libraries. PROC MIGRATE streamlines the process of moving files forward.

For information about using PROC MIGRATE and the Compatibility Calculator, see Migration at the **`Technical Support Web site`**.

Benefits of Migrating SAS Files

Migrating SAS files enables you to do the following:

- have Update access to unsupported data files
- have access to indexes, integrity constraints, and other features
- use long names for formats and informats
- use more than 32,767 variables
- use suppressed transcoding of a specified variable

- avoid the overhead of reading or writing to 32–bit files in a 64–bit SAS session

How to Migrate a SAS Library

To migrate a SAS library, use the MIGRATE procedure. If you are migrating from a 32-bit to a 64-bit environment, and catalogs are present in the library, you must have access to a 32–bit Release 8 SAS/CONNECT or SAS/SHARE server.

Note: If Release 8.2 files were created on a 64–bit UNIX computer, then these files are native on UNIX in SAS 9 for computers with the same data representation as those files that were created in Release 8.2. You do not need to migrate these files.

For information about the MIGRATE procedure and how to use the PROC MIGRATE Calculator, see Migration at the **`Technical Support Web site`**.

Additional Resources

- For more information about the MIGRATE procedure and file compatibility, see Migration at the **`Technical Support Web site`**.
- For more information about reading Version 6 data sets, see "Reading Version 6 Files" on page 48.
- For more information about CEDA, see "Definition of Cross-Environment Data Access (CEDA)" in Chapter 32 of *SAS Language Reference: Concepts*.

Creating a SAS File to Use with an Earlier Release

The V9 engine differs slightly from previous SAS engines. The V9 engine supports longer format and informat names than previous SAS engines. For a discussion about how to ensure compatibility between releases, see *SAS Language Reference: Concepts*. You can also go to Migration at the **`Technical Support Web site`** for information about cross-release compatibility.

Reading SAS Files from Previous Releases or from Other Hosts

Reading Version 6 Files

Using the V6 read-only engine, SAS can read Release 6 data sets that were created by compatible computer types. In most cases, SAS invokes the V6 engine automatically, and you do not have to specify it. The following examples demonstrate how you can use the V6 engine.

- If you are running SAS 9.3 on Linux, you can use the V6 engine to read Release 6 data sets that were created with any Intel ABI release of SAS, such as SCO UNIX.
- If you are running SAS 9.3 on HP-UX, you can use the V6 engine to read Release 6 data sets that were created on HP-UX, Solaris, AIX, or IRIX.

For a list of the compatible computers types for V6, see "Compatible Computer Types for Release 6.12 through SAS 9.3" on page 44. For more information about the compatibility of Version 6 files with SAS 9.3, see *SAS Language Reference: Concepts*.

Reading Version 8 or Later Files from Compatible Computer Types

If your files were created in 64-bit SAS, they are compatible with SAS 9.3. You do not need to use CEDA to read your files. To view tables that show compatible computer types for Release 6.12 through SAS 9.3, see "Compatible Computer Types for Release 6.12 through SAS 9.3" on page 44.

Reading Version 8 or Later Files from Incompatible Computer Types

Compatibility of Existing SAS Files with SAS 9.3

In Release 8.2, both 32–bit and 64–bit SAS were available for the AIX, HP-UX, and Solaris operating environments. In SAS 9, SAS for these environments is 64-bit only. Some SAS files that were created in 32-bit releases of SAS cannot be read by the V9 engine.

SAS automatically tries to use CEDA to read data sets. If you use CEDA to read a data set, and include `msglevel=i` in your code, then SAS writes a note to the log.

Release 8.2 files that are 64–bit and that were created on a UNIX computer are native on UNIX for SAS 9. You do not need to migrate these files.

The following table lists the supported processing for each SAS file under CEDA.

Table 2.5 *Supported Processing for Release 8 32–Bit Files in SAS 9*

File Type	Support
SAS files	input processing, output processing (In SAS 9, if you create a new data file from the 32–bit file, the new file is generally 64–bit. For more information about CEDA, see *SAS Language Reference: Concepts*, or Migration at the `Technical Support Web site`.
MDDB file	input processing
PROC SQL view	input processing
SAS/ACCESS view for Oracle or Sybase	input processing
SAS/ACCESS view other than for Oracle or Sybase	no support
SAS catalog	no support
stored compiled DATA step program	no support
DATA step view	no support
item store	no support

Note: In SAS 9, if you create a new file from a 32–bit file, the new file is generally 64–bit. For more information about CEDA, see *SAS Language Reference: Concepts*, or Migration at the **Technical Support Web site**.

Accessing Version 8 or Later Files with CEDA

CEDA enables a SAS data set that was created in Version 8 or later in any directory-based operating environment (such as UNIX and Windows) to be read by a SAS session that is running in another directory-based environment. In SAS 9.3, if you try to access a data set that was created in a previous release, then SAS automatically uses CEDA to process the file. For example, if you are running SAS 9.3 on Linux, SAS uses CEDA to process a data set that was created in Release 8 on a 64-bit Solaris host. With CEDA, you have Read and Write access to these files. However, you will not be able to update the file. For information about compatibility, see Migration at the **Technical Support Web site**.

For best system performance, it is better to use data sets that are in the native format. Otherwise, CEDA might require additional CPU resources and might reduce system performance.

If you need to access 32-bit SAS data sets, SAS/ACCESS views from Oracle or Sybase, SQL views, or MDDB files from a 64-bit SAS session, then you can access these files using CEDA. CEDA provides Read and Write access to these files. However, CEDA does not support Update processing. CEDA consumes additional resources each time you read or write to these files.

Catalogs and other SAS files (not including SAS data sets) contain data structures that are known only to the application that created them. These catalogs and files might contain data objects other than character or numeric objects and, therefore, cannot be shared between 64-bit SAS and earlier 32-bit releases of SAS.

Referring to SAS Files by Using Librefs in UNIX Environments

Techniques for Referring to a SAS File

If you want to read or write to a permanent SAS file, you can refer to the SAS file in one of two ways:

- Refer to the data file directly by using its pathname in the appropriate statements (such as DATA, SET, MERGE, UPDATE, OUTPUT, and PROC).
- Assign a libref to the SAS library (directory) that contains the data file and use the libref as the first level of a two-level filename.

What Is a Libref?

A libref is an alias that you can use to refer to the library during a SAS session or job. You will probably want to use a libref when one of the following is true:

- The data file pathname is long and must be specified several times within a program.
- The pathname might change. If the pathname changes, you need to change only the statement assigning the libref, not every reference to the file.

- Your application will be used on other platforms. Using librefs makes it easier to port an application to other operating environments.
- You need to concatenate libraries. For more information, see "Assigning a Libref to Several Directories (Concatenating Directories)" on page 54.

Librefs can be stored in the SAS registry. For more information, see "Customizing Your SAS Registry Files" on page 18.

Assigning Librefs

Methods for Assigning Librefs

You can use any of the following items to assign a SAS libref:

- LIBNAME statement
- LIBNAME function
- DMLIBASSIGN command
- LIBNAME window
- SAS Explorer window

A libref assignment remains in effect for the duration of the SAS job, session, or process unless you clear the libref or use the same libref in another LIBNAME statement or LIBNAME function.

If you assign a libref from a SAS process, that libref is valid only within that SAS process. If you clear a libref from within a SAS process, that libref is not cleared from other SAS processes.

Using the LIBNAME Statement

The LIBNAME statement identifies a SAS library to SAS, associates an engine with the library, enables you to specify options for the library, and assigns a libref to it. For information about LIBNAME statement syntax, see "LIBNAME Statement: UNIX" on page 333.

Using the LIBNAME Function

The LIBNAME function takes the same arguments and options as the LIBNAME statement. For more information about the LIBNAME function, see "LIBNAME Function" in *SAS Functions and CALL Routines: Reference*.

Using the DMLIBASSIGN Command

Perform the following steps to assign a libref using the DMLIBASSIGN command:

1. Issue the DMLIBASSIGN command in the command window.

   The New Library dialog box appears.

2. Specify the libref in the **Name** field.
3. Specify an engine for the libref in the **Engine** field by selecting the default engine or another engine from the menu. Depending on the engine that you select, the fields in the **Library Information** area might change.
4. Click **Enable at startup** to assign this libref when you invoke SAS.

5. Specify the necessary information for the SAS library in the **Library Information** area. Depending on the engine that you select, there might not be a **Path** field available for input.
6. Specify LIBNAME options in the **Options** field. These options can be specific to your host or engine, including options that are specific to a SAS engine that accesses another software vendor's relational database system.
7. Click **OK**.

Using the LIBNAME Window

Perform the following steps to assign a libref from the LIBNAME window:

1. Issue the LIBNAME command in the command window.

   The LIBNAME window appears.
2. From the **File** menu, select **New**.

   The New Library dialog box appears.
3. Complete the fields in the New Library dialog box as described in "Using the DMLIBASSIGN Command" on page 51.
4. Click **OK**.

Using the SAS Explorer Window

Perform the following steps to assign a libref from the SAS Explorer window:

1. From the **File** menu, select **New** when the **Libraries** node in the tree structure is active.

   The New dialog box appears.
2. Select **Library**, and then click **OK**.

   The New Library dialog box appears.
3. Complete the fields in the New Library dialog box as described in "Using the DMLIBASSIGN Command" on page 51.
4. Click **OK**.

Permanently Assigning a Libref

You might want to save a libref so that it is valid between SAS sessions. You can assign a libref permanently by using one of the following methods:

- Specify the LIBNAME statement or LIBNAME function in an autoexec file. For more information, see "LIBNAME Function" in *SAS Functions and CALL Routines: Reference* or "LIBNAME Statement: UNIX" on page 333.
- Select **Enable at startup** when you assign a libref using the DMLIBASSIGN command, LIBNAME window, or SAS Explorer window. Selecting this option will save the libref in the SAS registry. For more information about these methods, see "Assigning Librefs" on page 51.
- Use environment variables as librefs. Include these environment variables in your start-up files so that these variables are set when SAS is invoked.

Accessing a Permanent SAS Library by Using a Libref

After you have defined a libref, you can use the libref in one of two ways to access a permanent SAS library:

- as the first level of a two-level SAS filename:

  libref.member-name
  where *libref* is the first-level name referring to the directory where the file is stored, and *member-name* is the name of the file being read or created.

- as the value of the USER= option. (For more information, see "Using One-Level Names to Access Permanent Files (User Library)" on page 61.)

For example, these SAS statements access the data file Final.sas7bdat in the Sales library that is stored in the **/users/myid/mydir** directory:

```
libname sales '/users/myid/mydir';
data sales.final;
```

Specifying Pathnames in UNIX Environments

Rules for Specifying Directory and Pathnames

Whether you specify a data filename directly in the various SAS statements, or you specify the library name in a LIBNAME statement and then refer to the libref, the same rules apply for specifying UNIX directory and file pathnames.

Specify directory and file pathnames in quotation marks. The level of specification depends on your current directory.

Example 1: Access a File That Is Not in the Current Directory

If **/u/2011/budgets** is not your current directory, then to access the data file named May, you must specify the entire pathname:

```
data '/u/2011/budgets/may';
```

If you wanted to use a libref, you would specify:

```
libname budgets '/u/2011/budgets';
data budgets.may;
```

Example 2: Access a File in the Current Directory

If **/u/2011/budgets** is your current directory, you could specify only the filenames:

```
data 'quarter1';
   merge 'jan' 'feb' 'mar';
run;
```

Note: If you omit the quotation marks, then SAS assumes that these data sets are stored in the Work directory.

If you wanted to use a libref, you would specify:

```
libname budgets '.';
data budgets.quarter1;
   merge budgets.jan budgets.feb budgets.mar;
run;
```

Valid Character Substitutions in Pathnames

You can use the character substitutions in the following table to specify pathnames.

Table 2.6 *Character Substitutions in Pathnames*

Characters	Meaning
~/	$HOME/ Can be used only at the beginning of a pathname.
~name/	name's home directory (taken from file **/etc/passwd**). Can be used only at the beginning of a pathname.
!sasroot	name of **sasroot** directory (see “The !SASROOT Directory” on page 443.) Specified only at the beginning of a pathname.
.	current working directory.
..	parent of current working directory.
$VARIABLE	environment variable VARIABLE.

Assigning a Libref to Several Directories (Concatenating Directories)

Introduction to Concatenating Directories

You can use the LIBNAME statement to assign librefs and engines to one or more directories, including the Work directory.

If you have SAS data sets located in multiple directories, you can treat these directories as a single SAS library by specifying a single libref and concatenating the directory locations, as in the following example:

```
libname income ('/u/2011/revenue', '/u/2011/costs');
```

This statement indicates that the two directories, **/u/2011/revenue** and **/u/2011/costs**, are to be treated as a single SAS library.

If you have already assigned librefs to your SAS libraries, you can use these librefs to indicate that you want to concatenate the libraries, as in this example:

```
libname income ('/u/2011/corpsale', '/u/2011/retail');
libname costs ('/u/2011/salaries', '/u/2011/expenses');
libname profits (income, costs, '/u/2011/capgain');
```

This statement indicates that the five directories, **`/u/2011/corpsale`**, **`/u/2011/retail`**, **`/u/2011/salaries`**, **`/u/2011/expenses`**, and **`/u/2011/capgain`**, are to be treated as a single SAS library.

How SAS Accesses Concatenated Libraries

When you concatenate SAS libraries, SAS uses a protocol for accessing the libraries, which depends on whether you are accessing the libraries for read, write, or update. (A protocol is a set of rules.)

SAS uses the protocol in the following sections to determine which directory is accessed. (The protocol illustrated by these examples applies to all SAS statements and procedures that access SAS files, such as the DATA, UPDATE, and MODIFY statements in the DATA step, and the SQL and APPEND procedures.)

Accessing Files for Input and Update

When a SAS data set is accessed for input or update, the first SAS data set that is found by that name is the one that is accessed. For example, if you submit the following statements and the data set **`old.species`** exists in both directories, the one in the **`mysasdir`** directory is the one that is printed:

```
libname old ('mysasdir','saslib');
proc print data=old.species;
run;
```

The same would be true if you opened **`old.species`** for update with the FSEDIT procedure.

Accessing Files for Output

If the data set is accessed for output, it is always written to the first directory, provided that the directory exists. If the directory does not exist, an error message is displayed. For example, if you submit the following statements, SAS writes the **`old.species`** data set to the first directory (**`mysasdir`**), and replaces any existing data set with the same name:

```
libname old ('mysasdir','saslib');
data old.species;
   x=1;
   y=2;
run;
```

If a copy of the **`old.species`** data set exists in the second directory, it is not replaced.

Accessing Data Sets with the Same Name

If you use the DATA and SET statements to access data sets with the same name, the DATA statement uses the output rules and the SET statement uses the input rules. When you execute the following statements, assume that **`test.species`** originally exists only in the second directory, **`mysasdir`**. Execute the following statements:

```
libname test ('sas','mysasdir');
data test.species;
   set test.species;
   if value1='y' then
```

```
      value2=3;
run;
```

The DATA statement opens **test.species** for output according to the output rules. That is, SAS opens a data set in the first of the concatenated libraries (**sas**). The SET statement opens the existing **test.species** data set in the second directory (**mysasdir**), according to the input rules. Therefore, the original **test.species** data set is not updated. After the DATA step executes, two **test.species** data sets exist, one in each directory.

Using Multiple Engines for a Library in UNIX Environments

You can assign multiple librefs to a single directory, and specify a different engine with each libref. For example, after the following statements are executed, data sets that are referenced by **one** are created and accessed using the default engine, while data sets that are referenced by **two** are created and accessed using the sequential engine:

```
libname one v9 '/users/myid/educ';
libname two v8 '/users/myid/educ';
```

Note: Keeping different types of libraries in one directory is not recommended because you must remember the appropriate engine for accessing each library. SAS cannot determine the right engine for accessing libraries in a directory that contains libraries of different types. For more information, see "Omitting Engine Names from the LIBNAME Statement" on page 337.

Using Environment Variables as Librefs in UNIX Environments

An environment variable can be used as a libref. The variable name must be in all uppercase characters, and the variable value must be the full pathname of the directory. That is, the name of the directory must begin with a slash.

Note: SAS on UNIX does not support the assignment of the User libref using the USER environment variable.

If you want to use the library in **/users/mydir/educ**, and you want to refer to it with the EDUC environment variable. You can define the variable at the following times:

- Before you invoke SAS. See "Defining Environment Variables in UNIX Environments" on page 24. For example, in the Korn shell, you would use:

  ```
  export EDUC=/users/mydir/educ
  ```

- After you invoke SAS. You can use the X statement (see "Executing Operating System Commands from Your SAS Session" on page 15) and the SAS **setenv** command:

  ```
  x setenv EDUC /users/mydir/educ;
  ```

You cannot specify an engine when you define a libref as an environment variable, so SAS determines which engine to use as described in "Omitting Engine Names from the LIBNAME Statement" on page 337.

After the libref is defined, you can use it to access data sets stored in the library:

```
proc print data=educ.class;
run;
```

Note: If a variable and a libref have the same name, but refer to different libraries, SAS uses the libref.

Librefs Assigned by SAS in UNIX Environments

SAS automatically defines three librefs:

Sashelp
: contains a group of catalogs that contain information that is used to control various aspects of your SAS session. The Sashelp library is in the **`!SASROOT`** directory. See "The !SASROOT Directory" on page 443.

Sasuser
: contains SAS catalogs that enable you to tailor features of SAS (such as window size, font settings, and printer entries) for your needs. If the defaults in the Sashelp library are not suitable for your applications, you can modify them and store your personalized defaults in your Sasuser library.

Work
: is the temporary, or scratch, library automatically defined by SAS at the beginning of each SAS session or job. The Work library stores two types of temporary files: those files that you create, and those files that are created internally by SAS as part of normal processing.

These librefs and the library libref are reserved librefs. If your site also has SAS/GRAPH software, the maps libref might be automatically defined. All of these libraries are described in "Permanent and Temporary Libraries" in *SAS Language Reference: Concepts*. Sasuser and Work have operating system dependencies.

Sasuser Library

What Is the Sasuser Library?

The Sasuser library contains the customizations (such as window size and positioning, colors, fonts, and printer entries) that you specified for your SAS session. When you invoke SAS, it looks for the **`Sasuser`** directory to find these customizations. If this directory does not exist, SAS uses the SASUSER system option to create it. The default directory is set in the system configuration file (sasv9.cfg) and is usually similar to the following:

```
-sasuser ~/sasuser.v91
```

This specification tells SAS to create a directory for the Sasuser libref in your home directory. To determine the value of this directory for your system, use PROC OPTIONS or **`libname sasuser LIST`**.

You can permit Read-Only access to the Sasuser library by using the RSASUSER system option. For information about the SASUSER and RSASUSER system options, see “SASUSER System Option: UNIX” on page 418 and “RSASUSER System Option: UNIX” on page 412.

After the Sasuser library has been created, SAS automatically assigns the same Sasuser libref to it each time you start a SAS session. It cannot be cleared or reassigned during a SAS session. If you delete the library, SAS re-creates it the next time you start a session. Because SAS assigns the libref for you, you do not need to use a LIBNAME statement before referencing this library.

Contents of the Sasuser Library

Your customizations are stored in one of the following locations in the Sasuser library:

- “Sasuser.Profile Catalog” on page 58
- “Sasuser.Registry Catalog” on page 59
- “Sasuser.Prefs File ” on page 60

Sasuser.Profile Catalog

Overview of the Sasuser.Profile Catalog

The Sasuser.Profile catalog is the profile.sas7bcat file in your Sasuser library. This catalog enables you to customize the way you work with SAS. SAS uses this catalog to store function key definitions, fonts for graphics applications, window attributes, and other information from interactive windowing procedures. SAS saves changes that you make to function key definitions, window attributes (such as size, color, and position), PMENU settings, and so on, in the Sasuser.Profile catalog. The information in the Sasuser.Profile catalog is accessed automatically by SAS when you need it for processing.

How SAS Accesses the Sasuser.Profile Catalog

SAS creates the Sasuser.Profile catalog the first time it tries to find it and it does not exist. If you are using an interactive windowing environment, then creating the Sasuser.Profile catalog occurs during system initialization in your first SAS session. If you are using one of the other modes of execution, the Sasuser.Profile catalog is created the first time you execute a SAS procedure that requires it.

When the Sasuser.Profile Catalog Does Not Exist

If the Sasuser.Profile catalog does not exist, then, at invocation, SAS checks for the Sashelp.Profile catalog. (This catalog will exist only if you have copied your Sasuser.Profile catalog to the Sashelp library.) If the Sashelp.Profile catalog exists, then SAS copies it to the Sasuser library, and this catalog becomes your new Sasuser.Profile catalog. If the Sashelp.Profile catalog does not exist, then SAS creates Sasuser.Profile using the default settings for a SAS session. The default settings for your SAS session are stored in several catalogs in the Sashelp library. If you make changes to key settings or other options, then the new information is stored in your Sasuser.Profile catalog. To restore the original default settings to the Sasuser.Profile catalog, use the CATALOG procedure or the CATALOG window to delete entries from your Sasuser.Profile catalog. By default, SAS then uses the corresponding entry from the Sashelp library.

Checking for an Uncorrupted Sasuser.Profile Catalog

When you invoke SAS, SAS checks for an existing, uncorrupted Sasuser.Profile catalog. If the catalog is found, SAS copies the Sasuser.Profile catalog to Sasuser.Profback. This backup catalog is used if Sasuser.Profile becomes corrupted.

If you invoke SAS and determine that your customizations have been lost, then your Sasuser.Profile catalog is either corrupted or locked by another SAS session that was started with the same user ID. If either of these conditions are true, then SAS uses Sashelp.Profile or Sasuser.Profback to replace the locked or corrupted Sasuser.Profile catalog.

If Your Sasuser.Profile Catalog Is Locked or Corrupted

If your Sasuser.Profile catalog is locked, then SAS checks for Sashelp.Profile. If Sashelp.Profile exists, SAS copies it to Work.Profile, and then saves the customizations to the Work.Profile catalog instead of the Sasuser.Profile catalog. This Work.Profile catalog is used for the duration of the SAS session. Because the contents of the Work directory are temporary, any customizations that you save to the Work.Profile catalog will be lost at the end of the SAS session.

If your Sasuser.Profile catalog is corrupted, SAS copies the corrupted catalog to Sasuser.Badpro.SAS, and then checks for Sasuser.Profback. If Sasuser.Profback exists, then SAS copies it to Sasuser.Profile. Any changes that you made to the Sasuser.Profile catalog during the previous session will be lost. If your Sasuser.Profile catalog is being used by multiple SAS sessions, then you can specify the RSASUSER system option to permit Read-Only access to the Sasuser library. Because this permission is Read-Only, you will not be able to save any customizations to your Sasuser.Profile catalog during that SAS session.

For more information about the Sasuser.Profile catalog and its related catalogs, as well as information about recovering locked or corrupted profile catalogs, see *SAS Language Reference: Concepts*.

Sasuser.Registry Catalog

Overview of the Sasuser.Registry Catalog

The Sasuser.Registry catalog is the regstry.sas7bitm file in your Sasuser library. If you change any Universal Printing entries or libref assignments during a SAS session, then SAS saves the changes in the Sasuser.Registry catalog.

How SAS Accesses the Sasuser.Registry Catalog

At invocation, SAS looks in the **`Sasuser`** directory to see whether it can write to the Sasuser.Registry catalog. If SAS cannot write to this catalog, then the following warning appears in the SAS log:

```
WARNING:  Unable to open SASUSER.REGISTRY.  WORK.REGISTRY will be used instead.
NOTE:  All registry changes will be lost at the end of the session.
```

If SAS can read the Sasuser.Registry catalog, then SAS copies the Sasuser.Registry catalog to create a Work.Registry catalog (in the Work library). This Work.Registry catalog will be used for the duration of the SAS session. Because the contents of the Work library are temporary, then any customizations that you save to the Work.Registry catalog will be lost at the end of the SAS session. However, the customizations saved in the Sasuser.Registry catalog will still exist.

If SAS cannot read the Sasuser.Registry catalog, then SAS creates the Work.Registry catalog using the default settings for a SAS session. In this case, SAS issues an additional warning to the SAS log:

```
WARNING:  Unable to copy SASUSER.REGISTRY to WORK.REGISTRY.
```

Sasuser.Prefs File

The settings that you specify in the Preferences dialog box (with the exception of resources on the **General** tab) are saved in the Sasuser.Prefs file. For more information about these resources, see "Modifying X Resources through the Preferences Dialog Box" on page 166.

Work Library

The Work library is the temporary library that is automatically defined by SAS at the beginning of each SAS session or job. The Work library stores temporary SAS files that you create, as well as files created internally by SAS.

To access files in the Work library, specify a one-level name for the file. The libref WORK is automatically assigned to these files unless you have assigned the User libref.

When you invoke SAS, it assigns the WORK libref to a subdirectory of the directory specified in the WORK system option described in "WORK System Option: UNIX" on page 435. This subdirectory is usually named **`SAS_workcode_nodename`**, where:

`workcode`
: is a 12-character code. The first four characters are randomly generated numbers. The next eight characters are based on the hexadecimal process identification number of the SAS session.

`nodename`
: is the name of the UNIX computer where the SAS process is running.

This libref cannot be cleared or reassigned during a SAS session.

The WORKINIT and WORKTERM system options control the creation and deletion of the Work library. For more information, see "WORKINIT System Option" in *SAS System Options: Reference* and "WORKTERM System Option" in *SAS System Options: Reference*.

Note: If a SAS session is terminated improperly (for example, with the **`kill –9`** command), SAS will not delete the **`SAS_workcode_nodename`** directory. You might want to use the "cleanwork command" on page 450 to delete the directories.

Multiple Work Directories

SAS can make the distribution of Work libraries dynamic by distributing Work libraries across several directories. This functionality eliminates the potential problem of filling up a single volume with all of the Work directories.

The WORK system option contains the PATHNAME argument, which can be a directory, or a file that contains a list of directories, that SAS can use for allocating

Work libraries. Individual Work libraries will still reside in a single directory. You can use the WORK system option in a configuration file or in the command line.

When the argument to WORK is a list of directories in a file, you can specify a method for choosing which directory to use for WORK. If you specify METHOD=RANDOM, then SAS chooses at random a directory from the list of available directories. If you choose METHOD=SPACE, then SAS chooses the directory that has the most available space.

For more information, see "WORK System Option: UNIX" on page 435.

Using One-Level Names to Access Permanent Files (User Library)

Introduction to One-Level Names

SAS data sets are referenced with a one- or two-level name. The two-level name has the form *libref.member-name*, where *libref* refers to the SAS library in which the data set resides, and *member-name* refers to the particular *member* within that library. The one-level name has the form *member-name* (without a *libref*). In this case, SAS stores the files in the temporary Work library. To override this action and have files with one-level names stored in a permanent library, you must first assign the User libref to an existing directory. To refer to temporary SAS files while User is assigned, use a two-level name with WORK as the libref.

Techniques for Assigning the User Libref

You have three ways to assign the User libref:

- Assign the User libref directly using the LIBNAME statement:

  ```
  libname user '/users/myid/mydir';
  ```

- Specify the USER= system option before you start the SAS session. For example, you can assign the User libref when you invoke SAS:

  ```
  sas -user /users/myid/mydir
  ```

- Specify the USER= system option after you start the SAS session. First, assign a libref to the permanent library. Then, use the USER= system option in an OPTIONS statement to equate that libref to User. For example, these statements assign the libref User to the directory with libref *mine*:

  ```
  libname mine '/users/myid/mydir';
  options user=mine;
  ```

For information about the USER system option, see "USER System Option: UNIX" on page 433.

Note: SAS on UNIX does not support the assignment of the User libref using the USER environment variable.

Accessing Disk-Format Libraries in UNIX Environments

You will probably create and access libraries on disk more than any other type of library. The default engine and the compatibility engines allow Read, Write, and Update access to SAS files on disk. They also support indexing and compression of observations.

In the following example, the In libref is assigned to a directory that contains the Stats1 data set:

```
libname in '/users/myid/myappl';
proc print data=in.stats1;
run;
```

Remember, a *SAS-data-library* must exist before SAS can read from or write to this directory. For example, if you want to create the SAS data set Orders in a directory, use the X statement to issue the **`mkdir`** UNIX command. Then, you can use the LIBNAME statement to associate the libref with the directory:

```
x mkdir /users/publish/books;
libname books '/users/publish/books';
data books.orders;
  ... more SAS statements ...
run;
```

By default, the LIBNAME statement associates the V9 engine with the directory.

Accessing Sequential-Format Libraries in UNIX Environments

Benefits and Limitations of Sequential Engines

The sequential engines enable you to access libraries in sequential format on disk. The sequential engines do not support indexing and compression of observations.

Note: Before using sequential engines, read the information about sequential libraries in *SAS Language Reference: Concepts*.

Reading and Writing Sequential Files

Using a Staging Directory

If you have files on tape, use a staging directory so that files can be processed directly from disk. You can use the UNIX **`tar`** command to move SAS data sets between the staging directory and tape. (Do not use the UNIX **`cp`** command.)

Writing Sequential Data Sets to Named Pipes

Why Use Named Pipes?

You can send output to and read input from the operating environment by using named pipes. For example, you might want to compress a data set or send it to a sequential access management system without creating intermediate files.

Syntax of the LIBNAME Statement

You can read from and write to named pipes from within your SAS session by specifying the pipe name in the LIBNAME statement:

LIBNAME *libref* '*pipename*';

Because you cannot position a pipe file, SAS uses a sequential engine to ensure sequential access. You do not have to specify the engine name.

Example: Creating a SAS Data Set Using a Named Pipe

To create a SAS data set and compress the data set without creating an intermediate, uncompressed data set, create a named pipe (such as **`mypipe`**) and enter the **`compress`** command:

```
mkfifo mypipe; compress <mypipe >sasds.Z
```

In your SAS session, assign a libref to the pipe and begin writing to the data set:

```
libname x 'mypipe';
data x.a;
  ...more SAS statements...
output;
run;
```

The data is sent to **`mypipe`** and then compressed and written to the data set. When SAS closes the data set, compression finishes, and you have a compressed, sequential data set in **`sasds.Z`**.

If you begin writing to a named pipe before the task on the other end (in this case, the **`compress`** command) begins reading, your SAS session will be suspended until the task begins to read.

Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments

Introduction to the BMDP, OSIRIS, and SPSS Files

SAS includes three interface library engines, BMDP, OSIRIS, and SPSS, that enable you to access external data directly from a SAS program. All of these engines are read-only.

Because they are sequential, these engines cannot be used with the POINT= option in the SET statement, or with the FSBROWSE, FSEDIT, or FSVIEW procedures. You can use PROC COPY, PROC DATASETS, or a DATA step to copy a BMDP or OSIRIS system file or an SPSS export file to a SAS data set, and then perform these functions on the SAS data set. Also, some procedures (such as PROC PRINT) give a warning message about the engine being sequential.

With these engines, the physical filename that is associated with a libref is an actual filename, not a directory. This association is an exception to the rules concerning librefs.

You can use the CONVERT procedure to convert BMDP, OSIRIS, and SPSS files to SAS files. For more information, see the “CONVERT Procedure: UNIX” on page 297.

The BMDP Engine

What Is the BMDP Engine?

The BMDP interface library engine enables you to read BMDP files from the BMDP statistical software application directly from a SAS program. The BMDP engine is a read-only engine. The following discussion assumes that you are familiar with the BMDP save file terminology. For more information, see the documentation that is provided by BMDP Statistical Solutions on the Web site.

Note: This engine is available for AIX, HP-UX, and Solaris.

Syntax for Accessing BMDP Save Files

To read a BMDP save file, issue a LIBNAME statement that explicitly specifies the BMDP engine. In this case, the LIBNAME statement has the following form:

LIBNAME *libref* BMDP '*filename*';

where:

libref
: specifies a SAS libref.

filename
: specifies a BMDP physical filename.

Note: If the libref appears previously as a fileref, omit *filename* because SAS uses the physical filename that is associated with the fileref.

This engine can read save files that are created only on UNIX.

Because a single physical file can contain multiple save files, you reference the CODE= value as the member name of the data set within the SAS language. For example, if the save file contains CODE=ABC and CODE=DEF, and the libref is MyLib, you reference the files as MyLib.ABC and MyLib.DEF. All CONTENT types are treated the same. Even if member DEF has the value CONTENT=CORR, it is treated as if the value was CONTENT=DATA.

If you know that you want to access the first save file in the physical file or if there is only one save file, refer to the member name as _FIRST_. This reference is convenient if you do not know the CODE= value.

Example: BMDP Engine

Assume that the physical file mybmdp.dat contains the save file ABC. The following SAS code associates the libref mylib with the BMDP physical file and executes the CONTENTS and PRINT procedures on the save file:

```
libname mylib bmdp 'mybmdp.dat';
proc contents data=mylib.abc;
run;

proc print data=mylib.abc;
run;
```

The following example uses the LIBNAME statement to associate the libref mylib2 with the BMDP physical file. Then, it writes the data for the first save file in the physical file:

```
libname mylib2 bmdp 'mybmdp.dat';
proc print data=mylib2._first_;
run;
```

The OSIRIS Engine

What Is the OSIRIS Engine?

The Inter-University Consortium for Political and Social Research (ICPSR) uses the OSIRIS file format for distribution of its data files. SAS provides the OSIRIS interface library engine to support the many users of the ICPSR data and to be compatible with PROC CONVERT.

With the OSIRIS engine, you can read OSIRIS data and dictionary files directly from a SAS program. The following discussion assumes that you are familiar with the OSIRIS file terminology and structure. If you are not, refer to the documentation provided by ICPSR.

Notes about the OSIRIS Data Dictionary Files

Because OSIRIS software does not run outside the z/OS environment, the layout of an OSIRIS data dictionary is consistent across operating environments. However, the OSIRIS engine is designed to accept a data dictionary from any other operating environment on which SAS runs. It is important that the dictionary and data files not be converted from EBCDIC to ASCII; the engine expects EBCDIC data.

The dictionary file should consist of fixed-length records of length 80. The data file should contain records that are large enough to hold the data that is described in the dictionary.

Syntax for Accessing an OSIRIS File

To read an OSIRIS file, issue a LIBNAME statement that explicitly specifies the OSIRIS engine. In this case, the syntax of the LIBNAME statement has the following form:

LIBNAME *libref* OSIRIS '*data-filename*' DICT='*dictionary-filename*';

where:

libref
: specifies a SAS libref.

'*data-filename*'
: specifies the physical filename of the data file.

  If the libref appears also as a fileref, omit *data-filename*.

DICT='*dictionary-filename*'
: specifies the physical filename of the dictionary file. If *dictionary-filename* is an environment variable or a fileref, do not enclose it in quotation marks. The DICT= option is required.

OSIRIS data files do not have member names. Therefore, use whatever member name you want.

To use the same dictionary file with different data files, use a separate LIBNAME statement for each one.

Example: OSIRIS Engine

In the following example, the data file is **`/users/myid/osr/dat`**, and the dictionary file is **`/users/myid/osr/dic`**. The example associates the libref mylib with the OSIRIS files, and executes the CONTENTS and the PRINT procedures:

```
libname mylib osiris '/users/myid/osr/dat'
   dict='/users/myid/osr/dic';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

The SPSS Engine

What Is the SPSS Engine?

The SPSS engine is a read-only engine. With the SPSS interface library engine, you can read only SPSS export files. This engine does not read SPSS-X native files.

Syntax for Accessing an SPSS Export File

To read an SPSS export file, issue a LIBNAME statement that explicitly specifies the SPSS engine. In this case, the syntax of the LIBNAME statement has the following form:

LIBNAME *libref* SPSS '*filename*';

where:

libref
: specifies a SAS libref.

'*filename*'
: specifies the physical filename.

Note: If the libref appears also as a fileref, omit *filename* because SAS uses the physical filename that is associated with the fileref.

Export files must be created by the SPSS EXPORT command and can originate from any operating environment. Export files must be transported to and from your operating environment in ASCII format. If they are transported in binary format, other operating environments will not be able to read them.

Because SPSS-X files do not have internal names, refer to them by any member name that you like. A common extension for export files is .por, but this extension is not required.

SPSS can have system-missing and user-defined missing data. When you use the SPSS engine or PROC CONVERT, the missing values (user-defined or system-missing) are converted to system-missing values. User-defined missing values have to be recoded as valid values. When the data set is converted, you can use PROC FORMAT to make the translation. For example, -1 to .A and -2 to .B.

Reformatting SPSS Files

SAS cannot use an SPSS file that contains a variable with a numeric format that has a larger number of decimal places than the width of the entire variable. For example, if an SPSS file has a variable with a width of 17 and has 35 decimal places, SAS will return errors when you try to run a DATA step on the file or view it with the table viewer. To use the SPSS file with SAS, you have to reformat the variable.

You can reformat the variable by reducing the number of decimal spaces to a value that fits within the width of the variable. In the following example, the statement **`revision=cat(format,formatl,'.2');`** converts the number of decimal spaces to 2. This value reduces the number of decimal spaces so that the number is not greater than the width of the variable.

```
libname abc spss 'FILENAME.POR';
proc contents data=abc._all_ out=new;
run;

filename sascode temp;
data _null_;
   set new;
   file sascode
   if formatd > formatl then do;
      revision=cat(format,formatl,'.2');
      put 'format' +1 name +1 revision ';' ;
   end;
run;

data temp;
    set abc._all_;
   %inc sascode/source2;
run;
```

Note: The OPTIONS NOFMTERR statement does not allow SAS to use a data set with a DATA step or the table viewer. You have to reformat numeric variables that have a larger decimal value than their width before you can use a DATA step or the table viewer.

Example: SPSS Engine

The following example associates the libref mylib with the physical file **`/users/myid/mydir/myspssx.por`** to execute the CONTENTS and PRINT procedures on the export file:

```
libname mylib spss '/users/myid/mydir/myspssx.por';
proc contents data=mylib._first_;
proc print data=mylib._first_;
run;
```

In the next example, the FILENAME statement associates the fileref mylib2 with the **`/users/myid/mydir/aspssx.por`** SPSS physical file, and the LIBNAME statement associates the libref with the SPSS engine. The PRINT procedure writes the data from the portable file.

```
filename mylib2 '/users/myid/mydir/aspssx.por';
libname mylib2 spss;
proc print data=mylib2._first_;
run;
```

Support for Links in UNIX Environments

SAS provides limited support for hard links and symbolic links in UNIX environments. You can create links that point to a SAS data set or SAS catalog. If you reference the link in a SAS program, SAS follows the link to find the data set or catalog.

For example, you can create a symbolic link in the **/tmp** directory to the **/home/user/mydata.sas7bdat** data set by entering the following command at the UNIX prompt:

```
ln -s /home/user/mydata.sas7bdat /tmp/mydata.sas7bdat
```

The following SAS code uses the symbolic link in the **/tmp** directory to find the **mydata.sas7bdat** data set. This code does not change the symbolic link, but it does sort the data in the data set.

```
libname tmp '/tmp';

proc sort data=tmp.mydata;
   by myvariable;
run;
```

If you are running in the SAS windowing environment, you can use the SAS Explorer window to view the symbolic links that are stored within a specific directory. Any symbolic link that points to a nonexistent SAS file will have a file size of **0.0KB** and a modified date of **31DEC59:19:00:00**.

Note: SAS does not support links for a version data set or for a data set that has an index.

Chapter 3

Using External Files and Devices

Introduction to External Files and Devices in UNIX Environments

At times during a SAS session, you might want to use external files, that is, files that contain data or text, or files in which you want to store data or text. These files are created and maintained by other applications or by SAS. You can create, read, write, and delete external files from within SAS.

You can use external files in a SAS session to perform the following functions:

- hold raw data to be read with the INPUT statements
- store printed reports created by a SAS procedure
- submit a file containing SAS statements for processing
- store data written with PUT statements

For SAS, external files and devices can serve both as sources of input and as receivers of output. The input can be either raw data to be read in a DATA step or SAS statements to be processed by SAS. The output can be one of the following:

- the SAS log, which contains notes and messages produced by the program
- the formatted output of SAS procedures
- data written with PUT statements in a DATA step

You might also want to use peripheral devices such as a printer, plotter, or your own terminal. UNIX treats these I/O devices as if they were files. Each device is associated with a file, called a special file, which is treated as an ordinary disk file. When you write to a special file, the associated device is automatically activated. All special files reside in the `dev` directory or its subdirectories. Although there are some differences in how you use the various devices, the basic concept is the same for them all.

UNIX also enables you to use pipes to send data to and from operating system commands as if they were I/O devices.

If you need to access an external file containing a transport data library, refer to *Moving and Accessing SAS Files*.

Accessing an External File or Device in UNIX Environments

Specifying a Pathname or a Fileref

To access an external file or device, you need to specify its pathname or fileref in the appropriate SAS statements:

FILE
: specifies the current output file for PUT statements.

%INCLUDE
: includes a file that contains SAS source statements that are executed when you submit a program from the Program Editor.

  TIP If you use %INCLUDE, the line limit is 6000 bytes.

INFILE
: identifies an external file that you want to read with an INPUT statement.

In the SAS statement, refer to the file or device in one of two ways:

- Specify the pathnames for the external files. For more information, see “Specifying Pathnames in UNIX Environments” on page 72.
- Assign a fileref to a device, one or more files, or a directory, and use the fileref when you want to refer to the file, directory, or device.

  In most cases, you will want to use a fileref.

What Is a Fileref?

A fileref is nickname that you assign to a file or device. You assign the fileref once, and then use it as needed. Filerefs are especially useful under the following conditions:

- The pathname is long and has to be specified several times within a program.
- The pathname might change. If the pathname changes, you need to change only the statement that assigns the fileref, not every reference to the file.

You can assign filerefs in the File Shortcuts window of the Explorer, with the FILENAME statement, with the FILENAME function, or by defining the fileref as an environment variable.

Note: For a complete description of the FILENAME statement and the FILENAME function, see “FILENAME Statement” in *SAS Statements: Reference* and “FILENAME Function” in *SAS Functions and CALL Routines: Reference*.

Specifying Pathnames in UNIX Environments

Rules for Specifying Pathnames

You can reference an external file directly by specifying its pathname in the FILE, INFILE, or %INCLUDE statements. You can reference the file indirectly by specifying a fileref and a pathname in the FILENAME statement and then using the fileref in the FILE, INFILE, or %INCLUDE statements.

Whether you reference a file directly or indirectly, you need to specify its pathname in the appropriate statement. In most cases, you must enclose the name in quotation marks. For example, the following INFILE statement refers to the file **`/users/pat/cars`**:

```
infile '/users/pat/cars';
```

The following FILE statement directs output to a specified special device file:

```
file '/dev/ttyp1';
```

Note: If a filename has leading blanks, then the blanks will be trimmed.

The level of specification depends on your current directory. You can use the character substitutions shown in Table 2.6 on page 54 to specify the pathname. You can also use wildcards as described in "Using Wildcards in Pathnames (Input Only)" on page 73.

Omitting Quotation Marks in a Filename

You can omit the quotation marks in a filename if one of the following is true:

- There is not already a fileref defined with that filename.
- The file has the filename extension expected by the statement that you are using to refer to the file. If you do not enclose a filename in quotation marks, the FILE and INFILE statements assume a file extension of .dat, and the %INCLUDE statement assumes a file extension of .sas.
- The file is located in the current directory.
- The filename is written with all lowercase characters.

For example, if the current directory is **`/users/mkt/report`** and it includes file **`qtr.sas`**, you can reference **`qtr.sas`** in any of the following statements:

```
%include '/users/mkt/report/qtr.sas';
%include 'qtr.sas';
file 'qtr.sas';
```

If there is no **`qtr`** fileref already defined, you can omit the quotation marks and the filename extension in the %INCLUDE statement:

```
%include qtr;
```

Working with Mixed Case or Uppercase Filenames

Filenames in the UNIX operating system are case sensitive. This means that a file named **`PROGRAM`** is not the same as a file named **`program`**. When you reference the name of a file that is written in mixed case or uppercase, and that filename is not enclosed in quotation marks, SAS converts the filename to lowercase. If the filename does not have a file extension, SAS adds the missing file extension.

For example, if you specify **`%include code(PROGRAM);`** in your program, SAS converts the filename PROGRAM to lowercase, and adds an extension of .sas to the filename. PROGRAM becomes **`program.sas`**.

Interpreting the Messages in the SAS Log

When you execute the following program, SAS converts **`TEMP`** to **`temp`**, and adds an extension of .sas to the filename:

```
filename inc_code 'your-directory';
%include inc_code(TEMP);
```

SAS writes the following messages to the SAS log:

```
WARNING: Physical file does not exist, A.../your-directory/TEMP.sas.
ERROR: Cannot %INCLUDE member TEMP in the aggregate INC_CODE.
```

The warning message shows only the original filename (TEMP.sas), and not the lowercase conversion (temp.sas). This situation might cause confusion if a file named TEMP.sas does exist.

To avoid this confusion, include the file extension with the filename if the filename contains an extension, or enclose the mixed case or uppercase filename in quotation marks if the filename does not have an extension. For example:

```
%include code(TEMP.sas);
%include code("TEMP");
```

In both of these cases, SAS does not convert TEMP to lowercase.

Using Wildcards in Pathnames (Input Only)

Descriptions of the Valid Wildcards

You can use the *, ?, and [ ] wildcards to specify pathnames in the FILENAME (only if the fileref is to be used for input), INFILE, and %INCLUDE statements and the INCLUDE command.

*
: matches one or more characters, except for the period at the beginning of filenames.

?
: matches any single character.

[ ]
: matches any single character from the set of characters defined within the brackets. You can specify a range of characters by specifying the starting character and ending character separated by a hyphen.

Wildcards are supported for input only. You cannot use wildcards in the FILE statement.

Example 1: Selecting Files by Including a Wildcard in a String

The following example reads input from every file in the current directory that begins with the string **`wild`** and ends with .dat:

```
filename wild 'wild*.dat';
data;
   infile wild;
   input;
run;
```

Example 2: Reading Each File in the Current Directory

The following example reads input from every file in every subdirectory of the current working directory:

```
filename subfiles '*/*';
data;
   infile subfiles;
   input;
run;
```

If new files are added to any of the subdirectories, they can be accessed with the Subfiles fileref without changing the FILENAME statement.

Example 3: Wildcards in Filenames When Using Aggregate Syntax

You can also use wildcards in filenames, but not in directory names, when you use aggregate syntax:

```
filename curdir ".";
data;
  infile curdir('wild*');
  input;
run;
```

In the example above, the period in the FILENAME statement refers to the current directory.

See "Valid Character Substitutions in Pathnames" on page 54 for information about character substitutions available in UNIX.

Example 4: Associating a Fileref with Multiple Files

The following statement associates the fileref MyRef with all files that begin with alphabetic characters. Files beginning with numbers or other characters such as the period or tilde are excluded.

```
filename myref '[a-zA-Z]*.dat';
```

The following statement associates MyRef with any file beginning with Sales (in either uppercase, lowercase, or mixed case) and a year between 2010 and 2019:

```
filename myref '[Ss][Aa][Ll][Ee][Ss]201[0-9].dat';
```

Assigning Filerefs to External Files or Devices with the FILENAME Statement

Introduction to the FILENAME Statement

The most common way to assign a fileref to an external file or device is with the FILENAME statement. There are several forms of the FILENAME statement, depending on the type of device that you want to access. For more information, see "FILENAME Statement: UNIX" on page 323.

Accessing DISK Files

The most common use of the FILENAME statement is to access DISK files. The FILENAME syntax for a DISK file is the following:

FILENAME *fileref* <DISK> *'pathname'* <*options*>;

The following FILENAME statement associates the fileref **`myfile`** with the external file **`/users/mydir/myfile`**, which is stored on a disk device:

```
filename myfile disk '/users/mydir/myfile';
```

The following FILENAME statement assigns a fileref of **`prices`** to the file **`/users/pat/cars`**. The FILE statement then refers to the file using the fileref:

```
filename prices '/users/pat/cars';
data current.list;
   file prices;
   ...PUT statements...
run;
```

For more information about using DISK files, see "Concatenating Filenames in UNIX Environments" on page 77.

Note: If a filename has leading blanks, then they will be trimmed.

Debugging Code with DUMMY Devices

You can substitute the DUMMY device type for any of the other device types. This device type serves as a tool for debugging your SAS code without actually reading or writing to the device. After debugging is complete, replace the DUMMY device name with the proper device type, and your program will access the specified device type.

Here is the FILENAME syntax for a DUMMY file:

FILENAME *fileref* DUMMY *'pathname'* <*options*>;

Output to DUMMY devices is discarded.

Sending Output to PRINTER Devices

The PRINTER device type enables you to send output directly to a printer. Here is the FILENAME syntax to direct a file to a PRINTER:

FILENAME *fileref* PRINTER '<*printer*> <*printer-options*>' <*options*>;

For example, this SAS program sends the output file to the BLDG3 printer:

```
filename myfile printer 'bldg3';

data test;
   file myfile;
   put 'This will appear in bldg3 .';
run;
```

For more information, see "Printing the Contents of a Window" on page 98 and "Using the PRINTTO Procedure in UNIX Environments" on page 100.

Using Temporary Files (TEMP Device Type)

The TEMP device type associates a fileref with a temporary file stored in the same directory as the Work library. (See “Work Library” on page 60.) Using the TEMP device type enables you to create a file that lasts only as long as the SAS session.

Here is the FILENAME syntax for a TEMP file:

FILENAME *fileref* TEMP <*options*>;

For example, this FILENAME statement associates Tmp1 with a temporary file:

```
filename tmp1 temp;
```

Accessing TERMINAL Devices Directly

To access a terminal directly, use the TERMINAL device type. Here is the FILENAME syntax to associate a file with a terminal:

FILENAME *fileref* TERMINAL <*'terminal-pathname'*> <*options*>;

The *terminal-pathname* must be a pathname of the special file associated with the terminal. Check with your UNIX system administrator for information. Enclose the name in quotation marks. If you omit the terminal pathname, the fileref is assigned to your terminal.

For example, this FILENAME statement associates the fileref **`here`** with your terminal:

```
filename here terminal;
```

The following FILENAME statement associates the fileref **`thatfile`** with another terminal:

```
filename thatfile terminal '/dev/tty3';
```

Assigning Filerefs to Files on Other Systems (FTP, SFTP, and SOCKET Access Types)

You can access files on other systems in your network by using the FTP, SFTP, and SOCKET access methods. Here are the forms of the FILENAME statement:

FILENAME *fileref* FTP *'external-file'* <*ftp-options*>;

FILENAME *fileref* SFTP *'external-file'* <*sftp-options*>;

FILENAME *fileref* SOCKET *'external-file'* <*tcpip-options*>;

FILENAME *fileref* SOCKET *':portno'* SERVER <*tcpip-options*>;

These access methods are documented in *SAS Statements: Reference*. Under UNIX, the FTP access method supports an additional option:

MACH=*'machine'*
: identifies which entry in the **`.netrc`** file should be used to get the user name and password. The **`.netrc`** file resides on the host on which the SAS program is running. See the UNIX man page for more information about the **`.netrc`** file. You cannot specify the MACH option together with the HOST option in the FILENAME statement.

If you are transferring a file to UNIX from the z/OS operating environment and you want to use either the S370V or S370VB format to access that file, then the file must be of type RECFM=U and BLKSIZE=32760 before you transfer it.

CAUTION:

When you use the FTP access method to create a remote file, the UNIX permissions for that file are set to -rw-rw-rw-, which makes the file world-readable and world-writable. See the UNIX man page for `chmod` for information about changing file permissions.

Concatenating Filenames in UNIX Environments

You can concatenate filenames in the FILENAME, %INCLUDE, and INFILE statements. Concatenating filenames enables you to read those files sequentially.

FILENAME *fileref* ("*pathname-1*" ... "*pathname-n*");

%INCLUDE '("*filename-1*" ... "*filename-n*")';

%INCLUDE "('*filename-1*' ... '*filename-n*')";

INFILE '("*filename-1*" ... "*filename-n*")';

INFILE "('*filename-1*' ... '*filename-n*')";

You can enclose the pathnames in single or double quotation marks and separate them with commas or blank spaces. You can use the characters shown in Table 2.6 on page 54 and the wildcards described in "Using Wildcards in Pathnames (Input Only)" on page 73 to specify the pathnames.

Assigning a Fileref to a Directory (Using Aggregate Syntax)

Introduction to Aggregate Syntax

Aggregate Syntax

Aggregate syntax enables you to assign a fileref to a directory and then work with any file in that directory by specifying its filename in parentheses after the fileref.

FILENAME *fileref directory-name*;

Aggregate syntax is especially useful when you have to refer to several files in one directory.

Example 1: Referring to a File Using Aggregate Syntax

To refer to a file in the directory, specify the fileref followed by the individual filename in parentheses. For example, you can refer to the file cars.dat in the directory **`/users/pat`** as shown in this example:

```
filename prices '/users/pat';
data current.list;
   file prices(cars);
   ...other SAS statements...
run;
```

Example 2: Using Aggregate Syntax with Filerefs Defined by Environment Variables

You can also use aggregate syntax with filerefs that have been defined using environment variables. (See "Using Environment Variables to Assign Filerefs in UNIX Environments" on page 78.) For example:

```
x setenv PRICES /users/pat;
data current.list;
   file prices(cars);
   ...other SAS statements...
run;
```

Assigning a Fileref to Several Directories

In the FILENAME statement, you can concatenate directory names and use the fileref to refer to any file within those directories:

FILENAME *fileref* ("*directory-1*" ... "*directory-n*");

When you concatenate directory names, you can use aggregate syntax to refer to a file in one of the directories. For example, assume that the Report.sas file resides in the directory associated with the MYPROGS environment variable. When SAS executes the following code, it searches for Report.sas in the pathnames that are specified in the FILENAME statement and it executes the program.

```
filename progs ("$MYPROGS" "/users/mkt/progs");
%inc progs(report);
```

SAS searches the pathnames in the order specified in the FILENAME statement until

- it finds the first file with the specified name. Even if you use wildcards (see "Using Wildcards in Pathnames (Input Only)" on page 73) in the filename, SAS matches only one file.
- it encounters a filename in the list of pathnames that you specified in the FILENAME statement.

Using Environment Variables to Assign Filerefs in UNIX Environments

Requirements for Variable Names

An environment variable can also be used as a fileref to refer to DISK files. The variable name must be in all uppercase characters, and the variable value must be the full pathname of the external file. That is, the filename must begin with a slash.

Note: If a variable and a fileref have the same name but refer to different files, SAS uses the fileref. For example, the %INCLUDE statement below refers to file **/users/myid/this_one**.

```
filename ABC '/users/myid/this_one';
x setenv ABC /users/myid/that_one;
%include ABC;
```

Reading a Data File

If you want to read the data file **`/users/myid/educ.dat`**, but you want to refer to it with the INED environment variable, you can define the variable at two times:

- Before you invoke SAS, see "Defining Environment Variables in UNIX Environments" on page 24. For example, in the Korn shell, you use the following:

  ```
  export INED=/users/myid/educ.dat
  ```

- After you invoke SAS by using the X statement (see "Executing Operating System Commands from Your SAS Session" on page 15) and the SAS **`setenv`** command:

  ```
  x setenv INED /users/myid/educ.dat;
  ```

After INED is associated with the file **`/users/myid/educ.dat`**, you can use **`ined`** as a fileref to refer to the file in the INFILE statement:

```
infile ined;
```

Writing to an External File

The same method applies if you want to write to an external file. For example, you can define OUTFILE before you invoke SAS:

```
OUTFILE=/users/myid/scores.dat
export OUTFILE
```

Then, use the environment variable name as a fileref to refer to the file:

```
file OUTFILE;
```

Filerefs Assigned by SAS in UNIX Environments

Filerefs for Standard Input, Standard Output, and Standard Error

Often a command's arguments or options tell the command what to use for input and output, but in case they do not, the shell supplies you with three standard files: one for input (standard input), one for output (standard output), and one for error messages (standard error). By default, these files are all associated with your terminal: standard input with your keyboard, and both standard output and standard error with your terminal's display. When you invoke SAS, it assigns a fileref to each file that it opens, including the three standard files. SAS assigns the filerefs Stdin, Stdout, and Stderr to standard input, standard output, and standard error, respectively.

File Descriptors

What Is a File Descriptor?

Each file has an internal file descriptor assigned to it. By default, 0 is the file descriptor for standard input, 1 is the file descriptor for standard output, and 2 is the file descriptor for standard error. As other files are opened, they get other file descriptors. In the Bourne shell and in the Korn shell, you can specify that data be written to or be read from a file using the file descriptor as described in "File Descriptors in the Bourne and Korn Shells" on page 80.

File Descriptors in the Bourne and Korn Shells

If you are using the Bourne shell or the Korn shell, SAS assigns filerefs of the following form to files that have a file descriptor (see "Filerefs Assigned by SAS in UNIX Environments" on page 79) larger than 2.

```
FILDESnumber
```

number is a two-digit representation of the file descriptor. You can use these filerefs in your SAS applications.

For example, if you invoke SAS with the following command, then the operating environment opens the file sales_data and assigns file descriptor 4 to it:

```
sas salespgm 4< sales_data
```

SAS assigns the fileref FILDES04 to the file and executes the application **salespgm**. When the application reads input from FILDES04, it reads the file sales_data. Using file descriptors as filerefs enables you to use the same application to process data from different files without changing the application to refer to each file. In the command that you use to invoke the application, you assign the appropriate file descriptor to the file to be processed.

Reserved Filerefs in UNIX Environments

The following filerefs are reserved.

DATALINES fileref in the INFILE statement
: specifies that input data immediately follow a DATALINES statement. You need to use INFILE DATALINES only when you want to specify options in the INFILE statement to read instream data.

LOG fileref in the FILE statement
: specifies that output lines produced by PUT statements be written to the SAS log. LOG is the default destination for output lines.

PRINT fileref in the FILE statement
: specifies that output lines produced by PUT statements be written to the same print file as output produced by SAS procedures.

Sharing External Files in a UNIX Environment

Sharing External Files

If more than one user has simultaneous Write access to an external file, or if a single user has Write access to the same file from different SAS sessions, the results of sharing the file can be unpredictable. To remedy this situation, you can use a statement or a system option to restrict Write access to one user, while allowing multiple users Read access. For more information, see "Sharing SAS Files" on page 40.

Options to Use for File Locking: External Files

File locking applies to all files that are opened. You can turn off file locking for external files in the following ways:

- Use the LOCKINTERNAL option in the FILENAME statement.
- Use the FILELOCKS system option.

File Locking for External Files: The LOCKINTERNAL Statement Option

You can control file locking for external files by using the LOCKINTERNAL option in the FILENAME statement. The AUTO option value locks a file exclusively for Write access, or non-exclusively for Read access. For example, if a file is opened for update or output, then all other access from internal processes will be blocked. If a file is opened for input, then other users can also open the file for input. In this case, opening the file for update and output will be blocked. The SHARED option value allows for all of the behavior of the AUTO option, except that the file can be shared by one writer and multiple readers. The external file that is associated with the fileref is the file that is locked. By default, multiple users can simultaneously read an external file. For more information, see "FILENAME Statement: UNIX" on page 323.

File Locking for External Files: The FILELOCKS System Option

You can control file locking for external files (as well as for SAS files) by using the FILELOCKS system option. This option enables you to apply a behavior globally to individual files or directories. Using FILELOCKS restricts writer access to one user. With file locking turned on, multiple SAS sessions are able to simultaneously read the same file. You can use FILELOCKS at start-up, in the OPTIONS statement, or in the command line. You can specify multiple instances of the FILELOCKS option. Each instance is added to an internal table of paths and settings. For more information, see "FILELOCKS System Option: UNIX" on page 381.

Reading from and Writing to UNIX Commands (PIPE)

What Are Pipes?

Pipes enable your SAS application to receive input from any UNIX command that writes to standard output and to route output to any UNIX command that reads from standard input. In UNIX commands, the pipe is represented by a vertical bar (|). For example, to find the number of files in your directory, you could redirect the output of the **ls** command through a pipe to the **wc** (word count) command:

```
ls | wc -w
```

Syntax of the FILENAME Statement to Assign a Fileref to a Pipe

Under UNIX, you can use the FILENAME statement to assign filerefs not only to external files and I/O devices, but also to a pipe. Here is the syntax of the FILENAME statement:

FILENAME *fileref* PIPE '*UNIX-command*' <*options*>;

fileref
: is the name by which you reference the pipe from SAS.

PIPE
: identifies the device-type as a UNIX pipe.

'*UNIX-command*'
: is the name of a UNIX command, executable program, or shell script to which you want to route output or from which you want to read input. The commands must be enclosed in either double or single quotation marks.

options
: control how the external file is processed. For an explanation of these options, see "FILENAME Statement: UNIX" on page 323.

Whether you are using the command as input or output depends on whether you use the *fileref* in a reading or writing operation. For example, if the fileref is used in an INFILE statement, then SAS assumes that the input comes from a UNIX command. If the fileref is used in a FILE statement, then SAS assumes that the output goes to a UNIX command.

Using the Fileref for Reading

Specifying a Fileref for Reading

When the fileref is used for reading, the specified UNIX command executes, and any output sent to its standard output or standard error is read through the fileref. In this case, the standard input of the command is connected to **/dev/null**.

Example 1: Sending the Output of the Process Command to a SAS DATA Step

The following SAS program uses the PIPE device-type keyword to send the output of the **ps** (process) command to a SAS DATA step. The resulting SAS data set contains data about every process currently running SAS:

```
filename ps_list pipe "ps -e|grep 'sas'";
data sasjobs;
   infile ps_list;
   length process $ 80;
   input process $ char80.;
run;
proc print data=sasjobs;
run;
```

The **ps -e** command produces a listing of all active processes in the system, including the name of the command that started the task. In BSD-based UNIX systems, you use the **ps -ax** command.

The operating environment uses pipes to send the output from **ps** to the **grep** command, which searches for every occurrence of the string **'sas'**. The FILENAME statement connects the output of the **grep** command to the fileref **ps_list**. The DATA step then creates a data set named **sasjobs** from the INFILE statement that points to the input source. The INPUT statement reads the first 80 characters on each input line.

Example 2: Using the Stdin Fileref to Read Input

In the next example, the Stdin fileref is used to read input through a pipe into the SAS command, which, in turn, executes the SAS program. By placing the piping operation outside the SAS program, the program becomes more general. The program in the previous example has been changed and stored in file ps.sas:

```
data sasjobs;
   infile stdin;
   length process $ 80;
   input process $ char80.;
run;
proc print data=sasjobs;
run;
```

To run the program, use pipes to send the output of **ps** to **grep** and from **grep** into the SAS command:

```
ps -e|grep 'sas'|sas ps.sas &
```

The output will be stored in **ps.lst**, and the log will be stored in **ps.log**, as described in "The Default Routings for the SAS Log and Procedure Output in UNIX Environments" on page 93.

Using the Fileref for Writing

Specifying a Fileref for Writing

When the fileref is used for writing, the output from SAS is read in by the specified UNIX command, which then executes.

Example 1: Sending Mail Using Pipes

In this example, any data sent to the **mail** fileref are piped to the **mail** command and sent to user PAT:

```
filename mail pipe 'mail pat';
```

Example 2: Starting a Remote Shell and Printing Output

Consider this FILENAME statement:

```
filename letterq pipe 'remsh alpha lp -dbldga3';
```

Any data sent to the **letterq** fileref is passed to the UNIX command, which starts a remote shell on the computer named Alpha. Note that the form of the command that starts a remote shell varies among the various UNIX operating systems. The shell then prints the **letterq** output on the printer identified by the destination BLDGA3. Any messages that are produced by the **lp** command are sent to the SAS log.

Sending Electronic Mail Using the FILENAME Statement (EMAIL)

Advantages of Sending Electronic Mail from within SAS

SAS lets you send electronic mail using SAS functions in a DATA step or in SCL. Sending e-mail from within SAS enables you to do the following:

- Use the logic of the DATA step or SCL to subset e-mail distribution based on a large data set of e-mail addresses.
- Send e-mail automatically upon completion of a SAS program that you submitted for batch processing.
- Direct output through e-mail based on the results of processing.
- Send e-mail messages from within a SAS/AF frame application, customizing the user interface.

Initializing Electronic Mail

By default, SAS uses SMTP (Simple Mail Transfer Protocol) to send e-mail. SMTP, unlike some external scripts, supports attachments. This default is specified by the EMAILSYS system option. For information about how to change the e-mail protocol, see "EMAILSYS System Option: UNIX" on page 379.

Before you can send e-mail from within SAS, your system administrator might need to set the EMAILHOST system option to point to the SMTP server. For more information, see "EMAILHOST= System Option" in *SAS System Options: Reference*.

Components of the DATA Step or SCL Code Used to Send E-mail

In general, a DATA step or SCL code that sends electronic mail has the following components:

- a FILENAME statement with the EMAIL device-type keyword
- options specified in the FILENAME or FILE statements indicating the e-mail recipients, subject, and any attached files
- PUT statements that contain the body of the message
- PUT statements that contain special e-mail directives (of the form !EM_*directive*!) that can override the e-mail attributes (TO, CC, BCC, SUBJECT, ATTACH) or perform actions (such as SEND, ABORT, and start a NEWMSG)

Syntax of the FILENAME Statement for Electronic Mail

To send electronic mail from a DATA step or SCL, issue a FILENAME statement of the following form:

FILENAME *fileref* EMAIL '*address*' <*email-options*>;

The FILENAME statement accepts the following options:

fileref
: is a valid fileref.

'*address*'
: is the destination e-mail address of the user to which you want to send e-mail. You must specify an address here, but you can override its value with the TO e-mail option.

email-options
: can be any of the following:

  TO=*to-address*
  : specifies the primary recipients of the electronic mail. If an address contains more than one word, enclose it in quotation marks. To specify more than one address, enclose the group of addresses in parentheses, enclose each address in quotation marks, and separate each address with a space. For example, **`to='joe@someplace.org'`** and **`to=("joe@smplc.org" "jane@diffplc.org")`** are valid TO values.

    Note: You can send an e-mail without specifying a recipient in the TO= option as long as you specify a recipient in either the CC= or BCC= option.

  CC=*cc-address*
  : specifies the recipients that you want to receive a copy of the electronic mail. If an address contains more than one word, enclose it in quotation marks. To specify more than one address, enclose the group of addresses in parentheses, enclose each address in quotation marks, and separate each address with a space. For example, **`cc='joe@someplace.org'`** and **`cc=("joe@smplc.org" "jane@diffplc.org")`** are valid CC values.

  BCC=*bcc-address*
  : specifies the recipients that you want to receive a blind copy of the electronic mail. Individuals listed in the **`bcc`** field will receive a copy of the e-mail. The BCC field does not appear in the e-mail header, so that these e-mail addresses cannot be viewed by other recipients.

    If a BCC address contains more than one word, enclose it in quotation marks. To specify more than one address, enclose the group of addresses in parentheses, enclose each address in quotation marks, and separate each address with a space. For example, **`bcc='joe@someplace.org'`** and **`bcc=("joe@smplc.org" "jane@diffplc.org")`** are valid BCC values.

  SUBJECT='*subject*'
  : specifies the subject of the message. If the subject text is longer than one word (that is, it contains at least one blank space), you must enclose it in quotation marks. You also must use quotation marks if the subject contains any special characters. For example, **`subject=Sales`** and **`subject='June Report'`** are valid subjects. Any subject not enclosed in quotation marks is converted to uppercase.

  ATTACH='*filename.ext*' | ATTACH = ('*filename.ext*' <*attachment-options*>)
  : specifies the physical name of the files to be attached to the message and any options to modify attachment specifications. Enclose *filename.ext* in quotation marks. To attach more than one file, enclose the group of filenames in parentheses. For example, **`attach='/u/userid/opinion.txt'`** and **`attach=("june11.txt" "july11.txt")`** are valid file attachments.

    By default, SMTP e-mail attachments are truncated at 256 characters. To send longer attachments, you can specify the LRECL= and RECFM= options from the FILENAME statement as the *attachment-options*. For more information about the LRECL= and RECFM= options, see “FILENAME Statement: UNIX” on page 323.

For more information about the options that are valid when you are using SMTP, see “FILENAME Statement, EMAIL (SMTP) Access Method” in *SAS Statements: Reference*.

Specifying E-mail Options in the FILE Statement

You can also specify the *email-options* in the FILE statement inside the DATA step. Options that you specify in the FILE statement override any corresponding options that you specified in the FILENAME statement.

Defining the Body of the Message

In your DATA step, after using the FILE statement to define your e-mail fileref as the output destination, use PUT statements to define the body of the message.

Specifying E-mail Directives in the PUT Statement

You can also use PUT statements to specify e-mail directives that change the attributes of your electronic message or perform actions with it. Specify only one directive in each PUT statement; each PUT statement can contain only the text associated with the directive that it specifies.

The following are the directives that change the attributes of your message:

!EM_TO!*addresses*
: Replace the current primary recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotation marks.

!EM_CC!*addresses*
: Replace the current copied recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotation marks.

!EM_BCC!*addresses*
: Replace the current blind copied recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotation marks.

!EM_SUBJECT!*subject*
: Replace the current subject of the message with *subject*.

!EM_ATTACH!*pathname*
: Replace the names of any attached files with *pathname*.

The following are the directives that perform actions:

!EM_SEND!
: Sends the message with the current attributes. By default, SAS sends a message when the fileref is closed. The fileref closes when the next FILE statement is encountered or the DATA step ends. If you use this directive, SAS sends the message when it encounters the directive, and again at the end of the DATA step.

!EM_ABORT!
: Aborts the current message. You can use this directive to stop SAS from automatically sending the message at the end of the DATA step.

!EM_NEWMSG!
: Clears all attributes of the current message, including TO, CC, SUBJECT, ATTACH, and the message body.

Example: Sending E-mail from the DATA Step

Suppose that you want to share a copy of your config.sas file with your coworker Jim, whose user ID is JBrown. If your e-mail program handles alias names and attachments, you could send it by submitting the following DATA step:

```
filename mymail email 'JBrown'
         subject='My CONFIG.SAS file'
         attach='config.sas';

data _null_;
  file mymail;
  put 'Jim,';
  put 'This is my CONFIG.SAS file.';
  put 'I think you might like the
       new options I added.';
run;
```

The following example sends a message and two attached files to multiple recipients. It specifies the e-mail options in the FILE statement instead of the FILENAME statement:

```
filename outbox email 'ron@acme.com';

data _null_;
  file outbox

        /* Overrides value in filename statement */
     to=('ron@acme.com' 'lisa@acme.com')
     cc=('margaret@yourcomp.com'
         'lenny@laverne.abc.com')
     subject='My SAS output'
     attach=('results.out' 'code.sas')
     ;
  put 'Folks,';
  put 'Attached is my output from the
      SAS program I ran last night.';
  put 'It worked great!';
run;
```

You can use conditional logic in the DATA step to send multiple messages and control which recipients get which message. For example, suppose you want to send customized reports to members of two different departments. If your e-mail program handles alias names and attachments, your DATA step might look like the following:

```
filename reports email 'Jim';

data _null_;
  file reports;
  infile cards eof=lastobs;
  length name dept $ 21;
  input name dept;

     /* Assign the TO attribute      */
  put '!EM_TO!' name;

     /* Assign the SUBJECT attribute */
  put '!EM_SUBJECT! Report for ' dept;
  put name ',';
  put 'Here is the latest report for ' dept '.';

     /* ATTACH the appropriate report */
  if dept='marketing' then
      put '!EM_ATTACH! mktrept.txt';
  else

    put '!EM_ATTACH! devrept.txt';

     /* Send the message */
  put '!EM_SEND!';

    /* Clear the message attributes */
  put '!EM_NEWMSG!';

  return;

   /* Abort the message before the */
   /*   RUN statement causes it to */
   /*   be sent again.           */
lastobs: put '!EM_ABORT!';

  datalines;
Susan          marketing
Jim            marketing
Rita           development
Herb           development
;
run;
```

The resulting e-mail message and its attachments are dependent on the department to which the recipient belongs.

Note: You must use the !EM_NEWMSG! directive to clear the message attributes between recipients. The !EM_ABORT! directive prevents the message from being automatically sent at the end of the DATA step.

Example: Sending E-mail Using SCL Code

The following example is the SCL code behind a frame entry design for e-mail. The frame entry includes several text entry fields that let the user enter information:

mailto
the user ID to send mail to

copyto
the user ID to copy (CC) the mail to

attach
the name of a file to attach

subject
the subject of the mail

line1
the text of the message

The frame entry also contains a button named SEND that causes this SCL code (marked by the **`send:`** label) to execute.

```
send:

   /* set up a fileref */
rc = filename('mailit','userid','email');

   /* if the fileref was successfully set up
      open the file to write to */
if rc = 0 then do;
      fid = fopen('mailit','o');
      if fid > 0 then do;

         /* fput statements are used to
            implement writing the
            mail and the components such as
            subject, who to mail to, and so on. */
      fputrc1  = fput(fid,line1);
      rc = fwrite(fid);

      fputrc2  = fput(fid,'!EM_TO! '||mailto);
      rc = fwrite(fid);
      fputrc3  = fput(fid,'!EM_CC! '||copyto);
      rc = fwrite(fid);

      fputrc4  = fput(fid,'!EM_ATTACH! '||attach);
      rc = fwrite(fid);
      fputrc5  = fput(fid,'!EM_SUBJECT! '||subject);
      rc = fwrite(fid);

         closerc  = fclose(fid);
      end;
   end;
return;

cancel:
   call execcmd('end');
return;
```

Chapter 4

Printing and Routing Output

Overview of Printing Output in UNIX Environments

When you print text or graphics, SAS needs to know where the output should go, how it should be written, and how the output should look. Universal Printing is the default printing mechanism in UNIX. Universal Printing supports PostScript, PCL, GIF, PNG, SVG, EMF, and PDF files in all environments. For more information about Universal Printing, see “Universal Printing” in Chapter 15 of *SAS Language Reference: Concepts*.

Forms printing is an older method of text printing available from SAS. It involves using the FORM subsystem, which consists of the Form window. For information, see “Forms Printing” in Chapter 15 of *SAS Language Reference: Concepts*.

If you are printing graphics, the output is controlled by native SAS/GRAPH drivers. See the online Help for SAS/GRAPH for information about native SAS/GRAPH drivers.

Previewing Output in UNIX Environments

Previewing Output Using Universal Printing

With Universal Printing, you can preview your output before you send it to a printer, plotter, or external file. To preview your output, you first need to define a previewer for your system. For more information, see “Universal Printing” in Chapter 15 of *SAS Language Reference: Concepts*.

Previewing Output from within SAS/AF Applications

To preview output from within a SAS/AF application, use the DMPRTMODE and DMPRTPREVIEW commands to turn on preview mode, print the output, open the Print Preview dialog box, and then turn preview mode off. For example, the following code prints the GRAPH1 object using the host drivers and displays it in the Preview dialog box:

```
/* Turn on preview mode. */
CALL EXECCMDI ("DMPRTMODE PREVIEW");

/* Print the graph */
GRAPH1._PRINT_();

/* Open the Preview dialog box */
CALL EXECCMDI ("DMPRTPREVIEW");

/* Turn off preview mode */
CALL EXECCMDI ("DMPRTMODE NORMAL");
```

The Default Routings for the SAS Log and Procedure Output in UNIX Environments

For each SAS job or session, SAS automatically creates two types of output:

SAS log
: contains information about the processing of SAS statements. As each program step executes, notes are written to the SAS log along with any applicable error or warning messages.

SAS output
: is also called the procedure output file or print file. Whenever a SAS program executes a PROC step or a DATA step that produces printed output, SAS sends the output to the SAS output file. The default destination for SAS output is HTML.

The following table shows the default routings of the SAS log and output files.

Table 4.1 *Default Routings of the SAS Log and Output Files*

Processing Mode	SAS Log File	SAS Output File
batch	`*filename*.log`	`*filename*.lst`
windowing environment	Log window	HTML
interactive line	terminal	terminal

By default, both the log file and the output file are written to your current directory. Your system administrator might have changed these default routings.

Changing the Default Routings in UNIX Environments

Techniques for Routing Output

There are five primary methods for routing your output.:

- Using the default HTML destination.
- Using the Print dialog box. The Print dialog box is available when you are using the SAS windowing environment.
- Issuing windowing environment commands. The PRTFILE, PRINT, and FILE commands can be issued from any command line and can be used to send output to external files or to other devices defined with the FILENAME statement.
- Using the PRINTTO procedure. You can use the PRINTTO procedure in any mode. Using the FILENAME statement with the PRINTTO procedure is the most flexible way of routing your output.

- Using SAS system options, such as PRINT, LOG, ALTPRINT, or ALTLOG, to specify alternate destinations.

Determining Which Technique to Use When Changing the Routing

Use the following table to help you decide which method you should choose to change the routing.

Table 4.2 *Decision Table: Changing the Default Destination*

Output destination for your SAS log or procedure output	Processing mode	Method	See
a printer	any mode	FILENAME statement (UPRINTER or PRINTER device type) and PRINTTO procedure	"Using the PRINTTO Procedure in UNIX Environments" on page 100
	windowing environment	DMPRINT command	"Using the Print Dialog Box in UNIX Environments" on page 96
		Print dialog box	"Using the Print Dialog Box in UNIX Environments" on page 96
		FILENAME statement and PRTFILE, PRINT, and FILE commands	"Printing the Contents of a Window" on page 98
an external file	any mode	PRINTTO procedure and FILENAME statement	"Using the PRINTTO Procedure in UNIX Environments" on page 100
	windowing environment	Print dialog box	"Using the Print Dialog Box in UNIX Environments" on page 96
		FILENAME statement and PRTFILE, PRINT, and FILE commands	"Printing the Contents of a Window" on page 98
	batch	LOG and PRINT system options	"Using SAS System Options to Route Output" on page 102
a UNIX command (pipe)	any mode	FILENAME statement and PRINTTO procedure	"Using the PRINTTO Procedure in UNIX Environments" on page 100
	windowing environment	FILENAME statement and PRTFILE and PRINT commands	"Printing the Contents of a Window" on page 98
its usual location and to an external file	any mode	ALTLOG and ALTPRINT system options	"Using SAS System Options to Route Output" on page 102
	windowing environment	FILE command	"Using the FILE Command " on page 100

Output destination for your SAS log or procedure output	Processing mode	Method	See
		Print dialog box	"Using the Print Dialog Box in UNIX Environments" on page 96
a terminal	batch	FILENAME statement and PRINTTO procedure	"Routing Output to a Terminal " on page 102

Routing SAS Logging Facility Messages to SYSLOGD

The SAS logging facility enables the categorization and collection of log event messages, and then writes them to a variety of output devices. The logging facility supports problem diagnosis and resolution, performance and capacity management, and auditing and regulatory compliance. The following features are provided:

- Log events are categorized using a hierarchical naming system that enables you to configure logging at a broad or a fine-grained level.
- Log events can be directed to multiple output destinations, including files, operating system facilities, databases, and client applications. For each output destination, you can specify:
  - the categories and levels of log events to report
  - the message layout, including the types of data to be included, the order of the data, and the format of the data
  - filters based on criteria such as diagnostic levels and message content
- Logging diagnostic levels can be adjusted dynamically without starting and stopping processes.
- Performance-related log events can be generated for processing by an Application Response Measurement (ARM) 4.0 server.

The logging facility is used by most SAS server processes. You can also use the logging facility within SAS programs.

In the UNIX operating environment, logging facility messages can be written to SYSLOGD.

For information about using the logging facility in the UNIX operating environment, see the *SAS Logging: Configuration and Programming Reference*.

Using the Print Dialog Box in UNIX Environments

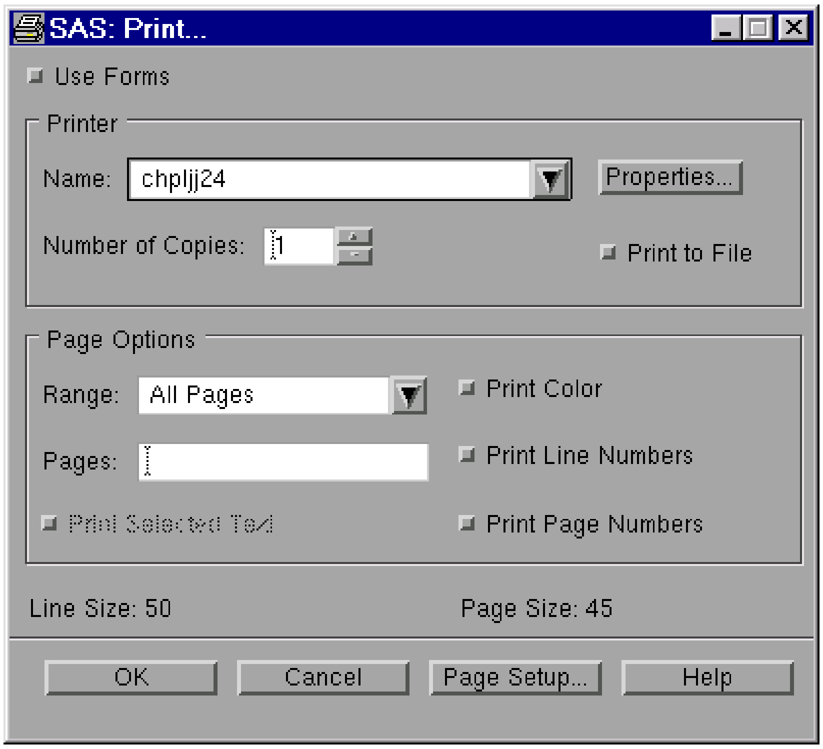
Printing from Text Windows

Open the Print Dialog Box from a Text Window

To print part or all of the contents of a window, complete the following steps:

1. Click in the window to make it the active window. If you want to mark and print only selected lines of text, mark the text before you open the Print dialog box.
2. Issue the DMPRINT command or select **File** ⇨ **Print** to open the Print dialog box.

Display 4.1 *Print Dialog Box*


SAS: Print...

Use Forms

Printer

Name: chpljj24

Number of Copies: 1

Print to File

Page Options

Range: All Pages

Pages:

Print Selected Text

Print Color

Print Line Numbers

Print Page Numbers

Line Size: 50

Page Size: 45

Default Printing Mode

In UNIX, the default printing mode is Universal Printing. For more information about how to use Universal Printing, click **Help** on the Print dialog box.

Specifics for Forms Printing

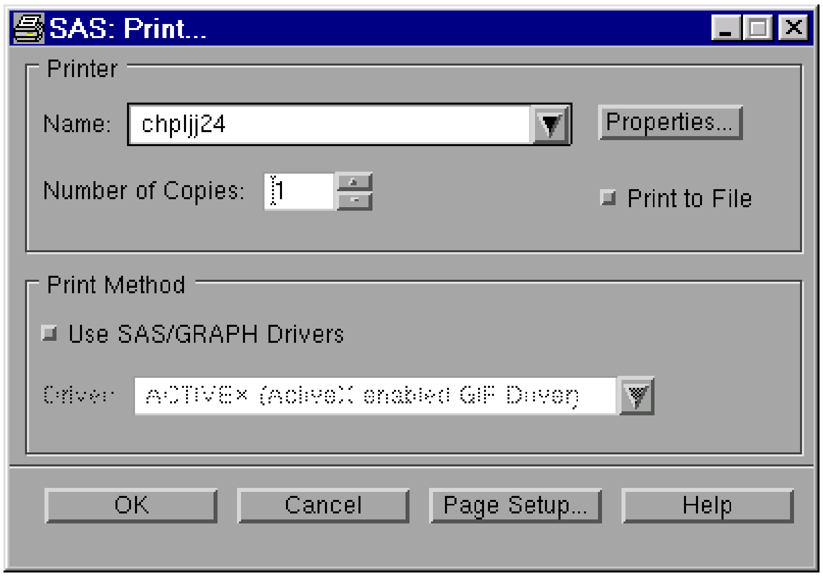
To use forms for printing, select **Use forms**. SAS prompts you to enter a spool command and the name of your system printer. When you click **OK**, SAS prints the contents of the active window using the command and printer name that you specified and additional information from your default form. For more information about forms printing, see "Forms Printing" in Chapter 15 of *SAS Language Reference: Concepts*.

Troubleshooting Print Server Errors
After clicking **OK**, if SAS displays a clock icon for a long time and you are sending output to a network printer, your printer server might be down. If so, you will eventually see a message in the shell where you invoked your SAS session that indicates that the server is down.

Printing from GRAPH Windows

Open the Print Dialog Box from the GRAPH Window
With Universal Printing, you can use the Print dialog box to print the contents of a SAS/GRAPH window. Click in the window to make it the active window, and then issue the DMPRINT command or select **File** ⇨ **Print** to open the Print dialog box.

Display 4.2 *Print Dialog Box for Graphs*


SAS: Print...

Printer

Name: chpljj24

Number of Copies: 1

Print to File

Print Method

Use SAS/GRAPH Drivers

Driver: ACTIVEX (ActiveX enabled GIF Driver)

Note: In most cases, fonts set through the Print dialog box have no effect when you print from GRAPH windows. However, some SAS/GRAPH drivers use Universal Printing and can be affected by the fonts set in the dialog box. Make sure that you specify the correct options on a GOPTIONS statement.

Specifics for SAS/GRAPH Drivers
To print output using a SAS/GRAPH driver, select **Use SAS/GRAPH Drivers**. Select the down arrow beside the **Driver** field to display the available drivers. Make sure that your printer destination has been set inside the device using the GDEVICE procedure or the GOPTIONS statement. For complete information about printing from GRAPH windows, refer to *SAS/GRAPH: Reference* and the online Help for SAS/GRAPH.

Troubleshooting Print Server Errors
After clicking **OK**, if SAS displays a clock icon for a long time and you are sending output to a network printer, your printer server might be down. If so, you will eventually see a message in the shell where you invoked your SAS session that indicates that the server is down.

Using Commands to Print in UNIX Environments

Differences between the PRTFILE, PRINT, and FILE Commands

In the SAS windowing environment, you can use the PRTFILE, PRINT, and FILE commands to send the contents of the active window to an output device.

The following table lists the results of each of these commands.

Table 4.3 *Routing Output Commands*

Command	Action Performed
PRTFILE	specifies the filename or fileref for your output.
FILE	sends the contents of the active window to the filename or fileref that you specify.
PRINT	sends the contents of the active window either: • to your default printer when issued from the command line of the window. • to the location specified with the PRTFILE command.

Sending Output to a UNIX Command

If you want to send your output to a UNIX command, you can use the FILENAME statement. The FILENAME statement enables you to create filerefs that point to printers, plotters, or external files or filerefs that pipe to a UNIX command. For more information, see "FILENAME Statement: UNIX" on page 323.

Specifying the Print File

When you issue the PRINT command, SAS sends your output to your default printer, unless you specify a print file. You can specify a print file by entering the PRTFILE command (for example, **`PRTFILE`** ***`file-spec`*** **`CLEAR | APPEND | REPLACE)`**. The *file-spec* argument can be either a fileref or a filename.

If you are using forms printing, and you select **File** ⇨ **Print**, a window appears that enables you to select file options. When you select **Print to File**, the Save As window appears. Enter the location to which you want your file saved. This option is available only when Universal Printing is turned off.

Printing the Contents of a Window

Using PRTFILE and PRINT with a Fileref

You can use the PRTFILE command, followed by the PRINT command, to print the contents of windows. PRTFILE establishes the destination, and PRINT sends the contents of the window to that destination. If you do not specify a destination with the

PRTFILE command, PRINT automatically sends the window contents to your default printer.

Steps for Sending Output Directly to a Printer

If you want to send output directly to a printer, you must first submit the FILENAME statement to assign a fileref to the PRINTER or PIPE device. For example, to print the contents of your Output window, complete the following steps:

Table 4.4 *Printing the Contents of Your Output Window*

Step	Action	Example

1	Submit a FILENAME statement or FILENAME function to associate a fileref with a system printer (PRINTER device type) or a UNIX command (PIPE device type). Enclose the printer name or UNIX command in either single or double quotation marks.	`filename myrpt printer 'bldga2';`
		or
		`filename ascout pipe 'lp -dmyljet';`
		For more information, see "Examples of FILENAME Statements Using PRINTER and PIPE " on page 99.

2	Issue the PRTFILE command as described in "Specifying the Print File" on page 98 . Specify the fileref from your FILENAME statement or FILENAME function.	`prtfile myrpt`
3	Issue the PRINT command from the command line of the windows whose contents you want to print. If you are sending output to a system printer or if you are using forms-based printing, then you can print the contents of more than one window.	
4	Enter **A** in the dialog box that warns you that the destination file already exists. The **A** value tells SAS to append the window contents to the destination file.	
5	Submit a FILENAME statement or FILENAME function to clear (deassign) the fileref.	`filename myrpt clear;`

To clear the print file setting, issue the PRTFILE CLEAR command.

Examples of FILENAME Statements Using PRINTER and PIPE

The following statement associates MyRpt with the system printer named BldgA2 and specifies two copies of every printout:

```
filename myrpt printer 'bldga2 -n2';
```

(See the documentation for your print command for information about other options that you can specify.)

The following statement enables you to print output using the **`lp`** command on the printer named myljet:

```
filename ascout pipe 'lp -dmyljet';
```

The following statement sends output to the **lp** command and redirects any error messages produced by this command to the LpError file in your home directory:

```
filename myrpt pipe 'lp 2>$HOME/lperror';
```

Note: Redirecting standard error is allowed only in the Bourne and Korn shells.

If you frequently use the same print command and destination, you can add the appropriate FILENAME statement to your autoexec file. For more information, see "Customizing Your SAS Session by Using System Options" on page 18.

Using the FILE Command

You can use the FILE command to copy the contents of many different windows to external files. Issue the FILE command on the command line of the window whose contents you want to copy. For example, to copy the contents of the Log window to **/u/myid/log/app1**, issue the following command on the command line of the Log window:

```
file '/u/myid/log/app1'
```

If the file does not exist, SAS creates it. If the file already exists, a dialog box asks you whether you want to replace it or to append data to the existing data.

If you have already associated a fileref with your external file, you can use the fileref instead of the filename:

```
file myref
```

If you use the FILE command to save your output, carriage-control information is not saved (that is, page breaks are removed from the output). You might want to use the PRINT command with the FILE option instead:

PRINT FILE=*fileref* | '*pathname*'

Using the PRINTTO Procedure in UNIX Environments

Important Note about the PRINTTO Procedure

Any time you use PROC PRINTTO to route output, you must close the output device before PROC PRINTTO will release the output or log and send it to the destination that you have specified. To close the output device, issue PROC PRINTTO without any parameters:

```
proc printto;
run;
```

Issuing PROC PRINTTO without any parameters closes the output device, generates output, and reroutes the log and procedure output to their default destinations. For a list of the default destinations, see Table 4.1 on page 93.

For more information, see "PRINTTO Procedure: UNIX" on page 307 and Chapter 38, "PRINTTO Procedure" in *Base SAS Procedures Guide*.

Using the LOG= and PRINT= Options

When you use the PRINTTO procedure with its LOG= and PRINT= options, you can route the SAS log or SAS procedure output to an external file or a fileref from any mode. Specify the external file or the fileref in the PROC PRINTTO statement. The following example routes procedure output to **`/u/myid/output/prog1`**:

```
proc printto print='/u/myid/output/prog1' new;
run;
```

The NEW option causes any existing information in the file to be cleared. If you omit the NEW option from the PROC PRINTTO statement, the SAS log or procedure output is appended to the existing file.

If you plan to specify the same destination several times in your SAS program, you can assign a fileref to the file using a FILENAME statement. (For information and examples, see "Assigning Filerefs to External Files or Devices with the FILENAME Statement" on page 74.)

Routing Output to a Universal Printer

You can direct output directly to your Universal Printer by using the UPRINTER device type:

```
filename myoutput uprinter;
proc printto print=myoutput;
run;
```

Output will be sent to your default Universal Printer. This output will be in PostScript or PCL format.

Routing Output to a Printer

You can direct output directly to your system printer by using the PRINTER device type:

```
filename myoutput printer;
proc printto print=myoutput;
run;
```

Output will be sent to your default system printer or, if you have specified the SYSPRINT system option, to the printer specified with that option. This method will produce output in ASCII format.

Piping Output to a UNIX Command

You can also use the PIPE device type to send output to a UNIX command. When you specify the print command, you might also want to specify a destination for any error messages that are produced by the print command. Enclose the UNIX command in either single or double quotation marks. The following example associates the fileref MyOutput with the print command **`lp`**, which will send output to the printer named myljet:

```
filename myoutput pipe 'lp -dmyljet';
proc printto print=myoutput;
run;
```

You can send the SAS log to the same printer by using the LOG= option:

```
filename mylog pipe 'lp -dmyljet';
proc printto log=mylog;
run;
```

The log and procedure output continue to be routed to the designated external file until another PROC PRINTTO statement reroutes them.

Routing Output to a Terminal

In batch mode, you can direct output to a terminal by associating a fileref with a terminal and then using PROC PRINTTO to send output to that fileref. In the FILENAME statement, specify the TERMINAL device-type and the special file associated with the terminal. For example, the following statements send the SAS log to the terminal that is associated with the **`/dev/tty3`** special file:

```
filename term terminal '/dev/tty3';
proc printto log=term;
run;
```

Using SAS System Options to Route Output

Changing the Output Destination Using the LOG, PRINT, ALTLOG, and ALTPRINT System Options

You can use SAS system options to change the destination of the SAS log and procedure output. The options that you use depend on which task you want to accomplish:

- To route your SAS log or procedure output to an external file instead of to their default destinations, use the LOG and PRINT system options.
- To route the log or output to an external file in addition to their default destinations, use the ALTLOG and ALTPRINT system options. This method works in all modes of running SAS.

LOG and PRINT are normally used in batch and interactive line modes. These system options have no effect in the windowing environment. If you are running in the windowing environment, use the ALTLOG and ALTPRINT system options.

You can specify these options in the following locations:

- the SAS command
- a configuration file
- the SASV9_OPTIONS environment variable

For example, you could specify these options in the SAS command as follows:

```
sas -log '/u/myid/log' -print '/u/myid/prt'
sas -altlog '/u/myid/log' -altprint '/u/myid/prt'
```

For more information, see “Ways to Specify a SAS System Option” on page 18.

Printing Large Files with the PIPE Device Type in UNIX Environments

When you print a file with the **lp** command, a symbolic link is created from the file to the **/usr/spool** directory. When you pipe output to the **lp** command, the output is copied under the **/usr/spool** directory.

If you experience problems printing large files using the PIPE device type, you can circumvent the problem in either of the following ways:

- save the print file to a disk file and then print it with the **lp** command. Issue the PRINT command from the Output or Log window:

  ```
  print file='bigfile'
  ```

  Exit your SAS session and print the file, or use the SAS X command to print the file from within your SAS session:

  ```
  x 'lp -dmylsrjt bigfile'
  ```

- create a fileref using the PIPE device type that can handle large files. For example, the following fileref saves the print file to disk, prints the saved file, and then removes the file:

  ```
  filename myfile pipe 'cat >bigfile;lp -dmylsrlt bigfile;rm bigfile;';
  ```

Changing the Default Print Destination in UNIX Environments

When you print a file, SAS looks in the following locations to determine where to send output. The locations are listed in order of precedence:

1. The destination specified in Universal Printing or the form printer device that you are using. See Universal Printing or forms printing in *SAS Language Reference: Concepts* for more information.
2. The value specified in the SYSPRINT system option. You can use the SYSPRINT option to set your default print destination. Use the SYSPRINT system option to specify the destination option that is used with your print command. For example, if your print command is **lp**, you can set the default destination to the printer named myljet by entering the following OPTIONS statement:

   ```
   options sysprint='-dmyljet';
   ```

3. The value of the $LPDEST environment variable. For more information, see “Defining Environment Variables in UNIX Environments” on page 24 .

SAS uses the first destination that it finds. If you specify a destination in all three locations, SAS uses the destination specified by Universal Printing.

Changing the Default Print Command in UNIX Environments

UNIX uses **`lp`** as the default print command. You can use the PRINTCMD system option to specify a different print command. For example, you can change your default print command to **`lpr`** by entering the following at SAS invocation:

```
sas -printcmd "lpr"
```

You can also customize your default print command in your SAS configuration file. If you use this method, then you will not have to change the default print command every time you invoke SAS. For more information, see "PRINTCMD System Option: UNIX" on page 409.

Controlling the Content and Appearance of Output in UNIX Environments

Overview of Controlling the Content and Appearance of Output

Some of the attributes of the SAS log and procedure output depend on the destination to which they are being sent. For example, if the log and output are being sent to your display, the default line and page size are derived from your display. If one or both of these files are sent to the system printer or written to a file, the default line size and page size depend on your printer and page setup. The line size and page size for your current settings can be seen in the Print dialog box.

Some of the attributes of the SAS log and procedure output depend on the mode in which you are running. For example, if you are running in interactive line mode, SAS source statements are not echoed to the SAS log. If you are using the SAS windowing environment all source statements are written to the log as they are submitted. In batch mode, the log and procedure output are formatted for a standard system printer.

For more information about specifying system options, see "Customizing Your SAS Session by Using System Options" on page 18.

SAS Log Options

Use the following options to control the contents of the log. For more information about specifying options, see "SAS System Options under UNIX" on page 348.

FULLSTIMER
NOFULLSTIMER
: controls whether a list of resources (such as I/O performed, page faults, elapsed time, and CPU time) used for each PROC or DATA step is written to the log. NOFULLSTIMER is the default.

LINESIZE=*width*
: controls the line length used. *Width* can be any value from 64 to 256.

NEWS
NONEWS
: controls whether messages are written to the SAS log. NEWS is the default.

NOTES
NONOTES
: controls printing of NOTES on the log. NOTES is the default setting for all execution modes. Specify NOTES unless your SAS program is completely debugged.

PAGESIZE=*n*
: controls the number of lines that are printed on each page. *N* can be any number from 15 to 32767.

SOURCE
NOSOURCE
: controls whether SAS source statements are written to the log. NOSOURCE is the default setting in interactive line mode. Otherwise, SOURCE is the default.

SOURCE2
NOSOURCE2
: controls whether SAS statements that are included with %INCLUDE statements are written to the log. NOSOURCE2 is the default setting for all execution modes.

STIMER
NOSTIMER
: controls whether user CPU time and elapsed time are written to the log. STIMER is the default.

Procedure Output Options

Use these system options to control the contents of the procedure output for the LISTING destination:

CENTER
NOCENTER
: controls whether the printed results are centered or left-aligned on the procedure output page. CENTER is the default.

DATE
NODATE
: controls whether the date is written at the top of each procedure output page. DATE is the default.

LINESIZE=*width*
: controls the line length used. *Width* can be any value from 64 to 256.

NUMBER
NONUMBER
: controls whether the output page number is written on each procedure output page. NUMBER is the default.

PAGENO=*n*
: resets the current page number in the print file. The default page number at the beginning of the SAS session is 1. The pages are numbered sequentially throughout the SAS session unless the PAGENO option is specified in an OPTIONS statement during the session.

PAGESIZE=*n*
: controls the number of lines that are printed on each page. *N* can be any number from 15 to 32,767.

Chapter 5
Accessing Shared Executable Libraries from SAS

Overview of Shared Libraries in SAS

What Is a Shared Library?

Shared libraries in UNIX are libraries that contain executable programs that are written in any of several programming languages. In UNIX, the names of these programs typically end with a .so or .sl extension. However, they are not constrained to this naming convention.

Shared libraries are a mechanism for storing useful routines that might be needed by multiple applications. When an application needs a routine that resides in an external shared library, it loads the shared library, invokes the routine, and unloads the shared library upon completion.

Invoking Shared Libraries from within SAS

SAS provides routines and functions that let you invoke these external routines from within SAS. You can access the shared library routines from the DATA step, the IML procedure, and SCL code. You use the MODULE family of SAS CALL routines and functions (including MODULE, MODULEN, and MODULEC), as well as the SAS/IML functions and CALL routines (including MODULEIC, MODULEIN, and MODULEI), to invoke a routine that resides in a shared library. This documentation refers to the MODULE family of CALL routines and functions generically as the MODULE function.

For information about MODULE, MODULEN, and MODULEC, see the *SAS Functions and CALL Routines: Reference*. For information about MODULEIC, MODULEIN, and MODULEI, see the *SAS/IML User's Guide*.

Steps for Accessing an External Shared Library

Use the following steps to access an external shared library routine:

1. Create a text file that describes the shared library routine that you want to access, including the arguments that it expects and the values that it returns (if any). This attribute file must be in a special format, as described in “The SASCBTBL Attribute Table” on page 108.
2. Use the FILENAME statement to assign the SASCBTBL fileref to the attribute file that you created.
3. In a DATA step or SCL code, use a CALL routine or function (MODULE, MODULEN, or MODULEC) to invoke the shared library routine. The specific function that you use depends on the type of expected return value (none, numeric, or character). (You can also use MODULEI, MODULEIN, or MODULEIC within a PROC IML step.) The MODULE functions are described in “CALL MODULE Routine: UNIX” on page 260.

CAUTION:
Only experienced programmers should access external routines in shared libraries. By accessing a function in a shared library, you transfer processing control to the external function. If done improperly, or if the external function is not reliable, you might lose data, get unreliable results, or receive severe errors.

The SASCBTBL Attribute Table

Introduction to the SASCBTBL Attribute Table

Because the MODULE function invokes an external routine that SAS knows nothing about, you must supply information about the routine's arguments so that the MODULE function can validate them and convert them, if necessary. For example, suppose you want to invoke a routine that requires an integer as an argument. Because SAS uses floating-point values for all of its numeric arguments, the floating-point value must be converted to an integer before you invoke the external routine. The MODULE function looks for this attribute information in an attribute table that is referred to by the SASCBTBL fileref.

What Is the SASCBTBL Attribute Table?

The attribute table is a sequential text file that contains descriptions of the routines that you can invoke with the MODULE function. The table defines how the MODULE function should interpret supplied arguments when it builds a parameter list to pass to the called routine.

The MODULE function locates the table by opening the file that is referenced by the SASCBTBL fileref. If you do not define this fileref, the MODULE function simply calls the requested shared library routine without altering the arguments.

CAUTION:

Using the MODULE function without defining an attribute table can cause SAS to crash, produce unexpected results, or result in severe errors. You need to use an attribute table for all external functions that you want to invoke.

Syntax of the Attribute Table

The Attribute Table

The attribute table should contain the following items:

- a description in a ROUTINE statement for each shared library routine that you intend to call
- descriptions in ARG statements for each argument that is associated with the routine you intend to call

At any point in the attribute table file, you can create a comment using an asterisk (*) as the first non-blank character of a line or after the end of a statement (following the semicolon). You must end the comment with a semicolon.

ROUTINE Statement

Here is the syntax of the ROUTINE statement:

ROUTINE *name* MINARG=*minarg* MAXARG=*maxarg*
<CALLSEQ=BYVALUE|BYADDR>
<TRANSPOSE=YES|NO> <MODULE=*shared-library-name*>
<RETURNS=DBLPTR | CHAR<*n*> | DOUBLE | LONG | PTR | SHORT | [U]INT32 | [U]INT64 | ULONG | USHORT>

The following are descriptions of the ROUTINE statement attributes:

ROUTINE *name*

starts the ROUTINE statement. You need a ROUTINE statement for every shared library function that you intend to call. The value for *name* must match the routine name or ordinal that you specified as part of the *module* argument in the MODULE function, where *module* is the name of the shared library (if not specified by the MODULE attribute) and the routine name or ordinal. For example, in order to specify **`libc,getcwd`** in the MODULE function call, the ROUTINE *name* should be **`getcwd`**.

The *name* argument is case sensitive, and is required for the ROUTINE statement.

MINARG=*minarg*
: specifies the minimum number of arguments to expect for the shared library routine. In most cases, this value will be the same as MAXARG; but some routines do allow a varying number of arguments. This attribute is required.

MAXARG=*maxarg*
: specifies the maximum number of arguments to expect for the shared library routine. This attribute is required.

CALLSEQ=BYVALUE | BYADDR
: indicates the calling sequence method used by the shared library routine. Specify BYVALUE for call-by-value and BYADDR for call-by-address. The default value is BYADDR.

  Fortran and COBOL are call-by-address languages. C is usually call-by-value, although a specific routine might be implemented as call-by-address.

  The MODULE function does not require that all arguments use the same calling method. You can identify any exceptions by using the BYVALUE and BYADDR options in the ARG statement.

TRANSPOSE=YES | NO
: specifies whether SAS transposes matrices that have both more than one row and more than one column before it calls the shared library routine. This attribute applies only to routines called from within PROC IML with MODULEI, MODULEIC, and MODULEIN.

  TRANSPOSE=YES is necessary when you are calling a routine that is written in a language that does not use row-major order to store matrices. (For example, Fortran uses column-major order.)

  For example, consider this matrix with three columns and two rows:

```
columns
                  1   2   3
             ------------
       rows  1 | 10  11  12
             2 | 13  14  15
```

  PROC IML stores this matrix in memory sequentially as 10, 11, 12, 13, 14, 15. However, Fortran routines will expect this matrix as 10, 13, 11, 14, 12, 15.

  The default value is NO.

MODULE=*shared-library-name*
: names the executable module (the shared library) in which the routine resides. You do not need to specify this attribute if the name of the shared library is the same name as the routine. If you specify the MODULE attribute here in the ROUTINE statement, then you do not need to include the module name in the *module* argument of the MODULE routine (unless the shared library routine name that you are calling is not unique in the attribute table). The MODULE routine is described in "CALL MODULE Routine: UNIX" on page 260.

  You can have multiple ROUTINE statements that use the same MODULE name. You can also have duplicate routine names that reside in different shared libraries.

  The MODULE function searches the directories that are defined in each operating system's library path environment variable when it attempts to load the shared library argument provided in the MODULE attribute. The following table lists this environment variable for each UNIX operating system that SAS supports.

Table 5.1 *Shared Library Environment Variable Name*

Operating Environment	Environment Variable Name
Solaris	$LD_LIBRARY_PATH
AIX/R	$LIBPATH
HP-UX	$LD_LIBRARY_PATH or $SHLIB_PATH
Linux	$LD_LIBRARY_PATH

Note: For more information about these environment variables, see the man pages for your operating environment.

You can also use the PATH system option to point to the directory that contains the shared library specified in the MODULE= option. Using the PATH system option overrides your system's environment variable when you load the shared library. For more information, see "PATH System Option: UNIX" on page 408.

RETURNS=DBLPTR | CHAR<*n*> | DOUBLE | LONG | PTR | SHORT | [U]INT32 | [U]INT64 | ULONG | USHORT

specifies the type of value that the shared library routine returns. This value will be converted as appropriate, depending on whether you use MODULEC (which returns a character) or MODULEN (which returns a number). The following are the possible return value types:

DBLPTR
: pointer to a double-precision floating point number (instead of using a floating-point register). See the documentation for your shared library routine to determine how it handles double-precision floating-point values.

CHAR<*n*>
: pointer to a character string up to *n* bytes long. The string is expected to be null-terminated and will be blank-padded or truncated as appropriate. If you do not specify *n*, the MODULE function uses the maximum length of the receiving SAS character variable.

DOUBLE
: double-precision floating-point number.

LONG
: long integer.

PTR
: character string being returned.

SHORT
: short integer.

[U]INT32
: 32–bit unsigned integer.

[U]INT64
: 64-bit unsigned integer.

ULONG
: unsigned long integer.

USHORT
: unsigned short integer.

If you do not specify the RETURNS attribute, you should invoke the routine with only the MODULE and MODULEI CALL routines. You will get unpredictable values if you omit the RETURNS attribute and invoke the routine using the MODULEN and MODULEIN functions or the MODULEC and MODULEIC functions.

ARG Statement

The ROUTINE statement must be followed by as many ARG statements as you specified in the MAXARG= option. The ARG statements must appear in the order in which the arguments will be specified within the MODULE function.

Here is the syntax for each ARG statement:

ARG *argnum* NUM|CHAR <INPUT|OUTPUT|UPDATE> <NOTREQD|REQUIRED> <BYADDR|BYVALUE> <FDSTART> <FORMAT=*format*>;

Here are the descriptions of the ARG statement attributes:

ARG *argnum*
: defines the argument number. This a required attribute. Define the arguments in ascending order, starting with the first routine argument (ARG 1).

NUM | CHAR
: defines the argument as numeric or character. This attribute is required.

  If you specify NUM here but pass the routine a character argument, the argument is converted using the standard numeric informat. If you specify CHAR here but pass the routine a numeric argument, the argument is converted using the BEST12. informat.

INPUT | OUTPUT | UPDATE
: indicates the argument is either input to the routine, an output argument, or both. If you specify INPUT, the argument is converted and passed to the shared library routine. If you specify OUTPUT, the argument is not converted, but is updated with an outgoing value from the shared library routine. If you specify UPDATE, the argument is converted, passed to the shared library routine, and updated with an outgoing value from the routine.

  You can specify OUTPUT and UPDATE only with variable arguments (that is, no constants or expressions are allowed).

NOTREQD | REQUIRED
: indicates whether the argument is required. If you specify NOTREQD, then the MODULE function can omit the argument. If other arguments follow the omitted argument, identify the omitted argument by including an extra comma as a placeholder. For example, to omit the second argument to routine XYZ, you would specify:

```
call module('XYZ',1,,3);
```

  CAUTION:
  Be careful when using NOTREQD; the shared library routine must not attempt to access the argument if it is not supplied in the call to MODULE. If the routine does attempt to access it, you might receive unexpected results or severe errors.

  The REQUIRED attribute indicates that the argument is required and cannot be omitted. REQUIRED is the default value.

BYADDR | BYVALUE
: indicates whether the argument is passed by reference or by value.

BYADDR is the default value unless CALLSEQ=BYVALUE was specified in the ROUTINE statement. In that case, BYVALUE is the default. Specify BYADDR when you are using a call-by-value routine that also has arguments to be passed by address.

FDSTART
: indicates that the argument begins a block of values that are grouped into a structure whose pointer is passed as a single argument. Note that all subsequent arguments are treated as part of that structure until the MODULE function encounters another FDSTART argument.

FORMAT=*format*
: names the format that presents the argument to the shared library routine. Any formats supplied by SAS, PROC FORMAT style formats, or SAS/TOOLKIT formats are valid. Note that this format must have a corresponding valid informat if you specified the UPDATE or OUTPUT attribute for the argument.

The FORMAT= attribute is not required, but is recommended because format specification is the primary purpose of the ARG statements in the attribute table.

CAUTION:

Using an incorrect format can produce invalid results, cause SAS to crash, or result in serious errors.

The Importance of the Attribute Table

The MODULE function relies heavily on the accuracy of the information in the attribute table. If this information is incorrect, unpredictable results can occur (including a system crash).

Consider an example routine **`xyz`** that expects two arguments: an integer and a pointer. The integer is a code indicating what action takes place. For example, action 1 means that a 20-byte character string is written into the area that is pointed to by the second argument, the pointer.

Suppose you call **`xyz`** using the MODULE function, but you indicate in the attribute table that the receiving character argument is only 10 characters long:

```
routine xyz minarg=2 maxarg=2;
arg 1 input num byvalue format=ib4.;
arg 2 output char format=$char10.;
```

Regardless of the value given by the LENGTH statement for the second argument to MODULE, MODULE passes a pointer to a 10-byte area to the **`xyz`** routine. If **`xyz`** writes 20 bytes at that location, the 10 bytes of memory following the string provided by MODULE are overwritten, causing unpredictable results:

```
data _null_;
    length x $20;
    call module('xyz',1,x);
run;
```

The call might work fine, depending on which 10 bytes were overwritten. However, overwriting can cause you to lose data or cause your system to crash.

Also, note that the PEEKLONG and PEEKCLONG functions rely on the validity of the pointers that you supply. If the pointers are invalid, it is possible that severe errors will result. For example, this code causes an error:

```
data _null_;
   length c $10;
```

```
      /* trying to copy from address 0!!!*/
   c = peekclong(0,10);
run;
```

Special Considerations When Using Shared Libraries

32-Bit and 64-Bit Considerations

Compatibility between Your Shared Libraries and SAS

Starting in SAS 9, SAS is a 64-bit application that runs on all supported UNIX environments that are 64-bit enabled. The only exception is the Linux version of SAS, which can be a 32-bit or 64-bit application. When you call external routines in shared libraries, the shared library needs to be compatible with SAS.

For example, if you are running a 64-bit version of SAS on Solaris, you need to call a routine that is located in libc.so. In order for this shared library to be compatible with SAS, it needs to be a 64-bit shared library.

To determine whether a vendor supplied library is 32-bit or 64-bit, you can use the FILE command. The following output shows the results of using this command on Solaris for a 32-bit and 64-bit library:

```
$ file libc.so
libc.so:  ELF 32-bit MSB dynamic lib SPARC Version 1, dynamically linked,
not stripped

$ file ./libc.so
./libc.so:  ELF 64-bit MSB dynamic lib SPARCV9 Version 1, dynamically linked,
not stripped
```

Memory Storage Allocated by the Shared Library

When specifying your SAS format and informat for each routine argument in the FORMAT attribute of the ARG statement, you need to consider the amount of memory the shared library allocates for the parameters that it receives and returns. To determine how much storage is being reserved for the input and return parameters of the routine in the external shared library, you can use the sizeof( ) C function.

The following table lists the typical memory allocations for C data types for 32-bit and 64-bit systems:

Table 5.2 *Memory Allocations for C Data Types*

Type	32-Bit System Size (Bytes)	32-Bit System Size (Bits)	64-Bit System Size (Bytes)	64-Bit System Size (Bits)
char	1	8	1	8
short	2	16	2	16
int	4	32	4	32

Type	32-Bit System Size (Bytes)	32-Bit System Size (Bits)	64-Bit System Size (Bytes)	64-Bit System Size (Bits)
long	4	32	8	64
long long	8	64	8	64
float	4	32	4	32
double	8	64	8	64
pointer	4	32	8	64

For information about the SAS formats to use for your data types, see "Specifying Formats and Informats to Use with MODULE Arguments" on page 120.

Naming Considerations When Using Shared Libraries

Naming Constraints

SAS loads external shared libraries that meet the following naming constraints:

- The name is eight characters or less.
- The name does not contain a period.

If the name of your external shared library is greater than eight characters or contains a period, then you can create a symbolic link to point to the destination of the shared library. Once the link is created, you can add the name of the symbolic link to the MODULE statement in the SASCBTBL attribute table. When you are ready to execute your SAS program, use the PATH system option to point to the directory that contains the symbolic link.

Example of Creating a Symbolic Link

The Hewlett-Packard shared library libc.sl that is installed in the **`/usr/lib/pa20_64`** directory contains a period in the name. Before SAS will load this shared library, you need to create a symbolic link that meets the naming convention of eight characters or less and no period. The symbolic link shown in the following example points to the target location of libc.sl:

```
$ ln -s /usr/lib/pa20_64/libc.sl /tmp/libclnk
```

After the symbolic link is created, you can then update the MODULE= option in the SASCBTBL attribute table, as shown in the following code:

```
routine name minarg=2 maxarg=2 returns=short module=libclnk;
arg 1 char output byaddr fdstart format=$cstr9.;
arg 2 char output format=$cstr9.;
```

To load the shared library during your invocation of SAS, type the following command:

```
/usr/local/sasv91/sas -path /tmp module.sas
```

Using PEEKLONG Functions to Access Character String Arguments

Because the SAS language does not provide pointers as data types, you can use the SAS PEEKLONG functions to access the data stored at these address values.

For example, the following program demonstrates how the address of a pointer is supplied and how it can set the pointer to the address of a static table containing the contiguous integers 1, 2, and 3. It also calls the **`useptr`** routine in the **`useptr`** shared library on a 64-bit operating system.

```
static struct MYTABLE {
int value1;
int value2;
int value3;
} mytable = {1,2,3};

useptr(toset)
char **toset;
{
   *toset = (char *)&mytable
}
```

The following is the SASCBTBL attribute table entry:

```
routine useptr minarg=1 maxarg=1;
arg 1 char update format=$char20.;
```

The following is the SAS code:

```
data _null_;
   length ptrval $20 thedata $12;
   call module('*i','useptr',ptrval);
   thedata=peekclong(ptrval,12);

   /* Converts hexadecimal data to character data */
   put thedata=$hex24.;

   /* Converts hexadecimal positive binary values to fixed or floating point value */
   ptrval=hex40.;
run;
```

SAS writes the following output to the log.

Output 5.1 *Log Output for Accessing Character Strings with the PEEKCLONG Function*

```
thedata=000000010000000200000003 ptrval=800003FFFF0C
```

In this example, the PEEKCLONG function is given two arguments, a pointer via a numeric variable and a length in bytes. PEEKCLONG returns a character string of the specified length containing the characters at the pointer location.

For more information about the PEEKLONG functions, see “PEEKLONG Function: UNIX” on page 278.

Accessing Shared Libraries Efficiently

The MODULE function reads the attribute table that is referenced by the SASCBTBL fileref once per step (DATA step, PROC IML step, or SCL step). It parses the table and stores the attribute information for future use during the step. When you use the MODULE function, SAS searches the stored attribute information for the matching routine and module names. The first time you access a shared library during a step, SAS loads the shared library and determines the address of the requested routine. Each shared library that you invoke stays loaded for the duration of the step, and is not reloaded in subsequent calls. All modules and routines are unloaded at the end of the step.

In the following example, the attribute table has the following basic form:

```
* routines XYZ and BBB in FIRST.Shared Library;
routine XYZ minarg=1 maxarg=1 module=FIRST;
arg 1 num input;
routine BBB minarg=1 maxarg=1 module=FIRST;
arg 1 num input;
* routines ABC and DDD in SECOND.Shared Library;
routine ABC minarg=1 maxarg=1 module=SECOND;
arg 1 num input;
routine DDD minarg=1 maxarg=1 module=SECOND;
arg 1 num input;
```

The DATA step code looks like the following:

```
filename sascbtbl 'myattr.tbl';
data _null_;
   do i=1 to 50;
      /* FIRST.Shared Library is loaded only once */
      value = modulen('XYZ',i);
      /* SECOND.Shared Library is loaded only once */
      value2 = modulen('ABC',value);
      put i= value= value2=;
   end;
run;
```

In this example, MODULEN parses the attribute table during DATA step compilation. In the first loop iteration (i=1), FIRST.Shared Library is loaded and the XYZ routine is accessed when MODULEN calls for it. Next, SECOND.Shared Library is loaded and the ABC routine is accessed. For subsequent loop iterations (starting when i=2), FIRST.Shared Library and SECOND.Shared Library remain loaded, so the MODULEN function simply accesses the XYZ and ABC routines. SAS unloads both shared libraries at the end of the DATA step.

Note that the attribute table can contain any number of descriptions for routines that are not accessed for a given step. The presence of the attribute table does not cause any additional overhead (apart from a few bytes of internal memory to hold the attribute descriptions). In the above example, BBB and DDD are in the attribute table but are not accessed by the DATA step.

Grouping SAS Variables as Structure Arguments

Passing an Argument to a Structure

A common need when calling external routines is to pass a pointer to a structure. Some parts of the structure might be used as input to the routine, while other parts might be replaced or filled in by the routine. Even though SAS does not have structures in its language, you can indicate to the MODULE function that you want a particular set of arguments grouped into a single structure. You indicate this grouping by using the FDSTART option of the ARG statement to flag the argument that begins the structure in the attribute table. SAS gathers that argument and all the arguments that follow (until encountering another FDSTART option) into a single contiguous block, and passes a pointer to the block as an argument to the shared library routine.

Example: Grouping Your System Information as Structure Arguments

This example uses the **`uname`** routine, which is part of the **`/usr/lib/pa20_64/libc.sl`** shared library in the HP-UX operating environment. This routine returns the following information about your computer system:

- the nodename on which you are executing SAS.
- the version of the operating system.
- the vendor of the operating system.
- the computer identification number.
- model type of your computer.
- the unique identification number of your class of hardware. This value could be a serial number.

The following is the C prototype for this routine:

```
int uname(struct utsname *name);
```

In C, the **`utsname`** structure is defined with the following members:

```
#define UTSLEN 9
#define SNLEN 15

char sysname[UTSLEN];
char nodename[UTSLEN];
char release[UTSLEN];
char version[UTSLEN];
char machine[UTSLEN];
char idnumber[SNLEN];
```

Each of the above structure members are null-terminated strings.

To call this routine using the MODULE function, you use the following attribute table entries:

```
* attribute table entry;
routine uname minarg=6 maxarg=6 returns=short module=libc;
arg 1 char output byaddr fdstart format=$cstr9.;
arg 2 char output                format=$cstr9.;
arg 3 char output                format=$cstr9.;
arg 4 char output                format=$cstr9.;
arg 5 char output                format=$cstr9.;
arg 6 char output                format=$cstr15.;
```

The following example shows the SAS source code to call the **`uname`** routine from within the DATA step:

```
x 'if [ ! -L ./libc ]; then ln -s /usr/lib/pa20_64/libc.sl ./libc ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib:/lib:/usr/lib/pa20_64'
```

```
data _null_;
   length sysname $9 nodename $9 release $9 version $9 machine $9 idnumber $15.
   retain sysname nodename release version machine idnumber " ";
   rc=modulen('uname', sysname, nodename, release, version, machine, idnumber)
   put rc = ;
   put sysname = ;
   put nodename = ;
   put release  = ;
   put version  = ;
   put machine  = ;
   put idnumber = ;
run;
```

SAS writes the following output to the log:

Output 5.2 *Grouping SAS Variables as a Structure*

```
rc=0
sysname=HP-UX
nodename=garage
release=B.11.00
version=u
machine=9000/800
idnumber=103901537
```

Using Constants and Expressions as Arguments to the MODULE Function

You can pass any type of expression as an argument to the MODULE function. The attribute table indicates whether the argument is for input, output, or update.

You can specify input arguments as constants and arithmetic expressions. However, because output and update arguments must be able to be modified and returned, you can pass only a variable for them. If you specify a constant or expression where a value that can be updated is expected, SAS issues a warning message pointing out the error. Processing continues, but the MODULE function cannot perform the update (meaning that the value of the argument that you wanted to update will be lost).

Consider these examples. Here is the attribute table:

```
* attribute table entry for ABC;
routine abc minarg=2 maxarg=2;
arg 1 input format=ib4.;
arg 2 output format=ib4.;
```

Here is the DATA step with the MODULE calls:

```
data _null_;
  x=5;
  /* passing a variable as the    */
  /*   second argument - OK       */
  call module('abc',1,x);

  /* passing a constant as the    */
  /*   second argument - INVALID  */
  call module('abc',1,2);
  /* passing an expression as the */
  /*   second argument - INVALID  */
  call module('abc',1,x+1);
run;
```

In the above example, the first call to MODULE is correct because **`x`** is updated by the value that the **`abc`** routine returns for the second argument. The second call to MODULE is not correct because a constant is passed. MODULE issues a warning indicating you have passed a constant, and passes a temporary area instead. The third call to MODULE is not correct because an arithmetic expression is passed, which causes a temporary location from the DATA step to be used, and the returned value to be lost.

Specifying Formats and Informats to Use with MODULE Arguments

Using the FORMAT Attribute in the ARG Statement

You specify the SAS format and informat for each shared library routine argument by specifying the FORMAT attribute in the ARG statement. The format indicates how numeric and character values should be passed to the shared library routine and how they should be read back upon completion of the routine.

Usually, the format that you use corresponds to a variable type for a given programming language. The following sections describe the proper formats that correspond to different variable types in various programming languages.

C Language Formats

Table 5.3 *C Language Formats*

C Type	SAS Format/Informat for 32-Bit Systems	SAS Format/Informat for 64-Bit Systems
double	RB8.	RB8.
float	FLOAT4.	FLOAT4.
signed int	IB4.	IB4.
signed short	IB2.	IB2.
signed long	IB4.	IB8.
char *	IB4.	IB8.
unsigned int	PIB4.	PIB4.
unsigned short	PIB2.	PIB2.
unsigned long	PIB4.	PIB8.
char[*w*]	$CHAR*w*. or $CSTR*w*. (see “$CSTRw. Format” on page 122)	$CHAR*w*. or $CSTR*w*. (see “$CSTRw. Format” on page 122)

Note: For information about passing character data other than as pointers to character strings, see "$BYVALw. Format" on page 123.

Fortran Language Formats

Table 5.4 *Fortran Language Formats*

Fortran Type	SAS Format/Informat
integer*2	IB2.
integer*4	IB4.
real*4	RB4.
real*8	RB8.
character**w*	$CHAR*w*.

The MODULE function can support Fortran character arguments only if they are not expected to be passed by a descriptor.

PL/I Language Formats

Table 5.5 *PL/I Language Formats*

PL/I Type	SAS Format/Informat
FIXED BIN(15)	IB2.
FIXED BIN(31)	IB4.
FLOAT BIN(21)	RB4.
FLOAT BIN(31)	RB8.
CHARACTER(*w*)	$CHAR*w*.

The PL/I descriptions are added here for completeness. These descriptions do not guarantee that you will be able to invoke PL/I routines.

COBOL Language Formats

Table 5.6 *COBOL Language Formats*

COBOL Format	SAS Format/Informat	Description
PIC S*xxxx* BINARY	IB*w*.	integer binary
COMP-2	RB8.	double-precision floating point

COBOL Format	SAS Format/Informat	Description
COMP-1	RB4.	single-precision floating point
PIC *xxxx* or S*xxxx*	F*w*.	printable numeric
PIC *yyyy*	$CHAR*w*.	character

The following COBOL specifications might not match properly with the formats supplied by SAS because zoned and packed decimal are not truly defined for systems based on Intel architecture.

Table 5.7 *COBOL Specifications and SAS Formats and Informats*

COBOL Format	SAS Format/Informat	Description
PIC S*xxxx* DISPLAY	ZD*w*.	zoned decimal
PIC S*xxxx* PACKED-DECIMAL	PD*w*.	packed decimal

The following COBOL specifications do not have true native equivalents and are usable only in conjunction with the corresponding S370F*xxx* format and informat, which enables IBM mainframe-style representations to be read and written in the UNIX environment.

Table 5.8 *COBOL Specifications Used with the S370Fxxx Group of Formats and Informats*

COBOL Format	SAS Format/Informat	Description
PIC *xxxx* DISPLAY	S370FZDU*w*.	zoned decimal unsigned
PIC S*xxxx* DISPLAY SIGN LEADING	S370FZDL*w*.	zoned decimal leading sign
PIC S*xxxx* DISPLAY SIGN LEADING SEPARATE	S370FZDS*w*.	zoned decimal leading sign separate
PIC S*xxxx* DISPLAY SIGN TRAILING SEPARATE	S370FZDT*w*.	zoned decimal trailing sign separate
PIC *xxxx* BINARY	S370FIBU*w*.	integer binary unsigned
PIC *xxxx* PACKED-DECIMAL	S370FPDU*w*.	packed decimal unsigned

$CSTRw. Format

If you pass a character argument as a null-terminated string, use the $CSTR*w*. format. This format looks for the last non-blank character of your character argument and passes a copy of the string with a null terminator after the last non-blank character. For example, consider the following attribute table entry:

```
* attribute table entry;
routine abc minarg=1 maxarg=1;
arg 1 input char format=$cstr10.;
```

With this entry, you can use the following DATA step:

```
data _null_;
     rc = module('abc','my string');
     run;
```

The $CSTR format adds a null terminator to the character string **`my string`** before passing it to the **`abc`** routine. Adding a null terminator to the character string and then passing the string to the **`abc`** routine is equivalent to the following attribute entry:

```
* attribute table entry;
routine abc minarg=1 maxarg=1;
arg 1 input char format=$char10.;
```

The entry would have the following DATA step:

```
data _null_;
     rc = module('abc','my string'||'00'x);
run;
```

The first example is easier to understand and easier to use when using variable or expression arguments.

The $CSTR informat converts a null-terminated string into a blank-padded string of the specified length. If the shared library routine is supposed to update a character argument, use the $CSTR informat in the argument attribute.

$BYVALw. Format

When you use a MODULE function to pass a single character by value, the argument is automatically promoted to an integer. If you want to use a character expression in the MODULE call, you must use the special format or informat called $BYVAL*w*. The $BYVAL*w*. format and informat expects a single character and will produce a numeric value, the size of which depends on *w*. $BYVAL2. produces a short, $BYVAL4. produces a long, and $BYVAL8. produces a double. Consider this example using the C language:

```
long xyz(a,b)
  long a; double b;
  {
  static char c = 'Y';
  if (a == 'X')
     return(1);
  else if (b == c)
     return(2);
  else return(3);
  }
```

In this example, the **`xyz`** routine expects two arguments, a long and a double. If the long is an **`x`**, the actual value of the long is 88 in decimal. This result happens because an ASCII **`X`** is stored as hexadecimal 58, and this value is promoted to a long, represented as 0x00000058 (or 88 decimal). If the value of **`a`** is **`x`**, or 88, then a 1 is returned. If the second argument, a double, is **`y`** (which is interpreted as 89), then 2 is returned.

If you want to pass characters as the arguments to **xyz** then in the C language, you would invoke them as follows:

```
x = xyz('X',(double)'Z');
y = xyz('Q',(double)'Y');
```

The characters are invoked in this way because the **X** and **Q** values are automatically promoted to integers (which are the same as longs for the sake of this example), and the integer values corresponding to **Z** and **Y** are cast to doubles.

To call **xyz** using the MODULEN function, your attribute table must reflect the fact that you want to pass characters:

```
routine xyz minarg=2 maxarg=2 returns=long;
arg 1 input char byvalue format=$byval4.;
arg 2 input char byvalue format=$byval8.;
```

Note that it is important that the BYVALUE option appears in the ARG statement as well. Otherwise, MODULEN assumes that you want to pass a pointer to the routine, instead of a value.

Here is the DATA step that invokes MODULEN and passes it characters:

```
data _null_;
     x = modulen('xyz','X','Z');
     put x= ' (should be 1)';
     y = modulen('xyz','Q','Y');
     put y= ' (should be 2)';
run;
```

Understanding MODULE Log Messages

If you specify **i** in the control string parameter to MODULE, SAS prints several informational messages to the log. You can use these messages to determine whether you have passed incorrect arguments or coded the attribute table incorrectly.

Consider this example that uses MODULEIN from within the IML procedure. It uses the MODULEIN function to invoke the **changi** routine (which is stored in theoretical TRYMOD.so). In the example, MODULEIN passes the constant 6 and the matrix x2, which is a 4x5 matrix to be converted to an integer matrix. The attribute table for **changi** is as follows:

```
routine changi module=trymod returns=long;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
```

The following IML step invokes MODULEIN:

```
proc iml;
   x1 = J(4,5,0);
   do i=1 to 4;
      do j=1 to 5;
         x1[i,j] = i*10+j+3;
      end;
   end;
   y1= x1;
          x2 = x1;
                   y2 = y1;
   rc = modulein('*i','changi',6,x2);
   ....
```

The `'*i'` control string causes the lines shown in the following output to be written in the log.

Output 5.3 *MODULEIN Log Output*

```
---PARM LIST FOR MODULEIN ROUTINE---  CHR PARM 1 885E0AA8 2A69 (*i)
CHR PARM 2 885E0AD0 6368616E6769 (changi)
NUM PARM 3 885E0AE0 0000000000001840
NUM PARM 4 885E07F0
0000000000002C400000000000002E40000000000000304000000000000031400000000000003240
0000000000003840000000000000394000000000000003A400000000000003B400000000000003C40
0000000000004140000000000008041400000000
---ROUTINE changi LOADED AT ADDRESS 886119B8 (PARMLIST AT 886033A0)--- PARM 1 06000000    <CALL-BY-VALUE>
PARM 2 88604720
0E0000000F000000100000001100000012000000180000001900000001A0000001B0000001C000000
22000000230000002400000025000000260000002C0000002D0000002E0000002F00000030000000
---VALUES UPON RETURN FROM changi ROUTINE---   PARM 1 06000000    <CALL-BY-VALUE>
PARM 2 88604720
140000001F0000002A00000035000000400000008200000008D00000098000000A3000000AE000000
F0000000FB000000060100001101000001C0100005E010000690100007401000007F0100008A010000
---VALUES UPON RETURN FROM MODULEIN ROUTINE---  NUM PARM 3 885E0AE00000000000001840
NUM PARM 4 885E07F0
00000000000034400000000000003F4000000000000045400000000000000804A40000000000005040
0000000000406040000000000000A06140000000000000063400000000000606440000000000C06540
0000000000006E400000000000606F4000000000
```

The output is divided into four sections:

- The first section describes the arguments passed to MODULEIN.

  The CHR PARM *n* portion indicates that character parameter *n* was passed. In the example, 885E0AA8 is the actual address of the first character parameter to MODULEIN. The value at the address is hexadecimal 2A69, and the ASCII representation of that value ('*i') is in parentheses after the hexadecimal value. The second parameter is printed similarly. Only these first two arguments have their ASCII equivalents printed because other arguments might contain unreadable binary data.

  The remaining parameters appear with only hexadecimal representations of their values (NUM PARM 3 and NUM PARM 4 in the example).

  The third parameter to MODULEIN is numeric, and it is at address 885E0AE0. The hexadecimal representation of the floating-point number 6 is shown. The fourth parameter is at address 885E07F0, which points to an area containing all the values for the 4x5 matrix. The `*i` option prints the entire argument. Be careful if you use this option with large matrices, because the log might become quite large.

- The second section of the log lists the arguments that are to be passed to the requested routine and, in this case, changed. This section is important for determining whether the arguments are being passed to the routine correctly. The first line of this section contains the name of the routine and its address in memory. It also contains the address of the location of the parameter block that MODULEIN created.

  The log contains the status of each argument as it is passed. For example, the first parameter in the example is call-by-value (as indicated in the log). The second parameter is the address of the matrix. The log shows the address, along with the data to which it points.

  Note that all the values in the first parameter and in the matrix are long integers because the attribute table states that the format is IB4.

- In the third section, the log contains the argument values upon return from **changi**. The call-by-value argument is unchanged, but the other argument (the matrix) contains different values.
- The last section of the log output contains the values of the arguments as they are returned to the MODULEIN CALL routine.

Examples of Accessing Shared Executable Libraries

Example 1: Updating a Character String Argument

This example uses the **tmpnam** routine in the shared library supplied by Solaris, libc.so, which is installed in the **/usr/lib/sparcv9** directory. The **tmpnam** routine generates a unique filename that can be used safely as a temporary filename. The temporary filename is typically placed in the **/var/tmp** directory.

Here is the C prototype for this routine:

```
char * tmpnam(char *s);
```

The attribute table for this prototype would be the following:

```
routine tmpnam minarg=1 maxarg=1 returns=char255. module=libc;
arg 1 char output byaddr format=$cstr255;
```

The SAS source code would be the following:

```
x 'if [ ! -L ./libc ] ; then ln -s /usr/lib/sparcv9/libc.so.1 ./libc ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib/sparcv9:/usr/lib:/lib';

data _null_;
   length tempname $255 tname $255;
   retain tempname tname " ";
   tname = modulec ('tmpnam', tempname);
   put tempname = ;
   put tname = ;
run;
```

The SAS log output would be the following:

Output 5.4 *Updating a Character String Argument*

```
tempname=/var/tmp/aaaKraydG
tname=/var/tmp/aaaKraydG
```

The POSIX standard for the maximum number of characters in a pathname is defined in **/usr/include/limits.h** to be 255 characters, so this example uses 254 as the length of the generated filename (**tempname**) with one character reserved for the null terminator. The $CSTR255. informat ensures that the null terminator and all subsequent characters are replaced by trailing blanks when control returns to the DATA step.

Example 2: Passing Arguments by Value

This example calls the **`access`** routine that is supplied by most UNIX vendors. This particular **`access`** routine is in the Hewlett-Packard shared library, libc.sl, which is installed under the **`/usr/lib/pa20_64`** directory.

Here is the C prototype for this routine:

```
int access(char *path, int amode);
```

The **`access`** routine checks the file that is referenced by the accessibility path according to the bit pattern contained in **`amode`**. You can use the following integer values for **`amode`** that correspond to the type of permission for which you are testing:

4	Read access
2	Write access
1	Execute (search) access
0	Check existence of file

A return value of 0 indicates a successful completion and the requested access is permitted. A return value of –1 indicates a failure and the requested access is not permitted.

Because the **`amode`** argument is a pass-by-value, this example includes the BYVALUE specification for *arg 2* in the attribute table. If both arguments were a pass-by-value, one could use the CALLSEQ=BYVALUE attribute in the ROUTINE statement and it would not be necessary to specify the BYVALUE option in *arg 2*.

The attribute table would be the following:

```
routine access minarg=2 maxarg=2 returns=short module=libc;
arg 1 char input byaddr format=$cstr200.;
arg 2 num input byvalue format=ib4.;
```

The SAS source code would be the following:

```
x 'if [ ! -L ./libc ] ; then ln -s /usr/lib/pa20_64/libc.so ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib/pa20_64:/usr/lib:/lib' ;

data _null_;
  length path $200.;
  path='/dev';

  /* A non-root user is testing for write permission in the /dev directory */
  rc = modulen("*ie",'access',path,2);
  put rc = ;
run;
```

The SAS log output would be the following:

Output 5.5 *Log Output If Request Access Is Permitted*

```
rc=-1
```

If you changed the SAS source code to check for a Write permission in the user's $HOME directory, the output would be different:

```
data _null_;
   length homedir $200.;
   homedir=sysget('HOME');

   /* A user is testing for write permissions in their $HOME directory */
   rc = modulen("*ie",'access',homedir,2);
   put rc = ;
run;
```

In this case, the SAS log output would be the following:

Output 5.6 *Log Output for Successful Completion (Access Permitted)*

```
rc=0
```

Example 3: Using PEEKCLONG to Access a Returned Pointer

This example uses the **`strcat`** routine, which is part of the Red Hat Linux shared library libc-2.2.3.so. This library is typically installed in the **`/lib/i686`** directory. This routine concatenates two strings and returns a pointer to the newly concatenated string.

Here is the C prototype for this routine:

```
char *strcat(char, *dest, const char *src);
```

The proper SASCBTBL attribute table would be the following:

```
routine strcat minarg=2 maxarg=2 returns=ulong module=libc;
arg 1 char input format=$cstr200.;
arg 2 char input format=$cstr200.;
```

The following example shows the SAS code:

```
filenamesascbtbl './sascbtbl.txt';

data _null_;
   file sascbtbl;
   put "routine strcat minarg=2 maxarg=2 returns=ulong module=libc;";
   put "arg 1 char input format=$cstr200.;";
   put "arg 2 char input format=$cstr200.;";
run;

data _null_;
   length string1 string2 newstring $200;
   length chptr $20;
   string1='This is string one and';
   string2=' this is string two.';
   chptr=modulec('strcat', string1, string2);
   newstring=peekclong(chptr,200);
   put newstring=;
run;
```

SAS writes the following output to the log:

Output 5.7 *Log Output for Using PEEKCLONG to Access a Returned Pointer*

```
newstring=This is string one and this is string two.
```

The PEEKCLONG function was used here because the Red Hat Linux shared library **`/lib/i686/libc-2.2.3.so`** is a 32-bit library. The following output demonstrates this:

```
$pwd
/lib/i686

$file ./libc-2.2.3.so
libc-2.2.3.so:  ELF 32-bit LSB shared object, Intel 80386, version 1, not stripped
```

For more information about the PEEKLONG and PEEKCLONG functions, see "PEEKLONG Function: UNIX" on page 278 and "PEEKCLONG Function" in *SAS Functions and CALL Routines: Reference*.

Example 4: Using Structures

"Grouping SAS Variables as Structure Arguments" on page 117 describes how to use the FDSTART attribute to pass several arguments as one structure argument to a shared library routine. The passing of several arguments as one structure is another example of using structures with another routine in an external shared library.

The **`statvfs`** routine that is available under most UNIX operating systems retrieves file system information. This example uses the **`statvfs`** routine that is in the Solaris libc.so.1 shared library and typically installed in the **`/usr/lib/sparcv9`** directory.

Here is the C prototype for this routine:

```
int statvfs(const char *path, struct statvfs *buf);
```

The **`statvfs`** routine will return a 0 if the routine completes successfully and –1 if there is a failure.

The **`statvfs`** structure is defined with the following members:

```
unsigned long f_bsize;         /* preferred file system block size */
unsigned long f_frsize;        /* fundamental file system block */
unsigned long f_blocks;        /* total number of blocks on file system in units */
unsigned long f_bfree;         /* total number of free blocks */
unsigned long f_bavail;        /* number of free blocks available to non-superuser */
unsigned long f_files;         /* total number of file nodes (inodes) */
unsigned long f_ffree;         /* total number of free file nodes */
unsigned long f_favail;        /* number of inodes available to non-superuser */
unsigned long f_fsid;          /* file system id (dev for now) */
char          f_basetype[16];  /* target fs type name, null-terminated */
unsigned long f_flag;          /* bit mask of flags */
unsigned long g f_namemax;     /* maximum filename length */
char          f_fstr[32];      /* file system specific string */
```

The SASCBTBL attribute table would be the following:

```
routine statvfs
   minarg=14
   maxarg=14
   returns=short
   module=libc;
arg 1  char input  byaddr         format=$char256.;
arg 2  num  output byaddr fdstart format=pib8.;
arg 3  num  output                format=pib8.;
arg 4  num  output                format=pib8.;
arg 5  num  output                format=pib8.;
arg 6  num  output               format=pib8.;
arg 7  num  output               format=pib8.;
arg 8  num  output               format=pib8.;
arg 9  num  output               format=pib8.;
arg 10 num  output               format=pib8.;
arg 11 char output               format=$cstr16.;
arg 12 num  output               format=pib8.;
arg 13 num  output               format=pib8.;
arg 14 char output               format=$cstr32.;
```

The SAS source code to call the **statvfs** routine from within the DATA step would be the following:

```
x 'if [ ! -L ./libc ]; then ln -s /usr/lib/sparcv9/libc.so.1 ./libc ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib/sparcv9:/usr/lib:/lib';

data _null_;
   length f_basetype $16. f_fstr $32.;
   retain f_bsize f_frsize f_blocks f_bfree f_bavail f_files f_ffree f_favail
          f_fsid f_flag f_namemax 0;
   retain f_basetype f_fstr ' ';
   rc=modulen ('statvfs' , '/tmp', f_bsize, f_frsize, f_blocks, f_bfree, f_bavail,
               f_files, f_ffree, f_favail, f_fsid, f_basetype, f_flag,
               f_namemax, f_fstr);
   put rc = ;
   put f_bsize = ;
   put f_frsize = ;
   put f_blocks = ;
   put f_bfree = ;
   put f_bavail = ;
   put f_files = ;
   put f_ffree = ;
   put f_favail = ;
   put f_fsid = ;
   put f_basetype = ;
   put f_flag = ;
   put f_namemax = ;
   /* Determining the total bytes available in the file system and then dividing the
    total number of bytes by the number of bytes in a gigabyte */
   gigsfree = ((f_bavail * f_bsize)/1073741824);
   put 'The total amount of space available in /tmp is 'gigsfree 4.2' Gigabytes.';
run;
```

The SAS log output would be the following:

Output 5.8 *Log Output for Using Structures*

```
rc=0
f_bsize=8192
f_frsize=8192
f_blocks=196608
f_bfree=173020
f_bavail=173020
f_files=884732
f_ffree=877184
f_favail=877184
f_fsid=2
f_basetype=tmpfs
f_flag=4
f_namemax=255

The total amount of space available in /tmp is 1.32 Gigabytes.
```

Example 5: Invoking a Shared Library Routine

This example shows how to pass a matrix as an argument within PROC IML. The example creates a 4x5 matrix. Each cell is set to 10*x*+*y*+3, where *x* is the row number and *y* is the column number. For example, the cell at row 1 column 2 is set to (10*1)+2+3, or 15.

The example invokes several routines from the theoretical TRYMOD shared library. It uses the **`changd`** routine to add 100*x*+10*y* to each element, where *x* is the C row number (0 through 3) and *y* is the C column number (0 through 4). The first argument to **`changd`** specifies the extra amount to sum. The **`changdx`** routine works just like **`changd`**, except that it expects a transposed matrix. The **`changi`** routine works like **`changd`** except that it expects a matrix of integers. The **`changix`** routine works like **`changdx`** except that integers are expected.

Note: A maximum of three arguments can be sent when invoking a shared library routine from PROC IML.

In this example, all four matrices x1, x2, y1, and y2 should become set to the same values after their respective MODULEIN calls. Here are the attribute table entries:

```
routine changd module=trymod returns=long;
arg 1 input num format=rb8. byvalue;
arg 2 update num format=rb8.;
routine changdx module=trymod returns=long
   transpose=yes;
arg 1 input num format=rb8. byvalue;
arg 2 update num format=rb8.;
routine changi module=trymod returns=long;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
routine changix module=trymod returns=long
   transpose=yes;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
```

Here is the PROC IML step:

```
proc iml;
   x1 = J(4,5,0);
   do i=1 to 4;
      do j=1 to 5;
         x1[i,j] = i*10+j+3;
         end;
   end;
   y1= x1; x2 = x1; y2 = y1;
   rc = modulein('changd',6,x1);
   rc = modulein('changdx',6,x2);
   rc = modulein('changi',6,y1);
   rc = modulein('changix',6,y2);
   print x1 x2 y1 y2;
run;
```

The following are the results of the PRINT statement:

Output 5.9 *Invoking a Shared Library Routine from PROC IML*

```
X1
 20       31       42       53       64
130      141      152      163      174
240      251      262      273      284
350      361      372      383      394
 X2
 20       31       42       53       64
130      141      152      163      174
240      251      262      273      284
350      361      372      383      394
 Y1
 20       31       42       53       64
130      141      152      163      174
240      251      262      273      284
350      361      372      383      394
 Y2
 20       31       42       53       64
130      141      152      163      174
240      251      262      273      284
350      361      372      383      394
```

Chapter 6

Viewing Output and Help in the SAS Remote Browser

What Is Remote Browsing?

Remote browsing enables you to view SAS documentation, URLs that are specified in the WBROWSE command, and ODS output in the Web browser on your local computer. In the past, all Web documentation was displayed by executing a Netscape browser on the SAS server. By displaying this documentation locally, you have faster access to the documentation and you free up resources on the SAS server that were used by Netscape.

A small software agent called the remote browser server runs on your local computer. When SAS needs to display HTML content, it connects to the remote browser server and sends the URL that references the content. The remote browser server then passes the URL to a browser for display. If the remote browser server is not running on your computer, SAS displays a dialog box that contains the URL that you need to use to download the remote browser server.

Two system options are provided to configure remote browsing: HELPHOST and HELPPORT. These options specify the host name and port number of the computer where HTML content is to be displayed. In most cases, these options do not need to be set. HELPHOST defaults to the host name that is specified in the X11 DISPLAY environment variable, or to the IP address that is specified in the SSH_CLIENT environment variable if the client connects to the UNIX host by using SSH with X11 forwarding enabled. HELPPORT defaults to the standard port for the remote browser server.

Using Remote Browsing with ODS Output

The SAS Output Delivery System (ODS) can be used to generate graphical reports of your SAS data. Remote browsing enables you to view your output directly from a SAS session as the output is generated or on demand from the Results window.

Remote browsing displays ODS output in many formats. If your browser does not have the appropriate plug-in for output that is not HTML, the browser displays a dialog box rather than the output. This dialog box enables you to download your output to your computer and view it using a local program such as Excel for an XSL file.

The automatic display of ODS output (HTML, PDF, and RTF only) is turned off by default. You can turn on the automatic display of ODS output by issuing the AUTONAVIGATE command in the Results window or by selecting **View results as they are generated** from the **Results** tab of the Preferences dialog box.

Installing the Remote Browser Server

You can install the remote browser server directly from your SAS session. If SAS is unable to make a connection for remote browsing, SAS displays a dialog box that contains the URL that you need to download the installer. Use this URL to download and install the remote browser server. Do not exit SAS. To install the remote browser server, follow these steps:

1. Type the URL that appears in the dialog box into your browser and press ENTER, or use the **Copy URL** button in the dialog box to copy the URL, and then paste it into your browser.
2. After the download page is displayed, download the installer that is appropriate for your computer.
3. Run the installer.
   - In the Windows environment, the remote browser server is added to your start-up items, so that the server will start whenever you log in. An icon is displayed in your system tray to indicate that the remote browser server is running.
   - In the Linux environment, manually add the command `rbrowser` to the start-up script for your windowing environment. The remote browser server will initially run minimized.

System Options for Remote Browsing

After the remote browser server is running on your computer, you can run the remote browsing system by specifying the HELPHOST and HELPPORT system options.

- The HELPHOST system option specifies the name of your host computer where the remote browsing system is to be displayed. If you do not specify this option, then the host name specified in the X display name is used. For more information, see "HELPHOST System Option: UNIX" on page 388.

- The HELPPORT system option specifies the port number for the remote browser server that is installed on your computer. Under UNIX, you can use the default value for this option. For more information, see the "HELPPORT= System Option" in *SAS System Options: Reference*.

You can set these options in your configuration file, at SAS invocation, or during your SAS session in the OPTIONS statement or in the SAS System Options window.

Setting Up the SAS Remote Browser

Setting Up the SAS Remote Browser at SAS Invocation

The following syntax is specific for UNIX operating environments and shows how you might set up the SAS Remote Browser if your remote browser server is using network port 12000:

```
sas93 -helpport 12000
```

Because you did not specify the HELPHOST system option, SAS uses the host name that is specified in the X display name.

Setting Up the SAS Remote Browser During a SAS Session

The syntax in this example applies to UNIX environments.

You can set up the remote browsing system during a SAS session by using the OPTIONS statement or the SAS System Options window. The following example uses the OPTIONS statement to change the value of the HELPPORT system option:

```
options helpport=12000;
```

Because you did not specify the HELPHOST system option, its value remains unchanged.

Remote Browsing and Firewalls

For General Users

If your network has a firewall between desktop computers and the computer that is hosting SAS, Web browsers cannot display Web pages from your SAS session. Usually, this problem is indicated by a time-out or connection error from the Web browser. If you receive a time-out or connection error, contact your system administrator.

For System Administrators

To enable the display of Web pages when a firewall exists between desktop computers and the computer that is hosting SAS, a firewall rule must be added that allows a Web browser to connect to SAS. The firewall rule specifies a range of network ports for which SAS remote browsing connections are allowed. Contact the appropriate system administrator who can select and configure a range of network ports for remote browsing. The range depends on the number of simultaneous SAS users. A value of approximately three times the number of simultaneous SAS users should reserve a sufficient number of network ports.

After the firewall rule is added, SAS must be configured to listen for network connections in the network port range. Normally, SAS selects any free network port, but the HTTPSERVERPORTMIN and the HTTPSERVERPORTMAX system options limit the network ports that SAS can select. Add these system options to your SAS configuration file. Set HTTPSERVERPORTMIN to the lowest port in the network range. Set HTTPSERVERPORTMAX to the highest port in the network range. For example, if the system administrator defined a network port range of 8000 to 8200, the system options would be the following:

```
httpserverportmin=8000
httpserverportmax=8200
```

After these system options are set, desktop computers can display Web pages. If there is an insufficient number of network ports, or the system options are specified incorrectly, a message appears in the SAS log.

For more information about these system options, see the “HTTPSERVERPORTMIN= System Option” in *SAS System Options: Reference* and “HTTPSERVERPORTMAX= System Option” in *SAS System Options: Reference*.

Part 2

SAS Windowing Environment

Chapter 7
Working in the SAS Windowing Environment

Definition of the SAS Windowing Environment

The SAS windowing environment refers to the windows that open when you invoke SAS. These windows include: the Program Editor, Log, Output, Explorer, and Results. These windows appear when you start SAS from your X workstation or through an X emulator. For more information about these windows, see the online SAS Help and Documentation.

The SAS windowing environment supports the use of X-based graphical user interfaces (GUIs). In UNIX environments, SAS provides an X Window System interface that is based on the Motif style.

Many features of the SAS windowing environment are controlled by X resources. For example, colors, window sizes, the appearance of the SAS ToolBox, and key definitions are all controlled through X resources. “Customizing the SAS Windowing Environment” on page 164 provides general information about resources, such as how to specify resources, and describes all of the resources that you can use to customize the interface.

Description of SAS in the X Environment

Definition of X Window System

The X Window System is a networked windowing system. If several computers are on a network, you can run an X server that, in turn, serves X applications (as clients) from all the other computers in the network.

X Window Managers

In UNIX environments, SAS features an X Window System interface that is based on Motif. This interface uses the window manager on your system to manage the windows on your display. Any window manager that is compliant with the Inter-Client Communication Conventions Manual (ICCCM) can be used with the Motif interface to SAS. Vendors provide at least one window manager with the X Window System environment. A common window manager is the GNOME. KDE is an alternative window manager. You should read the documentation that is supplied by the vendor for the window manager that you are using.

All window managers perform the same basic functions, but they differ in their style and in their advanced functions. The appearance and function of the interface to SAS depends to some extent on your X window manager. Most window managers provide some type of frame around a window. The window manager also governs the placement, sizing, stacking, and appearance of windows, as well as their interaction with the keyboard. The basics of interacting with SAS are the same for all window managers: opening menus, moving windows, responding to dialog boxes, dragging text, and so on.

SAS Window Session ID

When you run SAS on an X workstation, SAS shares the display with other X applications, including other SAS sessions. To enable you to distinguish between different applications and SAS sessions, SAS generates a SAS window session ID for each session by appending a number to the application name, which, by default, is **`SAS`**. This session ID appears in the window title bar for each SAS window and in the window icon title. The SAS sessions are assigned sequentially. Your first SAS session is not assigned a number, so the session ID is **`SAS`**; your second SAS session is assigned the session ID **`SAS2`**, and so on. Although the default application name is **`SAS`**, you can use the **`-name`** X option or the **`-title`** X option to change the instance name. The instance name can be up to 64 characters long and is displayed in the case in which it was entered, which can be lowercase, mixed case, or uppercase.

Workspace and Gravity in a SAS Session

When you use SAS on an X workstation, the display might be shared by many concurrent applications. When SAS windows from different sessions and windows from other applications appear on the display, the display can become cluttered. To help alleviate this problem, the windows for a SAS session first appear within an application workspace (AWS). The AWS defines a rectangular region that represents a virtual display in which SAS windows are initially created. SAS attempts to position the AWS in relation to the upper left corner of your display. In other words, the workspace gravitates toward a certain direction (session gravity) on the display. Some window manager configurations might override the placement that SAS has chosen for a window.

If you issue windowing commands or execute SAS procedures that create new SAS windows, the same rules of initial position and size apply to these windows: they are initially placed in the SAS AWS. You can use the WSAVE command to save the current window positions (or geometry). For more information, see "Customizing Session Workspace, Session Gravity, and Window Sizes in UNIX Environments" on page 206.

Window Types

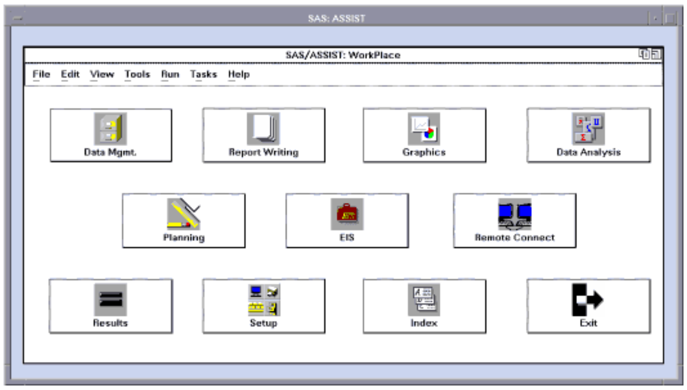
Top-Level Windows

SAS uses primary and interior windows. Some SAS applications consist of one or more primary windows controlled by the X window manager, in addition to the interior windows controlled by SAS. The SAS windowing environment primary windows, as well as most SAS application windows, initially appear as top-level windows. Top-level windows interact directly with the X window manager. They have a full title bar along with other window manager decorations. You can manipulate them individually after they appear on the display.

Interior Windows

Interior windows behave differently from primary windows. SAS/ASSIST software is an application with interior windows. Interior windows are contained within container windows, which might not be primary windows. The following display shows an interior window in SAS/ASSIST software.

Display 7.1 *Sample Interior Window*



SAS provides some degree of window management for interior windows. Specifically, interior windows have the following sizing and movement capabilities:

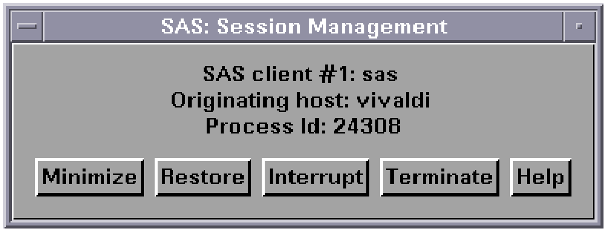
- You can move interior windows by clicking the interior window title bar and dragging the window to the desired location. If the destination of the interior window is outside the bounds of the container window, the container window changes according to the value of the `SAS.awsResizePolicy` resource. (The space within the container window is the application workspace, which is described in "Workspace and Gravity in a SAS Session" on page 141.) For more information, see "Overview of X Resources" on page 164.
- Interior windows cannot be minimized individually. Clicking on the container window icon button minimizes the container window and its interior windows.
- A **push-to-back** button (the small overlapping squares in the upper right corner) is available with interior windows. However, you cannot push an active window behind an inactive window.

The SAS Session Manager (motifxsassm) in UNIX

What Is the SAS Session Manager?

The SAS Session Manager for X (**`motifxsassm`**) is an X client that is run by SAS when you use the SAS windowing environment. The SAS Session Manager is automatically minimized when you start SAS. The SAS: Session Management dialog box for the SAS Session Manager appears as shown in the following display:

Display 7.2 *SAS: Session Management Dialog Box*



The SAS: Session Management dialog box lists the following information:

- which SAS session it controls
- the host computer from which the SAS session was invoked
- the UNIX process identifier of the SAS session

Features of the SAS Session Manager

The buttons in the SAS: Session Management dialog box enable you to do the following tasks:

Minimize
: maps and minimizes all windows of the SAS session. This function is performed with standard X library calls and will work with most X window managers.

Restore
: restores all of the windows that are open in the SAS session that is controlled by that SAS Session Manager. This function is performed with standard X library calls and will work with most X window managers.

Interrupt
: sends a UNIX signal to SAS. When SAS receives the signal, it displays the Tasking Manager dialog box (see "Interrupting a SAS Session" on page 144).

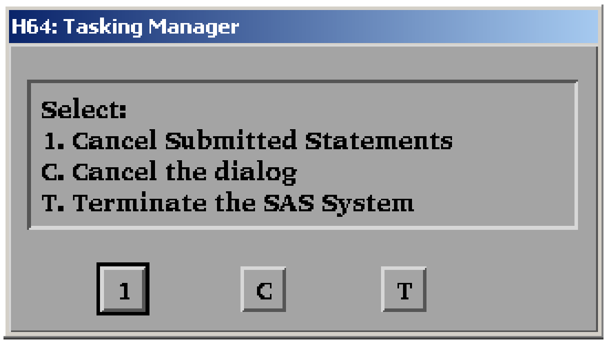
Terminate
: displays a dialog box that asks you to confirm whether you want to terminate the SAS session.

Help
: provides Help for the SAS: Session Management dialog box.

Interrupting a SAS Session

When you click **Interrupt** in the SAS: Session Management dialog box, and if no PROC or DATA step is executing, the following Tasking Manager dialog box appears:

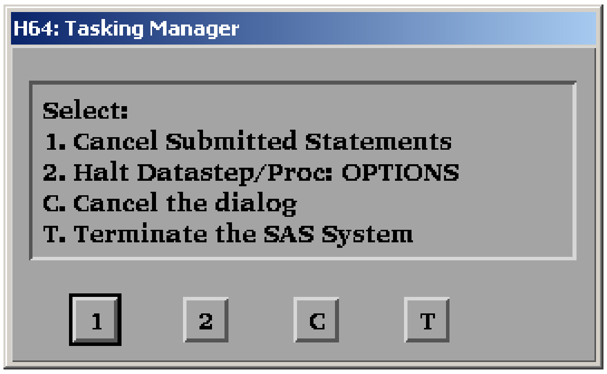
Display 7.3 *Tasking Manager Dialog Box*


H64: Tasking Manager

Select:
1. Cancel Submitted Statements
C. Cancel the dialog
T. Terminate the SAS System

If a PROC or DATA step is executing, the following Tasking Manager dialog box appears:

Display 7.4 *Tasking Manager Dialog Box: DATA Step or PROC Step Executing*


H64: Tasking Manager

Select:
1. Cancel Submitted Statements
2. Halt Datastep/Proc: OPTIONS
C. Cancel the dialog
T. Terminate the SAS System

Click one of the following buttons in the Tasking Manager dialog box:

1
: causes the current PROC or DATA step statements to be deleted.

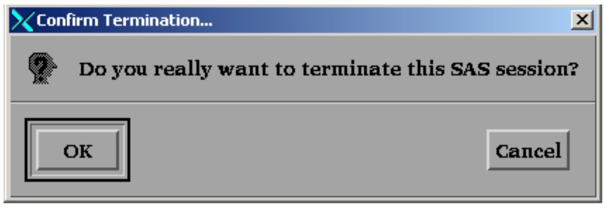
2
: causes the current PROC or DATA step to receive a request to interrupt processing. You are prompted to confirm this action.

C
: closes the dialog box without affecting SAS processing.

T
: forces SAS to terminate the SAS session. You are prompted to confirm termination.

The following Confirm Termination dialog box appears:

Display 7.5 *Confirm Termination Dialog Box*



If you click **OK**, the SAS Session Manager sends a UNIX signal to the SAS session that forces the session to terminate.

CAUTION:

Terminating your SAS session might result in data loss or data corruption. Before terminating your SAS session, you should attempt to end SAS using one of the methods described in "Methods for Exiting SAS" on page 27.

Using the Host Editor from within Your SAS Session

When you issue the HOSTEDIT command, SAS passes the request to the SAS Session Manager, which then invokes your host editor, so the SAS Session Manager must be running for the HOSTEDIT command to take effect. When you issue the HOSTEDIT command, SAS creates a temporary file that contains the data from the active SAS window and passes this file to your host editor. (These temporary files are stored in the directory specified by the SAS WORK option.) When you save your file in the host editor, the file is copied back into the SAS window if the window is able to be written to, and the temporary files are deleted when the SAS session ends. For more information, see "Configuring SAS for Host Editor Support in UNIX Environments" on page 160.

Closing the SAS Session Manager

If you close the SAS: Session Management dialog box, you cannot retrieve the SAS Session Manager. To display the SAS Session Manager again, you can reinvoke **`!SASROOT/utilities/bin/motifxsassm`** with the *-pid* or the *-sessionid* arguments. Execute these commands at the UNIX prompt or use them with the X statement:

```
!SASROOT/utilities/bin/motifxsassm -pid pid
```

```
!SASROOT/utilities/bin/motifxsassm -sessionid integer
```

Disabling the SAS Session Manager

You can disable the SAS Session Manager in the following ways:

- Select **Tools** ⇨ **Options** ⇨ **Preferences**.

  On the **General** tab, deselect the **Start Session manager** check box.

- Specify the following X resource, in lowercase, on the SAS command line at invocation:

  ```
  sas -xrm 'SAS.startSessionManager: False'
  ```

  Specifying the **`sas.startSessionManager`** X resource will deselect the **Start Session manager** check box in the Preferences dialog box.

Note: SAS saves the settings in the Preferences dialog box when it exits. If you have disabled the SAS Session Manager during your session, then the next time you invoke SAS, the SAS Session Manager will not run. To start the SAS Session Manager, select the **Start Session manager** check box in the Preferences dialog box or specify the following command, in lowercase, on the SAS command line at invocation:

```
sas -xrm 'SAS.startSessionManager: True'
```

Displaying Function Key Definitions in UNIX Environments

Benefits of Assigning Function Key Definitions

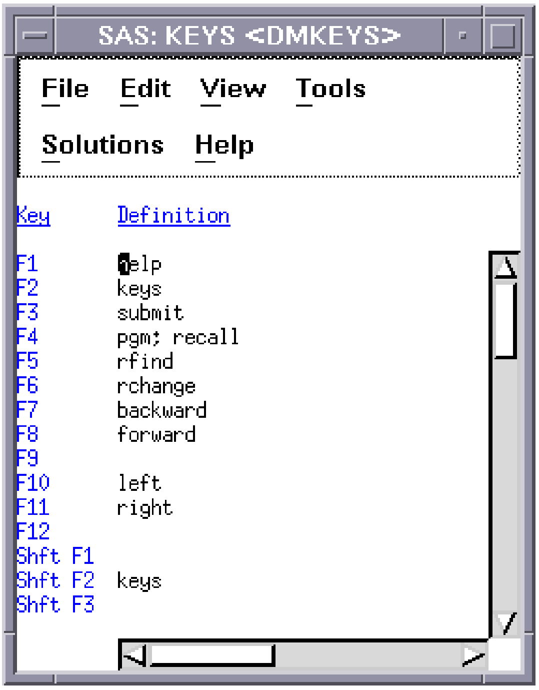
Function keys provide quick access to commands. They enable you to issue commands, insert text strings, and insert commands in programs. Function key definitions can be different on different terminals. These definitions can be fully customized.

How to Display Function Key Definitions

You can open the KEYS (DMKEYS) window to display all of your function key definitions in one of the following ways:

- Press **F2**.
- Issue the KEYS command.
- Select **Tools** ⇨ **Options** ⇨ **Keys**.

Display 7.6 *SAS: KEYS (DMKEYS) Window*



```
Key       Definition

F1        help
F2        keys
F3        submit
F4        pgm; recall
F5        rfind
F6        rchange
F7        backward
F8        forward
F9
F10       left
F11       right
F12
Shft F1
Shft F2   keys
Shft F3
```

To view a single key definition without bringing up the KEYS window, use the KEYDEF command and specify the key definition that you want to view. For example, the following command displays the definition for key F4:

```
keydef f4
```

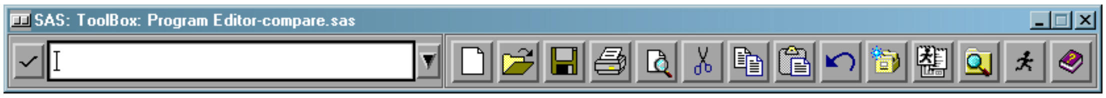
For information about customizing key definitions, see “Customizing Key Definitions in UNIX Environments” on page 184. For information about the Keys window and the KEYDEF command, see the online SAS Help and Documentation.

The SAS ToolBox in UNIX Environments

Introduction to the SAS ToolBox

The SAS ToolBox has two parts as illustrated in the following display:

Display 7.7 *The SAS ToolBox*



- A command window that enables you to quickly enter any command in the active SAS window. For information about commands that are available under UNIX, see “SAS Commands under UNIX” on page 224 and the SAS commands section in the Base SAS section in the online SAS Help and Documentation.
- A toolbar that contains several tool icons. When you select a tool icon, SAS immediately executes the command that is associated with that icon. You can customize both the toolbar and the tool icons. For more information, see “Using the Tool Editor” on page 178.

The name of the active window is displayed in the title bar of the SAS ToolBox. For example, if the Log window were active, the title bar would say SAS ToolBox: Log instead of SAS ToolBox: Program Editor.

Under UNIX, the default SAS ToolBox automatically appears at the bottom of the SAS windows stack by default. To control its configuration, you use the Preferences dialog box. (See “Modifying the SAS ToolBox Settings ” on page 170.)

Customizing the Default SAS ToolBox

The default SAS ToolBox is automatically copied to your SASUSER.PROFILE.DMS.TOOLBOX regardless of whether you customize the ToolBox. If you invoke an application that does not have an associated PMENU entry, the default toolbox is displayed for that application. If you then customize the toolbox for that application, the customized toolbox is stored in SASUSER.PROFILE.DEFAULT.TOOLBOX, where DEFAULT is the same entry name as the PMENU entry for the window or application.

You can customize the default SAS ToolBox, create multiple toolboxes and switch between them, and create application-specific toolboxes (such as with SAS/AF applications) that are automatically loaded when the application is loaded. Only one toolbox is displayed at a time, and the tools in the toolbox change as you move between applications. For more information, see "Customizing Toolboxes and Toolsets in UNIX Environments" on page 177.

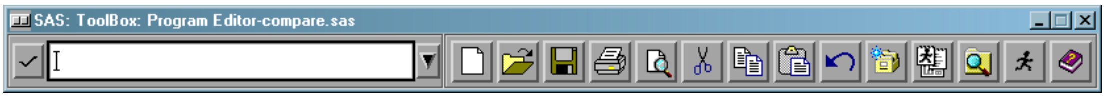
Default Configuration for the Command Window and the Toolbar

By default, the toolbar and the command window are joined and are automatically displayed when SAS initializes unless one of the following conditions applies:

- You executed your SAS job in a non-windowing environment mode.
- The `SAS.defaultToolBox` or `SAS.defaultCommandWindow` resource is set to **False**. The default value is **True**. For more information about the resources that control the toolbox, see "X Resources That Control Toolbox Behavior" on page 177.
- You deselect **Display tools window**, **Display command window**, or **Combine windows** from the **ToolBox** tab in the Preferences dialog box.

The following display shows the command window and the toolbar in their default configuration.

Display 7.8 Default Configuration for Command Window and Toolbar



Opening and Closing the Command Window and the Toolbar

The following table lists the steps that you can use to open and close the command window and toolbar.

Table 7.1 Steps for Opening and Closing the Command Window and the Toolbar

Window	How to Open	How to Close
Command Window and Toolbar	To open both windows, complete any of the following steps: • Issue the COMMAND WINDOW command. • Issue the TOOLLOAD command. For more information, see "TOOLLOAD Command: UNIX" on page 238. • Select **Tools** ⇨ **Options** ⇨ **Toolbox**.	To close these windows, complete any of the following steps: • Select `Close` from the **ToolBox** window menu. • Enter the TOOLCLOSE command as described in "TOOLCLOSE Command: UNIX" on page 237. • Select **Tools** ⇨ **Options** ⇨ **Toolbox** so that **ToolBox** is deselected.

Window	How to Open	How to Close
Command Window	To open only the command window, deselect **Combine Windows** on the tab of the Preferences dialog box, and complete any of the following steps: • Select **Display command window** in the **ToolBox** tab of the Preferences dialog box. • Issue the COMMAND WINDOWS command.	To close only the command window, complete the following steps: • Deselect **Display command window** on the **ToolBox** tab of the Preferences dialog box. • Select **Close** from the window menu.
Toolbar	To open only the toolbar, deselect **Combine windows** on the **ToolBox** tab of the Preferences dialog box, and complete any of the following steps: • Select **Display tools window** on the **ToolBox** tab of the Preferences dialog box. • Issue the TOOLLOAD command. See “TOOLLOAD Command: UNIX” on page 238. • Select **Tools** ⇨ **Options** ⇨ **Toolbox**.	To close only the toolbar, deselect **Combine windows** on the **ToolBox** tab of the Preferences dialog box, and complete any of the following steps: • Deselect **Display tools window** on the **ToolBox** tab of the Preferences dialog box. • Issue the TOOLCLOSE command as described in “TOOLCLOSE Command: UNIX” on page 237. • Select **Tools** ⇨ **Options** ⇨ **Toolbox** so that **Toolbox** is deselected.

Executing Commands

You can execute commands from either the command window or the toolbar. The following table gives more details about how to execute commands.

Table 7.2 Executing Commands in the Command Window and the Toolbar

Location	Execution
Command Window	To execute a command, complete the following steps: 1. Click in the command window. 2. Enter the command. 3. Press ENTER or click the check mark. The command is executed in the active SAS window. You can use the up and down arrow keys to scroll through previously entered commands, or you can select a previous command from the drop-down list. Use the left mouse button to select a command from the drop-down list. Use the right mouse button to select and execute a command from the list.

Location	Execution
Toolbar	To execute a command, click a tool icon in the toolbar to execute the command or commands that are associated with that icon. If you place the cursor over an icon for the amount of time specified by the `SAS.toolBoxTipDelay` resource, a pop-up window displays text that describes the command for that icon.

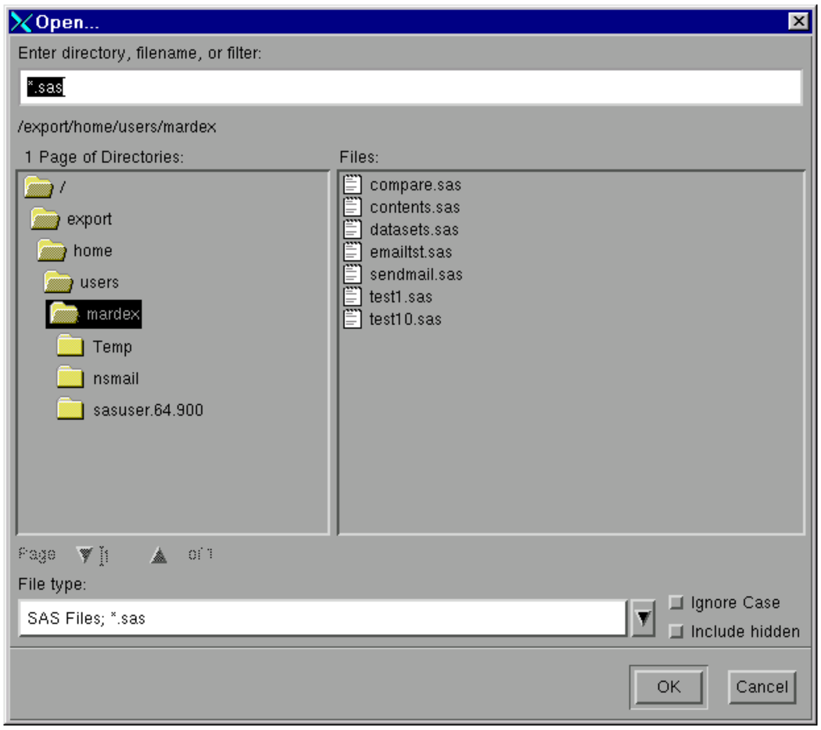
Opening Files in UNIX Environments

Opening the Open Dialog Box

Opening the Dialog Box

The Open dialog box enables you to select files from the host file system. To open this dialog box, select **File ⇨ Open**.

Display 7.9 *Open Dialog Box*



Description of the Open Dialog Box Options

The following table describes the options found in the Open dialog box.

Table 7.3 *Options in the Open Dialog Box*

Option	Description
Enter directory, name or filter	is where you can enter the name of the directory, file, or file filter (file type) that you want to open. The directory shown in the **Filter** field is the currently selected directory. You can change this directory either by selecting a name from the **Page of Directories** list or by entering the new name directly into the field. The dialog box displays non-readable directories with a different icon. To display a list of all the files in a directory, enter the asterisk (*) wildcard in the **Filter** field or select **All Files; *** as the file type.
Page of Directories	contains the names of the directories specified in the **Filter** and **Page** fields.
Files	contains the files in the selected directory that match the filter specified.
Page	enables you to change the directories that are listed in the **Page of Directories** list. A new page is defined when the number of entries in the **Page of Directories** list exceeds twice the screen height. To change pages, use the right or left arrows next to the **Page** field.
File type	enables you to select the type or types of files to be shown in the **Files** list. You can display a list of possible file filters by selecting the down arrow next to the field. Click on a file filter to select it.
Ignore Case	specifies that both uppercase and lowercase names be included in the display. (If you select **All Files; *** as the filter, both uppercase and lowercase names are displayed if you select **Ignore Case**.)
Include hidden	includes or excludes hidden files and directories from the graphical display.

Specifying the Initial Filter and Directory Using SAS Resources

You can specify the initial filter in the **File type** field by assigning a value to the `SAS.pattern` resource. However, the Open dialog box retains its filter between invocations, so the `SAS.pattern` resource applies only to the first invocation of the Open dialog box. You can also use the `SAS.directory` resource to specify the directory that you want when you first invoke the Open dialog box.

For more information about specifying SAS resources, see "Overview of X Resources" on page 164.

Using Regular Expressions in Filenames

Everything that you enter into the Open dialog box is treated as a regular expression. When you are opening or saving a file and you want to use a regular expression special character as part of the filename, precede the character with a backslash (\). For example, to write to a file named $Jan, enter \$Jan as the filename.

For more information about regular expressions, refer to UNIX man page 5 for regexp:

```
man 5 regexp
```

Changing Your Working Directory in UNIX Environments

What Is Your Working Directory?

The working directory is the operating system directory to which many SAS commands and actions apply. By default, SAS uses the current directory as the working directory when you begin your SAS session.

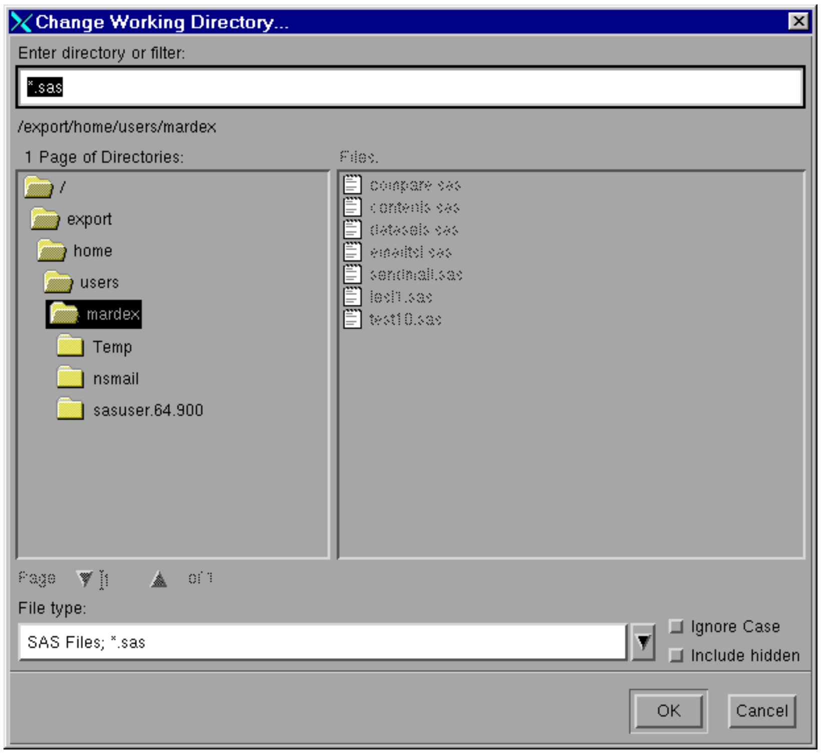
Changing Your Working Directory

You can change the working directory during your SAS session. You can use the Change Working Directory dialog box to select a new directory, or you can use the X command, the X statement, the CALL SYSTEM routine, or the %SYSEXEC macro statement to issue the change directory (`cd`) command. For information about the X command and statement, the CALL SYSTEM routine, and the %SYSEXEC macro statement, see “Executing Operating System Commands from Your SAS Session” on page 15.

The Change Working Directory Dialog Box

To open the Change Working Directory dialog box, issue the DLGCDIR command or select **Tools** ⇨ **Options** ⇨ **Change Directory**.

Display 7.10 *The Change Working Directory Dialog Box*



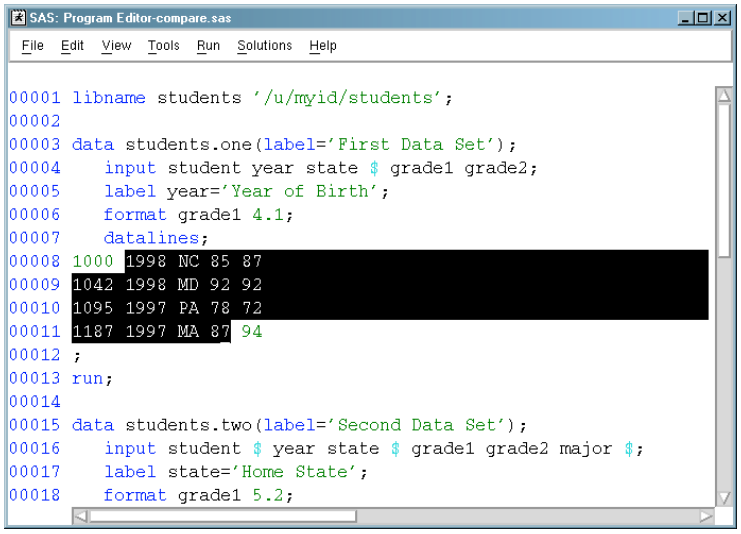
The Change Working Directory dialog box works exactly the same as the Open dialog box, except that you cannot select a file from the list. For an explanation of the options in the Change Working Directory dialog box, see "Description of the Open Dialog Box Options" on page 150.

Selecting (Marking) Text in UNIX Environments

Difference between Marking Character Strings and Blocks

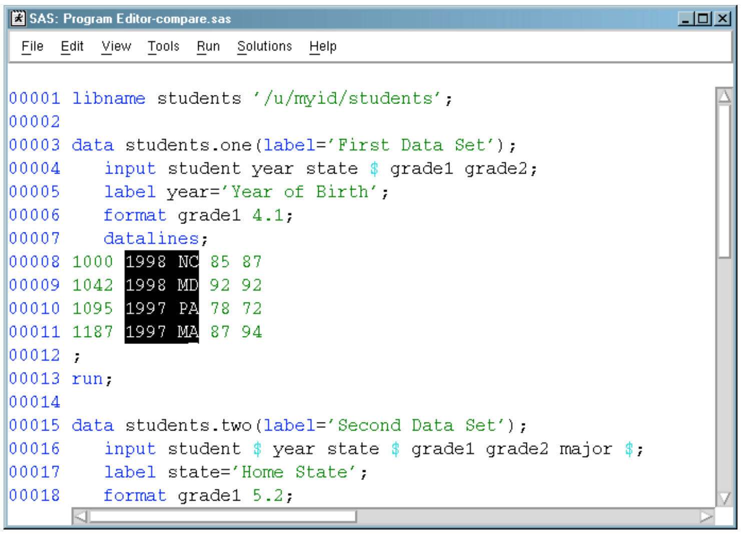
When you select text in a SAS window, you can select character strings or blocks. Character strings include the text in successive columns of one or more rows, as shown in the following display. Blocks are rectangular blocks that include the same columns from successive rows, as shown in Display 7.12 on page 154.

Display 7.11 *Strings That Are Marked*


SAS: Program Editor-compare.sas
File Edit View Tools Run Solutions Help

```
libname students '/u/myid/students';

data students.one(label='First Data Set');
   input student year state $ grade1 grade2;
   label year='Year of Birth';
   format grade1 4.1;
   datalines;
1000 1998 NC 85 87
1042 1998 MD 92 92
1095 1997 PA 78 72
1187 1997 MA 87 94
;
run;

data students.two(label='Second Data Set');
   input student $ year state $ grade1 grade2 major $;
   label state='Home State';
   format grade1 5.2;
```

Display 7.12 *Blocks That Are Marked*


SAS: Program Editor-compare.sas
File Edit View Tools Run Solutions Help

```
libname students '/u/myid/students';

data students.one(label='First Data Set');
   input student year state $ grade1 grade2;
   label year='Year of Birth';
   format grade1 4.1;
   datalines;
1000 1998 NC 85 87
1042 1998 MD 92 92
1095 1997 PA 78 72
1187 1997 MA 87 94
;
run;

data students.two(label='Second Data Set');
   input student $ year state $ grade1 grade2 major $;
   label state='Home State';
   format grade1 5.2;
```

Techniques for Selecting Text

Select Text with the Mouse

To select your text, complete the following steps:

1. Position the cursor at the beginning of the text that you want to mark.
2. Press and hold the left mouse button. If you want to select a block instead of a string, press and hold the CTRL key before you press the left mouse button.
3. Drag the mouse pointer over the text that you want to mark.
4. Press and hold down the ALT key (or EXTEND char key or META key, depending on your keyboard) while you release the mouse button. The marks that are generated by the mouse are called drag marks.

To extend an area of marked text, press and hold the SHIFT key, and use the left mouse button and the ALT key (and the CTRL key, if you are marking a block) to mark the new ending position. To unmark the selected text, press the mouse button anywhere in the window.

Select Text with the MARK Command

You can issue the MARK command from the command line, or you can assign it to a function key. With the MARK command, you can select more than one area of text in the same window at the same time. For more information about the MARK command, see the online SAS Help and Documentation.

To select your text, complete the following steps:

1. Position the cursor at the beginning of the text that you want to mark.
2. Issue the MARK command. If you want to select a block instead of a string, add the BLOCK argument to the MARK command.
3. Move the cursor to the end of the text that you want to mark.
4. Issue the MARK command a second time.

To unmark the selected text, issue the UNMARK command.

Select Text by Using the Edit Menu

To select your text using the **Edit** menu, complete the following steps:

1. Position the cursor at the beginning of the text that you want to mark.
2. Select **Edit** ⇨ **Select**.
3. Position the cursor at the end of the text that you want to mark.
4. Press the left mouse button.

To unmark the selected text, select **Edit** ⇨ **Deselect**.

Copying or Cutting and Pasting Selected Text in UNIX Environments

Techniques for Copying or Cutting and Pasting Selected Text

After you have marked text, you can copy or cut the text and paste it in another location.

- To copy text, select the Copy icon from the ToolBox, issue the STORE or WCOPY command, or select **Edit** ⇨ **Copy**.
- To cut text, select the Cut icon from the ToolBox, issue the CUT or WCUT command, or select **Edit** ⇨ **Cut**.
- To paste the cut or copied text, select the Paste icon from the ToolBox, issue the PASTE or WPASTE command, or select **Edit** ⇨ **Paste**.

For more information about the CUT, PASTE, and STORE commands, refer to online SAS Help and Documentation.

How SAS Uses the Automatic Paste Buffer

When you end a drag mark by releasing the mouse button without holding down the ALT key, SAS performs an end-of-mark action that might automatically generate a STORE command to save the contents of the mark into a SAS paste buffer. If the STORE command is generated automatically, you do not have to explicitly copy the text before you paste it.

Disabling the Automatic Paste Buffer

You can disable the automatic paste buffer in the following ways:

- Set the `SAS.markPasteBuffer` resource.
- Deselect **Automatically store selection** on the **Editing** tab in the Preferences dialog box: **Tools** ⇨ **Options** ⇨ **Preferences**.

For more information, see "Customizing Cut and Paste in UNIX Environments" on page 203.

Copying and Pasting Text between SAS and Other X Clients

You can cut or copy and paste text between X clients if you associate the default SAS paste buffer with a paste buffer specific to X. For example, if you associate the default SAS paste buffer with the paste buffer, you can copy and paste text between xterm windows and SAS windows. To associate the SAS buffer with an X buffer, specify the `SAS.defaultPasteBuffer` resource:

```
SAS.defaultPasteBuffer:  XTERM
```

For more information about using paste buffers, see "Customizing Cut and Paste in UNIX Environments" on page 203.

Using Drag and Drop in UNIX Environments

Difference between Default and Non-Default Drag and Drop

The SAS windowing environment on UNIX offers two types of drag and drop: default and non-default. Default drag and drop enables you to move text from one place to another. Non-default drag and drop enables you to choose whether to move or copy the text, submit the text if you are dragging SAS code, or cancel the drag and drop operation. With default drag and drop, you can drag text between SAS windows in different SAS sessions and between SAS windows and other Motif applications, such as Netscape, that support drag and drop. Non-default drag and drop is available only between windows in the same SAS session.

Limitations of Drag and Drop in UNIX

Under UNIX, you cannot drag and drop files or RTF (Rich Text Format) text.

How to Drag and Drop Text

To drag and drop text, first mark the text in one of the ways described in "Selecting (Marking) Text in UNIX Environments" on page 153. To use default drag and drop, use the middle mouse button to drag the text where you want it. To use non-default drag and drop, press and hold the ALT (or EXTEND CHAR) key before you release the mouse button.

Searching for and Replacing Text Strings in UNIX Environments

What Are the Find and Replace Dialog Boxes?

The Find and Replace dialog boxes enable you to search for and replace strings in SAS text editor windows such as the Program Editor, the SCL editor, or NOTEPAD.

Opening the Find Dialog Box

To search for a string, open the Find dialog box by issuing the DLGFIND command or by selecting **Edit** ⇨ **Find**.

Description of the Find Dialog Box Options

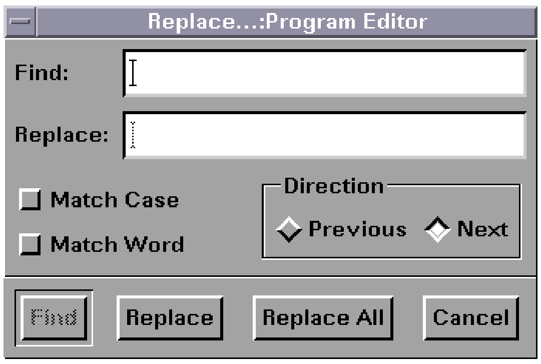
The Find dialog box works like the Replace dialog box, except it does not have the **Replace** field or the **Replace** and **Replace All** buttons.

For a description of the options in the Find dialog box, see "Description of the Replace Dialog Box Options " on page 158.

Opening the Replace Dialog Box

To replace one text string with another, open the Replace dialog box by issuing the DLGREPLACE command or by selecting **Edit** ⇨ **Replace**.

Display 7.13 *Replace Dialog Box*



Description of the Replace Dialog Box Options

To find a character string, enter the string in the **Find** field, and click **Find**. To change a character string, enter the string in the **Find** field, enter its replacement in the **Replace** field, and click **Replace**. To change every occurrence of the string to its replacement string, click **Replace All**.

You can tailor your find or replace operation using the following buttons:

Match Case
tells the search to match the uppercase and lowercase characters exactly as you entered them.

Match Word
searches for the specified string delimited by space, end-of-line, or end-of-file characters.

Previous
searches from the current cursor position toward the beginning of the file.

Next
searches from the current cursor position toward the end of the file.

Sending Mail from within Your SAS Session in UNIX Environments

Default E-mail Protocol in SAS

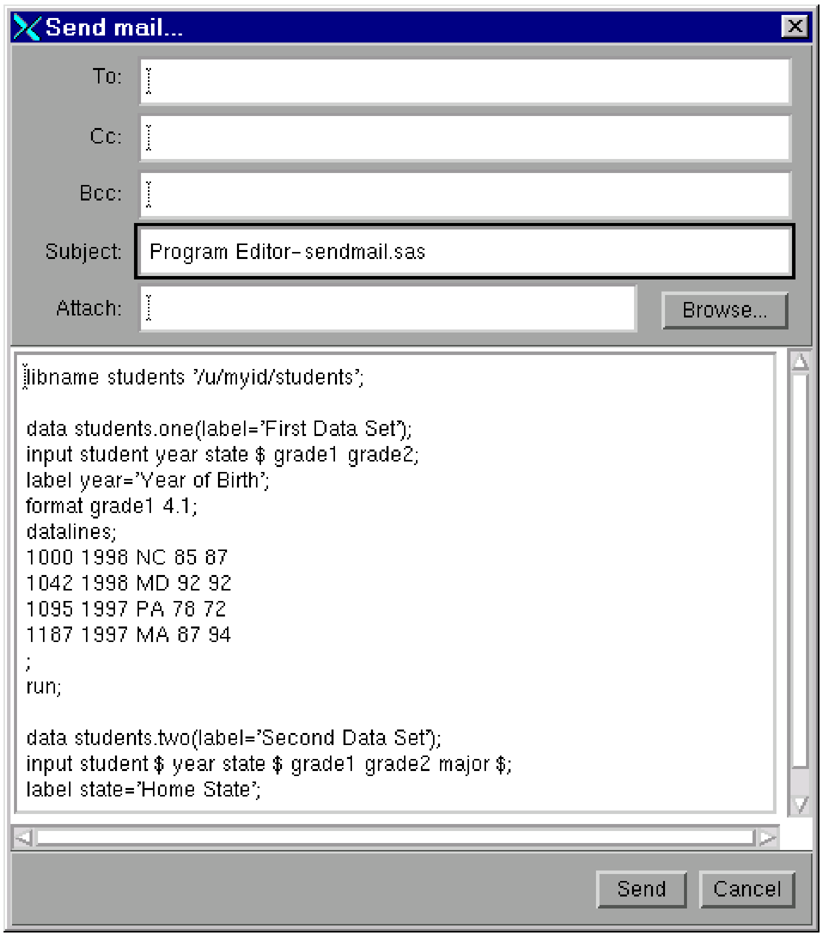
By default, SAS uses SMTP (Simple Mail Transfer Protocol) to send e-mail from within your SAS session. You can use the EMAILSYS system option to specify which script or protocol you want to use for sending electronic mail. For more information, see "EMAILSYS System Option: UNIX" on page 379.

For more information about the SMTP e-mail interface, see *SAS Language Reference: Concepts*.

What Is the Send Mail Dialog Box?

The Send Mail dialog box enables you to send e-mail without leaving your current SAS session. To invoke the dialog box, issue the DLGSMAIL command or select **File** ⇨ **Send Mail**.

Display 7.14 Send Mail Dialog Box



Subject: Program Editor- sendmail.sas

```
libname students '/u/myid/students';

data students.one(label='First Data Set');
input student year state $ grade1 grade2;
label year='Year of Birth';
format grade1 4.1;
datalines;
1000 1998 NC 85 87
1042 1998 MD 92 92
1095 1997 PA 78 72
1187 1997 MA 87 94
;
run;

data students.two(label='Second Data Set');
input student $ year state $ grade1 grade2 major $;
label state='Home State';
```

Sending E-mail by Using the Send Mail Dialog Box

To send e-mail, complete the following steps as needed:

1. Enter the IDs of the e-mail recipients in the **To**, **Cc**, and **Bcc** fields. Separate multiple addresses with either spaces or commas.
2. Edit the entry in the **Subject** field as needed.
3. Enter the name of the file that you want to send in the **Attach** field. Separate multiple filenames with spaces. You can also use **Browse** to select a file.

   Note: Some external scripts do not support sending e-mail attachments.
4. Type your message in the message area or edit the contents grabbed from the active SAS text window.
5. Click **Send**.

To cancel a message, click **Cancel**.

Sending the Contents of a Text Window

You can e-mail the contents of an active SAS text window (such as the Program Editor or the Log) by using the Send mail dialog box. To open the Send mail dialog box, select

File ⇨ **Send Mail**. SAS automatically copies the contents of the active SAS window and includes the text in the body of your e-mail. You can change or add to the e-mail message in the Send mail dialog box.

If you do not want to include the contents of the active SAS window in your message, select **Edit** ⇨ **Clear All** before invoking the Send mail dialog box.

Sending the Contents of a Non-Text Window

To send the contents of a non-text window (such as a graph generated by SAS/GRAPH or an image from your PROC REPORT output), select **File** ⇨ **Send Mail** from the active SAS window. SAS automatically copies the image data to a temporary file and enters that filename into the **Attach** field of the Send mail dialog box. To change the default file type for this temporary file, see "Changing the Default File Type" on page 160 .

SAS only copies the portion of the image that is visible in the active window, along with the window frame and title. This behavior is similar to using the DLGSCRDUMP command. For more information, see "DLGSCRDUMP Command: UNIX" on page 230.

If you do not want to attach this image to your e-mail, clear the contents of the **Attach** field.

Note: Some external scripts do not support sending e-mail attachments.

Changing the Default File Type

You can change the default file type for the temporary file that SAS creates by using the Preferences dialog box. To open the Preferences dialog box, follow these steps:

1. Select **Tools** ⇨ **Options** ⇨ **Preferences**.
2. On the **DMS** tab in the **Image type for Email attachments** box, select one of the following file types:
   - Portable Network Graphics (.png)
   - Graphics Interchange Format (.gif)
   - Tagged Image File Format (.tif)

Configuring SAS for Host Editor Support in UNIX Environments

Requirements for Using a Host Editor

SAS supports the use of a host text editor with the Motif interface, so you can use an editor such as vi or Emacs with your SAS session. There is no host editor set as the default host editor, so you must specify one to use this feature. Host editor support requires the use of the motifxsassm client. For more information, see "The SAS Session Manager (motifxsassm) in UNIX" on page 143.

Invoking and Using Your Host Editor

How to Open and Use the Host Editor

To use your host text editor with SAS, complete the following steps:

1. Specify the command required to invoke your editor with the EDITCMD system option.
2. Invoke the editor as needed with the HOSTEDIT command.

The HOSTEDIT command passes data from a SAS window to the host editor. When you save data in the host editor, the data is copied back into the SAS window if the window can be written to.

After you return to the SAS text editor window, you can issue the UNDO command to undo all of the changes that you made with your host editor. You must issue the UNDO command a second time to return to the state of the window before the HOSTEDIT command was issued. If you issue the HOSTEDIT command in a read-only window, you can save your editing changes to an external file, but the SAS text editor window remains unchanged.

For more information, see “EDITCMD System Option: UNIX” on page 378 and “HOSTEDIT Command: UNIX” on page 234.

Example 1: Invoking SAS to Use xedit with the HOSTEDIT Command

Some systems have an X-based editor installed that is called xedit. If you want to use xedit with the HOSTEDIT command, you can invoke SAS with the following command:

```
sas -editcmd  '/usr/local/bin/xedit'
```

Example 2: Invoking SAS to Use vi

The vi editor is a terminal-based editor that requires a terminal window. The xterm client's **`-e`** option runs a program when the xterm client is invoked. To use the EDITCMD option to display an xterm client in conjunction with vi, invoke SAS as follows:

```
sas -editcmd '/usr/bin/X11/xterm -e /usr/bin/vi'
```

Troubleshooting the Transfer of Text Attributes

Text attributes, such as color and highlighting, are not transferred between a host editor window and a SAS text editor window. Issue the HEATTR ON command to display a dialog box that will warn you if you are editing text with highlighting and color attributes that will be removed by the host editor. This dialog box prompts you to continue or abort the HOSTEDIT command. Specify HEATTR OFF to suppress this dialog box.

Getting Help in UNIX Environments

The **Help** menu is always available within your SAS session. Here are descriptions of the Help topics that are available from the **Help** menu:

Using This Window
provides help information that is relevant to the active window. You can access the same information by clicking the **Help** button or pressing the F1 key.

SAS Help and Documentation
provides tutorials and sample programs to help you learn how to use SAS, comprehensive documentation for all products installed at your site, and information about contacting SAS for additional support.

Note: If you set the block unrequested pop-up windows option in your browser's Preferences dialog box, then the online SAS Help and Documentation might not display.

Getting Started with SAS Software
opens a tutorial that will help you get started with SAS.

SAS on the Web
provides links to useful areas on the SAS Web site, including the customer support center, frequently asked questions, sending feedback to SAS, and the SAS home page. (See the `Technical Support Web site`.)

About SAS 9
opens the About SAS 9 dialog box, which provides information about SAS software, your operating environment, and Motif.

Chapter 8

Customizing the SAS Windowing Environment

Overview of Customizing SAS in X Environment

The SAS windowing environment supports the use of X-based graphical user interfaces (GUIs). In UNIX environments, SAS provides an X Window System interface that is based on the Motif style. For more information about SAS in the X environment, see "Description of SAS in the X Environment" on page 140.

You can customize your working environment by using X resources.

Overview of X Resources

Introduction to X Resources

X clients usually have characteristics that can be customized; these properties are known as X resources. Because SAS functions as an X client, many aspects of the appearance and behavior of the SAS windowing environment are controlled by X resources. For example, X resources can be used to define a font, a background color, or a window size. The resources for an application, such as SAS, are placed in a resource database.

SAS functions correctly without any modifications to the resource database. However, you might want to change the default behavior or appearance of the interface. There are several ways to specify your customizations. Some methods modify all SAS sessions displayed on a particular X server. Some methods affect all SAS sessions run on a particular host. Other methods affect only a single SAS session.

If you need more information about X Window System clients and X resources, see the documentation provided by your vendor.

Syntax for Specifying X Resources

A resource specification has the following format:

resource-string: *value*

The *resource string* usually contains two identifiers and a separator. The first identifier is the client or application name (SAS), the separator is a period (.) or asterisk (*) character, and the second identifier is the name of the specific resource. The *value* given can be a Boolean value (True or False), a number, or a character string, depending on the resource type.

The application name and resource name can both specify an instance value or a class value. A specification for a class applies to a larger scope than a single instance.

The following are sample resource specifications:

```
SAS.startSessionManager: True
SAS.maxWindowHeight: 100
SAS.awsResizePolicy: grow
```

See your X Window System documentation for more information about resource specifications.

Methods for Customizing X Resources

The following list describes the methods that you can use to customize X resources.

- Use the Font dialog box, the Preferences dialog box, or the Resource Helper to customize your SAS session. All of these tools write X resource definitions out to a location that SAS will read the next time you start a SAS session. For more information about these tools, see "Modifying X Resources through the Preferences Dialog Box" on page 166, "Setting X Resources with the Resource Helper" on page 171, and "Customizing Fonts in UNIX Environments" on page 192.

  Note: The settings that you specify in the Preferences dialog box will override any command line settings.

- Specify session-specific resources by using the **-xrm** option on the command line for each invocation of SAS. For example, the following command specifies that SAS will not display the Confirm dialog box when you exit your SAS session:

  ```
  sas -xrm 'SAS.confirmSASExit: False'
  ```

  You can specify the **-xrm** option as many times as needed. You must specify the **-xrm** option for each resource.

  Note: If you normally invoke SAS with a shell script, you should protect the quotation marks from the shell with the backslash (\) character:

  ```
  sasscript -xrm \'SAS.confirmSASExit: False\'
  ```

- Add resource definitions to a file in your home directory. If you place resources in a file that X Toolkit normally searches for when applications are invoked, these resources will be loaded when you invoke SAS. For information about where the X Toolkit searches for resources, see the documentation for the X Window System.

  You can also add resources to the resource database after SAS has initialized by running the **xrdb** utility. For example, the following command merges the definitions in the MyResources file into the resource database:

  ```
  xrdb -merge myresources
  ```

- Create a subdirectory for storing resource definitions. (This subdirectory is usually named **app-defaults**.) Set the XUSERFILESEARCHPATH environment variable to the pathname of this subdirectory. You can use **%N** to substitute an application class name for a file when specifying the XUSERFILESEARCHPATH environment variable. Specify the definition for this environment variable in the initialization file for your shell (for example, the $HOME/.login, $HOME/.cshrc, or $HOME/.profile files), to ensure that the XUSERFILESEARCHPATH variable is defined for each shell that is started.

Create a file called **`SAS`** in the subdirectory identified by XUSERFILESEARCHPATH. Include your resource definitions in this file.

Note: Alternatively, you could set the XAPPLRESDIR environment variable to the pathname of the subdirectory that stores your resource definitions. The XAPPLRESDIR and XUSERFILESEARCHPATH environment variables use a slightly different syntax to specify the location of your resource definitions. The location specified by the XUSERFILESEARCH environment variable takes precedence over the location specified by the XAPPLRESDIR variable. For more information, see the UNIX X man page.

- If you want the customized resource definitions to be used for all users on a particular host, create a file called **`SAS`** to contain your resource definitions, and store this file in the system **`app-defaults`** directory.

For more information about X resources, see the X Window System documentation supplied by your vendor or other documentation about the X Window System.

Modifying X Resources through the Preferences Dialog Box

What Is the Preferences Dialog Box?

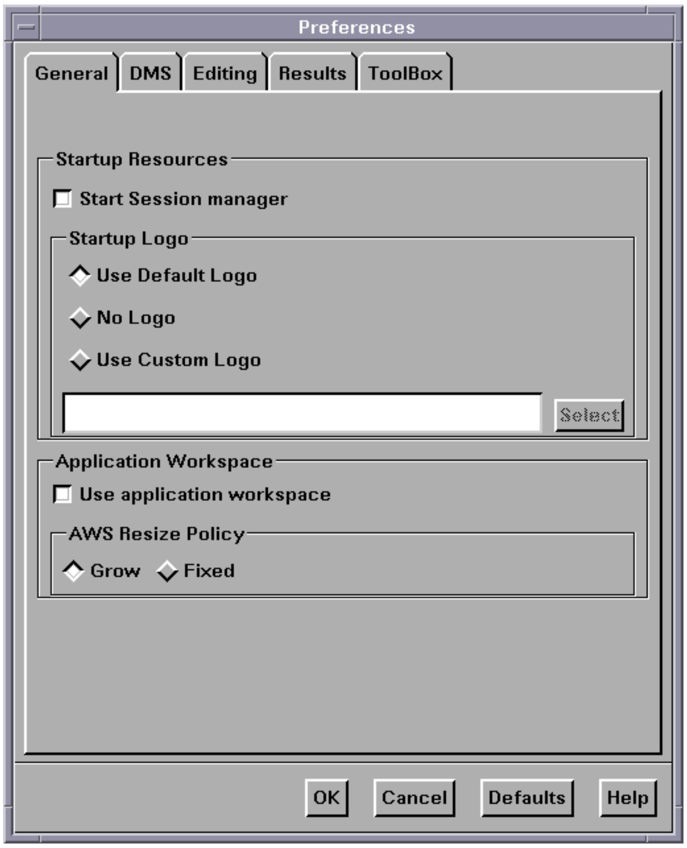
The Preferences dialog box enables you to control the settings of certain X resources. Changes that are made through the Preferences dialog box (with the exception of those resources on the **General** tab) become effective immediately, and the settings are saved in the SasuserPrefs file in your Sasuser directory.

Note: The settings that you specify in the Preferences dialog box will override any command line settings for the current session.

Opening the Preferences Dialog Box

You can open the Preferences dialog box by issuing the DLGPREF command or by selecting **Tools** ⇨ **Options** ⇨ **Preferences**.

Display 8.1 *Preferences Dialog Box*



Description of the Options in the Preferences Dialog Box

Modifying the General Settings

To modify the General settings, select the **General** tab in the Preferences dialog box, and then select from the items in the window:

Start Session manager

specifies whether you want the SAS Session Manager to be started automatically when you start your SAS session. If you want to use your host editor in your SAS session, the SAS Session Manager must be running. The SAS Session Manager enables you to interrupt or terminate your SAS session and minimize and restore all of the windows in a SAS session. See "The SAS Session Manager (motifxsassm) in UNIX" on page 143 and "Configuring SAS for Host Editor Support in UNIX Environments" on page 160 for more information. Clicking the **Start Session manager** box sets the **`SAS.startSessionManager`** resource.

Startup Logo

specifies whether you want SAS to display an XPM file while your SAS session is being initialized and, if so, which file.

If you select **Use Default Logo**, SAS uses the default file for your site. If you select **No Logo**, then no file is displayed. If you select **Use Custom Logo**, then you can either enter the XPM filename directly in the text field or click **Select** to open the File Selection dialog box. Selecting this box sets the **`SAS.startupLogo`** resource.

Use application workspace
confines all windows displayed by an application to a single application work space. Selecting this box sets the **`SAS.noAWS`** resource. You must exit and reopen the windows for changes to this resource to take effect.

Note: In the UNIX operating environment, the application workspace (AWS) is turned on by default. If you are using the EFI window, and you want the window to remember its exact position and size, then you must turn off the AWS. To do this, select **Tools** ⇨ **Options** ⇨ **Preferences**, and deselect the **Use application workspace** box. Be sure to change the AWS back to its default setting when your work in the EFI window is completed.

AWS Resize Policy
controls the policy for resizing AWS windows as interior windows are added and removed. (For more information, see "Workspace and Gravity in a SAS Session" on page 141 and "Window Types" on page 142.)

Grow
The AWS window attempts to grow any time an interior window is grown or moved (to make all of its interior windows visible), but it will not shrink to remove unused areas.

Fixed
The AWS window attempts to size itself to the size of the first interior window and will not attempt any further size changes.

Selecting this box sets the **`SAS.awsResizePolicy`** resource.

Modifying the DMS Settings

To modify the DMS settings, select the **DMS** tab in the Preferences dialog box, and then select from the items in the window:

Use menu access keys
activates menu mnemonics. When mnemonics are turned on, you can select menu items by entering the single, underlined letter in the item. Selecting this box sets the **`SAS.usePmenuMnemonics`** resource.

Confirm exit
displays the Exit dialog box when you exit your SAS session. Selecting this box sets the **`SAS.confirmSASExit`** resource.

Save Settings on Exit
tells SAS to issue the WSAVE ALL command when you exit your SAS session. This command saves the global settings, such as window color and window position, that are in effect for all windows that are currently open. These settings are saved in your Sasuser.Profile catalog. Selecting this box sets the **`SAS.wsaveAllExit`** resource.

Note: For the WSAVE command to work, your window manager must support explicit window placement. See the documentation for your window manager to determine how to configure your window manager. For example, if you are running Exceed, open the Screen Definition Settings dialog box and deselect **Cascade Windows**.

Backup Documents
enables you to specify whether you want SAS to automatically save (at the interval specified by the **`SAS.autoSaveInterval`** resource) the documents that you currently have open. Selecting this box sets the **`SAS.autoSaveOn`** resource.

Image type for Email attachments
specifies the default file type for the temporary file that SAS creates when sending the contents of a non-text window via e-mail. Examples of non-text windows include

a graph generated by SAS/GRAPH or an image from your PROC REPORT output. For more information, see "Sending the Contents of a Non-Text Window" on page 160.

Modifying the Editing Settings

To modify the Editing settings, select the **Editing** tab in the Preferences dialog box, and then select from the items in the window:

Default paste buffer
: defines an alias for the default SAS buffer. The following list describes the paste buffer alias names and the X buffer with which each name is associated.

XPRIMARY
: X primary selection (**PRIMARY**)

XSCNDARY
: X secondary selection (**SECONDARY**)

XCLIPBRD
: X clipboard (**CLIPBOARD**)

XTERM
: exchange protocol used by the xterm client

XCUTn
: X cut buffer where *n* is between 0 and 7, inclusive

Selecting this box sets the **`SAS.defaultPasteBuffer`** resource. See "Controlling Drop-down Menus in UNIX Environments" on page 203 for more information about cut-and-paste buffers.

Automatically store selection
: generates a STORE command every time you mark a region of text with the mouse. Selecting this box sets the **`SAS.markPasteBuffer`** resource.

Cursor
: controls the editing mode in SAS text editor windows. Selecting the **Insert** and **Overtype** boxes sets the **`SAS.insertModeOn`** resource to **True** and **False**, respectively.

Modifying the Results Settings

To modify the results settings, click the **Results** tab in the Preferences dialog box. The items on the **Results** tab affect output that is produced through ODS. (For a complete description of ODS, see the *SAS Output Delivery System: User's Guide*.) Select from the items in this dialog box:

Create Listing
: opens the ODS LISTING destination, which produces monospace output. Selecting this box is equivalent to entering the ODS LISTING SELECT ALL statement.

Create HTML
: opens the ODS HTML destination, which produces output that is formatted in HTML. HTML is the default output type.

Folder
: specifies a destination directory for HTML files. Specifying a directory in this field is equivalent to specifying a directory with the PATH option in the ODS HTML statement.

Use WORK Folder
tells ODS to send all HTML files to your `Work` directory. Selecting this box is equivalent to specifying the pathname of your `Work` directory with the PATH option in the ODS HTML statement.

Style
specifies the style definition to use for HTML output. The style definition controls such aspects as color, font name, and font size. Specifying a style in this field is equivalent to specifying a style with the STYLE option in the ODS HTML statement. You can specify any style that is defined in the \ODS\PREFERENCES\STYLES key in the SAS registry. You can open the SAS registry by issuing the REGEDIT command, or by selecting **Solutions** ⇨ **Accessories** ⇨ **Registry Editor**.

The default style is HTMLBlue.

View results as they are generated
tells SAS to automatically display results when they are generated. If you select this box, make sure that **Password protect HTML file browsing** is deselected.

Password protect HTML file browsing
tells SAS to prompt you for your password before sending HTML files to your browser. If you select this box, make sure that **View results as they are generated** is deselected. Selecting this box sets the `SAS.htmlUsePassword` resource.

Use ODS Graphics
enables you to automatically generate graphs when running procedures that support ODS graphics. **Use ODS Graphics** is turned on by default.

Modifying the SAS ToolBox Settings

The items on the **ToolBox** tab of the Preferences dialog box affect both the ToolBox and the command window. To modify these settings, select the **ToolBox** tab in the Preferences dialog box:

Display tools window
determines whether to display the default toolbox. Selecting this check box sets the `SAS.defaultToolBox` resource.

Display command window
determines whether to display the command window. Selecting this check box sets the `SAS.defaultCommandWindow` resource.

Auto Complete Commands
specifies whether SAS automatically fills in the remaining letters of a command as you type a command in the command window that begins with the same letter as a command that you have entered previously. If both this box and **Save Commands** are selected, then SAS can automatically fill in commands that were entered in previous sessions. Selecting this check box sets the `SAS.autoComplete` resource.

Save Commands
specifies whether SAS saves the commands that you enter in the command window and how many commands are saved. You can specify a number from 0 to 50. If you specify 0, no commands will be saved. If you specify 1 or more, that number of commands is saved in the file `commands.hist` in your Sasuser directory. If this box is selected, then SAS will be able to automatically fill in (see **Auto Complete Commands**) commands that were entered in previous sessions. Selecting this field sets the `SAS.commandsSaved` resource.

Combine windows
combines the ToolBox and command window into one window. The ToolBox and command window are combined by default. Selecting this check box sets the **`SAS.useCommandToolBoxCombo`** resource.

Use arrow decorations
adds arrows to both ends of the combined ToolBox and command window. Selecting this check box sets the **`SAS.useShowHideDecorations`** resource.

Always on top
keeps the ToolBox or the combined ToolBox and command window on top of the window stack. This check box is selected by default, which might cause problems with window managers and other applications that want to be on top of the window stack. If you have such a situation, turn off this feature. Selecting this check box sets the **`SAS.toolBoxAlwaysOnTop`** resource.

Toolbox Persistent
specifies whether the ToolBox that is associated with the Program Editor window stays open when you close the Program Editor. By default, the Program Editor ToolBox stays open whenever you close the Program Editor window. If you deselect this box, then the ToolBox will close if you close the Program Editor. Selecting this check box sets the **`SAS.isToolBoxPersistent`** resource.

The items in the Tools area affect the individual tools in the ToolBox.

Use large tools
controls whether tool icons are displayed as 24x24 or 48x48 pixels. The default is 24x24. Selecting this check box sets the **`SAS.useLargeToolBox`** resource.

Use tip text
specifies whether the ToolTip text is displayed when you position your cursor over a tool in the toolbox. Some window managers might place the toolbox tip behind the toolbox. If the toolbox tip is placed behind the toolbox in your environment, deselect this box. Selecting this check box sets the **`SAS.useToolBoxTips`** resource.

delay
controls the delay in milliseconds before popping up the toolbox tip. Selecting this check box sets the **`SAS.toolBoxTipDelay`** resource. You can enter a value directly into the field or use the arrows to the right of the field to change the value.

Setting X Resources with the Resource Helper

Introduction to the Resource Helper

With Resource Helper, you can customize the key definitions and the colors of the SAS interactive interface. Resource Helper creates SAS resource definitions and stores them in a location where the Resource Manager can find them. See "How the Resource Helper Searches for X Resources " on page 176 for a list of the locations that Resource Helper searches for resource definitions. Resource settings that are saved with Resource Helper will take effect the next time you start a SAS session.

You can start Resource Helper from within a SAS session or from your shell prompt.

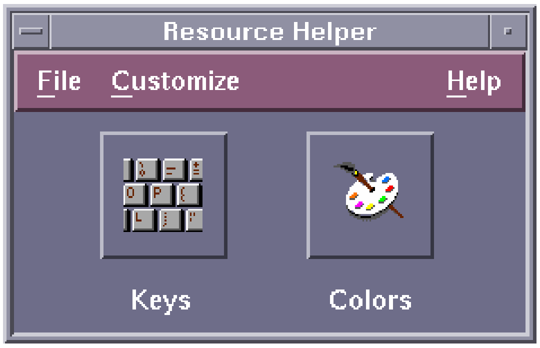
How to Start the Resource Helper

Start the Resource Helper from a SAS Session

Start the SAS Resource Helper from a SAS window by entering the following command on the command line in the command window:

```
reshelper
```

Display 8.2 *Main Window for Resource Helper*



Start the Resource Helper from a Shell Prompt

Resource Helper is installed in the **`/utilities/bin`** subdirectory in the directory where SAS is installed (**`!SASROOT`**). The name of the executable module is **`reshelper`**. For example, if SAS is installed in **`/usr/local/sas93`**, you start Resource Helper by typing the following command:

```
/usr/local/sas93/utilities/bin/reshelper &
```

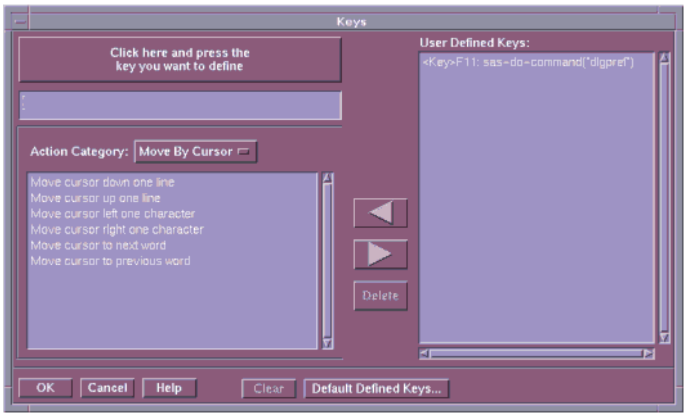
Defining Keys with the Resource Helper

How to Define a Key

To define a key, follow these steps:

1. Start the Resource Helper (see "How to Start the Resource Helper" on page 172) and select the Keys icon.

   Display 8.3 *Keys Window for Resource Helper*



Action Category: Move By Cursor

Move cursor down one line
Move cursor up one line
Move cursor left one character
Move cursor right one character
Move cursor to next word
Move cursor to previous word

User Defined Keys:

<Key>F11: sas-do-command("dlgpref")

Key definitions are divided into several **Action Categories**:

- **Move By Cursor**
- **Move By Field**
- **Edit**
- **Miscellaneous**
- **All Actions**

2. Select **Click here and press the keys you want to define**.
3. Press the key or combination of keys that you want to assign an action to. For example, press F12. If a default SAS translation has already been assigned to the key combination, Resource Helper displays the default translation.
4. Select the action category menu button to open a list of action categories. Select the action category that you want. For example, if you want to define a key to delete the current field, select **Edit** as your **Action Category**. Resource Helper will display a list of actions in that category.
5. Select an action from the list. For example, **Delete current field**. Resource Helper can assign only one action to a translation. If the action that you select requires an argument (such as **`sas-action-routine`**), Resource Helper prompts you for the argument.

   Resource Helper displays the key combination and its new definition:

   ```
   None<Key>F12: sas-delete()
   ```

   Note: If you select the **`sas-action-routine`** sas-function-key action routine, then the key definition is automatically displayed in the Keys window. If you choose another action routine and if you want the definition to appear in the Keys window, you will need to define a window label for the key. See "Syntax of the SAS.keysWindowLabels Resource" on page 188 for information about defining labels in the Keys window.
6. Select the right arrow to add this key translation to the list of **User-Defined Keys**.
7. Click **OK** to exit the Keys window after you have finished defining key translations.
8. To save your translations permanently, from the Resource Helper drop-down menus, select **File** ⇨ **Save Resources**.

To modify a key definition that is already in the **User-Defined Keys** list, select the definition, select the left arrow to remove the definition from the list, and edit the definition.

To delete a definition from **User-Defined Keys**, select it and click **Delete**.

Clear clears the key definition edit window.

Default Defined Keys displays the default key definitions for your system.

Troubleshooting Incorrect Key Definitions

In most cases, using Resource Helper is much easier and faster than defining the resources yourself. However, the X Window System searches for resources in several places, so it is possible for Resource Helper to pick up the wrong key symbol for the key you are trying to define. If you get unexpected results while using Resource Helper, you might need to define your key resources yourself. For more information, see "Defining Key Translations" on page 185.

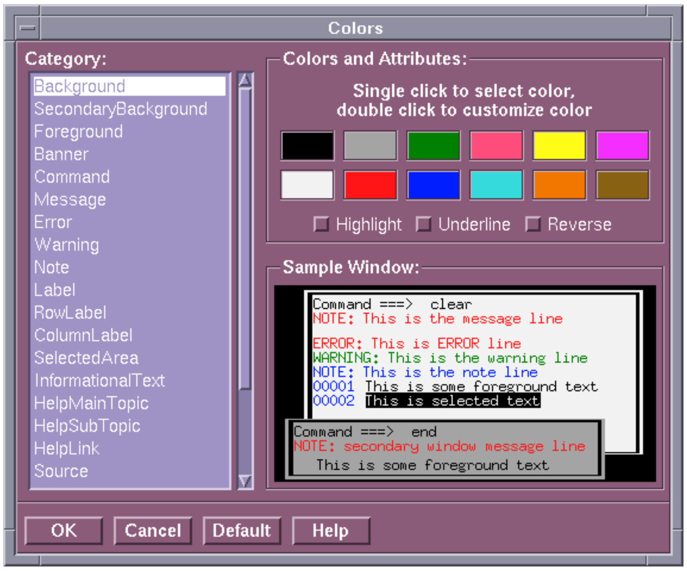
Modifying the Color of a SAS Window Using the Resource Helper

How to Use the Color Window

You can modify the color of part of a SAS window as follows:

1. Start Resource Helper and select the Colors icon.

Display 8.4 *Colors Window for Resource Helper*



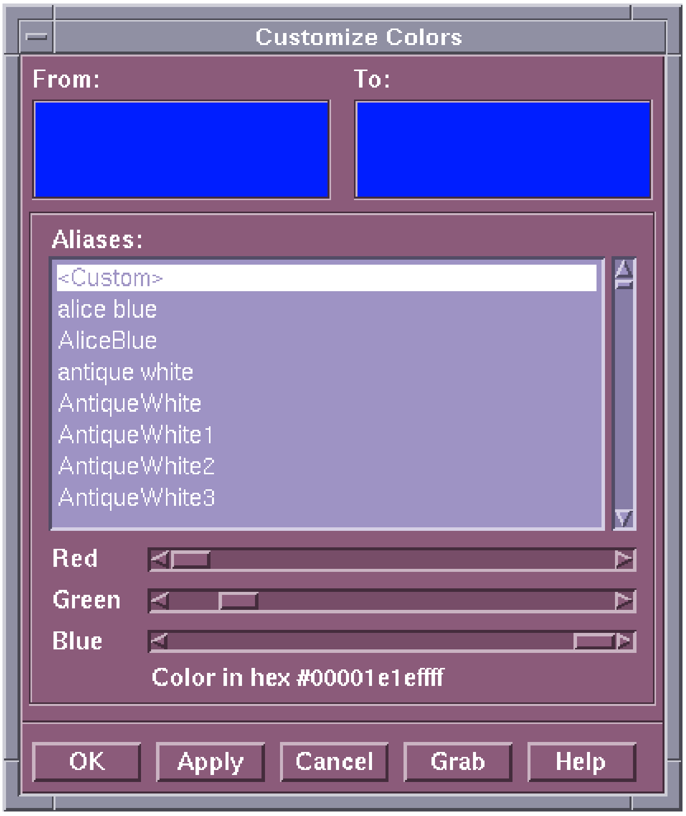
```
Command ===>  clear
NOTE: This is the message line

ERROR: This is ERROR line
WARNING: This is the warning line
NOTE: This is the note line
00001 This is some foreground text
00002 This is selected text

Command ===>  end
NOTE: secondary window message line
  This is some foreground text
```

2. Select a category from the **Category** area.

3. Click a color or attribute in the Colors and Attributes window, or double-click a color to open the **Customize Colors** area, shown in the following display.

   You can also change the attributes of some categories of SAS windows. The attributes options are HIGHLIGHT, UNDERLINE, or REVERSE.

Display 8.5 *Customize Colors Window for Resource Helper*



You can customize a color by doing the following:

- selecting a new **Alias**
- moving the **Red, Blue,** or **Green** sliders
- selecting **Grab** and clicking a color anywhere on your screen

4. Click **OK** to exit the Customize Colors window after you have finished defining your color settings.

The result is displayed in the `Sample Window`. The hexadecimal value of the color is displayed at the bottom of the window.

Example: Change the Color of a SAS Window

The following example shows how to change the color of a SAS window.

1. Double-click **Red** in the Customize Colors window.

   The **From:** display shows the red currently used by the SAS windowing environment.

2. Click **Aquamarine** under **Aliases** and observe the change in the **To:** display.
3. Move the **Red**, **Green**, and **Blue** sliders with your mouse and note the changes in the color of the **To:** display.
4. Click **Apply** and note the difference in the color displayed as **Red** in the Colors area.
5. Click **OK** to save your changes.

Return to the Default Settings

Click **Defaults** to restore your color settings to their default values.

Permanently Save Your Color Settings

To save your color settings permanently, from the Resource Helper drop-down menus, select **File** ⇨ **Save Resources**.

How the Resource Helper Searches for X Resources

The following list describes the locations where the Resource Helper searches for resource definitions and the order in which it searches these locations.

1. Resource Helper loads the resources in the file pointed to by the XENVIRONMENT environment variable. If XENVIRONMENT is not set, Resource Helper loads the resources in the `~/.Xdefaults-`*hostname* file, where *hostname* is the name of the server on which Resource Helper is running.
2. Resource Helper loads the resources defined in the RESOURCE_MANAGER property. If the RESOURCE_MANAGER property is the first location in which Resource Helper finds resources, the RESOURCE_MANAGER property will override any resources that you generate with Resource Helper.

   To determine whether any resources have been defined in your RESOURCE_MANAGER property, issue the following command:

   ```
   xrdb -q | more
   ```

   If no listing is returned, the RESOURCE_MANAGER property does not exist. In this case, Resource Helper loads the resources defined in the `~/.Xdefaults` file.
3. Resource Helper loads the resources in the file pointed to by the XUSERFILESEARCHPATH environment variable.

   You can use `%N` to substitute an application class name for a file when specifying the XUSERFILESEARCHPATH environment variable. For example, to point to `/usr/local/resources` as the location of all the resources for any application, issue the following command in the Bourne or Korn shells:

   ```
   export XUSERFILESEARCHPATH=\
   /usr/local/resources/%N
   ```

   In the C shell, the command is the following:

   ```
   setenv XUSERFILESEARCHPATH \
   /usr/local/resources/%N
   ```

   As a result, when SAS is invoked, the file pointed to by XUSERFILESEARCHPATH is the following:

   ```
   /usr/local/resources/SAS
   ```

   `SAS` is the application class name for SAS.
4. Resource Helper loads the resources in the file specified by the XAPPLRESDIR environment variable. The application's class name is appended to the XAPPLRESDIR environment variable and the resulting string is used to search for resources. For example, you can issue the following command in the Bourne or Korn shells:

   ```
   export XAPPLRESDIR=/usr/local/app-defaults
   ```

   If you do this, then at the next invocation of SAS, the application's class name is appended to the path:

```
/usr/local/app-defaults/SAS
```

In the C shell, the command is the following:

```
setenv XAPPLRESDIR /usr/local/app-defaults
```

5. Resource Helper loads the resources in the file named **`~/SAS`**.
6. Resource Helper loads the resources in the file or substitution specified by the XFILESEARCHPATH environment variable.

   Note: To determine whether an environment variable has been set, you can issue the following command:

   ```
   env|grep <environment_variable>
   ```

7. Resource Helper loads the resources defined in **`/usr/lib/X11/app-defaults`**. Resource Helper does not need to have Write access to this file, but it must be able to read the file and add the SAS resources to a resource file that has Write access. Resource Helper does not generate a warning message if the file is not present or if it cannot read the file.
8. Resource Helper loads the fallback resources that are defined in the SAS code.

Except for the **`/usr/lib/X11/app-defaults`** file, Resource Helper tries to write the new resources to the same directory and file where it first found SAS resources. This location must be a file that has Write access and a directory that has Write access. If Resource Helper cannot write to the file, the SAS resources in that file remain in effect and any new or modified resources generated by Resource Helper will not take effect. If this situation happens, Resource Helper displays an error dialog box that contains the file or directory and suggests a way to fix the problem.

Customizing Toolboxes and Toolsets in UNIX Environments

Techniques for Customizing Toolboxes

You can customize toolboxes in the following ways:

- Through the Preferences dialog box. The Preferences dialog box enables you to customize the appearance and behavior of toolboxes. For information about using the Preferences dialog box, see "Modifying X Resources through the Preferences Dialog Box" on page 166 and "Modifying the SAS ToolBox Settings " on page 170.
- By specifying SAS resources in your resource file. For a description of the SAS resources that affect toolboxes, see "X Resources That Control Toolbox Behavior" on page 177.
- Through the Tool Editor. The Tool Editor enables you to customize the individual tools in a toolbox. For more information, see "Using the Tool Editor" on page 178.

X Resources That Control Toolbox Behavior

You can control the behavior of toolboxes with the following SAS resources:

`SAS.autoComplete: True | False`
specifies whether SAS automatically fills in the remaining letters of a command as you type a command in the command window that begins with the same letter as a command that you have entered previously. The default value is True.

`SAS.commandsSaved: number-of-commands-saved`
specifies whether SAS saves the commands that you enter in the command window and how many commands are saved. You can specify a number from 0 to 50. If you specify 0, no commands will be saved. If you specify 1 or more, that number of commands is saved in the file `commands.hist` in your Sasuser directory. If you specify 1 or more for this resource and `SAS.autoComplete` is True, then SAS can automatically fill in commands that were entered in previous sessions. The default value is 25.

`SAS.defaultToolBox: True | False`
controls opening the default toolbox when SAS is invoked. The default is True.

`SAS.isToolBoxPersistent: True | False`
controls whether the toolbox that is associated with the Program Editor stays open when you close the Program Editor. The default value is True.

`SAS.toolBoxAlwaysOnTop: True | False`
controls whether the toolbox is always on top of the window stack. The default value of True might cause problems with window managers that are not Motif interface window managers or other applications that want to be on top of the window stack. If you have such a situation, set this resource to False.

`SAS.toolBoxTipDelay: delay-in-milliseconds`
sets the delay in milliseconds before displaying the toolbox tip. The default is 750.

`SAS.useCommandToolBoxCombo: True | False`
controls whether the command window and toolbox are joined or separated. The `SAS.defaultToolBox` and `SAS.defaultCommandWindow` resources control whether the toolbox and command window are displayed. If both are displayed, this resource controls whether they are joined or separated. The default value is True.

`SAS.useLargeToolBox: True | False`
controls whether tool icons in the toolbox are displayed as 24x24 pixels or 48x48 pixels. The default is False (24x24 pixels).

`SAS.useShowHideDecorations: True | False`
controls whether the combined command window and toolbox window has arrows at the left and right. You can use these arrows to hide or show portions of the window as they are needed. The default value is False.

`SAS.useToolBoxTips: True | False`
determines whether toolbox tip text is displayed. Some window managers might place the toolbox tip behind the toolbox. If the toolbox tip is placed behind the toolbox in your environment, set this resource to False. The default is True.

Using the Tool Editor

What Is a Toolset?

The Tool Editor enables you to create custom toolsets for your SAS applications. A toolset is a set of predefined tools that is associated with an application. Toolsets make it easier for individual users to customize their application toolboxes. If you create a toolset for an application, you can select **Actions** in the Tool Editor and choose the tools that you want to appear in the toolboxes. You do not have to define the icons, commands, tip text, and IDs for those tools.

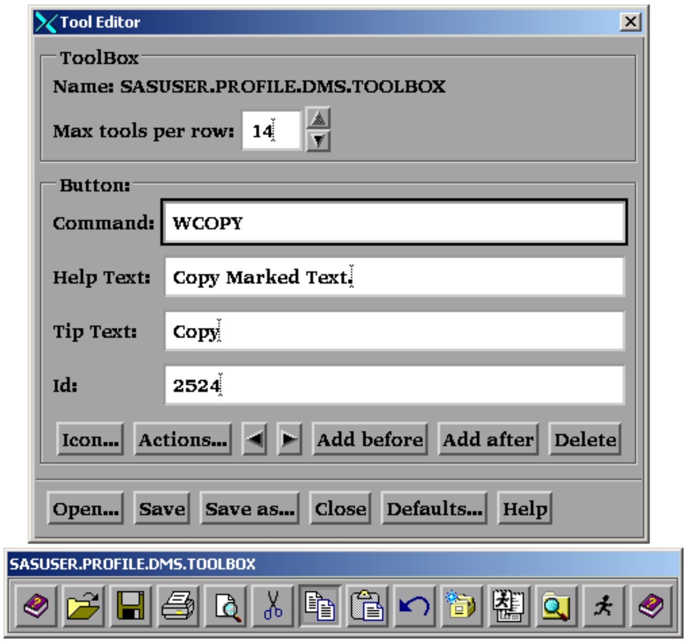
For example, you can define a default toolbox for your application that includes tools for opening files, cutting, copying, and pasting text, and saving files. You can define a toolset that includes those tools and tools for opening the Preferences dialog box, opening the Replace dialog box, and entering the RECALL command. These additional tools will not appear in your toolboxes unless you add them to your toolboxes with the Tool Editor. For more information, see “Changing the Attributes of an Existing Tool” on page 180 and “Create or Customize an Application- or Window-Specific Toolset” on page 183.

Invoking the Tool Editor

You can change the appearance and contents of a toolbox using the Tool Editor. To invoke the Tool Editor, select **Tools** ⇨ **Options** ⇨ **Edit Toolbox**. Alternatively, you can issue the TOOLEDIT command as described in “TOOLEDIT Command: UNIX” on page 238.

The following display shows an example of a Tool Editor dialog box that was opened from the **Tools** menu in the Program Editor window:

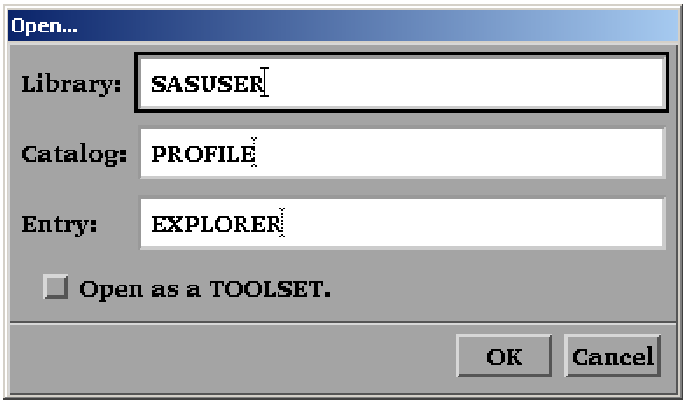
Display 8.6 *Tool Editor Dialog Box*



Name: SASUSER.PROFILE.DMS.TOOLBOX
Max tools per row: 14
Command: WCOPY
Help Text: Copy Marked Text.
Tip Text: Copy
Id: 2524

By default, the Tool Editor enables you to edit the current toolbox. To edit a different toolbox, click the **Open** button in the Tool Editor dialog box. Specify a library, catalog, and entry name for the toolbox that you want to edit. The following display shows the Open dialog box:

Display 8.7 *The Open Dialog Box*



Library: SASUSER
Catalog: PROFILE
Entry: EXPLORER

After You Invoke the Tool Editor

After you invoke the Tool Editor, the toolbox displays in preview mode. In preview mode, clicking a tool icon to select a tool makes that icon the current icon. Its associated commands are displayed in the **Command** field. Attributes in the **Help Text**, **Tip Text**, and **Id** fields might also display, depending on whether this information was added when the tool was created or updated. For more information about the fields and buttons in the Tool Editor dialog box, click the **Help** button.

Changing the Appearance of the Entire Toolbox

The items in the **ToolBox** area of the Tool Editor affect the entire toolbox:

Name
: displays the catalog entry that you are editing. The default toolbox is named SASUSER.PROFILE.DMS.TOOLBOX.

Max tools per row
: specifies how the icons in the toolbox are arranged. The default value creates a horizontal toolbox. One tool per row creates a vertical toolbox.

Changing the Attributes of an Existing Tool

When you open the Tool Editor, the first tool in the toolset is selected, and the attributes for this tool appear in the **Button** area of the Tool Editor dialog box. If you click another icon in the toolset, the Tool Editor displays the attributes of that tool.

Alternatively, you can select a tool from the toolset that is displayed when you click the **Actions** button. When you click **OK** after you select a tool, the attributes in the **Button** area of the Tool Editor are updated to correspond to the new tool.

Note: Clicking the **Actions** button displays a toolset only if a toolset is associated with (has the same entry name as) the toolbox that you are editing. For more information, see “Saving Changes to the Toolbox or Toolset” on page 182.

When you have selected the tool that you want to change, you can then select an attribute field in the Tool Editor and type the changes that you want to make.

To modify the attributes of a tool, follow these steps:

1. From the toolset, select the tool that you want to change.
2. In the **Button** area, select an attribute field associated with a button and change the text as appropriate:

Command
: specifies the command or commands that you want executed when you click the icon. You can use any windowing environment command that is available under UNIX. For information about commands that are valid in all operating environments, see SAS Help and Documentation. Separate commands with a semicolon (;). For example, you could create an icon to open the Change Working Directory dialog box by using the DLGCDIR command.

Help Text
: is used for applications that are designed to be run under Windows. The help text is displayed in the AWS status bar on Windows when a toolbox is ported to and loaded on those windows.

Tip Text
: specifies the text that is displayed when you position the cursor over the icon.

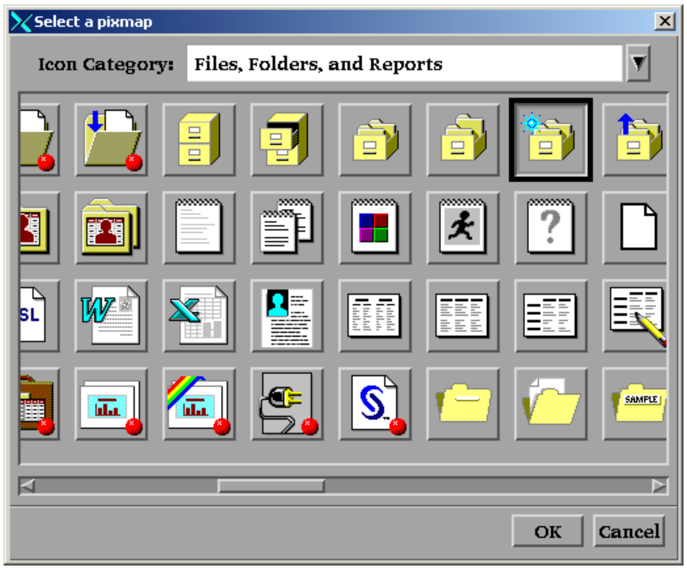
Id
: is useful if you are creating toolboxes for SAS/AF applications. The ID is the identifier of the corresponding menu item in the application. This number is the value assigned to the item in the ID option of the ITEM statement in PROC PMENU. If you specify an ID, then the application can set the state of the PMENU item to match the state of the tool in the toolbox. It can make the PMENU item active or inactive to match whether the tool in the toolbox is active or inactive. If you do not specify an ID, the ID defaults to 0.

3. Change the icon if necessary.

   a. Click the **Icon** button or double-click an icon in the preview toolbox. The Tool Editor opens the Select a pixmap dialog box, which displays the icons that are provided with SAS. These icons are divided into several categories such as SAS windows, data, analysis, numbers and symbols, files, folders, and reports, and so on. To change categories, select the arrow to the right of the **Icon Category** field and select a new category.

   b. Select the icon that you want to use, and then select **OK**.

   The following display shows the Select a pixmap dialog box:



4. Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 182.

Adding Tools to the Toolbox

To add a tool to the toolbox, follow these steps:

1. Select the icon next to where you want to add the new tool.
2. Select **Add before** or **Add after**. The Tool Editor adds a new icon to the toolbox and clears the Button fields.
3. Enter the appropriate information in the Button fields as described in "Changing the Attributes of an Existing Tool" on page 180.
4. Change the attributes of the icon, if necessary, as described in "Changing the Attributes of an Existing Tool" on page 180.
5. Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 182.

Change the Order of the Tools in the Toolbox

To change the position of a tool in the toolbox, select the tool icon, and then click the left or right arrows to move the tool.

Deleting Tools from the Toolbox

To delete a tool from the toolbox, follow these steps:

1. Select the tool that you want to delete.
2. Click **Delete**.
3. Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 182.

Returning to the Default Settings

To return all tools in the current Toolbox to their default settings, click **Defaults**. The Tool Editor asks you to verify your request. Click **Yes**, **No**, or **Cancel**.

Saving Changes to the Toolbox or Toolset

You can save the changes to the catalog entry shown in the **Name** field or create a new toolbox with a different name.

If you are customizing a window- or application-specific toolbox or toolset for your own use, you should save the customized toolbox or toolset in your Sasuser.Profile catalog using the same entry name as the PMENU entry for the window or application. SAS searches for toolboxes and toolsets first in the Sasuser.Profile catalog, and then in the application catalog.

If you are a SAS/AF application developer or site administrator and you are editing a window- or application-specific toolbox that you want to be accessible to all users, you must save the TOOLBOX entry with the same library, catalog, and entry name as the PMENU entry for the window or application. To associate a toolset with a specific toolbox, save the TOOLSET entry with the same library, catalog, and entry name as the TOOLBOX entry. You will need Write permission to the appropriate location. For example, to store a customized toolbox for the graphics editor, the site administrator needs to store the toolbox in SASHELP.GI.GEDIT.TOOLBOX.

Clicking the **Save** button saves the toolbox information to the catalog entry shown in the **Name** field. Clicking the **Save As** button prompts you to enter a different library, catalog, and entry name. You can also choose to save the toolbox as a toolset. If you save the toolbox as a toolset, the entry type will be TOOLSET. Otherwise, the entry type is always TOOLBOX. (Saving a set of tools as a TOOLSET does not change your

TOOLBOX entry.) For more information about toolsets, see "Customizing Toolboxes and Toolsets in UNIX Environments" on page 177 and "Create or Customize an Application- or Window-Specific Toolset" on page 183.

If you click the **Close** or the **Open** button without first saving your changes, the Tool Editor prompts you to save the changes to the current toolbox or toolset before continuing.

After you save the toolbox or toolset, the Tool Editor remains open for additional editing, and the **Name** field changes to the name of the new entry (if you entered a new name).

Creating a New Toolbox

To create an entirely new toolbox, choose from the following methods:

- Edit an existing toolbox using the Tool Editor and then save the toolbox by clicking the **Save as** button as described in "Saving Changes to the Toolbox or Toolset" on page 182.
- Open the Sasuser.Profile catalog in the Explorer window and add a new toolbox by selecting **File** ⇨ **New** ⇨ **Toolbox**.

Create or Customize an Application- or Window-Specific Toolbox

If you are an application developer and want to create or edit an existing application toolbox, follow these steps:

1. Delete any existing TOOLBOX entry in your Sasuser.Profile for the window or application that you want to customize.

   Deleting the copy of the toolbox in your Sasuser.Profile enables you to pick up a copy of the toolbox that is supplied with SAS when you invoke the Tool Editor.
2. Create or edit the application toolbox as described in "Creating a New Toolbox" on page 183 or "Using the Tool Editor" on page 178.
3. Save the edited toolbox as described in "Saving Changes to the Toolbox or Toolset" on page 182.
4. Inform your users that you have changed the window or application toolbox.

   If users want to use the new toolbox, they must delete the corresponding TOOLBOX entry from their Sasuser.Profile. The new toolbox will then be automatically loaded when the window or application is invoked. If a user does not delete the corresponding TOOLBOX entry from their Sasuser.Profile, that copy of the toolbox will be loaded instead of the new toolbox.

The TOOLLOAD and TOOLCLOSE commands are most useful when you are developing SAS/AF applications. You can use the EXECCMDI routine with these commands to enable your application to open and close the toolbox and to give users of your applications access to several toolboxes during the course of their work. For a description of the EXECCMDI routine, see *SAS Component Language: Reference*.

Create or Customize an Application- or Window-Specific Toolset

You define application- or window-specific toolsets in the same way that you create an application- or window-specific toolbox. There are only two differences:

- To create a new toolset, start by defining a toolbox as described in "Creating a New Toolbox" on page 183.
- After you have defined the toolbox, save it as a TOOLSET entry, not as a TOOLBOX entry.

Note: If you are an application developer, ensure that you delete any existing TOOLSET entry for your application as described in "Create or Customize an Application- or Window-Specific Toolbox" on page 183 before you modify your application's toolset.

Customizing Key Definitions in UNIX Environments

Techniques for Customizing Your Key Definitions

There are four ways to customize your key definitions:

- through the Keys window

  To open the Keys window, issue the KEYS command or select **Tools** ⇨ **Options** ⇨ **Keys**. If you change any key definitions through the Keys window for the primary SAS windowing environment windows, the definitions are stored in the Sasuser.Profile catalog in the entry DMKEYS.KEYS. Key definitions for other SAS windows are stored in catalog entries named BUILD.KEYS, FSEDIT.KEYS, and so on.

  See the online SAS Help and Documentation for more information about the KEYS command and the Keys window.

- with the KEYDEF command

  The KEYDEF command enables you to redefine individual function keys:

  ```
  keydef keyname <command|~text-string>
  ```

  For example, if you specify **`keydef F8 dlgpref`**, then the **F8** key will open the Preferences dialog box.

  For more information about the KEYDEF command, see the Base SAS section in the online SAS Help and Documentation.

- through the Resource Helper (reshelper)

  Resource Helper generates SAS resource specifications based on keys and functions that you select. You can use Resource Helper to change the function of any key that is listed in the Keys window. For more information about the Resource Helper, see "Setting X Resources with the Resource Helper" on page 171 and "Defining Keys with the Resource Helper" on page 172.

  In most cases, Resource Helper is much easier and faster than defining the resources yourself. However, because the X Window System searches for resources in several places, it is possible for Resource Helper to pick up the wrong key symbol for the key you are trying to define. Also, unless the action routine that you assign to your keys is the **`sas-function-key`** routine, then Resource Helper does not provide a way to change the key labels in the Keys window. In both of these cases, you will need to define your key resources yourself.

- by defining the **`SAS.keyboardTranslations`** and **`SAS.keysWindowLabels`** resources in your resources file as described in "Defining Key Translations" on page 185.

You can define most of the keys on your keyboard. However, a few keys have dedicated functions that are associated with them. For example, the mouse buttons are dedicated to the cursor and cut-and-paste operations and are not available for user customization.

Defining Key Translations

What Is a Key Translation?

Key customization for the X Window System consists of defining a key sequence and an action to be executed when that key sequence is typed on the keyboard. This customization is known as binding keys to actions; together they are referred to as a translation.

What Is the SAS.keyboardTranslations Resource?

The **`SAS.keyboardTranslations`** resource specifies the set of key bindings that SAS uses in all SAS windows. The default value for the **`SAS.keyboardTranslations`** resource is determined at run time based on the vendor identification string reported by the X server that you are using as the display. These defaults are listed in the files contained in **`!SASROOT/X11/resource_files`**. To modify the default bindings that are supplied by SAS, you must modify the **`SAS.keyboardTranslations`** resource.

Note: The X Toolkit Intrinsics translations that are specified in this resource apply to both the user area and the command line of all SAS windows that are affected by this resource. This resource does not affect windows that are controlled by Motif interface resources, such as the Command window, the Open or Import dialog boxes, and some other drop-down menu dialog boxes.

Steps for Creating a Key Definition

To create a key definition, follow these steps:

1. Determine the keysyms for the keys that you want to define.

   Keysyms are the symbols that are recognized by the X Window System for each key on a keyboard. For more information, see "Determining Keysyms" on page 186.

2. Modify or add the **`SAS.keyboardTranslations`** resource in your resource file to include the definitions of the keys that you want to define.

   Use a keyboard action routine to define which action you want the key to perform. The definition in the right column in the Keys window will no longer control the function of any keys that are defined with a keyboard action routine other than **`sas-function-key`**. The definitions of those keys in the Keys window become labels that have no effect. For more information, see "Syntax of the SAS.keyboardTranslations Resource" on page 187.

3. Modify or add the **`SAS.keysWindowLabels`** resource in your resource file.

   The **`SAS.keysWindowLabels`** resource specifies the set of valid labels that will appear in the Keys window. Modify this resource only if you want to add new labels or modify existing labels in the left column in the Keys window.

   The **`SAS.keysWindowLabels`** resource defines only the mnemonics used in the Keys window. For a specific key to perform an action, you must specify a

SAS.keyboardTranslations definition for the key. For more information, see "Syntax of the SAS.keysWindowLabels Resource" on page 188.

4. Start a SAS session and open the Keys window.
5. In the right column in the Keys window, enter a command name or other description of each key that you have defined.

For examples of key definitions, see "Examples: Defining Keys Using SAS Resources" on page 191.

Determining Keysyms

You can use the **xev** utility to determine the keysyms that are associated with the keys on your keyboard. The **xev** utility is distributed with most UNIX operating systems, but if **xev** is not installed in your operating environment, contact your UNIX system administrator for information about other methods that are available in your UNIX environment. The **xev** utility writes a message for each X event that occurs. The **KeyPress** event specifies the keysym for each key that is pressed.

To define keys, follow these steps:

1. Start the **xev** utility on the X server for which you want to define keys.

   The **xev** client displays a small Event Tester window that lists the X events that occur. (The **xev** client generates a large amount of output, so you might want to save the output to a file for later review. You can issue the UNIX **script** command to save the output to a file.)
2. Give keyboard focus to the Event Tester window by clicking the mouse pointer on the window, if necessary.
3. Press the key that you want to define, and watch for the **KeyPress** event to be listed.

   The listing contains a number of items that are separated by commas. One of the fields in the **KeyPress** event lists the keysym name that is associated with the key that was pressed.

   For example, when the 0 key on the keypad of a Dell PC 105 keyboard is pressed, with the NumLock modifier toggled on, it generates the following output:

   ```
   KeyPress event, serial 32, synthetic NO,
     window 0x1a00001, root 0x5d, subw 0x1a00002,
     time 600120687, (37,41), root:(240,458),
     state 0x10, keycode 90 (keysym 0xffb0, KP_0),
     same_screen YES,
     XLookupString gives 1 bytes: (30) "0"
     XmbLookupString gives 1 bytes: (30) "0"
     XFilterEvent returns: False
   ```

   In this example, the keysym name is **KP_0**.

Note: SAS defines a set of virtual keysyms with the **SAS.defaultVirtualBindings** resource. Virtual keysyms all begin with **osf**, such as **osfPageDown**, **osfClear**, and **osfPrimaryPaste**. If you remap these virtual bindings instead of using the defaults supplied by SAS, you might get unexpected results. If you specify a key translation that does not work, you might be trying to redefine a key that is bound to a virtual keysym. In this case, you must specify the virtual keysym in the **SAS.keyboardTranslations** resource instead of the keysym that is displayed by the **xev** utility. To determine the virtual keysym that is bound to a key, you can start the Resource Helper, click **Keys**, and press the key or key combination that you want to define. Resource Helper will display the virtual keysym name. You can also refer to the key definition files in **`/X11/resource_files`** in the directory where SAS is installed (**`!SASROOT`**) and to the UNIX man pages for VirtualBinding or xmbind.

Syntax of the SAS.keyboardTranslations Resource

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in this topic, optional syntax is shown with square brackets ([]). The angle brackets that are shown in this topic are part of the syntax and should be entered exactly as shown.

Here is the syntax of the **`SAS.keyboardTranslations`** resource:

SAS.keyboardTranslations: #override \

[*modifier*] <Key>*keysym* : *action-routine* \n\

[*modifier*] <Key>*keysym* : *action-routine*

#override
: indicates that this definition should override any existing bindings for the specific keys that you define without affecting any other keys. If you omit the *#override* directive, the new bindings replace all of the default bindings, and none of the other keys on the keyboard will be available.

  Note: For information about the *#augment* and *#replace* directives, see the documentation for the X Window System.

modifier
: can be one of the following:

  - Alt
  - Ctrl
  - Meta
  - Shift
  - Lock
  - Mod1
  - Mod2
  - Mod3
  - Mod4
  - Mod5
  - None
  - blank space

  The list of valid modifiers varies depending on your keyboard. To display a list of valid modifiers for your keyboard, enter the **`xmodmap`** UNIX command. For more information, refer to the UNIX man page for **`xmodmap`**.

<Key>
: is required. It signals the beginning of the keysym.

keysym
: is the key symbol recognized by X for the key that you are defining. For more information, see "Determining Keysyms" on page 186.

action-routine
: is what you want the key to do. You can specify any action routine described in "SAS Keyboard Action Names" on page 188.

\n
: enables the X translation manager to determine where one translation sequence ends and the next one begins. Do not enter \n after the end of the last translation.

\
: prevents the newline character at the end of the line from being interpreted as part of the definition. Using this character is a stylistic convention that allows each translation to be listed on a separate line. Do not enter a backslash after the end of the last translation.

Note: SAS does not prevent you from specifying invalid keys in the `SAS.keyboardTranslations` resource. In some cases, invalid keys will produce warnings in the shell window.

Syntax of the SAS.keysWindowLabels Resource

Note: The square brackets ([]) in the following syntax indicate that *(InternalKeyName)* is optional.

Here is the syntax of the `SAS.keysWindowLabels` resource:

SAS.keyWindowLabels: \

KeyWindowLabel [(*InternalKeyName*)] \n\

KeyWindowLabel [(*InternalKeyName*)]

KeyWindowLabel
: is the label (1 to 8 characters) that you want to appear in the Keys window.

InternalKeyName
: is the character string that is passed to the `sas-function-key` action routine in the corresponding `SAS.keyboardTranslations` key binding. (*InternalKeyName* is used by SAS to correlate Keys window entries to key definitions in the KEYS modules loaded from SAS catalogs or defined in the SAS Keys window.) If the *InternalKeyName* is not specified, SAS uses the *KeyWindowLabel* as the *InternalKeyName*.

\n and \
: serves the same purpose as in the `SAS.keyboardTranslations` resource. For more information, see "Syntax of the SAS.keyboardTranslations Resource" on page 187.

SAS Keyboard Action Names

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in this topic optional syntax is shown with square brackets ([]). The angle brackets that are shown in this topic are part of the syntax and should be entered exactly as shown.

SAS declares a set of keyboard actions during X initialization. You can think of these keyboard actions as simple functions. When the actions are executed, they act on the window that currently has keyboard input focus.

The following list of keyboard actions represents action routines registered by the Motif interface for use with X Toolkit keyboard event translations.

`sas-cursor-down()`
moves the cursor down one line in the SAS window. The cursor does not wrap when it reaches the bottom of the SAS window interior.

`sas-cursor-left()`
moves the cursor left one character in the SAS window. The cursor does not wrap when it reaches the left side of the SAS window interior.

`sas-cursor-right()`
moves the cursor right one character in the SAS window. The cursor does not wrap when it reaches the right side of the SAS window interior.

`sas-cursor-up()`
moves the cursor up one line in the SAS window. The cursor does not wrap when it reaches the top of the SAS window interior.

`sas-delete()`
deletes all text in the current field.

`sas-delete-begin()`
deletes text from the current cursor position to the beginning of the current text field.

`sas-delete-char()`
deletes the character under the text cursor and leaves the cursor in place.

`sas-delete-end()`
deletes text from the current cursor position to the end of the current text field.

`sas-delete-prev-chr()`
deletes the character to the left of the text cursor and moves the cursor back one space.

`sas-delete-prev-word()`
deletes text to the start of the previous word from the current cursor position. If the cursor is in the interior of a word when the action is invoked, the text from the cursor position to the start of the word is deleted.

`sas-delete-word()`
deletes text from the current cursor position to the end of the current or next word.

`sas-do-command()`
accepts one or more text string parameters that are interpreted as SAS commands to be executed when the action is invoked. The action might be invoked with multiple parameters. The parameters are concatenated with semicolon delimiters supplied by the **`sas-do-command`** action between the parameters. The assembled SAS command string is then submitted for execution. For example, the following translation syntax can be used to define a HOME, SUBMIT key sequence for all SAS windowing environment windows:

```
<Key>KP_F3: sas-do-command(HOME;SUBMIT)
```

`sas-function-key("InternalKeyName")`
invokes the SAS commands associated with the function key identified by the *InternalKeyName* label. *InternalKeyName* is the character string (1 to 8 characters long) that is passed to the keysWindowLabels resource. Enclose *InternalKeyName* in quotation marks. For a description of internal key names, see “Defining Key Translations” on page 185.

`sas-home-cursor()`
is the equivalent of the HOME command. It is provided for convenience so that the HOME action could be defined for all SAS windowing environment windows.

sas-insert-char(["InsertionString"])
inserts or overwrites the character entered into the input field under the text cursor. Insert or overstrike behavior is determined by the **sas-toggle-insert** action, which has a mode that is reflected by the text cursor style displayed; the block cursor indicates overstrike mode, and the underline cursor indicates Insert mode. Normally, **sas-insert-char** translates the XKeyEvent into the appropriate character and inserts it at the SAS text cursor location. If you specify the parameter, the text string represented by this parameter is inserted at the SAS text cursor location. White space in the string is interpreted by the X Toolkit as a parameter delimiter unless you enclose the string in double quotation marks. See your X Window System documentation for information about embedding quotation marks in the string parameter. To include an escaped quotation mark, use the following syntax:

```
Shift<Key>KP_1:  sas-insert-char("One\\"1\\"")
```

This syntax produces the text string **One"1"** at the SAS text cursor location.

sas-kp-application()
sets the workstation's numeric keypad to allow function key translations to be reinstated. This action only works for those keypad keys that are bound to **sas-function-key()** actions. Keypad bindings to other actions are not affected by this translation.

sas-kp-numeric()
sets the workstation's keypad to generate numeric characters instead of its previous function key assignment. This action only works for keypad keys that are bound to **sas-function-key()** actions. Keypad bindings to other actions are not affected by this translation.

sas-move-begin()
moves the cursor to the beginning of the current text field.

sas-move-end()
moves the cursor to the end of the current text field.

sas-new-line()
generates an end-of-line event when invoked. This action is context sensitive. If the action is entered on the SAS command line, the text entered will be submitted for execution. If invoked in the SAS application client area, the action depends on the attributes of the text area under the text cursor. In simplest terms, this action is the general line terminator for an input field.

sas-next-field()
advances the SAS application to the next field in the SAS window client area.

sas-next-word()
skips the text cursor forward to the beginning of the next word in the current text field. If **sas-next-word** does not find the beginning of a word in the current text field, it advances to the next SAS application field. If you are typing in the SAS command line area of the window, the cursor will not wrap into the SAS window client area.

sas-page-down()
scrolls the current window contents forward by one page.

sas-page-end()
moves the text cursor to the end of the current page.

sas-page-top()
moves the text cursor to the top of the current page.

`sas-page-up()`
scrolls the window contents backward by one page.

`sas-prev-field()`
returns the SAS application to the previous field in the SAS window client area.

`sas-prev-word()`
skips the text cursor backward to the beginning of the previous word in the current text field. If **`sas-prev-word`** does not find the beginning of a previous word in the current text field, it returns to the end of the previous SAS application field. If you are typing in the SAS command line area of the window, the cursor will not wrap into the SAS window client area.

`sas-to-bottom()`
moves the text cursor to the absolute bottom of the window's text range.

`sas-to-top()`
moves the text cursor to the absolute top of the window's text range.

`sas-toggle-insert()`
switches the associated window line-editing behavior between insert and overstrike modes. This switching applies only to the SAS command line and the SAS window client area. The current mode is indicated by the cursor style in use. The block cursor indicates overstrike mode, and the underline cursor indicates Insert mode.

`sas-xattr-key(<KeyType>[,<KeyParam>])`
processes SAS extended attribute keys. The *KeyType* parameter must be one of the following values: XACOLOR, XAATTR, XACLEAR. For *KeyType* XACOLOR, the 12 DMS color names are valid parameters; for *KeyType* XAATTR, the valid values are HIGHLIGHT, REVERSE, BLINK, and UNDERLINE; for XACLEAR, no parameter is required. The BLINK attribute is not supported in the Motif interface. However, if you specify the BLINK attribute, it will be displayed when the catalog is ported to other operating environments.

Examples: Defining Keys Using SAS Resources

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in these examples, optional syntax is shown with square brackets ([]). The angle brackets that are shown in these examples are part of the syntax and should be entered exactly as shown.

In the following example, the **`sas-do-command`** action routine specifies that the COMMAND command is to override any existing definition for **`KP_0`**.

```
SAS.keyboardTranslations: #override \n\
    None<Key>KP_0: sas-do-command(COMMAND)
```

All other keys retain their current definitions.

The following example binds the key sequence CTRL-K to the KEYS command and specifies that CTRL-D deletes the character under the cursor. Commands entered in the Keys window for CTRL-K and CTRL-D will have no effect.

```
SAS.keyboardTranslations: #override\
    Ctrl<Key>k: sas-do-command(keys)\n\
    Ctrl<Key>d: sas-delete-char()
```

The following example specifies that the key associated with the keysym **`hpClearLine`** performs the command entered beside the **`MyClrLn`** label in the Keys window.

```
SAS.keyboardTranslations: #override \
   <Key>hpClearLine : sas-function-key("ClearLn")
SAS.keysWindowLabels: MyClrLn(ClearLn)
```

The character string that appears inside the parentheses in the **`SAS.keysWindowLabels`** resource must match the string entered as the parameter to the **`sas-function-key`** routine. The label (**`MyClrLn`**) can be any character string, and the keysym **`hpClearLine`** must be a valid keysym for your keyboard.

Customizing Fonts in UNIX Environments

Difference between the System Font and Fonts That Are Used in the Windowing Environment

SAS uses two main types of fonts:

- The system font is used in most dialog boxes and menus. SAS inherits the system font defined by the CDE **`*.systemFont`** resource. If this resource is not defined, SAS uses a Helvetica font.
- DMS fonts are used in SAS windows. You can change the SAS font either through the Fonts dialog box or by specifying the resources in your resources file. The font must be a fixed or monospace font.

Note: It is best to change fonts before invoking any applications. Changing fonts while applications are running might result in unexpected behavior.

How SAS Determines Which Font to Use

SAS determines the normal (not bold) default font as follows:

1. If you saved a font in SASUSER.PROFILE.DMSFONT.UNXPREFS through the Font dialog box, this font is used as the default normal font.
2. If you did not save a font through the Font dialog box, but you set the **`SAS.DMSFont`** resource, SAS uses the font specified by this resource as the default font.
3. If you did not set the **`SAS.DMSFont`** resource, SAS uses any font that matches the pattern *Font, which might be defined or inherited.
4. If you did not specify or inherit any resources matching *Font, but you did set the **`SAS.DMSFontPattern`** resource, SAS uses this resource to determine which font to use. The **`SAS.DMSfontPattern`** resource has no effect if any resources matching *Font are inherited or defined.
5. If no resources were set, SAS chooses a font from the fonts that are available on your X server.

If you did not specify a value for the **`SAS.DMSboldFont`** resource, SAS uses the default normal font to determine the default bold font. If the normal **`SAS.DMSFont`** has an XLFD name associated with it, then SAS selects the matching bold font and loads it. If SAS cannot automatically select or load a bold font, the normal font is also used for the bold font.

In many cases, font names are given aliases so that a shorter name can be used to refer to a font that has an XLFD name associated with it. The name used in determining a bold font is based on the XA_FONT font property for the normal font.

Customizing Fonts by Using the Fonts Dialog Box

Introduction to the Fonts Dialog Box

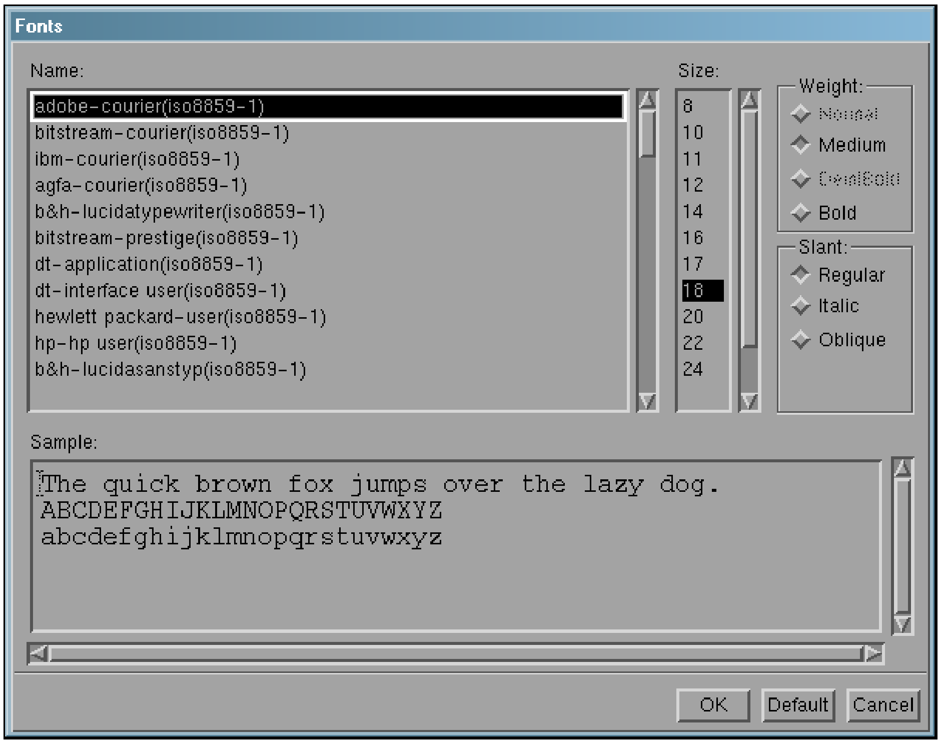
The Fonts dialog box enables you to change the windowing environment font for the entire SAS session. If you change the font, the font that you select is stored in SASUSER.PROFILE.DMSFONT.UNXPREFS and will be used in future SAS sessions.

How to Change the Default Font

You can change the default font by opening the Fonts dialog box. To open the Fonts dialog box, use one of the following methods:

- Issue the DLGFONT command in the command window.
- Select **Tools** ⇨ **Options** ⇨ **Fonts**.

Display 8.8 *Fonts Dialog Box*


Fonts
Name:
adobe-courier(iso8859-1)
bitstream-courier(iso8859-1)
ibm-courier(iso8859-1)
agfa-courier(iso8859-1)
b&h-lucidatypewriter(iso8859-1)
bitstream-prestige(iso8859-1)
dt-application(iso8859-1)
dt-interface user(iso8859-1)
hewlett packard-user(iso8859-1)
hp-hp user(iso8859-1)
b&h-lucidasanstyp(iso8859-1)
Size:
8
10
11
12
14
16
17
18
20
22
24
Weight:
Normal
Medium
DemiBold
Bold
Slant:
Regular
Italic
Oblique
Sample:
The quick brown fox jumps over the lazy dog.
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
OK
Default
Cancel

- Select a font name and, if desired, a size, weight, and slant. (Not all fonts are available in all sizes, weights, or slants.) The **Sample** field shows what the selected font looks like.
- Click **OK** to change the existing font to the selected font.

To return to the default font, click **Default**.

To cancel any changes and exit the Fonts dialog box, click **Cancel**.

Specifying Font Resources

You can customize the fonts that are used in the SAS windowing environment with the following resources:

`SAS.DMSFont: font-name`
specifies the font that you want to be used as the default normal font. The default normal font is Courier.

`SAS.DMSboldFont: font-name`
specifies the font that you want to be used as the default bold font.

`SAS.DMSDBfont: font-name`
specifies the multi-byte normal character set font used by the SAS windowing system for operating environments that support multi-byte character sets.

`SAS.DMSDBboldFont: font-name`
specifies the multi-byte bold character set font used by the SAS windowing system for operating environments that support multi-byte character sets.

`SAS.DMSfontPattern: XLFD-pattern`
specifies an X Logical Font Description, or XLFD pattern that you want SAS to use to determine the windowing environment font. Most fonts in the X Window System are associated with an XLFD, which contains a number of different fields delimited by a hyphen (–) character. The fields in the XLFD indicate properties such as the font family name, weight, size, resolution, and whether the font is proportional or monospaced. Refer to your X Window System documentation for more information about the XLFD and font names used with X.

The *XLFD-pattern* that you specify for **`SAS.DMSfontPattern`** must contain the same number of fields as an XLFD. An asterisk (*) character means that any value is acceptable for that particular field. For example, the following pattern matches any font that has a regular slant, is not bold, is monospaced, and is an iso8859 font:

```
SAS.DMSFontPattern: -*-*-*-r-*--*-*-*-*-m-*-iso8859-1
```

SAS uses the *XLFD-pattern* to choose a font as follows:

- SAS queries the X server for the list of fonts that match the **`SAS.DMSfontPattern`** resource.
- SAS excludes all fonts that have X and Y resolution values different from the current X display, all fonts that have variable character cell sizing (such as proportional fonts), and all fonts that have point sizes smaller than 8 points or larger than 15 points. If this step results in an empty list, SAS chooses a generic (and usually fixed) font.
- The font with the largest point size is chosen from the remaining list.

`SAS.fontPattern: XLFD-pattern`
specifies an XLFD font pattern that describes the candidate fonts used to resolve SAS graphics font requests. Using this pattern allows the user to optimize or control the use of X fonts within the context of various SAS graphics applications. The default value of **`*`** usually does not affect performance to a significant degree. You might want to restrict the font search if you are running SAS on a server with an excessive number of fonts or that is operating in performance-limited environment.

`SAS.systemFont: font-name`
specifies the system font. The SAS font is used in SAS windows. The system font is used in most dialog boxes and menus. SAS typically inherits the system font from the font resources set by the X window environment, such as the Common Desktop

Environment (CDE), or K Desktop Environment (KDE). If the **`*.systemFont`** resource, SAS uses a 12-point Helvetica font.

Specifying Font Aliases

Font Aliases

If your server does not provide fonts to match all of the fonts that are supplied by SAS, you can use font alias resources to substitute the fonts that are available on your system. (Ask your system administrator about the fonts that are available.) Use the following syntax to specify font aliases in your resource file:

```
SAS.supplied-fontAlias: substitute-family
```

supplied-font is the name of the font supplied by SAS. *substitute-family* is the family name of the font that you want to substitute.

CAUTION:

Do not specify a SAS font as a font alias. There might be a conflict if you specify a font supplied by SAS as a font alias, as in the following example:

```
SAS.timesRomanAlias: symbol
```

Assigning this value to a font alias prevents the selection of any symbol fonts through the font selection dialog box, because they are specified as the Times Roman alias.

The following table lists SAS font alias resource names.

Table 8.1 *SAS Font Alias Resources*

Resource Name	Class Name
SAS.timesRomanAlias	TimesRomanAlias
SAS.helveticaAlias	HelveticaAlias
SAS.courierAlias	CourierAlias
SAS.symbolAlias	SymbolAlias
SAS.avantGardeAlias	AvantGardeAlias
SAS.bookmanAlias	BookmanAlias
SAS.newCenturySchoolbookAlias	NewCenturySchoolbookAlias
SAS.palatinoAlias	PalatinoAlias
SAS.zapfChanceryAlias	ZapfChanceryAlias
SAS.zapfDingbatsAlias	ZapfDingbatsAlias

Example: Substitute the Lucida Font for Palatino

Suppose that your system does not have a Palatino font, but has the following Lucida font:

```
b&h-lucida-bold-r-normal-sans-
    10-100-75-75-p-66-iso8859-1
```

To substitute Lucida for Palatino, include the following line in your resource file:

```
SAS.palatinoAlias: lucida
```

Customizing Colors in UNIX Environments

Methods for Customizing the Color Settings in Your SAS Session

SAS provides a default set of colors and attribute settings for the elements of all SAS windows. You can customize the colors in your SAS session in the following ways:

- through Resource Helper (reshelper).

  Resource Helper enables you to customize any color. For more information, see "Setting X Resources with the Resource Helper" on page 171 and "Modifying the Color of a SAS Window Using the Resource Helper" on page 174.

- through the SASCOLOR window, as described in "Customizing Colors by Using the SASCOLOR Window" on page 196.

  You can customize any window element for most SAS windows with the SASCOLOR window.

- with the COLOR command as described in "Syntax of the COLOR Command" on page 197.

  The COLOR command affects only the specified element of the active window. Changes made with the COLOR command override changes entered through any of the other methods described here.

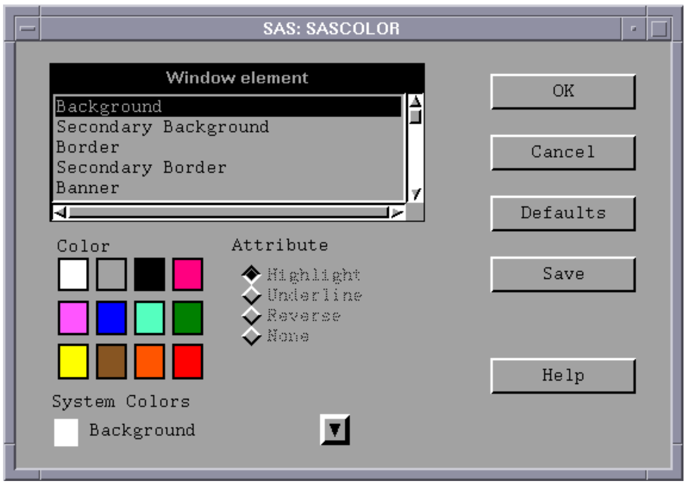
- by entering the color resource specifications yourself.

  You can enter specific RGB values or color names for any of the X resources that control color. For more information, see "Defining Color Resources" on page 198.

Customizing Colors by Using the SASCOLOR Window

You can use the SASCOLOR window to change the color and highlighting of specific elements of SAS windows. To open the SASCOLOR window, issue the SASCOLOR command or select **Tools** ⇨ **Options** ⇨ **Colors**:

Display 8.9 *SASCOLOR Window*



To change a color for a window element, select the element name, and then select color and attribute that you want assigned to the element.

The BLINK attribute is not supported. The HIGHLIGHT attribute causes text to be displayed in bold font.

When you click **Save**, your changes are saved to the catalog entry SASUSER.PROFILE.SAS.CPARMS.

Note: Close and reopen any active windows for new color settings to take effect.

For more information about the SASCOLOR window, see the online SAS Help and Documentation.

Syntax of the COLOR Command

You can use the COLOR command to set the color for specific elements of the active window:

```
color field-type <color|NEXT <highlight>>
```

field-type
: specifies an area of the window such as background, command, border, message, and so on.

color
: specifies a color such as blue (which can be abbreviated B), red (R), green (G), cyan (C), pink (P), yellow (Y), white (W), black (K), magenta (M), gray (A), brown (B), or orange (O).

NEXT
: changes the color to the next available color.

highlight
: can be H (which causes text to be displayed in a bold font), U (underlined), or R (reverse video). The BLINK attribute is not supported.

To save your changes, issue the WSAVE command. The changes are saved to SASUSER.PROFILE.window.WSAVE.

Note: The WSAVE command is not available for all SAS windows. For example, with SAS/FSP changes are saved either through the EDPARMS or the PARMS window.

(To determine whether WSAVE is available for a SAS window, refer to the product documentation.)

For more information about the COLOR and WSAVE commands, see the online SAS Help and Documentation.

Defining Color Resources

Types of Color Resources

Color resources fall into two categories:

- foreground and background definitions

  These resources enable you to customize the RGB values that are used to define the 12 DMS colors. Because each color could be used as either a background or a foreground color, you can specify different RGB values or color names for each color for each usage. For example, you can specify that when blue is used as a foreground color, color #0046ED is used, and when blue is used as a background color, CornflowerBlue is used.

- window element definitions

  These resources, which are referred to as CPARMS resources, enable you to specify which of the 12 DMS colors you want to use for each window element. For example, you can specify that message text is displayed in magenta.

These two types of resources work together. The CPARMS color values use the current foreground and background definitions. For example, the following resources specify that the background of your primary windows will be CornflowerBlue:

```
SAS.blueBackgroundColor: CornflowerBlue
SAS.cparmBackground: DmBlue
```

Specifying RGB Values or Color Names for Foreground and Background Resources

SAS uses **`SAS.systemBackground`**, **`SAS.systemForeground`**, and the resources listed in the following table to determine the colors to be used in its windows.

`SAS.systemForeground: color`
specifies the color for the foreground system color in the SASCOLOR window.

`SAS.systemBackground: color`
specifies the color for the background system color in the SASCOLOR window.

`SAS.systemSecondaryBackground: color`
sets the system secondary background color and specifies the color for the secondary background system color in the SASCOLOR window.

You can specify color names such as MediumVioletRed or RGB values such as #0000FF for all of the foreground and background resources. See your X Window System documentation for information about RGB color values.

The following table lists all of the foreground and background color resources and their class names. All of these resources are of the type String.

Table 8.2 *Foreground and Background Color Resources*

Resource Name	Class Name
SAS.systemForeground	SystemForeground
SAS.systemBackground	SystemBackground
SAS.systemSecondaryBackground	Background
SAS.blackForegroundColor	BlackForegroundColor
SAS.blueForegroundColor	BlueForegroundColor
SAS.brownForegroundColor	BrownForegroundColor
SAS.cyanForegroundColor	CyanForegroundColor
SAS.grayForegroundColor	GrayForegroundColor
SAS.greenForegroundColor	GreenForegroundColor
SAS.magentaForegroundColor	MagentaForegroundColor
SAS.orangeForegroundColor	OrangeForegroundColor
SAS.pinkForegroundColor	PinkForegroundColor
SAS.redForegroundColor	RedForegroundColor
SAS.whiteForegroundColor	WhiteForegroundColor
SAS.yellowForegroundColor	YellowForegroundColor
SAS.blackBackgroundColor	BlackBackgroundColor
SAS.blueBackgroundColor	BlueBackgroundColor
SAS.brownBackgroundColor	BrownBackgroundColor
SAS.cyanBackgroundColor	CyanBackgroundColor
SAS.grayBackgroundColor	GrayBackgroundColor
SAS.greenBackgroundColor	GreenBackgroundColor
SAS.magentaBackgroundColor	MagentaBackgroundColor
SAS.orangeBackgroundColor	OrangeBackgroundColor
SAS.pinkBackgroundColor	PinkBackgroundColor
SAS.redBackgroundColor	RedBackgroundColor

Resource Name	Class Name
SAS.whiteBackgroundColor	WhiteBackgroundColor
SAS.yellowBackgroundColor	YellowBackgroundColor

Defining Colors and Attributes for Window Elements (CPARMS)

You can define the colors and attributes for specific window elements by assigning values to SAS resources known as CPARMS. Each CPARMS resource defines the color and attribute of a specific window element, such as the background in a secondary window or the border of a primary window.

You can specify multiple color and attribute names in the same resource definition, but only the final color and attribute will be used:

```
SAS.cparmResource: DmColorName|DmAttrName\
<+DmColorName|DmAttrName>
```

Resource can be any of the CPARMS resources listed in the following table. All of these resources are of type DmColor, and their default values are dynamic–that is, the default values are determined at run time.

Table 8.3 *SAS CPARMS Resources*

Resource Name	Color and attribute settings	Class Name	Default Color
SAS.cparmBackground	for backgrounds within all primary windows displayed in a SAS session.	CparmBackground	DmWhite
SAS.cparmBanner	for a banner within a window.	CparmForeground	DmBlack
SAS.cparmBorder	for the border of a primary window.	CparmBackground	DmBlack
SAS.cparmByline	for BY lines written to the Output window.	CparmForeground	DmBlue
SAS.cparmColumn	for text labels for column information. You can use this resource within the SAS editor to identify editing lines and in spreadsheet windows to label spreadsheets.	CparmForeground	DmBlue/ Underline
SAS.cparmCommand	for the command data entry field when menus are disabled.	CparmForeground	DmBlack
SAS.cparmData	for general lines written to the Log window or the Output window.	CparmForeground	DmBlack

Resource Name	Color and attribute settings	Class Name	Default Color
SAS.cparmError	for ERROR lines that are written to the Log window or Output window.	CparmForeground	DmRed
SAS.cparmFootnote	for FOOTNOTE lines written to the Output window.	CparmForeground	DmBlue
SAS.cparmForeground	for all text fields within a SAS windowing environment window that can be edited.	CparmBackground	DmBlack
SAS.cparmHeader	for HEADER lines written to the Output window.	CparmForeground	DmBlue
SAS.cparmHelpLink	for links to additional levels of information in the Help system.	CparmForeground	DmGreen/ Underline
SAS.cparmHelpMainTopic	for topic words or phrases in the Help system.	CparmForeground	DmBlack
SAS.cparmHelpSubTopic	for topic words or phrases in the Help system.	CparmForeground	DmBlack
SAS.cparmInfo	for text that is displayed in a window as an aid to the user. For example: `Press Enter to continue`	CparmForeground	DmBlack
SAS.cparmLabel	for text that precedes a widget. For example, the text **`Name:`** in the following example is a label: `Name: ______________`	CparmForeground	DmBlack
SAS.cparmMark	for areas that have been selected for operations such as FIND, CUT, and COPY.	CparmForeground	DmBlack/ DmReverse
SAS.cparmMessage	for the message field.	CparmForeground	DmRed
SAS.cparmNote	for NOTE lines that are written to the Log window or the Output window.	CparmForeground	DmBlue
SAS.cparmSecondaryBackground	for backgrounds in secondary windows.	CparmForeground	DmGray
SAS.cparmSecondaryBorder	for the border of a secondary window.	CparmForeground	DmBlack
SAS.cparmSource	for SAS source lines that are written to the Log window.	CparmForeground	DmBlack

Resource Name	Color and attribute settings	Class Name	Default Color
SAS.cparmText	for text labels for row information. You can use this resource within the SAS editor to identify editing lines and in spreadsheet windows to label spreadsheet rows.	CparmForeground	DmBlue
SAS.cparmTitle	for TITLE lines written to the Output window.	CparmForeground	DmBlue
SAS.cparmWarning	for WARNING lines written to the Log window or the Output window.	CparmForeground	DmGreen

DmColorName can be any one of the following colors:

- DmBLUE
- DmRED
- DmPINK
- DmGREEN
- DmCYAN
- DmYELLOW
- DmWHITE
- DmORANGE
- DmBLACK
- DmMAGENTA
- DmGRAY
- DmBROWN

DmAttrName can be any one of the following attributes:

- DmHIGHLIGHT
- DmUNDERLINE
- DmREVERSE

For example, the following resources specify that all background colors are gray and all foreground colors are black:

```
SAS.cparmBackground: DmGRAY
SAS.cparmForeground: DmBLACK
```

These resources specify that errors should be displayed in red with reverse video, and warnings should be displayed in yellow with reverse video and a bold font:

```
SAS.cparmError: DmRED + DmREVERSE
SAS.cparmWarning: DmHIGHLIGHT + DmYELLOW + DmREVERSE
```

SAS looks for default CPARMS resources in two places:

- If your on-site SAS support personnel entered color and attribute settings in the SASHELP.BASE.SAS.CPARMS catalog entry, then these settings become the default for your site.
- If you saved settings in SASUSER.PROFILE.SAS.CPARMS, then these settings override the settings specified for your site.

Controlling Contrast

During interactive move or stretch operations, such as rubber banding and dragging rectangles in SAS/INSIGHT software, you might find it hard to see the outline of the graphics primitive because of the lack of contrast between the primitive and the background. The XCONTRAST command makes the primitive visible against the background. The rendering performance and the aesthetic appearance of the primitive is compromised for the sake of visibility. You can enter XCONTRAST to act as a toggle, or you can specify XCONTRAST ON or XCONTRAST OFF.

In some color combinations, text fields, buttons, check boxes, and other foreground categories might not be visible. The **`SAS.dmsContrastCheck`** resource makes these categories legible.

`SAS.dmsContrastCheck: True | False`
: controls whether contrast mapping is applied to non-graphic foreground colors in a SAS window. The default value is False. A value of True specifies that DMS foreground colors will be remapped if necessary to produce a contrast. Some color usage based on graphic operations are not affected by this resource.

Controlling Drop-down Menus in UNIX Environments

Drop-down menus are controlled by the following resources:

`SAS.pmenuOn: True | False`
: forces the global PMENU state on regardless of the information stored by the WSAVE command. The WSAVE state of an individual window takes precedence over the global state. The default is True. (You can also use the PMENU ON and PMENU OFF commands to turn drop-down menus on and off.)

`SAS.usePmenuMnemonics: True | False`
: specifies whether mnemonics are attached to the drop-down menus for the current SAS session. The default is True.

Customizing Cut and Paste in UNIX Environments

Instructions for Cutting and Pasting Text

For instructions about cutting and pasting text, see “Selecting (Marking) Text in UNIX Environments” on page 153 and “Copying or Cutting and Pasting Selected Text in UNIX Environments” on page 155.

Types of Paste Buffers

There are four SAS paste buffers. Each SAS paste buffer is associated with an X paste buffer:

XPRIMARY
: is associated with X primary selection (PRIMARY).

XSCNDARY
: is associated with the X secondary selection (SECONDARY).

XCLIPBRD
: is associated with the X clipboard selection (CLIPBOARD). This paste buffer enables you to use the MIT X Consortium xclipboard client with SAS.

XTERM
: is associated with the paste buffer used by the xterm client. XTERM is the default buffer. DEFAULT is an alias for XTERM. If you copy or cut text into the XTERM buffer, the text is actually copied or cut into all four of the paste buffers. When you paste text from the XTERM buffer, the text is pasted from the XPRIMARY buffer.

XCUT*n*
: is associated with X cut buffer*n* where 0 <= *n* <= 7.

Selecting a Paste Buffer

If you are not sure which X data exchange protocols your other X clients are using, you should use the XTERM paste buffer. You can specify your default paste buffer with the **`SAS.defaultPasteBuffer`** resource:

```
SAS.defaultPasteBuffer: XTERM
```

If you know that the X clients in your workstation environment all use the X PRIMARY selections to exchange data, you should use the XPRIMARY paste buffer:

```
SAS.defaultPasteBuffer: XPRIMARY
```

This specification uses both SAS and X resources more efficiently and provides for the on-demand transfer of data between clients.

Solaris OpenWindows desktop clients use the CLIPBOARD selection as the basis for their copy-and-paste operations. If you use the SAS XCLIPBRD paste buffer, you can exchange text directly with these clients.

You can also use the SAS XCLIPBRD paste buffer to interact with Motif clients that use the Motif clipboard mechanism for text exchanges. This clipboard mechanism makes it unnecessary to have a dedicated client such as xclipboard. For example, you can use XCLIPBRD to exchange text directly with the Motif xmeditor application when you select the **Cut**, **Copy**, or **Paste** items from the xmeditor **Edit** drop-down menu.

The Motif quick-copy data exchange and Motif clipboard data exchange mechanisms are specific to the Motif interface toolkit and are not currently supported as SAS paste buffers. However some dialog boxes, such as the File Selection dialog box, use Motif interface text widgets. In these dialog boxes, the Motif quick copy and clipboard data exchange mechanisms are available.

Manipulating Text Using a Paste Buffer

If you want SAS to automatically copy selected text into your paste buffer every time you mark a region of text with the mouse, you should also specify your paste buffer name in the **`SAS.markPasteBuffer`** resource:

```
SAS.markPasteBuffer: XTERM
```

Alternatively, because DEFAULT is an alias for XTERM, you could specify the following:

```
SAS.markPasteBuffer: DEFAULT
```

The **`SAS.markPasteBuffer`** definition causes SAS to automatically issue a STORE command whenever you select text.

The STORE command, as well as the CUT and PASTE commands, support a BUFFER= option that specifies which buffer to use. When these commands are issued from function keys or drop-down menus whose definitions do not include the BUFFER= option, if the **`SAS.markPasteBuffer`** resource is not defined, these commands use BUFFER=DEFAULT. If this resource is defined, these commands use BUFFER=*buffer-name*.

You can customize your normal cut, copy, or paste keys to issue any of these commands with the BUFFER= option. For example, you can override the **`SAS.keyboardTranslations`** definition for the **osfCopy** and **osfPaste** keys with the following specifications:

```
SAS.keyboardTranslations: #override \
 <Key>osfCopy: sas-do-command(\"STORE BUFFER=XCLIPBRD\") \n\
 <Key>osfPaste: sas-do-command(\"PASTE BUFFER=XCLIPBRD\")
```

For more information about customizing keys, see "Customizing Key Definitions in UNIX Environments" on page 184.

Notes about Preserving Text and Attribute Information

When you cut or copy and paste text between SAS sessions using the XTERM, XPRIMARY, or XSCNDARY paste buffers, the color and attribute information is preserved. However, if you copy and paste the same text into an xterm window while using the vi editor, the color and attribute information is lost. If you change the definition for **`SAS.defaultPasteBuffer`** and **`SAS.markPasteBuffer`** to XCUT0, then you will not retain the text and color attributes when you copy and paste text between two SAS sessions.

When you use the xclipboard client, SAS text attributes are not preserved in exchanges made between SAS sessions. However, when you use the XCLIPBRD paste buffer without a clipboard manager such as the xclipboard client, SAS text attributes are preserved in exchanges between SAS sessions.

Customizing Session Workspace, Session Gravity, and Window Sizes in UNIX Environments

SAS uses the following resources to determine the size of the session workspace, the gravity of the workspace, and the size of the windows. The default values for these resources are listed in Table 8.4 on page 210.

`SAS.awsResizePolicy: grow | fixed`
: controls the policy for resizing AWS windows as interior windows are added and removed. The following values are valid:

grow
: the AWS window will attempt to grow any time an interior window is grown or moved, in order to show all interior windows, but it will not shrink to remove dead areas.

fixed
: the AWS window will attempt to size itself to the size of the first interior window and will not attempt any further size changes.

`SAS.maxWindowHeight: units`
: specifies the number of units for the maximum height of a window. The unit is specified by the **`SAS.windowUnitType`** resource.

`SAS.maxWindowWidth: units`
: specifies the number of units for the maximum width of a window. The unit is specified by the **`SAS.windowUnitType`** resource.

`SAS.noAWS: True | False`
: controls whether each of your application's windows appears in its own native window rather than in an application work space (AWS). The default is True; each application runs in its own native window.

`SAS.scrollBarSize: pixels`
: specifies the default size of the scroll bar in pixels.

`SAS.sessionGravity: value`
: controls the region of the screen where SAS will attempt to place its windows. This resource might be ignored by some window manager configurations. Possible values include the following:

- **CenterGravity**
- **EastGravity**
- **WestGravity**
- **SouthGravity**
- **NorthGravity**
- **SouthEastGravity**
- **NorthEastGravity**
- **SouthWestGravity**
- **NorthWestGravity**

`SAS.sessionGravityXOffset: offset`
: specifies an x offset to be added when SAS attempts to place a window in the gravity region.

`SAS.sessionGravityYOffset: offset`
: specifies a y offset to be added when SAS attempts to place a window in the gravity region.

`SAS.windowHeight: units`
: specifies the number of units for the default height of a window. The unit is specified by the **`SAS.windowUnitType`** resource.

`SAS.windowUnitType: character | pixel | percentage`
: specifies the unit type for **`SAS.windowWidth`**, **`SAS.windowHeight`**, **`SAS.maxWindowWidth`**, and **`SAS.maxWindowHeight`**. Possible values include the following:

  character
  : units specify the number of rows and columns.

  pixel
  : units specify the number of pixels.

  percentage
  : units specify the percentage of the screen.

`SAS.windowWidth: units`
: specifies the number of units for the default width of a window. The unit is specified by the **`SAS.windowUnitType`** resource.

Specifying User-Defined Icons in UNIX Environments

Why Specify User-Defined Icons?

You can add your own icons to those icons that are supplied with SAS. For example, if you want to use your own color icons in the toolbox, define the **`SAS.colorUiconPath`**, **`SAS.colorUiconCount`**, and **`SAS.sasUiconx`** resources. Then, when you are defining tools in the Tool Editor, the Tool Editor will include your icons in the display of icons that you can choose for each tool.

How SAS Locates a User-Defined Icon

The resource name that is used to locate the icon bitmap filename for user icon number *x* is **`SAS.sasUiconx`**. For example, the following resource defines the filename **`myicon`** for the user icon **`1`**:

```
SAS.sasUicon1: myicon
```

If the resource name is not defined, SAS generates a filename of the form **`sasui`*`nnn`*`.xbm`** or **`sasui`*`nnn`*`.xpm`**. The path elements from the **`SAS.uiconPath`** or **`SAS.colorUiconpath`** resource are searched in sequence until the icon file is found or until the search path is exhausted.

For example, the following set of X resources defines a collection of color icons:

```
SAS.colorUiconPath: /users/jackaroe/pixmaps/
SAS.colorUiconCount: 7
SAS.sasUicon1: adsetup
SAS.sasUicon2: adverse
SAS.sasUicon3: altmenu
SAS.sasUicon4: batch
SAS.sasUicon5: is
SAS.sasUicon6: patgrps
SAS.sasUicon7: pctchg
```

The Motif interface will search for icon **sasUicon1** in a file named **/users/jackaroe/pixmaps/adsetup.xpm**.

X Resources for Specifying User-Defined Icons

SAS uses the following resources to determine the number of user-defined icons that are available and their location.

SAS.colorUiconPath: search-path
specifies the file search path for locating user-defined color icon files. This string resource specifies the directory paths to be searched for an icon file. These files should be in X Pixmap (xpm) format. Use a comma to separate individual directory pathnames. For example, the following string first searches for icon files in the **/usr/lib/X11/pixmaps** directory and then in the **/usr/lib/X11/pixmaps/SAS** directory:

```
SAS.colorUiconPath : /usr/lib/X11/pixmaps, \
/usr/lib/X11/pixmaps/SAS
```

SAS.colorUiconCount: num-icons
specifies the number of user-defined color icons that are available for SAS to use.

SAS.uiconCount: num-icons
specifies the number of user-defined icons that are available for use in the SAS session.

SAS.uiconPath: search-path
specifies the file search path for locating user-defined icon bitmap files. This string resource specifies the directory paths to be searched for an icon file. These files should be in X Bitmap (xbm) format. Use a comma to separate individual directory pathnames. For example, the following string will first search for bitmap files in the **/usr/lib/X11/bitmaps** directory and then in the **/usr/lib/X11/bitmaps/SAS** directory:

```
SAS.uiconPath : /usr/lib/X11/bitmaps,\
/usr/lib/X11/bitmaps/SAS
```

SAS.sasUiconx: name
associates a value with the filename of an X Bitmap or Pixmap file. *x* is a number assigned to the file. A file extension of **.xbm** or **.xpm** is automatically supplied.

Miscellaneous Resources in UNIX Environments

You can also customize the following resources:

SAS.altVisualId: ID
specifies a visual type ID.

`SAS.autoSaveInterval: minutes`
specifies how often (in number of minutes) that the data from the Program Editor window should be saved.

`SAS.autoSaveOn: True | False`
specifies that data from the Program Editor window should be saved to a file at intervals specified by the **`SAS.autoSaveInterval`** resource.

`SAS.confirmSASExit: True | False`
controls whether SAS displays the Exit dialog box when you enter the DLGENDR command or select **File** ⇨ **Exit**. The default is True.

`SAS.defaultCommandWindow: True | False`
specifies whether the command window is invoked when you start your SAS session. The default is True.

`SAS.directory: directory-pathname`
specifies the directory that you want when you first invoke the Open dialog box. By default, the Open dialog box uses the current directory.

`SAS.helpBrowser: pathname`
specifies the pathname of the World Wide Web browser to use for viewing the online Help or when the WBROWSE command is issued. The default browser is Netscape.

`SAS.htmlUsePassword: True | False`
specifies whether SAS prompts you to enter your password before sending HTML files to your browser. The default value is True.

`SAS.insertModeOn: True | False`
controls the editing mode in SAS editor windows. The default is False (overtype).

`SAS.noDoCommandRecall: True | False`
controls whether SAS commands that are submitted through the **`sas-do-command()`** action routine are recorded in the command recall buffer. The default value of True causes commands to be omitted from the command recall buffer; a value of False causes them to be recorded.

`SAS.pattern: default-pattern`
specifies the default pattern that you want to be used as the file filter when you first invoke the Open and Import Image dialog boxes. This pattern is displayed in the text field at the top of the dialog box. By default, the dialog box uses the first filter in the File type list. The pattern resource has no effect on the File type field.

`SAS.selectTimeout: seconds`
specifies the X Toolkit selection conversion time-out value in units of seconds. This time-out value determines the amount of time that SAS will wait for a request to convert an X Toolkit selection to complete. The default value should be adequate in most cases.

`SAS.startupLogo: xpm-filename | None | ""`
specifies the XPM file that you want SAS to display when it is initialized. If the string is empty, SAS uses the default logo.

`SAS.startSessionManager: True | False`
specifies whether SAS automatically starts the SAS Session Manager when a new SAS session is started. Using your own host editor with SAS requires that the SAS Session Manager be running. The default is True.

`SAS.suppressMenuIcons: True | False`
specifies whether SAS displays any menu icons other than the check box and toggle button icons in cascade or pop-up menus. Suppressing the icons reduces memory

usage and improves how quickly the menus display on slower X servers. The default is False.

`SAS.suppressTutorialDialog: True | False`
specifies whether SAS displays the Getting Started Tutorial dialog box at the start of your SAS session. True suppresses the dialog box. You might want to suppress this dialog box if you have previously used SAS. The default is False.

`SAS.useNativeXmTextTranslations: True | False`
specifies whether any XmText widget translations are inherited by all instances of the Text, Combo Box, and Spin Box widgets used by the SAS X Motif user interface. When the value is False, the SAS keys windows translations supersede any user or system-supplied XmText translations. The default value is True.

The following example shows SAS XmText translations:

```
SAS*XmText*translations:   #override \n\
Ctrl<Key>e:end-of-line()\n\
Ctrl<Key>u:delete-to-start-of-line()\n\
Ctrl<Key>k:delete-to-end-of-line()\n\
Ctrl<Key>f:forward-character()\n\
Ctrl<Key>b:backward-character()\n\
Ctrl<Key>a:beginning-of-line()\n\
Ctrl<Key>c:copy-clipboard()\n\
Ctrl<Key>v:paste-clipboard()\n\
```

`SAS.wsaveAllExit: True | False`
specifies whether SAS should issue the WSAVE ALL command when you end your session. This command saves the global settings, such as window color and window position, that are in effect for all windows that are currently open. The default is False.

Note: For the WSAVE command to work, your window manager must support explicit window placement. See the documentation for your window manager to determine how to configure your window manager. For example, if you are running Exceed, open the Screen Definition Settings dialog box and deselect **Cascade Windows**.

Summary of X Resources for SAS in UNIX Environments

The following table lists the instance and class names, type, and default values for many of the SAS resources. See the following sections for additional resources of specific types:

- "Specifying Font Aliases" on page 195
- "Defining Color Resources" on page 198
- "Defining Colors and Attributes for Window Elements (CPARMS)" on page 200

Table 8.4 *SAS Resources*

Resource Name	Class Name	Type	Default
SAS.altVisualId	AltVisualId	Integer	NULL

Resource Name	Class Name	Type	Default
SAS.autoComplete	AutoComplete	Boolean	True
SAS.autoSaveInterval	AutoSaveInterval	Integer	10
SAS.autoSaveOn	AutoSaveOn	Boolean	True
SAS.awsResizePolicy	AWSResizePolicy	String	grow
SAS.colorUiconCount	UiconCount	Integer	0
SAS.colorUiconPath	UiconPath	String	NULL
SAS.commandsSaved	CommandsSaved	Integer	25
SAS.confirmSASExit	ConfirmSASExit	Boolean	True
SAS.defaultCommandWindow	DefaultCommandWindow	Boolean	True
SAS.defaultPasteBuffer	DefaultPasteBuffer	String	XTERM
SAS.defaultToolBox	DefaultToolBox	Boolean	True
SAS.directory	Directory	String	NULL
SAS.dmsContrastCheck	DmsContrastCheck	Boolean	False
SAS.DMSDBFont	Font	String	*dynamic*
SAS.DMSDBboldFont	Font	String	*dynamic*
SAS.DMSboldFont	Font	String	*dynamic*
SAS.DMSFont	Font	String	*dynamic*
SAS.DMSfontPattern	DMSFontPattern	String	-*-*-*-r-*--*-*-*-*-m-*-iso8859-1
SAS.fontPattern	FontPattern	String	*
SAS.helpBrowser	HelpBrowser	String	Netscape
SAS.htmlUsePassword	HtmlUsePassword	Boolean	True
SAS.insertModeOn	InsertModeOn	Boolean	False
SAS.isToolBoxPersistent	IsToolBoxPersistent	Boolean	True
SAS.keyboardTranslations	KeyboardTranslations	Translation	*dynamic*
SAS.keysWindowLabels	KeysWindowLabels	String	*dynamic*
SAS.markPasteBuffer	MarkPasteBuffer	String	XTERM

Resource Name	Class Name	Type	Default
SAS.maxWindowHeight	WindowHeight	Dimension	95
SAS.maxWindowWidth	WindowWidth	Dimension	95
SAS.noAWS	NoAWS	Boolean	True
SAS.noDoCommandRecall	NoDoCommandRecall	Boolean	True
SAS.pattern	Pattern	String	NULL
SAS.pmenuOn	PmenuOn	Boolean	True
SAS.sasUicon	SasUicon	String	NULL
SAS.scrollBarSize	ScrollBarSize	Dimension	17
SAS.selectTimeout	SelectTimeout	Integer	60
SAS.sessionGravity	SASGravity	String	NorthWestGravity
SAS.sessionGravityXOffset	SASGravityOffset	Integer	0
SAS.sessionGravityYOffset	SASGravityOffset	Integer	0
SAS.startSessionManager	StartSessionManager	Boolean	True
SAS.startupLogo	StartUpLogo	String	NULL
SAS.suppressMenuIcons	SuppressMenuIcons	Boolean	False
SAS.suppressTutorialDialog	SuppressTutorialDialog	Boolean	False
SAS.systemFont	SystemFont	String	"-adobe-helvetica-medium-r-normal--12–*–*–*–*–*–*–*"
SAS.toolBoxAlwaysOnTop	ToolBoxAlwaysOnTop	Boolean	True
SAS.toolBoxTipDelay	ToolBoxTipDelay	Integer	750
SAS.uiconCount	UiconCount	Integer	0
SAS.uiconPath	UiconPath	String	NULL
SAS.useCommandToolBoxCombo	UseCommandToolBoxCombo	Boolean	True
SAS.useLargeToolBox	UseLargeToolBox	Boolean	False
SAS.useNativeXmTextTranslations	UseNativeXmTextTranslations	Boolean	False
SAS.usePmenuMnemonics	UsePmenuMnemonics	Boolean	True

Resource Name	**Class Name**	**Type**	**Default**
SAS.useShowHideDecorations	UseShowHideDecorations	Boolean	False
SAS.useToolBoxTips	UseToolBoxTips	Boolean	True
SAS.wsaveAllExit	WsaveAllExit	Boolean	False
SAS.windowHeight	WindowHeight	Dimension	50
SAS.windowWidth	WindowWidth	Dimension	67
SAS.windowUnitType	WindowUnitType	String	percentage

Part 3

Application Considerations

Chapter 9
Data Representation

Numeric Variable Length and Precision in UNIX Environments

The default length of numeric variables in SAS data sets is 8 bytes. (You can control the length of SAS numeric variables with the LENGTH or ATTRIB statements in the DATA step.)

The issue of numeric precision affects the return values of almost all SAS math functions and many numeric values returned from SAS procedures. Numeric values in SAS for UNIX are represented as IEEE double-precision floating-point numbers. The decimal precision of a full 8-byte number is effectively 15 decimal digits.

The following table specifies the significant digits and largest integer that can be stored exactly in SAS numeric variables.

Table 9.1 *Significant Digits and Largest Integer by Length for SAS Variables under UNIX*

Length in Bytes	Significant Digits Retained	Largest Integer Represented Exactly
3	3	8,192
4	6	2,097,152
5	8	536,870,912
6	11	137,438,953,472
7	13	35,184,372,088,832
8	15	9,007,199,254,740,992

When you are specifying variable lengths, keep in mind that the length of a variable affects both the amount of disk space used and the number of I/O operations required to read and write the data set.

If you know that the value of a numeric variable will be an integer between -8192 and 8192 inclusive, you can use a length of 3 to store the number and thus save space in your data set. For example:

```
data mydata;
   length num 3;
   ...more SAS statements...
run;
```

Numeric dummy variables (variables whose only purpose is to hold 0 or 1) can be stored in a variable whose length is 3 bytes.

CAUTION:

Use the LENGTH statement to reduce length only for variables whose values are always integers. Fractional numbers lose precision if they are truncated. In addition, you must ensure that the values of your variable will always be represented exactly in the number of bytes that you specify. You can do this programmatically in a DATA step with the TRUNC function. No warnings or errors are issued when the length that you specify in the LENGTH statement results in the truncation of data.

For more information about specifying variable lengths and optimizing system performance, see *SAS Language Reference: Concepts*.

Missing Values in UNIX Environments

In SAS on UNIX, missing values are represented by IEEE Not-a-Number values. An IEEE Not-a-Number value is an IEEE floating-point bit pattern that represents something other than a valid numeric value. These numbers are not computationally derivable.

Reading and Writing Binary Data in UNIX Environments

Different computers store numeric binary data in different forms. For more information about compatible computer types, see "Compatible Computer Types in UNIX Environments" on page 44. If you try to move binary data in flat files across systems that are incompatible, problems will occur. A safer way to move data is by using SAS data sets.

SAS provides several sets of informats and formats for handling binary data. Some of these informats and formats are host dependent. For example, the IB*w.d*, PD*w.d*, PIB*w.d*, and RB*w.d* informats and formats read and write data in native mode. That is, they use the byte-ordering system that is standard for the computer. If you create a file using the IB*w.d* format on a 64-bit HP-UX host and then use the IB*w.d* informat to read the same file on a 32-bit Linux host, you will get unpredictable results.

For more information about all of the informats and formats, see *SAS Formats and Informats: Reference*.

Converting a UNIX Datetime Value to a SAS Datetime Value

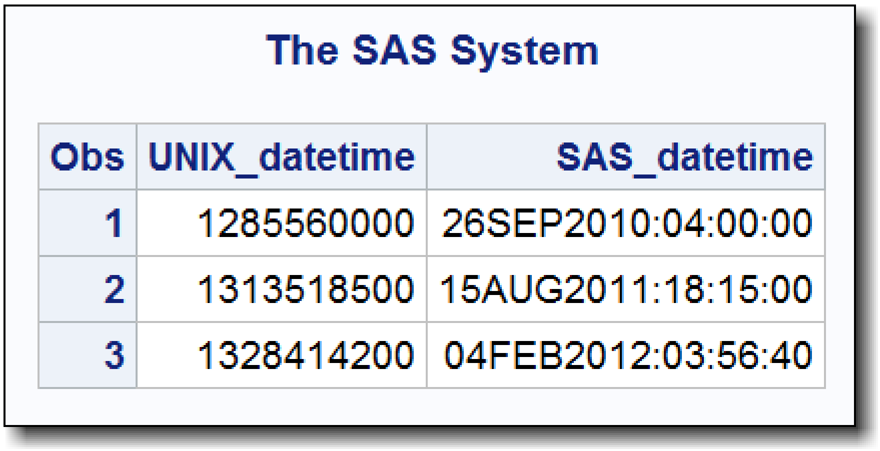
A UNIX datetime value is stored as the number of seconds since January 1, 1970. A SAS datetime value is stored as the number of seconds since January 1, 1960. To convert a UNIX datetime value to a SAS datetime value, you must add 10 years in seconds to the UNIX datetime value.

The INTNX function converts a UNIX datetime value to a SAS datetime value, as shown in the example below:

```
data UNIX_to_SAS;
   input UNIX_datetime;
      /* The INTNX function accounts for leap years. */
   SAS_datetime = intnx('DTyear',UNIX_datetime,10,'s');
   format SAS_datetime datetime20.;
datalines;
1285560000
1313518500
1328414200
;
proc print data=UNIX_to_SAS;
run;
```

The following output displays the results.

Display 9.1 *Conversion of a UNIX Datetime Value to a SAS Datetime Value*



The SAS System

Obs	UNIX_datetime	SAS_datetime
1	1285560000	26SEP2010:04:00:00
2	1313518500	15AUG2011:18:15:00
3	1328414200	04FEB2012:03:56:40

For more information, see “INTNX Function” in *SAS Functions and CALL Routines: Reference*.

Part 4

Host-Specific Features of the SAS Language

Chapter 10
Commands under UNIX

SAS Commands under UNIX

This section describes commands that you can enter on the command line in the windowing environment of SAS. The commands that are described here have behavior or syntax that is specific to UNIX environments. Each command description includes a brief "UNIX specifics" section that explains which aspect of the command is specific to UNIX. If the information under "UNIX specifics" says "all," then the command applies to the UNIX operating environment and is described only in this document.

The following commands are not supported in UNIX environments:

CASCADE	RESIZE	WGROW
DCALC	SCROLLBAR	WMOVE
ICON	SMARK	WSHRINK
PCLEAR	TILE	ZOOM

Dictionary

AUTOSCROLL Command: UNIX

Specifies how often the Log and Output windows scroll to display output.

UNIX specifics: valid arguments and default values

Syntax

AUTOSCROLL <*n*>

Optional Argument

n

specifies the number of lines that the window should scroll when it receives a line of data that cannot fit.

Details

The AUTOSCROLL command controls the scrolling of lines as they are written to the Log and Output windows. The default value for AUTOSCROLL in the Log and Output windows is `1`. Processing is slower when AUTOSCROLL displays one line at a time. To expedite processing, you can specify a greater AUTOSCROLL value in your autoexec.sas file. Specifying a value of 0 optimizes processing and results in the fastest scrolling (similar to jump scrolling in xterm windows). To add the AUTOSCROLL command to your autoexec.sas file, you must use the DM command. The following example maximizes scrolling in both the Log and Output windows:

```
dm 'output; autoscroll 0; log; autoscroll 0; pgm;';
```

CAPS Command: UNIX

Changes the default case of text.

UNIX specifics: all

Syntax

CAPS <ON | OFF>

Optional Arguments

ON
turns capitalization on.

OFF
turns capitalization off.

COLOR Command: UNIX

Specifies the color and highlighting of selected portions of a window.

UNIX specifics: valid field types and attributes

Syntax

COLOR *field-type color*|NEXT <*highlight*>

Required Arguments

field-type
specifies an area of the window such as background, command, border, message, and so on.

color
specifies a color such as blue (which can be abbreviated as B), red (R), green (G), cyan (C), pink (P), yellow (Y), white (W), black (K), magenta (M), gray (A), brown (B), or orange (O).

NEXT
changes the color to the next available color.

Optional Argument

highlight
can be H (which causes text to be displayed in a bold font), U (underlined), or R (reverse video). The BLINK attribute is not supported.

Details

Under UNIX, you cannot use the COLOR command to change the colors in these field types: BORDER, MENU, MENUBORDER, SCROLLBAR, or TITLE. Also, the H

(HIGHLIGHT) and B (BLINK) attributes are not supported. For more information about the COLOR command, see the online Help for the Program Editor window.

DLGABOUT Command: UNIX

Opens the About SAS dialog box.

UNIX specifics: all

Syntax

DLGABOUT

Details

The About SAS dialog box displays information such as the release of SAS that you are running, your site number, the operating system, the version of Motif that you are using, and the color information from your PC.

To access this dialog box from the menu, select **Help** ⇨ **About SAS 9**.

DLGCDIR Command: UNIX

Opens the Change Working Directory dialog box.

UNIX specifics: all

Syntax

DLGCDIR

Details

The Change Working Directory dialog box enables you to select a new working directory. To access this dialog box from the menu, select **Tools** ⇨ **Options** ⇨ **Change Directory**.

DLGENDR Command: UNIX

Opens the Exit dialog box.

UNIX specifics: all

Syntax

DLGENDR

Details

The Exit dialog box prompts you to confirm that you want to exit SAS. If you choose **OK**, the SAS session ends. If you have set the **`SAS.confirmSASExit`** resource to

False, this command becomes equivalent to the BYE command. To access this dialog box from the menu, select **File** ⇨ **Exit**.

See Also

"Miscellaneous Resources in UNIX Environments" on page 208

DLGFIND Command: UNIX

Opens the Find dialog box.

UNIX specifics: all

Syntax

DLGFIND

Details

The Find dialog box enables you to search for text strings. To access this dialog box from the menu, select **Edit** ⇨ **Find**.

See Also

Commands:

- "DLGREPLACE Command: UNIX" on page 229

DLGFONT Command: UNIX

Opens the Fonts dialog box.

UNIX specifics: all

Syntax

DLGFONT

Details

The Font dialog box enables you to dynamically change the SAS font. To access this dialog box from the menu, select **Tools** ⇨ **Options** ⇨ **Fonts**.

See Also

Commands:

- "SETDMSFONT Command: UNIX" on page 237

Other References:

- "Customizing Fonts in UNIX Environments" on page 192

DLGOPEN Command: UNIX

Opens the Open dialog box.

UNIX specifics: all

Syntax

DLGOPEN <FILTERS='*filters*' <IMPORT><SUBMIT|NOSUBMIT><VERIFY>>

Optional Arguments

FILTERS='*filters*'
specifies one or more file filters to use as search criteria when displaying files. For example, the following command displays all files in the current directory that have a **.sas** extension and adds ***.txt** to the **File type** box in the dialog box:

```
DLGOPEN FILTERS="*.sas *.txt"
```

You can specify multiple filters; they all appear in the box. If you do not specify any filters, the dialog box displays a default list. See the description of the **SAS.pattern** resource in "Miscellaneous Resources in UNIX Environments" on page 208 for information about specifying a default file pattern.

IMPORT
invokes the Import Image dialog box, which enables you to import graphic files to SAS/GRAPH applications.

SUBMIT|NOSUBMIT
specifies whether the SUBMIT command is pushed after the file is opened.

VERIFY
checks whether the DLGOPEN command is appropriate for the active window.

Details

The Open and Import Image dialog boxes enable you to select a file to read into the active window. If the active window is a SAS/GRAPH window, then the Import Image dialog box is displayed. Otherwise, the Open dialog box is displayed. To access these dialog boxes from the menu, select **File** ⇨ **Open** or **File** ⇨ **Import Image**.

For more information, see "Specifying Images in SAS/GRAPH Programs" in the *SAS/GRAPH: Reference*.

DLGPREF Command: UNIX

Opens the Preferences dialog box.

UNIX specifics: all

Syntax

DLGPREF

Details

The Preferences dialog box enables you to dynamically change certain resource settings. To access this dialog box from the menu, select **Tools** ⇨ **Options** ⇨ **Preferences**.

See Also

"Modifying X Resources through the Preferences Dialog Box" on page 166

DLGREPLACE Command: UNIX

Opens the Change dialog box.

UNIX specifics: all

Syntax

DLGREPLACE

Details

The Change dialog box enables you to search for and replace text strings. To access this dialog box from the menu, select **Edit** ⇨ **Replace**.

See Also

Commands:

- "DLGFIND Command: UNIX" on page 227

DLGSAVE Command: UNIX

Opens the Save As or Export dialog box.

UNIX specifics: all

Syntax

DLGSAVE <FILTERS='*filters*' <EXPORT><VERIFY>>

Optional Arguments

FILTERS='*filters*'
specifies one or more file filters to use as search criteria when displaying files. For example, the following command displays all files in the current directory that have a **`.sas`** extension and adds **`*.txt`** to the **File type** box in the dialog box:

```
DLGSAVE FILTERS="*.sas *.txt"
```

You can specify multiple filters; they all appear in the dialog box. If you do not specify any filters, the dialog box displays a default list.

EXPORT
invokes the Export dialog box, enabling you to export graphic files in your SAS session.

VERIFY
checks whether the DLGSAVE command is appropriate for the active window.

Details

To access this dialog box from the menu, select **File** ⇨ **Save as** or **File** ⇨ **Export as Image**.

For more information, see “Specifying Images in SAS/GRAPH Programs” in the *SAS/GRAPH: Reference*.

DLGSCRDUMP Command: UNIX

Saves the active SAS/GRAPH window as an image file using the filename and file type that you specify.

UNIX specifics: all

Syntax

DLGSCRDUMP <'*filename.ext*' 'FORMAT=*file-type*'>

Details

DLGSCRDUMP saves the active GRAPH window as an image file by using the filename and file type that you specify. If you do not specify arguments, DLGSCRDUMP opens the Export dialog box and enables you to choose a filename and file type. You can save displays in any image format that is supported by SAS/GRAPH.

For more information, see “Specifying Images in SAS/GRAPH Programs” in the *SAS/GRAPH: Reference*.

DLGSMAIL Command: UNIX

Opens the Send Mail dialog box.

UNIX specifics: all

Syntax

DLGSMAIL

Details

The Send Mail dialog box enables you to send electronic mail while working in SAS. To access this dialog box from the menu, select **File** ⇨ **Send mail**.

See Also

System Options:

- "EMAILSYS System Option: UNIX" on page 379

Other References:

- "Sending Electronic Mail Using the FILENAME Statement (EMAIL)" on page 84
- "Sending Mail from within Your SAS Session in UNIX Environments" on page 158

FILE Command: UNIX

Writes the contents of the current window to an external file.

UNIX specifics: valid values for *encoding-value* and *host-options*

Syntax

FILE <*file-specification*> <ENCODING='*encoding-value*'> <*portable-options*> <*host-options*>

Optional Arguments

file-specification
can be any of the following:

single filename
SAS writes the file in the current directory. If you enclose the filename in quotation marks, SAS uses the filename exactly as you specify it. If you do not enclose the filename in quotation marks and if you do not specify a filename extension, SAS uses .sas, .log, or .lst, depending on whether you issue the command from the Program Editor, Log, or Output window.

entire pathname
SAS does not assume any filename extensions, even if you do not enclose the pathname in quotation marks.

fileref
SAS specifies a fileref to assign to an external file.

ENCODING='*encoding-value*'
specifies the encoding to use when writing to the output file. The value for ENCODING= indicates that the output file has a different encoding from the current session encoding.

When you write data to the output file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Overview to SAS Language Elements That Use Encoding Values" in Chapter 20 of *SAS National Language Support (NLS): Reference Guide*.

portable-options
are options for the FILE command that are valid in all operating environments. For more information about these options, see *SAS System Options: Reference*.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=BLK=
specifies the number of bytes that are physically written in one I/O operation. The default is 8K. The maximum is 1G–1.

LRECL=
: specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the length of each output record. The output record is truncated or padded with spaces to fit the specified size.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are truncated.

NEW|OLD
: indicates that a new file is to be opened for output. If the file already exists, then it is deleted and re-created. This is the default action.

RECFM=
: specifies the record format. Values for the RECFM= option are listed below:

D	default format (same as variable).
F	fixed format. That is, each record has the same length. Do not use RECFM=F for external files that contain carriage-control characters.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format. SAS writes carriage-control characters.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS).

UNBUF
: tells SAS not to perform buffered writes to the file on any subsequent FILE statement. This option applies especially when you are writing to a data collection device.

Details

If you do not enter a file specification, SAS uses the filename from the previous FILE or INCLUDE command. In this case, SAS first asks whether you want to overwrite the file. If you have not issued any FILE or INCLUDE commands, you receive an error message that indicates that no default file exists.

FILL Command: UNIX

Specifies the fill character.

UNIX specifics: default character

Syntax

FILL <*fill-character*>

Optional Argument

fill-character
specifies the character to be used to fill out a line.

Under UNIX, the default fill character is an underscore (_).

FONTLIST Command: UNIX

Opens the Select Font window, which lists available software fonts.

UNIX specifics: all

Syntax

FONTLIST

Details

The FONTLIST command opens windows that list all of the software fonts that are available in your operating environment. This feature is useful if you want to choose a font to use in a SAS program, typically with a FONT= or FTEXT= option.

Issuing the FONTLIST command from the SAS command line opens the Select Font window, which contains two buttons, **Copy** and **System**. Clicking **System** opens the Fonts window, from which you can select and preview all available system fonts. After you select the desired font and font attributes, click **OK**. The Select Font window reopens with your selected font name displayed. Clicking **Copy** places the font name in the copy buffer so that you can paste the selected font name into your SAS program.

GSUBMIT Command: UNIX

Submits SAS code stored in a paste buffer.

UNIX specifics: valid buffer names

Syntax

GSUBMIT BUF=*buffername*|“*statement1*;*statementN*...;”

Required Arguments

buffername
can be XPRIMARY, XSCNDARY, XCLIPBRD, XTERM, or XCUT*n* where 0<=*n*<=7. For more information, see “Customizing Cut and Paste in UNIX Environments” on page 203.

statementN
can be any SAS statement.

HOME Command: UNIX

Toggles the cursor position between the current position and the command line.

UNIX specifics:	keyboard equivalent

Syntax

HOME

Details

Keyboards vary among the different UNIX operating environments. To determine which key is assigned to the HOME command, look in the Keys window. To open the Keys window, issue the KEYS command.

See Also

"Customizing Key Definitions in UNIX Environments" on page 184

HOSTEDIT Command: UNIX

Starts the UNIX editor, specified by the EDITCMD system option, in the current window.

Alias:	HED
UNIX specifics:	all

Syntax

HOSTEDIT

Details

When you issue the HOSTEDIT command from a SAS text editor window, the contents of the buffer for that window are written to a temporary file in the `/tmp` directory. A command invoking the host editor that was specified in the EDITCMD system option is passed to the SAS Session Manager. The SAS Session Manager issues the command to the operating environment to invoke the editor for the temporary file.

The X display that is used with the HOSTEDIT command is the same one that is used with your SAS session.

See Also

System Options:

- "EDITCMD System Option: UNIX" on page 378

Other References:

- "Configuring SAS for Host Editor Support in UNIX Environments" on page 160

INCLUDE Command: UNIX

Copies the entire contents of an external file into the current window.

UNIX specifics: valid values for *encoding-value* and *portable-options*

Syntax

INCLUDE <*file-specification*> <ENCODING='*encoding-value*'> <*portable-options*> <*host-options*>

Optional Arguments

file-specification
can be any of the following:

- a single filename. SAS searches for the file in the current directory. If you enclose the filename in quotation marks, then SAS uses the filename exactly as you specify it. If you do not enclose the filename in quotation marks and if you do not specify a filename extension, then SAS searches for .sas.
- an entire pathname. SAS does not assume any filename extensions, even if you do not enclose the pathname in quotation marks.
- a fileref.

ENCODING='*encoding-value*'
specifies the encoding to use when reading from the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see “Overview to SAS Language Elements That Use Encoding Values” in Chapter 20 of *SAS National Language Support (NLS): Reference Guide*.

portable-options
are options for the INCLUDE command that are valid in all operating environments. See *SAS System Options: Reference* for information about these options.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically read in one I/O operation. The default is 8K. The maximum is 1G–1.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the number of bytes to be read as one record.
- If RECFM=N, then the value for the LRECL= option must be at least 256.

- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are truncated.

RECFM=
: specifies the record format. Values for the RECFM= option are listed below:

D	default format (same as variable).
F	fixed format. That is, each record has the same length.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format.
V	variable format. Each record ends with a newline character.

Details

If you do not enter a file specification, then SAS uses the filename from the previous FILE or INCLUDE command. In this case, SAS first asks whether you want to overwrite the file. If you have not issued any FILE or INCLUDE commands, then you receive an error message to indicate that no default file exists.

SETAUTOSAVE Command: UNIX

Turns autosave on and off.

UNIX specifics: all

Syntax

SETAUTOSAVE <ON|OFF>

Details

The SETAUTOSAVE command turns autosave on or off for the Program Editor. However, the value set for autosave in the Preferences dialog box has precedence. To open the Preferences dialog box, select **Tools** ⇨ **Options** ⇨ **Preferences**. Autosave is controlled by the **Backup Documents** check box on the **DMS** tab. On this tab, there is also a field in which you can specify the interval for these backups.

If you turn autosave on using the SETAUTOSAVE command and the **Backup Documents** check box is selected, then SAS automatically saves the contents of the Program Editor into a file named `pgm.asv` in your current directory at the interval specified on the **DMS** tab.

If you issue this command but do not specify ON or OFF, SAS displays the current autosave setting.

See Also

- "Modifying the DMS Settings" on page 168
- "Miscellaneous Resources in UNIX Environments" on page 208

SETDMSFONT Command: UNIX

Specifies a windowing environment font for the current session.

UNIX specifics: all

Syntax

SETDMSFONT "*font-specification*"

Required Argument

font-specification
specifies an X Logical Font Description (XLFD) pattern that you want SAS to use in order to determine the windowing environment font.

Details

Most fonts in the X Window System are associated with an XLFD, which contains a number of different fields that are delimited by a dash (-) character. The fields in the XLFD indicate properties such as the font family name, weight, size, resolution, and whether the font is proportional or monospaced. See your X Window system documentation for more information about the XLFD and font names that are used with the X Window System.

See Also

Commands:

- "DLGFONT Command: UNIX" on page 227

TOOLCLOSE Command: UNIX

Closes the tool box.

UNIX specifics: all

Syntax

TOOLCLOSE

Details

The TOOLCLOSE command closes the toolbox.

See Also

Commands:

- "TOOLLOAD Command: UNIX" on page 238

TOOLEDIT Command: UNIX

Opens the Tool Editor on the specified toolbox.

UNIX specifics: all

Syntax

TOOLEDIT <*library.catalog.entry*>

Details

If you do not specify an entry name, the Tool Editor edits the toolbox for the active window.

TOOLLARGE Command: UNIX

Toggles the size of the SAS ToolBox window.

UNIX specifics: all

Syntax

TOOLLARGE <ON|OFF>

Required Arguments

ON
sets the size of the icons in the SAS ToolBox to 48x48.

OFF
sets the size of the icons in the SAS ToolBox to 24x24.

Details

If you do not specify ON or OFF, the TOOLLARGE command toggles the size of the SAS ToolBox. The size of the SAS ToolBox changes for your current session only; the new size is not saved.

You can also use the menu to change the size of the SAS ToolBox through the Preferences dialog box. Select **Tools** ⇨ **Options** ⇨ **Preferences**. Select the **ToolBox** tab, and then select **Use large tools**. If you change the size of the SAS ToolBox through the Preferences dialog box, the new size is saved, and SAS displays the large toolbox in subsequent sessions.

TOOLLOAD Command: UNIX

Loads a specific toolbox.

UNIX specifics: all

Syntax

TOOLLOAD <*library.catalog.entry*>

Details

If you do not specify an entry name, TOOLLOAD loads the toolbox for the active window.

See Also

Commands:

- "TOOLCLOSE Command: UNIX" on page 237

TOOLTIPS Command: UNIX

Toggles the ToolTip text for an icon in the toolbox.

UNIX specifics:	all

Syntax

TOOLTIPS <ON|OFF>

Required Arguments

ON
specifies that the ToolTip text is displayed when you move the cursor over an icon in the toolbox.

OFF
specifies that the ToolTip text is not displayed.

Details

If you do not specify ON or OFF, the TOOLTIPS command turns the ToolTip text on or off, depending on the current setting.

You can also use the Preferences dialog box to specify whether ToolTip text is displayed by selecting **Tools** ⇨ **Options** ⇨ **Preferences**. Select the **ToolBox** tab, and then select **Use tip text** .

See Also

"Changing the Attributes of an Existing Tool" on page 180

WBROWSE Command: UNIX

Opens a World Wide Web (WWW) browser.

UNIX specifics:	all

Syntax

WBROWSE <"*url*">

Details

WBROWSE invokes the Web browser that is specified by the resource **`SAS.helpBrowser`**. If you specify a URL, the document that the URL identifies is automatically displayed. If you do not specify a URL, the SAS home page is displayed.

See Also

"Miscellaneous Resources in UNIX Environments" on page 208

WCOPY Command: UNIX

Copies the marked contents of the active window to the default buffer.

UNIX specifics: all

Syntax

WCOPY

Details

In Base SAS windows, this command executes the STORE command. For information about the STORE command, see online SAS Help and Documentation.

WCUT Command: UNIX

Moves the marked contents of the active window to the default buffer.

UNIX specifics: all

Syntax

WCUT

Details

In Base SAS windows, this command executes the CUT command.

This command is valid only when the active window is a text editor window, such as Program Editor or NOTEPAD.

For information about the CUT command, see online SAS Help and Documentation.

WDEF Command: UNIX

Redefines the active window.

UNIX specifics: behavior is controlled by the SAS.awsResizePolicy resource

Syntax

WDEF *starting-row starting-col nrows ncols*

Details

The WDEF command operates in the application workspace assigned to the SAS session. The WDEF command does not operate in the AWS container window, except when the container window needs to be enlarged so that you can view a SAS window contained in it. AWS resize behavior is controlled by the `SAS.awsResizePolicy` resource.

See Also

- "Miscellaneous Resources in UNIX Environments" on page 208
- "X Window Managers" on page 141

WPASTE Command: UNIX

Pastes the contents of the default buffer into the active window.

UNIX specifics: all

Syntax

WPASTE

Details

In Base SAS windows, this command executes the PASTE command. For information about the PASTE and command, see online SAS Help and Documentation.

WUNDO Command: UNIX

Undoes one line of text entry, or undoes the last cut, copy, or paste action.

UNIX specifics: all

Syntax

WUNDO

Details

In Base SAS windows, this command executes the UNDO command. In SAS/GRAPH windows, WUNDO is invalid.

One execution of the WUNDO command undoes text entry for only one line at a time. If you issue the WUNDO command again, the previous line of text is undone.

If you use the CC command to copy and paste a block of text, and then issue the WUNDO command, the block of text that you copied is deleted. If you use the DD command to delete a block of text, and then issue the WUNDO command, the block of text that you deleted is restored.

Note: The WUNDO command cannot replace lines that the SUBMIT command removes. It cannot reverse the effects of submitted SAS statements.

X Command: UNIX

Enables you to enter UNIX commands without ending the SAS session.

UNIX specifics: all

Syntax

X *UNIX-command*

X '*cmd1*;*cmd2*....<;*cmd-n*>'

Details

When you enter the X command, SAS starts a shell to execute the commands that you specified. The commands that you enter are processed differently, depending on whether you enter one command or more than one command.

See Also

"Executing Operating System Commands from Your SAS Session" on page 15

XSYNC Command: UNIX

Changes X synchronization during a SAS session.

UNIX specifics: all

Syntax

XSYNC <ON|OFF>

Details

This command turns off the buffering that is normally done by the X Window System. X synchronization is off by default. Turning it on is useful when you are debugging applications, although it drastically reduces performance.

If you do not specify ON or OFF, XSYNC toggles the synchronization. The XSYNC command is valid from any SAS window.

Chapter 11

Data Set Options under UNIX

SAS Data Set Options under UNIX

This section describes SAS data set options that exist only in the UNIX environment, as well as options whose behavior or syntax is specific to UNIX. Each data set option description includes a brief “UNIX specifics” section that explains which aspect of the data set option is specific to UNIX. For data set options that have behavior or syntax specific to UNIX, see *SAS Data Set Options: Reference* for a complete description of the option.

Specify data set options following the data set name in SAS statements as follows:

...data-set-name(*option-1*=*value-1 option-2*=*value-2*,...)

A few data set options are also SAS system options (for example, BUFSIZE=). If the same option is specified both as a system option and as a data set option, SAS uses the value given with the data set option. For more information about SAS system options, see “Customizing Your SAS Session by Using System Options” on page 18 and “SAS System Options under UNIX” on page 348.

To view a table of all of the data set options available under UNIX, see “Summary of SAS Data Set Options in UNIX Environments” on page 243.

Summary of SAS Data Set Options in UNIX Environments

SAS data set options are listed in the following table. The table lists the name of each option, a brief description, whether the option can be used for a data set opened for input, output, or update, and a list of engines for which the option is valid. The **See** column tells you where to look for more information about an option. Use the following legend to locate the additional information.

COMP
: See the description of the data set option in this section.

DS
: See *SAS Data Set Options: Reference*.

NLS
: See *SAS National Language Support (NLS): Reference Guide*.

Table 11.1 *Summary of SAS Data Set Options*

Option Name	Description	When Used	Engines	See
ALTER=	specifies a password for a SAS file that prevents users from replacing or deleting the file, but permits Read and Write access.	output, update	V9, V8, V6	DS, COMP
BUFNO=	specifies the number of buffers to be allocated for processing a SAS data set.	input, output, update	V9, V8, V6	DS, COMP
BUFSIZE=	specifies the size of a permanent buffer page for an output SAS data set.	output	V9, V8	DS, COMP
CNTLLEV=	specifies the level of shared access to SAS data sets.	input, update	V9, V8	DS
COMPRESS=	controls the compression of observations in a new output SAS data set.	output	V9, V8, V6	DS
DLDMGACTION=	specifies the action to take when a SAS data set in a SAS library is detected as damaged.	input, output, update	V9, V8	DS
DROP=	for an input data set, excludes the specified variables from processing; for an output data set, excludes the specified variables from being written to the data set.	input, output, update	all	DS
ENCODING=	overrides the encoding for the input or output SAS data set.	input, output	V9, V8	NLS
ENCRYPT=	specifies whether to encrypt an output SAS data set.	output	all	DS
FIRSTOBS=	specifies the first observation that SAS processes in a SAS data set.	input, update	all	DS
GENMAX=	requests generations for a SAS data set, and specifies the maximum number of versions.	output, update	V9, V8	DS

Option Name	Description	When Used	Engines	See
GENNUM=	specifies a particular generation of a SAS data set.	input, output, update	V9, V8	DS
IDXNAME=	directs SAS to use a specific index to meet the conditions of a WHERE expression.	input, update	V9, V8, V6	DS
IDXWHERE=	specifies whether SAS uses an index or uses a sequential search, to match the conditions of a WHERE expression.	input, update	V9, V8, V6	DS
IN=	creates a Boolean variable that indicates whether the data set contributed data to the current observation.	input, update	all	DS
INDEX=	defines an index for a new output SAS data set.	output	V9, V8, V6	DS
KEEP=	for an input data set, specifies the variables to process; for an output data set, specifies the variables to write to the data set.	input, output, update	all	DS
LABEL=	specifies a label for a SAS data set.	input, output, update	all	DS
OBS=	specifies the last observation that SAS processes in a data set.	input, update	all	DS
OBSBUF=	determines the size of the view buffer for processing a DATA step view.	input	V9, V8	DS
OUTREP=	specifies the data representation for the output SAS data set.	output	V9, V8	DS, NLS
POINTOBS=	controls whether to process a compressed SAS data set by observation number or by sequential access.	output	V9, V8	DS
PW=	assigns a READ, WRITE, or ALTER password to a SAS file and enables access to a password-protected file.	input, output, update	V9, V8, V6	DS, COMP
PWREQ=	specifies whether to display a dialog box for a SAS data set password.	input, output, update	V9, V8, V6	DS
READ=	assigns a password to a SAS file and enables access to a Read-protected SAS file.	input, output, update	V9, V8, V6	DS
RENAME=	changes the name of a variable.	input, output, update	all	DS

Option Name	Description	When Used	Engines	See
REPEMPTY=	specifies whether a new, empty data set can overwrite an existing SAS data set that has the same name.	output	V9, V8	DS
REPLACE=	specifies whether a new SAS data set that contains data can overwrite an existing data set that has the same name.	output	all	DS
REUSE=	specifies whether new observations can be written to freed space in compressed SAS data sets.	output	V9, V8, V6	DS
SORTEDBY=	indicates how the SAS data set is currently sorted.	input, output, update	V9, V8, V6	DS
SPILL=	specifies whether to create a spill file for non-sequential processing of a DATA step view.	output	V9, V8	DS
TOBSNO= (valid only for data sets that are accessed through a SAS server by way of the REMOTE engine)	specifies the number of observations to send in a client/server transfer.	input, output, update	V9, V8	DS
TYPE=	specifies the data set type for a specially structured SAS data set.	input, output, update	all	DS
USEDIRECTIO=	turns on direct file I/O for the file that you specify. To use this data set option, you must specify the ENABLEDIRECTIO statement option in the LIBNAME statement where the libref was assigned.	input, output, update	V9, V8	COMP
WHERE=	selects observations in a SAS data set that match the specified conditions.	input, output, update	all	DS
WHEREUP=	specifies whether to evaluate new observations and updated observations against a WHERE expression.	output, update	V9, V8, V6	DS
WRITE=	assigns a WRITE password to a SAS data set and enables access to a Write-protected SAS file.	output, update	V9, V8, V6	DS

Note: The TOBSNO= option is valid only for data sets that are accessed through a SAS server from the REMOTE engine.

Dictionary

ALTER= Data Set Option: UNIX

Specifies a password for a SAS file that prevents users from replacing or deleting the file, but permits Read and Write access.

Valid in:	DATA step and PROC steps
Category:	Data Set Control
Default:	none
Engine:	V9, V8, V6
See:	"ALTER= Data Set Option" in *SAS Data Set Options: Reference*

Syntax

ALTER=*alter-password*

Required Argument

alter-password
must be a valid SAS name. See "Words in the SAS Language" in Chapter 3 of *SAS Language Reference: Concepts*.

Details

The ALTER= option applies to all types of SAS files except catalogs. You can use this option to assign an *alter-password* to a SAS file or to access a Read-protected, Write-protected, or Alter-protected SAS file.

BUFNO= Data Set Option: UNIX

Specifies the number of buffers to be allocated for processing a SAS data set.

Valid in:	DATA step and PROC steps
Category:	Data Set Control
Default:	1
Engine:	V9, V8, V6
UNIX specifics:	default value
See:	"BUFNO= Data Set Option" in *SAS Data Set Options: Reference*

Syntax

BUFNO=*n* | *n*K | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K**
: specifies the number of buffers in multiples of 1 (bytes); 1,024 (kilobytes). For example, a value of `8` specifies 8 buffers, and a value of `1k` specifies 1024 buffers.

***hex*X**
: specifies the number of buffers as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example the value `2dx` specifies 45 buffers.

MIN
: sets the minimum number of buffers to 0, which causes SAS to use the minimum optimal value for the operating environment.

MAX
: sets the number of buffers to the maximum possible number in your operating environment, up to the largest four-byte signed integer, which is 2^{31}–1, or approximately 2 billion.

Details

The buffer number is not a permanent attribute of the data set; it is valid only for the current SAS step. BUFNO= applies to SAS data sets that are opened for input, output, or update.

See Also

Data Set Options:

- “ BUFSIZE= Data Set Option: UNIX” on page 248

System Options:

- “BUFNO System Option: UNIX” on page 372

BUFSIZE= Data Set Option: UNIX

Specifies the size of a permanent buffer page for an output SAS data set.

Valid in:	DATA step and PROC steps
Category:	Data Set Control
Default:	0
Engine:	V9, V8
UNIX specifics:	valid range
See:	“BUFSIZE= Data Set Option” in *SAS Data Set Options: Reference*

Syntax

BUFSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
: specifies the buffer size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). For example, a value of `8` specifies 8 bytes, and a value of `3m` specifies 3,145,728 bytes.

The buffer size can range from 1K to 2G–1. For values greater than 1G, use the *n*M option.

***hex*X**
: specifies the page size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, `2dx` sets the page size to 45 bytes.

MAX
: sets the buffer page size to the maximum possible number in your operating environment, up to the largest four-byte, signed integer, which is 2^{31}–1, or approximately 2 billion bytes.

Details

The BUFSIZE= data set option specifies the buffer size for data sets you are creating. This option is valid only for output data sets.

If you use the default value (0) when you create a SAS data set, the engine calculates a buffer size to optimize CPU and I/O use. This size is the smallest multiple of 8K that can hold 80 observations, but is not larger than 64K.

If you specify a nonzero value when you create a SAS data set, the engine uses that value. If that value cannot hold at least one observation or is not a valid buffer size, the engine rounds the value up to a multiple of 1K.

See Also

System Options:

- "BUFSIZE System Option: UNIX" on page 373

PW= Data Set Option: UNIX

Assigns a READ, WRITE, or ALTER password to a SAS file, and enables access to a password-protected SAS file.

Valid in:	DATA step and PROC steps
Category:	Data Set Control
Default:	none
Engine:	V9, V8, V6
See:	"PW= Data Set Option" in *SAS Data Set Options: Reference*

Syntax

PW=*password*

Required Argument

password
must be a valid SAS name. See “Words in the SAS Language” in Chapter 3 of *SAS Language Reference: Concepts*.

Details

The PW= option applies to all types of SAS files except catalogs. You can use this option to assign a password to a SAS file or to access a password-protected SAS file.

USEDIRECTIO= Data Set Option: UNIX

Turns on direct file I/O for a library that contains the file to which the ENABLEDIRECTIO option has been applied.

Valid in:	DATA step
Category:	Data Set Control
Default:	Off
Engine:	V9, V8
UNIX specifics:	To use this option, you must also use the ENABLEDIRECTIO option in the LIBNAME statement where the libref was assigned.

Syntax

USEDIRECTIO=

Details

The USEDIRECTIO= data set option turns on direct file I/O for a data set that is listed on a DATA statement. The associated libref must have been defined with the ENABLEDIRECTIO option in the LIBNAME statement.

Using ENABLEDIRECTIO on a LIBNAME statement makes direct file I/O possible for data sets in that library. Direct I/O itself is not turned on. You must use the USEDIRECTIO= option to produce direct file I/O.

You can turn on direct file I/O in two ways:

- Use both the ENABLEDIRECTIO and USEDIRECTIO= options in the LIBNAME statement:

  ```
  libname libref-name '.' ENABLEDIRECTIO USEDIRECTIO=yes;
  ```

  In this case, SAS uses direct file I/O on all SAS I/O data sets that are opened using the libref *libref-name*.

- Use ENABLEDIRECTIO in the LIBNAME statement and use USEDIRECTIO= in a DATA statement:

  ```
  libname libref-name '.' ENABLEDIRECTIO;
  data libref-name.data-set-name (USEDIRECTIO=yes);
  ```

  In this case, *libref-name.data-set-name* will be opened for direct file I/O. Other SAS I/O data sets referenced by *libref-name* will not use direct file I/O.

USEDIRECTIO= by itself has no effect. Neither of the following statements open a data set for direct file I/O:

```
libname libref-name '.' USEDIRECTIO=yes;
data libref-name.data-set-name (USEDIRECTIO=yes);
```

Example

The following example uses the ENABLEDIRECTIO LIBNAME option to enable files that are associated with the libref **`test`** to be opened for direct I/O. The USEDIRECTIO= data set option opens **`test.file1`** for direct I/O. **`test.file2`** is not opened for direct I/O.

```
LIBNAME test'.'ENABLEDIRECTIO;
data test.file1(USEDIRECTIO=yes);
   ... more SAS statements ...
run;
data test.file2;
   ... more SAS statements ...
run;
```

See Also

Statements:

- "LIBNAME Statement: UNIX" on page 333

Chapter 12

Formats under UNIX

SAS Formats under UNIX

This section describes SAS formats that have behavior or syntax that is specific to UNIX environments. Each format description includes a brief "UNIX specifics" section that explains which aspect of the data set option is specific to UNIX. Each format is described in this documentation and in *SAS Formats and Informats: Reference*.

Dictionary

HEX*w*. Format: UNIX

Converts real binary (floating-point) numbers to hexadecimal representation.

Category:	Numeric
Alignment:	left
Default:	8
Range:	1 to 16
UNIX specifics:	floating-point representation
See:	"HEXw. Format" in *SAS Formats and Informats: Reference*

Details

The HEX*w.* format converts a real (floating-point) binary number to its hexadecimal representation. When you specify a width value of 1 through 15, the real binary number is truncated to a fixed-point integer before being converted to a hexadecimal number. When you specify 16 for the width, SAS writes the floating-point value of the number, but does not truncate it.

Note: UNIX systems vary widely in their floating-point representation. For more information, see "Reading and Writing Binary Data in UNIX Environments" on page 218.

$HEX*w.* Format: UNIX

Converts character values to hexadecimal representation.

Category:	Character
Alignment:	left
Default:	2
Range:	1 to 32767
UNIX specifics:	produces ASCII codes
See:	"$HEXw. Format" in *SAS Formats and Informats: Reference*

Details

Under UNIX, the $HEX*w.* format produces hexadecimal representations of ASCII codes for characters, with each byte requiring two columns. Therefore, you need twice as many columns to output a value with the $HEX*w.* format.

IB*w.d* Format: UNIX

Writes integer binary (fixed-point) values.

Category:	Numeric
Alignment:	left
Default:	4
Ranges:	1 to 8, 0–10
UNIX specifics:	byte order
See:	"IBw.d Format" in *SAS Formats and Informats: Reference*

Details

The IB*w.d* format writes integer binary (fixed-point) values. Integers are stored in integer-binary, or fixed-point, form. For example, the number 2 is stored as 00000002. If the format includes a *d* value, the data value is multiplied by 10^d.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 218.

PD*w.d* Format: UNIX

Writes data in packed decimal format.

Category:	Numeric
Alignment:	left
Default:	1
Ranges:	1 to 16, 0–31
UNIX specifics:	data representation
See:	"PDw.d Format" in *SAS Formats and Informats: Reference*

Details

The PD*w.d* format writes values in packed decimal format. In packed decimal data, each byte contains two digits. The *w* value represents the number of bytes, not the number of digits. The value's sign is the first byte. Because the entire first byte is used for the sign, you should specify at least a width of 2.

The PD*w.d* format writes missing numerical data as –0. When the PD*w.d* informat reads a value of –0, the result is a value of 0.

For more information, see "Reading and Writing Binary Data in UNIX Environments" on page 218.

PIB*w.d* Format: UNIX

Writes positive integer binary (fixed-point) values.

Category:	Numeric
Alignment:	left
Default:	1
Ranges:	1 to 8, 0–10
UNIX specifics:	byte order
See:	"PIBw.d Format" in *SAS Formats and Informats: Reference*

Details

The PIB*w.d* format writes fixed-point binary values, treating all values as positive. Thus, the high-order bit is part of the value, rather than the value's sign. If a *d* value is specified, the data value is multiplied by 10^d.

For more information, see "Reading and Writing Binary Data in UNIX Environments" on page 218.

RB*w.d* Format: UNIX

Writes real binary (floating-point) data in real binary format.

Category:	Numeric
Alignment:	left
Default:	4
Ranges:	2 to 8, 0–10
UNIX specifics:	floating-point representation
See:	"RBw.d Format" in *SAS Formats and Informats: Reference*

Details

The RB*w.d* format writes numeric data in real binary (floating-point) notation. SAS stores all numeric values in floating-point.

Real binary is the most efficient format for representing numeric values because SAS already represents numbers this way and no conversion is needed.

For more information, see "RBw.d Informat: UNIX" on page 283 and "Reading and Writing Binary Data in UNIX Environments" on page 218.

ZD*w.d* Format: UNIX

Writes numeric data in zoned decimal format.

Category:	Numeric
Alignment:	left
Default:	1
Range:	1 to 32
UNIX specifics:	data representation
See:	"ZDw.d Format" in *SAS Formats and Informats: Reference*

Details

The ZD*w.d* format writes zoned decimal data. This format is also known as overprint trailing numeric format. Under UNIX, the last byte of the field includes the sign with the last digit. The conversion table for the last byte is as follows:

Table 12.1 *Conversion Table*

Digit	ASCII Character	Digit	ASCII Character
0	{	–0	}
1	A	–1	J

2	B	–2	K
3	C	–3	L
4	D	–4	M
5	E	–5	N
6	F	–6	O
7	G	–7	P
8	H	–8	Q
9	I	–9	R

For more information, see "ZDw.d Informat: UNIX" on page 285 and "Reading and Writing Binary Data in UNIX Environments" on page 218.

Chapter 13

Functions and CALL Routines under UNIX

SAS Functions and CALL Routines under UNIX

This section describes SAS functions and CALL routines whose behavior is specific to UNIX environments. Each function and CALL routine description includes a brief "UNIX specifics" section that explains which aspect of the function and CALL routine is specific to UNIX. For more information about all of these functions and CALL routines, see *SAS Functions and CALL Routines: Reference*.

Dictionary

BYTE Function: UNIX

Returns one character in the ASCII collating sequence.

Category:	Character
UNIX specifics:	Uses the ASCII collating sequence
See:	"BYTE Function" in *SAS Functions and CALL Routines: Reference*

Syntax

BYTE(*n*)

Required Argument

n
specifies an integer that represents a specific ASCII character. The value of *n* can range from 0 to 255.

Details

If the BYTE function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 1.

CALL MODULE Routine: UNIX

Calls a specific routine or module that resides in a shared executable library.

Category:	External Files
UNIX specifics:	All
See:	"CALL MODULE Routine" in *SAS Functions and CALL Routines: Reference*

Syntax

CALL MODULE(*<cntl>* ,*module*,*arg-1*,*arg-2...*,*arg-n*);

num=**MODULEN**(*<cntl>* ,*module*,*arg-1*,*arg-2…*,*arg-n*);

char=**MODULEC**(*<cntl>* ,*module*,*arg-1…*,*arg-2*,*arg-n*);

CALL MODULEI (*<cntl>* ,*modulearg-1*,*arg-2...*,*arg-n*);

num=**MODULEIN**(*<cntl>* ,*module*,*arg-1*,*arg-2...*,*arg-n*)

char=**MODULEIC**(*<cntl>* ,*module*,*arg-1*,*arg-2...*,*arg-n*);

Required Arguments

module

specifies the name of the external module to use. The *module* can be specified as a shared library and the routine name or ordinal value, separated by a comma. You do not need to specify the shared library name if you specified the MODULE attribute for the routine in the SASCBTBL attribute table, as long as the routine name is unique (that is, no other routines have the same name in the attribute file).

The module must reside in a shared library, and it must be able to be called externally. Note that while the shared library name is not case sensitive, the routine name is based on the restraints of the routine's implementation language, so the routine name is case sensitive.

If the shared library supports ordinal-value naming, you can provide the shared library name followed by a decimal number, such as 'XYZ,30'.

You can specify *module* as a SAS character expression instead of as a constant; most often, though, you will pass it as a constant.

arg-1, arg-2, ...arg-n

specifies the arguments to pass to the requested routine. Use the proper attributes for the arguments (that is, numeric arguments for numeric attributes and character arguments for character attributes).

CAUTION:

Be sure to use the correct arguments and attributes. If you use incorrect arguments or attributes for a shared library function, you can cause SAS to crash, or you will see unexpected results.

Optional Argument

cntl

is an optional control string whose first character must be an asterisk (*), followed by any combination of the following characters:

I	prints the hexadecimal representations of all arguments to the MODULE function and to the requested shared library routine before and after the shared library routine is called. You can use this option to help diagnose problems that are caused by incorrect arguments or attribute tables. If you specify the I option, the E option is implied.
E	prints detailed error messages. Without the E option (or the I option, which supersedes it), the only error message that the MODULE function generates is "Invalid argument to function," which is usually not enough information to determine the cause of the error.
S*x*	uses *x* as a separator character to separate field definitions. You can then specify *x* in the argument list as its own character argument to serve as a delimiter for a list of arguments that you want to group together as a single structure. Use this option only if you do not supply an entry in the SASCBTBL attribute table. If you do supply an entry for this module in the SASCBTBL attribute table, you should use the FDSTART option in the ARG statement in the table to separate structures.
H	provides brief help information about the syntax of the MODULE routines, the attribute file format, and the suggested SAS formats and informats.

For example, the control string '*IS/' specifies that parameter lists be printed and that the string '/' is to be treated as a separator character in the argument list.

Details

The following functions permit vector and matrix arguments. You can use them only within the IML procedure:

- CALL MODULEI
- MODULEIN
- MODULEIC

For more information, see the documentation for SAS/IML Studio: User's Guide.

The MODULE functions execute a routine *module* that resides in an external (outside SAS) shared library with the specified arguments *arg-1* through *arg-n*.

The MODULE call routine does not return a value, while the MODULEN and MODULEC functions return a number *num* or a character *char*, respectively. Which routine you use depends on the expected return value of the shared library function that you want to execute.

MODULEI, MODULEIC, and MODULEIN are special versions of the MODULE functions that permit vector and matrix arguments. Their return values are still scalar. You can invoke these functions only from PROC IML.

Other than this name difference, the syntax for all six routines is the same.

The MODULE function builds a parameter list by using the information in *arg-1* to *arg-n* and by using a routine description and argument attribute table that you define in a separate file. Before you invoke the MODULE routine, you must define the fileref of SASCBTBL to point to this external file. You can name the file whatever you want when you create it.

This way, you can use SAS variables and formats as arguments to the MODULE function and ensure that these arguments are properly converted before being passed to the shared library routine.

CAUTION:
Using the MODULE function without defining an attribute table can cause SAS to crash, produce unexpected results, or result in severe errors. You need to use an attribute table for all external functions that you want to invoke.

See Also

Functions:

- “PEEKLONG Function: UNIX” on page 278

Other References:

- “The SASCBTBL Attribute Table” on page 108

CALL SLEEP Routine: UNIX

For a specified period of time, suspends the execution of a program that invokes this CALL routine.

Category:	Special
UNIX specifics:	All
See:	“CALL SLEEP Routine” in *SAS Functions and CALL Routines: Reference*

Syntax

CALL SLEEP(*n*<,*unit*>);

Required Argument

n

is a numeric constant that specifies the number of units of time for which you want to suspend execution of a program.

Optional Argument

unit

specifies the unit of time in seconds, which is applied to *n*. For example, 1 corresponds to 1 second, .001 corresponds to 1 millisecond, and 5 corresponds to 5 seconds.

Default: .001

Details

CALL SLEEP puts the DATA step in which it is invoked into a nonactive wait state, using no CPU time and performing no input or output. If you are running multiple SAS processes, each process can execute CALL SLEEP independently without affecting the other processes.

Note: Extended sleep periods can trigger automatic host session termination based on time-out values set at your site. Contact your host system administrator to determine the time-out values that are used at your site.

CALL SYSTEM Routine: UNIX

Submits an operating environment command for execution.

Category:	Special
UNIX specifics:	*Command* must evaluate to a valid UNIX command
See:	"CALL SYSTEM Routine" in *SAS Functions and CALL Routines: Reference*

Syntax

CALL SYSTEM(*command*);

Required Argument

command

specifies any of the following:

- a UNIX command enclosed in quotation marks
- an expression whose value is a UNIX command
- the name of a character variable whose value is a UNIX command

Details

The CALL SYSTEM routine issues operating system commands. The output of the command appears in the window from which you invoked SAS.

The value of the XSYNC system option affects how the CALL SYSTEM routine works.

Note: The CALL SYSTEM routine can be executed within a DATA step. However, neither the X statement nor the %SYSEXEC macro program statement is intended for use during the execution of a DATA step.

In the following example, for each record in **`answer.week`**, if the **`resp`** variable is **`y`**, the CALL SYSTEM routine will mail a message:

```
data _null_;
   set answer.week;
   if resp='y' then
      do;
         call system('mail mgr < $HOME/msg');
      end;
   run;
```

See Also

"Executing Operating System Commands from Your SAS Session" on page 15

COLLATE Function: UNIX

Returns a character string in an ASCII collating sequence.

Category:	Character
UNIX specifics:	Uses ASCII collating sequence
See:	"COLLATE Function" in *SAS Functions and CALL Routines: Reference*

Syntax

COLLATE(*start-position* <,*end-position*>) | (*start-position*<,,*length*>)

Required Argument

start-position
specifies the numeric position in the collating sequence of the first character to be returned.

Optional Arguments

end-position
specifies the numeric position in the collating sequence of the last character to be returned.

length
specifies the number of characters in the collating sequence.

Details

The COLLATE function returns a string of ASCII characters. The ASCII collating sequence contains 256 positions, referenced with the numbers 0 through 255. Characters above 127 correspond to characters used in European languages as defined in the ISO 8859 character set.

Unless you assign the return value of the COLLATE function to a variable with a defined length less than 200, the ASCII collating sequence string is padded with spaces to a length of 200. If the ASCII collating sequence is greater than 200 characters, you must specify the length for the return string in a LENGTH statement. Otherwise, the returned string will be truncated to a length of 200 characters. For more information, see the following examples.

Examples

Example 1: Truncating the Variable Length to 200 Characters

Because the following code does not include a LENGTH statement, the length attribute for the ADDRESS variable is truncated to 200 characters:

```
data sales;
   Address=collate(1,241);
run;
proc contents;
run;
```

Output 13.1 *Portion of PROC CONTENTS Output*

```
Alphabetic List of Variables and Attributes
#          Variable      Type      Len
1          Address       Char      200
```

Because length of ADDRESS is limited to 200 characters, the returned string from the COLLATE function will be limited to 200 characters.

Example 2: Specifying a Length Greater Than 200 Characters

To specify a length greater than 200 characters for a specific variable, you can use the LENGTH statement. In the following code, the length of ADDRESS is specified as 240 characters:

```
data sales;
   length Address $240;
   Address=collate(1,241);
run;
proc contents;
run;
```

Output 13.2 *Portion of PROC CONTENTS Output*

```
Alphabetic List of Variables and Attributes
#      Variable      Type      Len
1      Address       Char      240
```

Because the length of ADDRESS is set to 240 characters, the returned string from the COLLATE function will contain 240 characters.

See Also

Statements:

- "LENGTH Statement: UNIX" on page 333

DINFO Function: UNIX

Returns information about a directory.

Category:	External Files
UNIX specifics:	Directory pathname is the only information available
See:	"DINFO Function" in *SAS Functions and CALL Routines: Reference*

Syntax

DINFO(*directory-id*, *info-item*)

Required Arguments

directory-id
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

info-item
specifies the information item to be retrieved. DINFO returns a blank if the value of *info-item* is invalid.

Details

Directories that are opened with the DOPEN function are identified by a *directory-id*. Use DOPTNAME to determine the names of the available system-dependent information items. Use DOPTNUM to determine the number of directory information items available.

Under UNIX, the only *info-item* available is Directory, which is the pathname of *directory-id*. If *directory-id* points to a list of concatenated directories, then Directory is the list of concatenated directory names.

See Also

Functions:

- "DOPEN Function: UNIX" on page 266
- "DOPTNAME Function: UNIX" on page 267
- "DOPTNUM Function: UNIX" on page 268

DOPEN Function: UNIX

Opens a directory, and returns a directory identifier value.

Category: External Files

UNIX specifics: *fileref* can be assigned with an environment variable

See: "DOPEN Function" in *SAS Functions and CALL Routines: Reference*

Syntax

DOPEN(*fileref*)

Required Argument

fileref
specifies the fileref assigned to the directory. In a DATA step, *fileref* can be a character expression, a string enclosed in quotation marks, or a DATA step variable whose value contains the fileref. In a macro, *fileref* can be any expression.

Details

DOPEN opens a directory and returns a directory identifier value (a number greater than 0) that is used to identify the open directory in other SAS external file access functions. If the directory could not be opened, DOPEN returns 0. The directory to be opened must be identified by a fileref.

DOPTNAME Function: UNIX

Returns directory attribute information.

Category: External Files

UNIX specifics: Directory is the only item available

See: "DOPTNAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

DOPTNAME(*directory-id*, *nval*)

Required Arguments

directory-id
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

nval
specifies the sequence number of the information item.

Details

Under UNIX, the only directory information item available is Directory, which is the pathname of the *directory-id*. The *nval*, or sequence number, of Directory is 1. If *directory-id* points to a list of concatenated directories, then Directory is the list of concatenated directory names.

DOPTNUM Function: UNIX

Returns the number of information items that are available for a directory.

Category:	External Files
UNIX specifics:	Directory is the only item available
See:	"DOPTNUM Function" in *SAS Functions and CALL Routines: Reference*

Syntax

DOPTNUM(*directory-id*)

Required Argument

directory-id
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

Details

Under UNIX, only one information item is available for a directory. The name of the item is Directory; its value is the pathname or list of pathnames for *directory-id*, and its sequence number is 1. Because only one information item is available for a directory, this function will return a value of 1.

FDELETE Function: UNIX

Deletes an external file or an empty directory.

Category:	External Files
UNIX specifics:	*fileref* can be assigned with an environment variable
See:	"FDELETE Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FDELETE("*fileref*")

Required Argument

fileref
specifies the fileref that is assigned to the external file or directory. The fileref cannot be associated with a list of concatenated filenames or directories. If the fileref is associated with a directory, the directory must be empty. You must have permission to delete the file. See the UNIX man page for **`chmod`** for more information about permissions.

Under UNIX, *fileref* can also be an environment variable. The *fileref* must be enclosed in double quotation marks.

Details

FDELETE returns 0 if the operation was successful, or a nonzero number if it was not successful.

FEXIST Function: UNIX

Verifies the existence of an external file that is associated with a fileref.

Category:	External Files
UNIX specifics:	*fileref* can be assigned with an environment variable
See:	"FEXIST Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FEXIST(*fileref*)

Required Argument

fileref
specifies the fileref assigned to the external file or directory. In a DATA step, *fileref* can be a character expression, a string enclosed in quotation marks, or a DATA step variable whose value contains the fileref. In a macro, *fileref* can be any expression.

Under UNIX, *fileref* can also be an environment variable. The *fileref* or the environment variable that you specify must be enclosed in double quotation marks.

Details

The FEXIST function returns a value of 1 if the external file that is associated with *fileref* exists, and a value of 0 if the file does not exist.

FILEEXIST Function: UNIX

Verifies the existence of an external file by its physical name.

Category:	External Files
UNIX specifics:	*filename* can be assigned with an environment variable
See:	"FILEEXIST Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FILEEXIST(*filename*)

Required Argument

filename
specifies a fully qualified physical filename of the external file. In a DATA step, *filename* can be a character expression, a string in quotation marks, or a DATA step variable. In a macro, *filename* can be any expression.

Under UNIX, *filename* can also be an environment variable. The *filename* or the environment variable that you specify must be enclosed in double quotation marks.

Details

FILEEXIST returns 1 if the external file exists, and 0 if the external file does not exist.

You can check for the existence of a directory by using FILEEXIST.

FILENAME Function: UNIX

Assigns or deassigns a fileref for an external file, directory, or output device.

Category:	External Files
UNIX specifics:	*fileref* can be assigned with an environment variable; valid values of *device-type* and *host-options*
See:	“FILENAME Function” in *SAS Functions and CALL Routines: Reference*

Syntax

FILENAME(*fileref*, *filename* <,*device-type*<,*“host-options”*<,*dir-ref*>>>)

Required Arguments

fileref
specifies the *fileref* to assign to an external file. In a DATA step, *fileref* can be a character expression, a string enclosed in quotation marks, or a DATA step variable whose value contains the fileref. In a macro (for example, in the %SYSFUNC function), *fileref* is the name of a macro variable (without an ampersand) whose value contains the fileref to assign to the external file. (For information, see the “FILENAME Function” in *SAS Functions and CALL Routines: Reference*.)

Under UNIX, the *fileref* can be a UNIX environment variable. The *fileref* or the environment variable that you specify must be enclosed in double quotation marks.

filename
specifies the external file. Specifying a blank filename (“ ”) deassigns a *fileref* that was previously assigned.

Under UNIX, the filename differs according to the device type. For more information that is appropriate for each device, see “Device Information in the FILENAME Statement” on page 327. Remember that UNIX filenames are case sensitive.

In a DATA step, *filename* can be a character expression, a string enclosed in quotation marks, or a DATA step variable whose value contains the filename. In a macro, *filename* can be any expression.

Optional Arguments

device-type
specifies the type of device or the access method that is used if the fileref points to an input or output device or location that is not a physical file. It can be any one of the devices listed in “Device Information in the FILENAME Statement” on page 327. DISK is the default device type.

host-options

are options that are specific to UNIX. You can use any of the options that are available in the FILENAME statement. See "FILENAME Statement: UNIX" on page 323 for a description of the host options.

Requirement: Enclose host options in quotation marks. If you have multiple host options, then all of the host options must be enclosed in one set of quotation marks. The following example shows the syntax:

```
rc=filename("try","MISCHL.FLAT.FILE1","ftp",
               'user="mischl1",host="sdcunx",prompt');
```

dir-ref

specifies the fileref that is assigned to the directory in which the external file resides.

Details

FILENAME returns a 0 if the operation is successful, and a nonzero number if it was not successful.

If you use the FTP access method to communicate with a remote system, SAS might return the following error message:

```
ERROR: Physical file does not exist.
```

This error is likely to occur when the fully qualified data set name is specified within single quotation marks. For example:

```
FILENAME fileref FTP 'system.dataset.name' USER='username'
                 PASS='password' HOST='ip_address';
```

By default, SAS appends the profile prefix to the beginning of the data set name. To prevent the profile prefix from being appended, enclose the data set name with both double and single quotation marks:

```
FILENAME fileref FTP "'external_file'" USER='username'
                 PASS='password' HOST='ip_address';
```

FILEREF Function: UNIX

Verifies whether a fileref has been assigned for the current SAS session.

Category:	External Files
UNIX specifics:	*fileref* can be assigned with an environment variable
See:	"FILEREF Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FILEREF(*fileref*)

Required Argument

fileref

specifies the fileref assigned to be validated. In a DATA step, *fileref* can be a character expression, a string enclosed in quotation marks, or a DATA step variable whose value contains the fileref. In a macro, *fileref* can be any expression.

Under UNIX, *fileref* can also be a UNIX environment variable. The *fileref* or the environment variable that you specify must be enclosed in double quotation marks.

Details

A negative return code indicates that the fileref exists, but the physical file associated with the fileref does not exist. A positive value indicates that the fileref is not assigned. A value of zero indicates that the fileref and external file both exist.

For more information, see “FILENAME Function: UNIX” on page 270.

FINFO Function: UNIX

Returns the value of a file information item for an external file.

Category:	External Files
UNIX specifics:	*info-item* is available
See:	“FINFO Function” in *SAS Functions and CALL Routines: Reference*

Syntax

FINFO(*file-id*, *info-item*)

Required Arguments

file-id
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

info-item
specifies the name of the file information item to be retrieved. This value is a character value. *Info-item* is either a variable containing a valid value or the valid value in quotation marks.

Under UNIX, *info-item* for disk files can have one of the following values:

- Filename
- Owner Name
- Group Name
- Access Permission
- File Size (bytes)

If you concatenate filenames, then an additional *info-item* is available: File List.

If you are using pipe files, then the only valid value for *info-item* is PIPE Command.

Details

The FINFO function returns the value of a system-dependent information item for an external file that was previously opened and assigned a *file-id* by the FOPEN function. FINFO returns a blank if the value given for *info-item* is invalid.

For an example of how to use the FINFO function, see “Example: File Attributes When Using the Pipe Device Type” on page 274.

See Also

Functions:

- "FOPEN Function" in *SAS Functions and CALL Routines: Reference*

FOPTNAME Function: UNIX

Returns the name of an item of information about an external file.

Category:	External Files
UNIX specifics:	Information items available
See:	"FOPTNAME Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FOPTNAME(*file-id*, *nval*)

Required Arguments

file-id
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

nval
specifies the number of the file information item to be retrieved. The following table shows the values that *nval* can have in UNIX operating environments for single, pipe, and concatenated files:

File Information Items

nval	Single File	Pipe Files	Concatenated Files
1	Filename	PIPE Command	Filename
2	Owner Name		File List
3	Group Name		Owner Name
4	Access Permission		Group Name
5	File Size (bytes)		Access Permission
6			File Size (bytes)

Details

FOPTNAME returns a blank if an error occurs.

Example: File Attributes When Using the Pipe Device Type

The following example creates a data set that contains the NAME and VALUE attributes returned by the FOPTNAME function when you are using pipes:

```
data fileatt;
   length name $ 20 value $ 40;
   drop fid j infonum;
   filename mypipe pipe 'UNIX-command';
   fid=fopen("mypipe","s");
   infonum=foptnum(fid);
   do j=1 to infonum;
      name=foptname(fid,j);
      value=finfo(fid,name);
      put 'File attribute' name 'has a value of ' value;
      output;
   end;
run;
```

The following statement should appear in the SAS log.

Output 13.3 *SAS Log Output*

```
File attribute Pipe Command has a value of UNIX-command
```

Unix-command is the UNIX command or program where you are piping your output or where you are reading your input. This command or program must be either fully qualified or defined in your PATH environment variable.

See Also

Functions:

- "FINFO Function: UNIX" on page 272
- "FOPTNUM Function: UNIX" on page 274
- "FOPEN Function" in *SAS Functions and CALL Routines: Reference*

FOPTNUM Function: UNIX

Returns the number of information items that are available for an external file.

Category: External Files

UNIX specifics: Information items available

See: "FOPTNUM Function" in *SAS Functions and CALL Routines: Reference*

Syntax

FOPTNUM(*file-id*)

Required Argument

file-id
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

Details

Under UNIX, five information items are available for all types of files:

- Filename
- Owner Name
- Group Name
- Access Permission
- File Size (bytes)

If you concatenate filenames, then an additional information item is available: File List. If you are using pipe files, then the only information item available is PIPE Command.

The *open-mode* specified in the FOPEN function determines the value that FOPTNUM returns.

Table 13.1 *Open Mode and FOPTNUM Values*

Open Mode	FOPTNUM Value	Information Items Available
Append Input Update	6 for concatenated files 5 for single files	All information items available.
Output	5 for concatenated files 4 for single files	Because the file is open for output, the File Size information type is unavailable.
Sequential (using Pipe Device Type)	1	The only information item available is PIPE Command.

For an example of how to use the FOPTNUM function, see "Example: File Attributes When Using the Pipe Device Type" on page 274.

See Also

Functions:

- "FINFO Function: UNIX" on page 272
- "FOPTNAME Function: UNIX" on page 273
- "FOPEN Function" in *SAS Functions and CALL Routines: Reference*

MODEXIST Function: UNIX

Determines whether a product image exists in the release of SAS that you have installed.

Category:	Numeric
UNIX specifics:	*pathname* is available
See:	"MODEXIST Function" in *SAS Functions and CALL Routines: Reference*

Syntax

MODEXIST ('*product-name*' | '*pathname*')

Required Arguments

'product-name'
specifies a character constant, variable, or expression that is the name of the product image that you are checking.

'pathname'
specifies the pathname for the product image that you are checking.

Details

The MODEXIST function searches the directories that are listed in the *pathname* argument for an executable module. The name of the executable module is passed to MODEXIST. MODEXIST returns 1 if the module is found, and 0 if the module is not found.

MOPEN Function: UNIX

Opens a file by directory ID and member name, and returns either the file identifier or a 0.

Category:	External Files
UNIX specifics:	OPEN modes
See:	"MOPEN Function" in *SAS Functions and CALL Routines: Reference*

Syntax

MOPEN(*directory-id*,*member-name*<,*open-mode*<,*record-length*<,*record-format*>>>)

Required Argument

open-mode
specifies the type of access to the file:

A	APPEND mode allows writing new records after the current end of the file.
I	INPUT mode allows reading only (default).

O	OUTPUT mode defaults to the OPEN mode specified in the host option in the FILENAME statement or function. If no host option is specified, it allows writing new records at the beginning of the file.
S	Sequential input mode is used for pipes and other sequential devices such as hardware ports.
U	UPDATE mode allows both reading and writing.
W	Sequential update mode is used for pipes and other sequential devices such as ports.

Details

Note: This version is a simplified version of the MOPEN function syntax. For the complete syntax and its explanation, see the “MOPEN Function” in *SAS Functions and CALL Routines: Reference* .

MOPEN returns the identifier for the file, or 0 if the file could not be opened.

PATHNAME Function: UNIX

Returns the physical name of a SAS library or an external file, or returns a blank.

Category:	SAS File I/O
UNIX specifics:	*fileref* or *libref* argument can also specify a UNIX environment variable
See:	“PATHNAME Function” in *SAS Functions and CALL Routines: Reference*

Syntax

PATHNAME((*fileref* | *libref*) <,*search-ref*>)

Required Arguments

fileref
specifies the fileref that is assigned to the external file. In a DATA step, *fileref* can be a character expression, a string enclosed in quotation marks, or a DATA step variable whose value contains the fileref. In a macro, *fileref* can be any expression.

The value of *fileref* can be a UNIX environment variable.

libref
specifies the libref that is assigned to a SAS library. In a DATA step, *libref* can be a character expression, a string enclosed in quotation marks, or a DATA step variable whose value contains the libref. In a macro, *libref* can be any expression.

The value of *libref* can be a UNIX environment variable.

Optional Argument

search-ref
specifies whether to search for a fileref or a libref.

F	specifies a search for a fileref.
L	specifies a search for a libref.

Details

PATHNAME returns the physical name of an external file or SAS library, or a blank if *fileref* or *libref* is invalid.

For more information about using a UNIX environment variable for *fileref* or *libref*, see "FILENAME Function: UNIX" on page 270.

PEEKLONG Function: UNIX

Stores the contents of a memory address in a numeric variable on 32-bit and 64-bit platforms.

Category:	Special
UNIX specifics:	All
See:	"PEEKLONG Function" in *SAS Functions and CALL Routines: Reference*

Syntax

PEEKCLONG(*address*,*length*);

PEEKLONG(*address*,*length*);

Required Arguments

address
specifies the character string that is the memory address.

length
specifies the data length.

Details

CAUTION:
Use the PEEKLONG functions only to access information returned by one of the MODULE functions.

The PEEKLONG function returns a value of *length* that contains the data that starts at the memory *address*.

Here are the variations of the PEEKLONG functions:

PEEKCLONG
accesses character strings.

PEEKLONG
accesses numeric values.

Usually, when you need to use one of the PEEKLONG functions, you will use PEEKCLONG to access a character string. The PEEKLONG function is mentioned for completeness.

RANK Function: UNIX

Returns the position of a character in the ASCII collating sequence.

Category:	Character

UNIX specifics: Uses ASCII collating sequence

See: "RANK Function" in *SAS Functions and CALL Routines: Reference*

Syntax

RANK(*x*)

Required Argument

x

specifies a character constant, variable, or expression that contains a character in the ASCII collating sequence. If the length of *x* is greater than 1, you receive the rank of the first character in the string.

Details

Because UNIX uses the ASCII character set, the RANK function returns an integer that represents the position of a character in the ASCII collating sequence.

SYSGET Function: UNIX

Returns the value of the specified operating environment variable.

Category: Special

UNIX specifics: *environment-variable* is a UNIX environment variable

See: "SYSGET Function" in *SAS Functions and CALL Routines: Reference*

Syntax

SYSGET('*environment-variable*')

Required Argument

environment-variable

is the name of a UNIX environment variable.

Details

The SYSGET function returns the value of an environment variable as a character string. For example, this statement returns the value of the HOME environment variable:

```
here=sysget('HOME');
```

TRANSLATE Function: UNIX

Replaces specific characters in a character expression.

Category: Character

UNIX specifics: *to* and *from* arguments are required

See: "TRANSLATE Function" in *SAS Functions and CALL Routines: Reference*

Syntax

TRANSLATE(*source*,*to-1*,*from-1*<,…*to-n*,*from-n*>)

Required Arguments

source
specifies a constant, variable, or expression that contains the original character value.

to
specifies the characters that you want TRANSLATE to use as substitutes.

from
specifies the characters that you want TRANSLATE to replace.

Details

Note: This version is a simplified version of the TRANSLATE function syntax. For the complete syntax and its explanation, see the "TRANSLATE Function" in *SAS Functions and CALL Routines: Reference*.

Under UNIX, you must specify pairs of *to* and *from* arguments, and you can use a comma as a placeholder.

Chapter 14

Informats under UNIX

SAS Informats under UNIX

This section describes SAS informats that have behavior or syntax that is specific to UNIX environments. Each informat description includes a brief "UNIX specifics" section that explains which aspect of the informat is specific to UNIX. All of these informats are described in this documentation and in *SAS Formats and Informats: Reference*.

Dictionary

HEX*w*. Informat: UNIX

Converts hexadecimal positive binary values to either fixed-point or floating-point binary values.

Category:	Numeric
Default:	8
Range:	1 to 16
UNIX specifics:	floating-point representation
See:	"HEXw. Informat" in *SAS Formats and Informats: Reference*

Details

The HEX*w*. informat converts the hexadecimal representation of positive binary numbers to real floating-point binary values. The width value of the HEX*w*. informat determines whether the input represents an integer (fixed-point) or real (floating-point) binary number. When you specify a width of 1 through 15, the informat interprets the input hexadecimal as an integer binary number. When your specify 16 for the width value, the informat interprets the input hexadecimal as a floating-point value.

For more information, see "Reading and Writing Binary Data in UNIX Environments" on page 218.

$HEX*w*. Informat: UNIX

Converts hexadecimal data to character data.

Category:	Character
Default:	2
Range:	1 to 32,767
UNIX specifics:	values are interpreted as ASCII values
See:	"$HEXw. Informat" in *SAS Formats and Informats: Reference*

Details

The $HEX*w*. informat converts every two digits of hexadecimal data into one byte of character data. Use the $HEX*w*. informat to encode hexadecimal values into a character variable when your input data is limited to printable characters. SAS under UNIX interprets values that are read with this informat as ASCII values.

IB*w.d* Informat: UNIX

Reads integer binary (fixed-point) values.

Category:	Numeric
Default:	4
Ranges:	1 to 8, 0 to 10
UNIX specifics:	byte values
See:	"IBw.d Informat" in *SAS Formats and Informats: Reference*

Details

The IB*w.d* informat reads fixed-point binary values. For integer binary data, the high-order bit is the value's sign: 0 for positive values, 1 for negative. Negative values are represented in two's-complement notation. If the informat includes a *d* value, the data value is divided by 10^d.

For more information, see "Reading and Writing Binary Data in UNIX Environments" on page 218.

PD*w.d* Informat: UNIX

Reads data that is stored in packed decimal format.

Category:	Numeric
Default:	1
Ranges:	1 to 16, 0 to 31
UNIX specifics:	data representation
See:	“PDw.d Informat” in *SAS Formats and Informats: Reference*

Details

The PD*w.d* informat reads packed decimal data. Although it is usually impossible to type packed decimal data directly from a console, many programs write packed decimal data.

Each byte contains two digits in packed decimal data. The value's sign is the first byte. Because the entire first byte is used for the sign, you should specify at least a width of 2.

The PD*w.d* format writes missing numerical data as –0. When the PD*w.d* informat reads a value of –0, the result is a value of 0.

For more information, see “Reading and Writing Binary Data in UNIX Environments” on page 218.

PIB*w.d* Informat: UNIX

Reads positive integer binary (fixed-point) values.

Category:	Numeric
Default:	1
Ranges:	1 to 8, 0 to 10
UNIX specifics:	byte order
See:	“PIBw.d Informat” in *SAS Formats and Informats: Reference*

Details

The PIB*w.d* informat reads integer binary (fixed-point) values. Positive integer binary values are the same as integer binary (see “IBw.d Informat: UNIX” on page 282), except that all values are treated as positive. Thus, the high-order bit is part of the value rather than the value's sign. If the informat includes a *d* value, the data value is divided by 10^d.

For more information, see “Reading and Writing Binary Data in UNIX Environments” on page 218.

RB*w.d* Informat: UNIX

Reads numeric data that is stored in real binary (floating-point) notation.

Category:	Numeric
Default:	4
Ranges:	2 to 8, 0 to 10
UNIX specifics:	floating-point representation; supports single-precision numbers only for those applications that truncate numeric data
See:	"RBw.d Informat" in *SAS Formats and Informats: Reference*

Details

The RB*w.d* informat reads numeric data that is stored in real binary (floating-point) notation. SAS stores all numeric values in floating-point.

It is usually impossible to type floating-point binary data directly from a console, but many programs write floating-point binary data. Use caution if you are using the RB*w.d* informat to read floating-point data created by programs other than SAS because the RB*w.d* informat is designed to read only double-precision data.

All UNIX systems that are currently supported by SAS use the IEEE standard for floating-point representation. This representation supports both single-precision and double-precision floating-point numbers. Double-precision representation has more bytes of precision, and the data within the representation is interpreted differently. For example, for single-precision, the value of 1 in hexadecimal representation is **3F800000**. For double-precision, the hexadecimal representation of 1 is **3FF0000000000000**.

The RB*w.d* informat is designed to read only double-precision data. It supports widths less than 8 only for applications that truncate numeric data for space-saving purposes. RB4. does *not* expect a single-precision floating-point number; it expects a double-precision number truncated to four bytes. Using the example of 1 above, RB4. expects **3FF00000** to be the hexadecimal representation of the four bytes of data to be interpreted as 1. If given **3F800000**, the single-precision value of 1, a different number results.

External programs such as those programs that are written in the C and Fortran languages can produce only single-precision or double-precision floating-point numbers. No length other than four or eight bytes is allowed. RB*w.d* allows a length of 3 through 8, depending on the storage that you need to save.

The FLOAT4. informat has been created to read a single-precision floating-point number. If you read 3F800000 with FLOAT4., the result is a value of 1.

To read data created by a C or Fortran program, you need to decide on the proper informat to use. If the floating-point numbers require an eight-byte width, you should use the RB8. informat. If the floating point numbers require a four-byte width, you should use FLOAT4.

Consider the following C example:

```
#include <stdio.h>
main() {
FILE *fp;
float x[3];
fp = fopen("test.dat","wb");
x[0] = 1; x[1] = 2; x[2] = 3;
fwrite((char *)x,sizeof(float),3,fp);
fclose(fp);
}
```

The file test.dat contains `3f8000004000000040400000` in hexadecimal representation.

The following statements read test.dat correctly:

```
data _null_;
   infile 'test.dat';
   input (x y z) (float4.);
run;
```

Also available is the IEEE*w.d* informat, which reads IEEE floating-point data. On UNIX systems, IEEE8. is equivalent to RB8., and IEEE4. is equivalent to FLOAT4. IEEE*w.d* can be used on any platform, as long as the original IEEE binary data originated on a platform that uses the IEEE representation.

For more information, see "Reading and Writing Binary Data in UNIX Environments" on page 218.

ZD*w.d* Informat: UNIX

Reads zoned decimal data.

Category:	Numeric
Default:	1
Range:	1 to 32
UNIX specifics:	last byte includes the sign; data representation
See:	"ZDw.d Informat" in *SAS Formats and Informats: Reference*

Details

The ZD*w.d* informat reads zoned decimal data; it is also known as overprint trailing numeric format. Under UNIX, the last byte of the field includes the sign along with the last digit. The conversion table for the last byte is as follows:

Digit	ASCII Character	Digit	ASCII Character
0	{	–0	}
1	A	–1	J
2	B	–2	K
3	C	–3	L
4	D	-4	M
5	E	–5	N
6	F	-6	O
7	G	–7	P
8	H	–8	Q

9	I	–9	R

For more information, see “ZDw.d Format: UNIX” on page 256 and “Reading and Writing Binary Data in UNIX Environments” on page 218.

Chapter 15

Macro Facility under UNIX

About the Macro Facility under UNIX

Most features of the SAS macro facility are valid in all operating environments. This documentation discusses only those components of the macro facility that depend on the UNIX environment. For more information, see the following documentation:

- *SAS Macro Language: Reference*
- *SAS Macro Facility Tips and Techniques*
- the online Help for the macro facility

Automatic Macro Variables in UNIX Environments

The following automatic macro variables are valid in all operating environments, but their values are determined by the operating environment:

SYSCC
: contains the current SAS condition code. Upon exit, SAS translates this condition code to a return code that has a meaningful value for the operating environment.

Note: The value of SYSCC might not match the return code returned by the operating system.

Under UNIX, the following codes can be returned:

0
: Normal completion

1
: SAS issued warnings

2
: SAS issued errors

3
: ABORT;

4
: ABORT RETURN *n*;

5
: ABORT ABEND *n*;

6
: Internal error

Note: When ERRORCHECK=NORMAL, the return code will be 0, even if an error exists in a LIBNAME or FILENAME statement, or in a LOCK statement in SAS/SHARE software. Also, the SAS job or session will not abort when the %INCLUDE statement fails due to a nonexistent file. For more information, see the “ERRORCHECK= System Option” in *SAS System Options: Reference*.

SYSDEVIC
: contains the name of the current graphics device. The current graphics device is determined by the DEVICE system option. Contact your on-site SAS support personnel to determine which graphics devices are available at your site. For information, see “DEVICE System Option: UNIX” on page 377 and “DEVICE= System Option” in *SAS System Options: Reference*.

SYSENV
: reports whether SAS is running interactively. Values for SYSENV are **FORE** when the TERMINAL system option is in effect, and **BACK** when the NOTERMINAL system option is in effect.

SYSJOBID
: lists the process identification number (PID) of the process that is executing SAS (for example, 00024).

SYSMAXLONG
: returns the maximum long integer value allowed under UNIX, which is 9,007,199,254,740,992. On 32-bit systems, the maximum is 2,147,483,647.

SYSRC
: holds the decimal value of the exit status code that is returned by the last UNIX command executed from your SAS session. The following output shows an interactive line mode SAS session that shows two sample SYSRC values:

Output 15.1 *Sample SYSRC Values*

```
1? x 'data';
/bin/ksh: data: not found
2? %put UNIX exit status code is &sysrc;
UNIX exit status code is 256
3? x 'date';
Tue Mar 15 09:41:27 CST 2011
4? %put UNIX exit status code is now &sysrc;
UNIX exit status code is now 0
```

SYSSCP
: returns the abbreviation for your processor architecture, such as **`HP IPF`**, **`SUN 64`**, or **`AIX 64`**.

SYSSCPL
: returns the name of the specific UNIX environment that you are using, such as **`HP-UX`**, **`SunOS`**, or **`AIX`**. This variable returns the same value that is returned by the UNIX command **`uname`**.

Macro Statements in UNIX Environments

The arguments that can be entered with the following statements depend on the operating environment:

%SYSEXEC
: executes UNIX commands. It is similar to the X statement described in "Executing Operating System Commands from Your SAS Session" on page 15. The %SYSEXEC statement enables you to execute operating environment commands immediately and, if necessary, determine whether they executed successfully by examining the value of the automatic macro variable SYSRC. You can use the %SYSEXEC statement inside a macro or in open code. The form of the %SYSEXEC statement is as follows, where *command* can be any UNIX command:

%SYSEXEC <*command*>;

For example, the following code writes the status of the default printer to your UNIX shell:

```
%sysexec lpstat;
```

Entering %SYSEXEC without a UNIX command starts a new shell, except under the X interface to SAS. For more information, see "Executing Operating System Commands from Your SAS Session" on page 15.

Macro Functions in UNIX Environments

The following functions have operating environment dependencies:

%SCAN
: searches for a word that is specified by its position in a string. Here is the form of the %SCAN function:

%SCAN(*argument*,*n*,<*delimiters*>);
On ASCII systems, the default delimiters are the following:

```
blank . < ( + & ! $ * ) ; ^ - / , % |
```

%SYSGET
returns the character string that is the value of the environment variable passed as the argument. Both UNIX and SAS environment variables can be translated using the %SYSGET function. A warning message is written if the global variable does not exist. Here is the form of the %SYSGET function:

%SYSGET(*environment-variable*);

For example, the following code writes the value of the HOME environment variable to the SAS log:

```
%let var1=%sysget(HOME); %put &var1;
```

SAS System Options Used by the Macro Facility in UNIX Environments

The following system options have operating environment dependencies:

MSYMTABMAX
specifies the maximum amount of memory available to all symbol tables (global and local, combined). Under UNIX, the default value for this option is 4M.

MVARSIZE
specifies the maximum number of bytes for any macro variable stored in memory. Under UNIX, the default value for this option is 32K.

SASAUTOS
specifies the AUTOCALL library. For more information, see "The SASAUTOS System Option" on page 291.

Using Autocall Libraries in UNIX Environments

What Is an Autocall Library?

An autocall library contains files that define SAS macros. The following sections discuss aspects of autocall libraries that are dependent on the operating environment. For more information, see *SAS Macro Language: Reference*.

Available Autocall Macros

There are two types of autocall macros, those macros that are provided by SAS, and those macros that you define yourself. To use the autocall facility, you must have the MAUTOSOURCE system option set.

When SAS is installed, the SASAUTOS system option is defined in the configuration file to refer to the location of the default macros supplied by SAS. The products licensed at your site determine the autocall macros that you have available. You can also define your own autocall macros and store them in one or more directories. SAS does not recognize autocall macros if their filenames are written in uppercase or in mixed case. Use only filenames that are lowercase.

Guidelines for Naming Macro Files

Macro names in SAS are case insensitive, but they all map to a lowercase filename. If you store autocall macros in a UNIX directory, the file extension must be **`.sas`**, and the filename must be entirely in lowercase. In the UNIX environment, each macro file in the directory must contain a macro definition with a macro name that matches the filename. For example, a file named **`prtdata.sas`** should define a macro named **`prtdata`**.

The SASAUTOS System Option

To use your own autocall macros in your SAS program, specify their directories with the SASAUTOS system option. For more information, see "SASAUTOS System Option: UNIX" on page 414.

Note: The SASAUTOS system option under UNIX does not recognize filenames that are in uppercase or mixed case.

You can set the SASAUTOS system option when you start SAS, or you can use it in an OPTIONS statement during your SAS session. However, autocall libraries specified with the OPTIONS statement override any previous specification.

If you use the CONFIG system option to specify a configuration file, add your autocall library to the library concatenation supplied by SAS. If you use the default configuration files (sasv9.cfg), specify your autocall library there.

Autocall libraries are searched in the order in which you specify them.

Example: Setting Up and Testing a Macro in an Autocall Library

This example shows how to set up and test a macro in an autocall library.

The following output shows the results of executing two UNIX (**`cat`**) commands to display the contents of two files, and a SAS command to run the autocall.sas program:

Output 15.2 *AUTOCALL Library Example*

```
$ cat maclib/testauto.sas
%macro testauto;
x echo 'Autocall library is working.';
%mend testauto;
$ cat source/autocall.sas
filename sysautos ('!SASROOT/sasautos' '$HOME/test/sasautos');
options mautosource sasautos=(sysautos '$HOME/macros/maclib');
%testauto
%TestAuto
%TESTAUTO
$ sas source/autocall.sas
Autocall library is working.
Autocall library is working.
Autocall library is working.
```

Chapter 16

Procedures under UNIX

SAS Procedures under UNIX

This section describes SAS procedures that have behavior or syntax that is specific to UNIX environments. Each procedure description includes a brief "UNIX specifics" section that explains which aspect of the procedure is specific to UNIX. Each procedure is described in both this documentation and in the *Base SAS Procedures Guide*.

Dictionary

CATALOG Procedure: UNIX

Manages entries in SAS catalogs.

UNIX specifics: FILE= option in the CONTENTS statement

See: Chapter 9, "CATALOG Procedure" in *Base SAS Procedures Guide*

Syntax

PROC CATALOG CATALOG=<*libref.*> *catalog* <ENTRYTYPE=*etype*> <KILL> ;
  CONTENTS <OUT=*SAS-data-set*> <FILE=*fileref*> ;

Optional Argument

fileref
names a file specification that is specific to the UNIX operating environment.

Details

Note: This version is a simplified version of the CATALOG procedure syntax. For the complete syntax and its explanation, see Chapter 9, “CATALOG Procedure” in *Base SAS Procedures Guide*.

The FILE= option in the CONTENTS statement of the CATALOG procedure accepts a fileref. If the name specified does not correspond to a fileref, a file with that name and an extension of **`.lst`** is created in the current directory. For example, if **`myfile`** is not a fileref, the following code creates the file **`myfile.lst`** in your current directory:

```
proc catalog catalog=sasuser.profile;
   contents file=myfile;
run;
```

SAS writes the following output to the Log:

```
NOTE: 6 entries have been written to the output file /users/userid/MYFILE.lst.
```

Note: The filename that is created is always stored in lowercase, even if you specified it in uppercase. In the SAS log, however, the filename is listed in uppercase.

CIMPORT Procedure: UNIX

Restores a transport file that was created by the CPORT procedure.

UNIX specifics: name and location of transport file

See: Chapter 11, “CIMPORT Procedure” in *Base SAS Procedures Guide*

Syntax

PROC CIMPORT *destination=libref* | *<libref.> member-name <option(s)>* ;

Required Arguments

destination
identifies the files in the transport file as a single SAS data set, single SAS catalog, or multiple members of a SAS library.

libref* | *<libref.>member-name
specifies the name of the SAS data set, catalog, or library to be created from the transport file.

Note that this version is a simplified version of the CIMPORT procedure syntax. For the complete syntax and its explanation, see the CIMPORT procedure.

Details

Note: Starting in SAS 9.1, you can use the MIGRATE procedure to migrate a SAS library from a previous release. For more information, see “Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments” on page 47, the MIGRATE procedure, and Cross-Release Compatibility at the **`Technical Support Web site`**.

The CIMPORT procedure *imports* a transport file that was created (*exported*) by the CPORT procedure. The transport file can contain a SAS data set, a SAS catalog, or an entire SAS library.

Typically, the INFILE= option is used to designate the source of the transport file. If this option is omitted, CIMPORT uses the default file Sascat.dat in the current directory as the transport file.

Note: CIMPORT works only with transport files created by the CPORT procedure. If the transport file was created using the XPORT engine with the COPY procedure, then another PROC COPY must be used to restore the transport file. For more information, see Chapter 14, "COPY Procedure" in *Base SAS Procedures Guide*.

Example: Moving Data Sets

For this example, a SAS library that contains multiple SAS data sets was exported to a file (called **`transport-file`**) using the CPORT procedure on a foreign host. The transport file is then moved by a binary transfer to the receiving host.

The following code extracts all of the SAS data sets and catalogs stored within the transport file and restores them to their original state in the new library, called **`SAS-library`**.

```
libname newlib 'SAS-library';
filename tranfile 'transport-file';

proc cimport lib=newlib infile=tranfile;
run;
```

See Also

Procedures:

- "CPORT Procedure: UNIX" on page 301

Other References:

- "Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments" on page 47
- Chapter 1, "Moving and Accessing SAS Files between Operating Environments," in *Moving and Accessing SAS Files*

CONTENTS Procedure: UNIX

Prints the description of the contents of one or more files from a SAS library.

UNIX specifics:	information displayed in the SAS output
See:	Chapter 13, "CONTENTS Procedure" in *Base SAS Procedures Guide*

Syntax

PROC CONTENTS<*option(s)*> ;

Comparisons

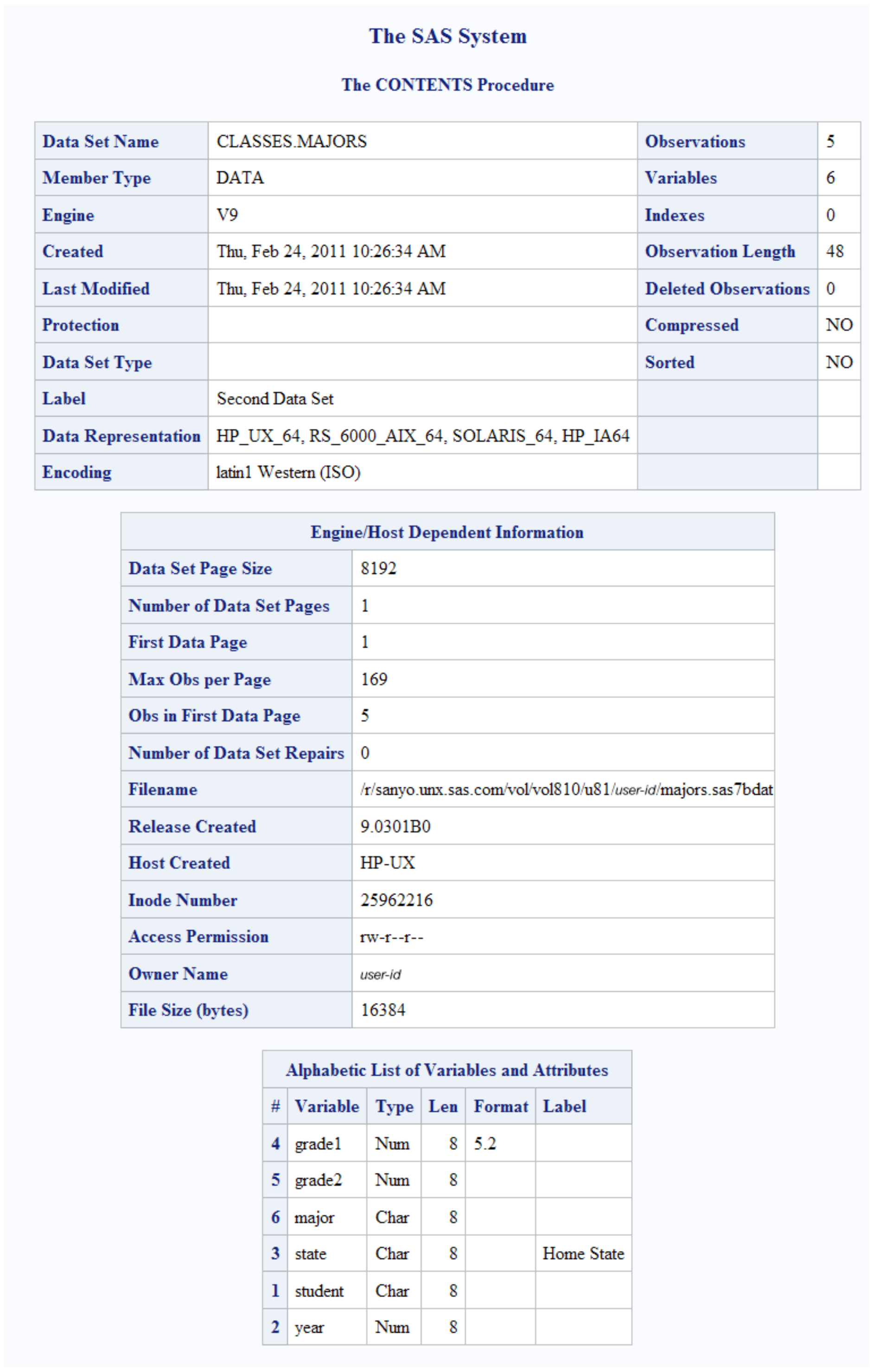
The CONTENTS procedure produces the same information as the CONTENTS statement in the DATASETS procedure. (See "DATASETS Procedure: UNIX" on page 302 for a comparison.)

Example: Executing PROC CONTENTS

The following SAS code creates two data sets, **`classes.grades`** and **`classes.majors`**, and executes PROC CONTENTS to describe the **`classes.majors`** data set:

```
libname classes '.';

data classes.grades (label='First Data Set');
   input student year state $ grade1 grade2;
   label year='Year of Birth';
   format grade1 4.1;
   datalines;
1000 1980 NC 85 87
1042 1981 MD 92 92
1095 1979 PA 78 72
1187 1980 MA 87 94
;

data classes.majors(label='Second Data Set');
   input student $ year state $ grade1 grade2 major $;
   label state='Home State';
   format grade1 5.2;
   datalines;
1000 1980 NC 84 87 Math
1042 1981 MD 92 92 History
1095 1979 PA 79 73 Physics
1187 1980 MA 87 74 Dance
1204 1981 NC 82 96 French
;

proc contents data=classes.majors;
run;
```

Display 16.1 *Output from the CONTENTS Procedure*



The SAS System

The CONTENTS Procedure

Data Set Name	CLASSES.MAJORS	**Observations**	5
Member Type	DATA	**Variables**	6
Engine	V9	**Indexes**	0
Created	Thu, Feb 24, 2011 10:26:34 AM	**Observation Length**	48
Last Modified	Thu, Feb 24, 2011 10:26:34 AM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label	Second Data Set		
Data Representation	HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64		
Encoding	latin1 Western (ISO)		

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	169
Obs in First Data Page	5
Number of Data Set Repairs	0
Filename	/r/sanyo.unx.sas.com/vol/vol810/u81/*user-id*/majors.sas7bdat
Release Created	9.0301B0
Host Created	HP-UX
Inode Number	25962216
Access Permission	rw-r--r--
Owner Name	*user-id*
File Size (bytes)	16384

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
4	grade1	Num	8	5.2	
5	grade2	Num	8		
6	major	Char	8		
3	state	Char	8		Home State
1	student	Char	8		
2	year	Num	8		

CONVERT Procedure: UNIX

Converts BMDP and OSIRIS system files and SPSS export files to SAS data sets.

UNIX specifics: all

Syntax

PROC CONVERT *product-specification* <*option-list*> ;

Required Argument

product-specification

Product-specification can be one of the following:

BMDP=*fileref* <(CODE=*code* CONTENT=*content-type*)>

converts the first member of a BMDP save file created under UNIX (AIX) into a SAS data set. Here is an example:

```
filename save '/usr/mydir/bmdp.dat';
proc convert bmdp=save;
run;
```

If you have more than one save file in the BMDP file referenced by the *fileref* argument, you can use two options in parentheses after *fileref*. The CODE= option specifies the code of the save file that you want, and the CONTENT= option specifies the content of the save file. For example, if a file with **`code=judges`** has a content of DATA, you can use the following statements:

```
filename save '/usr/mydir/bmdp.dat';
proc convert bmdp=save(code=judges
                       content=data);
run;
```

OSIRIS=*fileref|libref*

specifies a fileref or libref for the OSIRIS file to be converted into a SAS data set. You must also include the DICT= option.

SPSS=*fileref|libref*

specifies a fileref or libref for the SPSS export file that is to be converted into a SAS data set. The SPSS file must be created by using the SPSS EXPORT command, but it can be from any operating environment.

Optional Argument

option-list

Option-list can be one of the following:

DICT=*fileref|libref*

specifies a fileref or libref of the dictionary file for the OSIRIS file. DICT= is valid only when used with the OSIRIS product specification.

FIRSTOBS=*n*

gives the number of the observation where the conversion is to begin, so that you can skip observations at the beginning of the BMDP, OSIRIS, or SPSS file.

OBS=*n*

specifies the number of the last observation to be converted. This option enables you to exclude observations at the end of the file.

OUT=*SAS-data-set*

names the SAS data set that will hold the converted data. If OUT= is omitted, SAS still creates a Work data set and automatically names it DATA*n*, just as if you had omitted a data set name in a DATA statement. For more information, see

For more information, see “Introduction to SAS Files, Libraries, and Engines in UNIX Environments” on page 35.

Details

Converting System Files

The CONVERT procedure converts BMDP and OSIRIS system files, and SPSS export files to SAS data sets. The procedure is supplied for compatibility. The procedure invokes the appropriate engine to convert files.

PROC CONVERT produces one output data set, but no printed output. The new data set contains the same information as the input system file. Exceptions are noted in “How Missing Values Are Handled” on page 299.

The procedure converts system files from these products:

- BMDP saves files up to and including the most recent release of BMDP (available for AIX, HP-UX, and Solaris only).
- OSIRIS saves files through and including OSIRIS IV. (Hierarchical file structures are not supported.)

Because the BMDP, OSIRIS, and SPSS products are maintained by other organizations, changes might be made that make new files incompatible with the current version of PROC CONVERT. SAS upgrades PROC CONVERT to support changes to these products only when a new version of SAS is released.

How Missing Values Are Handled

If a numeric value in the output data set has no value or has a system missing value, PROC CONVERT assigns it a missing value.

How Variable Names Are Assigned

The following sections explain how names are assigned to the SAS variables created by the CONVERT procedure.

CAUTION:

Make sure that the translated names will be unique. Variable names are translated as indicated in the following sections.

Variable Names in BMDP Output

Variable names from the BMDP save file are used in the SAS data set, but nontrailing blanks and all special characters are converted to underscores in the SAS variable names. The subscript in BMDP variable names, such as x(1), becomes part of the SAS variable name with the parentheses omitted: X1. Alphabetic BMDP variables become SAS character variables of corresponding length. Category records from BMDP are not accepted.

Variable Names in OSIRIS Output

For single-response variables, the V1 through V9999 name becomes the SAS variable name. For multiple-response variables, the suffix R*n* is added to the variable name where *n* is the response. For example, V25R1 would be the first response of the multiple-response V25. If the variable after V1000 has 100 or more responses, responses above 99 are eliminated. Numeric variables that OSIRIS stores in character, fixed-point binary, or floating-point binary mode become SAS numeric variables. Alphabetic variables become SAS character variables; any alphabetic variable of length greater than 200 is truncated to 200. The OSIRIS variable description becomes a SAS variable label, and OSIRIS print format information becomes a SAS format.

Variable Names in SPSS Output

SPSS variable names and variable labels become variable names and labels without change. SPSS alphabetic variables become SAS character variables of the same length. SPSS blank values are converted to SAS missing values. SPSS print formats become SAS formats, and the SPSS default precision of no decimal places becomes part of the variables' formats. The SPSS DOCUMENT data is copied so that the CONTENTS procedure can display it. SPSS value labels are not copied.

Comparison with Interface Library Engines

The CONVERT procedure is closely related to the interface library engines BMDP, OSIRIS, and SPSS. (In fact, the CONVERT procedure uses these engines.) For example, the following two sections of code provide identical results:

```
filename myfile 'mybmdp.dat';
proc convert bmdp=myfile out=temp;
run;

libname myfile bmdp 'mybmdp.dat';
data temp;
   set myfile._first_;
run;
```

However, the BMDP, OSIRIS, and SPSS engines provide more extensive capability than PROC CONVERT. For example, PROC CONVERT converts only the first BMDP member in a save file. The BMDP engine, in conjunction with the COPY procedure, copies all members.

Examples

Example 1: Converting a BMDP Save File

The following statements convert a BMDP save file and produce the temporary SAS data set **temp**, which contains the converted data:

```
filename bmdpfile 'bmdp.savefile';
proc convert bmdp=bmdpfile out=temp;
run;
```

Example 2: Converting an OSIRIS File

The following statements convert an OSIRIS file and produce the temporary SAS data set **temp**, which contains the converted data:

```
filename osirfile 'osirdata';
filename dictfile 'osirdict';
proc convert osiris=osirfile dict=dictfile
             out=temp;
run;
```

Example 3: Converting an SPSS File

The following statements convert an SPSS file and produce the temporary SAS data set **temp**, which contains the converted data:

```
filename spssfile 'spssfile.num1';
proc convert spss=spssfile out=temp;
run;
```

See Also

"Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments" on page 63

CPORT Procedure: UNIX

Writes SAS data sets and catalogs into a transport file.

UNIX specifics:	name and location of transport file
See:	Chapter 15, "CPORT Procedure" in *Base SAS Procedures Guide*

Syntax

PROC CPORT *source-type=libref* | *<libref.> member-name <option(s)>* ;

Required Arguments

source-type
identifies the files to export as either a single SAS data set, single SAS catalog, or multiple members of a SAS library.

libref* | *<libref.> member-name
specifies the name of the SAS data set, catalog, or library to be exported.

Details

Note: This version is a simplified version of the CPORT procedure syntax. For the complete syntax and its explanation, see Chapter 15, "CPORT Procedure" in *Base SAS Procedures Guide*.

Starting in SAS 9.1, you can use the MIGRATE procedure to migrate a SAS library from a previous release. For more information, see the "Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments" on page 47, and the **`Technical Support Web site`**.

The CPORT procedure creates a transport file to later be restored (*imported*) by the CIMPORT procedure. The transport file can contain a SAS data set, SAS catalog, or an entire SAS library.

Typically, the FILE= option is used to specify the path of the transport file. The value of the FILE= option can be a fileref defined in a FILENAME statement or an environment variable. If this option is omitted, CPORT creates the default file Sascat.dat in the current directory as the transport file.

Example: Exporting Files

In this example, a SAS library called **`oldlib`** contains multiple SAS data sets and is being exported to the file called **`transport-file`**:

```
libname oldlib 'SAS-data-library';
filename tranfile 'transport-file';
```

```
proc cport lib=oldlib file=tranfile;
run;
```

This transport file is then typically moved by binary transfer to a different host, where the CIMPORT procedure will be used to restore the SAS library.

See Also

Procedures:

- "CIMPORT Procedure: UNIX" on page 294

Other References:

- "Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments" on page 47
- Chapter 1, "Moving and Accessing SAS Files between Operating Environments," in *Moving and Accessing SAS Files*

DATASETS Procedure: UNIX

Manages SAS files, and creates and deletes indexes and integrity constraints for SAS data sets.

UNIX specifics: Directory information, CONTENTS statement output

See: Chapter 16, "DATASETS Procedure" in *Base SAS Procedures Guide*

Syntax

PROC DATASETS<*option(s)*> ;

  CONTENTS <*option(s)*;>

Optional Argument

CONTENTS *options*
the value for *options* can be the following:

DIRECTORY
prints a list of information specific to the UNIX operating environment.

Details

Note: This version is a simplified version of the DATASETS procedure syntax. For the complete syntax and its explanation, see Chapter 16, "DATASETS Procedure" in *Base SAS Procedures Guide*.

The output from the DATASETS procedure shows you the libref, engine, and physical name that are associated with the library, as well as the names and other properties of the SAS files that are contained in the library. Some of the SAS library information, such as the filenames and access permissions, that is displayed in the SAS log by the DATASETS procedure depends on the operating environment and the engine. The information generated by the CONTENTS statement also varies according to the device type or access method associated with the data set.

If you specify the DIRECTORY option in the CONTENTS statement, the directory information is displayed in both the Log and Output windows.

The CONTENTS statement in the DATASETS procedure generates the same engine and host-dependent information as the CONTENTS procedure.

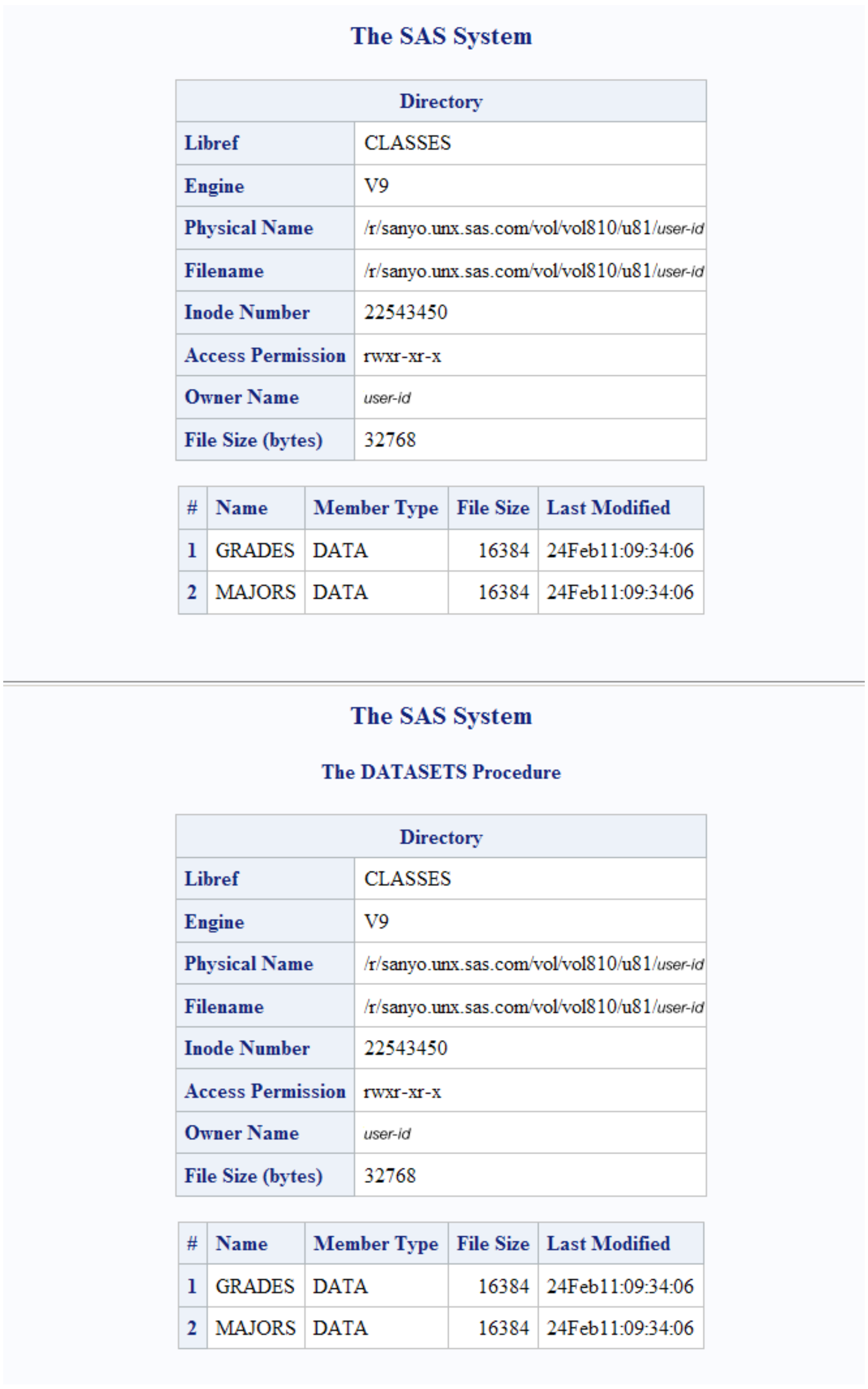
Example

The following SAS code creates two data sets, **`classes.grades`** and **`classes.majors`**, and executes PROC DATASETS using **`classes.majors`** as the input data set.

The first page of output from this example is produced by the DIRECTORY option in the CONTENTS statement. This information also appears in the SAS log. Page 2 in this output describes the data set **`classes.majors`** and appears only in the SAS output:

```
libname classes '.';

data classes.grades (label='First Data Set');
   input student year state $ grade1 grade2;
   label year='Year of Birth';
   format grade1 4.1;
   datalines;
1000 1980 NC 85 87
1042 1981 MD 92 92
1095 1979 PA 78 72
1187 1980 MA 87 94
;

data classes.majors(label='Second Data Set');
   input student $ year state $ grade1 grade2 major $;
   label state='Home State';
   format grade1 5.2;
   datalines;
1000 1980 NC 84 87 Math
1042 1981 MD 92 92 History
1095 1979 PA 79 73 Physics
1187 1980 MA 87 74 Dance
1204 1981 NC 82 96 French
;

proc datasets library=classes;
   contents data=majors directory;
run;
```

Display 16.2 *Output from the DATASETS Procedure*



The SAS System

Directory	
Libref	CLASSES
Engine	V9
Physical Name	/r/sanyo.unx.sas.com/vol/vol810/u81/*user-id*
Filename	/r/sanyo.unx.sas.com/vol/vol810/u81/*user-id*
Inode Number	22543450
Access Permission	rwxr-xr-x
Owner Name	*user-id*
File Size (bytes)	32768

#	Name	Member Type	File Size	Last Modified
1	GRADES	DATA	16384	24Feb11:09:34:06
2	MAJORS	DATA	16384	24Feb11:09:34:06

The SAS System

The DATASETS Procedure

Directory	
Libref	CLASSES
Engine	V9
Physical Name	/r/sanyo.unx.sas.com/vol/vol810/u81/*user-id*
Filename	/r/sanyo.unx.sas.com/vol/vol810/u81/*user-id*
Inode Number	22543450
Access Permission	rwxr-xr-x
Owner Name	*user-id*
File Size (bytes)	32768

#	Name	Member Type	File Size	Last Modified
1	GRADES	DATA	16384	24Feb11:09:34:06
2	MAJORS	DATA	16384	24Feb11:09:34:06

The SAS System

The DATASETS Procedure

Data Set Name	CLASSES.MAJORS	**Observations**	5
Member Type	DATA	**Variables**	6
Engine	V9	**Indexes**	0
Created	Thu, Feb 24, 2011 09:34:06 AM	**Observation Length**	48
Last Modified	Thu, Feb 24, 2011 09:34:06 AM	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label	Second Data Set		
Data Representation	HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64		
Encoding	latin1 Western (ISO)		

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	169
Obs in First Data Page	5
Number of Data Set Repairs	0
Filename	/r/sanyo.unx.sas.com/vol/vol810/u81/*user-id*/majors.sas7bdat
Release Created	9.0301B0
Host Created	HP-UX
Inode Number	25962217
Access Permission	rw-r--r--
Owner Name	*user-id*
File Size (bytes)	16384

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
4	grade1	Num	8	5.2	
5	grade2	Num	8		
6	major	Char	8		
3	state	Char	8		Home State
1	student	Char	8		
2	year	Num	8		

See Also

Procedures:

- Chapter 13, “CONTENTS Procedure” in *Base SAS Procedures Guide*

OPTIONS Procedure: UNIX

Lists the current settings of SAS system options.

UNIX specifics:	options available only under UNIX
See:	Chapter 32, “OPTIONS Procedure” in *Base SAS Procedures Guide*

Syntax

PROC OPTIONS <*option(s)*>;

Optional Argument

options

HOST | NOHOST
: displays only host options (HOST) or only portable options (NOHOST). PORTABLE is an alias for NOHOST.

Details

The Basics

Note:

This version is a simplified version of the OPTIONS procedure syntax. For the complete syntax and its explanation, see Chapter 32, “OPTIONS Procedure” in *Base SAS Procedures Guide*.

PROC OPTIONS lists the current settings of the system options that are available in all operating environments, as well as the system options that are available in the UNIX environment. If you specify the HOST option in the PROC OPTIONS statement, it lists those options that are available only under UNIX (host options). The option values that are displayed by PROC OPTIONS depend on the default values of SAS, the default values specified by your site administrator, the default values in your own configuration file, any changes made in your current session through the System Options window or OPTIONS statement, and possibly, the device on which you are running SAS.

For information about a specific option, see “SAS System Options under UNIX” on page 348.

Identifying Where an Option Was Set

You can specify the VALUE option in a PROC OPTIONS statement to identify the option’s scope and the method that was used to set the option. If an option’s value is set using multiple methods (for example, in an OPTIONS statement or in multiple configuration files), the VALUE option identifies which method determines the effective value. If the method is a configuration file, the VALUE option identifies the file path.

Use the following PROC OPTIONS statement to specify the VALUE option:

```
proc options option=option-name value;
```

For more information, see Chapter 32, “OPTIONS Procedure” in *Base SAS Procedures Guide*.

See Also

"Order of Precedence for Processing SAS Configuration Files" on page 23

PMENU Procedure: UNIX

Defines menu facilities for windows that are created with SAS software.

UNIX specifics:	ATTR= and COLOR= options in the TEXT statement have no effect; ACCELERATE= and MNEMONIC= options in the ITEM statement are ignored
See:	Chapter 36, "PMENU Procedure" in *Base SAS Procedures Guide*

Syntax

PROC PMENU <CATALOG=<*libref.*>*catalog*> <DESC '*entry-description*'> ;

Optional Arguments

CATALOG=<*libref.*>*catalog*
specifies the catalog in which you want to store PMENU entries. If you omit *libref*, the PMENU entries are stored in a catalog in the Sasuser library. If you omit CATALOG=, the entries are stored in the Sasuser.Profile catalog.

DESC '*entry-description*'
provides a description of the PMENU catalog entries created in the step.

Details

Note: This version is a simplified version of the PMENU procedure syntax. For the complete syntax and its explanation, see Chapter 36, "PMENU Procedure" in *Base SAS Procedures Guide*.

The PMENU procedure defines PMENU facilities for windows created by using the WINDOW statement in Base SAS software, the %WINDOW macro statement, the BUILD procedure of SAS/AF software, or the SAS Component Language (SCL) PMENU function with SAS/AF and SAS/FSP software.

Under UNIX, the following options are ignored:

- ATTR= and COLOR= options in the TEXT statement. The colors and attributes for text and input fields are controlled by the CPARMS colors specified in the SASCOLOR window. For more information, see "Customizing Colors in UNIX Environments" on page 196.
- ACCELERATE= and the MNEMONIC= options in the "TITLE Statement" in *SAS Statements: Reference*.

PRINTTO Procedure: UNIX

Defines destinations for SAS procedure output and the SAS log.

UNIX specifics:	Valid values of *file-specification*
See:	Chapter 38, "PRINTTO Procedure" in *Base SAS Procedures Guide*

Syntax

PROC PRINTTO <*options*>;

Optional Arguments

LOG=*file-specification*
specifies a fully qualified pathname (in quotation marks), an environment variable, a fileref, or a file in the current directory (without extension).

PRINT=*file-specification*
specifies a fully qualified pathname (in quotation marks), an environment variable, a fileref, or a file in the current directory (without extension). If you specify a fileref that is defined with the PRINTER device-type keyword, output is sent directly to the printer.

Details

Note: This version is a simplified version of the PRINTTO procedure syntax. For the complete syntax and its explanation, see Chapter 38, "PRINTTO Procedure" in *Base SAS Procedures Guide*.

The following statements send any SAS log entries that are generated after the RUN statement to the external file that is associated with the fileref **myfile**:

```
filename myfile '/users/myid/mydir/mylog';
proc printto log=myfile;
run;
```

If **myfile** has not been defined as a fileref, then PROC PRINTTO creates the file **myfile.log** in the current directory.

The following statements send any procedure output that is generated after the RUN statement to the file **/users/myid/mydir/myout**:

```
proc printto print='/users/myid/mydir/myout';
run;
```

The following statements send the procedure output from the CONTENTS procedure directly to the system printer:

```
filename myfile printer;
proc printto print=myfile;
run;
proc contents data=oranges;
run;
```

To redirect the SAS log and procedure output to their original default destinations, run PROC PRINTTO without any options:

```
proc printto;
run;
```

If filerefs **myprint** and **mylog** have not been defined, then the following statements send any SAS procedure output to **myprint.lst** and any log output to **mylog.log** in the current directory:

```
proc printto print=myprint log=mylog;
run;
```

If filerefs **myprint** and **mylog** had been defined, the output would have gone to the files associated with these filerefs.

See Also

"Overview of Printing Output in UNIX Environments" on page 92

SORT Procedure: UNIX

Sorts observations in a SAS data set by one or more variables, and then stores the resulting sorted observations in a new SAS data set or replaces the original data set.

UNIX specifics: sort utilities available

See: Chapter 50, "SORT Procedure" in *Base SAS Procedures Guide*

Syntax

PROC SORT<*options*> <*collating-sequence-option*>;

Optional Argument

option

SORTSIZE=*memory-specification*
: specifies the maximum amount of memory available to the SORT procedure. For more information about the SORTSIZE= option, see "SORTSIZE= Option" on page 310.

TAGSORT
: stores only the BY variables and the observation numbers in temporary files. The TAGSORT option has no effect on a UNIX host that uses SyncSort.

  For more information about the TAGSORT option, see "TAGSORT Option" on page 311.

DETAILS
: specifies that PROC SORT write messages to the SAS log detailing whether the sort was performed in memory. (This option is a statement option.)

  If the sort was not performed in memory, then the details that are written to the SAS log include the number of utility files that were used and their sizes.

  Tip: Using the DETAILS option can help determine an ideal SORTSIZE value.

Details

The Basics

Note: This version is a simplified version of the SORT procedure syntax. For the complete syntax and its explanation, see Chapter 50, "SORT Procedure" in *Base SAS Procedures Guide*.

The SORT procedure sorts observations in a SAS data set by one or more character or numeric variables, either replacing the original data set or creating a new, sorted data set. By default under UNIX, the SORT procedure uses the ASCII collating sequence.

The SORT procedure uses the sort utility that is specified by the SORTPGM system option. Sorting can be done by SAS or by the `syncsort` utility. You can use all of the options that are available to the SAS sort utility, such as the SORTSEQ and NODUPKEY options. In some situations, you can improve your performance by using the NOEQUALS option. If you specify an option that is not supported by the host sort, then the SAS sort will be used instead. For more information about all of the options that are available, see Chapter 50, “SORT Procedure” in *Base SAS Procedures Guide*.

SORTSIZE= Option

Limiting the Amount of Memory Available to PROC SORT

You can use the SORTSIZE= system option in the PROC SORT statement to limit the amount of memory that is available to the SORT procedure. This option can reduce the amount of swapping SAS must do to sort the data set.

Note: If you do not specify the SORTSIZE= option, PROC SORT uses the value of the SORTSIZE system option. The SORTSIZE system option can be defined in the command line or in the SAS configuration file.

Syntax of the SORTSIZE= Option

The syntax of the SORTSIZE= system option is as follows:

SORTSIZE=*memory-specification*

memory-specification can be one of the following:

n	specifies the amount of memory in bytes.
*n*K	specifies the amount of memory in 1-kilobyte multiples.
*n*M	specifies the amount of memory in 1-megabyte multiples.
*n*G	specifies the amount of memory in 1-gigabyte multiples.

Default Value of the SORTSIZE= Option

The default SAS configuration file sets this option based on the value of the SORTSIZE system option. To view the default value for your operating environment, execute the following code:

```
proc options option=sortsize;
run;
```

You can override the default value of the SORTSIZE system option in one of the following ways:

- by specifying a different SORTSIZE= value in the PROC SORT statement
- by submitting an OPTIONS statement that sets the SORTSIZE system option to a new value
- by setting the SORTSIZE system option in the command line during the invocation of SAS

Improving Performance with the SORTSIZE= Option

The SORTSIZE system option limits the amount of memory that is available to PROC SORT. In general, you should set the SORTSIZE= option to no larger than the amount of memory that is available to the SAS process through the MEMSIZE option.

When the SORTSIZE= value is large enough to fit the entire data set in memory, you can achieve optimal sort performance provided that your computer system has the same SORTSIZE= value of physical RAM free. If you do not have enough of physical RAM, then your computer will start swapping the extra memory pages to disk and negate the performance gains of using memory.

If the entire data set to be sorted will not fit in the space that is allocated by SORTSIZE, SAS creates a temporary utility file to store the data. In this case, SAS uses a sort algorithm that is tuned to sort using disk space instead of memory. These temporary utility files are placed in the SAS WORK location, but these files can be pointed to a different file system so that I/O is not impeded when you use the UTILLOC system option.

A SORTSIZE value of up to 512M is generally the optimal value to use if your data set can be sorted in memory. You should never set SORTSIZE to a value that is greater than 512M unless you can fit the file in physical RAM for sorting. SORTSIZE should always be set to a value that is at least 8M smaller than MEMSIZE.

Note: You can also use the SORTSIZE system option, which has the same effect as the SORTSIZE= option in the PROC SORT statement.

TAGSORT Option

The TAGSORT option in the PROC SORT statement is useful when there might not be enough disk space to sort a large SAS data set. When you specify the TAGSORT option, only the sort keys (that is, the variables specified in the BY statement) and the observation number for each observation are stored in the temporary utility files. The sort keys, together with the observation number, are referred to as tags. At the completion of the sorting process, the tags are used to retrieve the records from the input data set in sorted order. Thus, in cases where the total number of bytes of the sort keys is small compared with the length of the record, temporary disk use is reduced considerably.

You must have enough disk space to hold an additional copy of the data set (the output data set) and the utility file that contains the tags. By default, this utility file is stored in the Work library. If this directory is too small, you can change this directory by using the WORK system option. For more information, see "WORK System Option: UNIX" on page 435.

Note: Note that while using the TAGSORT option might reduce temporary disk use, the processing time could be higher. However, on systems with limited available disk space, the TAGSORT option might enable data sets to be sorted in situations where that would otherwise not be possible.

Disk Space Considerations for PROC SORT

You need to consider the following information when determining the amount of disk space needed to run PROC SORT:

input SAS data set
: PROC SORT uses the SAS input data set specified by the DATA= option.

output SAS data set
: PROC SORT stores the output SAS data set in the location that is specified by the OUT= option. If you use the SAS single-threaded sort, and the OUT= option is not specified, PROC SORT stores the output SAS data set in the Work library.

utility file
: UTILLOC affects the storage location of the utility file only when the SAS multi-threaded sort is used. The SAS single-threaded sort still stores its utility file in the Work directory. If the UTILLOC system option is not set, the utility file is stored in the Work directory. The utility file is the size of the uncompressed input SAS data file, and includes additional sortkey data that is appended to the front of each record.

  Note: You can use the UTILLOC system option to specify a location in which applications can store utility files.

temporary output SAS data set

During the sort, PROC SORT creates its output in the directory specified in the OUT= option (or directory of the input SAS data set if the OUT= option is not specified). The temporary data set has the same filename as the original data set, except it has an extension of .lck. After the sort completes successfully, the original data set is deleted, and the temporary data set is renamed to match the original data set. Therefore, you need to have enough available disk space in the target directory to hold two copies of the data set.

You can reduce the amount of disk space that is needed by specifying the OVERWRITE option in the PROC SORT statement. When OVERWRITE is specified, SORT, if possible, deletes the input data set before it attempts to write the replacement output data set. Deleting the input data set first can free storage space. This option should be used only with a data set that is backed up, or with a data set that you can reconstruct. For more information, see Chapter 50, “SORT Procedure” in *Base SAS Procedures Guide*.

Performance Tuning for PROC SORT

How SAS Determines the Amount of Memory to Use

The MEMSIZE system option limits the amount of memory that is available to the SAS process. The SORTSIZE system option limits the amount of memory that is available to PROC SORT. The REALMEMSIZE system option specifies the amount of real (not virtual) memory that will be made available to SAS.

While memory settings below the default values for MEMSIZE and SORTSIZE might adversely affect sorting and SAS performance, making large amounts of memory available might be of no benefit. The key for determining whether additional memory might improve performance is whether the sort will fit in memory. If the sorted file requires more memory than is allocated, then a SORTSIZE value in the range of 64–512M is generally the optimal value. SORTSIZE should always be set to a value that is at least 8M smaller than MEMSIZE.

For information about setting the REALMEMSIZE system option, see “REALMEMSIZE System Option: UNIX” on page 410.

Note: If you receive an out of memory error, then increase the value of MEMSIZE. For more information, see “MEMSIZE System Option: UNIX” on page 399.

Guidelines for Setting the REALMEMSIZE System Option

You can use the REALMEMSIZE system option with PROC SORT to determine how much memory to use. It is important that the REALMEMSIZE value reflects the amount of memory that is available on your system. For optimal performance, the maximum value for the memory setting for all of your applications (including file cache), should never exceed the amount of physical RAM on your computer. The default value for REALMEMSIZE is 80% of the MEMSIZE setting. If REALMEMSIZE is set too high, then PROC SORT might use more memory than is actually available. Using too much memory will cause excessive paging and adversely impact system performance.

In general, REALMEMSIZE should be set to the amount of physical memory (not including swap space) that you expect to be available to SAS at run time. A good starting value is the amount of physical memory installed on the computer less the amount that is being used by running applications and the operating system. You can experiment with the REALMEMSIZE value until you reach optimum performance for your environment. In some cases, optimum performance can be achieved with a very low REALMEMSIZE value. A low value could cause SAS to use less memory and leave more memory for the operating system to perform I/O caching.

For more information, see “REALMEMSIZE System Option: UNIX” on page 410.

Using Other Options That Affect Performance

The THREADS system option controls whether threaded procedures will use threads. It is available as both a system option and as a procedural override in PROC SORT.

The CPUCOUNT option is directly related to the THREADS option and defaults to the number of CPUs on your computer. Depending on your file system and the number of concurrent users, you might benefit from lowering the CPUCOUNT on machines that have many CPUs. When the value of CPUCOUNT equals ACTUAL, SAS returns the number of physical CPUs that are associated with the operating environment where SAS is executing.

The UTILLOC system option allows for the spreading of utility files, and is a good option for balancing I/O.

The DETAILS option, specified in the PROC SORT statement, causes PROC SORT to write messages to the SAS log detailing whether the sort was performed in memory. If the sort was not performed in memory, then the details that are written include the number of utility files and their sizes.

For more information about the THREADS, CPUCOUNT, and UTILLOC system options see *SAS System Options: Reference*.

Creating Your Own Collating Sequences

If you want to provide your own collating sequences or change a collating sequence provided for you, use the TRANTAB procedure to create or modify translation tables. For more information, see Chapter 17, "TRANTAB Procedure" in *SAS National Language Support (NLS): Reference Guide*. When you create your own translation tables, they are stored in your Sasuser.Profile catalog, and they override any translation tables by the same name that are stored in the Host catalog.

Note: System managers can modify the Host catalog by copying newly created tables from the Profile catalog to the Host catalog. Then, all users can access the new or modified translation table.

If you are using the SAS windowing environment and want to see the names of the collating sequences that are stored in the Host catalog, issue the following command from any window:

```
catalog sashelp.host
```

If you are not using the SAS windowing environment, then issue the following statements to generate a list of the contents in the Host catalog:

```
proc catalog catalog=sashelp.host;
contents;
run;
```

Entries of type TRANTAB are the collating sequences.

To see the contents of a particular translation table, use the following statements:

```
proc trantab table=table-name;
list;
run;
```

The contents of collating sequences are displayed in the SAS log.

Specifying the Host Sort Utility

Introduction to Using the Host Sort

SAS supports one host sort utility on UNIX called **`syncsort`**. You can use this sorting application as an alternative sorting algorithm to the SAS sort. SAS determines which sort to use by the values that are set for the SORTNAME, SORTPGM, SORTCUT, and SORTCUTP system options.

Setting the Host Sort Utility as the Sort Algorithm

To specify a host sort utility as the sort algorithm, complete the following steps:

1. Specify the name of the host utility (**`syncsort`**) in the SORTNAME system option.
2. Set the SORTPGM system option to tell SAS when to use the host sort utility.
   - If you specify SORTPGM=HOST, then SAS always prefers to use the host sort utility.
   - If you specify SORTPGM=BEST, then SAS chooses the best sorting method (either the SAS sort or the host sort) for the situation.

Sorting Based on Size or Observations

The sort routine that SAS uses can be based on either the number of observations in a data set, or on the size of the data set. When the SORTPGM system option is set to BEST, SAS uses the first available and pertinent sorting algorithm based on the following order of precedence:

- host sort utility
- SAS sort utility

The SORTCUT system option is based on the number of observations in a data set. The SORTCUTP system option is based on the size of the data set. SAS looks at the values for the SORTCUT and SORTCUTP system options to determine which sort routine to use. If the number of observations is greater than or equal to the value of SORTCUT, SAS uses the host sort utility. If the number of bytes in a data set is greater than the value of SORTCUTP, SAS uses the host sort utility.

If SORTCUT and SORTCUTP are set to zero, SAS uses the SAS sort utility. If you specify both system options, and either condition is met, SAS uses the host sort utility.

When the following OPTIONS statement is in effect, the host sort utility (**`syncsort`**) is used when the number of observations is 500 or greater:

```
options sortpgm=best sortcut=500;
```

In this example, the host sort utility is used when the size of the data set is greater than 40M:

```
options sortpgm=best sortcutp=40M;
```

For more information about these sort options, see “SORTCUT System Option: UNIX” on page 420, “SORTCUTP System Option: UNIX” on page 421, and “SORTPGM System Option: UNIX” on page 424.

Changing the Location of Temporary Files Used by the Host Sort Utility

By default, the host sort utilities use the location that is specified in the -WORK option for temporary files. To change the location of these temporary files, specify a location by using the SORTDEV system option. Here is an example:

```
options sortdev="/tmp/host";
```

For more information, see “SORTDEV System Option: UNIX” on page 422.

Passing Options to the Host Sort Utility

To specify options for the sort utility, use the SORTANOM system option. For a list of valid options, see “SORTANOM System Option: UNIX” on page 419.

Passing Parameters to the Host Sort Utility

To pass parameters to the sort utility, use the SORTPARM system option. The parameters that you can specify depend on the host sort utility. For more information, see "SORTPARM System Option: UNIX" on page 423.

Specifying the SORTSEQ= Option with a Host Sort Utility

The SORTSEQ= option enables you to specify the collating sequence for your sort. For a list of valid values, see Chapter 50, "SORT Procedure" in *Base SAS Procedures Guide*.

CAUTION:

> **If you are using a host sort utility to sort your data, then specifying the SORTSEQ= option might corrupt the character BY variables if the sort sequence translation table and its inverse are not one-to-one mappings.** In other words, for the sort to work, the translation table must map each character to a unique weight, and the inverse table must map each weight to a unique character.

If your translation tables do not map one-to-one, then you can use one of the following methods to perform your sort:

- Create a translation table that maps one-to-one. Once you create a translation table that maps one-to-one, you can easily create a corresponding inverse table using the TRANTAB procedure. If your translation table is not mapped one-to-one, then you will receive the following note in the SAS log when you try to create an inverse table:

  ```
  NOTE:  This table cannot be mapped one to one.
  ```

  For more information, see Chapter 17, "TRANTAB Procedure" in *SAS National Language Support (NLS): Reference Guide*.
- Use the SAS sort. You can specify the SAS sort using the SORTPGM system option. For more information, see "SORTPGM System Option: UNIX" on page 424.
- Specify the collation order options of your host sort utility. See the documentation for your host sort utility for more information.
- Create a view with a single BY variable. For an example, see "Example: Creating a View with a Single BY Variable" on page 315.

Note: After using one of these methods, you might need to perform subsequent BY processing using either the NOTSORTED option or the NOBYSORTED system option. For more information about the NOTSORTED option, see "BY Statement" in *SAS Statements: Reference*. For more information about the NOBYSORTED system option, see "BYSORTED System Option" in *SAS System Options: Reference*.

Example: Creating a View with a Single BY Variable

The following example shows how to create a view by using a single BY variable. SAS uses the BEST argument in the SORTPGM system option to sort the data. By using BEST, SAS selects either the host sort or the SAS sort. (Sorting can also be performed by a DBMS when you use a SAS/ACCESS engine.)

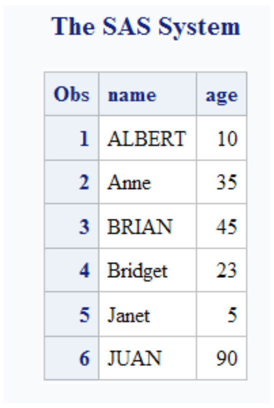
```
options sortpgm=best msglevel=i;

data one;
   input name $ age;
   datalines;
Anne 35
ALBERT 10
JUAN 90
```

```
Janet 5
Bridget 23
BRIAN 45
;

data oneview / view=oneview;
   set one;
   name1=upcase(name);
run;

proc sort data=oneview out=final(drop=name1);
   by name1;
run;

proc print data=final;
run;
```

Log 16.1 *Log Output*

```
NOTE: SAS threaded sort was used.
```

Display 16.3 *Output from Creating a View with a Single BY Variable*



The SAS System

Obs	name	age
1	ALBERT	10
2	Anne	35
3	BRIAN	45
4	Bridget	23
5	Janet	5
6	JUAN	90

See Also

Procedures:

- Chapter 17, "TRANTAB Procedure" in *SAS National Language Support (NLS): Reference Guide*

System Options:

- "MEMSIZE System Option: UNIX" on page 399
- "REALMEMSIZE System Option: UNIX" on page 410
- "SORTANOM System Option: UNIX" on page 419
- "SORTCUT System Option: UNIX" on page 420
- "SORTCUTP System Option: UNIX" on page 421
- "SORTDEV System Option: UNIX" on page 422
- "SORTNAME System Option: UNIX" on page 422

- “SORTPARM System Option: UNIX” on page 423
- “SORTPGM System Option: UNIX” on page 424
- “SORTSIZE System Option: UNIX” on page 424
- “UTILLOC= System Option” in *SAS System Options: Reference*

Chapter 17
Statements under UNIX

SAS Statements under UNIX

This section describes SAS statements that exhibit behavior or syntax that is specific to UNIX environments. Each statement description includes a brief “UNIX specifics” section that explains which aspect of the statement is specific to UNIX. If the information under “UNIX specifics” says “all”, then the statement is described only in this documentation. Otherwise, the statement is described in this documentation and in *SAS Statements: Reference*.

Dictionary

ABORT Statement: UNIX

Stops executing the current DATA step, SAS job, or SAS session.

Valid in: in a DATA step

UNIX specifics: values of *n*

See: "ABORT Statement" in *SAS Statements: Reference*

Syntax

ABORT <ABEND | RETURN> <*n*>;

Details

The *n* option enables you to specify the value of the exit status code that SAS returns to the shell when it stops executing. The value of *n* can range from 0 to 255. Normally, a return code of 0 is used to indicate that the program ran with no errors, and return codes greater than 0 are used to indicate progressively more serious error conditions. Return codes of 0–6 and those codes that are greater than 977 are reserved for use by SAS.

See Also

"Determining the Completion Status of a SAS Job in UNIX Environments" on page 26

ATTRIB Statement: UNIX

Associates a format, informat, label, or length with one or more variables.

Valid in: in a DATA step

UNIX specifics: length specification

See: "ATTRIB Statement" in *SAS Statements: Reference*

Syntax

ATTRIB *variable-list*-1 *attribute-list*-1 <*...variable-list-n attribute-list-n*> ;

Required Argument

attribute-list

LENGTH=<$>*length*
specifies the length of the variables in *variable-list*. The minimum length that you can specify for a numeric variable depends on the floating-point format used by your system. Because most systems use the IEEE floating-point format, the minimum is 3 bytes.

Details

Note: The explanation of the ATTRIB statement syntax is a simplified form. For complete syntax and description, see the "ATTRIB Statement" in *SAS Statements: Reference*.

See Also

"Numeric Variable Length and Precision in UNIX Environments" on page 217

FILE Statement: UNIX

Specifies the current output file for PUT statements.

Valid in:	in a DATA step
UNIX specifics:	valid values for *file-specification*, *host-options*, and *encoding-value*
See:	"FILE Statement" in *SAS Statements: Reference*

Syntax

FILE *file-specification* <ENCODING='*encoding-value*' > <*options*> <*host-options*> ;

Required Argument

file-specification
can be any of the file specification forms that are discussed in "Accessing an External File or Device in UNIX Environments" on page 71.

Optional Arguments

ENCODING='*encoding-value*'
specifies the encoding to use when writing to the output file. The value for ENCODING= indicates that the output file has a different encoding from the current session encoding.

When you write data to the output file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Overview to SAS Language Elements That Use Encoding Values" in Chapter 20 of *SAS National Language Support (NLS): Reference Guide*.

options
can be any of the options for the FILE statement that are valid in all operating environments. For a description of these options, see the *SAS Statements: Reference*.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically written in one I/O operation. The default is 8K. The maximum is 1G–1.

TERMSTR=
controls the end-of-line or record delimiters in files formatted by UNIX or the PC. This option enables the sharing of these formatted files between the two hosts. The following are values for the TERMSTR= option:

CRLF	Carriage return line feed. Use TERMSTR=CRLF to read files formatted by a PC.
LF	Line feed. This parameter is used to read files formatted by UNIX. LF is the default.

Use TERMSTR=CRLF to read a file that was created on a PC. If the PC-formatted file was created using TERMSTR=LF, then the TERMSTR= option is unnecessary.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the length of each output record. The output record is truncated or padded with blanks to fit the specified size.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are divided into multiple records.

MOD
indicates that data written to the file should be appended to the file.

NEW | OLD
specifies whether a new file or an existing file will be used for output. If you specify NEW, a new file is to be opened for output. If the file already exists, it is deleted and re-created. If you specify OLD, the previous contents of the file are replaced. NEW is the default.

RECFM=
specifies the record format. Values for the RECFM= option are the following:

D	default format (same as variable).
F	fixed format. That is, each record has the same length. Do not use RECFM=F for external files that contain carriage-control characters.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format. SAS writes carriage-control characters.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS).

UNBUF
tells SAS not to perform buffered Writes to the file on any subsequent FILE statement. This option applies especially when you are writing to a data collection device.

Details

The ENCODING= option is valid only when the FILE statement includes a file specification that is not a reserved fileref. If the FILE statement includes the ENCODING= argument and the reserved filerefs Log or Print as the *file-specification*, then SAS issues an error message. The ENCODING= value in the FILE statement overrides the value of the ENCODING= system option.

You can set the permissions of the output file by issuing the `umask` command from within the SAS session. For more information, see "Executing Operating System Commands from Your SAS Session" on page 15.

See Also

"Using External Files and Devices" on page 70

FILENAME Statement: UNIX

Associates a SAS fileref with an external file or output device; disassociates a fileref and external file; lists attributes of external files.

Valid in:	anywhere
UNIX specifics:	*device-type*, *external-file*, *host-options*, and *encoding-value*
See:	"FILENAME Statement" in *SAS Statements: Reference*

Syntax

FILENAME *fileref* <*device-type*> '*external-file*' <ENCODING='*encoding-value*'> <'*host-options*'> <LOCKINTERNAL= AUTO | SHARED> ;

FILENAME *fileref* *device-type* <'*external-file*'> <ENCODING='*encoding-value*'> <'*host-options*'> <LOCKINTERNAL= AUTO | SHARED> ;

FILENAME *fileref* ('*pathname-1*' ... '*pathname-n*') <ENCODING='*encoding-value*'> <'*host-options*'> <LOCKINTERNAL= AUTO | SHARED> ;

FILENAME *fileref* *directory-name* <ENCODING='*encoding-value*'> <LOCKINTERNAL= AUTO | SHARED> ;

FILENAME *fileref* <*access-method*> '*external-file*' *access-information*;

FILENAME *fileref* CLEAR | _ALL_ CLEAR;

FILENAME *fileref* LIST | _ALL_ LIST;

Required Arguments

fileref

is the name by which you reference the file. Under UNIX, the value of *fileref* can be an environment variable.

Note: You cannot clear a fileref that is defined by an environment variable. Filerefs that are defined by an environment variable are assigned for the entire SAS session.

For more information, see "Using Environment Variables to Assign Filerefs in UNIX Environments" on page 78.

'*external-file*'

differs according to device type. "Device Information in the FILENAME Statement" on page 327 shows the information appropriate to each device. Remember that UNIX filenames are case sensitive. For more information, see "Specifying Pathnames in UNIX Environments" on page 72.

Note: If a filename has leading blanks, then the blanks will be trimmed.

Optional Arguments

device-type

specifies a device for the output, such as a disk, terminal, printer, pipe, and so on. The device-type keyword must follow *fileref* and precede *pathname*. "Device Information in the FILENAME Statement" on page 327 describes the valid device types. DISK is the default device type. If you are associating the fileref with a DISK file, then you do not need to specify the device type.

ENCODING='*encoding-value*'

specifies the encoding to use when reading from or writing to the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding. When you write data to an external file, SAS transcodes the data from the session encoding to the specified encoding.

Note: The UPRINTER device type does not support the ENCODING option.

For valid encoding values, see "Overview to SAS Language Elements That Use Encoding Values" in Chapter 20 of *SAS National Language Support (NLS): Reference Guide*.

'*host-options*'

are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=

specifies the number of bytes that are physically written or read in one I/O operation. The default is 8K. The maximum is 1G–1. If you specify RECFM=S370VBS, then you should specify BLKSIZE=32760 in order to avoid errors with records longer than 255 characters.

TERMSTR=

controls the end of line and record delimiters in files formatted by UNIX or the PC. This option enables the sharing of these formatted files between the two hosts. The following values for the TERMSTR= option are valid:

CR	Carriage return. Use TERMSTR=CR to read files formatted by an Apple Macintosh.
CRLF	Carriage return line feed. Use TERMSTR=CRLF to read files formatted by a PC.
LF	Line feed. This parameter is used to read files formatted by UNIX. LF is the default.

If you are working on UNIX and reading a file that was created on a PC, specify TERMSTR=CRLF unless the file was created with the TERMSTR=LF option. If you are writing a file that will be read on a PC, specify TERMSTR=CRLF.

If you are working on a PC and reading a file that was created on UNIX, specify TERMSTR=LF unless the file was created with the TERMSTR=CRLF option.

If you are writing a file that will be read on UNIX, specify TERMSTR=LF.

LRECL=

specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines either the number of bytes to be read as one record or the length of each output record. The output record is truncated or padded with blanks to fit the specified size.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are divided into multiple records on output and truncated on input.
- If RECFM=S370VBS, then you should specify LRECL=32760 in order to avoid errors with records longer than 255 characters.

MOD
: indicates that data written to the file should be appended to the file.

NEW | OLD
: specifies whether a new or existing file will be used for output. If you specify NEW, a new file is to be opened for output. If the file already exists, it is deleted and re-created. If you specify OLD, the previous contents of the file are replaced. NEW is the default.

RECFM=
: specifies the record format. Values for the RECFM= option are the following:

D	default format (same as variable).
F	fixed format. That is, each record has the same length. Do not use RECFM=F for external files that contain carriage-control characters.
N	binary format. The file consists of a stream of bytes with no record boundaries. N is not valid for the PIPE device type. If you do not specify the LRECL option, then by default SAS reads 256 bytes at a time from the file.
P	print format. On output, SAS writes carriage-control characters.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS). If you specify RECFM=S3270VBS, then you should specify BLKSIZE=32760 and LRECL=32760 in order to avoid errors with records longer than 255 characters.

The RECFM= option is used for both input and output.

LOCKINTERNAL=AUTO | SHARED
: specifies the SAS system locking that is to be used for the files that are listed in a FILENAME statement. LOCKINTERNAL can have one of the following values:

AUTO
: locks a file so that in a SAS session, if a user has Write access to a file, then no other users can have Read or Write access to the file. If a user has Read access to a file, no other user can have Write access to the file, but multiple users can have Read access.

SHARED
: locks a file so that in a SAS session, two users do not have simultaneous Write access to the file. The file can be shared simultaneously by one user who has Write access and multiple users who have Read access.

Default: AUTO

UNBUF
: tells SAS not to perform buffered Writes to the file on any subsequent FILE statement. This option applies especially when you are reading from or writing to a data collection device. As explained in *SAS Statements: Reference*, it also prevents buffered Reads on INFILE statements.

'pathname-1'...'pathname-n'
: are pathnames for the files that you want to access with the same fileref. Use this form of the FILENAME statement when you want to concatenate filenames. Concatenation of filenames is available only for DISK files, so you do not have to specify the *device-type*. Separate the pathnames with either commas or blank spaces. Enclose each pathname in quotation marks. Table 2.6 on page 54 shows character substitutions that you can use when specifying a pathname. If the fileref that you are defining is to be used for input, then you can also use wildcards as described in “Using Wildcards in Pathnames (Input Only)” on page 73. Remember that UNIX filenames are case-sensitive.

directory-name
: specifies the directory that contains the files that you want to access. For more information, see “Assigning a Fileref to a Directory (Using Aggregate Syntax)” on page 77.

access-method
: can be CATALOG, SOCKET, FTP, SFTP, or URL. “Device Information in the FILENAME Statement” on page 327 describes the information expected by these access methods.

access-information
: differs according to the access method. “Device Information in the FILENAME Statement” on page 327 shows the information appropriate to each access method.

CLEAR
: clears the specified fileref or, if you specify _ALL_, clears all filerefs that are currently defined.

Note: You cannot clear a fileref that is defined by an environment variable. Filerefs that are defined by environment variables are assigned for the entire SAS session.

ALL
: refers to all filerefs currently defined. You can use this keyword when you are listing or clearing filerefs.

LIST
: writes to the SAS log the pathname of the specified fileref or, if you specify _ALL_, lists the definition for all filerefs that are currently defined. Filerefs defined as environment variables appear only if you have already used those filerefs in a SAS statement. If you are using the Bourne shell or the Korn shell, SAS cannot determine the name of a pre-opened file, so it displays the following string instead of a filename:

```
<File Descriptor number>
```

For more information, see “Using Environment Variables to Assign Filerefs in UNIX Environments” on page 78.

Details

File Locking

File locking of external files is controlled at the FILENAME statement level by the LOCKINTERNAL option. If you use the AUTO (default) value for LOCKINTERNAL, then SAS locks a file exclusively for one user who has Write access, or SAS locks a file non-exclusively for multiple users who have Read access. For example, if a file is opened in UPDATE or OUTPUT mode, then all other access from internal processes will be blocked. If a file is opened in INPUT mode, then multiple users can read the file, but UPDATE and OUTPUT functions are blocked.

If you use the SHARED value for LOCKINTERNAL, then SAS allows one user Write access to a file as well as allowing multiple users to read the file.

Device Information in the FILENAME Statement

The following table lists the relationship between device type or access method and the related external file.

Table 17.1 *Device Information in the FILENAME Statement*

Device or Access Method	Function	External File
CATALOG	references a SAS catalog as an external file.	is a valid two-, three-, or four-part SAS catalog name followed by catalog options (if needed). See *SAS Language Reference: Concepts* for more information.
DISK	associates the fileref with a DISK file.	is either the pathname for a single file or, if you are concatenating filenames, a list of pathnames separated by spaces or commas and enclosed in parentheses. The level of specification depends on your location in the file system. Table 2.6 on page 54 shows character substitutions that you can use when specifying a UNIX pathname.
DUMMY	associates a fileref with a null device.	None. DUMMY enables you to debug your application without reading from or writing to a device. Output to this device is discarded.
EMAIL	sends electronic mail to an address.	is an address and e-mail options. For more information, see "Sending Electronic Mail Using the FILENAME Statement (EMAIL)" on page 84.

Device or Access Method	Function	External File
FTP	reads from or writes to a file from any computer on a network that is running an FTP server.	is the pathname of the external file on the remote computer followed by FTP options. See *SAS Language Reference: Concepts*, and "Assigning Filerefs to Files on Other Systems (FTP, SFTP, and SOCKET Access Types)" on page 76 for more information. If you are transferring a file to UNIX from the z/OS operating environment and you want to use either the S370V or S370VB format to access that file, then the file must be of type RECFM=U and BLKSIZE=32760 before you transfer it. If you FTP to a z/OS computer, only one member of a z/OS PDS can be written to at a time. If you need to write to multiple members at the same time, a z/OS PDSE or a UNIX System Services directory should be used.
PIPE	reads input from or writes output to a UNIX command.	is a UNIX command. For more information, see Chapter 4, "Printing and Routing Output," on page 91.
PLOTTER	sends output to a plotter.	is a device name and plotter options. For more information, see "Printing the Contents of a Window" on page 98, and "Using the PRINTTO Procedure in UNIX Environments" on page 100.
PRINTER	sends output to a printer.	is a device name and printer option. For more information, see "Printing the Contents of a Window" on page 98, and "Using the PRINTTO Procedure in UNIX Environments" on page 100.
SFTP	reads from or writes to a file from any host computer that you can connect to on a network with an SSHD server running.	is the pathname of the external file on the remote computer, followed by SFTP options. For more information, see "FILENAME Statement, SFTP Access Method" in *SAS Statements: Reference*, and "Assigning Filerefs to Files on Other Systems (FTP, SFTP, and SOCKET Access Types)" on page 76.
SOCKET	reads and writes information over a TCP/IP socket.	depends on whether the SAS application is a server application or a client application. In a client application, *external-file* is the name or IP address of the host and the TCP/IP port number to connect to, followed by any TCP/IP options. In a server application, *external-file* is the port number to create for listening, followed by the SERVER keyword, and then any TCP/IP options. See *SAS Statements: Reference* for more information.
TEMP	associates a fileref with an external file stored in the Work library.	None
TERMINAL	associates a fileref with a terminal.	is the pathname of a terminal.

Device or Access Method	Function	External File
UPRINTER	sends output to the default printer that was set up through the Printer Setup dialog box.	None

URL	enables you to use the URL of a file to access it remotely.	is the name of the file that you want to read from or write to on a URL server. The URL must be in one of these forms:
		`http://`*`hostname`*`/`*`file`*
		`http://`*`hostname`*`:`*`portno`*`/`*`file`*

WebDAV	enables you to use WebDAV (Web Distributed Authoring and Versioning) to read from or write to a file from any host machine that you can connect to on a network with a WebDAV server running.	is the name of the file that you want to read from or write to a WebDAV server. The external file must be in one of these forms:
		`http://`*`hostname`*`/`*`path-to-the-file`*
		`https://`*`hostname`*`/`*`path-to-the-file`*
		`http://`*`hostname`*`:`*`port`*`/`*`path-to-the-file`*
		`https://`*`hostname`*`:`*`port`*`/`*`path-to-the-file`*

See Also

- "Using External Files and Devices" on page 70
- Chapter 4, "Printing and Routing Output," on page 91

FOOTNOTE Statement: UNIX

Writes up to 10 lines of text at the bottom of the procedure or DATA step output.

Valid in:	anywhere
UNIX specifics:	maximum length of footnote
See:	"FOOTNOTE Statement" in *SAS Statements: Reference*

Syntax

FOOTNOTE <*n*> <'*text*'|"*text*"> ;

Details

The maximum footnote length is 255 characters. If the length of the specified footnote is greater than the value of the LINESIZE option, SAS truncates the footnote to the line size.

%INCLUDE Statement: UNIX

Brings a SAS programming statement, data lines, or both into a current SAS program.

Valid in:	anywhere
UNIX specifics:	*source*, if a file specification is used; valid values for *encoding-value*
See:	"%INCLUDE Statement" in *SAS Statements: Reference*

Syntax

%INCLUDE *source-1* <...*source-n*> </<SOURCE2><S2=*length*> <ENCODING='*encoding-value*'><*host-options*>> ;

Required Argument

source

describes the location that you want to access with the %INCLUDE statement. The three possible sources are a file specification, internal lines, or keyboard entry. The file specification can be any of the file specification forms that are discussed in "Accessing an External File or Device in UNIX Environments" on page 71.

Note: When using aggregate syntax, if the member name contains a leading digit, enclose the member name in quotation marks. If the member name contains a macro variable reference, use double quotation marks.

Optional Arguments

ENCODING='*encoding-value*'

specifies the encoding to use when reading from the specified source. The value for ENCODING= indicates that the specified source has a different encoding from the current session encoding.

When you read data from the specified source, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see "Overview to SAS Language Elements That Use Encoding Values" in Chapter 20 of *SAS National Language Support (NLS): Reference Guide*.

host-options

consists of statement options that are valid under UNIX. The following options are available:

BLKSIZE=*block-size*
BLK=*block-size*

specifies the number of bytes that are physically read or written in an I/O operation. The default is 8K. The maximum is 1M.

LRECL=*record-length*

specifies the record length (in bytes). Under UNIX, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

RECFM=*record-format*

specifies the record format. The following values are valid under UNIX:

D	default format (same as variable).

F	fixed format. That is, each record has the same length.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS).

The S370 values are valid with files laid out as z/OS files only. That is, files are binary, have variable-length records, and are in EBCDIC format. If you want to use a fixed-format z/OS file, first copy it to a variable-length, binary z/OS file.

Details

If you specify any options in the %INCLUDE statement, remember to precede the options list with a forward slash (/).

See Also

"Introduction to External Files and Devices in UNIX Environments" on page 70

INFILE Statement: UNIX

Identifies an external file to read with an INPUT statement.

Valid in:	in a DATA step
UNIX specifics:	valid values for *encoding-value*, *file-specification*, and *host-options*
See:	"INFILE Statement" in *SAS Statements: Reference*

Syntax

INFILE *file-specification* <ENCODING='*encoding-value*'> <*options*> <*host-options*> ;

Required Argument

file-specification
can be any of the file specification forms that are discussed in the "Accessing an External File or Device in UNIX Environments" on page 71.

Optional Arguments

ENCODING='*encoding-value*'
specifies the encoding to use when reading from the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see “Overview to SAS Language Elements That Use Encoding Values” in Chapter 20 of *SAS National Language Support (NLS): Reference Guide*.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically read in one I/O operation. The default is 8K. The maximum is 1G–1.

TERMSTR=
controls the end of line or record delimiters in files formatted by UNIX or the PC. This option enables the sharing of these formatted files between the two hosts. The following are values for the TERMSTR= option:

CR	Carriage return. Use TERMSTR=CR to read files formatted by an Apple Macintosh.
CRLF	Carriage return line feed. Use CRLF to read files formatted by a PC.
LF	Line feed. This parameter is used to read files formatted by UNIX. LF is the default.

Use TERMSTR=CRLF to read a file that was created on the PC. If the PC-formatted file was created using TERMSTR=LF, then the TERMSTR= option is unnecessary.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the number of bytes to be read as one record.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are truncated.

RECFM=
specifies the record format. The following values are valid under UNIX:

D	default format (same as variable).
F	fixed format. That is, each record has the same length.
N	binary format. The file consists of a stream of bytes with no record boundaries. If you do not specify the LRECL option, then, by default, SAS reads 256 bytes at a time from the file.
P	print format.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS).

Details

The ENCODING= option is valid only when the INFILE statement includes a file specification that is not a reserved fileref. If the INFILE statement includes the ENCODING= argument and the reserved filerefs DATALINES or DATALINES4 as a *file-specification*, then SAS issues an error message. The ENCODING= value in the INFILE statement overrides the value of the ENCODING= system option.

See Also

"Using External Files and Devices" on page 70

LENGTH Statement: UNIX

Specifies the number of bytes for storing variables.

Valid in:	in a DATA step
UNIX specifics:	valid numeric variable lengths
See:	"LENGTH Statement" in *SAS Statements: Reference*

Syntax

LENGTH <*variable-1*> <*...variable-n*> <$> *length* <DEFAULT=*n*>;

Required Arguments

length
can range from 3 to 8 for numeric variables under UNIX. The minimum length that you can specify for a numeric variable depends on the floating-point format used by your system. Because most systems use the IEEE floating-point format, the minimum is 3 bytes.

DEFAULT=*n*
changes the default number of bytes that are used for storing the values of newly created numeric variables from 8 to the value of *n*. Under UNIX, *n* can range from 3 to 8.

See Also

"Data Representation" on page 217

LIBNAME Statement: UNIX

Associates or disassociates a SAS library with a libref (a shortcut name); clears one or all librefs; lists the characteristics of a SAS library; concatenates SAS libraries; implicitly concatenates SAS catalogs; turns off file locking.

Valid in:	anywhere
UNIX specifics:	*engine*, *library*, and *engine/host-options*
See:	"LIBNAME Statement" in *SAS Statements: Reference*

Syntax

LIBNAME *libref <engine>* '*SAS-library*' *<options> <engine/host-options>* ;

LIBNAME *libref <engine>* ('*library-1*'<,...'*library-n*'>) *<options>* ;

LIBNAME *libref* ('*library-1*'|*libref-1*,...,'*library-n*'|*libref-n*);

LIBNAME *libref* CLEAR | _ALL _ CLEAR;

LIBNAME *libref* LIST | _ALL _ LIST;

Required Argument

libref

is any valid libref as documented in "LIBNAME Statement" in *SAS Statements: Reference*. SAS reserves some librefs for special system libraries. For more information about reserved librefs, see "Librefs Assigned by SAS in UNIX Environments" on page 57.

Optional Arguments

engine

is one of the library engines supported under UNIX. For a description of the engines, see "Details" on page 335. If no engine name is specified, SAS determines which engine to use as described in "Omitting Engine Names from the LIBNAME Statement" on page 337.

'*SAS-library*'

differs based on the engine that you specify and based on your current working directory. Table 17.2 on page 336 describes what each engine expects for this argument. Specify directory pathnames as described in "Specifying Pathnames in UNIX Environments" on page 53. You cannot create directories with the LIBNAME statement. The directory that you specify must already exist, and you must have permissions to it. Enclose the library name in quotation marks. Remember that UNIX pathnames are cases sensitive.

'*library-n*'|*libref-n*

are pathnames or librefs (that have been assigned) for the libraries that you want to access with one libref. Use these forms of the LIBNAME statement when you want to concatenate libraries. Separate the pathnames with either commas or blank spaces. Enclose library pathnames in quotation marks. Do not enclose librefs in quotation marks. For more information about concatenating libraries, see "Assigning a Libref to Several Directories (Concatenating Directories)" on page 54.

options

are LIBNAME statement options that are available in all operating environments. For information about these options, see "LIBNAME Statement" in *SAS Statements: Reference*.

engine/host-options

can be any of the options described in "Engine and Host Options" on page 337.

ALL

refers to all librefs currently defined. You can use this keyword when you are listing or clearing librefs.

CLEAR

clears the specified libref, or, if you specify _ALL_, clears all librefs that are currently defined. Sasuser, Sashelp, and Work remain assigned.

Note: When you clear a libref defined by an environment variable, the variable remains defined, but it is no longer considered a libref. You can still reuse it, either as a libref or a fileref. For more information, see "Using Environment Variables as Librefs in UNIX Environments" on page 56.

SAS automatically clears the association between librefs and their respective libraries at the end of your job or session. If you want to associate an existing libref with a different SAS library during the current session, you do not have to end the session or clear the libref. SAS automatically reassigns the libref when you issue a LIBNAME statement for the new SAS library.

LIST

writes to the SAS log the engine, pathname, file format, access permissions, and so on, that are associated with the specified libref, or, if you specify _ALL_, prints this information for all librefs that are currently defined. Librefs defined as environment variables appear only if you have already used those librefs in a SAS statement.

NOSETPERM

specifies that permission settings are not inherited from one library member to another when the library members are opened with the same libref. If you have two assignments to a path, one with the NOSETPERM option and the other without, the two assignments are treated as if the paths do not match. The LIBNAME statement with the NOSETPERM option does not inherit permission settings.

Once the NOSETPERM option is used to turn off permission settings for a libref, the option is in effect whenever you use the libref. There is no option that turns off the NOSETPERM option. To turn off the NOSETPERM option, submit the following statement:

```
libname libref clear;
```

Details

Types of Engines

There are two main types of engines:

View engines

enable SAS to read SAS views that are described by SAS/ACCESS software, the SQL procedure, and DATA step views. The use of SAS view engines is automatic because the name of the view engine is stored as part of the descriptor portion of the SAS data set.

Library engines

control access at the SAS library level. Every SAS library has an associated library engine, and the files in that library can be accessed only through that engine. There are two types of library engines:

native engines

access SAS files created and maintained by SAS. See the following table for a description of these engines.

interface engines

treat other vendors' files as if they were SAS files. For more information, see the following table and "Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments" on page 63.

Table 17.2 *Engine Names and Descriptions*

Engine Type	Name (Alias)	Description	SAS Library
default	V9 (BASE) V8	enables you to create new SAS data files and to access existing SAS data files that were created with Version 8 or SAS 9. The V8 and V9 engines are identical. This engine enables Read access to data files that were created with some earlier releases of SAS, but this engine is the only one that supports SAS 9 catalogs. This engine allows for data set indexing and compression, and is also documented in *SAS Language Reference: Concepts*.	is the pathname of the directory containing the library.
compatibility	V6	accesses any data file that was created by Releases 6.09 through 6.12. This engine is read-only.	is the pathname of the directory containing the library.
servers	SPD Server	enables communication between a client session and a data server. You must have the Scalable Performance Data Server licensed on your client computer to use this engine. See the *SAS Scalable Performance Data Server: User's Guide* for more information.	is the logical LIBNAME domain name for a SAS Scalable Performance Data Server (SPD Server) library on the server. The name server resolves the domain name into the physical path for the library.
	MDDB	enables communication between a client session and an MDDB server. You must have SAS/MDDB Server licensed on your client computer or on your server to use this engine.	
transport	XPORT	accesses transport data sets. This engine creates computer-independent SAS transport files that can be used under all hosts running Release 6.06 or later of SAS. This engine is documented in *Moving and Accessing SAS Files*.	is the pathname of either a sequential device or a disk file.
XML	XML	generates (writes) and processes (reads) any XML document, which is an application- and computer-independent file.	is the pathname of the XML document.
interface	BMDP	provides Read-Only access to BMDP files. This engine is available only on AIX, HP-UX, and Solaris.	is the pathname of the data file.

Engine Type	Name (Alias)	Description	SAS Library
	OSIRIS	provides Read-Only access to OSIRIS files.	is the pathname of the data file.
	SPSS	provides Read-Only access to SPSS files.	is the pathname of the data file.

Omitting Engine Names from the LIBNAME Statement

It is always more efficient to specify the engine name than to have SAS determine the correct engine. However, if you omit an engine name in the LIBNAME statement or if you define an environment variable to serve as a libref, SAS determines the appropriate engine.

If you have specified the ENGINE= system option, SAS uses the engine name that you specified. For a discussion of the ENGINE= system option, see "ENGINE= System Option: UNIX" on page 380.

Note: The ENGINE= system option specifies the default engine for libraries on disk only.

If you did not specify the ENGINE= system option, SAS looks at the extensions of the files in the given directory and uses these rules to determine an engine:

- If all the SAS data sets in the library were created by the same engine, the libref is assigned using that engine.

  Note: If the engine used to create the data sets is not the same as the default engine, then you will not be able to create a view or stored program. For more information, see "Using Multiple Engines for a Library in UNIX Environments" on page 56.

- If there are no SAS data sets in the given directory, the libref is assigned using the default engine.
- If there are SAS data sets from more than one engine, the system issues a message about finding mixed engine types and assigns the libref using the default engine.

Engine and Host Options

The LIBNAME statement accepts the following options:

ENABLEDIRECTIO
: specifies that direct file I/O can be available for all files that are opened in the library that is identified in the LIBNAME statement.

  A libref that is assigned to a directory with the ENABLEDIRECTIO option will not match another libref that is assigned to the same directory without ENABLEDIRECTIO. The two librefs will point to the same directory, but the files that are opened using one libref will be read from and written to using Direct I/O. Files that are opened using the other libref will be read from and written to using the regular disk I/O calls.

  You must use the ENABLEDIRECTIO option with the USEDIRECTIO= option to turn on direct I/O for the file or files whose libref is listed in the LIBNAME statement. The following example uses the ENABLEDIRECTIO and USEDIRECTIO= LIBNAME options. In this case, all files that are referenced with libref test will be opened for direct I/O:

```
LIBNAME test '.' ENABLEDIRECTIO USEDIRECTIO=yes;
```

TIP The following example uses the ENABLEDIRECTIO LIBNAME option to enable files that are associated with the libref test to be opened for direct I/O. The USEDIRECTIO= data set option opens test.file1 for direct I/O. test.file2 is not opened for direct I/O, although it is enabled for direct I/O:

```
LIBNAME test'.'ENABLEDIRECTIO;
data test.file1(USEDIRECTIO=yes);
   ... more SAS statements ...
run;
data test.file2;
   ... more SAS statements ...
run;
```

FILELOCKS=NONE | FAIL | CONTINUE
specifies whether file locking is turned on or off for the files that are opened under the libref in the LIBNAME statement. The FILELOCKS statement option works like the FILELOCKS system option, except that it applies only to the files that are associated with the libref. The following values for the FILELOCKS statement option are available:

NONE
turns file locking off. NONE specifies that SAS attempts to open the file without checking for an existing lock on the file. NONE does not place an operating system lock on the file. These files are not protected from shared Update access.

FAIL
turns file locking on. FAIL specifies that SAS attempts to place an operating system lock on the file. Access to the file is denied if the file is already locked, or if it cannot be locked.

CONTINUE
turns file locking on. CONTINUE specifies that SAS attempts to place an operating system lock on the file. If the file is already locked by someone else, an attempt to open it fails. If the file cannot be locked for some other reason (for example, if the file system does not support locking), the file is opened and a warning message is sent to the log.

The FILELOCKS option in the LIBNAME statement applies to most (but not all) of the SAS I/O files (for example, data sets and catalogs) opened under the libref that is listed in the LIBNAME statement.

For the FILELOCKS statement option, RESET is not a valid value as it is when you use the FILELOCKS system option.

Use the FILELOCKS system option instead of the FILELOCKS statement option to set the locking behavior for your files. (The FILELOCKS statement option will be deprecated in a future release of SAS.) Note that the FILELOCK option in the LIBNAME statement overrides the LIBNAME system option. For more information, see the FILELOCKS system option in the UNIX operating environment. You can also specify any of the options supported by the Scalable Performance Data Server. Refer to *Scalable Performance Data Server: User's Guide* at the **Technical Support Web site** for a description of these options.

FILELOCKWAIT=*n*
specifies the number of seconds SAS will wait for a locked file to become available to another process.

If the locked file is released before the number of seconds specified by *n*, then SAS locks the file for the current process and continues. If the file is still locked when the number of seconds has been reached, then SAS writes a “Locked File” error to the log and the DATA step fails.

Interaction	Specifying the FILELOCKWAIT= option can have an adverse effect on one or more SAS/SHARE server and client sessions that are waiting for the release of a SAS file that is locked by another process. One or more wait conditions could lead to failed processes for a SAS/SHARE server and clients.
Interaction	To prevent the possibility of a failed SAS/SHARE process, you can set FILELOCKWAIT=0, which cancels the amount of time that a SAS/SHARE server and clients would wait for the release of a locked file. Canceling the wait time would prevent a failed process. For more information, see the FILELOCKWAITMAX system option. See also the FILELOCKWAITMAX system option in the section about predefining a server library by using the LIBNAME statement in the *SAS/SHARE: User’s Guide*.
Range	0–600
Default	0

TRANSFERSIZE=*n*K | *n*M

specifies the size of a large block of data that is read from a file that is opened.

n

specifies an integer value.

K

specifies the size of the block in kilobytes.

M

specifies the size of the block in megabytes.

To use the TRANSFERSIZE option, you must have files open for direct I/O. That is, both the ENABLEDIRECTIO and USEDIRECTIO= options must be in effect. If you use TRANSFERSIZE without the ENABLEDIRECTIO and USEDIRECTIO= options, the option is accepted, but it has no effect.

In the following example, 128k blocks of data are read from the test.file1 because this file is opened for direct I/O. test.file2 is not open for direct I/O, and the TRANSFERSIZE option has no effect on this file:

```
LIBNAME test'.'ENABLEDIRECTIO TRANSFERSIZE=128k;
data test.file1(USEDIRECTIO=yes);
   ... more SAS statements ...
run;
data test.file2;
   ... more SAS statements ...
run;
```

In the following example, all the files that are listed in the DATA statements read 128k blocks of data because all the files are affected by the ENABLEDIRECTIO, USEDIRECTIO=, and TRANSFERSIZE options:

```
LIBNAME test'.'ENABLEDIRECTIO USEDIRECTIO=yes TRANSFERSIZE=128k;
data test.file1;
   ... more SAS statements ...
run;
data test.file2;
```

```
      ... more SAS statements ...
   run;
   data test.file3;
      ... more SAS statements ...
   run;
```

USEDIRECTIO= YES | NO
: if used with the ENABLEDIRECTIO statement option, turns on or turns off direct file I/O for all the files associated with the libref listed in the LIBNAME statement. (See ENABLEDIRECTIO in "Engine and Host Options" on page 337.)

  Requirement
  : Use USEDIRECTIO= with the ENABLEDIRECTIO statement option to turn on direct file I/O.

See Also

System Options:

- "FILELOCKS System Option: UNIX" on page 381

Other References:

- "Introduction to SAS Files, Libraries, and Engines in UNIX Environments" on page 35

SYSTASK Statement: UNIX

Executes asynchronous tasks.

Valid in:	anywhere
UNIX specifics:	all

Syntax

SYSTASK COMMAND "*operating-environment-command*" <WAIT | NOWAIT> <TASKNAME=*taskname*> <MNAME=*name-variable*> <STATUS=*status-variable*> <SHELL<="*shell-command*">> <CLEANUP> ;

SYSTASK LIST <_ALL_ | *taskname*> <STATE> <STATVAR> ;

SYSTASK KILL *taskname* <*taskname*...> ;

Required Arguments

COMMAND
: executes the *operating–environment-command*.

LIST
: lists either a specific active task or all of the active tasks in the system. A task is *active* if it is running or if it has completed and has not been waited for using the WAITFOR statement.

KILL
: forces the termination of the specified tasks.

operating-environment-command
specifies the name of a UNIX command (including any command-specific options) or the name of an X window system application. Enclose the command in either single or double quotation marks. If the command-specific options require quotation marks, repeat them for each option. For example:

```
SYSTASK COMMAND "xdialog -m ""There was an error."" -t ""Error"" -o";
```

Note: If the command name is a shell alias, or if you use the shell special characters tilde (~) and asterisk (*) in a pathname in a command, you need to specify the SHELL option so that the shell will process the alias or special characters:

```
SYSTASK COMMAND "mv ~usr/file.txt /tmp/file.txt" shell;
```

In this example, by using the SHELL option, the ~usr path is expanded on execution and is not executed directly.

Note: The *operating-environment-command* that you specify cannot require input from the keyboard.

TIP If using a shell alias results in an error even though the SHELL option is used, then the shell is not processing your shell initialization files. Use the actual SHELL command instead of the SHELL alias.

Optional Arguments

WAIT | NOWAIT
determines whether SYSTASK COMMAND suspends execution of the current SAS session until the task has completed. NOWAIT is the default. For tasks that start with the NOWAIT option, you can use the WAITFOR statement when necessary to suspend execution of the SAS session until the task has finished. For more information, see "WAITFOR Statement: UNIX" on page 344.

TASKNAME=*taskname*
specifies a name that identifies the task. Task names must be unique among all active tasks. A task is active if it is running or if it has completed and has not been waited for using the WAITFOR statement. Duplicate task names generate an error in the SAS log. If you do not specify a task name, SYSTASK will automatically generate a name. If the task name contains a blank character, enclose it in quotation marks.

Task names cannot be reused, even if the task has completed, unless you either issue the WAITFOR statement for the task or you specify the CLEANUP option.

MNAME=*name-variable*
specifies a macro variable in which you want SYSTASK to store the task name that it automatically generated for the task. If you specify both the TASKNAME option and the MNAME option, SYSTASK copies the name that you specified with TASKNAME into the variable that you specified with MNAME.

STATUS=*status-variable*
specifies a macro variable in which you want SYSTASK to store the status of the task. Status variable names must be unique among all active tasks.

SHELL<="*shell-command*">
specifies that the *operating-environment-command* should be executed with the operating system shell command. The shell will expand shell special characters that are contained in the *operating-environment-command*. If you specify a *shell-command*, SYSTASK uses the shell command that you specify to invoke the shell. Otherwise, SYSTASK uses the default shell. Enclose the shell command in quotation marks.

Note: The SHELL option assumes that the SHELL command that you specify uses the -i option to pass statements. Usually, your shell command will be **`sh`**, **`csh`**, **`ksh`**, or **`bash`**.

CLEANUP
specifies that the task should be removed from the LISTTASK output when the task completes. Once the task is removed, you can reuse the task name without issuing the WAITFOR statement.

If you have long-running jobs that use the SYSTASK command multiple times, use the WAITFOR statement or the CLEANUP option in the SYSTASK command to clear the memory. The WAITFOR statement releases memory by removing the information for all completed processes that were started by the SYSTASK command. The CLEANUP option clears memory when a specific job completes, and releases memory for further use. If you use the WAITFOR statement after a job has completed, the statement is ineffective because the job has already been cleaned up by the CLEANUP option.

Details

The SYSTASK statement enables you to execute host-specific commands from within your SAS session or application. Unlike the X statement, the SYSTASK statement runs these commands as asynchronous tasks, which means that these tasks execute independently of all other tasks that are currently running. Asynchronous tasks run in the background, so you can perform additional tasks while the asynchronous task is still running.

For example, to start a new shell and execute the UNIX **`cp`** command in that shell, you might use this statement:

```
systask command "cp /tmp/sas* ~/archive/" taskname="copyjob1"
                status=copysts1 shell;
```

The return code from the **`cp`** command is saved in the macro variable COPYSTS1.

The output from the command is displayed in the SAS log.

Because the syntax between UNIX and a PC can be different, converting PC SAS jobs to run on UNIX might result in an error in the conversion process. For example, entering the following command results in an error:

```
systask command "md directory-name" taskname="mytask";
```

An error occurs because **`md`** is a "make directory" command on a PC, but has no meaning in UNIX. In the conversion process, **`md`** becomes **`mkdir`**. You must use the SHELL option in the SYSTASK statement because **`mkdir`** is built into the UNIX shell, and it is not a separate command as it is on a PC.

SAS writes the error message to the log.

Note: Program steps that follow the SYSTASK statements in SAS applications usually depend on the successful execution of the SYSTASK statements. Therefore, syntax errors in some SYSTASK statements will cause your SAS application to abort.

There are two types of asynchronous processes that can be started from SAS:

Task
All tasks started with SYSTASK COMMAND are of type Task. For these tasks, if you do not specify STATVAR or STATE, then SYSTASK LIST displays the task name, type, and state, and the name of the status macro variable. You can use SYSTASK KILL to kill only tasks of type Task.

SAS/CONNECT Process
Tasks started from SAS/CONNECT with the SIGNON statement or command and RSUBMIT statement are of type SAS/CONNECT Process. To display SAS/CONNECT processes, use the LISTTASK statement to displays the task name, type, and state. To terminate a SAS/CONNECT process, use the KILLTASK statement. For information about SAS/CONNECT processes, see the *SAS/CONNECT User's Guide*.

Note: The preferred method for displaying any task (not just SAS/CONNECT processes) is to use the LISTTASK statement instead of SYSTASK LIST. The preferred method for ending a task is using the KILLTASK statement instead of SYSTASK KILL.

The SYSRC macro variable contains the return code for the SYSTASK statement. The status variable that you specify with the STATUS option contains the return code of the process started with SYSTASK COMMAND. To ensure that a task executes successfully, you should monitor both the status of the SYSTASK statement and the status of the process that is started by the SYSTASK statement.

If a SYSTASK statement cannot execute successfully, the SYSRC macro variable will contain a nonzero value. For example, there might be insufficient resources to complete a task or the SYSTASK statement might contain syntax errors. With the SYSTASK KILL statement, if one or more of the processes cannot be killed, SYSRC is set to a nonzero value.

When a task is started, its status variable is set to NULL. You can use the status variables for each task to determine which tasks failed to complete. Any task whose status variable is NULL did not complete execution. If a task terminates abnormally, then its status variable will be set to `-1`. For more information about the status variables, see “WAITFOR Statement: UNIX” on page 344.

Unlike the X statement, you cannot use the SYSTASK statement to start a new interactive session.

See Also

Statements:

- “WAITFOR Statement: UNIX” on page 344
- “X Statement: UNIX” on page 346

Other References:

- “Executing Operating System Commands from Your SAS Session” on page 15

TITLE Statement: UNIX

Specifies title lines for SAS output.

Valid in:	anywhere
UNIX specifics:	maximum length of title
See:	“TITLE Statement” in *SAS Statements: Reference*

Syntax

TITLE <*n*> <'*text*' | "*text*"> ;

Details

In interactive modes, the maximum title length is 254 characters. Otherwise, the maximum length is 200 characters. If the length of the specified title is greater than the value of the LINESIZE option, the title is truncated to the line size.

WAITFOR Statement: UNIX

Suspends execution of the current SAS session until the specified tasks finish executing.

Valid in:	anywhere
UNIX specifics:	all

Syntax

WAITFOR <_ANY | _ALL_> *taskname* <*taskname...*> <TIMEOUT=*seconds*> ;

Required Argument

taskname
specifies the name of the tasks that you want to wait for. For information about task names, see "SYSTASK Statement: UNIX" on page 340. The task names that you specify must match exactly the task names assigned through the SYSTASK COMMAND statement. You cannot use wildcards to specify task names.

Optional Arguments

ANY | _ALL_
suspends execution of the current SAS session until either one or all of the specified tasks finishes executing. The default setting is _ANY_, which means that as soon as one of the specified tasks completes executing, the WAITFOR statement will finish executing.

TIMEOUT=*seconds*
specifies the maximum number of seconds that WAITFOR should suspend the current SAS session. If you do not specify the TIMEOUT option, WAITFOR will suspend execution of the SAS session indefinitely.

Details

The WAITFOR statement suspends execution of the current SAS session until the specified tasks finish executing or until the TIMEOUT= interval has elapsed. If the specified task was started with the WAIT option, then the WAITFOR statement ignores that task. For a description of the WAIT option, see "SYSTASK Statement: UNIX" on page 340.

For example, the following statements start three different X client programs and waits for them to complete:

```
systask command "xv" taskname=pgm1;
systask command "xterm" taskname=pgm2;
systask command "xcalc" taskname=pgm3;
waitfor _all_ pgm1 pgm2 pgm3;
```

The WAITFOR statement can be used to execute multiple concurrent SAS sessions. The following statements start three different SAS jobs and suspend the execution of the current SAS session until those three jobs have finished executing:

```
systask command "sas myprog1.sas" taskname=sas1;
systask command "sas myprog2.sas" taskname=sas2;
systask command "sas myprog3.sas" taskname=sas3;
waitfor _all_ sas1 sas2 sas3;
```

Note: In this method, SAS terminates after each command, which can result in reduced performance. SAS/CONNECT can also be used for executing parallel SAS sessions. See the *SAS/CONNECT User's Guide* for more information.

If you have long-running jobs that use the SYSTASK command multiple times, use the WAITFOR statement or the CLEANUP option in the SYSTASK command to clear the memory. The WAITFOR statement releases memory by removing the information for all completed processes that were started by the SYSTASK command. The CLEANUP option clears memory when a specific job completes, and releases memory for further use. If you use the WAITFOR statement after a job has completed, the statement is ineffective because the job has already been cleaned up by the CLEANUP option.

The SYSRC macro variable contains the return code for the WAITFOR statement. If a WAITFOR statement cannot execute successfully, the SYSRC macro variable will contain a nonzero value. For example, the WAITFOR statement might contain syntax errors. If the number of seconds specified with the TIMEOUT option elapses, then the WAITFOR statement finishes executing, and SYSRC is set to a nonzero value if one of the following occurs:

- you specify a single task that does not finish executing
- you specify more than one task and the _ANY_ option (which is the default setting), but none of the tasks finishes executing
- you specify more than one task and the _ALL_ option, and any one of the tasks does not finish executing

Any task whose status variable is still NULL after the WAITFOR statement has executed did not complete execution. For a description of status variables for individual tasks, see "SYSTASK Statement: UNIX" on page 340.

See Also

Statements:

- "SYSTASK Statement: UNIX" on page 340
- "X Statement: UNIX" on page 346

Other References:

- *SAS/CONNECT User's Guide*
- "Executing Operating System Commands from Your SAS Session" on page 15

X Statement: UNIX

Issues an operating environment command from within a SAS session.

Valid in:	anywhere
UNIX specifics:	valid operating system command
See:	"X Statement" in *SAS Statements: Reference*

Syntax

X <'*operating-system-command*'> ;

Optional Argument

operating-system-command
specifies the UNIX command. If you specify only one UNIX command, you do not need to enclose it in quotation marks. Also, if you are running SAS from the Korn shell, you cannot use aliases.

Details

The X statement issues a UNIX command from within a SAS session. SAS executes the X statement immediately.

Neither the X statement nor the %SYSEXEC macro program statement is intended for use during the execution of a DATA step. The CALL SYSTEM routine, however, can be executed within a DATA step. For an example, see "CALL SYSTEM Routine: UNIX" on page 263.

Note: The X statement is not supported without arguments under the X Window System.

See Also

"Executing Operating System Commands from Your SAS Session" on page 15

Chapter 18
System Options under UNIX

SAS System Options under UNIX

Behavior or Syntax That Is Specific to UNIX Environments

This section describes SAS system options that have behavior or syntax that is specific to UNIX environments. Each system option description includes a brief “UNIX specifics” section that explains which aspect of the system option is specific to UNIX. If the information under “UNIX specifics” is “all,” then the system option is described only in this documentation. Otherwise, the system option is described in this documentation and in *SAS System Options: Reference*.

When You Use Parentheses in a Command Line

On a command line, if arguments are enclosed in quotation marks, then you must use a backslash before the open parenthesis and close parenthesis so that UNIX can interpret the arguments correctly.

List of SAS System Options for UNIX

See "Summary of All SAS System Options in UNIX Environments" on page 350 for a table of all of the system options available under UNIX.

Determining How a SAS System Option Was Set

Interactions between SAS options can lead to unintended consequences. For example, if you set both the FULLSTIMER system option and the NONOTES system option, the result will be that no FULLSTIMER information is written to the SAS log. Because it is possible to set one option in a configuration file and the other option in an OPTIONS statement, the reason for such a problem might not be readily apparent.

When you issue a PROC OPTIONS statement with the VALUE option to query the value of an option, the value of the option appears in the SAS log, along with the method (or location) that was used to set that option. If the option was set in a configuration file, then the **Config file name** field lists the name of the file. For example, the following output is displayed in the SAS log when you query the value of the MEMSIZE system option:

```
proc options option=MEMSIZE value;
run;
```

```
Option Value Information for SAS Option MEMSIZE
   Value: 100663296
   Scope: SAS Session
   How option value set: Config file
   Config file name: /usr/local/SAS/SASFoundation/9.3/sasv9_local.cfg
```

You can issue a PROC OPTIONS statement to query the value of the WORK system option. The WORK value can be set form a server configuration file, and environment setting, or a command line setting. The WORK path is generated by combining the initial server-specified WORK path with a host-specific value and an executive suffix. The following example shows the information that is written to the SAS log:

```
proc options option=WORK value;
run;
```

```
Option Value Information for SAS Option WORK
   Value: /sastemp/SAS_workA1234567_bcd89
   Scope: SAS Session
   How option value set: Config file
   Config file name: /usr/local/SAS/SASFoundation/9.3/sasv9_local.cfg
```

Restricted Options

Restricted options are system options whose values are determined by the site administrator. They cannot be overridden. The site administrator can create a restricted options table that specifies the option values that are restricted when SAS starts. Any attempt to modify a system option that is listed in the restricted options table results in a message to the SAS log. The message indicates that the system option has been restricted by the site administrator and cannot be updated. For more information, see “Restricted Options” in Chapter 1 of *SAS System Options: Reference*.

Summary of All SAS System Options in UNIX Environments

The following table lists every SAS system option available under UNIX. Many of these options have no host-specific behavior and are described in *SAS System Options: Reference*. If an option is available only under UNIX, it is described in this documentation. If an option is available under all environments, but has some environment-specific behavior, it is described in *SAS System Options: Reference* and this documentation. Use the following legend to know how to locate more information about an option:

Access
: *SAS/ACCESS for Relational Databases: Reference*

Comp
: indicates that the option is completely described in this section

Conn
: *SAS/CONNECT User's Guide*

DQ
: *SAS Data Quality Server Reference*

DST
: *Encryption in SAS*

ARM
: *SAS Interface to Application Response Measurement (ARM): Reference*

SOR
: *SAS System Options: Reference*

Macro
: *SAS Macro Language: Reference*

NLS
: *SAS National Language Support (NLS): Reference Guide*

SAS/SHARE
: *SAS/SHARE User's Guide*

SPD Engine
: *SAS Scalable Performance Data Engine: Reference*

Web
: indicates that the option is described in documentation on the SAS `Technical Support Web site`.

The table also shows the default value for each option and where you can specify the option:

- at initialization: in the SAS command, in the SASV9_OPTIONS environment variable, or in the configuration file
- in the OPTIONS statement
- in the System Options window

Table 18.1 *Summary of All SAS System Options*

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win-dow**	**See**
ALTLOG	NOALTLOG	X			Comp
ALTPRINT	NOALTPRINT	X			Comp
APPEND	none	X	X		Comp/ SOR
APPLETLOC	none	X	X	X	SOR
ARMAGENT	none	X	X	X	ARM
ARMLOC	ARMLOC.LOG	X	X	X	ARM
ARMSUBSYS	ARM_NONE	X	X	X	ARM
AUTHPROVIDERDOMAIN	NULL	X			SOR/ Comp
AUTOEXEC	see description	X			Comp
AUTOSAVELOC	none	X	X	X	Comp/ SOR
AUTOSIGNON	NOAUTOSIGNON	X	X	X	Conn
BINDING	DEFAULT	X	X	X	SOR
BLKSIZE	256	X	X	X	SOR
BOTTOMMARGIN	0.000	X	X	X	SOR

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
BUFNO	1	X	X	X	Comp/ SOR
BUFSIZE	0	X	X	X	Comp/ SOR
BYERR	BYERR	X	X	X	SOR
BYLINE	BYLINE	X	X	X	SOR
BYSORTED	BYSORTED	X			SOR
CAPS	NOCAPS	X	X	X	SOR
CARDIMAGE	NOCARDIMAGE	X	X	X	SOR
CATCACHE	0	X			Comp/ SOR
CBUFNO	0	X	X	X	SOR
CENTER	CENTER	X	X	X	SOR
CGOPTIMIZE		X	X	X	SOR
CHARCODE	NOCHARCODE	X	X	X	SOR
CLEANUP	see description	X	X	X	Comp/ SOR
CMDMAC	NOCMDMAC	X	X	X	Macro
CMPLIB	none	X	X	X	SOR
CMPMODEL	BOTH	X	X	X	SOR
CMPOPT	ALL	X	X		SOR
COLLATE	NOCOLLATE	X	X	X	SOR
COLORPRINTING	COLORPRINTING	X	X	X	SOR
COMAMID	TCP/IP	X	X	X	Conn/ Share
COMAUX1	none	X			Share

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
COMPRESS	NO	X	X	X	SOR/ SPDE
CONFIG	see description	X (also SASV9_ CONFIG environment variable)			Comp
CONNECTPERSIST	CONNECTPERSIST	X	X	X	Conn
CONNECTREMOTE	none	X	X	X	Conn
CONNECTSTATUS	CONNECTSTATUS	X	X	X	Conn
CONNECTWAIT	CONNECTWAIT	X	X	X	Conn
COPIES	1	X	X	X	SOR
CPUCOUNT	1-1024 or ACTUAL	X	X	X	SOR
CPUID	CPUID	X			SOR
DATASTMTCHK	COREKEYWORDS	X	X	X	SOR
DATE	DATE	X	X	X	SOR
DATESTYLE	MDY	X	X	X	SOR
DBCS	NODBCS	X			NLS
DBCSLANG	none	X			NLS
DBCSTYPE	see description	X			NLS
DBSLICEPARM	(THREADED_APPS, 2)	X	X	X	Access
DBSRVTP	NONE	X			Access
DEFLATION	6	X	X	X	SOR
DETAILS	NODETAILS	X	X	X	SOR

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
DEVICE	none	X (also, SASV9_ OPTIONS environment variable, GOPTIONS statement)	X	X	Comp/ SOR
DFLANG	ENGLISH	X	X	X	NLS
DKRICOND	ERROR	X	X	X	SOR
DKROCOND	WARN	X	X	X	SOR
DLCREATEDIR	NODLCREATEDIR	X	X	X	SOR
DLDMGACTION	REPAIR	X	X	X	SOR
DMR	NODMR	X			SOR/ Conn
DMS	DMS	X			SOR
DMSEXP	DMSEXP	X			SOR
DMSLOGSIZE	99999	X			SOR
DMSOUTSIZE	99999	X			SOR
DMSPGMLINESIZE	136	X			SOR
DMSSYNCHK	NODMSSYNCHK	X	X	X	SOR
DQLOCALE	none	X	X	X	DQ
DQSETUPLOC	none	X	X	X	DQ
DSNFERR	DSNFERR	X	X	X	SOR
DTRESET	NODTRESET	X	X	X	SOR
DUPLEX	NODUPLEX	X	X	X	SOR
ECHO	none	X			Comp

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_ OPTIONS, Config File	OPTIONS Statement	SAS Sys Opts Win- dow	See
ECHOAUTO	NOECHOAUTO	X			SOR
EDITCMD	none	X	X		Comp
EMAILAUTHPROTOCOL	none	X	X	X	SOR
EMAILFROM	none	X	X	X	SOR
EMAILHOST	LOCALHOST	X			SOR
EMAILID	none	X	X	X	SOR
EMAILPORT	25	X	X	X	SOR
EMAILPW	none	X	X	X	SOR
EMAILSYS	SMTP	X (except SASV9_ OPTIONS environment variable)	X	X	Comp
ENCODING	LATIN1	X			NLS
ENGINE	V9	X			Comp/ SOR
ERRORABEND	NOERRORABEND	X	X	X	SOR
ERRORBYABEND	NOERRORBYABEND	X	X	X	SOR
ERRORCHECK	NORMAL	X	X	X	SOR
ERRORS	20	X	X	X	SOR
EXPLORER	NOEXPLORER	X			SOR
FILELOCKS	FAIL	X	X	X	Comp
FILELOCKWAITMAX	600	X			Comp
FILESYNC	HOST	X			SOR
FIRSTOBS	1	X	X	X	SOR

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
FMTERR	FMTERR	X	X	X	SOR
FMTSEARCH	WORK LIBRARY	X	X	X	Comp/ SOR
FONTEMBEDDING	FONTEMBEDDING	X	X	X	SOR
FONTRENDERING	FREETYPE_POINTS	X	X	X	SOR
FONTSLOC	!SASROOT/misc/fonts	X			Comp/ SOR
FORMCHAR	`|----|+|---+=| -/ \<>*`	X	X	X	SOR
FORMDLIM	none	X	X	X	SOR
FORMS	DEFAULT	X	X	X	SOR
FSDBTYPE	DEFAULT	X			NLS
FSIMM	none	X			NLS
FSIMMOPT	none	X			NLS
FULLSTIMER	NOFULLSTIMER	X	X		Comp
GSTYLE	GSTYLE	X	X	X	SOR
GWINDOW	GWINDOW	X	X	X	SOR
HELPBROWSER	REMOTE	X	X	X	SOR
HELPENCMD	X (except SASV9_OPTIONS)				SOR
HELPHOST	null	X	X	X	Comp, SOR
HELPINDEX	/help/common.hlp/index.txt /help/common.hlp/ keywords.htm common.hhk	X			Comp
HELPLOC	!SASROOT/X11/native_help/en	X			Comp

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
HELPPORT	0	X	X	X	SOR
HELPTOC	/help/helpnav.help/config.txt /help/common.hlp /toc.htm, common.hhc	X			Comp
HTTPSERVERPORTMAX		X	X		SOR
HTTPSERVERPORTMIN		X	X		SOR
IBUFNO	0	X	X	X	SOR
IBUFSIZE	0	X	X	X	SOR
IMPLMAC	NOIMPLMAC	X	X	X	Macro
INITCMD	none	X			SOR
INITSTMT	none	X			SOR
INSERT	none		X	X	Comp/ SOR
INTERVALDS	none	X	X	X	SOR
INVALIDDATA	.	X	X	X	SOR
JPEGQUALITY	75	X	X	X	SOR
JREOPTIONS	none	X			Comp
LABEL	LABEL	X	X	X	SOR
LAST	_NULL_	X	X	X	SOR
LEFTMARGIN	0	X	X	X	SOR
LINESIZE	see description	X	X	X	Comp/ SOR
LOCALE	ENGLISH_UNITEDSTATES	X	X	X	NLS
LOG	see description	X			Comp

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
LOGPARM	none	X			SOR
LPTYPE	none	X	X		Comp
LRECL	256	X	X	X	SOR
MACRO	MACRO	X			Macro
MAPS	!SASROOT/maps	X	X	X	Comp/ SOR
MAUTOLOCDISPLAY	NOMAUTOLOCDISPLAY	X	X	X	Macro
MAUTOSOURCE	MAUTOSOURCE	X	X	X	Macro
MAXMEMQUERY	256M	X	X		Comp
MAXSEGRATIO	75	X	X	X	SPDE
MCOMPILENOTE	NONE	X	X	X	Macro
MEMSIZE	!SASROOT/sasv9.cfg	X			Comp
MERGENOBY	NOWARN	X	X	X	SOR
MERROR	MERROR	X	X	X	Macro
MEXECNOTE	NOMEXECNOTE	X	X	X	Macro
MEXECSIZE	65536	X	X	X	Macro
MFILE	NOMFILE	X	X	X	Macro
MINDELIMITER	none	X	X	X	Macro
MINPARTSIZE	16M	X			SPDE
MISSING	.	X	X	X	SOR
MLOGIC	NOMLOGIC	X	X	X	Macro
MLOGICNEST	NOMLOGICNEST	X	X	X	Macro
MPRINT	NOMPRINT	X	X	X	Macro

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win-dow**	**See**
MPRINTNEST	NOMPRINTNEST	X	X	X	Macro
MRECALL	NOMRECALL	X	X	X	Macro
MSG	!SASROOT/sasmsg	X			Comp
MSGCASE	NOMSGCASE	X			Comp
MSGLEVEL	N	X	X	X	SOR
MSTORED	NOMSTORED	X	X	X	Macro
MSYMTABMAX	4M	X	X	X	Comp/ Macro
MULTENVAPPLE	NOMULTENVAPPLE	X	X		SOR
MVARSIZE	32K	X	X	X	Comp/ Macro
NETENCRYPT	NONETENCRYPT	X	X	X	DST
NETENCRYPTALGORITHM	none	X	X	X	DST
NETENCRYPTKEYLEN	0	X	X	X	DST
NEWS	!SASROOT/misc/base/news	X			Comp/ SOR
NLSCOMPATMODE	NONLSCOMPATMODE	X			NLS
NOTES	NOTES	X	X	X	SOR
NUMBER	NUMBER	X	X	X	SOR
OBS	MAX	X	X	X	Comp/ SOR
OPLIST	NOOPLIST	X			Comp
ORIENTATION	PORTRAIT	X	X	X	SOR
OVP	NOOVP	X	X	X	SOR
PAGEBREAKINITIAL	NOPAGEBREAKINITIAL	X			SOR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_ OPTIONS, Config File	OPTIONS Statement	SAS Sys Opts Win-dow	See
PAGENO	1	X	X	X	SOR
PAGESIZE	see description	X	X	X	Comp/ SOR
PAPERDEST	none	X	X	X	SOR
PAPERSIZE	LETTER	X	X	X	SOR
PAPERSOURCE	none	X	X	X	SOR
PAPERTYPE	PLAIN	X	X	X	SOR
PARM	none	X	X	X	SOR
PARMCARDS	FT15F001	X	X	X	SOR
PATH	!SASROOT/sasexe	X			Comp
PDFACCESS	none	X	X	X	SOR
PDFASSEMBLY	NOPDFASSEMBLY	X	X	X	SOR
PDFCOMMENT	NOPDFCOMMENT	X	X	X	SOR
PDFCONTENT	none	X	X	X	SOR
PDFCOPY	PDFCOPY	X	X	X	SOR
PDFFILLIN	none	X	X	X	SOR
PDFPAGELAYOUT	DEFAULT	X	X	X	SOR
PDFPAGEVIEW	DEFAULT	X	X	X	SOR
PDFPASSWORD		X	X		SOR
PDFPRINT	HRES	X	X	X	SOR
PDFSECURITY	NONE	X	X	X	SOR
PRIMARYPROVIDERDOMAIN	none	X (except SASV9_ OPTIONS)			Comp/ SOR

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
PRINT	see description	X			Comp
PRINTCMD	none	X	X		Comp
PRINTERPATH	PostScript Level 1	X			SOR
PRINTINIT	NOPRINTINIT	X			SOR
PRINTMSGLIST	PRINTMSGLIST	X	X	X	SOR
QUOTELENMAX	QUOTELENMAX	X	X	X	SOR
REALMEMSIZE	0	X			Comp
REPLACE	REPLACE	X	X	X	SOR
REUSE	NO	X	X	X	SOR
RIGHTMARGIN	0.000	X	X	X	SOR
RLANG[1]	none	X			SOR
RSASUSER	NORSASUSER	X			Comp/ SOR
RTRACE	none	X			Comp
RTRACELOC	none	X			Comp
S	none	X	X	X	SOR
S2	none	X	X	X	SOR
S2V	S2	X	X	X	SOR
SASAUTOS	SASAUTOS fileref	X	X	X	Comp/ Macro
SASCMD	none	X	X	X	Conn
SASFRSCR	none	Valid in SAS Component Language			Conn
SASHELP	!SASROOT/sashelp	X			Comp/ SOR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_ OPTIONS, Config File	OPTIONS Statement	SAS Sys Opts Win- dow	See
SASMSTORE	none	X	X	X	Macro
SASSCRIPT	!SASROOT/misc/connect	X	X	X	Comp/ Conn
SASUSER	sasuser ~/sasuser.v93 (config file)	X			Comp/ SOR
SEQ	8	X	X	X	SOR
SERROR	SERROR	X	X	X	Macro
SET	none	X	X		Comp
SETINIT	NOSETINIT	X			SOR
SHARESESSIONCNTL	SERVER	X	X	X	Share
SIGNONWAIT	SIGNONWAIT	X	X	X	Conn
SKIP	0	X	X	X	SOR
SOLUTIONS	SOLUTIONS	X			SOR
SORTANOM	none	X	X		Comp
SORTCUT	0	X	X		Comp
SORTCUTP	0	X	X		Comp
SORTDEV	see description	X	X		Comp
SORTDUP	PHYSICAL	X	X	X	SOR
SORTEQUALS	SORTEQUALS	X	X	X	SOR
SORTNAME	none	X	X		Comp
SORTPARM	none	X	X		Comp
SORTPGM	BEST	X	X		Comp
SORTSEQ	none	X	X	X	NLS

		Can Be Specified In			
Name	**Default**	**SAS Invocation, SASV9_ OPTIONS, Config File**	**OPTIONS Statement**	**SAS Sys Opts Win- dow**	**See**
SORTSIZE	value of MAX	X	X	X	Comp/ SOR
SORTVALIDATE	NOSORTVALIDATE	X	X	X	SOR
SOURCE	SOURCE	X	X	X	SOR
SOURCE2	NOSOURCE2	X	X	X	SOR
SPDEINDEXSORTSIZE	32M	X	X	X	SPDE
SPDEMAXTHREADS	0	X			SPDE
SPDESORTSIZE	32M	X	X	X	SPDE
SPDEUTILLOC	none	X			SPDE
SPDEWHEVAL	COST	X			SPDE
SPOOL	NOSPOOL	X	X	X	SOR
SQLCONSTDATETIME	none	X	X	X	SOR
SQLMAPPUTTO	SAS_PUT	X (except SASV9_ OPTIONS)	X	X	Access
SQLREDUCEPUT	DBMS	X	X	X	SOR
SQLREDUCEPUTOBS	0	X	X	X	SOR
SQLREDUCEPUTVALUES	0	X	X	X	SOR
SQLREMERGE	none	X	X	X	SOR
SQLUNDOPOLICY	REQUIRED	X	X		SOR
SSLCALISTLOC	none	X	X	X	DST
SSLCERTLOC	none	X	X	X	DST
SSLCLIENTAUTH	NOSSLCLIENTAUTH	X	X	X	DST
SSLCRLCHECK	NOSSLCRLCHECK	X	X	X	DST

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_ OPTIONS, Config File	OPTIONS Statement	SAS Sys Opts Win- dow	See
SSLCRLLOC	none	X	X	X	DST
SSLPVTKEYLOC	none	X	X	X	DST
SSLPVTKEYPASS	none	X	X	X	DST
STARTLIB	STARTLIB	X			SOR
STDIO	NOSTDIO	X			Comp
STEPCHKPT	NOSTEPCHKPT	X			SOR
STEPCHKPTLIB	WORK	X			SOR
STEPRESTART	none	X			SOR
STIMEFMT	NLDATM2. HMS TIMEAMPM KB MEMFULL TSFULL NC	X	X		Comp
STIMER	STIMER	X	X		Comp
SUMSIZE	0	X	X	X	SOR
SVGCONTROLBUTTONS	NOSVGCONTROLBUTTONS	X	X	X	SOR
SVGHEIGHT	none	X	X	X	SOR
SVGPRESERVEASPECTRATIO	none	X	X	X	SOR
SVGTITLE	none	X	X	X	SOR
SVGVIEWBOX	none	X	X	X	SOR
SVGWIDTH	none	X	X	X	SOR
SVGX	none	X	X	X	SOR
SVGY	none	X	X	X	SOR
SYMBOLGEN	NOSYMBOLGEN	X	X	X	Macro
SYNTAXCHECK	SYNTAXCHECK	X	X	X	SOR
SYSIN	none	X			Comp

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_ OPTIONS, Config File	OPTIONS Statement	SAS Sys Opts Win- dow	See
SYSPARM	none	X	X	X	Macro
SYSPRINT	default system printer	X	X		Comp
SYSPRINTFONT	none	X	X	X	SOR
TBUFSIZE	0	X	X	X	Conn/ Share
TCPPORTFIRST	0	SAS invocation on the remote host			Conn
TCPPORTLAST	0	SAS invocation on the remote host			Conn
TERMINAL	TERMINAL	X			SOR
TERMSTMT	none	X			SOR
TEXTURELOC	!SASROOT/misc/textures	X	X	X	SOR
THREADS	THREADS	X	X	X	SOR
TOOLSMENU	TOOLSMENU	X			SOR
TOPMARGIN	0.000	X	X	X	SOR
TRAINLOC	none	X			SOR
TRANTAB	none	X	X	X	NLS
UPRINTCOMPRESSION	UPRINTCOMPRESSION	X	X	X	SOR
UNIVERSALPRINT	UNIVERSALPRINT	X			SOR
USER	none	X	X	X	Comp/ SOR
UTILLOC	WORK	X			SOR
UUIDCOUNT	100	X	X	X	SOR
UUIDGENDHOST	none	X			SOR
V6CREATEUPDATE	NOTE	X			SOR
VALIDFMTNAME	LONG	X	X	X	SOR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_ OPTIONS, Config File	OPTIONS Statement	SAS Sys Opts Win- dow	See
VALIDVARNAME	V7	X	X	X	SOR
VARLENCHR		X	X	X	SOR
VERBOSE	NOVERBOSE	X			Comp
VIEWMENU	VIEWMENU	X			SOR
VNFERR	VNFERR	X	X	X	SOR
WORK	see description	X			Comp/ SOR
WORKINIT	WORKINIT	X			Comp/ SOR
WORKPERMS	700	X			Comp
WORKTERM	WORKTERM	X	X	X	SOR
YEARCUTOFF	1920	X	X	X	SOR

[1] For 9.3, RLANG is available on Linux only.

Dictionary

ALTLOG System Option: UNIX

Specifies the destination for the SAS log.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES, LOGCONTROL
Default:	NOALTLOG
UNIX specifics:	all

Syntax

–ALTLOG *file-specification* | –NOALTLOG

Required Arguments

-ALTLOG *file-specification*
: specifies the location where an alternate SAS log is to be written. The *file-specification* argument can be any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the SAS log is placed in a file in the specified directory. The name of the file will be *filename*.log, where *filename* is the name of your SAS job. If you are running SAS interactively and specify only the path to a directory, the log is written to a file named sas.log within that path.

-NOALTLOG
: specifies that the SAS log is not copied.

Details

ALTLOG Basics

The ALTLOG system option specifies a destination to which a copy of the SAS log is written. All messages that are written to the SAS log are also written to the location specified in *file-specification*. You can use this option to capture log output for printing.

Note: You can use the LOG option in the PRINTTO procedure to redirect any portion of the log to an external file. The code for PROC PRINTTO will not appear in the SAS log for the current session, but it will appear in the SAS log that you created with the ALTLOG system option.

Note: When SAS is started with the OBJECTSERVER and NOTERMINAL system options and no log is specified, SAS discards all log and alternate log messages.

Using Directives with ALTLOG

Using directives in the ALTLOG system option enables you to control when log copies are open and closed, and how they are named, based on real-time events such as time, month, and day of week. For a list of directives, see “LOGPARM= System Option” in *SAS System Options: Reference*.

See Also

Options:

- “ALTPRINT System Option: UNIX” on page 368
- “PRINTTO Procedure: UNIX” on page 307

Other References:

- “Using SAS System Options to Route Output” on page 102
- “The SAS Log” in Chapter 9 of *SAS Language Reference: Concepts*

ALTPRINT System Option: UNIX

Specifies the destination for the output files from SAS procedures.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	NOALTPRINT
UNIX specifics:	all

Syntax

-ALTPRINT *file-specification* | -NOALTPRINT

Required Arguments

-ALTPRINT *file-specification*
specifies the location for copies of procedure output to be written. The *file-specification* argument can be any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the copy is placed in a file in the specified directory. The name of the file will be *filename.lst*, where *filename* is the name of your SAS job. If you are running SAS interactively and specify only the path to a directory, the output is written to a file named sas.lst.

-NOALTPRINT
causes any previous ALTPRINT specifications to be ignored.

Details

The ALTPRINT system option specifies a destination to which copies of the SAS procedure output file are written. All messages that are written to the SAS procedure output file are also written to the location specified in *file-specification*. You can use this option to capture the procedure output for printing.

See Also

System Options:

- "ALTLOG System Option: UNIX" on page 366

Other References:

- "Using SAS System Options to Route Output" on page 102

APPEND System Option: UNIX

Used when SAS starts; appends the specified value to the existing value at the end of the specified system option.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
UNIX specifics:	configuration file, SAS invocation syntax
Note:	This option cannot be restricted by a site administrator. For more information, see “Restricted Options” in Chapter 1 of *SAS System Options: Reference*.
See:	“APPEND= System Option” in *SAS System Options: Reference*

Syntax

-APPEND *system-option new-option-value*

Required Arguments

system-option
can be FMTSEARCH, HELPLOC, MAPS, MSG, SASAUTOS, SASHELP, SASSCRIPT, SET, AUTOEXEC, or CMPLIB.

new-option-value
is the new value that you want to append to the current value of *system-option*.

Details

By default, if you specify the AUTOEXEC, CMPLIB, FMTSEARCH, HELPLOC, MAPS, MSG, SASAUTOS, SASHELP, SASSCRIPT, or SET system option more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND system option to add the additional pathnames. For example, if you enter the following SAS command, the only location in which SAS looks for help files is **/apps/help**. The output of PROC OPTIONS shows only **/apps/help**:

```
sas -helploc /sas/help -helploc /apps/help
```

If you want SAS to look first in **/sas/help**, and then in **/apps/help**, use the APPEND option.

```
sas -helploc /sas/help -append helploc /apps/help
```

For the value of the HELPLOC option, PROC OPTIONS now shows the following:

```
('/sas/help' '/apps/help')
```

See Also

System Options:

- “APPEND= System Option” in *SAS System Options: Reference*
- “INSERT System Option: UNIX” on page 392

AUTHPROVIDERDOMAIN: UNIX

Associates a domain suffix with an authentication provider.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Initialization and operation
PROC OPTIONS GROUP=	EXECMODES
Alias:	AUTHPD
Default:	NULL
See:	"AUTHPROVIDERDOMAIN System Option" in *SAS System Options: Reference*

AUTOEXEC System Option: UNIX

Specifies the SAS autoexec file.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	autoexec.sas (see "Details" on page 370)
UNIX specifics:	all

Syntax

-AUTOEXEC *file-specification* | -NOAUTOEXEC

-AUTOEXEC \ (*file-specification-1* <...*file-specification-n*>\)

Required Arguments

-NOAUTOEXEC
specifies that SAS is not to process any autoexec files.

file-specification
specifies the SAS autoexec file to be used instead of the default autoexec.sas file. The *file-specification* argument can be a valid Windows filename or an environment variable that is associated with a pathname. For more information, see "SAS Autoexec File " in Chapter 1 of *SAS Companion for Windows*.

Details

The Autoexec File

The AUTOEXEC system option specifies the autoexec file. The autoexec file contains SAS statements that are executed automatically when you invoke SAS, or when you start another SAS process. The autoexec file can contain any SAS statements. For example, your autoexec file can contain LIBNAME statements for SAS libraries that you access routinely in SAS sessions.

SAS looks for the AUTOEXEC system option in the following order. It uses the first AUTOEXEC system option that it finds:

1. in the command line
2. in the SASV9_OPTIONS environment variable
3. in the configuration file

SAS uses the first AUTOEXEC option that it encounters and ignores all others.

If neither the AUTOEXEC nor NOAUTOEXEC system option is found, SAS looks for the autoexec file in three directories in the following order:

1. your current directory
2. your home directory
3. the **!SASROOT** directory (For more information, see "The !SASROOT Directory" on page 443.)

   SAS uses the first autoexec file that it finds to initialize the SAS session.

If you want to see the contents of the autoexec file for your session, use the ECHOAUTO system option when you invoke SAS. If you want to identify the data sources that the autoexec file is using, use the PROC OPTIONS statement:

```
proc options option=autoexec value;
run;
```

Inserting and Appending Autoexec Files

You can concatenate files in your autoexec file by using the following system options with the AUTOEXEC system option: INSERT on page 392 and APPEND on page 368. The autoexec file is always a UNIX file. If your filename contains embedded blanks or special characters, you must enclose the filename in quotation marks. Otherwise, quotation marks are optional when one or more filenames are specified.

You can use the following syntax to concatenate autoexec files:

```
-autoexec "(/path1/autoexec.sas  /path2/autoexec.sas  /path3/autoexec.sas)"
```

You can use the following syntax with the INSERT system option:

```
-insert autoexec "a.sas" -insert autoexec "b.sas"
```

You can use the following syntax with the APPEND system option:

```
-append autoexec "a.sas" -append autoexec "b.sas"
```

See Also

System Options:

- "APPEND System Option: UNIX" on page 368
- "INSERT System Option: UNIX" on page 392

Other References:

- "Customizing Your SAS Session by Using System Options" on page 18

AUTOSAVELOC System Option: UNIX

Specifies the location of the Program Editor autosave file.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	none
UNIX specifics:	valid values of *pathname*
See:	"AUTOSAVELOC= System Option" in *SAS System Options: Reference*

Syntax

-AUTOSAVELOC *fileref* | *pathname*

AUTOSAVELOC *fileref* | *pathname*

Required Arguments

fileref
specifies a fileref to the location where the autosave file is saved.

pathname
specifies the pathname of the autosave file. The *pathname* must be a valid UNIX pathname.

Details

By default, SAS saves the Program Editor autosave file, pgm.asv, in the open folder. You can use the AUTOSAVELOC system option to specify a different location for the autosave file.

See Also

Commands:

- "SETAUTOSAVE Command: UNIX" on page 236

BUFNO System Option: UNIX

Specifies the number of buffers to be allocated for processing a SAS data set.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Files: SAS files
PROC OPTIONS GROUP=	SASFILES, PERFORMANCE

Default:	1
UNIX specifics:	default value
See:	"BUFNO= System Option" in *SAS System Options: Reference*

Syntax

-BUFNO *n* | *n*K | *n*M| *n*G | *hex*X | MIN | MAX

BUFNO=*n* | *n*K | *n*M| *n*G| *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the number of buffers in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 buffers, a value of **.782k** specifies 801 buffers, and a value of **3m** specifies 3,145,728 buffers.

***hex*X**
specifies the number of buffers as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, **2dx** specifies 45 buffers.

MIN
sets the number of buffers to 0, and requires SAS to use the default value of 1.

MAX
sets the number of buffers to 2,147,483,647.

Details

The number of buffers is not a permanent attribute of the data set; it is valid only for the current SAS session or job.

BUFNO= applies to SAS data sets that are opened for input, output, or update.

Using BUFNO= can improve execution time by limiting the number of input/output operations that are required for a particular SAS data set. The improvement in execution time, however, comes at the expense of increased memory consumption.

Under UNIX, the maximum number of buffers that you can allocate is determined by the amount of memory available.

BUFSIZE System Option: UNIX

Specifies the size of a permanent buffer page for an output SAS data set.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Files: SAS files
PROC OPTIONS GROUP=	SASFILES, PERFORMANCE
Default:	0
UNIX specifics:	valid range

See: "BUFSIZE= System Option" in *SAS System Options: Reference*

Syntax

-BUFSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MAX

BUFSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the buffer page size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the buffer page size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, **`2dx`** sets the buffer page size to 45 bytes.

MAX
sets the buffer page size to 2,147,483,647.

Details

The buffer page size can range from 1K to 2G–1.

If you specify a nonzero value when you create a SAS data set, the BASE engine uses that value. If that value cannot hold at least one observation or is not a multiple of 1K, the engine rounds the value up to a multiple of 1K.

CATCACHE System Option: UNIX

Specifies the number of SAS catalogs to keep open.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Files: SAS files
PROC OPTIONS GROUP=	SASFILES
Default:	0
UNIX specifics:	Valid values for *n*
See:	"CATCACHE= System Option" in *SAS System Options: Reference*

Syntax

-CATCACHE *n* | *n*K | MIN | MAX

Required Arguments

***n* | *n*K**
specifies the number of open-file descriptors to keep in cache memory in multiples of 1 (*n*) or 1,024 (*n*K). You can specify decimal values for the number of kilobytes. For example, a value of **`8`** specifies 8 open-file descriptors, a value of **`.782k`** specifies 801 open-file descriptors, and a value of **`3k`** specifies 3,072 open-file descriptors.

If *n* > 0, SAS places up to that number of open-file descriptors in cache memory, instead of closing the catalogs.

MIN
sets the number of open-file descriptors that are kept in cache memory to 0.

MAX
sets the number of open-file descriptors that are kept in cache memory to 32,767.

Details

By using the CATCACHE system option to specify the number of SAS catalogs to keep open, you can avoid repeatedly opening and closing the same catalogs.

See Also

"Definitions for Optimizing System Performance" in Chapter 12 of *SAS Language Reference: Concepts*

CLEANUP System Option: UNIX

Specifies how to handle out-of-resource conditions.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Environment control: Error handling
PROC OPTIONS GROUP=	ERRORHANDLING
Default:	CLEANUP for interactive modes; NOCLEANUP otherwise
UNIX specifics:	behavior when running in interactive line mode and batch mode
See:	"CLEANUP System Option" in *SAS System Options: Reference*

Syntax

-CLEANUP | -NOCLEANUP

CLEANUP | NOCLEANUP

Required Arguments

CLEANUP
specifies that during the entire session, SAS attempts to perform automatic, continuous cleanup of resources that are not essential for execution. Nonessential resources include those resources that are not visible to the user (for example, cache

memory) and those resources that are visible to the user (for example, the Keys window).

CLEANUP does not prompt you before SAS attempts to clean up your disk. However, when an out-of-disk-space condition occurs and your display is attached to the process, you are prompted with a menu selection even if the CLEANUP option is on. If you do not want to be prompted for out-of-disk-space conditions, use the CLEANUP option with the NOTERMINAL option.

When the CLEANUP option is on, SAS performs automatic continuous cleanup. If not enough resources are recovered, the request for the resource fails, and an appropriate error message is written to the SAS log.

CLEANUP is the default in batch mode because there is no display attached to the process to accommodate prompting.

NOCLEANUP
specifies that SAS allows the user to choose how to handle an out-of-resource condition. When NOCLEANUP is in effect and SAS cannot execute because of a lack of resources, SAS automatically attempts to clean up resources that are not visible to the user (for example, cache memory). However, resources that are visible to the user (for example, the Keys window) are not automatically cleaned up. Instead, SAS prompts you before attempting to clean up your disk.

Details

The CLEANUP system option indicates whether you should be prompted with a menu of items to be cleaned up when SAS encounters an out-of-resource condition. In batch mode, SAS ignores this option, and if an out-of-resource condition occurs, the SAS session terminates.

CONFIG System Option: UNIX

Specifies the configuration file that is used when initializing or overriding the values of SAS system options.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable, SASV9_CONFIG environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	sasv9.cfg
UNIX specifics:	all

Syntax

-CONFIG *file-specification* | –NOCONFIG

Required Arguments

-CONFIG *file-specification*
specifies a configuration file to be read. The *file-specification* must resolve to a valid UNIX filename.

-NOCONFIG
specifies that any previous CONFIG specification should be ignored and that the default system options should be used.

Details

Configuration files contain system option specifications that execute automatically whenever SAS is invoked.

Specifying a configuration file disables the default configuration file list.

See Also

"Customizing Your SAS Session by Using System Options" on page 18

DEVICE System Option: UNIX

Specifies a device driver for graphics output for SAS/GRAPH software.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable, GOPTIONS statement
Category:	Graphics: Driver settings
PROC OPTIONS GROUP=	GRAPHICS
Default:	none
UNIX specifics:	valid device drivers
See:	"DEVICE= System Option" in *SAS System Options: Reference*

Syntax

-DEVICE *device-driver-name*

DEVICE=*device-driver-name*

Required Argument

device-driver-name
specifies the name of a device driver for graphics output.

Details

To see the list of device drivers that are available under UNIX, you can use the GDEVICE procedure. If you are using the SAS windowing environment, submit the following statements:

```
proc gdevice catalog=sashelp.devices;
run;
```

If you are running SAS in interactive line mode or batch mode, submit the following statements:

```
proc gdevice catalog=sashelp.devices nofs;
   list _all_;
run;
```

See Also

SAS/GRAPH: Reference

ECHO System Option: UNIX

Specifies a message to be echoed to the computer.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	none
UNIX specifics:	all

Syntax

-ECHO "*message*" | -NOECHO

Required Arguments

-ECHO "*message*"
specifies the text of the message to be echoed to the computer. The text must be enclosed in single or double quotation marks if the message is more than one word. Otherwise, the quotation marks are not needed.

-NOECHO
specifies that no messages are to be echoed to the computer.

Details

Messages that result from errors in the autoexec file are printed in the SAS log regardless of how the ECHO system option is set.

You can specify multiple ECHO options. The strings are displayed in the order in which SAS encounters them. See "How SAS Processes System Options Set in Multiple Places" on page 20 for information about how that order is determined.

For example, you can specify the following:

```
-echo "SAS 9.3 under UNIX is initializing."
```

The message appears in the Log window as SAS initializes.

See Also

System Options:

- "ECHOAUTO System Option" in *SAS System Options: Reference*

EDITCMD System Option: UNIX

Specifies the host editor to be used with the HOSTEDIT command.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Environment control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	none
UNIX specifics:	all

Syntax

-EDITCMD "*host-editor-pathname editor-options*"

EDITCMD="*host-editor-pathname editor-options*"

Details

The EDITCMD system option specifies the command that is issued to the operating environment. If you are using a terminal-based editor, such as vi, you must specify a command that runs the editor inside a terminal emulator window.

You can define the EDITCMD option using the SASV9_OPTIONS environment variable as part of a configuration file or on the command line to make the definition available automatically to SAS. The option must be specified as a string in quotation marks. You can use either single or double quotation marks. You can change the value for the EDITCMD option during a SAS session by issuing an OPTIONS statement.

The host editor that you specify is used when you issue the HOSTEDIT command. The HOSTEDIT command is valid only when you are running SAS in a windowing environment.

If you do not specify the full pathname, SAS searches the pathnames specified in the $PATH environment variable. For example, to use vi, you would specify the following:

```
sas -editcmd "/usr/bin/X11/xterm -e /usr/bin/vi"
```

See Also

"Configuring SAS for Host Editor Support in UNIX Environments" on page 160

EMAILSYS System Option: UNIX

Specifies the e-mail protocol to use for sending electronic mail.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Communications: E-mail
PROC OPTIONS GROUP=	EMAIL
Default:	SMTP
UNIX specifics:	all

Syntax

-EMAILSYS SMTP | *name-of-script*

EMAILSYS=SMTP | *name-of-script*

Required Arguments

SMTP
specifies the Simple Mail Transfer Protocol (SMTP) electronic mail interface.

name-of-script
specifies which script to use for sending electronic mail from within SAS. Some external scripts do not support sending e-mail attachments. These scripts are not supported by SAS.

Details

The EMAILSYS system option specifies which e-mail protocol to use for sending electronic mail from within SAS. Specifying SMTP supports sending e-mail attachments on UNIX, but might require changing the values of the EMAILHOST= and EMAILPORT= system options, depending on your site configuration.

You can set the EMAILSYS option at any time in your SAS session.

See Also

System Options:

- “EMAILHOST= System Option” in *SAS System Options: Reference*
- “EMAILPORT System Option” in *SAS System Options: Reference*

Other References:

- “Sending Mail from within Your SAS Session in UNIX Environments” on page 158
- “Sending Electronic Mail Using the FILENAME Statement (EMAIL)” on page 84
- “Sending E-Mail through SMTP” in Chapter 38 of *SAS Language Reference: Concepts*

ENGINE= System Option: UNIX

Specifies the default access method to use for SAS libraries.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Files: SAS files
PROC OPTIONS GROUP=	SASFILES
Default:	V9
UNIX specifics:	valid values of *engine-name*
See:	“ENGINE= System Option” in *SAS System Options: Reference*

Syntax

-ENGINE *engine-name*

Required Argument

engine-name
can be one of the following under UNIX:

BASE | V9
specifies the default SAS engine for SAS 9 through SAS 9.3 files.

V8
specifies the SAS engine for all SAS Version 9 files.

V7
specifies the SAS engine for all Version 7 files.

V6
specifies the SAS engine for Release 6.09 through Release 6.12. This engine is read-only.

See Also

- "Compatible Computer Types in UNIX Environments" on page 44
- Chapter 35, "SAS Engines," in *SAS Language Reference: Concepts*

FILELOCKS System Option: UNIX

Specifies whether file locking is turned on or off and what action should be taken if a file cannot be locked.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Files: External files | SAS files
PROC OPTIONS GROUP=	EXTFILES, SASFILES, ENVFILES
Default:	FAIL
UNIX specifics:	all

Syntax

-FILELOCKS *setting path* | *path setting*

-FILELOCKS NONE | FAIL | CONTINUE | RESET

FILELOCKS=(*setting path* | *path setting*)

FILELOCKS=NONE | FAIL | CONTINUE | RESET

Required Arguments

setting
specifies the operating environment locking value for the specified path. The following values are valid:

- NONE
- FAIL
- CONTINUE
- RESET

path
specifies a path to a UNIX directory. Enclose the path in single or double quotation marks.

Tip: The *path* argument can contain an environment variable.

NONE
turns file locking off. NONE specifies that SAS attempts to open the file without checking for an existing lock on the file. NONE does not place an operating system lock on the file. These files are not protected from shared Update access.

Tip: NONE does not suppress internal locking.

FAIL
turns file locking on. FAIL specifies that SAS attempts to place an operating system lock on the file. Access to the file is denied if the file is already locked, or if it cannot be locked. FAIL is the default value for FILELOCKS.

CONTINUE
turns file locking on. CONTINUE specifies that SAS attempts to place an operating system lock on the file. If a file is already locked by someone else, an attempt to open it fails. If the file cannot be locked for some other reason (for example, if the file system does not support locking), the file is opened and a warning message is sent to the log.

Tip: CONTINUE does not suppress internal locking.

RESET
specifies that all previous FILELOCKS settings will be deleted, and resets the global setting to the default value of FAIL. If you use the **`FILELOCKS=(`*`setting path`*`|`*`path setting`*`)`** syntax, then RESET resets only those files that are in *path*.

Details

The Basics of File Locking

In previous releases of SAS, the FILELOCKS system option was able to lock only SAS files. In SAS 9.2 and later, the FILELOCKS system option is able to lock external files as well.

The FILELOCKS system option enables you to lock both external files and SAS files based on global settings that you set in the FILELOCKS system option. External file locking applies to all files that are opened.

You can use multiple instances of the FILELOCKS option to establish different settings for different paths. One path can be a subdirectory of another path. In this case, the most specific matching path currently in effect governs operating system file locking. The following example shows how you can specify multiple instances of the FILELOCKS option in a configuration file:

```
filelocks = ('/u/myuserid/temp' NONE)
filelocks = ('/tmp' CONTINUE)
```

When the value of the FILELOCKS option is a set of *path* and *setting*, the path must be enclosed in quotation marks. If you use FILELOCKS on the command line, then quotation marks are not needed.

Note: To prevent data corruption, setting FILELOCKS to NONE or CONTINUE is not recommended.

Resetting Paths by Using the path and setting Arguments

The *path* and *setting* arguments enable you to apply a setting to a particular directory and its subtrees. If you set the value of *setting* to RESET, then the *path* and *setting* values are deleted.

For example, in the following case, **`filelocks=('/' reset)`**, the current values for *path* and *setting* are deleted, and FILELOCKS resets the values to the following default: **`('/' fail)`**.

When FILELOCKS Is Set to FAIL

When FILELOCKS is set to FAIL (the default value), the following actions occur:

- SAS prevents two sessions from simultaneously opening the same SAS file for update or output.
- SAS prevents one session from reading a SAS file that another SAS session has open for update or output.
- SAS prevents one session from writing to a file that another SAS session has open in Read mode.

See Also

System Options:

- "WORKINIT System Option: UNIX" on page 436

FILELOCKWAITMAX= System Option: UNIX

Sets an upper limit on the time SAS will wait for a locked file.

Valid in:	configuration file, SAS invocation
Category:	Files: SAS files
PROC OPTIONS GROUP=	SASFILES
Default:	600
UNIX specifics:	all

Syntax

FILELOCKWAITMAX = *wait-time*

Required Argument

wait-time
specifies the amount of time, in seconds, that SAS will wait for a locked file to become available.

Default: 600

Range: 0–600

Interactions:
Specifying the FILELOCKWAITMAX= system option can have an adverse effect on one or more SAS/SHARE server and client sessions that are waiting for the release of a SAS file that is locked by another process. One or more wait conditions could lead to failed processes for a SAS/SHARE server and clients.

To prevent the possibility of a failed SAS/SHARE process, you can set FILELOCKWAITMAX=0. Setting this option cancels the amount of time that a SAS/SHARE server and clients would wait for the release of a locked file. Canceling the wait time would prevent a failed process.

Details

The FILELOCKWAITMAX= system option enables you to limit or turn off the amount of time SAS will wait for a locked file. SAS uses the FILELOCKWAIT= LIBNAME option to wait for the file to become available. Using the FILELOCKWAITMAX= system option, an administrator can limit or turn off this behavior. Normally, SAS returns an error if the file that it attempts to access is locked. If you set FILELOCKWAITMAX= to 0, SAS fails immediately upon encountering a locked file. This option is used primarily by a system administrator.

See Also

System Options:

- “FILELOCKS System Option: UNIX” on page 381

FMTSEARCH System Option: UNIX

Specifies the order in which format catalogs are searched.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	(WORK LIBRARY)
UNIX specifics:	valid values for *catalog-specification*
See:	“FMTSEARCH= System Option” in *SAS System Options: Reference*

Syntax

-FMTSEARCH *(catalog-specification-1... catalog-specification-n)*

FMTSEARCH=*(catalog-specification-1... catalog-specification-n)*

Required Argument

catalog-specification
specifies the order in which format catalogs are searched until the desired member is found. The value of *libref* can be either *libref* or *libref.catalog*. If only the libref is given, SAS assumes that FORMATS is the catalog name.

Note: The value of *libref* must be in uppercase characters.

Details

To add additional *catalog-specification* entries, use the "INSERT System Option: UNIX" on page 392 or "APPEND System Option: UNIX" on page 368.

See Also

System Options:

- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

FONTSLOC System Option: UNIX

Specifies the location of the SAS fonts that are loaded during the SAS session.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	!SASROOT/misc/fonts
UNIX specifics:	valid pathname
See:	"FONTSLOC= System Option" in *SAS System Options: Reference*

Syntax

-FONTSLOC "*directory-specification*"

Required Argument

"*directory-specification*"
specifies the directory that contains the SAS fonts that are loaded during the SAS session. The *directory-specification* must be enclosed in double quotation marks.

Details

The directory must be a valid operating environment pathname.

FULLSTIMER System Option: UNIX

Specifies whether to write all available system performance statistics and the datetime stamp to the SAS log.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOFULLSTIMER
UNIX specifics:	all

Syntax

-FULLSTIMER | -NOFULLSTIMER

FULLSTIMER | NOFULLSTIMER

Required Arguments

FULLSTIMER
writes to the SAS log a list of the host-dependent resources that were used for each step and for the entire SAS session. A datetime stamp is included in the output.

NOFULLSTIMER
does not write to the SAS log a complete list of resources or a datetime stamp.

Details

SAS uses UNIX system calls for your operating environment to get the statistical information from FULLSTIMER. The datetime stamp is listed in the output. You can change the behavior and format of the statistical information by using the STIMFMT system option.

The following is an example of FULLSTIMER output:

Log 18.1 *FULLSTIMER Output*

```
NOTE: SAS initialization used:
            real time           0.84 seconds
            user cpu time       0.03 seconds
            system cpu time     0.03 seconds
            Memory                            236k
            OS Memory                         5672k
            Timestamp            3/16/2011  9:13:39 AM
            Page Faults                       37
            Page Reclaims                     0
            Page Swaps                        0
            Voluntary Context Switches        1336
            Involuntary Context Switches      1
            Block Input Operations            39
            Block Output Operations           0
```

Note: If both FULLSTIMER and STIMER system options are set, the FULLSTIMER statistics are written to the log.

FULLSTIMER displays the following statistics:

Table 18.2 *Description of FULLSTIMER Statistics*

Statistic	**Description**
Real Time	the amount of real time (clock time) that is spent to process the SAS job. Real time is also referred to as elapsed time.
User CPU Time	the CPU time that is spent in the user program.
System CPU Time	the CPU time that is spent to perform operating system tasks (system overhead tasks) that support the execution of your SAS code.
Memory	the amount of memory required to run a step.
OS Memory	the largest amount of operating system memory that is available to SAS during the step.
Timestamp	the date and time that a step was executed.
Page Faults	the number of pages that SAS tried to access but were not in main memory and required I/O activity.
Page Reclaims	the number of pages that were accessed without I/O activity.
Page Swaps	the number of times a process was swapped out of main memory.
Voluntary Context Switches	the number of times that the SAS process had to pause because of a resource constraint such as a disk drive.
Involuntary Context Switches	the number of times that the operating system forced the SAS session to pause processing to allow other process to run.
Block Input Operations	the number of I/O operations that are performed to read the data into memory.
Block Output Operations	the number of I/O operations that are performed to write the data to a file.

For more information about these statistics, see the man pages for the **`getrusage()`** and **`times()`** UNIX system calls.

Note: Starting in SAS 9, some procedures use multiple threads. On computers with multiple CPUs, the operating system can run more than one thread simultaneously. Consequently, CPU time might exceed real time in your FULLSTIMER output. For example, a SAS procedure could use two threads that run on two separate CPUs simultaneously. The value of CPU time would be calculated as the following:

```
CPU1 time + CPU2 time = total CPU time
1 second + 1 second = 2 seconds
```

Because CPU1 can run a thread at the same time that CPU2 runs a separate thread for the same SAS process, you can theoretically consume 2 CPU seconds in 1 second of real time.

See Also

System Options:

- "STIMER System Option: UNIX" on page 430
- "STIMEFMT System Option: UNIX" on page 427

HELPHOST System Option: UNIX

Specifies the name of the host computer where the remote browsing system is to be displayed.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, or SASV9_OPTIONS environment variable
Category:	Environment control: Help
PROC OPTIONS GROUP=	HELP
Default:	NULL
UNIX specifics:	all
See:	"HELPHOST System Option" in *SAS System Options: Reference*

Syntax

HELPHOST="*host*"

-HELPHOST "*host*"

Required Argument

"host"

specifies the name of the computer where the remote browsing system is to be displayed. Quotation marks or parentheses are required. The maximum number of characters is 2048.

Details

If you do not specify the HELPHOST option, the remote browsing system is displayed on the host that is specified in the X display setting.

Examples

Example 1: SAS Invocation

The syntax for specifying the HELPHOST system option for UNIX environments is shown in the following example:

```
sas93/helphost "my.computer.com"
```

Example 2: OPTIONS Statement: UNIX

The syntax for specifying the HELPHOST system option using the OPTIONS statement is shown in the following example:

```
options helphost="my.computer.com";
```

See Also

"Installing the Remote Browser Server" on page 134

HELPINDEX System Option: UNIX

Specifies one or more index files for the online SAS Help and Documentation.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Help
PROC OPTIONS GROUP=	HELP
Default:	/help/common.hlp/index.txt, /help/common.hlp/keywords.htm, common.hhk
UNIX specifics:	applet and HTML files must reside in the path specified by the HELPLOC option

Syntax

-HELPINDEX *index-pathname-1* < *index-pathname-2*< *index-pathname-3*>>

Required Arguments

index-pathname

specifies the partial pathname for the index that is to be used by the online SAS Help and Documentation. The *index-pathname* argument can be any or all of the following:

/help/*applet-index-filename*

specifies the partial pathname of the index file that is to be used by the SAS Documentation Java applet in a UNIX environment. *applet-index-filename* must have a file extension of .txt, and it must reside in a path that is specified by the HELPLOC system option. The default is **`/help/common.hlp/index.txt`**.

See the default index file for the format that is required for an index file.

/help/*accessible-index-filename*

specifies the partial pathname of an accessible index file that is to be used by the online SAS Help and Documentation in UNIX or z/OS environments. An accessible index file is an HTML file that can be used by Web browsers. *accessible-index-filename* must have a file extension of .htm, and it must reside in a path that is specified by the HELPLOC system option. The default pathname is **`/help/common.hlp/keywords.htm`**.

See the default index file for the format that is required for an index file.

HTML-Help-index-pathname

specifies the pathname of the Microsoft HTML Help index that is to be used by the online SAS Help and Documentation in Windows environments. The default pathname is **`common.hhk`**. For information about creating an index for Microsoft HTML Help, see your Microsoft HTML Help documentation.

Details

Use the HELPINDEX option if you have a customized index that you want to use instead of the index that SAS supplies. If you use one configuration file to start SAS in more than one operating environment, you can specify all of the partial pathnames in the HELPINDEX option. The order of the pathnames is not important, although only one pathname of each type can be specified.

When the HELPINDEX option specifies a pathname for UNIX or z/OS operating environments, SAS determines the complete path by replacing **`/help/`** in the partial pathname with the pathname specified in the HELPLOC option. If the HELPLOC option contains more than one pathname, SAS searches each path for the specified index.

For example, when the value of HELPINDEX is **`/help/common.hlp/myindex.htm`** and the value of HELPLOC is **`/u/myhome/myhelp`**, the complete path to the index is **`/u/myhome/myhelp/common.hlp/myindex.htm`**.

See Also

System Options:

- “HELPLOC System Option: UNIX” on page 390

HELPLOC System Option: UNIX

Specifies the location of the text and index files for the facility that is used to view the online SAS Help and Documentation.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Help
PROC OPTIONS GROUP=	HELP
Default:	!SASROOT/X11/native_help/en
UNIX specifics:	default *pathname*

Syntax

-HELPLOC (*pathname*<,*pathname-2*...,*pathname-n*>)

Required Argument

pathname
specifies one or more directory pathnames in which the online SAS Help and Documentation files are located.

Details

Specifying a value for the HELPLOC system option causes SAS to insert that value at the start of a list of concatenated values, the last of which is the default value. This behavior enables you to access help for your site without losing access to SAS Help and Documentation.

To add pathnames, use the INSERT or APPEND system options. For more information, see "INSERT System Option: UNIX" on page 392, and "APPEND System Option: UNIX" on page 368.

Example: Using the HELPLOC System Option

The following command contains two specifications of HELPLOC:

```
sas -insert helploc /app2/help -insert helploc /app1/help -append
```

The value of the system option is the following:

```
/app1/help, /app2/help, !SASROOT/X11/native_help
```

See Also

System Options:

- "INSERT= System Option" in *SAS System Options: Reference*
- "APPEND= System Option" in *SAS System Options: Reference*

HELPTOC System Option: UNIX

Specifies the table of contents files for the online SAS Help and Documentation.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Help
PROC OPTIONS GROUP=	HELP
Default:	/help/helpnav.hlp/config.txt, /help/common.hlp/toc.htm, common.hhc
UNIX specifics:	applet and HTML files must reside in the path specified by the HELPLOC option

Syntax

-HELPTOC *TOC-pathname-1 <TOC-pathname-2< TOC-pathname-3>>*

Required Argument

TOC-pathname

specifies a partial pathname for the table of contents that is to be used by the online SAS Help and Documentation. *TOC-pathname* can be any or all of the following:

/help/*applet-TOC-filename*

specifies the partial pathname of the table of contents file that is to be used by the SAS Documentation Java applet in a UNIX environment. The *applet-TOC-filename* must have a file extension of .txt, and it must reside in a path that is specified by the HELPLOC system option. The default is **`/help/helpnav.hlp/config.txt`**.

See the default table of contents file for the format that is required for an index file.

/help/*accessible-TOC-filename*
: specifies the partial pathname of an accessible table of contents file that is to be used by the online SAS Help and Documentation in UNIX or z/OS environments. An accessible table of contents file is an HTML file that can be used by Web browsers. The *accessible-TOC-filename* must have a file extension of .htm, and it must reside in a path that is specified by the HELPLOC system option. The default pathname is **`/help/common.hlp/toc.htm`**.

  See the default table of contents file for the format that is required for a table of contents.

HTML-Help-TOC-pathname
: specifies the complete pathname to the Microsoft HTML Help table of contents that is to be used by the online SAS Help and Documentation in Windows environments. The default pathname is **`common.hhc`**. For information about creating an index for Microsoft HTML Help, see your Microsoft HTML Help documentation.

Details

Use the HELPTOC system option if you have a customized table of contents that you want to use, instead of the table of contents that SAS provides. If you use one configuration file to start SAS in more than one operating environment, you can specify all of the partial pathnames in the HELPTOC option. The order of the pathnames is not important, although only one pathname of each type can be specified.

When the HELPTOC option specifies the pathname for UNIX or z/OS operating environments, SAS determines the complete path by replacing **`/help/`** in the partial pathname with the pathname specified in the HELPLOC option. If the HELPLOC option contains more than one pathname, SAS searches each path for the table of contents.

For example, when HELPTOC is **`/help/common.hlp/mytoc.htm`**, and the value of HELPLOC is **`/u/myhome/myhelp`**, the complete path to the table of contents is **`/u/myhome/myhelp/common.hlp/mytoc.htm`**.

See Also

System Options:

- "HELPLOC System Option: UNIX" on page 390

INSERT System Option: UNIX

Used when SAS starts; inserts the specified value at the beginning of the specified system option.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
UNIX specifics:	configuration file, SAS invocation syntax
Note:	This option cannot be restricted by a site administrator. For more information, see "Restricted Options" in Chapter 1 of *SAS System Options: Reference*.

See: "INSERT= System Option" in *SAS System Options: Reference*

Syntax

-INSERT *system-option new-option-value*

Required Arguments

system-option
can be FMTSEARCH, HELPLOC, MAPS, MSG, SASAUTOS, SASHELP, SASSCRIPT, SET, AUTOEXEC, or CMPLIB

new-option-value
is the new value that you want to insert at the beginning of the current value of *system-option*.

Details

By default, if you specify the AUTOEXEC, CMPLIB, FMTSEARCH, HELPLOC, MAPS, MSG, SASAUTOS, SASHELP, SASSCRIPT, or SET system option more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, use the INSERT system option to add the additional pathnames. For example, if you enter the following SAS command, the only location in which SAS looks for help files is **`/apps/help`**. The output of PROC OPTIONS shows only **`/apps/help`**.

```
sas -helploc /apps/help
```

If you want SAS to look in both the current path for help files, and in **`/sas/help`**, and if you want SAS to look first in **`/apps/help`**, then you must use the INSERT option.

```
sas -insert helploc /apps/help
```

If the current path for help files is **`!SASROOT/X11/native_help`**, then PROC OPTIONS now shows the following for the value of the HELPLOC option:

```
('/apps/help' '!SASROOT/X11/native_help')
```

See Also

System Options:

- "APPEND System Option: UNIX" on page 368
- "APPEND= System Option" in *SAS System Options: Reference*

JREOPTIONS System Option: UNIX

Identifies the Java Runtime Environment (JRE) options for SAS.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Initialization and operation
PROC OPTIONS GROUP=	EXECMODES
Default:	none

UNIX specifics:	all

Syntax

-JREOPTIONS (-*JRE-option-1*<-*JRE-option-n*>)

Required Argument

-JRE-option
specifies one or more JRE options.

JRE options must begin with a hyphen (-). Use a space to separate multiple JRE options. Valid values for *JRE-option* depend on your installation's JRE. For information about JRE options, see your installation's Java documentation.

Details

JRE options must be enclosed in parentheses. If you issue JREOPTIONS options on the command line, then you must put a backslash (\) before the open parenthesis and close parenthesis, as shown in the examples below. If you specify multiple JREOPTIONS options, then SAS appends JRE options to JRE options that are currently defined. Incorrect JRE options are ignored.

Example: Using JRE Options

```
-jreoptions \(-verbose\)
```

```
-jreoptions \(-Djava.class.path=myjava/classes/myclasses.jar:myjava2/
            classes/myclasses.jar -oss600k\)
```

LINESIZE System Option: UNIX

Specifies the line size of the SAS Log and Output windows.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log and procedure output
PROC OPTIONS GROUP=	LOG_LISTCONTROL, LOGCONTROL
Default:	the display width setting for the interactive modes; 132 for batch mode
UNIX specifics:	default values
See:	"LINESIZE= System Option" in *SAS System Options: Reference*

Syntax

-LINESIZE *n* | *hex*X | MIN | MAX

LINESIZE=*n* | *hex*X | MIN | MAX

Required Arguments

n
specifies the line size in characters. Valid values range between 64 and 256.

***hex*X**
specifies the line size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, `2dx` specifies 45 characters.

MIN
sets the line size to 64 characters.

MAX
sets the line size to 256 characters.

See Also

"Controlling the Content and Appearance of Output in UNIX Environments" on page 104

LOG System Option: UNIX

Specifies a destination for the SAS log when running in batch mode.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES, LOGCONTROL
Default:	a file in the current directory with the same filename as the SAS source file and an extension of .log
UNIX specifics:	all

Syntax

-LOG *file-specification* | -NOLOG

Required Arguments

-LOG *file-specification*
specifies the destination for the SAS log. The *file-specification* can be any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the log file is created in the specified directory. The default name for this file is *filename*.log, where *filename* is the name of your SAS job.

-NOLOG
suppresses the creation of the SAS log. Do not use this value unless your SAS program is thoroughly debugged.

Details

The LOG system option specifies a destination for the SAS log when running in batch mode. The LOG system option is valid in batch mode; it is ignored in interactive modes.

Using directives in the value of the LOG system option enables you to control when logs are open and closed and how they are named, based on real-time events such as time, month, day of week, and so on. For a valid list of directives, see "LOGPARM= System Option" in *SAS System Options: Reference*.

If you start SAS in batch mode or server mode and the LOGCONFIGLOC= option is specified, logging is performed by the SAS logging facility. The traditional SAS log option LOGPARM= is ignored. The traditional SAS log option LOG= is honored only when the %S{App.Log} conversion character is specified in the logging configuration file. For more information, see the SAS Logging Facility in *SAS Logging: Configuration and Programming Reference*.

Note: When SAS is started with the OBJECTSERVER and NOTERMINAL system options and no log is specified, SAS discards all log and alternate log messages.

See Also

System Options:

- "LOGPARM= System Option" in *SAS System Options: Reference*

Other References:

- "The SAS Log" in Chapter 9 of *SAS Language Reference: Concepts*
- "Using SAS System Options to Route Output" on page 102

LPTYPE System Option: UNIX

Specifies which UNIX command and options settings will be used to route files to the printer.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: Procedure output
PROC OPTIONS GROUP=	LISTCONTROL
Default:	none
UNIX specifics:	all

Syntax

-LPTYPE BSD | SYSV

LPTYPE=BSD | SYSV

Required Arguments

-LPTYPE BSD
causes SAS to use the `lpr` command to send files to the printer. The `lpr` command is usually supported on UNIX operating systems, such as HP-UX, that were developed at the University of California, Berkeley.

-LPTYPE SYSV
causes SAS to use the `lp` command to send files to the printer. The `lp` command is usually supported on operating systems derived from UNIX System V, such as Solaris.

Details

The LPTYPE option determines whether SAS is to use the `lpr` or the `lp` UNIX command to print files.

If you do not know whether to specify BSD or SYSV, check with your system administrator.

By default, SAS uses the `lpr` command if your operating system is derived from Berkeley's version. Otherwise, it uses the `lp` command.

See Also

System Options:

- "PRINTCMD System Option: UNIX" on page 409

MAPS System Option: UNIX

Specifies the name of the SAS library containing the SAS/GRAPH map data sets.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Graphics: Driver settings
PROC OPTIONS GROUP=	GRAPHICS
Default:	!SASROOT/maps (set in the installed !SASROOT/sasv9.cfg file)
UNIX specifics:	default value and *location-of-maps*
See:	"MAPS= System Option" in *SAS System Options: Reference*

Syntax

-MAPS *location-of-maps*

MAPS=*location-of-maps*

Required Argument

location-of-maps
specifies a libref, a valid UNIX pathname, or an environment variable associated with a pathname. Do not use a specific filename.

Details

The Basics

You can reassign the MAPS libref, but you cannot clear it.

Map files might have to be uncompressed before they are used. Use the CONTENTS statement in the DATASETS procedure to determine whether they are compressed.

Inserting and Appending Pathnames

By default, if you specify the MAPS system option more than one time, the last option that is specified is the option value that SAS uses.

If you want to add additional pathnames to the pathnames already specified by the MAPS system option, use the INSERT system option to add the additional pathnames. For example, if you enter the following SAS command, the only location in which SAS looks for help files is **`/apps/help`**. The output of PROC OPTIONS shows only **`/apps/help`**.

```
sas -helploc /apps/help
```

If you want SAS to look in both the current path for help files, and in **`/sas/help`**, and if you want SAS to look first in **`/apps/help`**, then you must use the INSERT option.

```
sas -insert helploc /apps/help
```

If you want SAS to look first in **`/sas/help`**, and then in **`/apps/help`**, then you must use the APPEND option.

```
sas -helploc /sas/help -append helploc /apps/help
```

If the current path for help files is **`!SASROOT/X11/native_help`**, then PROC OPTIONS now shows the following for the value of the HELPLOC option:

```
('/apps/help' '!SASROOT/X11/native_help')
```

See Also

System Options:

- "INSERT= System Option" in *SAS System Options: Reference*
- "APPEND= System Option" in *SAS System Options: Reference*

MAXMEMQUERY System Option: UNIX

Specifies the maximum amount of memory that can be allocated per request for certain procedures.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	System administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	256M
UNIX specifics:	all

Syntax

-MAXMEMQUERY *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

MAXMEMQUERY=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the limit in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of `8` specifies 8 bytes, a value of `.782k` specifies 801 bytes, and a value of `3m` specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, `2dx` sets the amount of memory to 45 bytes.

MIN
specifies 0 bytes, which indicates that there is no limit on the total amount of memory that can be allocated per request by each SAS procedure. These memory allocations are limited by the value of MEMSIZE.

MAX
specifies a limit to the amount of memory that is allocated. Memory allocations (9,007,199,254,740,992 byte limit on 64-bit machines) are limited by the value of MEMSIZE.

Details

Some SAS procedures use the MAXMEMQUERY option to specify the largest block of virtual memory that a procedure can request at one time. By contrast, the MEMSIZE option places a limit on the total amount of virtual memory that SAS dynamically allocates at any time. This virtual memory is supported by a combination of real memory and paging space. The operating environment begins paging when the amount of virtual memory that is required exceeds the real memory that is available. To prevent paging and the associated performance problems, the MAXMEMQUERY and MEMSIZE system options should be set to a subset of real memory.

MEMSIZE System Option: UNIX

Specifies the limit on the total amount of virtual memory that can be used by a SAS session.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	System administration: Memory
PROC OPTIONS GROUP=	MEMORY, PERFORMANCE
Default:	512M
UNIX specifics:	all

Syntax

-MEMSIZE *n* | *n*K | *n*M | *n*G | *n*T | *hex*X | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G | *n*T**
specifies the limit in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); 1,073,741,824 (gigabytes); or 1,099,511,627,776 (terabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of .25G specifies 268,435,456 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, `0F00000x` sets the value of the MEMSIZE option to 15,728,640 bytes. A value of `0x` is equivalent to using the MAX value.

MAX
specifies to set the memory size to the largest reasonable value depending on the amounts of physical memory and paging space that are available at the time that SAS is started.

Details

The Basics

The MEMSIZE system option limits the total amount of memory that is available to each SAS session. It places an enforced limit on the amount of virtual memory that SAS can dynamically allocate at execution. If MEMSIZE is set too low, your jobs will fail, and an error will appear in the SAS log indicating that insufficient memory was available. By contrast, the REALMEMSIZE and MAXMEMQUERY system options, the SORTSIZE= option in the SORT procedure, and the SUMSIZE= option in the SUMMARY procedure all provide for procedure tuning.

If you specify an unreasonably small numeric value for MEMSIZE (for example, 6K), then the setting automatically increases to a minimum reasonable value that will enable SAS to start. If you specify a numeric value in excess of 4,294,967,295 on a 32-bit version of SAS, then the setting automatically decreases to 4,294,967,295.

Numeric values in excess of 9,223,372,036,854,775,807 bytes will be rejected as invalid, and will prevent SAS from starting.

SAS does not automatically reserve or allocate the amount of memory that you specify in the MEMSIZE system option. SAS uses only as much memory as it needs to complete a process. For example, a DATA step might require only 20 MB of memory, so even though MEMSIZE is set to 500 MB, SAS will use only 20 MB of memory. While your SAS jobs are running, you can monitor the effects of larger memory settings by using system monitoring tools, such as VMSTAT and top. With some tools, address space might be allocated to memory, but pages might not be assigned to that memory. These tools will report a higher value than real memory actually used.

Setting the Size of MEMSIZE

Setting MEMSIZE=MAX sets MEMSIZE to 80% of physical memory. Setting MEMSIZE to MAX is the same as setting MEMSIZE to 0. Setting MEMSIZE to MAX is reasonable only if no processes that consume large amounts of memory are likely to become active after SAS has started. For example, if multiple instances of SAS are running concurrently, and all of the sessions were started with a MEMSIZE value of MAX, then one or more of these sessions can encounter out-of-memory conditions, or the operating system can run out of available paging space. MEMSIZE=MAX calculates a value that would help prevent the system from paging if all of the memory were allocated.

The optimal setting for this option depends on the other applications that are running and the system resources available at your site. The amount of memory available to SAS processes can also be limited by your system administrator.

If you set MEMSIZE to the maximum amount of memory that is reasonably attainable, some procedures scale themselves to the available memory. To determine the limit on the total amount of memory to be used by SAS, you can issue a PROC OPTIONS statement:

```
proc options option=MEMSIZE;
run;
```

Setting MEMSIZE to 0 is used as a test that can determine a good value to set for MEMSIZE.

To determine the optimal setting of MEMSIZE, execute a SAS procedure or DATA step with the FULLSTIMER option and MEMSIZE set to 0. Note the amount of memory that is used by the process, and then set MEMSIZE to a larger amount.

Comparisons

Some SAS procedures use the REALMEMSIZE system option to specify how much real memory the procedure can allocate and use without inducing excessive page swapping. By contrast, the MEMSIZE system option places a limit on the total amount of virtual memory that SAS dynamically allocates at any time. This virtual memory is supported by a combination of real memory and paging space.

The operating environment begins paging when the amount of virtual memory that is required exceeds the real memory that is available. To prevent paging and the associated performance problems, the REALMEMSIZE and MEMSIZE system options should be set to a subset of real memory.

See Also

System Options:

- "REALMEMSIZE System Option: UNIX" on page 410

Procedures:

- "SORT Procedure: UNIX" on page 309

MSG System Option: UNIX

Specifies the library that contains the SAS error messages.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Alias:	SASMSG
Default:	!SASROOT/sasmsg (set in the installed !SASROOT/sasv9.cfg file)

Syntax

-MSG *pathname*

-MSG ('*pathname*' '*pathname*' ...)

Required Argument

pathname
must resolve to a valid UNIX pathname. You can use an environment variable that resolves to a valid pathname.

Details

The MSG system option specifies the library that contains the SAS error messages. This option is set during the installation process and is not normally changed after installation.

To add additional pathnames, use the INSERT or APPEND system options. For more information, see "INSERT System Option: UNIX" on page 392, and "APPEND System Option: UNIX" on page 368.

See Also

System Options:

- "INSERT= System Option" in *SAS System Options: Reference*
- "APPEND= System Option" in *SAS System Options: Reference*

MSGCASE System Option: UNIX

Specifies whether notes, warnings, and error messages that are generated by SAS are displayed in uppercase characters.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOMSGCASE
UNIX specifics:	all

Syntax

-MSGCASE | -NOMSGCASE

Required Arguments

-MSGCASE
displays notes, warnings, and error messages in uppercase characters.

-NOMSGCASE
displays notes, warnings, and error messages in uppercase and lowercase characters.

Details

The MSGCASE system option specifies whether notes, warnings, and error messages that are generated by SAS are displayed in uppercase characters. User-generated messages and source lines are not affected by the MSGCASE system option.

MSGCASE is supported in NL formats. For information about NL formats, see *SAS National Language Support (NLS): Reference Guide*.

MSYMTABMAX System Option: UNIX

Specifies the maximum amount of memory available to the macro variable symbol tables.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Macro: SAS macro
PROC OPTIONS GROUP=	MACRO
Default:	4M (set in the installed !SASROOT/sasv9.cfg file)
UNIX specifics:	default value
See:	MSYMTABMAX= System Option in *SAS Macro Language: Reference*

Syntax

-MSYMTABMAX *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

MSYMTABMAX=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the maximum amount of memory that is available in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of 8 specifies 8 bytes, a value of .782k specifies 801 bytes, and a value of 3m specifies 3,145,728 bytes.

***hex*X**
specifies the maximum amount of memory that is available as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, `2dx` sets the maximum amount of memory to 45 bytes.

MIN
sets the amount of memory that is available to the minimum setting, which is 0 bytes. Setting the amount of memory to the minimum setting causes all macro symbol tables to be written to disk.

MAX
sets the amount of memory that is available to the maximum setting. On 64–bit computers, this value is 9,007,199,254,740,992 bytes.

MVARSIZE System Option: UNIX

Specifies the maximum size for in-memory macro variables.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Macro: SAS macro
PROC OPTIONS GROUP=	MACRO
Default:	32K (set in the installed !SASROOT/sasv9.cfg file)
UNIX specifics:	default value
See:	MVARSIZE System Option in *SAS Macro Language: Reference*

Syntax

-MVARSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

MVARSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the maximum macro variable size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the maximum macro variable size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, **`2dx`** sets the maximum macro variable size to 45 bytes.

MIN
sets the macro variable size to the minimum setting, which is 0 bytes. Setting the macro variable size to the minimum setting causes all macro variable values to be written to disk.

MAX
sets the macro variable size to the maximum setting, which is 65,534 bytes.

NEWS System Option: UNIX

Specifies a file that contains messages to be written to the SAS log.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES, LOGCONTROL

Default:	!SASROOT/misc/base/news (set in the installed !SASROOT/sasv9/cfg file)
UNIX specifics:	-NONEWS option
See:	"NEWS= System Option" in *SAS System Options: Reference*

Syntax

-NEWS *file-specification* | -NONEWS

Required Arguments

-NEWS *file-specification*
specifies an external file. This file contains the messages for the SAS log.

-NONEWS
specifies that the contents of the NEWS file are not displayed in the SAS log, even if the file exists. This option causes any previous NEWS specifications to be ignored.

Details

The contents of the NEWS file are displayed in the SAS log immediately after the SAS header.

See Also

"The SAS Log" in Chapter 9 of *SAS Language Reference: Concepts*

OBS System Option: UNIX

Specifies the last observation that SAS processes in a data set.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Files: SAS files
PROC OPTIONS GROUP=	SASFILES
Default:	MAX
UNIX specifics:	default value
See:	"OBS= System Option" in *SAS System Options: Reference*

Syntax

-OBS *n* | *n*K | *n*M | *n*G | *n*T | *hex*X | MIN | MAX

OBS=*n* | *n*K | *n*M | *n*G | *n*T | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G | *n*T**
specifies a number to indicate when to stop processing. Using one of the letter notations results in multiplying the integer by a specific value. That is, specifying K (kilo) multiplies the integer by 1,024, M (mega) multiplies by 1,048,576, G (giga)

multiplies by 1,073,741,824, or T (tera) multiplies by 1,099,511,627,776. You can specify a decimal value for *n* when it is used to specify a K, M, G, or T value. For example, a value of **`20`** specifies 20 observations or records, a value of **`.782k`** specifies 801 observations or records, and a value of **`3m`** specifies 3,145,728 observations or records.

***hex*X**
: specifies a number as a hexadecimal value to indicate when to stop processing. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, the hexadecimal value F8 must be specified as **`0F8x`** in order to specify the decimal equivalent of 248. For example, the value **`2dx`** specifies the decimal equivalent of 45.

MIN
: sets the number to 0 to indicate when to stop processing.

  If OBS=0 and the NOREPLACE option is in effect, SAS might still be able to take certain actions. For more information, see “OBS= System Option” in *SAS System Options: Reference*.

MAX
: sets the number 9,223,372,036,854,775,807 to indicate when to stop processing.

OPLIST System Option: UNIX

Specifies whether the settings of the SAS system options are written to the SAS log.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOOPLIST
UNIX specifics:	all

Syntax

-OPLIST | -NOOPLIST

Details

The OPLIST system option echoes only the system options specified on the command line; it does not echo any system options specified in the configuration file or in the SASV9_OPTIONS environment variable. (If you want to echo the contents of the configuration file, use the VERBOSE option.) For example, invoke SAS with the following command:

```
sas -nodms -fullstimer -nonews -oplist
```

SAS writes this line to the SAS log:

```
NOTE: SAS command line: -nodms -fullstimer -nonews -oplist
```

See Also

System Options:

- "VERBOSE System Option: UNIX" on page 434

PAGESIZE System Option: UNIX

Specifies the number of lines that compose a page of SAS output.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log and procedure output
PROC OPTIONS GROUP=	LOG_LISTCONTROL, LOGCONTROL
Default:	number of lines on your display for interactive modes; 60 for batch mode
UNIX specifics:	default values and range
See:	"PAGESIZE= System Option" in *SAS System Options: Reference*

Syntax

-PAGESIZE *n* | *n*K | *hex*X | MIN | MAX

PAGESIZE=*n* | *n*K | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K**
specifies the number of lines that compose a page in multiples of 1 (*n*) or 1,024 (*n*K). You can specify decimal values for the number of kilobytes. For example, a value of `800` specifies 800 lines, a value of `.782k` specifies 801 lines, and a value of `3k` specifies 3,072 lines.

***hex*X**
specifies the number of lines that compose a page as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A-F), and then followed by an X. For example, the value `2dx` specifies 45 lines.

MIN
sets the number of lines that compose a page to the minimum setting, which is 15.

MAX
sets the number of lines that compose a page to the maximum setting, which is 32,767.

Details

The default for interactive modes is the number of lines on your display. For batch mode, the default is 60.

See Also

- "Controlling the Content and Appearance of Output in UNIX Environments" on page 104
- "The SAS Log" in Chapter 9 of *SAS Language Reference: Concepts*

PATH System Option: UNIX

Specifies one or more search paths for SAS executable files.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	!SASROOT/sasexe (set in the installed !SASROOT/sasv9/cfg file)
UNIX specifics:	all

Syntax

-PATH *directory-specification*

Required Argument

directory-specification
specifies the search path for SAS executable files.

Details

The PATH system option identifies the search paths for SAS executable files. You can specify multiple PATH options to define the search order. The paths are searched in the order in which SAS encounters them. Therefore, specify at the beginning of the list the paths for the products that you run most frequently. For information about how that order is determined when you specify the PATH system option more that once, see "How SAS Processes System Options Set in Multiple Places" on page 20.

PRIMARYPROVIDERDOMAIN System Option: UNIX

Specifies the domain name of the primary authentication provider.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Initialization and operation
PROC OPTIONS GROUP=	EXECMODES
Alias:	PRIMPD
See:	"PRIMARYPROVIDERDOMAIN= System Option" in *SAS System Options: Reference*

PRINT System Option: UNIX

Specifies a destination for SAS output when running in batch mode.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files

PROC OPTIONS GROUP=	ENVFILES
Default:	the SAS output from a batch SAS program is written to a file in the current directory with the same filename as the SAS source file, with an extension of .lst
UNIX specifics:	all

Syntax

-PRINT *file-specification* | -NOPRINT

Required Arguments

-PRINT *file-specification*
specifies the location for the SAS procedure output file. The *file-specification* can be any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the procedure output file is created in the specified directory. The default name for this file is *filename*.lst, where *filename* is the name of your SAS job.

-NOPRINT
suppresses the creation of the SAS procedure output file.

Details

The PRINT system option specifies a destination for SAS output when running in batch mode. The PRINT system option is valid in batch mode; it is ignored in interactive modes.

See Also

"Using SAS System Options to Route Output" on page 102

PRINTCMD System Option: UNIX

Specifies the print command SAS is to use.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: Procedure output
PROC OPTIONS GROUP=	LISTCONTROL
Default:	none
UNIX specifics:	all

Syntax

-PRINTCMD "*print-command*"

PRINTCMD="*print-command*"

Required Argument

print-command
specifies the options that you can use with PRINTCMD.

Details

The syntax of the options passed to the print command is controlled by the LPTYPE system option. If LPTYPE is set to BSD, the command uses **`lpr`** command options. If LPTYPE is set to SYSV, the command uses **`lp`** command options.

If your site uses a print command (spooler) other than **`lp`** or **`lpr`**, *print-command* specifies its name. The PRINTCMD option overrides the LPTYPE setting.

When specified in an OPTIONS statement, the PRINTCMD option will not change the print commands assigned to previously defined filenames. For example, consider the following code:

```
filename pc1 printer;
proc printto print=pc1;
run;
proc print data=sales.week;
run;

options printcmd="netlp";

filename pc2 printer;
proc printto print=pc2;
run;
proc print data=sales.month;
run;
```

Output associated with PC2 will use the **`netlp`** command; output associated with PC1 will use the default print command.

See Also

System Options:

- "LPTYPE System Option: UNIX" on page 396

Other References:

- "Overview of Printing Output in UNIX Environments" on page 92

REALMEMSIZE System Option: UNIX

Specifies the amount of real (physical) memory SAS can expect to allocate.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	System administration: Memory
PROC OPTIONS GROUP=	MEMORY
Default:	0
UNIX specifics:	valid values

Syntax

-REALMEMSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the amount of memory to reserve in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes), or 1,073,741,824 (gigabytes). The value of *n* can be a decimal value. For example, a value of `8` specifies 8 bytes, a value of `.782k` specifies 801 bytes, and a value of `3m` specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, the value `2dx` sets the amount of memory to 45 bytes.

MIN
specifies a value of 0, which indicates that the memory usage is determined by SAS when SAS starts.

MAX
specifies to set the memory size to the largest permissible value. This value depends on the system limit.

Details

The Basics

The REALMEMSIZE system option sets a recommended upper limit on real memory for procedures that can use both real memory and utility disk space, such as PROC SUMMARY and PROC SORT. This upper limit helps to avoid virtual memory thrashing.

The REALMEMSIZE option should never be set above the amount of real memory. If the amount of real memory is insufficient for a job to run, then setting the MEMSIZE option above the amount of real memory might enable the job to run using a combination of real and virtual memory.

Comparisons

Some SAS procedures use the REALMEMSIZE system option to specify how much real memory the procedure can allocate and use without inducing excessive page swapping. By contrast, the MEMSIZE system option places a limit on the total amount of virtual memory that SAS dynamically allocates at any time. This virtual memory is supported by a combination of real memory and paging space.

The operating environment begins paging when the amount of virtual memory that is required exceeds the real memory that is available. To prevent paging and the associated performance problems, the REALMEMSIZE and MEMSIZE system options should be set to a subset of real memory.

See Also

System Options:

- "MEMSIZE System Option: UNIX" on page 399

Procedures:

- "SORT Procedure: UNIX" on page 309

RSASUSER System Option: UNIX

Controls whether members of the Sasuser library can be opened for update or for Read-Only access.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	NORSASUSER
UNIX specifics:	network considerations
See:	"RSASUSER System Option" in *SAS System Options: Reference*

Syntax

-RSASUSER | -NORSASUSER

Required Arguments

-RSASUSER
limits access to the Sasuser library to Read-Only access.

-NORSASUSER
prevents users from sharing members of the Sasuser library because it allows a user to open a file in the Sasuser library for Update access. Update access requires exclusive rights to the library member.

Details

If the Sasuser library is being shared by multiple users or the same user is running SAS multiple times simultaneously, the Sasuser library is often shared. By default, if one user has a member of the Sasuser library open for update, all other users are denied access to that SAS library member. For example, if one user is writing to the Sasuser.Profile catalog, no other user can even read data from the Profile catalog.

Specifying RSASUSER enables a group of users to share Sasuser library members by allowing all users Read-Only access to members. In the Profile catalog example, if RSASUSER is in effect, all users can open the Profile catalog for Read-Only access, allowing other users to concurrently read from the Profile catalog. However, no user can write information out to the Profile catalog; you receive an error message if you try to do so.

Specifying RSASUSER from the command line affects only that session's access to files. To enable a group of users to share members in the Sasuser library, the system manager should set RSASUSER in a common SAS configuration file, which is shared by all users who will be sharing the Sasuser library.

If you specify RSASUSER but no Profile catalog exists in the Sasuser library, the Profile catalog is created in the Work library.

Note: While the RSASUSER option is extremely useful for sharing information (such as the Profile catalog) stored in the Sasuser library, it is less practical when used in conjunction with SAS/ASSIST software or other SAS modules that require Update access to the Sasuser library.

See Also

"Sharing SAS Files in a UNIX Environment" on page 40

RTRACE System Option: UNIX

Produces a list of resources that are read or loaded during a SAS session.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	none
UNIX specifics:	all

Syntax

-RTRACE ALL | NONE

Required Arguments

ALL
produces a list of resources that are read or loaded during a SAS session.

NONE
turns off RTRACE on all files.

Details

The RTRACE system option produces a list of resources that are read or loaded during the execution of SAS. If you specify -RTRACE ALL but do not specify the RTRACELOC system option, the output is written to the SAS log.

See Also

System Options:

- "RTRACELOC System Option: UNIX" on page 413

RTRACELOC System Option: UNIX

Specifies the pathname of the file to which the list of resources that are read or loaded during a SAS session is written.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
UNIX specifics:	all

Syntax

-RTRACELOC *pathname*

RTRACELOC=*pathname*

Required Argument

pathname
specifies the file to which RTRACE information is written. The *pathname* must include the path and the filename for the RTRACE output.

Details

The RTRACELOC system option specifies the pathname of the file to which RTRACE information is written. If the *pathname* does not include a filename, the output will be directed to standard output. If you specify -RTRACE ALL, but do not specify the RTRACELOC system option, the output is written to the SAS log.

See Also

System Options:

- "RTRACE System Option: UNIX" on page 413

SASAUTOS System Option: UNIX

Specifies the autocall library.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Categories:	Environment control: Files Macro: SAS macro
PROC OPTIONS GROUP=	ENVFILES MACRO
Default:	SASAUTOS fileref
UNIX specifics:	syntax for specifying multiple *directory-specifications*
See:	"SASAUTOS= System Option" in *SAS Macro Language: Reference*

Syntax

-SASAUTOS '*directory-specification*' | *fileref*

-SASAUTOS ('*directory-specification1*' | *fileref1*,...,'*directory-specification-n*' | *filerefn*)

-NOSASAUTOS

SASAUTOS='*directory-specification*' | *fileref*

SASAUTOS =('*directory-specification1*' | *fileref1*,...,'*directory-specification-n*' | *filerefn*)

NOSASAUTOS

Required Arguments

directory-specification
specifies a pathname to an autocall macro library.

fileref
specifies a name (shorthand reference) that has been assigned to an autocall macro library.

Note that the SASAUTOS option uses filerefs, not librefs.

Details

Each autocall macro library consists of files in a UNIX directory. The *directory-specification* can be the pathname of a UNIX directory, a fileref, or an environment variable.

If you specify the pathname of a directory, you must enclose the name in quotation marks. You can omit the quotation marks only if you are specifying the option in the configuration file, in the SAS command, or in the SASV9_OPTIONS environment variable, and if the name cannot be taken to be a fileref.

If you specify a fileref, you must define it before attempting to use any of the autocall macros. You can define the fileref in a FILENAME statement, in an environment variable, or with the FILENAME function. See "Assigning Filerefs to External Files or Devices with the FILENAME Statement" on page 74.

How you specify multiple directory names, filerefs, or environment variables depends on where you specify the SASAUTOS option:

- If you specify the SASAUTOS option in the configuration file or in the SASV9_OPTIONS environment variable, use either multiple SASAUTOS options, or enclose the directory names in parentheses. Separate the names with a comma or a blank space.
- If you specify the SASAUTOS option in the SAS command, use the APPEND or INSERT system options to append to the end or insert at the beginning of the current SASAUTOS value. For example, the following code adds **`/users/userid/also`** to the end of the current SASAUTOS value, **`/users/userid/here`**:

  ```
  sas -sasautos /users/userid/here -append sasautos /users/userid/also
  ```

  For more information, see "APPEND= System Option" in *SAS System Options: Reference* , and "INSERT= System Option" in *SAS System Options: Reference*.
- If you specify the SASAUTOS option in the OPTIONS statement or in the SAS System Options window, you must enclose the directory names in parentheses. Separate the names with a comma or a blank space.

At configuration time, SAS concatenates all directories specified for SASAUTOS. However, after the session starts, any new directories that you specify override any current autocall libraries.

The NOSASAUTOS option causes SAS to ignore all previous SASAUTOS specifications (whether specified in the SAS command, in the configuration file, or in the SASV9_OPTIONS environment variable).

The default value of the SASAUTOS option is the SASAUTOS fileref. There is no UNIX directory assigned to the fileref, so you must define the SASAUTOS fileref if you want to use it as your autocall library.

Examples

Example 1: Specifying Multiple Environment Variables in the OPTIONS Statement

The following example shows the syntax to use if you are specifying multiple environment variables in the OPTIONS statement:

```
options sasautos=(AUTODIR, SASAUTOS);
```

The environment variables that you specify must be defined. For example, you could define the AUTODIR environment variable at SAS invocation by using the following code:

```
-set AUTODIR /tmp/sasautos
```

For more information about how to define an environment variable, see “SET System Option: UNIX” on page 418.

Example 2: Specifying a Fileref in the OPTIONS Statement

The fileref that you specify must be defined. For example, you could define the AUTODIR fileref using a FILENAME statement:

```
filename AUTODIR '/tmp/sasautos';
```

Once the fileref is defined, you can use it in an OPTIONS statement to set the autocall library.

```
options sasautos=AUTODIR;
```

See Also

System Options:

- “INSERT= System Option” in *SAS System Options: Reference*
- “APPEND= System Option” in *SAS System Options: Reference*
- “MAUTOSOURCE System Option” in *SAS Macro Language: Reference*
- “MRECALL System Option” in *SAS Macro Language: Reference*

SASHELP System Option: UNIX

Specifies the locations of Sashelp libraries.

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	!SASROOT/sashelp (set in the installed !SASROOT/sasv9.cfg file)
UNIX specifics:	*directory-specification* can also be an environment variable
See:	"SASHELP= System Option" in *SAS System Options: Reference*

Syntax

–SASHELP *directory-specification*

–SASHELP ('*directory-specification*', '*directory-specification*'...)

Details

This option is set in the installation process and is not normally changed after installation. An environment variable can be specified as the value of SASHELP.

To add additional directory specifications, use the INSERT or APPEND system option. For more information, see "INSERT System Option: UNIX" on page 392, and "APPEND System Option: UNIX" on page 368.

See Also

System Options:

- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

SASSCRIPT System Option: UNIX

Specifies one or more storage locations of SAS/CONNECT script files.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Communications: Networking and encryption
PROC OPTIONS GROUP=	COMMUNICATIONS
Default:	!SASROOT/misc/connect
UNIX specifics:	syntax for specifying multiple directory names

Syntax

-SASSCRIPT '*dir-name*' | ('*dir-name-1*',...,'*dir-name-n*')

SASSCRIPT='*dir-name*' | ('*dir-name-1*',...,'*dir-name-n*')

Details

How you specify multiple directory names in the same SASSCRIPT option depends on where you specify the SASSCRIPT option:

- If you specify the option in the configuration file or in the SASV9_OPTIONS environment variable, use either multiple SASSCRIPT options, or enclose the directory names in parentheses. Separate the names with a comma or a blank space.
- If you specify the option in the SAS command, use multiple SASSCRIPT options because parentheses cause syntax errors.
- If you specify the option in the OPTIONS statement or in the SAS System Options window, you must enclose the directory names in parentheses. Separate the names with a comma or a blank space.

See Also

System Options:

- “SASSCRIPT= System Option” in *SAS/CONNECT User's Guide*

SASUSER System Option: UNIX

Specifies the name of the Sasuser library.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	~/sasuser.v93 (set in the installed !SASROOT/sasv9.cfg file)
UNIX specifics:	*pathname* can be an environment variable
See:	“SASUSER= System Option” in *SAS System Options: Reference*

Syntax

–SASUSER *pathname*

Details

The *pathname* identifies the directory for the Sasuser library that contains a user's Profile catalog. You can use an environment variable to specify the pathname, for example:

```
sas -sasuser $HOME
```

SET System Option: UNIX

Defines an environment variable.

Valid in:	configuration file, SAS invocation, OPTIONS Statement, SASV9_OPTIONS environment variable

Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
UNIX specifics:	all

Syntax

–SET *variable-name value*

SET=*variable-name value*

Details

The SET option lets you define an environment variable that is valid within the SAS session and any shell started from within the SAS session. Using the SET option is similar to using the SAS **setenv** command. For information about executing system commands from within your SAS session, see "Executing Operating System Commands from Your SAS Session" on page 15.

A special use for the SET option is to specify the name of the **!SASROOT** directory:

```
-set SASROOT pathname
```

The pathname specified can then be used to expand **!SASROOT** (as shown in Table 2.6 on page 54).

After exiting your SAS session, environment variables that are set with the SET option no longer exist.

See Also

- "Introduction to the !SASROOT Directory" on page 443
- "Defining Environment Variables in UNIX Environments" on page 24

SORTANOM System Option: UNIX

Specifies certain options for the host sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Sort: Procedure options
PROC OPTIONS GROUP=	SORT
Default:	none
UNIX specifics:	all

Syntax

SORTANOM=*option(s)*

–SORTANOM *option(s)*

Required Argument

options
can be any one or more of the following:

B
tells SyncSort to run in multi-call mode, instead of single-call mode. (Refer to the documentation for `syncsort` for more information.)

Note: This option is available for `syncsort` only.

T
writes to the SAS log statistics about the external sorting process.

V
writes to the SAS log all of the commands that are passed to the host sort utility.

SORTCUT System Option: UNIX

Specifies the number of observations that SAS sorts. if the number of observations in the data set is greater than the specified number, the host sort program sorts the remaining observations.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Sort: Procedure options
PROC OPTIONS GROUP=	SORT
Default:	0
UNIX specifics:	all

Syntax

-SORTCUT *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

SORTCUT=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the number of observations in multiples of 1 (*n*); 1,024 (*n*K); 1,048,576 (*n*M); or 1,073,741,824 (*n*G). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`800`** specifies 800 observations, a value of **`.782k`** specifies 801 observations, and a value of **`3m`** specifies 3,145,728 observations.

***hex*X**
specifies the number of observations as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, the value **`2ffx`** specifies 767 observations.

MIN
specifies 0 observations.

MAX
specifies 9,007,199,254,740,992 observations.

Details

When you specify SORTPGM=BEST, SAS uses the value of the SORTCUT and SORTCUTP options to determine whether to use the host sort or the SAS sort. If the number of observations in the data set is greater than the number that you specify with SORTCUT, the host sort will be used. If both SORTCUT and SORTCUTP are either not defined or are set to 0, the SAS sort is used. If you specify both options and either condition is true, SAS chooses the host sort.

See Also

System Options:

- "SORTCUTP System Option: UNIX" on page 421
- "SORTPGM System Option: UNIX" on page 424

SORTCUTP System Option: UNIX

Specifies the number of bytes that SAS sorts. If the number of bytes in the data set is greater than the specified number, the host sort program sorts the remaining data set.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Sort: Procedure options
PROC OPTIONS GROUP=	SORT
Default:	0
UNIX specifics:	all

Syntax

-SORTCUTP *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

SORTCUTP=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**
specifies the number of bytes in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the number of bytes as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, the value **2dx** specifies 45 bytes.

MIN
specifies 0 bytes.

MAX
specifies 9,007,199,254,740,992 bytes.

Details

When you specify SORTPGM=BEST, SAS uses the value of the SORTCUT and SORTCUTP options to determine whether to use the host sort or the SAS sort. If the data set to be sorted is larger than the number of bytes (or kilobytes or megabytes) that you specify with SORTCUTP, the host sort will be used instead of the SAS sort. The value that you specify must be less than or equal to 2,147,483,647 bytes. If both SORTCUT and SORTCUTP are either not defined or are set to 0, the SAS sort is used. If you specify both options and either condition is true, SAS chooses the host sort.

The following equation computes the number of bytes to be sorted:

```
number-of-bytes= ((length-of-obs)+(length-of-all-keys))*number-of-obs
```

See Also

System Options:

- "SORTANOM System Option: UNIX" on page 419
- "SORTCUT System Option: UNIX" on page 420
- "SORTPGM System Option: UNIX" on page 424

SORTDEV System Option: UNIX

Specifies the pathname used for temporary files created by the host sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Sort: Procedure options
PROC OPTIONS GROUP=	SORT
Default:	same location as -WORK, which is set in the installed !SASROOT/sasv9.cfg file
UNIX specifics:	all

Syntax

SORTDEV='*directory-specification*'

-SORTDEV *directory-specification*

Details

The SORTDEV option specifies an alternative directory for temporary files created by the host sort program.

SORTNAME System Option: UNIX

Specifies the name of the host sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Sort: Procedure options
PROC OPTIONS GROUP=	SORT
Default:	none
UNIX specifics:	all

Syntax

SORTNAME='*host-sort-utility-name*'

-SORTNAME *host-sort-utility-name*

Details

The SORTNAME option specifies the name of the default host sort utility, **`syncsort`**.

See Also

System Options:

- "SORTPGM System Option: UNIX" on page 424

SORTPARM System Option: UNIX

Specifies parameters for the host sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Sort: Procedure options
PROC OPTIONS GROUP=	SORT
Default:	none
UNIX specifics:	all

Syntax

SORTPARM='*parameters*'

–SORTPARM '*parameters*'

Required Argument

parameters
specifies any parameters that you want to pass to the sort utility. For a description of these parameters, see the documentation for the sort that you are using.

SORTPGM System Option: UNIX

Specifies whether SAS sorts using the SAS sort utility or the host sort utility.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Sort: Procedure options
PROC OPTIONS GROUP=	SORT
Default:	BEST
UNIX specifics:	all

Syntax

-SORTPGM SAS | HOST | BEST

SORTPGM=SAS | HOST | BEST

Required Arguments

SAS
tells SAS to use the SAS sort.

HOST
tells SAS to use the sort that is specified by the SORTNAME system option.

BEST
tells SAS to determine the best routine to sort the data set: the SAS sort or the host sort that is specified by the SORTNAME system option. The settings of the SORTCUT and SORTCUTP system options determine whether SAS chooses the SAS sort or the host sort.

Details

The SORTPGM system option tells SAS whether to use the SAS sort, to use the host sort, or to determine which sort is best for the data set.

See Also

System Options:

- "SORTCUTP System Option: UNIX" on page 421
- "SORTNAME System Option: UNIX" on page 422
- "SORTSIZE System Option: UNIX" on page 424

SORTSIZE System Option: UNIX

Specifies the amount of memory available to the SORT procedure.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Categories:	Sort: Procedure options System administration: Memory
PROC OPTIONS GROUP=	SORT MEMORY
Default:	depends on your operating environment
UNIX specifics:	value of MAX
See:	"SORTSIZE= System Option" in *SAS System Options: Reference*

Syntax

–SORTSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

SORTSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

Required Arguments

***n* | *n*K | *n*M | *n*G**

specifies the number of bytes in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**

specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hexadecimal characters (0–9, A–F), and then followed by an X. For example, the value **`2dx`** sets the amount of memory to 45 bytes.

MIN

specifies 0 bytes, which indicates that there is no limit except the limitation specified by the MEMSIZE system option.

MAX

specifies the maximum addressable memory for the operating environment.

Details

The SORT procedure uses the SORTSIZE system option to limit the amount of memory that it acquires or allocates for sorting. The amount of memory that SAS uses for the SORT procedure also depends on the values of the MEMSIZE and REALMEMSIZE system options. By contrast with the SORTSIZE option, the MEMSIZE system option places a limit on the total amount of virtual memory that SAS dynamically allocates at any time. This virtual memory is supported by a combination of real memory and paging space. The operating environment begins paging when the amount of virtual memory that is required exceeds the real memory that is available. To prevent paging and the associated performance problems, the SORTSIZE system option should be set to a subset of real memory. You can set SORTSIZE to MAX if MEMSIZE is set to a subset of real memory. In most cases, you can set SORTSIZE=MAX because this value will limit the amount of memory that is used by the SORT procedure.

See Also

System Options:

- "MEMSIZE System Option: UNIX" on page 399

Procedures:

- "SORT Procedure: UNIX" on page 309

STDIO System Option: UNIX

Specifies whether SAS should use stdin, stdout, and stderr.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Input control: Data processing
PROC OPTIONS GROUP=	INPUTCONTROL
Default:	NOSTDIO
UNIX specifics:	all

Syntax

–STDIO | –NOSTDIO

Details

This option tells SAS to take its input from standard input (stdin), to write its log to standard error (stderr), and to write its output to standard output (stdout).

This option is designed for running SAS in batch mode or from a shell script. If you specify this option interactively, SAS starts a line mode session. The STDIO option overrides the DMS, DMSEXP, and EXPLORER system options.

The STDIO option does not affect the assignment of the Stdio, Stdin, and Stderr filerefs. For more information, see "Filerefs Assigned by SAS in UNIX Environments" on page 79.

For example, in the following SAS command, the file **`myinput`** is used as the source program, and files **`myoutput`** and **`mylog`** are used for the procedure output and log respectively.

```
sas -stdio < myinput > myoutput 2> mylog
```

If you are using the C shell, you should use parentheses:

```
(sas -stdio < myinput > myoutput ) >& output_log
```

See Also

"The Default Routings for the SAS Log and Procedure Output in UNIX Environments" on page 93

STIMEFMT System Option: UNIX

Specifies the format that is used to display the time on FULLSTIMER and STIMER output.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	512M
UNIX specifics:	all

Syntax

-STIMEFMT *value(s)*

STIMEFMT=*value(s)*

Required Arguments

value
specifies the options to use with STIMEFMT. The following options are available:

Datetime Stamp options
The datetime stamp options are described below:

TS	specifies to always display the datetime stamp as part of STIMER and FULLSTIMER.
TSFULL	specifies to display the datetime stamp as part of FULLSTIMER. TSFULL is the default.
TSOFF	turns off the datetime stamp for STIMER and FULLSTIMER.

Memory
is normally displayed as part of FULLSTIMER. The default memory output is displayed in kilobytes. The following options for memory are available:

MEMFULL	writes memory statistics as part of FULLSTIMER, but not as part of STIMER.
MEM	writes memory statistics as part of FULLSTIMER and STIMER.
KB	writes memory in kilobytes.
MB	writes memory in megabytes.
GB	writes memory in gigabytes.
C	adds commas to the numbers in the memory display.
NC	does not add commas to the numbers in the memory display.

Elapsed and CPU time
can be configured to display hours, minutes, seconds, or best fit in STIMER and FULLSTIMER.

Z | H | HOURS	writes the time as hours:minutes:seconds.
M | MINUTES	writes the time as minutes:seconds.
S | SECONDS	writes the time as seconds.
HMS	writes the format leaving out leading zeros for hours and minutes.

Counters
specifies that additional counters can be displayed as part of FULLSTIMER.

E | ENABLE	enables extra counters.
D | DISABLE	disables extra counters.

Help
provides two values that are used to access help for the STIMEFMT option:

FMT	lists the available datetime stamp formats.
OPT	lists other option values that are available.

Details

STIMEFMT Basics

The STIMEFMT system enables you to customize the format of output produced by the STIMER and FULLSTIMER system options. You can perform the following tasks using STIMEFMT:

- list the formats that are available:

  ```
  options stimefmt = fmt;
  ```

- list other options that are available:

  ```
  options stimefmt = opt;
  ```

- turn the datetime stamp on or off for STIMER:

  ```
  options stimefmt = tson | tsoff | tsfull;
  ```

- combine options as needed:

  ```
  options stimefmt = (tson YYNNDDS);
  ```

- separate a memory value with commas:

  ```
  options stimefmt = c;
  ```

- do not use commas when specifying values:

  ```
  options stimefmt = nc;
  ```

- select a unit for memory:

  ```
  options stimefmt = GB | MB | KB;
  ```

- turn on memory reporting for STIMER and FULLSTIMER:

  ```
  options stimefmt = mem;
  ```

- set the time display in the datetime stamp:

  ```
  options stimefmt = TOD | TIME | TIMEAMPM;
  (TOD and TIME specify military time.)
  ```

- control the display of CPU or real time by using hours or minutes

Formats for Displaying the Datetime Stamp

The format of the datetime stamp can be set to standard formats that are supported by SAS. These formats include the following:

```
ABS.           (Absolute seconds since Jan. 1, 1970)

DATE.          DATE9.

DDMMYY.         DDMMYY10.     DDMMYYB.
DDMMYYB10.      DDMMYYC.      DDMMYYC10.
DDMMYYD.        DDMMYYD10.    DDMMYYN.
DDMMYYN10.      DDMMYYP.      DDMMYYP10.
DDMMYYS.        DDMMYYS10.

ISO.            (ISO Standard Time)

MMDDYY.          MMDDYY10.     MMDDYY.
MMDDYYB10.       MMDDYYC.      MMDDYYC10.
MMDDYYD.         MMDDYYD10.    MMDDYYN.
MMDDYYN8.        MMDDYYP.      MMDDYYP10.
MMDDYYS.         MMDDYYS10.

NLDATM.          NLDATMAP.

YYMMDD.           YYMMDD10.     YYMMDDB.
YYMMDDB10.        YYMMDDC.      YYMMDDC10.
YYMMDDD.          YYMMDDD10.    YYMMDDN.
YYMMDDN8.         YYMMDDP.      YYMMDDP10.
YYMMDDS.          YYMMDDS10.

TOD.                (Writes time as military time.)
TIME.               (Writes time as military time.)
TIMEAMPM.           (Writes time as AM and PM.)
```

The syntax for the OPTIONS statement is listed below:

```
options stimefmt = fmt;
```

where *fmt* is a valid SAS format.

Using Multiple Values for the STIMEFMT Option

The STIMEFMT option can specify multiple values at the same time to enable you to set multiple settings. Multiple values must be enclosed in parentheses. For example:

```
options stimefmt = (h YYMMDD. gb c);
```

Displaying the Settings for the STIMEFMT Option

PROC OPTIONS always displays the current state of all settings for STIMEFMT. The following example shows log output when you execute PROC OPTIONS:

```
proc options option=stimefmt;
run;
```

Log 18.2 *Log Output from PROC OPTIONS*

```
    SAS (r) Proprietary Software Release 9.3  TS1B0

 STIMEFMT=(NLDATM2. HMS TIMEAMPM KB MEMFULL TSFULL NC)
                   Specified the output format for FULLSTIMER and STIMER.
                   This controls the timestamp, memory, CPU and elapsed time.
```

Resetting STIMEFMT to the Default Values

You can reset the settings for STIMEFMT to its default values by executing the following OPTIONS statement:

```
options stimefmt = normal;
```

See Also

System Options:

- “STIMER System Option: UNIX” on page 430
- “FULLSTIMER System Option: UNIX” on page 386

STIMER System Option: UNIX

Specifies whether to write a subset of system performance statistics to the SAS log.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	STIMER
UNIX specifics:	all

Syntax

–STIMER | –NOSTIMER

STIMER | NOSTIMER

Required Arguments

STIMER
writes only real time and CPU time to the SAS log.

NOSTIMER
does not write any statistics to the SAS log.

Details

The STIMER system option specifies whether a subset of all the performance statistics of your system that are available to SAS are written to the SAS log. (Using STIMEFMT can affect the output.) The following is an example of STIMER output:

Log 18.3 *STIMER Output*

```
real time   1.34 seconds
cpu time    0.04 seconds
```

STIMER displays the following statistics:

Table 18.3 *Description of STIMER Statistics*

Statistic	Description
real time	the amount of time spent to process the SAS job. Real time is also referred to as elapsed time.
CPU time	the total time spent to execute your SAS code and to perform system overhead tasks on behalf of the SAS process. This value is the combination of the user CPU and system CPU statistics from FULLSTIMER.

If both STIMER and FULLSTIMER are set, the FULLSTIMER statistics are written to the SAS log.

Note: Starting in SAS 9, some procedures use multiple threads. On computers with multiple CPUs, the operating system can run more than one thread simultaneously. Consequently, CPU time might exceed real time in your STIMER output. For example, a SAS procedure could use two threads that run on two separate CPUs simultaneously. The value of CPU time would be calculated as the following:

```
CPU1 time + CPU2 time = total CPU time
1 second + 1 second = 2 seconds
```

Because CPU1 can run a thread at the same time that CPU2 runs a separate thread, you can theoretically consume 2 CPU seconds in 1 second of real time.

See Also

System Options:

- "FULLSTIMER System Option: UNIX" on page 386
- "STIMEFMT System Option: UNIX" on page 427

SYSIN System Option: UNIX

Specifies the default location of SAS source code when running in batch mode.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
UNIX specifics:	all

Syntax

–SYSIN *filename* | -NOSYSIN

Required Arguments

-SYSIN *filename*
: specifies an external file. The value for *filename* must be a valid UNIX filename.

-NOSYSIN
: invokes SAS, processes the autoexec file, and then terminates SAS, returning you to the command prompt.

Details

This option applies only when you are using batch mode. It is not necessary to precede the filename with the SYSIN option if the filename immediately follows the keyword **SAS**. For example, the following two SAS commands are equivalent:

```
sas saspgms/report1.sas
sas -sysin saspgms/report1.sas
```

The syntax of the SYSIN system option also enables you to specify NOSYSIN. If you specify NOSYSIN, SAS is invoked, the autoexec file is processed, and then SAS terminates, returning you to the command prompt. The following example shows the syntax:

```
sas -nosysin -autoexec mysas.sas
```

This option is useful if you want to test an autoexec file without actually running a complete SAS session.

See Also

"Starting SAS Sessions in UNIX Environments" on page 4

SYSPRINT System Option: UNIX

Specifies the destination for printed output.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: Procedure output
PROC OPTIONS GROUP=	LISTCONTROL and ODSPRINT
Default:	default system printer
UNIX specifics:	all

Syntax

–SYSPRINT *destination* | '*destination option-list*'

SYSPRINT=*destination* | '*destination option-list*'

Required Arguments

destination
is the name of a hard-copy device at your site. Consult your system administrator for a list of available destinations.

option-list
is the list of options to pass to the `lp` (or `lpr`) command.

Details

The SYSPRINT option specifies a destination for printed output other than the default system printer. You can use the option list to pass options to the `lp` (or `lpr`) command.

Note: When a fileref is assigned, the SYSPRINT option is queried. If the value of the SYSPRINT option is later changed, the fileref does not pick up this change.

For more information, see "Changing the Default Print Command in UNIX Environments" on page 104.

See Also

Commands:

- "PRINTCMD System Option: UNIX" on page 409

Other References:

- "Overview of Printing Output in UNIX Environments" on page 92

USER System Option: UNIX

Specifies the name of the default permanent SAS library.

Valid in:	configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	none
UNIX specifics:	*pathname* must be a valid UNIX pathname
See:	"USER= System Option" in *SAS System Options: Reference*

Syntax

-USER *pathname*

USER='*pathname*' | *libref*

Required Arguments

pathname
identifies the directory containing your default permanent SAS library. It must be a directory name.

libref
is the libref associated with the directory containing your default permanent SAS library. It must already be assigned.

See Also

"Using One-Level Names to Access Permanent Files (User Library)" on page 61

VERBOSE System Option: UNIX

Specifies whether SAS writes the system option settings to the SAS log.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Log and procedure output control: SAS log
PROC OPTIONS GROUP=	LOGCONTROL
Default:	NOVERBOSE
UNIX specifics:	all

Syntax

–VERBOSE | –NOVERBOSE

Required Arguments

-VERBOSE
writes the settings of SAS system options from the configuration file, the SAS command, and the SASV9_OPTIONS environment variable to the SAS log. For the CONFIG option, VERBOSE lists the names of the configuration files.

-NOVERBOSE
does not write the settings of the system options to the SAS log.

Details

In previous releases of SAS, the output from the VERBOSE system option appeared as a simple list of options and their values. The list appeared in the window where SAS was invoked. Pressing the ENTER key advanced the list one line at a time. Pressing the space bar advanced the list page by page. Pressing the Q key displayed the entire list, and brought you back to the prompt.

For 9.3, the list of system options and their values is still created. In addition, SAS creates a list that identifies where the options were set. This list is written to a journal file, and then it is written to the SAS log. The advantage of writing to a global journal file is that if SAS fails to initialize, output is still available, even though a SAS log was not created.

See Also

System Options:

- "OPLIST System Option: UNIX" on page 406

Other References:

- "Customizing Your SAS Session by Using System Options" on page 18

WORK System Option: UNIX

Specifies the location of the Work library.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	set in the installed !SASROOT/sasv9.cfg file
UNIX specifics:	all
Note:	This option can be restricted by a site administrator. For more information, see "Restricted Options" in Chapter 1 of *SAS System Options: Reference*.
See:	"WORK= System Option" in *SAS System Options: Reference*

Syntax

–WORK *pathname*

Required Argument

pathname
specifies the location of the SAS Work library if *pathname* is a directory. If *pathname* describes a file, then SAS chooses a directory from that file as the location for the Work library for the current SAS session.

Details

The Basics

If the value of *pathname* is a directory, then SAS continues its initialization using the directory as the location for the Work library. If the value of *pathname* is a file, then SAS opens the file and selects one of the paths to use as the location for the Work library. SAS either selects a path at random or selects a path based on available space. Once a work location is picked at start-up, all of the work files for the session are sent to the single work directory.

Making the Allocation of Work Libraries More Dynamic

If *pathname* refers to a file, then that file contains a list of locations that can be used for the Work library. Individual SAS Work libraries still reside in a single directory. You use METHOD=RANDOM to specify that the Work directory is randomly chosen from the list of directories. SAS chooses one location per session as the location of the Work directory. This selection enables you to balance the I/O load across multiple hardware systems. You use METHOD=SPACE to specify the directory that has the most available space. If METHOD is not specified, SAS defaults to choosing a directory at random.

Examples

Example 1: Spreading a Processing Load across Multiple Volumes of Different Disks

The following example shows how to spread an I/O processing load across multiple volumes of different disks. In this case, you use METHOD=RANDOM. A file named **`/sasinfo/workfiles`** contains the following information:

```
/disk1/sastempfiles
/disk2/sastempfiles
/disk3/sastempfiles
method=random
```

The Work library for a particular SAS session will be placed on either disk1, disk2, or disk3. The configuration file or command line would include the following:

```
-work /sasinfo/workfiles
```

Example 2: Choosing the Directory That Has the Most Free Space

When you process your data, you can choose the directory that has the most free space. In this case, you use METHOD=SPACE. In the following example, **`/sasinfo/workfiles`** contains the following directories:

```
/disk1/sastempfiles
/disk2/sastempfiles
/disk3/sastempfiles
method=space
```

The Work library is placed on the disk with the most free space.

See Also

System Options:

- “WORKINIT System Option: UNIX” on page 436

WORKINIT System Option: UNIX

Initializes the Work library.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	WORKINIT
UNIX specifics:	WORKINIT does not erase files from previous sessions
See:	“WORKINIT System Option” in *SAS System Options: Reference*

Syntax

–WORKINIT | –NOWORKINIT

Required Arguments

-WORKINIT
specifies that a new subdirectory is to be created in the directory specified in the WORK option.

-NOWORKINIT
specifies that the system is to use the directory specified by the WORK option.

- If the system does not find any old subdirectories, it creates a new one.
- If the system finds more than one old subdirectory, it uses the latest one.
- If file locking is in effect (see "FILELOCKS System Option: UNIX" on page 381), the system looks for the latest unlocked directory. If it finds none, then it creates a new one.

Details

The WORKINIT option controls whether the Work library is initialized at SAS invocation.

See Also

System Options:

- "FILELOCKS System Option: UNIX" on page 381
- "WORK System Option: UNIX" on page 435

WORKPERMS System Option: UNIX

Sets the permissions of the SAS Work library when it is initially created.

Valid in:	configuration file, SAS invocation
Category:	Environment control: Files
PROC OPTIONS GROUP=	ENVFILES
Default:	700
UNIX specifics:	all

Syntax

–WORKPERMS *permission-value*

Required Argument

permission-value
specifies the octal value representing the permissions for the SAS Work directory. Values can be any octal value setting the permission of a UNIX directory. Examples of values include umask, 700, 755, 770, 775, and 777.

Details

The WORKPERMS system option enables you to change or remove the current file mode creation mask value when you initially create a SAS Work library. This means that you can change the value of *permission-value* to change file permissions for a new Work library.

XCMD System Option: UNIX

Specifies whether the X command is valid in the SAS session.

Valid in:	configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category:	Environment control: Display
PROC OPTIONS GROUP=	ENVDISPLAY
Default:	XCMD
UNIX specifics:	all

Syntax

-XCMD | -NOXCMD

Required Arguments

-XCMD
specifies that the X command is valid in the current SAS session.

-NOXCMD
specifies that the X command is not valid in the current SAS session.

Details

The XCMD system option specifies whether the X command is valid in the current SAS session.

You cannot use several SAS statements, objects, or facilities if you use the NOXCMD system option. Examples of these statements, objects, and facilities include the following:

- the PIPE device type in the FILENAME statement
- the CALL SYSTEM routine
- the %SYSEXEC macro
- any facility that SAS uses to execute a shell-level command

See Also

CALL Routines:

- "CALL SYSTEM Routine: UNIX" on page 263

Commands:

- "X Command: UNIX" on page 242

Macros:

- "%SYSEXEC" on page 289

Statements:

- "FILENAME Statement: UNIX" on page 323

Other References:

- "Executing Operating System Commands from Your SAS Session" on page 15

Part 5

Appendixes

Appendix 1
The !SASROOT Directory

Introduction to the !SASROOT Directory

When SAS is installed, its entire directory structure is located in a directory in your file system. This directory is called **`SASHOME`**. The **`SASHOME`** directory can be located anywhere in your file system. The default location for **`SASHOME`** is **`/usr/local/SAS`**. The traditional **`!SASROOT`** directory (SAS Foundation) is automatically installed in a subdirectory that is located in **`SASHOME`**. The default directory for **`!SASROOT`** is **`SASHOME/SASFoundation/9.x`**, where **`x`** is the SAS release.

Contents of the !SASROOT Directory

The **`!SASROOT`** directory contains the files required to use SAS. This directory includes invocation points, configuration files, sample programs, catalogs, data sets, and executable files. You do not need to know the organization of these directories to use SAS.

If all available SAS products are installed on your system, the **`!SASROOT`** directory contains the files and directories that are listed in the following tables:

Table A1.1 *SAS Files in the !SASROOT Directory*

SAS File	Description of Contents
sas	is the default invocation point for SAS.
sassetup	enables you to renew your SAS license.
setinit.sas	is the SAS file that was used to update the license information.
sasv9.cfg	is the default system configuration file for SAS. This file should not be edited. (See sasv9_local.cfg.)

SAS File	Description of Contents
sasv9_local.cfg	is the file where user-specified system options should be added. This file overrides the options in the default system configuration file, and prevents the options from being lost when you reinstall or upgrade SAS.

Table A1.2 *SAS Subdirectories in the !SASROOT Directory*

SAS Subdirectory	Description of Contents
bin	contains the invocation scripts for each language listed in the **`NLS`** directory. This directory also contains the sasenv script that sets the environment variables that are required by SAS. sasenv_local is the file that you modify. sasenv_local is also the last file that SAS reads when processing environment variables.
dbcs	contains the subdirectories for a DBCS installation.
install	contains the admin subfolder, which contains data files and subfolders that are used by sassetup. It also contains registry and sasregord subfolders, which contain data files that are used to build the SAS Registry during installation post-processing.
maps	is a SAS library that contains SAS data sets used by SAS/GRAPH software to produce maps. You receive some maps with SAS/GRAPH software. Additional maps are available in the SAS Map Data Library Series.
misc	contains miscellaneous components. This directory also contains components for various SAS products, such as script files for SAS/CONNECT software and thin client interfaces for SAS/SHARE software. In this directory, the **`DEPLOYMENT`** directory contains template files that are required by the SAS installation program. These template files should not be altered. There is also a **`SASSETUP`** directory, which contains program scripts that are used by the sassetup utility in **`!SASROOT`**.
nls	contains subdirectories for national language and locale support. These directories include **`DBCS`** (double-byte character set), **`DE`** (–LOCALE German), **`EN`** (–LOCALE en_US), **`ES`** (–LOCALE Spanish), **`FR`** (–LOCALE French), **`HU`** (–LOCALE Hungarian), **`IT`** (–LOCALE Italian), **`JA`** (–LOCALE ja_JP—Primary Japanese encoding), **`JA.SJIS`** (–LOCALE ja_JP—Secondary Japanese encoding), **`KO`** (–LOCALE ko_KR), **`NO`** (–LOCALE Norwegian), **`PB`** (–LOCALE pt_BR—Portuguese Brazilian), **`PL`** (–LOCALE Polish), **`RU`** (–LOCALE Russian), **`SV`** (–LOCALE Swedish), **`U8`** (–LOCALE en_US—for UTF-8 support), **`ZH`** (–LOCALE zh_CN—Chinese), and **`ZT`** (contains subfolders for traditional Chinese fonts supported in SAS). Most of these folders contain a sasv9.cfg configuration file, which makes the NLS-specific content available in SAS when SAS is invoked using the language-specific SAS invocation script. Each language directory contains a **`SASCFG`** subdirectory that contains the SAS Registry and SAS Desktop data sets that are generated during installation. This list is a snapshot of the SAS **`NLS`** subdirectories for national language support. The list will include support for additional locales when they become available.

SAS Subdirectory	Description of Contents
perl	contains Perl binaries and libraries that are used by the sassetup program and SAS feature testing tools.
samples	contains sample programs for different SAS products. These programs are organized by product subdirectory, and might not include samples for every SAS product.
sasautos	contains predefined SAS macros. See "Using Autocall Libraries in UNIX Environments" on page 290.
sasexe	contains executable files for different SAS products.
sashelp	is a SAS library that contains online Help files, menus, descriptions of graphics devices, and other catalogs used by SAS procedures that support windows.
sasmsg	contains files that contain all of the messages and notes that are used by SAS.
saspgm	contains various components of SAS products.
sastest	contains files that are used by the SAS feature testing tools.
utilities	contains man pages and utility programs. For more information, see "The Utilities Directory in UNIX Environments" on page 447.
X11	contains the files needed to run SAS with the X Window System. These files include bitmap files, online Help files, and resource files.

Appendix 2

Tools for the System Administrator

The Utilities Directory in UNIX Environments

The **`!SASROOT/utilities`** directory contains the following important subdirectories:

`man`
contains the online manual pages for SAS. "Installing Manual Pages" on page 447 describes how to make these pages accessible to users through the UNIX **`man`** command.

`bin`
contains the executable files for administrative tools. "Utilities in the /utilities/bin Directory" on page 448 describes some of the tools in this directory.

`src/auth`
contains source files and documentation for the UNIX Authentication API. The API enables administrators to add custom authentication methods to SAS authentication in UNIX environments. For more information, see "The UNIX Authentication API" on page 448.

Installing Manual Pages

To be able to read the manual pages in the **`utilities/man`** directory, copy the files to the **`man1`** subdirectory of the location of the other man files for your system. This location is usually **`/usr/man`** or **`/usr/local/man`**. Execute the UNIX **`man man`** command to determine the appropriate pathname for your system. When you have found the correct pathname, use the following command to copy the SAS man files:

```
cp -r sasroot/utilities/man/* pathname
```

pathname is the directory location of your system man files.

For example, the following command enables you to access online Help by copying the SAS man files from the **`!SASROOT`** directory to the **`man1`** file in your system's **`man`** directory:

```
cp /usr/local/SASHome/SASFoundation/9.3/utilities/man/* /usr/local/man/man1
```

After you issue this command, you can access online Help with the **`man sas`** command.

You can also add the directory to your system's MANPATH environment variable if it has been previously defined, or you can set you own MANPATH environment variable,

The UNIX Authentication API

The UNIX Authentication Application Programming Interface (API) is a set of predefined routines that provide user authentication, identification, and permissions verification for SAS when running in UNIX environments. The source files provide the ability to add site-specific behavior to the authentication/identification/permissions validations process.

The **`!SASROOT/utilities/src/auth/docs.pdf`** file describes how to implement custom authentication implementations, and documents the API itself. The document also includes an explanation of how SAS user authentication and identification integrates with authentication features provided by the operating environment. Administrators that need to implement custom behaviors should read the file and follow the instructions.

Utilities in the /utilities/bin Directory

The following table briefly describes some of the tools in the **`/utilities/bin`** directory. You can use the UNIX **`man`** command for information about these utilities.

Table A2.1 *Tools for the System Administrator*

Tool Name	Description
authcustom.so	sasauth module for site-specific authentication
authldap.so	sasauth module for LDAP authentication
authpam.so	sasauth module for PAM authentication
bdm	batch driver monitor
cfgpeh	stand-alone scramble command
cleanwork	tool to remove any leftover Work directories, utility directories, or both, whose associated SAS process has ended(See the "cleanwork command" on page 450.)
docsetup	documentation setup utility invoked by the installer
elsconf	tool to check ELS configuration

Tool Name	**Description**
elssrv	ELS server, tool to launch subprocesses
jproxy	tool used to launch the Java facilities within SAS
ke2j	double-byte input utility
ke2s	double-byte input utility
kj2e	double-byte input utility
ks2e	double-byte input utility
loadmgr	application lead manager
motifxsassm	Motif X session manager
objspawn	object spawner
patchname	resets the name of the `SASROOT` directory in the specified executable file.
rbrowser	remote browser server for the platform (only supported on Linux, but works on all other Motif platforms)
reshelper	resource helper for X Windows
sasauth	user identification and authentication utility
sasauth.conf	configuration file for sasauth; specifies authentication module used and other options
sasm.elm.mime	script for supporting e-mail from SAS
sasmailer	script for supporting e-mail from SAS
sasperm	user permissions utility
sastcpd	TCP/IP access daemon
sasumgmt	obtains and transcodes or decodes the user name and password into Unicode. It then calls the SAS authorization service to authenticate the user. It then exits with an exit status that indicates the success or failure of the authentication.
saswujms	Japanese input server
setuid	directory
setuid.sh	script to set some commands to run as root
tkdef.so	the location where SASROOT and TKPATH are patched for release 9.3 (other executables are not patched as they were in release 9.2)

cleanwork command

Deletes any leftover Work directories, utility directories, or both, whose associated SAS process has ended.

cleanwork *directory*<*-n*, *-hostmatch*>

directory
names the directory that contains the Work directory, the Utility directory, or both directories. The name must match the value specified in the WORK system option or the value specified in the UTILLOC system option.

Tip: Unless the cleanwork command is run by root, user permissions might prevent you from removing a directory.

-n
specifies that SAS list the entries in a directory that can be removed.

-hostmatch
specifies the name of a host from which you can remove Work directories that might still be active in a Network File System (NFS).

Details

The cleanwork command removes any subdirectories that were assigned to the Work library or directories assigned by the UTILLOC system option. cleanwork removes only those files that are associated with defunct SAS processes. Each subdirectory name has a format of the form:

`SAS_workcode_nodename`

`SAS_utilcode_nodename`

code
is a 12-character code. The first four characters are randomly generated numbers. The next eight characters are based on the hexadecimal representation of the process ID of the associated SAS process. Files that are associated with active processes are not removed.

nodename
specifies the name of the UNIX system where the SAS process is running.

For example, if you are working on nodename *jupiter*, then the cleanwork command removes all directories with inactive processes on *jupiter*. cleanwork does not remove a directory that is associated with an orphaned process if that process is still active. In this case, you need to manually kill the process, and then rerun cleanwork.

See Also

"Work Library" on page 60

Appendix 3

Text Editing Commands

Text-Editing Commands

Commands that are specific to the text editor are called text-editing commands because they perform editing functions in windows. Text-editing commands can be one of two types:

- line commands
- command-line commands

Most line commands rearrange or reformat text. They perform tasks such as moving, deleting, copying, and aligning lines or blocks of text. Command-line commands, in addition to rearranging and reformatting text, perform other tasks such as reversing the effects of commands or changing the default case of text.

This section describes commands that you can use in the UNIX environment, but that are not specific to UNIX. You can use these commands in any operating environment that supports text-editing commands.

Dictionary

AUTOADD Command

Controls automatic line addition.

Category: Text editing, command-line command

Syntax

AUTOADD <ON | OFF>

Without Arguments

The AUTOADD command is toggled ON or OFF. Issue the command once to reverse the current setting. If the current setting is ON, then issuing the AUTOADD command changes the setting to OFF. If the current setting is OFF, then issuing the AUTOADD command changes the setting to ON.

Required Arguments

ON
: turns on the AUTOADD command in the window so that lines are added automatically.

OFF
: turns off the AUTOADD command in the window so that lines are not added automatically.

Details

The AUTOADD command controls whether blank lines are added as you scroll past existing text. The number of lines that are added is determined by the setting of the VSCROLL command, which determines the default scroll amount forward or backward.

See Also

Commands:

- "AUTOFLOW Command" on page 453
- "AUTOSPLIT Command" on page 455
- "AUTOWRAP Command" on page 456

AUTOFLOW Command

Controls whether text is flowed when it is included, copied, or pasted.

Category: Text editing, command-line command

Syntax

AUTOFLOW <ON | OFF>

Without Arguments

The AUTOFLOW command is toggled on and off. Issue the command once to reverse the current setting. If the current setting is ON, then issuing the AUTOFLOW command changes the setting to OFF. If the current setting is OFF, then issuing the AUTOFLOW command changes the setting to ON.

Required Arguments

ON
: turns on the AUTOFLOW command in the window so that text flows when it is inserted in the window.

OFF
: turns off the AUTOFLOW command in the window so that text retains its previous position when it is inserted in the window.

Details

The AUTOFLOW command controls whether text inserted with the INCLUDE, PASTE, or COPY command automatically flows. When text is flowed, the left and right boundaries are determined by the settings that were specified with previous executions of the INDENT and BOUNDS commands. The AUTOFLOW command controls all text that is inserted in the window. It does not stop at paragraph boundaries.

Comparisons

The AUTOFLOW command controls whether text inserted in the window flows, while the TF command flows text that is already in the window.

See Also

Commands:

- "AUTOSPLIT Command" on page 455
- "AUTOWRAP Command" on page 456
- "BOUNDS Command" on page 456
- "INDENT Command" on page 467
- "TF Command" on page 484

AUTOSCROLL Command

Specifies how often the Log and Output windows scroll to display output.

UNIX specifics: valid arguments and default values

Syntax

AUTOSCROLL <*n*>

Optional Argument

n

specifies the number of lines that the window should scroll when it receives a line of data that cannot fit.

Details

The AUTOSCROLL command controls the scrolling of lines as they are written to the Log and Output windows. The default value for AUTOSCROLL in the Log and Output windows is `1`. Processing is slower when AUTOSCROLL displays one line at a time. To expedite processing, you can specify a greater AUTOSCROLL value in your autoexec.sas file. Specifying a value of 0 optimizes processing and results in the fastest scrolling (similar to jump scrolling in xterm windows). To add the AUTOSCROLL command to your autoexec.sas file, you must use the DM command. The following example maximizes scrolling in both the Log and Output windows:

```
dm 'output; autoscroll 0; log; autoscroll 0; pgm;';
```

AUTOSPLIT Command

Controls whether text is split at the cursor when you press ENTER or RETURN, or when you are at a carriage return.

Category: Text editing, command-line command

Syntax

AUTOSPLIT <ON | OFF>

Without Arguments

The AUTOSPLIT command acts is toggled on and off. The first time you issue the AUTOSPLIT command, it reverses the current setting. If the current setting is ON, then issuing the AUTOSPLIT command changes the setting to OFF. If the current setting is OFF, then issuing the AUTOSPLIT command changes the setting to ON.

Optional Arguments

ON
turns on the AUTOSPLIT command in the window so that when you press ENTER or RETURN, or when you are at a carriage return, text automatically splits at the cursor.

OFF
turns off the AUTOSPLIT command in the window so that when you press ENTER or RETURN, or when you are at a carriage return, text does not automatically split at the cursor.

Details

The AUTOSPLIT command controls whether text is split at the cursor when you press ENTER or RETURN, or when you are at a carriage return. All text on the line, starting with the character on which the cursor is resting, moves to the left margin of the next line. The cursor is repositioned so that it rests on the first character of the new line.

Comparisons

Entering a carriage return with the AUTOSPLIT command turned on is identical to issuing the TS command with the default numeric argument of 1. The results of a carriage return with the AUTOSPLIT command turned on can be reversed by the TC command or undone with the UNDO command.

See Also

Commands:

- "AUTOSCROLL Command" on page 454
- "AUTOWRAP Command" on page 456
- "TF Command" on page 484
- "TS Command" on page 484

AUTOWRAP Command

Controls whether text is wrapped when it is included, copied, or filed.

Category: Text editing, command-line command

Syntax

AUTOWRAP <ON | OFF>

Without Arguments

The AUTOWRAP command is toggled on or off. The first time you issue the AUTOWRAP command, it reverses the current setting. If the current setting is ON, then issuing the AUTOWRAP command changes the setting to OFF. If the current setting is OFF, then issuing the AUTOWRAP command changes the setting to ON.

Optional Arguments

ON
: turns on the AUTOWRAP command in the window so that text is wrapped when it is inserted in the window, or when it is moved to an external file.

OFF
: turns off the AUTOWRAP command in the window. Depending on the line length, text can be truncated as it is inserted in the window, or when it is moved to an external file.

Details

When the AUTOWRAP command is turned on, you can use the INCLUDE or COPY commands. These commands can insert a file, which has a line length that exceeds the boundaries of the window, in a window. The text in the file is not truncated. Instead, lines in the file are split at word boundaries. Conversely, the AUTOWRAP command enables you to use the FILE command to send text, which has a line length that exceeds the boundaries of a file, to an external file. The text in the file is not truncated. Lines are split at word boundaries. When the AUTOWRAP command is turned off, text can be truncated depending on the line length of the text and of the window or file.

See Also

Commands:

- "AUTOFLOW Command" on page 453
- "AUTOSPLIT Command" on page 455

BOUNDS Command

Sets left and right boundaries when text is flowed.

Category: Text editing, command-line command

Syntax

BOUNDS <*leftright*>

Without Arguments

The BOUNDS command displays a message identifying the current boundary settings.

Optional Arguments

left
: sets the left boundary by column position.

right
: sets the right boundary by column position.

Details

The BOUNDS command resets the left and right boundaries for text. Text is reset by column position and must already be in the window and flowed with the TF command. The BOUNDS command sets the left and right boundaries for text inserted in the window with the INCLUDE, COPY, and PASTE commands when the AUTOFLOW command is turned on. When the AUTOFLOW command is turned on, the left boundary setting is maintained when text is split with the TS command.

For example, specify the following command if you want the text flowed between columns 10 and 60:

```
bounds 10 60
```

Each time text is flowed after this BOUNDS command is issued, the text is flowed between spaces 10 and 60. The text is flowed until you issue another BOUNDS command, or the INDENT command is set to ON.

Setting the INDENT command to ON always overrides the current left boundary setting. To ensure that the left boundary setting is used, set the INDENT command to OFF.

Comparisons

The BOUNDS command affects the behaviors of the TF and TS commands. The BOUNDS command is similar to the INDENT command because both can set the left boundary. However, the BOUNDS command can set the right boundary. When text is flowed, setting the INDENT command to ON always sets the left boundary, which overrides the left boundary that is set by the BOUNDS command.

See Also

Commands:

- "AUTOFLOW Command" on page 453
- "INDENT Command" on page 467
- "TF Command" on page 484
- "TS Command" on page 484

C Command

Copies one line of text.

Category: Text editing, line command

Syntax

C

intervening text

A | B

Without Arguments

The C command copies one line of text to a new position anywhere in a window.

Optional Arguments

A

marks the target position of the line of text to be copied; in this case, after the position where the A argument is typed. You can place the A argument either before or after the line to be copied.

B

marks the target position of the line of text to be copied; in this case, before the position where the B argument is typed. You can place the B argument either before or after the line to be copied.

Comparisons

The C and CC commands enable you to specify a target position for the line of text anywhere in the window. The R and RR commands repeat the line or block of text immediately after it first appears.

See Also

Commands:

- "CC Command" on page 459
- "R Command" on page 477
- "RR Command" on page 479

CAPS Command

Changes the default case of text.

Category: Text editing, command-line command

Syntax

CAPS <ON | OFF>

Without Arguments

The CAPS command is toggled on and off. The first time you issue the CAPS command, it reverses the current setting. If the current setting is ON, then issuing the CAPS command changes the setting to OFF. If the current setting is OFF, then issuing the CAPS command changes the setting to ON.

Required Arguments

ON
: turns on the CAPS command. The case of characters that you enter after you turn on the CAPS command is uppercase. Character strings for the FIND and CHANGE commands are also translated into uppercase unless they are enclosed in quotation marks.

OFF
: turns off the CAPS command. The case of characters that you enter after you turn off the CAPS command is unchanged.

Details

The CAPS command changes the case for text not yet entered, or for text that is modified in a window. If you specify CAPS ON and enter text, the text is changed to uppercase as soon as you press ENTER or RETURN. The setting remains in effect for a window until the SAS session ends, or until the setting is changed by another CAPS command. You can use the WSAVE command to save the setting of the CAPS command beyond your current SAS session.

Comparisons

The CAPS ON command is similar to the CU and CCU commands, and to the CL and CCL commands, which change the case of existing text. However, the CAPS command changes the default case of text, not the case of existing text.

See Also

Command:

- "CL Command" on page 462
- "CCL Command" on page 460
- "CU Command" on page 462
- "CCU Command" on page 461

CC Command

Copies a block of lines of text.

Category: Text editing, line command

Syntax

CC

block of text

CC

intervening text

A | B

Without Arguments

The CC command copies a block of lines of text to a new position anywhere in a window.

Required Arguments

A

marks the target position of the lines of text to be copied; in this case, after the position where the A argument is typed. You can place the A argument either before or after the lines to be copied.

B

marks the target position of the lines of text to be copied; in this case, before the position where the B argument is typed. You can place the B argument either before or after the lines to be copied.

Details

The C and CC commands enable you to specify a target position for the lines of text anywhere in the window. The R and RR commands repeat the block of lines of text immediately after it first appears.

See Also

Commands:

- "C Command" on page 458
- "R Command" on page 477
- "RR Command" on page 479

CCL Command

Changes all characters in designated lines of text to lowercase.

Category: Text editing, line command

Syntax

CCL

block of text

CCL

Without Arguments
The CCL command changes to lowercase all characters in a block of lines of text.

Details

The CL and CCL commands change existing text to lowercase, while the CAPS OFF command makes the default case of text lowercase, which changes the case of new, inserted text. The CU and CCU commands, which change existing text to uppercase, accomplish the opposite of the CL and CCL commands.

See Also

Commands:

- “CL Command” on page 462
- “CAPS Command” on page 458
- “CU Command” on page 462
- “CCU Command” on page 461

CCU Command

Changes all characters in a designated block of lines of text to uppercase.

Category: Text editing, line command

Syntax

CCU

block of text

CCU

Without Arguments
The CCU command changes to uppercase all characters in a block of designated lines of text.

Details

The CU and CCU commands are similar to the CAPS ON command. The CU and CCU commands change existing text to uppercase, while the CAPS ON command makes the default case of text uppercase, which changes the case of new, inserted text. The CL and CCL commands, which change existing text to lowercase, accomplish the opposite of the CU and CCU commands.

See Also

Commands:

- “CU Command” on page 462
- “CAPS Command” on page 458

- "CL Command" on page 462
- "CCL Command" on page 460

CL Command

Changes all characters in a designated line of text to lowercase.

Category: Text editing, line command

Syntax

CL <*n*>

Without Arguments

The CL command changes to lowercase all characters in a designated line of text.

Required Argument

n

specifies the number of lines of text to be changed to lowercase. Follow the *n* argument with a space.

Details

The CL and CCL commands change existing text to lowercase, while the CAPS OFF command makes the default case of text lowercase. The case of new, inserted text is changed. The CU and CCU commands, which change existing text to uppercase, accomplish the opposite of the CL and CCL commands.

See Also

Commands:

- "CCL Command" on page 460
- "CAPS Command" on page 458
- "CU Command" on page 462
- "CCU Command" on page 461

CU Command

Changes all characters in a designated line of text to uppercase.

Category: Text editing, line command

Syntax

CU <*n*>

Without Arguments

The CU command changes to uppercase all characters in a designated line of text.

Optional Argument

n

specifies the number of lines of text to be changed to uppercase. Follow the *n* argument with a space.

Details

The CU and CCU commands are similar to the CAPS ON command. The CU and CCU commands change existing text to uppercase, while the CAPS ON command makes the default case of text uppercase, which changes the case of new, inserted text. The CL and CCL commands, which change existing text to lowercase, accomplish the opposite of the CU and CCU commands.

See Also

Commands:

- "CAPS Command" on page 458
- "CCU Command" on page 461
- "CL Command" on page 462
- "CCL Command" on page 460

CURSOR Command

Moves the cursor to the command line.

Category: Text editing, command-line command

Syntax

CURSOR

Without Arguments

The CURSOR command moves the cursor to the command line. The CURSOR command is designed to be executed with a function key.

Details

The CURSOR command can be used interchangeably with the HOME key.

Comparisons

The CURSOR command has the same results as pressing the HOME key.

D Command

Deletes a designated line.

Category: Text editing, line command

Syntax

D <*n*>

Without Arguments

The D command deletes only the designated line.

Required Argument

n

specifies the number of lines to delete. Follow the *n* argument with a space.

See Also

Commands:

- "DD Command" on page 464

DD Command

Deletes a designated block of lines.

Category: Text editing, line command

Syntax

DD

block of lines

DD

Without Arguments

The DD command deletes a block of lines of text.

See Also

Commands:

- "D Command" on page 464

DICT Command

Includes, releases, and creates an auxiliary dictionary.

Category: Text editing, command-line command

Syntax

DICT INCLUDE *dictionary-name* | FREE *dictionary-name* | CREATE *dictionary-name* <*size*>

Required Arguments

INCLUDE ***dictionary-name***
makes the auxiliary dictionary that is specified available in the current SAS session. Only a one-level name is accepted. The SASUSER.PROFILE catalog is checked first for the dictionary. Then, the SASHELP.BASE catalog is checked. If the auxiliary dictionary is not found, SAS issues an error message. If the auxiliary dictionary is made available from the SASHELP.BASE catalog, no changes to it are saved. If it is made available from the SASUSER.PROFILE catalog, changes to it are saved.

FREE ***dictionary-name***
releases the auxiliary dictionary that is specified. A newly created dictionary is not saved in the SASUSER.PROFILE catalog until you issue the DICT command with the FREE argument, or you end the current interactive windowing task. If the auxiliary dictionary has been modified, the changes are saved when you issue the DICT command with the FREE argument. These changes are saved unless the auxiliary dictionary was made available from the SASHELP.BASE catalog.

CREATE ***dictionary-name***
creates a new auxiliary dictionary as specified. The dictionary is initially empty. When the dictionary is released, it is saved in the SASUSER.PROFILE catalog. Only a one-level name is accepted.

size
specifies the size in bytes of the auxiliary dictionary. The default is 9,808 bytes.

Details

The DICT command includes, releases, and creates an auxiliary dictionary. The SPELL command checks spelling and flags unrecognized words. In addition, the SPELL command can create and update dictionaries.

See Also

Commands:

- "SPELL Command" on page 481

FILL Command

Places fill characters beginning at the current cursor position.

Category: Text editing, command-line command

Syntax

FILL <'*fill-character*'> <*n*>

Without Arguments

The FILL command displays a message identifying the fill character and the number of its repetitions.

Optional Arguments

'*fill-character*'
: specifies a customized character that must be enclosed in single quotation marks. The fill character remains in effect until you change it.

n
: specifies the exact number of fill characters. The number remains in effect until you change it.

Details

The FILL command places fill characters beginning at the current cursor position. The fill characters extend to the end of a line, or to the space before the next non-blank character, whichever occurs first. By default, the fill character is usually an underscore or hyphen. If you use the FILL arguments, you can change the fill character and the number of repetitions.

The FILL command is most easily issued with a function key. To place the fill characters at the cursor position, set one of your function keys to issue the FILL command. Move the cursor to a Program Editor field, and then press the function key. The fill characters are displayed.

The following example shows how you can change the default. Issuing the following command makes the default become 10 question marks:

```
fill '?' 10
```

The changed fill character is in effect for the duration of your SAS session, or until you change it. You can use the WSAVE command to permanently save the setting.

I Command

Inserts one or more blank lines.

Category: Text editing, line command

Syntax

I <A | B> <*n*>

Without Arguments

The I command inserts one or more blank lines immediately after the line on which you issued the command.

Optional Arguments

A

inserts one or more blank lines immediately after the line on which you issued the command. You cannot have any characters between the I command and the A argument.

B

inserts one or more the blank lines immediately before the line on which you issued the command. You cannot have any characters between the I command and the B argument.

n

specifies the number of blank lines to insert. Follow the *n* argument with a space. If you use the A or B argument, the *n* argument is specified last. For example, if line 00009 contains a PROC PRINT statement, the following I command specifies that you want to insert three blank lines before the line of text:

```
ib3  9  proc print data=final.educ;
```

Details

The I command inserts one or more blank lines. By default, the lines are blank. You can define content with the MASK command. The I command is most easily issued with a function key.

Comparisons

You can use the MASK command with the I command. The I command inserts one or more blank lines, which can include content set by the MASK command.

See Also

Commands:

- “MASK Command” on page 475

INDENT Command

Retains left margin indention when text is flowed.

Category: Text editing, command-line command

Syntax

INDENT <ON | OFF>

Without Arguments

The INDENT command is toggled on or off. The first time you issue the INDENT command, it reverses the current setting. If the current setting is ON, then issuing the INDENT command changes the setting to OFF. If the current setting is OFF, then issuing the INDENT command changes the setting to ON. When you reissue the INDENT command, it returns to the previous setting.

Optional Arguments

ON

turns on the INDENT command in the window.

Tips:

The INDENT ON command indents all lines.

When the INDENT command is turned on, and you issue the TF command, all of the lines in the paragraph are indented the same as the first line in the paragraph.

OFF

turns off the INDENT command in the window.

Details

The INDENT command specifies that the current left margin indention is used under the following conditions:

- when existing text in a window is flowed with the TF command
- when text is inserted in a window when the AUTOFLOW command is turned on
- when existing text in a window is split with the TS command

Comparisons

The left boundary can be set by both the INDENT and BOUNDS commands. However, when text is flowed, turning the INDENT command on always determines the left boundary, and overrides the left boundary set by the BOUNDS command.

Examples

Example 1

This example shows four lines of text. The TF command is typed in the number field of the first line. The first line of the paragraph is indented. The INDENT command is set to ON, and the default boundaries are 1 and 50:

```
tf 01       The purpose of Monday's meeting is to review
00002 the documentation plan and gather your responses. Please
00003 send a representative
00004 if you are unable to attend.
```

Example 2

The following example shows the result of pressing ENTER or RETURN to issue the TF command. The indention is used for all of the lines and the right boundary is 50:

```
tf 01       The purpose of Monday's meeting is to review
00002       the documentation plan and gather your responses. Please
00003       send a representative
00004       if you are unable to attend.
```

See Also

Commands:

- “AUTOFLOW Command” on page 453
- “BOUNDS Command” on page 456

- "TF Command" on page 484
- "TS Command" on page 484

JC Command

Centers a designated line of text.

Category: Text editing, line command

Syntax

JC <*n*>

Without Arguments

The JC command centers the designated line of text that contains the line command based on the left and right boundary settings.

Optional Argument

n

specifies the column position on which to center the designated line of text. Follow the *n* argument with a space.

Details

The JC command centers a designated line of text. Unless you specify a numeric argument, centering is based on the current boundary settings set by the BOUNDS command. A numeric argument overrides these boundary settings.

Comparisons

Like the JL, JJL, JR, and JJR commands, the JC and JJC commands align text.

See Also

Commands:

- "JJC Command" on page 469
- "BOUNDS Command" on page 456
- "JL Command" on page 472
- "JJL Command" on page 470
- "JR Command" on page 473
- "JJR Command" on page 471

JJC Command

Centers each line of text independently in a designated block of text.

Category: Text editing, line command

Syntax

JJC

block-of-text

JJC

Optional Argument

block-of-text
specifies a block of text to be centered.

Details

The JJC command centers a designated block of text. Each line in the block is centered independently. Centering is based on the current boundary settings set by the BOUNDS command.

Comparisons

Like the JL, JJL, JR, and JJR commands, the JC and JJC commands align text.

See Also

Commands:

- "JC Command" on page 469
- "BOUNDS Command" on page 456
- "JL Command" on page 472
- "JJL Command" on page 470
- "JR Command" on page 473
- "JJR Command" on page 471

JJL Command

Left-aligns a designated block of text.

Category: Text editing, line command

Syntax

JJL <*n*>

block-of-text

JJL <*n*>

Without Arguments

The JJL command left-aligns a designated block of text. Alignment is based on the left and right boundary settings.

Optional Arguments

n

specifies the column position on which to left-align the designated block of text. By default, the *n* argument is the left boundary setting. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command or in both. If it is specified in both, then the first numeric argument is used.

block-of-text

specifies a block of text to be left aligned.

Details

The JJL command left-aligns a designated block of text. Unless you specify a numeric argument, left-alignment is based on the current boundary settings set by the BOUNDS command. A numeric argument overrides these boundary settings.

Comparisons

Like the JC, JJC, JR, and JJR commands, the JL and JJL commands align text.

See Also

Commands:

- "JL Command" on page 472
- "BOUNDS Command" on page 456
- "JC Command" on page 469
- "JJC Command" on page 469
- "JR Command" on page 473
- "JJR Command" on page 471

JJR Command

Right-aligns a designated block of text.

Category: Text editing, line command

Syntax

JJR <*n*>

block-of-text

JJR <*n*>

Without Arguments

The JJR command right-aligns a designated block of text. Alignment is based on the left and right boundary settings.

Optional Arguments

n

specifies the column position on which to right-align the designated block of text. By default, the *n* argument is the right boundary setting. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command or in both. If it is specified in both, then the first numeric argument is used.

block-of-text

specifies a block of text to be right aligned.

Details

The JJR command right-aligns a designated block of text. Unless you specify a numeric argument, right-alignment is based on the current boundary settings set by the BOUNDS command. A numeric argument overrides these boundary settings.

Comparisons

Like the JC, JJC, JL, and JJL commands, the JR and JJR commands align text.

See Also

Commands:

- "JR Command" on page 473
- "BOUNDS Command" on page 456
- "JC Command" on page 469
- "JJC Command" on page 469
- "JL Command" on page 472
- "JJL Command" on page 470

JL Command

Left-aligns a designated line of text.

Category: Text editing, line command

Syntax

JL <*n*>

Without Arguments

The JL command left-aligns the designated line of text. Alignment is based on the left and right boundary settings.

Optional Argument

n

specifies the column position on which to left-align the designated line of text. By default, the *n* argument is the left boundary setting. Follow the *n* argument with a space.

Details

The JL command left-aligns a designated line of text. Unless you specify a numeric argument, left-alignment is based on the current boundary settings set by the BOUNDS command. A numeric argument overrides those boundary settings.

Comparisons

Like the JC, JJC, JR, and JJR commands, the JL and JJL commands align text.

See Also

Commands:

- "JJL Command" on page 470
- "BOUNDS Command" on page 456
- "JC Command" on page 469
- "JJC Command" on page 469
- "JR Command" on page 473
- "JJR Command" on page 471

JR Command

Right-aligns a designated line of text.

Category: Text editing, line command

Syntax

JR <*n*>

Without Arguments

The JR command right-aligns the designated line of text. Alignment is based on the left and right boundary settings.

Optional Argument

n

specifies the column position on which to right-align the designated line of text. By default, the *n* argument is the right boundary setting. Follow the *n* argument with a space.

Details

The JR command right-aligns a designated line of text. Unless you specify a numeric argument, right-alignment is based on the current boundary settings set by the BOUNDS command. A numeric argument overrides these boundary settings.

Comparisons

Like the JC, JJC, JL, and JJL commands, the JR and JJR commands align text.

See Also

Commands:

- "JJR Command" on page 471
- "BOUNDS Command" on page 456
- "JC Command" on page 469
- "JJC Command" on page 469
- "JL Command" on page 472
- "JJL Command" on page 470

KEYS Command

Enables you to assign function keys to tasks.

Category: Text editing, command-line command

Syntax

KEYS

Comparisons

The KEYS command enables you to access the KEYS window so that you can assign function keys to tasks.

M Command

Moves one line of text.

Category: Text editing, line command

Syntax

M <*n*>

intervening-text

A | B

Optional Arguments

n

specifies the number of lines of text to move. Follow the *n* argument with a space. Without the *n* argument, only the line of text that contains the line command is moved.

A

marks the target position of the line of text to be moved; in this case, after the line where the A argument is typed. You can place the A argument either before or after the line to be moved.

B

marks the target position of the line of text to be moved; in this case, before the line where the B argument is typed. You can place the B argument either before or after the line to be moved.

Details

The M command moves a designated line of text to a new position anywhere in a window.

See Also

Commands:

- "MM Command" on page 476

MASK Command

Defines the contents of one or more new lines.

Category: Text editing, line command

Syntax

MASK

Without Arguments

The MASK command defines, displays, and enables you to edit the contents of one or more new lines that are created by the I command.

Details

The MASK command defines, displays, and enables you to edit the contents of one or more new lines that are created by the I command. The default setting for the new lines are blank lines. To display or edit a new line, type MASK in the number field of the line, and then press ENTER or RETURN. The line with the contents defined by the MASK command is inserted. You can then edit the line. A line with the contents defined by the MASK command is inserted each time you issue the I command.

The contents defined by the MASK command remain in effect for that window throughout your current SAS session unless you change them. To change the contents, type over the text. If you want to return to the default (a blank line), do one of the following tasks:

- blank any characters in the text field of the MASK line.
- issue the CLEAR command with MASK as an argument:

  ```
  clear mask
  ```

  The contents of the MASK line are cleared, and a note appears in the log indicating that the MASK line has been cleared.

  You can use the RESET command or the D command to not display the contents of the MASK command. The MASK command remains in effect even when it is not displayed. For example, the MASK command remains in effect under the following conditions:

  - when you scroll past the MASK line and it is not displayed
  - when you issue the D or RESET command without blanking any characters in the text of the MASK line

  In some windows, such as the Program Editor window, you can use the WSAVE command to permanently save the contents of the MASK command.

See Also

Commands:

- "D Command" on page 464
- "RESET Command" on page 478

MM Command

Moves a block of text.

Category: Text editing, line command

Syntax

MM

block-of-text

MM

intervening-text

A | B

Required Arguments

A

marks the target position of the lines of text to be moved; in this case, after the line where the A argument is typed. You can place the A argument either before or after the lines to be moved.

B

marks the target position of the lines of text to be moved; in this case, before the line where the B argument is typed. You can place the B argument either before or after the lines to be moved.

See Also

Commands:

- "M Command" on page 474

NUMBERS Command

Adds or removes line numbers.

Category: Text editing, command-line command

Syntax

NUMBERS <ON | OFF>

Without Arguments

The NUMBERS command is toggled on or off. The first time you issue the NUMBERS command, it reverses the current setting. If the current setting is ON, then issuing the NUMBERS command changes the setting to OFF. If the current setting is OFF, then issuing the NUMBERS command changes the setting to ON.

Optional Arguments

ON

turns on the NUMBERS command in the window, so that the lines in the Program Editor are numbered.

OFF

turns off the NUMBERS command in the window, so that the lines in the Program Editor are not numbered.

Details

The NUMBERS command adds or removes line numbers for data lines in windows that allow text editing. When you issue the NUMBERS command to remove line numbers, the line numbers disappear and all text appears to shift left. When you issue the NUMBERS command to add line numbers, the numbers are displayed on the left, and all of the text appears to shift right. The alias for the NUMBERS command is NUMS.

R Command

Repeats a designated line.

Category: Text editing, line command

Syntax

R <*n*>

Without Arguments

The R command repeats a designated line one time.

Optional Argument

n

specifies the number of times to repeat the designated line. Follow the *n* argument with a space.

Details

The R command repeats a designated line immediately after the designated line. The default is one time.

Comparisons

The R and RR commands repeat the line or block of lines immediately after the line with the R or RR command. The C and CC commands enable you to copy one or more lines anywhere in a window.

See Also

Commands:

- "RR Command" on page 479
- "C Command" on page 458
- "CC Command" on page 459

RESET Command

Removes any pending line commands.

Category: Text editing, command-line command

Syntax

RESET

Without Arguments

The RESET command removes any pending line commands.

Details

The RESET command removes any pending line commands. It also removes any MASK lines that were created when the MASK command was issued. The display of the MASK line, not the setting of the MASK command, is removed. If you do not want to complete a command on a block of lines (such as the MM or CC command), you can issue the RESET command to remove the pending command.

Comparisons

The RESET command has the same result as the D command. The RESET command also removes the display of MASK lines from the MASK command.

See Also

Commands:

- "D Command" on page 464

RR Command

Repeats a block of lines.

Category: Text editing, line command

Syntax

RR <*n*>

block of text

RR <*n*>

Without Arguments

The RR command repeats the block of lines one time.

Required Argument

n

specifies the number of times to repeat the designated block of lines. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command or in both. If it is specified in both, the first numeric argument is used.

Details

The RR command repeats a designated block of lines immediately after the designated block of lines. The default is one time.

Comparisons

The R and RR commands repeat the line or block of lines immediately after the line with the R or RR command. The C and CC commands enable you to copy one or more lines anywhere in a window.

See Also

Commands:

- "R Command" on page 477
- "C Command" on page 458

- "CC Command" on page 459

> Command (Shift Right)

Shifts to the right a designated line of text.

Category: Text editing, line command

Syntax

> <*n*>

Without Arguments

The > command shifts a designated line of text one space to the right.

Optional Argument

n

specifies the number of spaces that the designated line of text shifts. Follow the *n* argument with a space.

Details

The > command shifts a designated line of text one or more spaces to the right. The line shifts the number of spaces that you specify with the *n* argument, or the line shifts at the left window border, whichever is less. This text-shift command does not lose characters when shifting.

Comparisons

The < and << commands shift text in the opposite direction from the > and >> commands. The),)), (, and ((commands are similar text-shift commands, which, depending on the extent of the shift, can lose characters.

See Also

Commands:

- ">> Command (Shift Right Block)" on page 480

>> Command (Shift Right Block)

Shifts to the right a designated block of text.

Category: Text editing, line command

Syntax

>> <*n*>

block of text

>> <*n*>

Without Arguments

The >> command shifts a designated block of lines of text one space to the right.

Optional Argument

n

specifies the number of spaces that the designated block of lines of text shifts. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command, or in both. If it is specified in both, the first numeric argument is used.

Details

The >> command shifts a designated block of lines of text one or more spaces to the right. The block of lines of text shifts the number of spaces that you specify with the *n* argument, or the block shifts at the left window border, whichever is less. This text-shift command does not lose characters when shifting.

Comparisons

The < and << commands shift text in the opposite direction from the > and >> commands. The),)), (, and ((commands are similar text-shift commands, which, depending on the extent of the shift, can lose characters.

See Also

Commands:

- "> Command (Shift Right)" on page 480

SPELL Command

Checks spelling and flags unrecognized words.

Category: Text editing, command-line command

Syntax

SPELL <ALL <SUGGEST>>

SPELL <NEXT | PREV | SUGGEST>

SPELL <REMEMBER <*dictionary-name*>>

Without Arguments

The SPELL command checks the first word if the cursor is positioned on the command line. Otherwise, the SPELL command checks the word on which the cursor is positioned. Assign a function key to the SPELL command by using the KEYS command. Use the function key to check the spelling of the word on which the cursor is positioned. If the word is recognized, the message "OK" appears. Otherwise, a message appears indicating that the word is unrecognized.

Optional Arguments

ALL

checks the spelling of all words. If all words are recognized, a message indicates that no unrecognized words have been found.

If a word is unrecognized, the SPELL: Unrecognized Words window appears, listing the unrecognized words and their corresponding line numbers. The window initially displays a blank field for the dictionary that you want to specify. Enter a new dictionary name, or the name of an existing dictionary. Select **Tools ⇨ Remember** to add the unrecognized words to the dictionary.

Tip: Specify the SPELL ALL SUGGEST command to display the SPELL: Suggestions window for each unrecognized word that is found.

SUGGEST

invokes the SPELL: Suggestions window, which displays the last unrecognized word and its line number, and suggestions for changing the unrecognized word. In this window, you can select **Tools ⇨ Remember** to add the unrecognized word to a dictionary. The window will then close, you are returned to the previous window, and a message indicates that the word is now recognized.

You can change the unrecognized word by positioning your cursor on a suggestion, and then pressing ENTER. The suggested word is highlighted. Select **Tools ⇨ Replace**. When you return to the Program Editor window, you will see that the unrecognized word has been changed. If you want to replace all occurrences of the recognized word, first, position the cursor on the phrase **ALL OCCURRENCES**, and then press ENTER. The phrase ALL OCCURRENCES is highlighted. When you return to the Program Editor window, you will see that all occurrences of the unrecognized word have been changed.

Alias: ?

NEXT

finds the next unrecognized word, based on the current cursor position. If all words from the current cursor position to the end of the file are recognized, a message indicates that the end of the file was reached. Otherwise, a message indicates that a word is unrecognized, and the cursor is positioned on the unrecognized word.

PREV

finds the previous unrecognized word, based on the current cursor position. If all words from the current cursor position to the beginning of the file are recognized, a message indicates that the beginning of the file was reached. Otherwise, a message indicates that a word is unrecognized, and the cursor is positioned on the unrecognized word.

REMEMBER *dictionary-name*

adds the last unrecognized word to an auxiliary dictionary, where *dictionary-name* is the name of an auxiliary dictionary. A message indicates that the word has been added to an auxiliary dictionary. If you are using only one auxiliary dictionary, you can omit *dictionary-name*. If no auxiliary dictionary is specified, and *dictionary-name* is omitted, the unrecognized word is saved in a temporary dictionary in the current SAS session only.

You can highlight a word from the SPELL: Unrecognized Words window or from the SPELL: Suggestions window, and then select **Tools ⇨ Remember**. The word that you added is now recognized.

Alias: ADD

Details

The SPELL command checks spelling and flags unrecognized words. You can use the SPELL command to do the following tasks:

- See suggestions for unrecognized words.
- Add unrecognized words to an auxiliary dictionary.
- Replace unrecognized words with suggestions.

The SPELL command checks words with a default dictionary. However, you can specify one or more auxiliary dictionaries to use in addition to the default dictionary.

Any dictionary that you create is stored in your SASUSER.PROFILE catalog. If you update a dictionary using the SPELL REMEMBER command, updates are saved to a temporary dictionary in the current SAS session. The temporary dictionary is created if you do not specify a dictionary name. If you specify a dictionary from the SASHELP.BASE catalog, updates are saved in that dictionary.

Comparisons

The SPELL command checks spelling and flags unrecognized words, and the DICT command includes or creates an auxiliary dictionary. The SPELL command can also create and update auxiliary dictionaries. Use the SPELL command to create a permanent auxiliary dictionary. The word list that is used by the SPELL command acts as a record of the words that are contained in the auxiliary dictionary.

See Also

Commands:

- "DICT Command" on page 465

TC Command

Connects two lines of text.

Category: Text editing, line command

Syntax

TC

Without Arguments

The TC command connects two lines of text.

Details

The TC command connects two lines of text. To connect two lines of text, type TC in the number field of a line, and press Enter or Return. The text from the second line moves to the first line. No space appears between text on the first line and text on the second line. To create a space between the last word of the first line and the first word of the second line, start the text of the second line in the second column.

The command does not truncate text.

Comparisons

The TC command is the opposite of the TS command, which splits text at the cursor. It is similar to the TF command, except that it breaks text at boundaries instead of flowing text in a paragraph by removing trailing blanks.

See Also

Commands:

- "TF Command" on page 484
- "TS Command" on page 484

TF Command

Flows text to a blank line or to the end of the text.

Category: Text editing, line command

Syntax

TF <A> <*n*>

Without Arguments

The TF command flows text to the first blank line, or to the end of the text, whichever comes first, based on left and right boundary settings.

Optional Arguments

A

flows text in a paragraph to the end of the text by removing trailing blanks, continuing over, but not deleting, blank lines. This argument, like the numeric argument, must be specified on the same line as the TF command. You cannot have any characters between the TF command and the A argument.

n

specifies a right boundary to temporarily override the right boundary set by the BOUNDS command. Follow the *n* argument with a space.

TS Command

Splits text at the cursor.

Category: Text editing, line command

Syntax

TS <*n*>

Without Arguments

The TS command splits the line of text at the cursor, and moves the remaining text to a new line.

Optional Argument

n

specifies how many lines down to move the remaining text. The default is one line. Follow the *n* argument with a space.

Details

The TS command splits the line of text at the cursor, and moves the remaining text to a new line starting at the left margin. If you specify a numeric argument, the TS command moves the text down the number of lines specified. With the AUTOFLOW command turned on, the TS command uses the left boundary that is specified by the BOUNDS command. If the INDENT command is turned on, the TS command uses the current indention at the left margin. With the AUTOFLOW command turned off, the left boundary and the current indention at the left margin are reset.

This example shows the effect of splitting two statements in a SAS program and placing each statement on a separate line. This example shows the text after you type the TS command on line 0001 and position the cursor after the first statement, and before you press Enter or Return:

```
ts 01 proc print data=temp; run;
```

When you press Enter or Return, the following result is displayed:

```
00001 proc print data=temp;
00002 run;
```

Comparisons

The TS command, with the default numeric argument of 1, is the same as entering a carriage return or pressing Enter or Return with the AUTOSPLIT command turned on. The TS command contrasts with the TC command, which connects two lines of text, and the TF command, which flows text to a blank line or to the end of the text. With the AUTOFLOW command turned on, the TS command is affected by both the BOUNDS and INDENT commands.

See Also

Commands:

- “AUTOSPLIT Command” on page 455
- “I Command” on page 466
- “TC Command” on page 483
- “TF Command” on page 484

UNDO Command

Cancels an action.

Syntax

UNDO

Without Arguments

The UNDO command cancels the most recent action in an active window that allows text editing.

Details

The UNDO command cancels the most recent action in an active window that allows text editing. The action must be a command that enters or modifies text. If you want to undo more than one action, you must continue to issue the UNDO command. Actions are undone one at a time, starting with the most recent action and moving backward.

Note: The UNDO command cannot undo the SUBMIT command. It cannot reverse the effects of submitted SAS statements.

Comparisons

Although you cannot undo the SUBMIT command, you can use the RECALL command to recall submitted statements back to the Program Editor window.

If you use the CC command to copy and paste a block of text, and then you issue the UNDO command, the block of text that you copied and pasted is deleted. If you use the DD command to delete a block of text, and then you issue the UNDO command, the block of text that you deleted is restored.

< Command

Shifts to the left a designated line of text.

Category: Text editing, line command

Syntax

< <*n*>

Without Arguments

The < command shifts a designated line of text one space to the left.

Optional Argument

n

specifies the number of spaces that the designated line of text shifts. Follow the *n* argument with a space.

Details

The < command shifts a designated line of text one or more spaces to the left. The line shifts the number of spaces that you specify with the *n* argument, or the line shifts at the left window border, whichever is less. This text-shift command does not lose characters when shifting.

Comparisons

The > and >> commands shift text in the opposite direction from the < and << commands. The),)), (, and ((commands are similar text-shift commands, which, depending on the extent of the shift, can lose characters.

See Also

Commands:

- "<< Command" on page 487

<< Command

Shifts to the left a designated block of text.

Category: Text editing, line command

Syntax

<< <*n*>

block-of-text

<< <*n*>

Without Arguments

The << command shifts a designated block of lines of text one space to the left.

Optional Argument

n

specifies the number of spaces that the designated block of lines of text shifts. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command, or in both. If it is specified in both, the first numeric argument is used.

Details

The << command shifts a designated block of lines of text one or more spaces to the left. The block of lines of text shifts the number of spaces that you specify with the *n* argument, or the block shifts at the left window border, whichever is less. This text-shift command does not lose characters when shifting.

Comparisons

The > and >> commands shift text in the opposite direction from the < and << commands. The),)), (, and ((commands are similar text-shift commands, which, depending on the extent of the shift, can lose characters.

> Command

Shifts to the right a designated line of text.

Category: Text editing, line command

Syntax

> <*n*>

Without Arguments

The > command shifts a designated line of text one space to the right.

Optional Argument

n

specifies the number of spaces that the designated line of text shifts. Follow the *n* argument with a space.

Details

The > command shifts a designated line of text one or more spaces to the right. The line shifts the number of spaces that you specify with the *n* argument, or the line shifts at the left window border, whichever is less. This text-shift command does not lose characters when shifting.

Comparisons

The < and << commands shift text in the opposite direction from the > and >> commands. The),)), (, and ((commands are similar text-shift commands, which, depending of the extent of the shift, can lose characters.

See Also

Commands:

- ">> Command" on page 488

>> Command

Shifts to the right a designated block of text.

Category: Text editing, line command

Syntax

>> <*n*>

block-of-text

>> <*n*>

Without Arguments

The >> command shifts a designated block of lines of text one space to the right.

Optional Argument

n

specifies the number of spaces that the designated block of lines of text shifts. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command, or in both. If it is specifies in both, the first numeric argument is used.

Details

The >> command shifts a designated block of lines of text one or more spaces to the right. The block of lines of text shifts the number of spaces that you specify with the *n* argument, or the block shifts at the left window border, whichever is less. This text-shift command does not lose characters when shifting.

Comparisons

The < and << commands shift text in the opposite direction from the > and >> commands. The),)), (, ((commands are similar text-shift commands, which, depending on the extent of the shift, can lose characters.

See Also

Commands:

- "> Command" on page 487

(Command

Shifts to the left one designated line of text.

Category: Text editing, line command

Syntax

(<*n*>

Without Arguments

The (command shifts a designated line of text one space to the left.

Optional Argument

n

specifies the number of spaces that the designated line of text shifts. The default is one space. Follow the *n* argument with a space.

Details

The (command shifts a designated line of text one or more spaces to the left. If the shift extends past the beginning of the current line, characters are lost.

Comparisons

The) and)) commands shift text in the opposite direction from the (and ((commands. The <, <<, >, and >> commands are similar text-shift commands, but they do not lose characters when shifting.

See Also

Commands:

- "((Command" on page 490

((Command

Shifts to the left a designated block of lines of text.

Category: Text editing, line command

Syntax

((*<n>*

block-of-text

((*<n>*

Without Arguments

The ((command shifts a designated block of lines of text one space to the left.

Optional Argument

n

specifies the number of spaces that the designated block of lines of text shifts. The default is one space. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command, or in both. If it is specified in both, the first numeric argument is used.

Details

The ((command shifts a designated block of lines of text one or more spaces to the left. If the shift extends past the beginning of the current line, characters are lost.

Comparisons

The) and)) commands shift text in the opposite direction from the (and ((commands. The <, <<, >, and >> commands are similar text-shift commands, but they do not lose characters when shifting.

See Also

Commands:

- "(Command" on page 489

) Command

Shifts to the right one designated line of text.

Category: Text editing, line command

Syntax

) <*n*>

Without Arguments

The) command shifts a designated line of text one space to the right.

Optional Argument

n

specifies the number of spaces that the designated line of text shifts. The default is one space. Follow the *n* argument with a space.

Details

The) command shifts a designated line of text one or more spaces to the right. If the shift extends past the end of the current line, characters are lost.

Comparisons

The (and ((commands shift text in the opposite direction from the) and)) commands. The <, <<, >, and >> commands are similar text-shift commands, but they do not lose characters when shifting.

See Also

Commands:

- “)) Command” on page 491

)) Command

Shifts to the right a designated block of lines of text.

Category: Text editing, line command

Syntax

)) <*n*>

block-of-text

)) <*n*>

Without Arguments

The)) command shifts a designated block of lines of text one space to the right.

Optional Argument

n

specifies the number of spaces that the designated block of lines of text shifts. The default is one space. Follow the *n* argument with a space. You can specify the numeric argument in the beginning or ending line of the block command, or in both. If it is specified in both, the first numeric argument is used.

Details

The (and ((commands shift text in the opposite direction from the) and)) commands. The <, <<, >, and >> commands are similar text-shift commands, but they do not lose characters when shifting.

See Also

Commands:

- “) Command” on page 491

Glossary

active window
a window that is open and displayed, and to which keyboard input is directed. Only one window can be active at a time.

aggregate syntax
a convenient way of referring to individual files in a single directory or folder. Instead of assigning a unique fileref to each file, you assign a fileref to the directory or folder. Then, to refer to a specific file in that folder, you enclose the filename in parentheses following the fileref. Aggregate syntax is used in the FILE, INFILE, and %INCLUDE statements.

background process
in UNIX environments, a process that executes independently of the shell. When a command is executing in a background process, you can enter other commands or start other background processes without waiting for your initial command to finish executing.

batch mode
a method of executing SAS programs in which a file that contains SAS statements plus any necessary operating environment commands is submitted to the computer's batch queue. After you submit the program, control returns to your computer, and you can perform other tasks. Batch mode is sometimes referred to as running in the background. The program output can be written to files or printed on an output device.

buffer
an area of computer memory that is reserved for use in performing input/output (I/O) operations.

cat
a UNIX command that means concatenate. This command is commonly used to list file contents and to concatenate files.

catalog
See SAS catalog.

class name
a name that provides a way to group individual X resources together. For example, DMSboldFont and DMSFont are two separate X resources, but they are both part of the Font class.

client
an application that requests either resources or services from a server, possibly over a network.

command interpreter
a program (such as the shell) that translates your commands into a language understood by the computer.

command line
the location in any SAS windowing environment window designated with Command ===>.

command prompt
the symbol after which you enter operating system commands. In UNIX environments, different shells use different command prompts. The default command prompt for the Bourne shell and the Korn shell is $, and the default prompt for the C shell is %.

container window
any SAS window that contains interior windows.

converting SAS files
the process of changing the format of a SAS file from the format that is appropriate for one version of SAS to the format that is appropriate for another version in the same operating environment.

current directory
the directory that you are working in at any given time. When you log on, your current directory is the starting point for relative pathnames.

device driver
a program that controls the interaction between a computer and an external device such as a printer or a disk drive.

directory
a special type of file in UNIX operating environments that contains a group of files or other directories.

download
to copy a file from a remote host to a local host.

encoding
a mapping of a coded character set to code values.

environment variable
in UNIX environments, a shell variable whose value or values can be accessed by any program that is executed from that shell. The shell assigns default values to some environment variables. For example, the type of terminal and the type of command prompt are specified by the default values of two environment variables.

error message
a message in the SAS log or Message window that indicates that SAS was not able to continue processing the program.

external file
a file that is created and maintained by a host operating system or by another vendor's software application. SAS can read data from and route output to external files. External files can contain raw data, SAS programming statements, procedure output, or output that was created by the PUT statement. A SAS data set is not an external file.

file descriptor
under UNIX operating systems, a nonnegative integer identifier used to refer to a file opened for reading or writing or both.

file extension
the classification of a file in a directory that identifies what type of information is stored in the file. For example, .sas7bcat is the file extension for UNIX, and .pdf is the file extension for Adobe Acrobat.

font
a typeface with a specific character shape, spacing, weight, and size. The characters in a font can be figures, symbols, or alphanumeric.

foreground process
in UNIX environments, a process that executes while you wait for the command prompt to reappear. You cannot execute additional commands while the initial command is being executed in a foreground process.

home directory
the directory in which a user is placed after logging in. The home directory is also called the login directory.

I/O time
an abbreviation for input/output time. I/O time is the time the server spends on moving data from storage into memory for work (input time), and moving the result out of memory to storage or to a display device, such as a terminal or a printer (output time).

index
a component of a SAS data set that enables SAS to access observations in the SAS data set quickly and efficiently. The purpose of SAS indexes is to optimize WHERE-clause processing and to facilitate BY-group processing.

interactive line mode
a method of running SAS programs in which you enter one line of a SAS program at a time at the SAS session prompt. SAS processes each line immediately after you press the ENTER or RETURN key. Procedure output and informative messages are returned directly to your display device.

libref
a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library.

local SAS session
a SAS session running on the local host. The local session accepts SAS statements and passes those that are remotely submitted to the remote host for processing. The local session manages the output and messages from both the local session and the remote session.

login directory
See home directory.

lp
under UNIX, a line-printer command, commonly used to direct output to a printer destination via the line printer daemon.

member
a type of SAS file in a SAS library. Types of SAS files include a data set, a view, a catalog, a stored program, and an access descriptor.

memory
the size of the work area that the central processing unit (CPU) must devote to the operations in a program.

motif
an X Window System graphical user interface (GUI) that is used in the UNIX environment.

network
an interconnected group of computers.

noninteractive mode
a method of running SAS programs in which you prepare a file of SAS statements and submit the program to the operating system. The program runs immediately and comprises your current session.

path
the route through a hierarchical file system that leads to a particular file or directory.

pathname
in UNIX operating systems, a filename that specifies all of the directories that lead to a particular file in the file hierarchy.

PID
See process ID.

pipe
under UNIX operating systems and derivatives, the facility that links one command to another so that the standard output of one becomes the standard input of the other.

process ID
a unique number that is assigned to each process by the operating system. Short form: PID.

protocol
a set of rules that govern data communications between computers and peripheral devices.

redirect
to direct output to a destination other than standard output or to read input from a source other than standard input.

remote browser server
a software agent that runs on your desktop and sends URLs to the browser to display.

remote browsing
a mechanism that is used by SAS to display HTML information (for example, help text and ODS HTML output) using a browser on your desktop.

SAS catalog
a SAS file that stores many different kinds of information in smaller units called catalog entries. A single SAS catalog can contain different types of catalog entries.

SAS library
a collection of one or more files that are recognized by SAS and that are referenced and stored as a unit. Each file is a member of the library.

sasauth
a SAS subprocess that performs user authentication and identification functions. The sasauth process is located in the !SASROOT/utilities/bin directory.

sasperm
a SAS subprocess that determines resource access privileges for users.

sasroot
a term that represents the name of the directory or folder in which SAS is installed at your site or on your computer.

sasuser profile catalog
a SAS catalog in which SAS stores information about attributes of your SAS windowing environment. For example, this catalog contains function-key definitions, fonts for graphics applications, window attributes, and other information that is used by interactive SAS procedures.

sequential access
a method of file access in which the records are read or written one after the other from the beginning of the file to the end.

server
software that provides either resources or services to requesting clients, possibly over a network.

session
a single period during which a software application is in use, from the time the application is invoked until its execution is terminated.

session gravity
in the X window interface to SAS, the resource that controls the region of the workstation display in which SAS attempts to place its windows.

shell
a UNIX command interpreter. Sample shells are sh, csh, and ksh.

shell script
a file containing commands that can be read and executed by the shell.

special file
under UNIX operating systems, an interface to an input or output device. Writing to or reading from the file activates the device.

standard error
under UNIX operating systems, the destination of a program's error messages. Standard error is also called stderr.

standard input
the primary source of data going into a command. Standard input comes from the keyboard unless it is being redirected from a file or piped from another command. Standard input is also called stdin.

standard output
the primary destination of data coming from a command. Standard output goes to the display unless it is being redirected to a file or piped to another command. Standard output is also called stdout.

swap
to move data or program code from a computer system's main memory to a storage device such as a hard disk, or vice versa.

toggle
an option, parameter, or other mechanism that enables you to turn on or turn off a processing feature.

toolbox
a part of the SAS windowing environment in which you can place icons that you can associate with SAS commands or macros. Selecting an icon executes its associated command or string of commands.

toolset
a set of predefined tools that is associated with an application. Toolsets make it easier for individual users to customize their application toolboxes.

Universal Printing
a feature of SAS software that enables you to send SAS output to PDF, Postscript, GIF, PNG, SVG, and PCL files, as well as directly to printers. The Universal Printing system also provides many options that enable you to customize your output, and it is available in all of the operating environments that SAS supports.

working directory
the directory in which a software application is invoked.

X resource
a characteristic of a window interface, such as font type, font size, color, gravity, and window size.

X resource file
in the X Window System, a file that stores attribute specifications for the windowing environment, such as color, gravity, font types and sizes, and window sizes.

X server
in an X Window System, the program that mediates access to the display, mouse, and keyboard from one or more application client programs.

X window system
a graphical windowing system that was developed at the Massachusetts Institute of Technology.

Index