CHAPTER 1

Getting Started with the SAS System in UNIX Environments

Introduction

Note: This book assumes that you have a basic knowledge of UNIX. △

Under UNIX, you can use the windowing environment, or you can run SAS in batch mode or interactive line mode. This chapter briefly describes each of these methods of running the SAS System under UNIX. For a more detailed description of the these methods, refer to the online help.

Selecting a Mode of Execution

UNIX is a multitasking 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. 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 tasks without waiting for your initial command to execute. You can run the SAS System in the foreground or in the background.

Note: Both the C shell and the Korn shell include commands that allow you to move jobs among three possible states: running in the foreground, running in the background, and suspended. △

You can run SAS in the SAS windowing environment, in interactive line mode, or in batch mode.

Windowing environment

You interact with SAS through windows using your keyboard, mouse, pull-down menus, pop-up menus, and icons. The windowing environment includes, but is not limited to, the Explorer, Program Editor, Output, Log, and Results windows. Explorer is a windowing environment for managing basic SAS software tasks such as viewing and managing data sets, libraries and members, applications, and output. The SAS Explorer is a central access point from which you can

- 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 the SAS System.

The Program Editor, Log, and Output windows enable you to edit and execute SAS programs and display output.

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).

Interactive line mode

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

- 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.

Batch mode

To run SAS in batch mode, you specify your SAS application name in the SAS 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. If you start your application with one of these UNIX commands, then you log off of your system and 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.

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

Starting SAS Sessions

Note: Before you start your SAS session, review the different methods for interrupting and terminating your SAS session (see "Interrupting or Terminating Your SAS Session" on page 10). Also, if you cannot stop your session, contact your system administrator; do not turn off your machine, especially if your machine is part of a network. △

Starting a SAS Session

The command that you use to invoke your SAS session is defined during the SAS installation process and is added to the list of commands that are recognized by the operating environment. Ask your system administrator what the command is that invokes SAS at your site. At many sites, the command to invoke SAS is simply **sas**, but a different command may have been defined during the SAS installation process at your site. This book assumes that the SAS System is invoked by the **sas** command.

The general form of the SAS command is as follows:

sas *<–option1...-option-n> <filename>*

You can use these arguments with the SAS command:

-option1 ... -option-n
: specifies a SAS system option to configure your session or an X command line option. See Chapter 17, "SAS System Options," on page 253 and "X Command Line Options" on page 6 for more information. If you omit any options (either on the command line or in the configuration file), the SAS System's (or site-specific) default options are in effect.

filename
: is 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 in your current directory and has a **.sas** extension, you can omit its extension. If the file is not in the current directory, specify its full pathname.

For example, to invoke an interactive SAS session, without specifying any SAS system options, enter

```
sas
```

The execution mode will depend on your default settings.

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

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

To run a SAS program and pass parameters to it, enter

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

The value **A B C** is assigned to the SYSPARM macro variable, which can be read by the program **progparm.sas**.

X Command Line Options

When you invoke some X clients, such as the SAS System, you can use command line options that are passed to the X Window System. The following list describes the X command line options that are available when you invoke a SAS session from the command prompt. In general, you should specify X Window System options after SAS options on the command line.

-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, you might enter

```
-display wizard:0.0
```

or

```
-display 10.22.1.1:0
```

-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 the title up to six characters long for your SAS session window. To use multiple words in the title, enclose the words in single or double quotes. 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'
```

The SAS System does not support the following X command line options because their functionality is not applicable to the SAS System or is provided by SAS resources. Refer to "Using X Resources to Customize the Motif Interface" on page 38 for more information on SAS resources.

-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 69 for a description of specifying the color of window borders.

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

-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. Refer to "Defining Colors and Attributes for Window Elements (CPARMS)" on page 69 for a description of specifying reverse video.

-selectionTimeout
Timeout length is specified by the **SAS.selectTimeout** resource.

-synchronous, +synchronous
The XSYNC command controls the X synchronization.

-xn1language
This option is ignored.

Running the SAS System on a Remote Host

When you invoke the SAS System 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 machine's name. For example, if your machine 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 the SAS System 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.

To run the SAS System on a remote host, follow these steps:

1 Make sure that the clients running on the remote host have permission to connect to your server. Most systems control this by using the **xhost** client. Other systems control access through a session manager. 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 +
```

To run this command automatically each time you log in, enter this command in a file named **.xhost**.

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

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 do this 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 **setenv** command:

```
setenv DISPLAY green:0.0
```

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

b Use the **-display** option. For example:

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

Note: This option is a command line option for the X Window system, not for the SAS System. Specifying this option in a SAS configuration file or in the SASV8_OPTIONS environment variable may cause problems when running other interfaces. △

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
```

If SAS cannot establish a connection to your display, it prints a message that indicates the nature of the problem and then terminates. Make sure that you have brought up the SAS session correctly. You may 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 the SAS System to bring your session up in line mode.

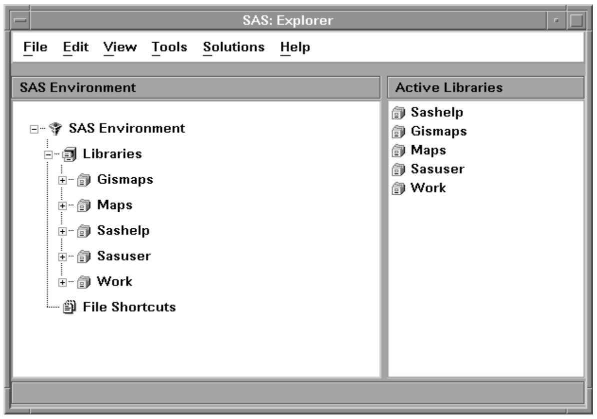
If you are unable to invoke SAS, try running another application such as **xclock**. If you cannot run the application, you might need to contact your system administrator for assistance.

Using SAS with the Windowing Environment

Your SAS session may default to the windowing environment interface. If not, you can invoke the Explorer only by specifying the EXPLORER system option:

```
sas -explorer
```

SAS opens the Explorer window.



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

```
sas -dms
```

You can use the DMSEXP system option to open the Program Editor, Output, Log, and Results windows and the Explorer:

```
sas -dmsexp
```

SAS also opens the toolbox from which you can open additional SAS windows. For information on using the windowing environment, refer to the online help. For more information on the toolbox, refer to Chapter 2, "Working in the SAS System Windowing Environment," on page 21.

To end your SAS session, enter the BYE or ENDSAS command on the command line or select

File ► Exit...

from the pull-down menu of the session that you want to end.

Using SAS in Batch Mode

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

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

The SAS command uses **.sas** by default, so if your filename ends with **.sas**, you do not need to include the extension in the SAS command. (Also, you do not need to specify the SYSIN option as in some other platforms.)

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

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

```
$ at 2am
sas weekly.rpt
<control-D>
warning: commands will be executed using /usr/bin/sh
job 8400.a at Wed Jun 10 02:00:00 1998
$
```

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

```
at 2am < cmdfile.sas
```

The SAS System sends the output to a file that has the same name as the program and an extension of **.lst**, and the log goes to a file with an extension of **.log**. Both of these files are written to your current directory. Refer to the man pages for these commands for more information on submitting jobs to the batch queue. For more details on routing output, see Chapter 6, "Routing Output," on page 125.

Note: If your program contains statements that start an interactive procedure such as the FSEDIT procedure, you will need to run your program in the foreground. △

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

```
INFILE STDIN;
```

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

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

For details on using external files, see Chapter 5, "Using External Files and Devices," on page 103. See also "File Descriptors in the Bourne and Korn Shells" on page 114 for another way to have a SAS program read data from an external file.

Using SAS in Interactive Line Mode

To start an interactive line mode session, invoke the SAS System 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.

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

You can end the session by pressing the EOF key (usually CTRL+D; see "Using Control Keys" on page 11) or by issuing the ENDSAS statement:

```
endsas;
```

Interrupting or Terminating Your SAS Session

There are three ways to interrupt or terminate your SAS session:

Enter the **kill** command.
The **kill** command sends an interrupt or quit signal to the SAS System, depending on which signal you specify. You can use the **kill** command to interrupt or terminate a SAS session running in any mode.

Press the interrupt or quit control key.
The control keys send the same signals as the **kill** command. However, they 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 batch job that has been submitted with the **batch**, **at**, **nohup**, or **cron** command.

Use the session manager.
Press the interrupt or terminate buttons in the session manager window. The session manager is available only when you run SAS in the windowing environment.

Using the kill Command

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 quit signal, specify **-SIGQUIT**. Use the **ps** command to determine the process identification number (*pid*) of the SAS session that you want to interrupt or terminate.

For example, suppose you want to stop a SAS job running in the background. First, issue the **ps** command to determine the PID of the SAS job.

```
> ps
    PID TTY       TIME COMMAND
   2103 ttyu0     0:00 motifxsa
   2111 ttyu0     0:01 sas
   2116 ttyu0     0:00 ps
   3856 ttyu2     0:03 ksh
```

Four PIDs appear, but only one is for a SAS program. (**motifxsa** is the SAS session manager. See “Using the SAS Session Manager (motifxsassm)” on page 33 for more information.) Therefore, to send the interrupt signal to that SAS program, you would issue this command:

```
kill -SIGINT 2111
```

The SAS System replies with a prompt:

```
Press Y to cancel submitted statements,
N to continue.
```

For more information, refer to the UNIX man pages for the **ps** and **kill** commands.

Using Control Keys

Control keys enable you to interrupt or terminate your session by simply 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.

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 **stty** UNIX man page before using it. Usually, one of these forms of the command will print all of the current terminal settings:

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

The output you see should contain lines similar to these:

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

The caret (^) stands for the CTRL key. In this example, control-C is the interrupt key and control-\ is the quit key.

Using the Session Manager

If you invoke SAS in the windowing environment, you can use the session manager to interrupt or terminate your SAS session. The session manager is automatically iconified when you start SAS. To interrupt or terminate your SAS session, open the session manager window and press $\boxed{\text{Interrupt}}$ or $\boxed{\text{Terminate}}$.

Executing Operating System Commands from Your SAS Session

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" on page 246 for information on executing commands asynchronously.

When you execute one or more UNIX commands *synchronously*, then 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. See "CALL SYSTEM" on page 197 and "Macro Statements" on page 215 for more information.

Executing a Single UNIX Command

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 you specify includes a semicolon, you must enclose the UNIX command in a macro quoting function. Refer to *SAS Macro Language: Reference* for more information on quoting functions. △

When you specify only one command, the SAS System checks to see whether the command is **`cd`**, **`pwd`**, or **`setenv`** 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. These three commands are built into the SAS System because they affect the environment of the current SAS session. When executed by the SAS System, 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.

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

```
x ls -l;
```

Inside a DATA step, you could use the CALL SYSTEM routine to execute **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.

Executing Several 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. Refer to *SAS Macro Language: Reference* for more information on quoting functions. △

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

For example, 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.

Starting a Shell

If you are not running in the SAS windowing environment, you can start a shell by not specifying any UNIX commands in the X statement:

X;

* The shell used depends on the SHELL environment variable.

The SAS System responds with

```
Enter 'exit' to return to your SAS session.
```

SAS then starts a shell.

Enter any UNIX commands. When you are ready to return to the SAS session, enter the **exit** command.

Even if you changed directories while in the shell, you will be in the same directory as when you started the shell.

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, the SAS System 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 Session

You can customize your SAS session by defining configuration and/or 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 data libraries. Refer to *SAS Language Reference: Dictionary* for a complete description of system options.)

The differences between configuration files and autoexec files are

- Configuration files can contain only SAS system option settings, while autoexec files can contain any valid SAS statement. For example, you may 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 data libraries and external files that you use most often.
- Configuration files are processed *before* the SAS System initializes, while autoexec files are processed immediately *after* the SAS System 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.

The configuration file (for Version 8) is typically named **sasv8.cfg**, and the autoexec file is named **autoexec.sas**. These files typically reside in your home directory.

Specifying System Options

SAS system options can be specified in one or more ways:

- in a configuration file
- in the SASV8_OPTIONS environment variable
- in the SAS command
- in an OPTIONS statement (either in a SAS program or an autoexec file)
- in the System Options window.

Table 17.1 on page 308 shows where each SAS system option can be specified.

Any options that do not affect the initialization of the SAS System, 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 SASV8_OPTIONS variable, or in the SAS command. These options determine how the SAS System 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 option 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.

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:

- Modify your current configuration file (see “Processing Configuration Files” on page 17) or create a new configuration file. Specify SAS system options in the file by preceding each with a hyphen. For ON/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. For example, a configuration file might contain these option specifications:

  ```
  -nocenter
  -verbose
  -linesize 64
  ```

  All SAS system options can appear in a configuration file.
- Specify SAS system options in the SASV8_OPTIONS environment variable before you invoke the SAS System. See “Defining Environment Variables” on page 17. For example, in the Korn shell, you would use:

  ```
  export SASV8_OPTIONS='-xwait -nodate'
  ```

  Settings that you specify in the SASV8_OPTIONS environment variable affect SAS sessions that are started when the variable is defined.
- Specify SAS system options in the SAS command. Precede each option with a hyphen:

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

  For ON/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.
- 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';
```

Refer to *SAS Language Reference: Dictionary* for more information on the OPTIONS statement. Not all options can be specified in the OPTIONS statement. To find out about a specific option, look up its name in Table 17.1 on page 308.

- Specify SAS system options in an OPTIONS statement in a autoexec file. For example, your autoexec file could contain the following statements:

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

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

In general, use quotes to enclose filenames and pathnames specified in the OPTIONS statement or the System Options window. Do not use quotes otherwise. Any exceptions are discussed under the individual option. You can use the abbreviations listed in Table 4.2 on page 88 to shorten the filenames and pathnames you specify.

Processing System Options Set in One Place

If the same option is set more than once within the SAS command, only the last setting is used; the others are ignored. For example, the DMS option is ignored in this case:

```
sas -dms -nodms
```

However, if the same option is set more than once within a configuration file, or within the SASV8_OPTIONS environment variable, only the first setting is used; the others are ignored. For example, the NODMS option is ignored in the following case:

```
-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 to add the new pathname. See "APPEND" on page 254 and "INSERT" on page 272 for more information.

Processing System Options Set in Multiple Places

When the same option is set in more than one place, the most recent specification is used. In the following list, settings in places listed first override settings in places listed farther down:

1. System Options window or OPTIONS statement (from a SAS session or job).
2. autoexec file that contains an OPTIONS statement (after the SAS System initializes)
3. SAS command
4. SASV8_OPTIONS environment variable
5. configuration files (before the SAS System initializes)

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

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. See “APPEND” on page 254 and “INSERT” on page 272 for more information.

Processing Configuration Files

The SAS System is shipped with a default configuration file in the **`sasroot`** directory. Your SAS Installation Representative 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 the option settings from each of these files in this order:*

1. **`sasv8.cfg`** in your current directory.
2. **`sasv8.cfg`** in your home directory.
3. **`.sasv8.cfg`** in your home directory. (Notice the leading period.)
4. **`sasv8.cfg`** in the **`sasroot`** directory. (See Appendix 1, “The sasroot Directory,” on page 317.)

You can bypass this search by:

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

```
sas -config filename
```

- □ specifying a configuration file in the SASV8_OPTIONS environment variable. See “Defining Environment Variables” on page 17. For example, in the Korn shell, you would use:

```
export SASV8_OPTIONS='-config filename'
```

- □ defining the environment variable SASV8_CONFIG. See “Defining Environment Variables” on page 17. For example, in the Korn shell, you would use:

```
export SASV8_CONFIG=filename
```

filename is the name of a file containing SAS system options.

If you have specified a configuration file in the SASV8_OPTIONS or SASV8_CONFIG environment variables, you can prevent the SAS System from using that file by specifying NOCONFIG in the SAS command.

To create a configuration file, you can copy **`sasv8.cfg`** from the **`sasroot`** directory and modify it to set the options you want.

Defining Environment Variables

Environment variables are variables that apply to both the current shell and to any subshells 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 SASV8_OPTIONS environment variable to specifiy system options and the SASV8_CONFIG environment variable to specify a

* For future versions of SAS, the extensions of these files will change accordingly.

configuration file. You can also use environment variables as filerefs and librefs in various statements and commands.

The way in which you define an environment variable depends on the shell that you are running. 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 into one command:

```
$ export scname=phonelist
```

If you change the value of **scname**, 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.

In the C shell, 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
```

Use the **echo** command and parameter substitution to display the values of individual environment variables, just as you do for shell variables, for example: **echo $SHELL**Use the **env** (or **printenv**) command to display all environment variables and their current values.

Determining the Completion Status of a SAS Job

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.

Table 1.1 on page 18 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 warning(s)	1
SAS System issued error(s)	2
User issued ABORT statement	3
User issued ABORT RETURN statement	4
User issued ABORT ABEND statement	5
User issued ABORT RETURN *n* statement	*n*
User issued ABORT ABEND *n* statement	*n*

If the user specifies 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 0-127. Numbers greater than 127 may not print what the user expects because the code is interpreted as a signed byte.