Paper 414-2013

Stealing the Admin's Thunder: SAS® Macros to Interact with the UNIX OS from within SAS® Enterprise Guide®

Thomas Kunselman, Southern California Edison

ABSTRACT

For SAS® users who are unfamiliar with the UNIX environment, simple tasks like copying, renaming, or changing the permission settings on a file can be very non-intuitive. Many of these tasks are not even possible through the SAS® Enterprise Guide® Server List Window. This paper will present several SAS macros that can be used to: view and kill UNIX host processes; display, compare, and manage folders and files, including copying subfolders and changing permissions and owners; display and set default file system permissions for new objects. Please note that for these macros to work, the X command must be allowed on the SAS server.

INTRODUCTION

When you're running SAS in a server environment, there are a few basic tasks that seem to be universal. It's useful to be able to share files with other users, yet protect those you don't want to share. The ability to copy the contents of directories and their subdirectories is a great timesaving feature, especially if functionality to migrate code from one folder to another for promotion purposes is added. Did that last program or task you submitted in SAS Enterprise Guide get stuck? Then how about a way to kill that process on the server?

One way to do this is by logging into the shell of the SAS host environment and complete these tasks from the command line. But many sites only allow access through SAS Enterprise Guide. Another option would be to submit the UNIX commands via a program in SAS Enterprise Guide using the X statement. With either of these options, learning commands like ls, umask, ps, chmod and kill can be a challenge for folks who have primarily used SAS on Windows or the IBM Mainframe.

This paper provides examples on how to accomplish the following tasks by invoking several user-written SAS macros via SAS Enterprise Guide Prompts.

- *Finding and killing your SAS processes on the host (ps; grep; kill)*
- *View and set UNIX permissions on directories and files (ls; chmod)*
- *Set default permissions for sharing (umask)*
- *Compare, copy, and migrate recursive contents of folders and sub-folders (cp)*

For several of the macros associated with the above tasks, examples of how they work, their requirements, and sample code fragments are included in this paper. Explanation of the complete macro code and SAS Enterprise Guide projects are outside of the scope of this paper due to space and time limitations.

GENERAL REQUIREMENTS

The examples here were created using SAS Enterprise Guide 5.1 connecting to SAS 9.3 Enterprise BI Server on AIX. Some project features may not be available in earlier versions of SAS Enterprise Guide. The operating system commands may behave differently on other flavours of UNIX/LINUX.

Most of these macros require that the SAS workspace server have the –allowxcmd option specified.

Some of these macros use a default SAS LIBRARY that can often be specified in the program task where the macro is invoked. In some instances, the default library is called SYSLIB and is defined in SAS Metadata to point to the SASUSER library. This is explained further in the FINDING AND KILLING PROCESSES section.

FINDING AND KILLING PROCESSES

The ability to kill SAS processes comes in very handy when SAS is running on a remote UNIX host. SAS processes can remain running on the server indefinitely under some of the following scenarios. A submitted task may not return control and you will not be able to stop it through SAS Enterprise Guide. Or the connection to the server is lost, or SAS Enterprise Guide is terminated abnormally. In cases where SAS Enterprise Guide has become unresponsive because a submitted task has not completed, you can start a second SAS Enterprise Guide instance on your local PC to find and kill the processes on the UNIX host. The Find_Process and Kill_Process macros assist with this.

THE FIND_PROCESS MACRO

When the KillProcess project is first opened, the first set of programs in the process flow must be executed. This program branch (Figure 1.) will define the Find_Process and Kill_Process macros, as well as invoke the Find_Process macro. The Find_Process macro returns a list of SAS processes (Output 1.) for the userid the SAS workspace server is spawned under, sorted by elapsed time since the process started.

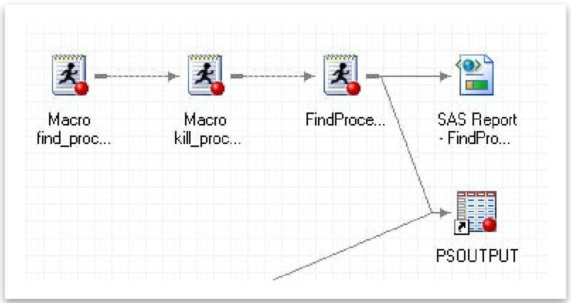
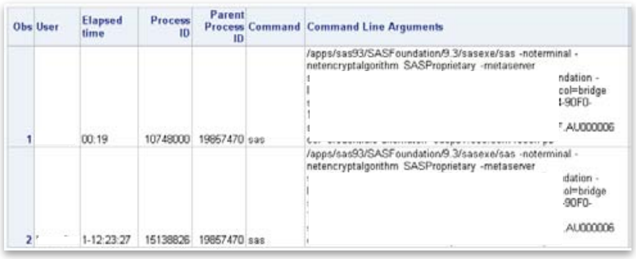


Figure 1. Branch from Macro Find_Process



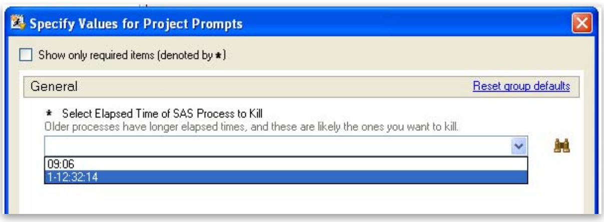
Obs	User	Elapsed time	Process ID	Parent Process ID	Command	Command Line Arguments
1	[redacted]	00:19	10748000	19857470	sas	/apps/sas93/SASFoundation/9.3/sasexe/sas -noterminal -netencryptalgorithm SASProprietary -metaserver [redacted] ndation - col=bridge 4-90F0- T.AU000006
2	[redacted]	1-12:23:27	15138826	19857470	sas	/apps/sas93/SASFoundation/9.3/sasexe/sas -noterminal -netencryptalgorithm SASProprietary -metaserver [redacted] idation - ol=bridge -90F0- .AU000006

Output 1. List of SAS processes

In Output 1, the Find_Process macro returns two SAS processes for the user. The first has been running for 19 seconds and is the SAS workspace server that was just invoked when we submitted the branch. The second SAS process has been running for a day and a half and needs to be killed.

THE KILL_PROCESS MACRO

After discovering a process that needs to be killed, running the Kill Process program (not shown) in the project will invoke the Kill_Process macro. The Kill Process program has a dynamic prompt attached that displays the elapsed times for each process (Display 1). This is used to select which process to kill. Once the prompt value has been selected and Run pressed, the process will be killed and the Find_Process macro will be executed again to show the most recent SAS process status on the server (Output 2).



Specify Values for Project Prompts

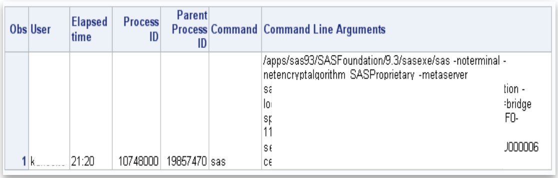
Show only required items (denoted by ∗)

General — Reset group defaults

∗ Select Elapsed Time of SAS Process to Kill
Older processes have longer elapsed times, and these are likely the ones you want to kill.

09:06
1-12:32:14

Display 1. Dynamic prompt displaying elapsed time



Obs	User	Elapsed time	Process ID	Parent Process ID	Command	Command Line Arguments
1	k[redacted]	21:20	10748000	19857470	sas	/apps/sas93/SASFoundation/9.3/sasexe/sas -noterminal -netencryptalgorithm SASProprietary -metaserver sa lo sp 11 se ce [redacted] tion - :bridge F0- J000006

Output 2. Updated list of SAS processes

HOW THE FIND_PROCESS MACRO WORKS

The Find_Process macro uses a filename statement to pipe the results of the ps and grep commands into a SAS dataset. PROC PRINT is used to display the contents of the data set for any Command column values that contain “sas”. The following code fragment provides the fileref used in the macro.

```
* Create a fileref to execute the ps command and pipe the output to SAS ;
filename psoutput PIPE  "ps -ef -o user,etime,pid,ppid,comm,args | grep &_USERID"
             LRECL=2000;
```

HOW THE KILL_PROCESS PROGRAM WORKS

Within the SAS Enterprise Guide project the Kill Process program is manually run separately, after the Find_Process branch has completed. The reason for this is that the Kill Process program has a dynamic prompt (Display 1) associated with it, and that prompt reads from the data set created by the Find_Process macro. If both the Find Process program branch and the Kill Process program were run at the same time, SAS Enterprise Guide will prompt before any program has executed, even if the programs are linked to run sequentially. In effect, the dynamic prompt is displayed, and the data set it is populated by, has not been created or is out of date.

After selecting the process to kill based on the elapsed time displayed in the prompt (Display 1), the Kill Process program determines the process ID from the PSOUTPUT results data set and passes it to the Kill_Process macro which issues the kill UNIX command to kill the process. The following code block represents the Kill Process program.

```
* The elapsed time of the SAS process to kill is selected using a dynamic prompt.,;
* and stored in the following macro variable;
%put &Prompt_PROCESSTIME;

* Find the process id based on elapsed time and put it in macro variable PROCESSID;
PROC SQL NOPRINT;
SELECT pid
INTO : PROCESSID
FROM SYSUSER.PSOUTPUT
WHERE etime eq "&Prompt_PROCESSTIME";
QUIT;

* Kill the process specified by the user;
%kill_process(&PROCESSID);

 * Rerun the find_process macro to update the process list in SYSUSER for the
dynamic prompt and ensure that the selected process has been killed;
* and display the current SAS processes;
%find_process(&SYSUSERID,SYSUSER);
```

The following code block is the Kill_Process macro definition. This gets defined in one of the steps when the Find Process program branch is run.

```
%macro kill_process(process_id);

%* Run the kill command on the user specified process id;
%sysexec kill -9 &PROCESS_ID ;

%mend kill_process;
```

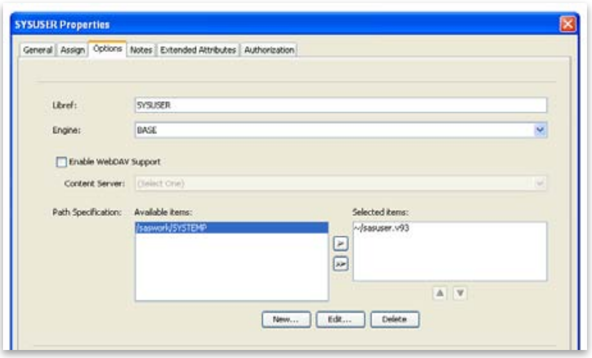
THE KILL PROCESS DYNAMIC PROMPT

Dynamic prompts within SAS Enterprise Guide use a data source that has been defined in SAS Metadata. Before using this project, the SAS administrator must create a library named SYSUSER and register the PSOUTPUT table. To prevent multiple users from overwriting the PSOUTPUT data set, SYSUSER points to the default location of the SASUSER library.

Defining the SYSUSER Library in Metadata

On this SAS host, the default location for the SASUSER library is ~/sasuser.v93. A new library is defined in metadata for the SYSUSER library. Display 2 shows the options tab of the library properties window for SYSUSER in SAS Management Console®.

The SYSUSER library is set to be Pre-Assigned in the Advanced options tab so that it is available after SAS Enterprise Guide connects to the SAS workspace server.



Libref: SYSUSER
Engine: BASE
☐ Enable WebDAV Support
Content Server: (Select One)
Path Specification: Available items: /saswork/SYSTEMP
Selected items: ~/sasuser.v93

Display 2. Library Metadata Definition

Registering the PSOUTPUT Table

SAS Management Console can be used to register the PSOUTPUT table in metadata. Prior to registering the table, the SYSUSER library must be defined and the PSOUTPUT data set has be physically created by running the Find_Process macro at least one time.

Once the PSOUTPUT table is registered, it can be used as an input data source to populate the dynamic prompt used to select a process to be killed. Because of the way that the SYSUSER library was defined, the prompt will automatically be pointed to the individualized location of the SASUSER library which is based on the credentials used to start the SAS workspace server.

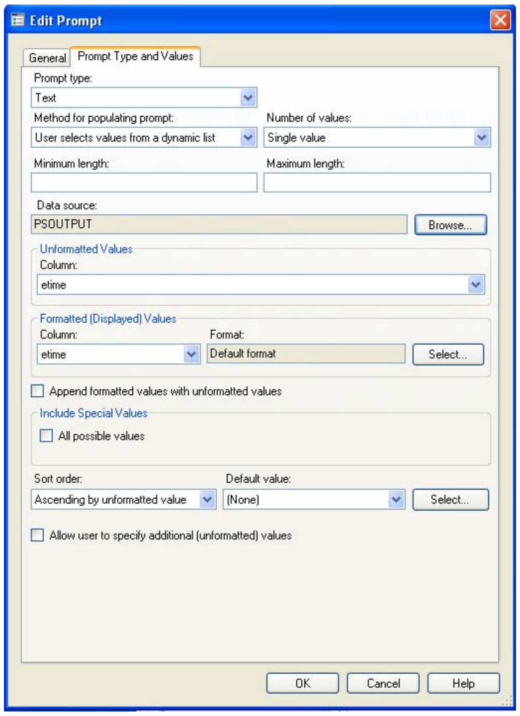
Creating the Dynamic Prompt to Select the Process

Using the Prompt Manager in SAS Enterprise Guide, a prompt called Prompt_PROCESSTIME is created.

On the “Prompt Type and Value” tab, select the “Method for populating Prompt” to be “User selects values from a dynamic list.” Using the Browse button, select the PSOUTPUT table from the SAS server folder structure in metadata.

Once the PSOUTPUT table has been selected, the column to populate the prompt can be selected. This will be the etime column, which is the elapsed time for each of the SAS processes running on the server. Change the “Sort Order” to be ascending, so the most recent processes are shown first, and the oldest are shown last in the drop down list.

Once the prompt has been created, it can be added to the program object in the process flow. The prompt will be displayed before the program is run on the SAS server.



Prompt type: Text
Method for populating prompt: User selects values from a dynamic list
Number of values: Single value
Minimum length:
Maximum length:
Data source: PSOUTPUT
Unformatted Values
Column: etime
Formatted (Displayed) Values
Column: etime
Format: Default format
☐ Append formatted values with unformatted values
Include Special Values
☐ All possible values
Sort order: Ascending by unformatted value
Default value: (None)
☐ Allow user to specify additional (unformatted) values

Display 3. Dynamic Prompt Setup

VIEWING AND CHANGING PERMISSIONS

One of the major advantages of using SAS in a server environment is the ability to easily share access to your SAS data sets and projects. Sharing files requires a basic understanding of UNIX filesystem permissions, and the ability to view and set permissions on directories and files. A detailed security discussion is beyond the scope of this paper. Additional information can be found in Wikipedia: The Free Encylopedia, File System Permissions.

This simple security scenario assumes the UNIX system administrator has user ids assigned to groups on the host. Directories and files can also belong to a group and have permissions set at the user, group, and other (everyone else) levels. If a file has read and write permissions for group A, but no permissions assigned for Other, then user id’s assigned to group A can read and write the file and everyone else would be denied access.

To steal the thunder and start sharing files, the ability to perform the following basic tasks is needed:

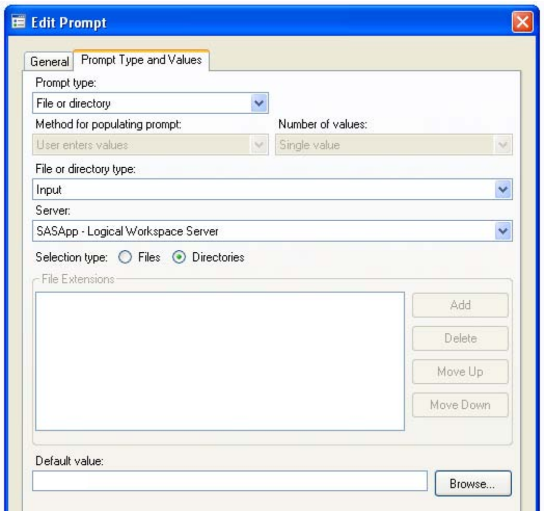
- Viewing security permissions on a folder or file.
- Setting security permissions on a folder or file.
- Assigning group ownership of a folder or file.

VIEWING PERMISSION SETTINGS

There are two ways to view permission settings using the Find_Files macro and the ViewSingle_Perms macro. The first allows a directory on the UNIX server to be selected and then will display the permission settings for the recursive contents. The other way is to select either a single file or directory. The example in this section demonstrates viewing permission settings on a single item. The recursive example is very similar to comparing and migrating directory recursive contents and can be found later in this paper.

The File or Directory Prompt Setup

There are two programs in the AIXPermissions.egp project that use the ViewSingle_Perms macro. The first is used to view permission settings on Directories and the other on Files. There are two because the “File or Directory” prompt used to provide input to the macro can only select files or directories per each prompt definition, not both, as shown by “Selection Type” in Display 4.



Prompt type: File or directory
Method for populating prompt: User enters values
Number of values: Single value
File or directory type: Input
Server: SASApp - Logical Workspace Server
Selection type: ○ Files [✓ Directories]
Default value:

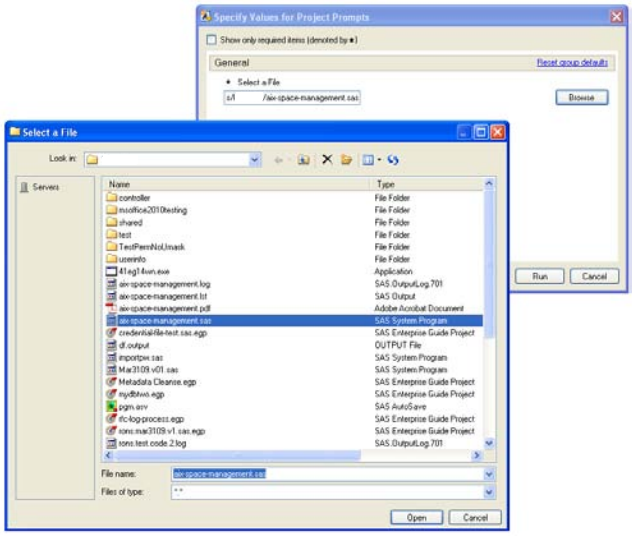
Display 4. File or Folder Prompt Setup

Using the File Prompt

The file prompt connects to the SAS server and allows you to select an individual file in a manner similar to looking at file system objects in the Server List window. Once you've selected a file, the Prompt_FILE macro variable is set with the path and name of the selected file. An example of how the file prompt appears in this program is shown in Display 5.

Viewing the Output

The output contains the owner and group of the file or directory selected. There are four columns displaying permission settings in the sample output displayed in Output 3. The Permissions column contains the traditional UNIX permission representation as output from the ls command. Search The Internets with keywords "UNIX permissions" to learn how to interpret this column. A more readable version is displayed in the PermUser, PermGroup, and PermOther columns.



Specify Values for Project Prompts

Show only required items (denoted by •)

General — Reset group defaults

• Select a File: s/I /aix-space-management.sas

Select a File

Look in:

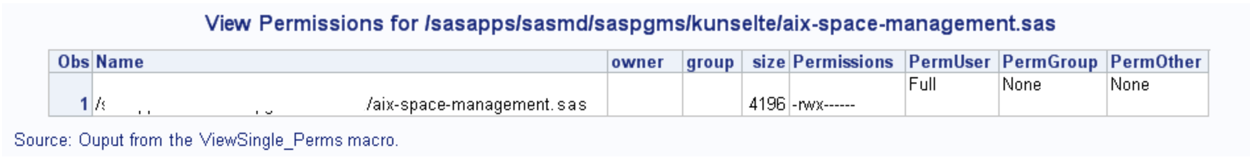
Name	Type
controller	File Folder
msoffice2010testing	File Folder
shared	File Folder
test	File Folder
TestPermNoUmask	File Folder
userinfo	File Folder
41eg14win.exe	Application
aix-space-management.log	SAS.OutputLog.701
aix-space-management.lst	SAS Output
aix-space-management.pdf	Adobe Acrobat Document
aix-space-management.sas	SAS System Program
credential-file-test.sas.egp	SAS Enterprise Guide Project
df.output	OUTPUT File
importpw.sas	SAS System Program
Mar3109.v01.sas	SAS System Program
Metadata Cleanse.egp	SAS Enterprise Guide Project
mydbtwo.egp	SAS Enterprise Guide Project
pgm.asv	SAS AutoSave
rfc-log-process.egp	SAS Enterprise Guide Project
rons.mar3109.v1.sas.egp	SAS Enterprise Guide Project
rons.test.code.2.log	SAS.OutputLog.701

File name: aix-space-management.sas

Files of type: *.*

Display 5. Prompt to Select a File

PermUser indicates that the owner of the file has full permissions to read, write, and execute. The PermGroup and PermOther columns indicates the access rights by members of the sas unix group and everyone else, respectively. In both these cases they have no access (None). Please note that read access is all that is required to run a SAS program. The best practice for SAS programs on UNIX is not to have the execute permission bit set.



View Permissions for /sasapps/sasmd/saspgms/kunselte/aix-space-management.sas

Obs	Name	owner	group	size	Permissions	PermUser	PermGroup	PermOther
1	/s /aix-space-management.sas			4196	-rwx------	Full	None	None

Source: Ouput from the ViewSingle_Perms macro.

Output 3. View Permission Settings Output

HOW VIEW PERMISSION SETTINGS MACROS WORK

There are two macros for retrieving and viewing permissions. The Find_Files macro was initially created to build a list of recursive directory contents for comparing and copying between two locations. Including the permission settings for each of the directory and file system objects was a natural extension for this macro and allows the retrieval of permission settings on all recursive contents of a specified directory. A second macro, ViewSingle_Perms, retrieves the permission settings on a single directory or file. There are two different ways in which permissions are retrieved.

Obtaining Permission Settings on Files with FOPTNAME Function

The following DATA step code can be used to retrieve information about a file on the host filesystem using a set of SAS file functions. The code block below is a simplified version of what's used in the Find_Files macro. One advantage of using these functions over reading output from UNIX commands is that you can view file permissions without having access to the X command on the SAS server.

This code only works for files. While there are equivalent functions for directories, they aren't useful for determining directory attributes under UNIX. The DOPTNAME function will only return one option, Directory, in a UNIX environment. Use the UNIX ls command output to retrieve information about directories.

```
DATA work.getfileinfo;
/* Assign a fileref to the file */
rc=FILENAME("myfile","/mydirectory/myfile.sas");

/* Open up the file using the fileref */
fid = FOPEN("myfile");
```

```
/* Determine the number of options available for this operating system */
numopts = FOPTNUM(fid);

/* Retrieve File Option Info looping through each option number */
DO _A = 1 TO numopts;
    optname = FOPTNAME(fid,_A);
    optval  = FINFO(fid, optname);

    /* Save the option values */
    SELECT(optname);
           WHEN('Filename') FileNamePath = optval;
           WHEN('RECFM') RECFM = optval;
           WHEN('LRECL') LRECL = optval;
           WHEN('File Size (bytes)') FileSize = optval;
           WHEN('Last Modified') FileLastModified = optval;
           WHEN('Create Time') FileCreateTime = optval;
           WHEN'Owner Name') OwnerName = optval;
           WHEN( 'Group Name') GroupName = optval;
           WHEN('Access Permission') Permissions = optval;
           /* Option names will vary based on the operating system */
           OTHERWISE PUT optname " needs to be defined in the DATA step.";
    END;
END;
RUN;
```

Obtaining Permission Settings on Directories with the UNIX ls Command

The following example is the bare bones code used in both the Find_Files and ViewSingle_Perms macros for determining directory permission settings. Because the fileref uses PIPE, the SAS server requires the X command to be enabled. This code will run the ls UNIX command and pipe the output into the DATA step where it's parsed like any space delimited line of text input.

```
/* Create a fileref to access the results of the UNIX ls command */
FILENAME getperms PIPE  "ls -ld /mydirectory" LRECL=2000;

/* Read in the filesystem permissions piped out */
DATA work._viewsingle_perms;
    LENGTH Permissions $10.
           PermUser $3.
           PermGroup $3.
           PermOther $3.
           PermOctalUser $3.
           PermOctalGroup $3.
           PermOctalOther $3.
           dummy $1. owner $8. group $8. size 8. month $3. day $2. year $4. Name
$255.;
    INFILE getperms TRUNCOVER;
    INPUT Permissions $ dummy $ owner $ group $ size month $ day $ year $ Name $;
    PermUser = SUBSTR(Permissions,2,3);
    PermGroup = SUBSTR(Permissions,5,3);
    PermOther = SUBSTR(Permissions,8,3);
FORMAT PermUser PermGroup PermOther $UnixPermText.;
run;
```

CHANGING PERMISSION SETTINGS

A very simple macro running the UNIX chmod command is used in the following examples to change directory or file permission settings. Programming complexity can be minimized by creating a set of prompts attached to a program in SAS Enterprise Guide to pass arguments to the ChangeFilePermissions macro. In the simple security model outlined in the previous section, the UNIX command chmod is used to add or remove read, write, and execute priveleges for the owner, group members, or everyone else. This allows SAS developers to change directory and folder permissions without the need to login directly to the SAS host or ask an administrator to make these changes for them.

Changing Permissions

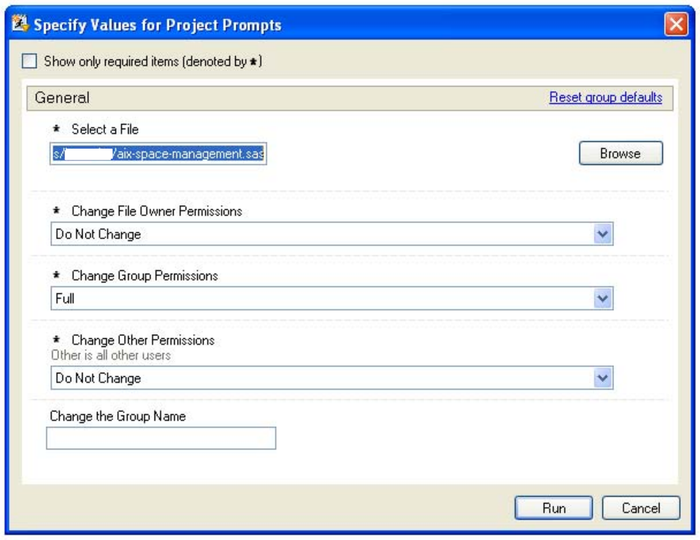
The ChangeFilePermissions program in the the AIXPermissions.egp project has prompts to select the file, set the owner, group, and other permissions, and change the name of the group the file belongs to. These prompts can be seen in Display 6.

The Select a File and Directory prompts are reused throughout the project and have already been described above. Although not shown here, the ChangeDirectoryPermissions version of the program also includes a prompt that allows permission changes to be applied recursively on the contents of the directory.

Each of the Change Permissions drop down boxes contains the combination of permissions below in Display 7. Selecting the default "Do Not Change" value will leave that set of permissions as is.

The UNIX group that the folder belongs to can be changed by typing the new group name in the "Change the Group Name" text box prompt.

Once the prompts have been completed and the Run button clicked, the macros will be executed and the View Settings Output will be displayed as above in Output 3 on Page 5.


Specify Values for Project Prompts
Show only required items (denoted by *)
General
Reset group defaults
Browse
Run
Cancel

* Select a File: s/[redacted]/aix-space-management.sas

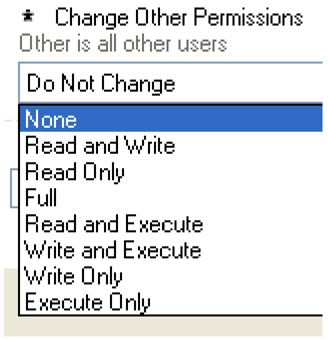
* Change File Owner Permissions: Do Not Change

* Change Group Permissions: Full

* Change Other Permissions
Other is all other users: Do Not Change

Change the Group Name:

Display 6. Change Permission Settings Prompt



* Change Other Permissions
Other is all other users
Do Not Change
None
Read and Write
Read Only
Full
Read and Execute
Write and Execute
Write Only
Execute Only

Display 7. Permission Combinations

HOW THE CHANGE FILE PERMISSIONS PROGRAM WORKS

The key to achieving simplicity in the macro is by defining the unformatted values in the Change Permissions prompts to be the actual arguments required by the UNIX chmod command. The formatted values to the left are replaced by the unformatted values in the prompt which are the chmod command's symbolic notation for changing permission settings.

The Change Permissions Prompt Setup

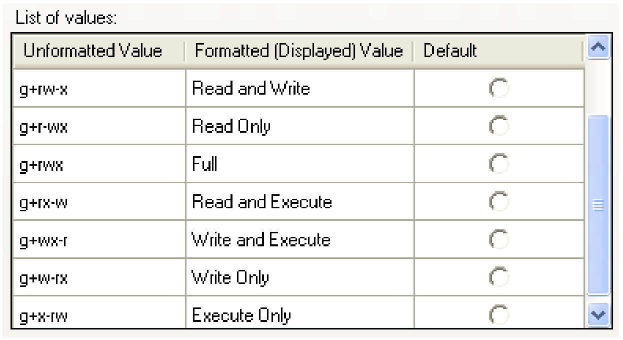
Explaining symbolic notation for file system permissions is beyond the scope of this paper. An explanation of symbolic modes and use with the chmod command can be found in Wikipedia: the Free Encyclopedia, chmod.

The unformatted values from the Change Group Permissions prompt are used exactly as is on the UNIX chmod command line. The g indicates that the permission to be changed is for the group level, the + and – indicates whether the permission is to be added or taken away, and the r,w, and x indicate read, write, and execute permission bits. The unformatted values from the prompt configuration window can be seen in Display 8.

The ChangeFilePermissions Macro Code

The macro takes two parameters, the CFP_FILE, which is the full path and filename, and the CFP_CHANGE which is the chmod symbolic notation for how the permissions are to be changed. If either of these prompts are blank or contain the value "NoChange" then the X command will not execute.

```
%ChangeFilePermissions(&Prompt_File,
&Prompt_PermGroupSet);
```



List of values:

Unformatted Value	Formatted (Displayed) Value	Default
g+rw-x	Read and Write	○
g+r-wx	Read Only	○
g+rwx	Full	○
g+rx-w	Read and Execute	○
g+wx-r	Write and Execute	○
g+w-rx	Write Only	○
g+x-rw	Execute Only	○

Display 8. Unformatted Values in Group Prompt

The code block below is the macro definition. The %BQUOTE is used to keep the + and - characters from the permission set prompts from being interpreted by the macro processor.

```
%macro ChangeFilePermissions(CFP_FILE, CFP_CHANGE);
%if %bquote(&CFP_CHANGE) ne NoChange and %bquote(&CFP_CHANGE) ne and
%bquote(&CFP_FILE) ne %then x chmod &CFP_CHANGE &CFP_FILE;
%mend;
```

DEFAULT PERMISSIONS FOR GROUP SHARING

Every new file or directory that is created on the UNIX server has a default permission settings. What those permissions are, depends on how the system is configured. In a secured system this often means that any created files cannot be read from or written to by anyone in the same UNIX group or anyone else on the host server. In a more open host environment, the permission default may be configured so that any new file created can be read by anyone who has access to the host.

The UNIX umask command can be used to customize this for the current SAS session started on the server by SAS Enterprise Guide. The Set_Default_Permissions macro was written to be automatically run in SAS Enterprise Guide startup options without prompting. If required, prompts could easily be created and assigned to a program for running this macro. The following options are recognized by this macro for setting default group permissions.

- *group_read_write:* Every new file will have permissions that grant members of the group, read and write permissions.
- *group_read_only:* Every new file will havepermissions that grant members of the owner group, read only permission.
- group_noread_nowrite: Every new file will have permissions so that only the owner can read and write the file, excluding everyone else.

```
%macro     set_default_permissions(permtype=)
           / des='Set default unix permissions for all newly created data and files'
           ;
    %if &PERMTYPE eq %str(group_read_write) %then %do;
           %sysexec umask 007;
           %put "NOTE: group_read_write: Every new file will have permissions that
grant members of the owner group, read and write permissions.";
    %end;
    %else %if &PERMTYPE eq %str(group_read_only) %then %do;
           %sysexec umask 027;
           %put "NOTE: group_read_only: Every new file will havepermissions that
grant members of the owner group, read only permission.";
    %end;
    %else %if &PERMTYPE = %str(group_noread_nowrite) %then %do;
           %sysexec umask 077;
    %put "NOTE: group_noread_nowrite: Every new file will have permissions so that
only the owner can read and write the file, excluding everyone else.";
    %end;
    %else %do;
           %put "ERROR: Invalid argument(&PERMTYPE).  The set_default_permissions
function requires one of the following arguments:";
           %put "NOTE: group_read_write: Every new file will have permissions that
grant members of the owner group, read and write permissions.";
           %put "NOTE: group_read_only: Every new file will havepermissions that
grant members of the owner group, read only permission.";
           %put "NOTE: group_noread_nowrite: Every new file will have permissions so
that only the owner can read and write the file, excluding everyone else.";
    %end;
%mend set_default_permissions;
```

This macro can be very useful for changing default group permissions inside SAS programs as new data sets and output are being created. These default permissions will also affect any new programs or projects saved to the SAS server with SAS Enterprise Guide for the duration of the SAS server connection.

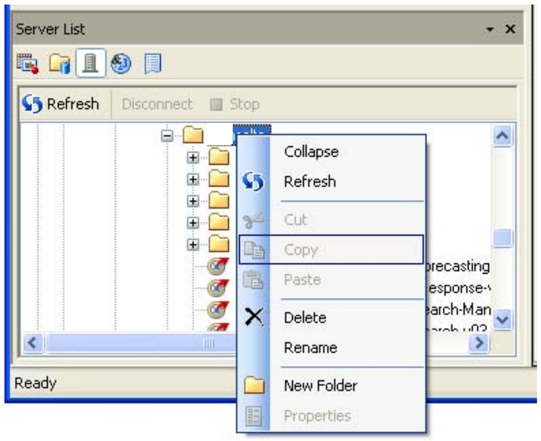
COMPARE, COPY, AND MIGRATE DIRECTORY CONTENTS

The need for macro utilities to provide support for recursive contents of directories quickly becomes apparent when working in a SAS server environment that only provides access to the file system through SAS Enterprise Guide. Right-clicking on a directory in the Server List window with the intent of copying it's entire contents to another location yields the menu shown in Display 9.

In addition to copying directory contents, migrating code between development, test, and production directories requires a means to compare and synchronize these environments.

All of the compare, copy, and migrate activities are built around two macros. The first is the FindFiles macro which has been previously discussed as part of viewing permission settings. This macro builds a recursive list of a directory's contents and was made possible by the WUSS 2012 presentation and paper, "Obtaining A List of Files in A Directory Using SAS Functions" by Jack Hamilton. For in-depth knowledge on how to accomplish this, please find a link to his paper in the References section on Page 12.

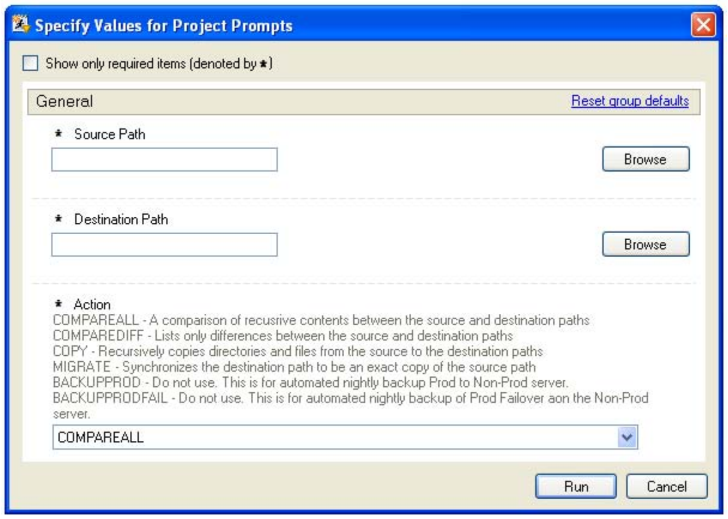
The second macro is called CompFiles. This is the macro that compares, copies, and migrates. The macro takes as input two directory paths for a source and destination, invokes the FindFiles macro to build the recursive contents list for each path, and then can compare, copy, or migrate between the source and destination paths.



Display 9. Copying a directory is not an option

THE COMPFILES PROMPTS

The Source Path and Destination Path prompt type for selecting directories has been discussed in prior sections. The Action prompt creates the Prompt_ACTION macro variable that passes the action text value to the CompFiles macro. The backup actions shown in Display 10 are used to copy the contents of fixed directories from one location to another. They are hardcoded in the macro and could easily be modified or removed. The Actions overview:


Specify Values for Project Prompts

Show only required items (denoted by *)

General — Reset group defaults

* Source Path

* Destination Path

* Action

COMPAREALL - A comparison of recusrive contents between the source and destination paths
COMPAREDIFF - Lists only differences between the source and destination paths
COPY - Recursively copies directories and files from the source to the destination paths
MIGRATE - Synchronizes the destination path to be an exact copy of the source path
BACKUPPROD - Do not use. This is for automated nightly backup Prod to Non-Prod server.
BACKUPPRODFAIL - Do not use. This is for automated nightly backup of Prod Failover aon the Non-Prod server.

COMPAREALL

Display 10. The CompFiles Macro Prompts

- COMPAREALL - A comparison of recursive content between the source and destination paths. This includes file size and last modified datetime and location in source or destination.
- COMPAREDIFF - Lists only differences between the recursive contents of the source and destination paths.
- COPY - Recursively copies directories and files from the source to the destination path overwriting existing content with the same name in the destination path. Files in the destination that are not in the source are left in place.
- MIGRATE - Synchronizes the destination path to be an exact copy of the source path. Newer files in the destination path will be overwritten by older files from the source if they have the same name and path. Files in the destination path that are not in the source path will be deleted.

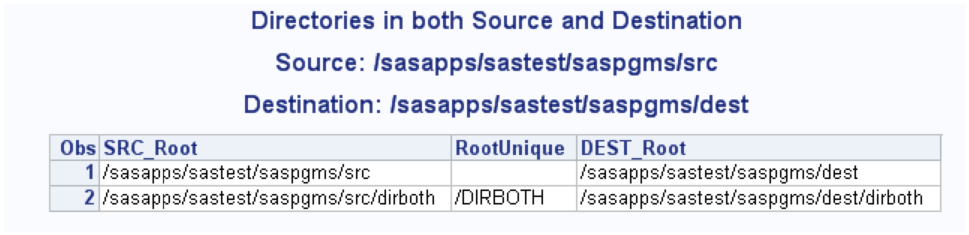
The COMPAREALL action and the MIGRATE action output is included below. The output from these actions are generated by merging the source and destination lists of files and creating output data sets based on which data sets the unique filenames are in. All of these actions require that the X command be enabled on the SAS server.

ACTION: COMPARE ALL

This action will compare the entire contents of two path locations. A report is generated on the following items:

- Directories in both Source and Destination.
- Directoriess in Source that are not in Destination.
- Directories in Destination that are not in the Source
- Files in both Source and Destination
- Files in Source that are not in Destination
- Files in Destination that are not in Source

Compare All Output

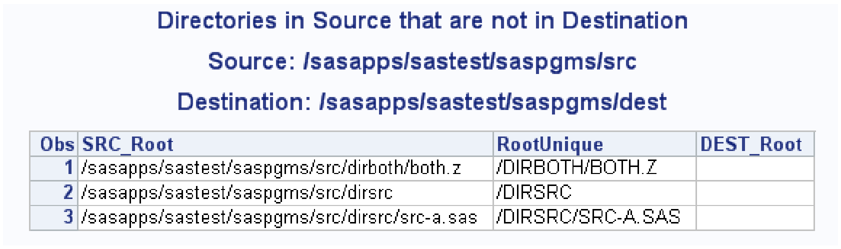


Directories in both Source and Destination

Source: /sasapps/sastest/saspgms/src

Destination: /sasapps/sastest/saspgms/dest

Obs	SRC_Root	RootUnique	DEST_Root
1	/sasapps/sastest/saspgms/src		/sasapps/sastest/saspgms/dest
2	/sasapps/sastest/saspgms/src/dirboth	/DIRBOTH	/sasapps/sastest/saspgms/dest/dirboth

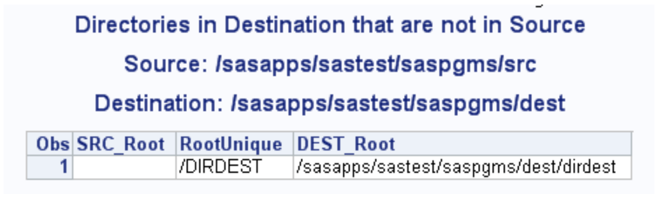


Directories in Source that are not in Destination

Source: /sasapps/sastest/saspgms/src

Destination: /sasapps/sastest/saspgms/dest

Obs	SRC_Root	RootUnique	DEST_Root
1	/sasapps/sastest/saspgms/src/dirboth/both.z	/DIRBOTH/BOTH.Z	
2	/sasapps/sastest/saspgms/src/dirsrc	/DIRSRC	
3	/sasapps/sastest/saspgms/src/dirsrc/src-a.sas	/DIRSRC/SRC-A.SAS	

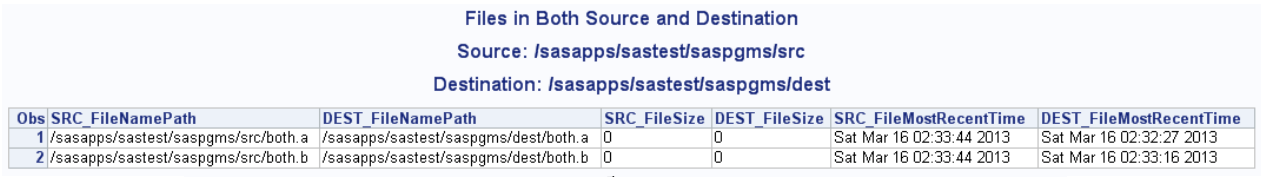


Directories in Destination that are not in Source

Source: /sasapps/sastest/saspgms/src

Destination: /sasapps/sastest/saspgms/dest

Obs	SRC_Root	RootUnique	DEST_Root
1		/DIRDEST	/sasapps/sastest/saspgms/dest/dirdest

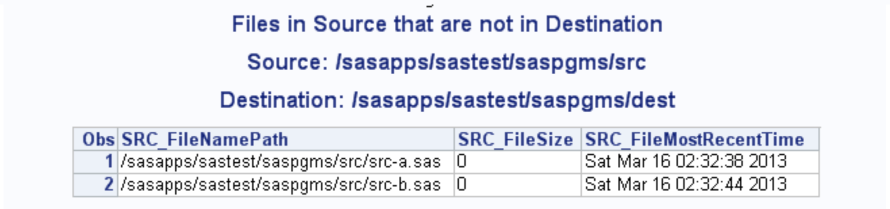


Files in Both Source and Destination

Source: /sasapps/sastest/saspgms/src

Destination: /sasapps/sastest/saspgms/dest

Obs	SRC_FileNamePath	DEST_FileNamePath	SRC_FileSize	DEST_FileSize	SRC_FileMostRecentTime	DEST_FileMostRecentTime
1	/sasapps/sastest/saspgms/src/both.a	/sasapps/sastest/saspgms/dest/both.a	0	0	Sat Mar 16 02:33:44 2013	Sat Mar 16 02:32:27 2013
2	/sasapps/sastest/saspgms/src/both.b	/sasapps/sastest/saspgms/dest/both.b	0	0	Sat Mar 16 02:33:44 2013	Sat Mar 16 02:33:16 2013

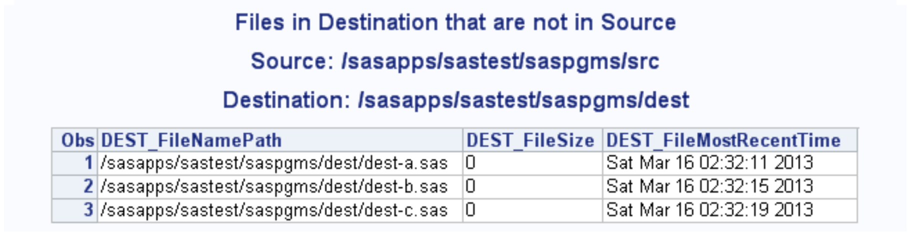


Files in Source that are not in Destination

Source: /sasapps/sastest/saspgms/src

Destination: /sasapps/sastest/saspgms/dest

Obs	SRC_FileNamePath	SRC_FileSize	SRC_FileMostRecentTime
1	/sasapps/sastest/saspgms/src/src-a.sas	0	Sat Mar 16 02:32:38 2013
2	/sasapps/sastest/saspgms/src/src-b.sas	0	Sat Mar 16 02:32:44 2013



Files in Destination that are not in Source

Source: /sasapps/sastest/saspgms/src

Destination: /sasapps/sastest/saspgms/dest

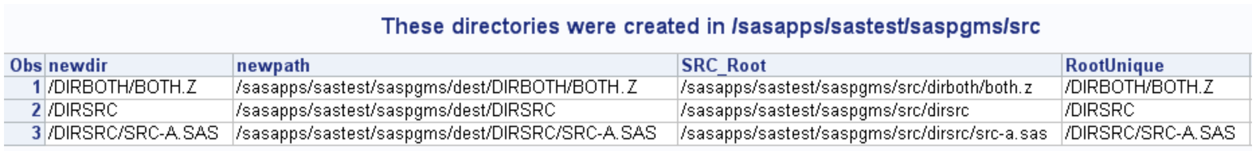
Obs	DEST_FileNamePath	DEST_FileSize	DEST_FileMostRecentTime
1	/sasapps/sastest/saspgms/dest/dest-a.sas	0	Sat Mar 16 02:32:11 2013
2	/sasapps/sastest/saspgms/dest/dest-b.sas	0	Sat Mar 16 02:32:15 2013
3	/sasapps/sastest/saspgms/dest/dest-c.sas	0	Sat Mar 16 02:32:19 2013

ACTION: MIGRATE

This action will compare the entire contents of two path locations and then synchronize the Destination directory to be identical to the source directory. Prior to running the actual MIGRATION action, the COMPAREDIFF action can be run to provide a report on exactly what items will be overwritten or deleted from the Destination directory.

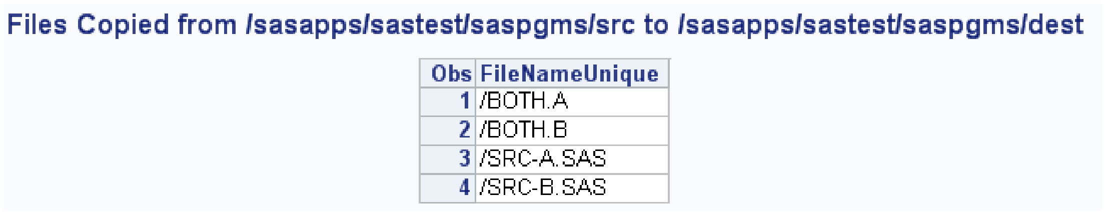
- Directories in Source that are not in Destination. These will be created in Destination.
- Directories in Destination that are not in Source. These will be deleted from Destination.
- Files in Source and Destination with Different Last Modified Dates. These will be overwritten in Destination. In the output below, BOTH.A and BOTH.B have a different update time, which means the files are overwritten in the Destination directory.
- Files in Source that are not in Destination. These will be copied to Destination.
- Files in Destination that are not in Source. These will be deleted from Destination.

Compare All Output



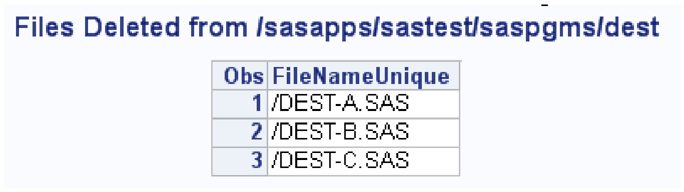
These directories were created in /sasapps/sastest/saspgms/src

Obs	newdir	newpath	SRC_Root	RootUnique
1	/DIRBOTH/BOTH.Z	/sasapps/sastest/saspgms/dest/DIRBOTH/BOTH.Z	/sasapps/sastest/saspgms/src/dirboth/both.z	/DIRBOTH/BOTH.Z
2	/DIRSRC	/sasapps/sastest/saspgms/dest/DIRSRC	/sasapps/sastest/saspgms/src/dirsrc	/DIRSRC
3	/DIRSRC/SRC-A.SAS	/sasapps/sastest/saspgms/dest/DIRSRC/SRC-A.SAS	/sasapps/sastest/saspgms/src/dirsrc/src-a.sas	/DIRSRC/SRC-A.SAS



Files Copied from /sasapps/sastest/saspgms/src to /sasapps/sastest/saspgms/dest

Obs	FileNameUnique
1	/BOTH.A
2	/BOTH.B
3	/SRC-A.SAS
4	/SRC-B.SAS



Files Deleted from /sasapps/sastest/saspgms/dest

Obs	FileNameUnique
1	/DEST-A.SAS
2	/DEST-B.SAS
3	/DEST-C.SAS

CONCLUSION

This paper covered a few of the reasons for interacting with the filesystem on the SAS server. Building the foundation for this interaction with SAS macros allows them to be used in different ways. The macros can be called directly from SAS programs to allow for automation, or they can be invoked with input from prompts in SAS Enterprise Guide. Prompting provides a friendlier interface that allows for greater control of the SAS server environment without the need for extensive operating system knowledge or knocking on the administrator's door for assistance.

REFERENCES

Aanderud, Tricia and Hall, Angela. 2012. Building Business Intelligence Using SAS: Content Development Examples. Pages 87-127. Cary, North Carolina: SAS Publishing

Hamilton, Jack. 2012. "Obtaining A List of Files in A Directory Using SAS Functions." SAS Conference Proceedings: Western Users of SAS Software 2012. Long Beach, California: WUSS. Available from http://www.wuss.org/proceedings12/55.pdf. Internet. Retrieved October, 2012.

Chmod. 25 February 2013, 7:47 UTC. In *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation Inc. Encyclopedia on-line. Available from http://en.wikipedia.org/wiki/Chmod. Internet. Retrieved 3 March 2013.

File System Permissions. 1 March 2013, 23:11 UTC. In *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation Inc. Encyclopedia on-line. Available from http://en.wikipedia.org/wiki/Filesystem_permissions. Internet. Retrieved 1 March 2013.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Thomas Kunselman
Thomas.Kunselman@gmail.com