Paper 3262 - 2015

Yes, SAS® can do! --- Manage External Files With SAS Programming

Justin Jia, TransUnion Canada, Burlington, Ontario, Canada

Amanda Lin, CIBC, Toronto, Ontario, Canada

ABSTRACT

Managing and organizing external files and directories play an important part in our data analysis and business analytics work. A good file management system can streamline project management and file organizations, and improve work efficiency significantly. Therefore it is needed under many circumstances to automate and standardize the file management processes through SAS® programming. Compared with managing SAS files via PROC DATASETS, managing external files is a much more challenging task which requires advanced programming skills. This paper presents and discusses various methods and approaches to managing external files with SAS programming. The illustrated methods and skills can have important applications in a wide variety of analytic work fields.

INTRODUCTION

In data analysis and business analytics using SAS programming, it is not unusual that we need to manage our projects and organize relating files and documents. A good file management system is important and supportive for us to streamline our documentation and project management and to improve work efficiency. For this purpose, we need create, rename, move, copy or delete files or directories on a local or remote computer with a specific operating system. The files that need management and organization could be: either SAS files such as SAS data sets, views, formats, catalogs and programs etc., or non-SAS external files or directories. To manage SAS files, it is quite easy because we can make use of the powerful PROC DATASETS procedure, which is specially designed for SAS file management uses. With this utility procedure, we can append SAS data sets, copy and move SAS files from one library to another library, rename and delete SAS files, modify the attributes of SAS data sets and variables in the data sets, create and delete indexes or integrity constraints on SAS data sets and so on. Because it is such a powerful and versatile tool, PROC DATASETS is therefore called the Swiss Army Knife in SAS file management. By using PROC DATASETS, SAS file management becomes an easy task to perform.

However, in our daily work, we often encounter circumstances that we need manage the non-SAS external files or directories. We need create, rename, copy, move and delete external files or directories for project management or file organization purposes. If the files under management are of a small number, we can manage and organize them easily through manual work. However, if they are of a big number, it is difficult and unrealistic to work on them manually. In other cases, for example, for ad-hoc and recurring analytic projects, the folder structure and file management are usually similar and repetitive in nature, therefore it would be efficient to automate and standardize the routine tasks via SAS programming. For these purposes, it is useful and beneficial for us to learn about how to manage external files via SAS programming.

Therefore, this paper will present and illustrate different methods and approaches of managing external files and directories with SAS programming. We will exemplify and discuss the advantages and disadvantages of these methods, and also introduce the use of new SAS file management functions and call routines. The creative ideas and programming skills can be widely used in many areas and fields.

I. SYSTEM STATEMENTS AND SHELL COMMANDS

The first means of managing external files through SAS is to exploit system statements and shell commands, including the X command/statement, %SYSEXEC macro statement and SYSTASK command. All these approaches share the same feature and functionality: they issue operating environment commands and invoke the operating system to execute the specified commands. In this paper, we will only focus on the Windows operating system, however, all the methods and approaches presented can be extended to other operating systems as long as the submitted commands follow the operating environment requirements.

X Command/Statement:

This command or statement allows SAS users to submit and execute operating system commands from a SAS session. In a Windows environment, the specified commands must be valid DOS or Windows commands. Its syntax is quite simple as follows[1]:

X <'command '> ;

no argument: it just opens a DOS command window.

command: specifies the to-be-executed operating environment command.

%SYSEXEC Macro Statement

The %SYSEXEC macro statement is similar to the X command/statement, which also issues operating environment commands. The specified commands will be immediately executed and the return code generated by the operating system will be assigned to the automatic macro variable SYSRC. Therefore we can use this macro variable to monitor if the command has been executed successfully or not. Return codes are integers. The default value of SYSRC is zero which indicates a successful execution, while any other value means unsuccessful.

The only difference between X command/statement and %SYSEXEC macro statement is that the host commands issued by %SYSEXEC should NOT be enclosed in quotation marks, while X command/statement and SYSTASK command do require quotation marks for the specified commands[2].

SYSTASK Statement

In addition to the X command/statement and %SYSEXC macro statement, we can also utilize SYSTASK statement to issue and execute operating environment commands from a SAS session or application. Unlike the X command and %SYSEXEC macro statement which are synchronous, SYSTASK statement is asynchronous. The syntax of SYSTASK statement is[3]:

SYSTASK COMMAND "OPERATING SYSTEM COMMAND" <options>;

It is worthy to note the very useful **STATUS= Stat-Var** option, which specifies a macro variable to contain the status of the task. Status variable names must be unique among all active tasks. The automatic SYSRC macro variable contains the return code for the SYSTASK statement, but the specified status macro variable retains the return code of executing the command in the operating system. To monitor whether a task has been executed successfully or not, we need look into both of them[3].

Below we give some examples of using them to manage external files and directories in a Windows environment. All the issued commands must be valid DOS or Windows commands. Please note: if the path name, directory name or file name contains blanks, we must enclose them in quotation marks, otherwise the submitted command cannot be executed correctly. However, as everyone knows, if we reference a macro variable enclosed in quotation marks, we must quote it with double quotation marks rather single quotations, otherwise the macro variable cannot be resolved. Although X command allows the repeated use of double quotation marks, however, SYSTASK statement requires the alternating use of single quotation marks and double quotation marks. In this case, we need to use %STR to mask the single quotation marks first, otherwise the referenced macro variable cannot be resolved. It is important to note that using the %UNQUOTE macro function is mandatory rather than optional. This is because, during macro execution, it unmasks and restores the normal tokenization of the single quotation marks after the PATH macro variable has been resolved. Without using it, the value of the single quotation marks is NOT unmasked before it reaches the SAS compiler, which may result in error messages[4].

```
********************** I. SAS Statements and Shell Commands. ********************;
options mprint symbolgen ;
%let Path=D:\WLMU2015\Projects;

************** A. Create New Directories. ***********;

X " mkdir &path.\WLMU999 ";
X " mkdir "&path.\WLMU 999"  "; *Note: quotation marks are required if the path
name or folder name contains blanks.;
X %unquote(%str(%') mkdir "&path.\WLMU 999" %str(%'));

%sysexec mkdir &path.\WLMU999 ;
%sysexec mkdir "&path.\WLMU 999" ;

systask command "mkdir  &path.\WLMU999" ;
systask command  %unquote(%str(%') mkdir  "&path.\WLMU 999" %str(%'));
```

```
************** B. Rename External Files or Directories. *********;
X "rename  "&path.\WLMU 999"   "New WLMU_999" ";
X %unquote(%str(%') rename "&path.\WLMU 999"   "New WLMU_999" %str(%'));

%sysExec rename "&Path.\WLMU 999"  "New WLMU_999" ;
%sysExec rename "&Path.\WLMU 999\Sales.pdf"  "2012 Sales.pdf" ;

systask command  %unquote(%str(%') rename "&path.\WLMU 999" "New WLMU_999"
%str(%'));
systask command  %unquote(%str(%') rename "&path.\WLMU 999\Sales.pdf"
"2012 Sales.pdf"  %str(%'));

************** C. Delete External Files or Directories. *********;
X "Del /q "&path.\Del\2012 Sales.pdf" "; *delete an individual file.;
X "Del /q "&path.\Del\*.pdf" ";  *delete all the PDF format files;
X "Del /q "&path.\Del\*.*" ";  *delete all the files;
X "rmdir /q "&Path.\Del" ";   *remove an empty directory named DEL.

%sysExec Del /q "&Path.\Del\2012 Sales.pdf";
%sysExec Del /q "&Path.\Del\*.pdf";
%sysExec Del /q "&Path.\Del\*.*";
%sysExec rmdir /q "&Path.\Del";

systask command  %unquote(%str(%') Del /q "&path.\Del\2012 Sales.pdf" %str(%'));
systask command  %unquote(%str(%') Del /q "&path.\Del\*.pdf" %str(%'));
systask command  %unquote(%str(%') Del /q "&path.\Del\*.*" %str(%'));
systask command  %unquote(%str(%') rmdir /q "&path.\Del"  %str(%'));

************** D. Copy and Move External Files or Directories. *********;
X "copy "&path.\Old\2012 Sales.pdf"  "&path.\New" "; * copy an individual file to a
new place without changing its name.;

X "copy "&path.\Old\2012 Sales.pdf"  "&path.\New\2012_Sales.pdf" "; * copy an
individual file to a new place and change its name.;

X "copy "&path.\Old\*.*"  "&path.\New" "; *copy all files in a folder to a
different folder.;

%sysexec copy "&path.\Old\2012 Sales.pdf" "&path.\New\2012_Sales.pdf" ;
%sysexec copy "&path.\Old\*.*"  "&path.\New" ;


systask command  %unquote(%str(%') copy "&path.\Old\2012 Sales.pdf"
"&path.\New\2012_Sales.pdf"  %str(%'));
systask command  %unquote(%str(%') copy "&path.\Old\*.*" "&path.\New"  %str(%'));

X "move "&path.\Old\*.*"  "&path.\New" ";
%sysexec move "&path.\Old\*.*" "&path.\New" ;
systask command  %unquote(%str(%') move "&path.\Old\*.*" "&path.\New"  %str(%'));

********** Copy all files in a directory including sub-directories. *********;

X "xcopy /E "&path.\Old\*.*"  "&path.\New" ";
%sysExec xcopy /E "&path.\Old\*.*" "&path.\New" ;
Systask command %unquote(%str(%') xcopy /E "&path.\Old\*.*" "&path.\New" %str(%'));

Note: The Xcopy "/E" option allows coping directories and subdirectories, including
empty ones.
```

Please note that the three approaches discussed above are all global commands and statements, they can exist anywhere in a SAS program. However, this also implies that we cannot conditionally execute them in an open code environment. For example, if we submit below SAS code:

```
data _null_;
Action= 0;
if Action=1 then do;
X "mkdir "&path.\Try" " ;
end;
run;
```

The code will fail to work because the X statement is a global statement. It will be executed every time and the new directory Try will be created regardless of the value of Action. Therefore we cannot use a DATA step to issue conditional processing commands when using these global commands and statements. This is a big drawback of using these methods to manage external files. To achieve conditional execution, please use the CALL SYSTEM routine or SYSTEM function, which will be addressed in the following section.

The advantages of above approaches are: 1) they use simple DOS or UNIX commands, easy to learn and use; 2) they are able to perform all kinds of file management tasks; 3) they are global commands and statements which can be placed anywhere in a SAS program. However, using them also brings about significant disadvantages such as: 1) they always prompt the annoying flashing DOS window; 2) they cannot perform conditional execution; 3) it is not easy and convenient to monitor the execution status of submitted operating system commands.

II. SAS PIPING FUNCTIONALITY

Another way of managing external files and directories is to make use of the SAS piping functionality. Piping is a channel of parallel communications between SAS and other external applications[5]. For example, piping enables our SAS application to receive input from any Windows or UNIX operating system command that writes to standard output, and to route output to any Windows or UNIX command that reads from standard input. Therefore, we can utilize pipes to issue and execute operating system commands.

On newer Windows systems, we can use the unnamed pipe feature on a SAS FILENAME statement to invoke a non-SAS program and redirect the program's input, output, and error messages to SAS[6]. If an error occurs during the execution of the external program, the program will write error messages to the standard error(STDERR). By default, error messages will be written to SAS log file when SAS captures STDERR from an external program. Below is the syntax of unnamed piping with FILENAME statement.

FILENAME fileref PIPE 'program name /command ' <options> ;

The Fileref is a valid file reference following SAS naming conventions. The "program-name/command" part specifies the external application program to invoke, including the full path to the external program and program options etc. The <options> specifies the valid FILENAME statement options if required.

Analogous to the global commands and statements in Section I, we can use unnamed pipes to issue operating environment commands to manage external files. Below code gives some examples.

```
************************ II. SAS Piping Functionality. ************************;

************************ Create a new directory. ***************************;
Filename Create PIPE "mkdir  &path.\By_Pipe"; * create a new directory.;
Filename Create PIPE  %unquote(%str(%')mkdir  "&path.\By Pipe" %str(%')); *create
a new directory, need to mask and unmask the single quotation marks.;

data _null_ ;
infile Create; * Or INFILE the Rename, Delete, Copy, Move pipes as defined below.
input ;
put _infile_;
run;

******************* Rename external files and directories. ******************;
Filename Rename PIPE %unquote(%str(%') rename "Path\Directory Name"   "New
Directory Name" %str(%')); *rename a directory.;

Filename Rename PIPE %unquote(%str(%') rename "Path\Old File Name"   "New File
Name" %str(%')); *rename an external file.
```

```
****************** Delete external files and directories. ******************;
Filename Delete PIPE  %unquote(%str(%') del /q "Path\File Name"  %str(%')); *delete
an external file.;

Filename Delete PIPE %unquote(%str(%') rmdir /q "Path\Directory Name"  %str(%'));
*delete an empty directory.;

****************** Copy/Move external files and directories. ******************;
Filename Copy PIPE %unquote(%str(%') copy /q "Path\File Name" "New Path\New File
Name"  %str(%')); *copy a file to a different location with a new name.;

Filename Move PIPE %unquote(%str(%') move "Path\File Name" "New Path\New File Name"
%str(%')); *move a file to a different location.;

Filename Xcopy PIPE %unquote(%str(%') xcopy /E "Path\*.* " "New Path"  %str(%'));
*Copy all files in a directory including sub-directories to a new location.
```

For example, as shown above, when we use the unnamed pipe Create to create a new directory, the specified DOS command will be directed to the Windows operating system by piping. In the following DATA step, the INFILE statement will invoke Windows to execute the specified command to create the “By_Pipe” or “By Pipe” directory, and then read in the data in the pipe resulting from the execution. If an error occurs during the execution, for example, when the By_Pipe directory already exists, we will get below message in SAS log:

```
16         Filename Create PIPE "mkdir  &path.\By_Pipe ";
23         data Create  ;
24         infile Create;
25         input ;
26         put  _infile_;
27         run;

NOTE: The infile CREATE is:
      Unnamed Pipe Access Device,
      PROCESS=mkdir  D:\WLMU2015\Projects\By_Pipe, RECFM=V, LRECL=256

Stderr output:
A subdirectory or file D:\WLMU2015\Projects\By_Pipe already exists.
NOTE: 0 records were read from the infile CREATE.
NOTE: The data set WORK.CREATE has 0 observations and 0 variables.
```

If we give a wrong path, below error messages will appear in SAS log:

```
NOTE: The infile CREATE is:
      Unnamed Pipe Access Device,
      PROCESS=mkdir  &path2.\By_Pipe, RECFM=V, LRECL=256

Stderr output:
The syntax of the command is incorrect.
The system cannot find the path specified.
NOTE: 0 records were read from the infile CREATE.
NOTE: The data set WORK.CREATE has 0 observations and 0 variables.
```

Therefore, when using SAS pipes to manage external files, we can look into SAS log to monitor whether the assigned command has been executed successfully or not. This is an advantage of using SAS piping functionality. Other advantages of utilizing SAS pipes are: 1) it is easy to use with simple DOS and UNIX commands; 2) unlike X commands and %sysexec statement, it does NOT invoke the fleeting DOS window; 3) it is able to perform all file management tasks. Despite these advantages, its big drawback is that it does not allow conditional execution.

III. SAS CALL ROUTINE AND SYSTEM FUNCTION

CALL SYSTEM Routine

Call System is a very useful SAS call routine. Its syntax is[7]:

CALL SYSTEM (command);

The Command argument can be a character string, a character expression or a character variable which specifies a valid operating system command or the name of an external application.

This call routine functions in the same way as the X command and %SYSEXEC macro statement. It submits an operating environment command or an external program for execution. The return code generated by the operating system is also assigned to the automatic system macro variable SYSRC. However, Call System routine differs from them in: 1) it is a local command rather than a global one, therefore it must be used in a DATA step; 2) it is a callable routine, which allows conditional execution in a SAS session.

SYSTEM Function

Just like the Call System routine, the System function also issues an operating environment command for execution during a SAS session[8]. However, it is better than the Call System routine because this function returns the system return code from executing the assigned command. Usually, a return code of 0 indicates a successful execution of a command and other value for an unsuccessful execution. We can use the return code directly to monitor the execution status in the same data step, which is preferred over the SYSRC macro variable because the macro variable cannot be directly referenced.

Below code gives an example of conditionally creating new directories via the Call System routine or System function.

```
******* Create New Directories With Call System Routine or SYSTEM Function *******;
data Create;
Name="pid1280";
NewDir="&path.\"||strip(Name);
Check=fileexist(NewDir);
if Check=0 then do;
command="mkdir "||'"'||NewDir||'"';
call system(command);   *Or use the SYSTEM function shown below.;
RC=symget("SYSRC");
/*RC= &SYSRC;*/

/*RC=system(command);*/

if RC="0" then put "Note: Directories Created Successfully: " NewDir;
else put "Note: Directories Cannot Be Created: " NewDir;
end;
else if Check=1 then put "Note: Directory Already Exists: " NewDir;
run;
```

As illustrated above, we can use a DATA step to conditionally execute the Call System routine. Determined by the FILEEXIST function, if the specified directory PID1280 does NOT exist, then the Call System routine is called to create this new directory. The status notation is written to SAS log through the use of the SYSRC macro variable and the PUT statements. It should be noted that the SYMGET function is needed to retrieve the value of SYSRC macro variable. This is because a directly-referenced macro variable is resolved at the compile time which precedes the execution of the data step, but CALL SYMPUT and CALL SYSTEM routines are execution phase statements which thus have no effects on the directly referenced macro variable. For this reason, we must use the SYMGET function rather than the commented direct reference to retrieve the SYSRC value. If we use the System function, we will not have these issues since the function returns the execution code itself. Therefore, the System function is more convenient to use than the CALL SYSTEM routine.

In the same way as we have shown in Section I, CALL SYSTEM routine and SYSTEM function can be used with other DOS/Windows commands to rename, delete, copy and move external files and directories. We do not repeat them here.

The advantages of using CALL SYSTEM routine and SYSTEM function are: 1) they are callable commands allowing conditional execution; 2) they are able to accomplish all management jobs. However, they do trigger the fleeting DOS window. If it is an annoyance to you, you can choose the true SAS functions in Section IV to perform your task.

IV. NEW SAS FILE MANAGEMENT FUNCTIONS

Starting from SAS 9.2 version, several new functions have been introduced for file management uses. These new functions enable us to manage both SAS files and external files conveniently. Hereby we illustrate their uses as follows.

DCREATE Function[9]

Syntax: DCREATE (directory-name, <parent-directory>)

As its name suggests, this new function enables SAS users to create a new directory in the operating environment. The directory name argument is the new directory to be created excluding the pathname, which can be a character constant, variable or expression. The parent-directory specifies the complete pathname of the directory where the new directory will be created in, which is also a text string of a character constant, variable or expression. This function returns the complete pathname of a new, external directory if it is executed successfully. If unsuccessfully, the return value will be a blank text string. Please note the default length of the return text string is 200, and it will return a null value too if the specified directory already exists in the parent directory.

Below code demonstrates an example of using this function to create a new project folder named PID9999 in the given parent directory. Moreover, we use the return result to monitor the execution of the command through the PUT statements, which will write the execution notations to SAS log.

```
****************** Create New Directories With DCREATE Function. ***************;
data _Null_;
pid="PID9999";
NewDir="&path.\"||strip(pid);
Check=fileexist(NewDir);
if Check=0 then do;
Created=dcreate(strip(pid), "&path.");
if not missing(Created) then put "Note: Directory Created Successfully: " NewDir ;
else put "Note: Directory Cannot Be Created: " NewDir;
end;
else if Check=1 then put "Note: Directory Already Exists: " NewDir;
run;
```

RENAME Function[10]

Syntax: RENAME(old-name, new-name , < type>);

SAS provides this new function since 9.2 version, which is a very powerful function capable of working on both SAS files and external files or directories. As shown in the above syntax, the Old Name and New Name arguments specify the current name and new name of a file or a directory, both could be a character constant, variable, or expression. For a SAS data file or a catalog entry, if a library reference is used, the Old Name can be a one-level or multiple level name with the assigned library reference. The new name argument is always a one-level name regardless of the file types (SAS library member, catalog entry, external file or directory etc.). For an external file or a directory or a SAS library member without using a library reference(in this case, it is treated as an external file), a full pathname must be used for both the old-name and new name. SAS will use the current directory as the value for old-name if it is not specified. The Type argument specifies the type of element to rename which could be a character constant, variable, or expression. You can omit it for SAS files, or assign one of the following five values to it:

ACCESS: a SAS/ACCESS descriptor that was created using SAS/ACCESS software.

CATALOG: a SAS catalog or catalog entry.

DATA: a SAS data set.

VIEW: a SAS data set view.

FILE: specifies an external file or directory.

The default value of TYPE is DATA specifying a SAS data set. Some examples of using the RENAME function are given below.

```
*************** Rename files and directories using RENAME function. **************;
libname Test "&path.\Test";

data _null_;
RC1=rename("Test.AAA", "BBB", "data"); *rename a SAS data set using libref.;
RC2=rename("&path.\Test\AAA.sas7bdat", "&path.\Test\BBB.sas7bdat", "file"); *rename
a SAS data set using the full pathname.;
RC3=rename("&path.\Test\Sales.pdf", "&path.\Test\2012 Sales.pdf", "file"); *rename
an external file.;
RC4=rename("&path.\Test", "&path.\New_Test", "file"); *rename a Windows drectory.;
RC5=rename("/&path./Test", "/&path./New_Test", "file"); *rename a Unix drectory.;
run;
```

Usually, we use the RENAME function to rename a SAS file, an external file or a directory. However, we can also use it to move or delete external files or directories if we specify the New Name with a path different from that of the Old Name. Please note, in this case, for a SAS file, we cannot use the SAS library reference, we must use a full path name instead. As shown below, if we define a temporary Trash directory for storing the moved files (or just use the Windows Recycle Bin directory), we can actually utilize this function to delete unwanted files from a given directory. This is an innovative application of this function discovered by the paper authors, which provides a neat trick in using this new SAS function.

```
******************** Move or Delete Files Using RENAME function *****************;
data _null_;
RC1=rename("&path.\Test\AAA.sas7bdat", "&path.\New Test\AAA.sas7bdat", "file");
*move a SAS data set to a different directory without changing its name.;
RC2=rename("&path.\Test\AAA.sas7bdat", "&path.\New Test\BBB.sas7bdat", "file");
*move a SAS data set to a different directory and change its name.;
RC3=rename("&path.\Test\Sales.pdf", "&path.\New Test\2012 Sales.pdf", "file");
*move an external file to a different directory and change its name.;
RC4=rename("&path.\Test\Sales.pdf", "&path.\Trash\Sales.pdf", "file"); *delete an
external file by moving it to a specified Trash directory.;
RC5=rename("&path.\Test\AAA", "&path.\New Test\AAA", "file"); *move a directory
along with its contents to a different directory, without changing its name.;
RC6=rename("&path.\Test\AAA", "&path.\New Test\BBB", "file"); *move a directory
along with its contents to a different directory, and also change its name. ;
run;
```

Compared with using DOS or Unix operating environment commands, the Rename function is easier and more convenient to use. Especially when we have a huge number of files to rename, it allows us to perform conditional execution or batch processing with the combined use of SAS pipes.

```
************* Rename Multiple Files in A Windows Directory. *************;
Filename Files PIPE %unquote(%str(%') dir  "&path.\pid1280\Reports" /b %str(%'));

data Rename;
length Old_Name New_Name $60;
infile Files lrecl=256 truncover;
input  Current_Name $256.;
Old_Name="&path.\pid1280\Reports\"||strip(Current_Name);
New_Name="&path.\pid1280\Reports\"||"2014_"||strip(Current_Name);
New_Name2="2014_"||strip(Current_Name);
RC=rename(Old_Name, New_Name, 'file');

/*Command="rename "||quote(strip(Old_Name))||"  "||quote(strip(New_Name2));*/
/*RC=system(command); *Or use Call System routine. ;/*call system(command);*/

if RC=0 then put "Note: Rename Operation Is Successful--" Old_Name " ===> "
New_Name;
else put "Note: Rename Operation Is Unsuccessful--" Old_Name " ===> " New_Name;
run;
```

As demonstrated above, using the unnamed pipe Files with the DOS DIR command will retrieve all the file names in a given directory. The following DATA step will read in the file names by using the INFILE and INPUT statements. The

following RENAME function will change the current file names to new file names by adding a prefix of “2014_” to each of them, which is the year of the documents. We can also fulfill it by using the CALL SYSTEM routine or SYSTEM function as shown in the commented part.

An alternative way to achieve this goal is to create and use a batch file for batch processing, which is illustrated as follows.

```
********************* Rename Multiple Files Via A Batch File. ********************;
Filename Files PIPE %unquote(%str(%') dir  "&path.\pid1280\Reports" /b %str(%'));
Filename Bat "&path.\Rename.bat";

data Bat;
length Old_Name New_Name $50;
infile Files lrecl=256 truncover;
file Bat;
input  Current_Name $256.;
Old_Name="&path.\pid1280\Reports\"||strip(Current_Name);
New_Name="2014_"||strip(Current_Name);
if upcase(Current_Name) ="RENAME.BAT" then delete;
Command="rename "||quote(strip(Old_Name))||"  "||quote(strip(New_Name));
if _N_=1 then put "*************** Rename Files/Folders By Batch
Processing.*********; ";
put command;
run;

X " &path.\rename.bat";
```

As shown above, we use the SAS pipe to retrieve all the file names in a directory, and then use a DATA step to read the data in the pipe and create a batch text file through the PUT statement. The created batch file is then executed in the Windows environment by the X command. The advantage of using a batch file is that, we can create it on a computer with SAS, and then execute it and accomplish all the operations on a computer without SAS.

In actual fact, SAS does not have a MOVE function developed so far. But, fortunately, according to our new discovery, we can make use of the RENAME function to move external files and directories. Please refer to the code in Work Application Examples section for details.

FDELETE Function[11]

Syntax: FDELETE(fileref | directory);

This is also a new function since SAS 9.2 version, whose functionality is to delete an external file or an empty directory. Of course, it can delete SAS files as well by just treating them as external files. Please note that this function must use a fileref rather than a physical name as its argument. The fileref, assigned by using either the FILENAME statement or the FILEMANE function, must be quoted with quotation marks. The specified fileref can be a character constant, variable, or expression, however, it cannot be a text concatenation. This function returns 0 for a successful operation and other value for an unsuccessful operation. If operating on a non-empty directory, this function will do nothing but return a non-zero code.

```
***************************** Delete a SAS or External File. ********************;
filename DEL  "&path.\Test\Sales.pdf";

data _Null_;
RC=fdelete("DEL");
MSG=sysmsg();
if RC=0 then put "Note: Files deleted successfully. ";
else put "Files cannot be deleted.";
put MSG;  run;

data _Null_;
RC1=filename("DEL", "&path.\Test\Sales.pdf");
if RC1=0 and fexist("DEL")=1 then do;
RC=fdelete("DEL");
MSG=sysmsg();
if RC=0 then put "Note: Files deleted successfully. ";
else put "Files cannot be deleted.";
```

```
put MSG;
RC=filename("DEL");
end;  run;

****************************** Delete An Empty Directory. ********************;
data _null_;
RC1=filename("DEL", "&path.\Test\AAA");
if RC1=0 and fexist("DEL")=1 then do;
RC=fdelete("DEL");
MSG=sysmsg();
put RC= ;
put MSG= ;
end;  run;
```

As illustrated above, to delete a SAS file or an external file, we first employ the FILENAME statement to assign a filefef, then use the FDELETE function in a DATA step to delete it. This function will return a code of 0 if the operation is successful. We can use PUT statement to write notes to SAS log. Furthermore, we can use the SYSMSG function to return the error or warning message text from executing the FDELETE function. For example, if the file does NOT exist, we will get "WARNING: Physical file does not exist, D:\WLMU2015\Projects\Test\Sales.pdf" in log. SYSRC and SYSMSG are two useful system functions to monitor the execution of SAS functions. SYSRC function returns the error code for the last system error during executing one of the data set functions or external functions, and SYSMSG returns the error or warning message text from processing the last data set or external function. For a successful operation, SYSRC returns a zero value and SYSMSG returns a blank text.

A better way to assign the fileref is to use the FILENAME function rather than the FILENAME statement because we can use the return code from this function to perform conditional execution. The second DATA step gives the code of using FILENAME to achieve this goal. The FDLETE function is called to delete the PDF file only if its existence is determined by the return codes of FILENAME and FEXIST functions. After deleting the PDF file, we use the FILENAME function again to deassign the fileref.

The 3rd DATA step illustrates to delete a directory using FILENAME and FDELET functions, and the return code and error message will appear in log through the PUT statements. For example, if the specified directory is not empty, SAS cannot delete it and we will get below notes in log on a Windows operating system.

```
RC=20047
MSG=ERROR: D:\WLMU2015\Projects\Test\AAA is not empty and cannot be deleted.
```

Similar to the approaches illustrated for the RENAME function, if we want to delete multiple files or empty directories in a directory, we can combine the FDELETE function with SAS piping to accomplish it.

DELETE Function[12]

Syntax: DELETE(name<,type,<,password<,generation>>>);

This new SCL function has syntax similar to the RENAME function. This function can delete a SAS file, an external file or an empty directory. It returns a code of 0 for a successful operation and non-zero value for an unsuccessful operation. **Please note: this function currently exists as a SAS Component Language (SCL) function rather than a normal SAS function, therefore it can only be used within SAS/AF applications rather than in base programming.**

As we can see from above discussions, using SAS functions to manage external files has distinctive advantages over other methods because:

1) We do not need to learn about the operating environment commands such as DOS and UNIX commands.
2) They are true SAS functions and are executed in the background without invoking the annoying flashing DOS window, providing a better fit to a SAS session.
3) They have simple syntax and are easy to use in SAS programming.
4) They are callable commands and allow conditional execution and batch processing.
5) They return execution codes which can be used to monitor whether the operation has been executed successfully or not.

Therefore it is highly recommended to utilize these new SAS functions in managing external files rather than using other approaches.

V. OTHER METHODS

New DLCREATEDIR System Option[13]

SAS introduced this new global system option in 9.3 version. If this option is put in effect, when we use LIBNAME to assign a SAS library but the specified directory does not exist, LIBNAME statement will automatically create such a directory and then assign the libref. To cancel its effect, just use the NODLCREATEDIR system option, which is the default system option. For example, when we run below code whereas AAA is a non-existing directory:

```
options dlcreatedir;
libname AAA "&path.\AAA";
options nodlcreatedir;
```

SAS will create the AAA directory automatically and give below notes in log:

```
NOTE: Library AAA was created.
NOTE: Libref AAA was successfully assigned as follows:
      Engine:        V9
      Physical Name: D:\WLMU2015\Projects\AAA.
```

Therefore, we can use this new system option to create a new directory. It is important to note that, for security reasons, this option can be restricted and disabled by an administrator to prohibit unwanted and uncontrolled folder-making. To create a series of nested directories and sub-directories, we can repeatedly use the LIBNAME statement in a top-to-down order, or use a single concatenated LIBNAME statement to achieve it. Below code shows these two methods along with the SAS log messages.

```
*********Create Multi-Level Directories With Multiple LIBNAME statements.********;
options dlcreatedir;
libname PID1280 "&path.\PID1280";
libname Reports "&path.\PID1280\Reports";
libname May "&path.\PID1280\Reports\May_2014";
libname PID1280 clear;
libname Reports clear;
libname May clear;

********Create Multi-Level Directories With One Single LIBNAME statement.*******;
options dlcreatedir;
libname PID1280 ("&path.\pid1280", "&path.\pid1280\Reports",
"&path.\pid1280\Reports\May_2014");
libname PID1280 clear;

NOTE: Libref PID1280 was successfully assigned as follows:
      Levels:           3
      Engine(1):        V9
      Physical Name(1): D:\WLMU2015\Projects\PID1280
      Engine(2):        V9
      Physical Name(2): D:\WLMU2015\Projects\pid1280\Reports
      Engine(3):        V9
      Physical Name(3): D:\WLMU2015\Projects\pid1280\Reports\May_2014
```

COPY FLAT FILES USING A DATA STEP

If we want to copy flat files from a directory to another directory, we can also utilize a DATA step to do it. As shown below, through the INFILE statement SAS will load each record from the external text file into the input buffer, the INPUT and PUT statements will read from the input buffer and then write the information out to a new external text file. In effect, the code will copy the text file to a new place and change its name as well.

```
*************************** Copy one single flat file.************************;
filename Read "&path.\Test\Old\Sales.txt" ;
filename Write "&path.\Test\New\2012_Sales.txt" ;
```

```
data _null_;
infile Read;
file Write;
input;
put _infile_;
run;
```

Along with the use of SAS piping, we can copy multiple flat files from one directory to another directory. However, please note that this approach is only feasible to copy small size flat files. To copy big size flat files, it will be of low efficiency due to the huge number of I/O operations involved in the DATA step.

In addition to all the methods and approaches addressed above, we can also use CALL MODULE and CALL EXECUTE routines to invoke operating system APIs to manage external files. However, these methods require an in-depth understanding of the computer language of the operating system and present a learning curve for most SAS users. They are difficult and disadvantageous compared to other readily usable approaches, therefore we will not discuss them in this paper.

VI. WORK APPLICATION EXAMPLES

In data analysis work, we often need to organize the directories of ad-hoc and recurring projects in a good folder structure, or manage the external files for efficient project management and documentation purposes. For example, for a new project, we need log it in our project management system and then create the below file structure for it. This is a repetitive routine task and can be automated by SAS programming. We can use below SAS macro to accomplish it, which will save time and ensure accuracy.

Physical Path\WLMU2015\Projects

Pid9999: project ID.

|***Archive****: stores out-dated reports, SAS files etc.*

|***Code****: keeps SAS code and LOG, PST files.*

|***Data****: retains permanent SAS data sets, formats etc.*

|***Input****: keeps source data files (flat files, PC files etc).*

|***Output****: stores other SAS outputs.*

|***Reports****: keeps SAS-created reports (in Excel, PDF, CSV, TXT, HTML, RTF formats).*

|***PID9999_Report_Jan_2014.xls****: monthly Excel report for January 2014.*

|***PID9999_Report_Jan_2014.pdf****: monthly PDF report for January 2014.*

….

| ***PID9999_Report_Dec_2014.xls****: monthly Excel report for December 2014.*

| ***PID9999_Report_Dec_2014.pdf****: monthly PDF report for December 2014.*

Table 1. Partial Printout of the Folder Structure Controlling File (LIST.xls)

No	Folder_Name	Create
1	Archive	Y
2	Code	Y
3	Data	Y
4	Input	Y
5	Output	N
6	Reports	Y

Below SAS macro can create the above fold structure quickly. Although we can utilize the various approaches presented previously, hereby we choose to use the DCREATE function because of its distinct advantages. When using this macro, we only need to specify the physical file path and project ID for the PATH and PID macro variables. The Excel file LIST.XLS is a structure-controlling file which specifies the directories to be created, its partial printout is shown above.

```
********************* SAS Macro For Creating New Directories. ******************;
%macro Create(path=, pid=);
options mprint symbolgen ;

Filename List "physical path\List.xls";
proc import datafile=List out=List DBMS=Excel replace;
getnames=Yes;  guessingrows=1000;    run;

data Create;
set List;
Project_Folder="&path.\"||"&pid.";
Folder_Name=upcase(Folder_Name);

if _N_=1 then do;
Check=fileexist(Project_Folder);
if Check=0 then do;
put "Note: Project folder does not exist "  Project_Folder;
Created=dcreate(strip("&pid."), "&path.");
if not missing(Created) then put "Note: Project folder created successfully: "
Project_Folder;
else put "Note: Project folder cannot be created: " Project_Folder;
end;
else if Check=1 then put "Note: Project folder already exists: " Project_Folder;
end;

if Create="Y" then do;
NewDir=Project_Folder||"\"||strip(Folder_Name);
Sub_Check=fileexist(NewDir);
if Sub_Check=0 then do;
Created=dcreate(strip(Folder_Name), Project_Folder);
if not missing(Created) then put "Note: Directory created successfully: " NewDir;
else put "Note: Directory cannot be created: " NewDir;
end;
else if Sub_Check=1 then put "Note: Directory already exists: " NewDir;
end; run;
%mend;
```

Below is a second SAS macro which is developed to archive the out-dated monthly reports. If a report is older than a given duration (in months) specified by the DURATION macro variable, it will be moved to the corresponding Archive folder through the creative use of the RENAME function.

```
********************* SAS Macro For Archiving Out-dated Reports. ****************;
%macro Archive(path=, pid=, duration=);
options mprint symbolgen ;

Filename Files PIPE %unquote(%str(%') dir  "&path.\&pid.\Reports" /b %str(%'));

data List;
infile Files lrecl=256 truncover;
input  Full_Name $256.;
Report_Name=strip(scan(Full_Name, 1, "."));
Suffix=strip(scan(Full_Name, 2, "."));
N=length(Report_Name);
MTH_Year=strip(substr(Report_Name, N-7));
DD_MTH_Year="15"||substr(MTH_YEAR, 1, 3)||substr(MTH_YEAR, 5, 4);
Age=INTCK("Month", input(DD_MTH_Year, date9.), today());
run;

data Move;
length Old_Name New_Name $100;
set List;
```

```
if Age > &Duration then do;
Old_Name="&path.\&pid.\Reports\"||strip(Full_Name);
New_Name="&path.\&pid.\Archive\"||strip(Full_Name);
RC=rename(Old_Name, New_Name, 'file');
if RC=0 then put "Note: Files moved to Archive folder: " Old_Name ;
else put "Note: Files cannot be moved: " Old_Name  ;
end;
run;
%mend;
```

CONCLUSION

As discussed in this paper, managing external files and directories is an important and essential part in data analysis and business analytics. A good file management can help to streamline project management and improve work efficiency. It will be both efficient and beneficial to automate and standardize the file management processes by using SAS programming. For this purpose, we have presented and discussed various approaches and methods to organize and manage external files and directories. The illustrated methods and programming skills can have important applications in a wide variety of work fields.

REFERENCES

[1] SAS Support Website,
http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-cmd-x.htm
[2] SAS Support Website,
http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000171045.htm
[3] SAS Support Website,
http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-stmt-systask.htm
[4] SAS Support Website,
http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000543618.htm
[5] SAS Support Website,
http://support.sas.com/rnd/scalability/connect/piping.html
[6] SAS Support Website,
http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#unnamed.htm
[7] SAS Support Website,
http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-callrout-system.htm
[8] SAS Support Website,
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245956.htm
[9] SAS Support Website,
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002986745.htm
[10] SAS Support Website,
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a003122440.htm
[11] SAS Support Website,
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245893.htm
[12] SAS Support Website,
http://support.sas.com/documentation/cdl/en/sclref/59578/HTML/default/viewer.htm#a000143503.htm
[13] SAS Support Website,
http://support.sas.com/documentation/cdl/en/lesysoptsref/64892/HTML/default/viewer.htm#n1pihdnfpj4b32n1t62lx0zdsmdn.htm

ACKNOWLEDGEMENTS

CONTACT INFORMATION

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

Justin Jia Trans Union Canada, Burlington, Ontario, Canada

Amanda Lin CIBC, Toronto, Ontario, Canada