Paper 021-29

Backing up File Systems with Hierarchical Structure Using SAS/CONNECT

Fagen Xie, Kaiser Permanent Southern California, California, USA
Wansu Chen, Kaiser Permanent Southern California, California, USA

Abstract

SAS programmers routinely backup data files between local and remote systems to avoid loss of data. People also back up data files regularly from development systems where files are purged periodically. However, SAS procedures UPLOAD and DOWNLOAD do not easily handle situations where data and program files are stored in a hierarchical structure. This paper introduced a recursive technique to transfer files and directories stored in a hierarchical structure between a local host and a remote host by extending the use of *UPLOAD* and *DOWNLOAD* procedures in SAS/CONNECT software. This technique applies to all operating systems. Also, the concept of calling a macro recursively within a macro can be easily applied to other circumstances.

Introduction

Program developers and data analysts routinely backup application data or project files between local and remote computer systems to avoid loss of data caused by system crash or virus intrusion through Internet. People also back up data files regularly from development systems where files are purged periodically. With the SAS procedures *UPLOAD* and *DOWNLOAD,* users can easily transfer or back up program or data files, catalogs, views and multi-dimensional database (MDDB) between a local host and a remote host (1). In the SAS/CONNECT manual, there is a simple demonstration showing how to create backup copies on a remote host of important local host files (2). However, the above two procedures do not easily handle situations where data and program files are stored in a hierarchical structure. This paper introduced a technique to transfer files stored in a hierarchical structure between a local host and a remote host by extending the use of *UPLOAD* and *DOWNLOAD* procedures in SAS/CONNECT software. In our sample programs, we used a PC as a local host and a UNIX system as a remote host. This application can be extended to any other operating systems with only slight modification.

Sign on to the Remote Host

Before a user performs backup, the first step is to initiate a connection from a local SAS session to a remote host by issuing the following SAS/CONNECT statements. Once the following "SIGNON" process is successful, a user can access the remote SAS session through the SAS/CONNECT link from the local host.

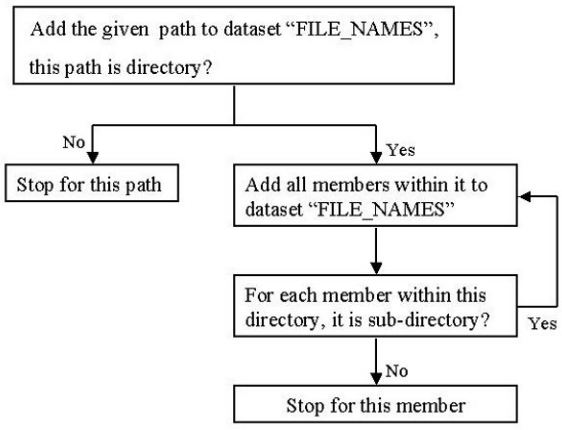
```
options   remote=remote-session-id comamid=communications-method;
filename rlink 'tcpunix.scr';
signon;
```

The "remote-session-id" specifies the name of the remote session to which a user wants to sign on. The "communications-method" can vary by host. Valid access method is TCP/IP between PC and Unix. Between PC and OS/390, it can be APPC, TCP/IP, EHLLAPI or TELNET. In this paper, we used "TCP". The "tcpunix.scr" is an external file on the local host that contains statements that control the connection. Scripts describe username, password, start up information of SAS for the remote SAS session and error handling messages. The username and password can be explicitly written in the script file or manually typed into the prompted window during SIGNON submission. Users can employ one of the sample scripts that are provided by the SAS Institute for various operating systems. These scripts can be found in the SAS/CONNECT product in your SAS system.

Recording Files and Directories with a Hierarchical Structure from Source Host

The **source host** refers to the host **from** which files and directories are transferred. In our example, it can be either a PC (local host) or a Unix (remote host).

To back up hierarchical files and directories, the key is to obtain all the detailed information including file names, directory names and names of their offspring within a given path in the source host. We allow the main root specified by 'frommain' to be either a file name or a directory name. The first step of the program is to check whether 'frommain' refers to a file name or a directory. If it is a file, the file name is saved to a data set called "FILE_NAMES". If it is a directory, we save the directory name and all members within the directory including files and sub-directories to "FILE_NAMES". For each member within this directory, we test whether the member is a file or a sub-directory. If a member is detected to be a sub-directory, we repeat the above step. The search goes on within each sub-directory until no more sub-directories can be found. The flow chart below shows the logic.



To achieve this goal, we developed the following SAS macro "printdir" to search and store the file and directory path names recursively. The macro calls itself repeatedly until there is no more sub-directory can be found.

```
%let  fromsep=\;
%let   tosep=/;
%let   mstart=1;
%let  frommain=C:\Userid\ medical_research;
%let   tomain=/home/Userid/medical_research;


%macro printdir(dir);
%local   maindir maintmp tmp j flag m_num;
%let      maindir=&&&dir;
%let     maintmp=&maindir;
data temp;
      rc=filename("file",trim(symget("maindir")));
           did=dopen("file");
           if   did > 0 then  do;
         call symput("flag",1);
         call symput("m_num",dnum(did));
                                  end;
    else call symput("flag",0);
run;
%if &m_num > 0 %then %do;
   %do i=%eval(&mstart) %to %eval(&m_num);
     %local m&i;
   %end;
                                %end;
data temp;
    rc=filename("file",trim(symget("maindir")));
   if did > 0 then do;
  do i=%eval(&mstart) to dnum(did);
                  call symput("m"||trim(left(i)),trim(dread(did,i)));
               end;
                           end;
run;

%if &flag  %then %do;
 data temp1;
    length file $200 file1 $200;
    file=symget("maintmp");
           flag='1';
           file1=tranwrd(file,trim(symget("frommain")),trim(symget("tomain")));
           file1=translate(file1,trim(symget("tosep")),trim(symget("fromsep")));
 run;
 data FILE_NAMES;
    set FILE_NAMES temp1;
 run;
```

```
  %do j=%eval(&mstart) %to %eval(&m_num);
    %let  tmp=m&j;
    %let frompath=&maintmp&fromsep&&&tmp;
    %printdir(frompath)
  %end;
                          %end;
%else %do;
  data temp1;
     length file $200 file1 $200;
     file=symget("maintmp");
           flag='0';
           file1=tranwrd(file,trim(symget("frommain")),trim(symget("tomain")));
           file1=translate(file1,trim(symget("tosep")),trim(symget("fromsep")));
  run;
  data FILE_NAMES;
     set FILE_NAMES temp1;
  run;
        %end;
%mend;

* initiating  an empty dataset "FILE_NAMES" used to save the all file and directory path
names;
 data FILE_NAMES;
 run;
%printdir(frompath);
```

The top six lines define the global macro variables. The variables “fromsep” and “tosep” store the delimiters of a path for the source host and the destination host, respectively. When a PC is the source host (transferring files from PC to UNIX), “fromsep” and “tosep” are set up to “\” and “/”, respectively. When a Unix is the source host (transferring files from UNIX to PC), we set “fromsep=/” and “tosep=\” (reversed).

“mstart” is defined to indicate the starting position when we count members within a directory. Since SAS functions “dnum()” returns the total number of members including “.” and “..” in a Unix system, we assign “mstart” to “1” if a PC is the source host and “3” if a Unix is the source host.

The variables “frommain” and “tomain” store the main root for the source host and the destination host, respectively.

The first part of the macro (first “temp” data step) is to check whether or not a given path is a file name or a directory name using function “dopen()”. If it is a directory name, the total number of members within this directory including files and sub-directories is assigned to “m_num” and “1” is assigned to “flag”. When flag=1, the second part of the macro (second “temp” data step) creates “m_num” local macro variables to store path names for all the members within the directory. The third part of the macro uses a DO

loop to check whether or not any of the members between “mstart” and “m_num” is a sub-directory. If it is, the macro references itself recursively until no more sub-directories can be found.

The above SAS macro needs to be executed on the source host. If the source host is the remote host (Unix), the macro must be submitted from the local host (PC) to the remote host (Unix) and executed on Unix remotely.

Examining Existence of Directories and Preparation for Making Non-existing Directories in Destination Host

The **destination host** refers to the host **to** which files and directories are transferred. In our example, it can be either a PC (local host) or a Unix (remote host).

Before we copy the directories from the source host to the destination host, we first need to check the existence of these directories that are stored in the data set FILE_NAMES.

When the Unix is the source host (transferring files and directories from Unix to PC, or referred to as download), we first need to transfer file FILE_NAMES from Unix to PC and then identify directories that do not exist on the PC. The file FILE_NAMES is updated to include a flag called “NEW_FLAG”. This flag indicates whether or not a given directory exists in the destination host.

When the PC is the source host (transferring files and directories from PC to UNIX, or referred to as upload), we first need to transfer file FILE_NAMES from PC to Unix and then identify directories that do not exist on the Unix. Same as mentioned above, “NEW_FLAG” is added into the file FILE_NAMES. Finally the updated file ‘FILE_NAMES’ is transferred back to PC.

For directories that do not currently exist in the destination host (NEW_FLAG=’’), the program creates a script called ‘backup.sas’ in the ‘data _null_” step. This script contains SAS codes for making these directories by using ‘X’ statement. ‘X’ can be used anywhere in a SAS program to execute an operating system command within a SAS session. This script will be referenced and executed in the next step. The inclusion of “rsubmit” and “endrsubmit” in this data step allows the X statement to run on Unix when Unix is the destination host (uploading files from PC to Unix).

```
%macro getfiles;
%if &backup   = download  %then %do;
  rsubmit;
  proc download data=FILE_NAMES;
  run;
  endrsubmit;
  data FILE_NAMES;
      set FILE_NAMES;
```

```
      if  trim(file1)=' ' then delete;
      if  fileexist(trim(file1)) then do;
      f1="f"||trim(left(_n_));
      rc=filename(f1,trim(file1));
            did=dopen("f1");
            if flag='1' and did > 0 then new_flag='1';
                                    end;
  drop rc did f1;
  run;
                        %end;
%else  %do;
  rsubmit;
  proc upload data= FILE_NAMES;
  run;
  data FILE_NAMES;
      set FILE_NAMES;
       if  trim(file1)=' ' then delete;
       if  fileexist(trim(file1)) then do;
      f1="f"||trim(left(_n_));
       rc=filename(f1,trim(file1));
      did=dopen(f1);
      if flag='1' and did > 0 then new_flag='1';
                                    end;
  drop rc did f1;
  run;
  proc download data= FILE_NAMES;
  run;
  endrsubmit;
     %end;
%mend getfiles;

%getfiles

* write the script to create all neccessary directories;
data _null_;
   set FILE_NAMES  end=final;
        if _n_=1 then do;
            file 'C:\temp\backup.sas' RECFM=V LRECL=300;
                if upcase(trim(symget("backup")))="UPLOAD" then put "rsubmit;";
                    end;
        if flag='1' and new_flag=' ' then do;
            file 'C:\temp\backup.sas';
            put "X 'mkdir " file1 "';";
                                    end;
        if final then do;
            file 'C:\temp\backup.sas';
            if upcase(trim(symget("backup")))="UPLOAD" then put "endrsubmit;";
```

```
            end;
run;
```

Backing up Files and Directories from Source Host to Destination Host

The first portion of this step is to generate SAS statements for transferring files that are recorded in file 'FILE_NAMES' using either PROC UPLOAD or PROC DOWNLOAD. These statements are appended to the end of SAS script 'backup.sas' that was created in the prior step. At this point, the SAS script 'backup.sas' contains statements necessary for making new directories and transferring files between PC and Unix.

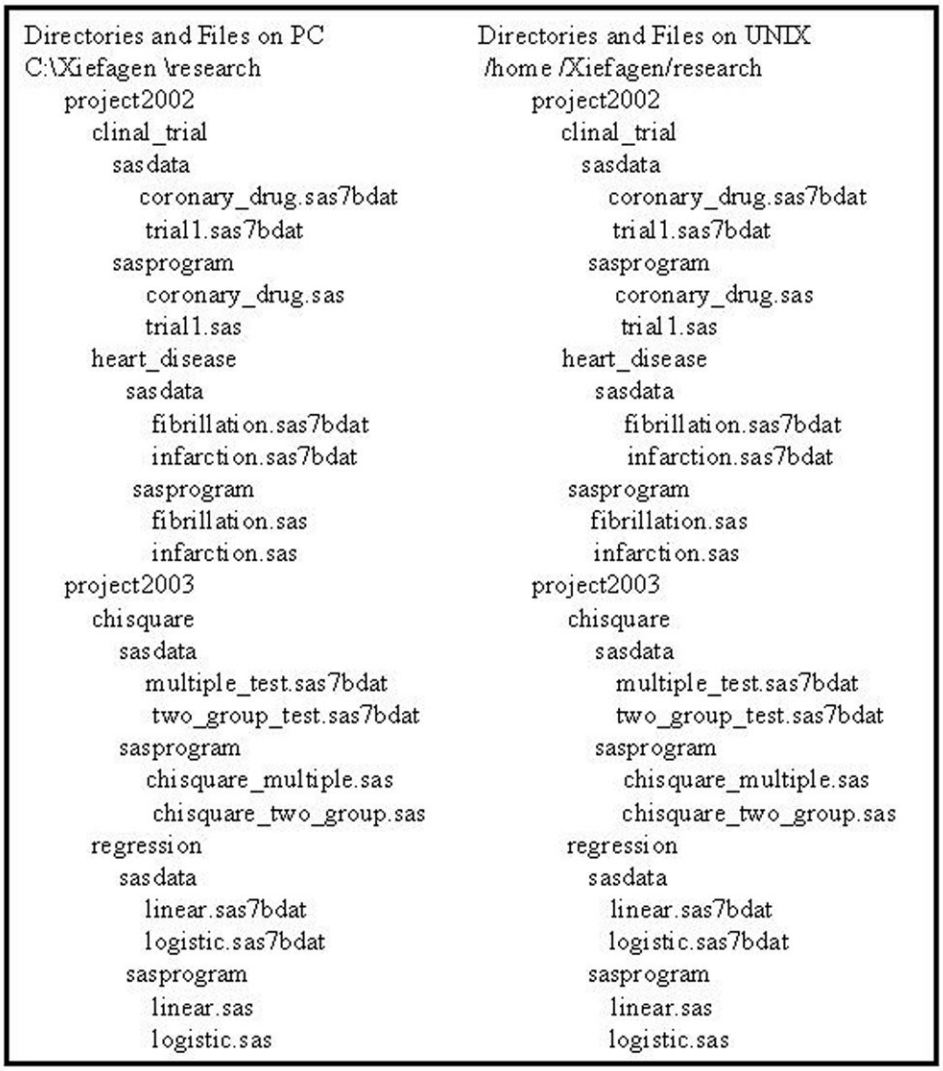
Finally, the SAS "DM" statement opens the program editor window and includes all the statements stored in 'backup.sas'. A 'DM' statement submits SAS program editor, log or procedure output commands as SAS statements. The default window is the program editor unless otherwise stated. Users can submit the SAS statements to begin the backup process. If one file exists in both hosts, the one in the source host always replaces the one in the destination host.

The last statement "SIGNOFF" allows users to log off from the remote host.

```
* write the script to upload or download all files;
data _null_;
   set FILE_NAMES end=final;
        if flag='0' then do;
            if trim(file1) > ' '  then do;
            file 'C:\temp\backup.sas' mod;
            put "rsubmit;";
            put "proc &backup binary infile='"file"' outfile='" file1"';";
            put "run;";
            put "endrsubmit;";
                                    end;
                 end;
        if  final then do;
             file 'C:\temp\backup.sas' mod;
             put "* signoff from remote host;";
                  put "signoff;";
                   end;
run;
* using DM command to open program editor to include 'backup.sas' file, and submit
  the program from that window to download or upload files & directories;
dm "inc 'C:\temp\backup.sas'";
signoff;  * shut down the remote SAS session, and log out from remote host.
```

Sample Files and Directories

We successfully backed up all files and directories in the following main root "C:\Xiefagen\research" from PC (local host) to UNIX (remote host) using the above programs. The following figure displays the files and directories in both PC and UNIX systems.



Conclusion

In this paper, we designed a recursive macro to back up files and directories with a hierarchical design between local host and remote host by using SAS/CONNECT software. Our work extended the application of SAS procedures DOWNLOAD and UPLOAD by allowing the existence of directories within a directory.

The major advantage of this method is that it only requires the main root path information in both the source host and the remote host. All files and directories within the root are transferred automatically to the destination host. Minimal manual operation is required.

The concept of calling a macro recursively within a macro can be easily applied to other circumstances.

References

1. SAS OnlineDoc available at http://v8doc.sas.com/sashtml. SAS Institute Inc., Cary, NC, USA, 1999.

2. SAS institute Inc., SAS/CONNECT Software: Usage and Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 31 pp.

Contact Information

Your comments and suggestions are greatly appreciated. Request for the programs or questions can be directed to the following authors.

Fagen Xie
Second Floor (Research & Evaluation)
100 S. Los Robles
Pasadena, CA 91101
Work Phone: (626) 564-3294
Email: fagen.xie@kp.org

Wansu Chen
Second Floor (Research & Evaluation)
100 S. Los Robles
Pasadena, CA 91101
Work Phone: (626) 564-3475
Email: wansu.chen@kp.org