PhUSE US Connect 2018

Paper SM-08

SAS and UNIX: rerunning your batches in a smarter way

Yaroslav Haiovyi, Experis/Intego Group, Kharkiv, Ukraine

ABSTRACT

Every programmer who works with huge amounts of data knows that it takes a lot of time and computational resources to rerun all the datasets and outputs every time after receiving a data update. Sometimes not all raw datasets received by a team are changed, but it is necessary to rerun all programs based on new data anyway. This paper is going to provide a review of UNIX tools that help a programmer to do this in a smarter way, namely the "makefile" command, which is usually included in UNIX systems, and to demonstrate a SAS® program which can be helpful in such situations. The algorithm is based on the idea that not all programs need to be rerun, but only those depending on the datasets that have been changed.

INTRODUCTION

Everyone who works as an SPA programmer and is responsible for rerunning all programs in a whole study knows how long it takes to rerun all programs. Usually this rerun requires one file with commands (further in this paper it is called the "runbatch" file), which is created manually and runs all programs step by step. It's OK if it is phase I or phase II, but if it is phase III or phase IV with a lot of patients and different assessments, it may take much more time. Moreover, if, for example, it is a Final CSR run, it may take more than 2 hours to rerun all analysis datasets and TLGs (tables, listings, graphs).

The question is as follows: "Can programmers save time and resources of the remote server where all these operations take place?". This paper will show you a possible solution to this question. UNIX environment, EXCELL® sheets and the SAS® program have been used to create a special file called the "makefile". This file in combination with the UNIX "make" command gives a powerful tool to manage files, use UNIX commands, check date stamps, use IF logic, etc. This tool with some hints which are part of the UNIX system may help to rerun a batch much faster. This paper describes the process of creating a "makefile" with the help of the SAS® program as well as some hints which allow to reduce the number of programs that need to be rerun. This allows to save time and resources of the remote server.

SOFTWARE ENVIRONMENT

1) UNIX environment with support for the "make" command and command line.
2) SAS® environment to develop and run the program (with the ability to read/write files in EXCELL® formats) in the UNIX environment, with support for the command line. A SAS® programmer needs to have the ability to create files in the folder where the program is located.
3) MS EXCELL® or any other program with the ability to create/change *.xls / *.xlsx files (SAS® can be used instead).

BATCH RUN

Programmers usually have a list of outputs which need to be created. These files have their own titles, footnotes, filters which should be added to the program. All of these things can be written in *.xls file format. A lot of other information may be added there. The serial number of the output, the program which creates the output, the UNIX command which creates the output etc., may be added to the *.xls list.

So, what do programmers do now? They create one "runbatch" file for all the programs or different "runbatch" files that run datasets, QC of datasets, TLGs and QC of TLGs separately.

Information from the *xls file moves to "runbatch" manually or via script. Every time new data is incoming the programmer needs to rerun all these "runbatches" to be sure all the outputs are up-to-date. If they work with Phase III on Final CSR, there may be over a thousand patients and different assessments and some datasets may have 1 million and more observations. Moreover, it may be about 500 outputs required to be done. AE (Adverse Events), LB (Laboratory Analysis) outputs from the safety part and/or outputs for efficacy analysis sometimes take a lot of time to run and get results. That's why this run usually takes place in an early morning or at the end of the working day. All the main work on the study is interrupted when it is being run.

This should be done every time the Team receives new data. Suppose that the incoming data is SDTM data. The issue is SAS® programmers don't get all the SDTM data updated. One batch of updates may have major updates (almost all SDTM datasets have been updated), while another batch may have a situation where only AEs have been updated. In both situations the programmer needs to use "runbatch" and rerun all the outputs even if they are not related to the updated SDTMs. It's obvious that the time to be spent to run non-related outputs and datasets is a wasted time. The programmer gets the same results but "runbatch" can't choose which command should be run and which shouldn't.

Here is an example of "runbatch" code:

```
X "sas ad_asl.sas"
X "sas qc_ad_asl.sas"
X "sas ad_aefga.sas"
X "sas qc_ad_aefga.sas"
...
```

This code runs strictly line by line with some possible interruptions. It's not the best way to run a batch. The time of programmers, the resources of the server where the batch runs can be reduced, which is shown in the following sections of the paper.

STEP BY STEP SOLUTION

SAS® allows to use not only *.sas and *sas7bdate files but also read/write files of other types. This helps to develop the program which is the main topic of the paper. The following steps should be taken to get results:

1) Creating an EXCELL® (*.xls or *.xlsx) list with information manually.
2) Run the SAS® program which uses information of the *xls file to create a "makefile".
3) Run the "makefile" in the UNIX environment to get result datasets or outputs which should be up-to-date.

*.XLS FILE FORMAT

Here is just an example of a possible *.xls file which may be used to create a "makefile". Of course, there might be titles, footnotes for TLGs, names with labels of datasets, etc., but in this situation only the columns necessary for the SAS® program have been simulated. The file "Lib.xls" consists of three lists: "LIB_", "DATA_", "TLG_".

1) The "LIB_" list contains libnames which should be used to find all data which needs to be rerun. UNIX commands like ".", ".." and UNIX shell commands (defined by user) "$SDTMFOLDER1" are allowed. Here is an example list:

```
Libname_ Path_
sdtm_1   $SDTMFOLDER1/project1/study1
sdtm_2   $SDTMFOLDER2/project2/study2
sdtm     ./SDTMFOLDER
ADAM     ./ADAMFOLDER
outputs  ./outputsfolder
```

This is an example of an OLE study where two parent studies (study_1, study_2) were merged into one OLE study (study_3). Libname "ADaM" is used for result analysis datasets, "outputs" is used for result outputs (TLGs).

- The "Libname_" column contains the name of libraries that are used in the SAS® program.
- The "Path_" column contains the path to the folders where the files are located.

2) The "DATA_" list contains information that is necessary to run the source dataset program and the QC program if needed. Also, there is information about SDTM datasets ADaMs is based on.

```
Libname_ Dataset_   Relations_Data     Relations_Prog
sdtm     dm         _empty_            _empty_
sdtm     ae         _empty_            _empty_
sdtm     za         _empty_            _empty_
ADAM     asl        sdtm.dm            ad_asl.sas
ADAM     aefga      sdtm.ga ADAM.asl   ad_aefga.sas

Command_1           Command_QC
_empty_             _empty_
_empty_             _empty_
_empty_             _empty_
sas ad_asl.sas      sas qc_ad_asl.sas
sas ad_aefga.sas    sas qc_ad_aefga.sas
```

The main columns are as follows.

- "Libname_" is the same column as in the "LIB_" list. It shows the short name of the place where the related file is located.
- The "Dataset_" column contains the names of datasets without format (*.sas7bdat).
- The "Relations_Data" column contains relations between the files. For example, the SDTM.DM dataset is not based on any other files because it has been unzipped from the archive of a new data batch. That's why there is the key word "_empty_". At the same time, ADaM ADAM.AEFGA is based on two datasets SDTM.ZA and ADAM.ASL (this is just an example of possible relations).
- "Relations_Prog" contains the name of the program which creates a dataset and (possibly) the name of an external file which needs to be used in the process of creation.
- The "Command_1" column contains information about the UNIX command which runs the 1-line program.
- The "Command_2" column contains information about the UNIX command which runs the QC program.

3) The "TLG_" list contains information about result outputs, ways of running and information about relations with other files

```
Libname_ Output_           Relations_Data         Relations_Prog
reports  t_ga.out          ADAM.asl ADAM.aefga   t_ga.sas
reports  l_ae_luc.out      ADAM.asl              l_ae_luc.sas

Command_1           Command_QC
sas t_ga.sas        sas qc_t_ga.sas
sas l_ae_luc.sas    _empty_
```

The main columns are as follows:

- "Libname_" is the same column as in the "LIB_" and "DATA_" list. It shows the short name of the place where a related file is located.
- "Output_" contains the name of the output which is created by the program.

- The "Relations_Data" column contains all datasets used to create the output.
- "Relations_Prog" contains the name of the program which creates the output.
- The "Command_1" column contains information about the UNIX command which runs the 1-line program.
- The "Command_2" column contains information about the UNIX command which runs the QC program.

WHAT IS "MAKEFILE"

"makefile" is a powerful tool which allows to develop user-defined rules to manage files and run commands. A simple makefile consists of "rules" of the following form:

```
target... : prerequisites ...
        commands
        ...
        ...
```

A **target** is usually the name of a file that is generated by a program. A target can also be the name of an action, but in this case it's better to look at the target as at the result of the programs' run (dataset and/or output).

A **prerequisite** is a file used as an input to create the target. A target often depends on several files. In this case datasets and the program itself are the prerequisites for the current target.

A **command** is an action that should be run. A rule may have more than one command, each on its own line. Please note: you need to put a tab character at the beginning of every command line! In this case "sas *** " or any other execute commands should be written to get the result dataset or output.

MAIN SAS® PROGRAMM

This section of the paper shows a step by step process of creating the "makefile".

1) the first step is reading all the three lists from "Lib.xls". Here is the part of code:

```
proc import
   datafile = './Lib.xlsx'
   DBMS     = xlsx
   OUT      = Work.LIB_
   sheet="LIB_";
run;
```

2) The next step is very important. The program has divided all datasets and outputs into 6 groups in the following order: "RAWs_and_Other", "SDTMs", "ADaMs", "Tables", "Listings", "Graphs". The division uses the following principles. First, the program takes information from the "DATA_" list and sorts it into groups "RAWs_and_Other", "SDTMs", "ADaMs". If there are no files in group, it will be deleted. SDTM files are usually kept in one folder, and this may be checked by the program. If the dataset is in the sdtm folder, this dataset belongs to the group "SDTMs". The same is the case with ADaMs. All other datasets belong to the highest priority group "RAWs_and_Other". It means that the first part of the "makefile" should check "RAWs_and_Other" datasets. Then the "makefile" should check the "SDTMs" group of datasets (and possibly run, if programmers receive SDTMv data from RAW files). Then the "makefile" should check and run the "ADaMs" group of files. Second, the program reads data from the "TLG_" list. Outputs may be simply divided into groups by their names. By naming convention, the name of the programs producing the tables begins with "t_", the name of listings begins with "l_", graphs – with "g_".

3) The next step is to check if the "makefile" already exists. This step takes place here because it's better to have a new version of the "makefile", only if all the previous steps run without errors.

```
%macro is_Makefile_exists( );
  %if %sysfunc(fileexist(&MakeFilename)) %then %do;
     X "rm ./&MakeFilename";
     X "touch &MakeFilename";
  %end;
  %else %do;
     X "touch &MakeFilename";
  %end;
%mend is_Makefile_exists;
```

Every time the programmer runs a SAS® program, they get a new version of the "makefile". It is required because ideally the *.xls file changes every time it is necessary to add some new output or remove the old one.

4) The next step is sorting each group. This needs to be done to prevent situations where AE runs before ASL. So the program checks relations and puts files in the right order.

5) Finally, the last step is to create the "makefile" using the SAS® program. This program consists of the following parts:

Preprocessing part. This part of code is necessary if a SAS® programmer wants to use additional makefile options. For example:

```
.IGNORE:  # option = ignore errors during run of makefile
```

Libraries part. The program creates all ways to the files shown in the *.xls file in the libraries part.

```
# <BEGIN> LIBRARIES  ==============================
sdtm_1   $SDTMFOLDER/project1/study1
sdtm_2   $SDTMFOLDER/project2/study2
# <END> LIBRARIES    ==============================
```

List and Run parts for each of the groups of files. The list group contains all names of files in the group which should be checked and possibly run. The Run group contains checks and commands to get result files from the List part. The first group of files is "RAW and Other" which contains RAW files and some specific datasets, for example with PK data, etc. which may be located in other folders, but their priority is the highest. The next is the "SDTMs" group, the "ADaM"s group, and the outputs group which is divided into three subgroups: "Tables" subgroup, "Listings" subgroup, "Graphs" subgroup. Here is an example of the ADaM List and the Run group.

```
# <BEGIN> ADaM datasets  ==============================
ADaMs:   \
$ADAM/asl.sas7bdat  \
$ADAM/aefga.sas7bdat     \
# <END> ADaM datasets    ==============================

# <BEGIN> RUN: ADaM datasets ==============================
$ADAM/asl.sas7bdat :     \
$sdtm/dm.sas7bdat \
ad_asl.sas
   sas ad_asl.sas
   sas qc_ad_asl.sas

$ADAM/aefga.sas7bdat :   \
$sdtm/dm.sas7bdat   $ADAM/asl.sas7bdat \
ad_aefga.sas
   sas ad_aefga.sas
```

```
    sas qc_ad_aefga.sas

# <END> RUN: ADaM datasets ==============================
```

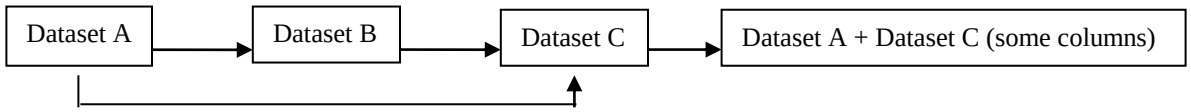
6) The "makefile" is ready to use.

HINTS OR "WHAT WAS/SHOULD BE APPROVED"

1) Loops. Ideally, a SAS® program should have the statement

```
x "make -&MakeFilename"
   /*where &MakeFilename contains the current name of makefile.*/
   /*It may be a lot of different makefiles in one folder.*/
```

to run the "makefile". But it is better to look at this file before running it. Just to be sure that everything is OK. There might be some impact of mistakes on the step of ordering the data run. For example, there might be conflicts when the data are going to be run with recursion: dataset C based on datasets B and A, dataset B based on dataset A, while A needs to be rerun after C, because it takes additional columns from it.



Situations like this were not added to the program because this is not a Good Programming Practice to create loops like this. Of course, if this happens, the programmer needs to check the "makefile" and add additional steps or move the existing one.

2) Overloaded files (for example, ASL dataset). It is obvious that the ASL dataset uses a lot of SDTM datasets and all other ADaMs use ASL as a prerequisite. There might be a situation, when only AE has been changed, but this doesn't impact the ASL dataset. In this case all ADaMs should be rerun, and, of course, all TLGs should be rerun, too. This takes time and resources which need to be saved in this case. The question is how to save time and how to keep the timestamp of the result output if it hasn't been changed after the rerun. Here the programmer can use a hint which allows to compare the new and old datasets and use the old one in case of equality, while using the new one in case of inequality. The hint hides in the "mv" command of the UNIX system. If a file has been renamed by this command, it will save its datestamp. Previously, the program had following instructions: if the datestamp of prerequisites is newer than the result file, the command is run and rewritten. Now, one more thing needs to be added: if the datestamp of prerequisites is newer than the result file, the program renames the old file, runs the command, and then compares the datasets. If they are equal, it removes the new one and renames back the old one. The hint is to save the old datestamp to avoid the chain of unnecessary runs.

CONCLUSION

This file may help programmers to save their time and server resources while running programs on new data. Yet, this is not recommended to be used on the final run since during final runs all the datastamps need to be the same. It means that the "makefile" cannot replace "runbatch" in all cases. Nevertheless, it is easy to use this SAS® program as a point of creating not only the "makefile" but also the "runbatch".

REFERENCES

1. Makefile manuals: ftp://ftp.gnu.org/old-gnu/Manuals/make-3.79.1/html_chapter/make_2.html
2. SAS® 9.4 language reference: http://documentation.sas.com/?docsetId=lrcon&docsetTarget=p0sz8gq6nvzcojn13pcqe3twvg1d.htm&docsetVersion=9.4&locale=en

CONTACT INFORMATION

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

Author Name: Yaroslav Haiovyi
Company: Experis/Intego Group
Address Gagarin Avenue 43/2
City / Postcode: Kharkiv 61001, Ukraine
Work Phone: +380 44 500 7020 ext. 2418
Email: yaroslav.haiovyi@intego-group.com

APPENDIX: THE BODY OF SAS PROGRAM

```
options mprint;
* CHANGABLE options;

%LET SDTMlib    SDTM ;   /* index(&SDTMlib)>0 => SDTM */
%LET ADAMlib    ADAM ;   /* index(&ADAMlib)>0 => ADAM */
%LET TableChk   t_   ;   /* substr(VAR,"2","1")=&TableChk */
%LET ListingChk l_   ;   /* substr(VAR,"2","1")=&ListingChk */
%LET GraphChk   g_   ;   /* substr(VAR,"2","1")=&GraphChk */
*****************************************************************;
%LET MakeFilename Makefile;
filename MFile "./&MakeFilename" ;
%LET EndLine '0A'x; /* WIN='0D0A'x ; UNIX= '0A'x */
%LET Tab '09'x; /* Tabulation */
*****************************************************************;
%macro is_Makefile_exists( );
  %if %sysfunc(fileexist(&MakeFilename)) %then %do;
     X "rm ./&MakeFilename";
     X "touch &MakeFilename";
  %end;
  %else %do;
     X "touch &MakeFilename";
  %end;
%mend is_Makefile_exists;
*****************************************************************;
*               1-st step: Reading data from LIB.xlsx file      ;
*****************************************************************;
/*libnames*/
proc import
   datafile = './Lib.xlsx'
   DBMS     = xlsx
   OUT      = work.LIB_
   REPLACE;
   sheet="LIB_";
run;
/*Datasets*/
proc import
   datafile = './Lib.xlsx'
   DBMS     = xlsx
   OUT      = work.DATA_
   REPLACE;
   sheet="DATA_";
run;
/*TLGs*/
proc import
   datafile = './Lib.xlsx'
   DBMS     = xlsx
   OUT      = work.TLG_
   REPLACE;
   sheet="TLG_";
run;

proc sort data= work.LIB_;    by LIBNAME_;  run;
proc sort data= work.DATA_;   by LIBNAME_;  run;
```

```
data work.DATA_; length LIBNAME_ $200.; set  work.DATA_; run;
data work.LIB_;  length LIBNAME_ $200.; set  work.LIB_; run;

data work.DATA_;
   merge work.DATA_(in=d_) work.LIB_;
   by LIBNAME_;
   if(d_);
run;

*************************************************************;
* add missing values if we don't need to rerun these files;
data work.DATA_;
   set work.DATA_;
   attrib FlagN     length=8.
          FlagDescr length=$20.;
   NUM_=_N_;
   if       (index(LIBNAME_,"&SDTMlib")>0) then do; FlagN=2;
FlagDescr="SDTMs"; end;
   else if (index(LIBNAME_,"&ADAMlib")>0)  then do; FlagN=3;
FlagDescr="ADAMs"; end;
   else                                          do; FlagN=1;
FlagDescr="RAWs_and_Other"; end;
run;

data work.DATA_;
   set work.DATA_;
   if (RELATIONS_DATA = "_empty_") then RELATIONS_DATA=" ";
   if (RELATIONS_PROG = "_empty_") then RELATIONS_PROG=" ";
   if (COMMAND_1  = "_empty_") then COMMAND_1 =" ";
   if (COMMAND_QC = "_empty_") then COMMAND_QC=" ";
   RELATIONS_DATA=compbl(RELATIONS_DATA); /*If we have seq of
spacebars, it replace this by 1 spacebar*/
run;
************************************;
data work.TLG_;
   set work.TLG_;
   attrib FlagN     length=8.
          FlagDescr length=$20.;
   NUM_=_N_;
   if (index(OUTPUT_,"&TableChk")  =1)  then do; FlagN=4;
FlagDescr="Tables"; end;
   else if (index(OUTPUT_,"&ListingChk")=1)then do; FlagN=5;
FlagDescr="Listings"; end;
   else if (index(OUTPUT_,"&GraphChk")  =1)then do; FlagN=6;
FlagDescr="Graphs"; end;
run;

data work.TLG_;
   set work.TLG_;
   if (RELATIONS_DATA = "_empty_") then RELATIONS_DATA=" ";
   if (RELATIONS_PROG = "_empty_") then RELATIONS_PROG=" ";
   if (COMMAND_1  = "_empty_") then COMMAND_1 =" ";
   if (COMMAND_QC = "_empty_") then COMMAND_QC=" ";
   RELATIONS_DATA=strip(compbl(RELATIONS_DATA)); /*If we have seq of
spacebars, replace this by 1 spacebar*/
```

```
run;
*************************************************************************;
*  Sort datasets ;
*************************************************************************;
%macro SortData( );

%LET MaxWords=0;
%LET RealOrder=0;
%LET MaxObsDataset=1000;

data work.DATA_(drop= tmp );
   set work.DATA_ END=Last_Line;
   retain retTmp 0;
   delim = ' ,';
   tmp=countw(RELATIONS_DATA, delim);
   if (tmp>retTmp) then retTmp=tmp;
   array Rel(50) $60.;
   length DATANAME $60.;
   DATANAME=strip(LIBNAME_)||"."||strip(DATASET_);
   do i = 1 to tmp;
      Rel[i] = scan(RELATIONS_DATA, i, delim);
   end;
   ORDER_=9999;
   call SYMPUT("RealOrder",1);
   if(Last_Line) then do;
      call SYMPUT("MaxObsDataset",_N_);
      call SYMPUT("MaxWords",retTmp);
   end;
run;

data work.DATA_;
   set work.DATA_;
   Counter=0;
   array Rel(50) $60.;
   do i = 1 to &MaxWords;
      if (not missing(Rel[i])) then Counter=Counter+1;
   end;
   if Counter=0 then ORDER_=&RealOrder;
run;

proc sort data= work.DATA_;   by ORDER_ DATANAME; run;

%DO REPEATER=1 %TO &MaxObsDataset;
  %DO J=1 %TO &MaxObsDataset;
   data work.DATA_(drop=TmpName);
      set work.DATA_ END=Last_Line;
      array Rel(50) $60.;
      length TmpName $60.;
      retain TmpName "0";
      if (&J=_N_ and ORDER_=&RealOrder) then TmpName=strip(DATANAME);
      do i = 1 to &MaxWords;
         if (strip(Rel[i]) eq strip(TmpName) ) then call missing(of
Rel[i]);
      end;
   run;
  %END;
```

```
  data _NULL_;
     tmp=&RealOrder+1;
     call SYMPUT("RealOrder",tmp);
  run;

  data work.DATA_;
     set work.DATA_ END=Last_Line;
     array Rel(50) $60.;
     Counter=0;
     do i = 1 to &MaxWords;
        if (not missing(Rel[i]) ) then do; Counter=Counter+1; end;
     end;
     if (Counter=0 and ORDER_=9999)  then ORDER_=&RealOrder;
  run;

  proc sort data= work.DATA_;     by ORDER_ DATANAME;  run;

  data work.DATA_;
     set work.DATA_ END=Last_Line;
     if(Last_Line and ORDER_ < 9999) then do;
        call SYMPUT("REPEATER","&MaxObsDataset");
     end;
  run;
%END;
%mend SortData;

%SortData( );

*******************************************************;
* Create SAS program which check differences ;
* between versions of dataset                ;
*******************************************************;
filename DiffChk "./MakeAutoCheck.sas" ;
%LET DiffCheckFile MakeAutoCheck.sas;

%macro is_Check_exists( );
  %if %sysfunc(fileexist(MakeAutoCheck.sas)) %then %do;
     X "rm ./MakeAutoCheck.sas";
     X "touch MakeAutoCheck.sas";
  %end;
  %else %do;
     X "touch MakeAutoCheck.sas";
  %end;
%mend is_Check_exists;

%is_Check_exists( );

data DIFFCHECK;
   attrib LINES length=$200;
      LINES='%let LibName_ =%scan(&sysparm,1," ");'; output;
      LINES='%let FileName_ =%scan(&sysparm,2," ");'; output;
      LINES='libname FLib_ "&LibName_";' ; output;
      LINES='%LET IS_EQUAL = empty ;' ; output;
      LINES='proc compare base= FLib_.&FileName_ compare=
FLib_.old_&FileName_ outstats=diffstat noprint;' ; output;
```

```
      LINES='run;                ' ; output;
      LINES='data diffstat;      ' ; output;
      LINES='    set diffstat;   ' ; output;
      LINES='    attrib DIFF_ length=8.;    ' ; output;
      LINES='    if (_TYPE_ = "NDIF") then do;    ' ; output;
      LINES='         if (_BASE_=0 and _COMP_=0) then DIFF_=0;' ;
output;
      LINES='         else DIFF_=1;    ' ; output;
      LINES='         output; ' ; output;
      LINES='    end;  ' ; output;
      LINES='run; ' ; output;
      LINES='%macro check_differences( );   ' ; output;
      LINES='    data diffstat;   ' ; output;
      LINES='       set diffstat;   ' ; output;
      LINES='       if (_TYPE_ = "NDIF") then do; ' ; output;
      LINES='            if (_BASE_=0 and _COMP_=0) then DIFF_=0; ' ;
output;
      LINES='            else DIFF_=1; ' ; output;
      LINES='            output; ' ; output;
      LINES='       end; ' ; output;
      LINES='    run;  ' ; output;
      LINES='    ' ; output;
      LINES='    data diffstat; ' ; output;
      LINES='       set diffstat END=Last_Line;     ' ; output;
      LINES='       retain FDIFF 0;    ' ; output;
      LINES='       if _N_=1 then FDIFF=0; ' ; output;
      LINES='       FDIFF=FDIFF + DIFF_; ' ; output;
      LINES='       if(Last_Line) then do;   ' ; output;
      LINES='          if(FDIFF ge 1) then do; call
SYMPUT("IS_EQUAL","N"); end;' ; output;
      LINES='        else if(FDIFF lt 1) then do; call
SYMPUT("IS_EQUAL","Y"); end;' ; output;
      LINES='       end; ' ; output;
      LINES='    run;  ' ; output;
      LINES='    %IF ("&IS_EQUAL"= "Y") %THEN %DO; ' ; output;
      LINES='       X "rm &LibName_./&FileName_..sas7bdat";' ; output;
      LINES='       X "mv &LibName_./old_&FileName_..sas7bdat
&LibName ./&FileName ..sas7bdat";' ; output;
      LINES='    %END; ' ; output;
      LINES='    %ELSE %IF ("&IS_EQUAL"= "N") %THEN %DO; ' ; output;
      LINES='       X "rm &LibName_./old_&FileName_..sas7bdat"; ' ;
output;
      LINES='    %END; ' ; output;
      LINES='%mend check_differences; ' ; output;
      LINES='%check_differences( );   ' ; output;
run;

data _null_;
   set DIFFCHECK;
   file DiffChk;
   put LINES;
run;

*****************************************************************;
*        Creating Makefile: add commands line-by-line           *;
*****************************************************************;
```

```
%is_Makefile_exists( ); /*create OR remove+create*/
*** Libnames definition;
data LIBPART;
   set work.LIB_  END=Last_Line ;
   attrib LINES length=$200;
   if ( N_ = 1) then do;
      LINES=".IGNORE:  # option = ignore errors during run of
makefile"||&EndLine;
      output;
      LINES= "CHECKLOG = .  # option = way to put checklog of
makefile" ||&EndLine;
      output;
      LINES="# <BEGIN> LIBRARIES  ==========="||&EndLine;      output;
      LINES=strip(LIBNAME_)||" = "||strip(PATH_);      output;
   end;
   else if (Last_Line) then do;
      LINES=strip(LIBNAME_)||" = "||strip(PATH_);   output;
      LINES=&EndLine||"# <END> LIBRARIES ========"||&EndLine; output;
   end;
   else do;
      LINES=strip(LIBNAME_)||" = "||strip(PATH_); output;
   end;
run;

data _null_;
   set LIBPART;
   file MFile;
   put LINES;
run;

*** OVERALL definition: RAW+Other, SDTMs, ADAMs, Tables, Listings,
Graphs;
data Block_(keep=FlagN);
   set work.TLG_ (drop=Libname_)
       work.DATA_(drop=Libname_);
run;

proc sort data=Block_ nodupkeys;
   by FlagN;
run;

data work.Block_(keep = COLLECT);
   set Block_ END=Last_Line;
   retain COLLECT "          ";
   COLLECT=strip(COLLECT)||strip(put(FlagN,best.));
   if (Last_Line) then output;
run;

data LIBOVERALL(drop=COLLECT);
   attrib LINES length=$200;
   set work.Block ;
      LINES="# <BEGIN> OVERALL  ==================="||&EndLine;
output;
      LINES="system: "||&tab||"\";    output;
   *Flag=1: PK data or smth else. Usually it it has relations with
other data.;
```

```
      if(index(COLLECT,"1")>0) then do; LINES="   "||"RAWs_and_Other
"||&tab||"\";  output; end;
   *Flag=2: SDTM data. Usually based on Flag 1 or has no relations;
      if(index(COLLECT,"2")>0) then do;LINES="   "||"SDTMs
"||&tab||"\";  output; end;
   *Flag=3: ADAMs. Usually based on SDTMs and Flag 1;
      if(index(COLLECT,"3")>0) then do;LINES="   "||"ADAMs
"||&tab||"\"; output;end;
   *Flag=4: Tables. Usually begin from t_;
      if(index(COLLECT,"4")>0) then do;LINES="   "||"Tables
"||&tab||"\"; output;end;
   *Flag=5: Listings. Usually begin from l_;
      if(index(COLLECT,"5")>0) then do;LINES="   "||"Listings
"||&tab||"\";  output;end;
   *Flag=6: Graphs. Usually begin from g_;
      if(index(COLLECT,"6")>0) then do;LINES="   "||"Graphs
"||&tab||"\";  output; end;
      LINES=&EndLine||"# <END> OVERALL  ========="||&EndLine;
output;
run;

data _null_;
   set LIBOVERALL;
   file MFile mod;
   put LINES;
run;

%macro List_and_Run_datasets(flag_= , Label= , Label_Comment= );
   *** RAW+Other / SDTM / ADAM definition;
   data LIBDATA_LIST;
      set work.DATA_(where=(FlagN=&flag_))  END=Last_Line ;
      attrib LINES length=$200;
      if (_N_= 1 and Last_Line) then do;
         LINES="# <BEGIN> &Label_Comment=============="||&EndLine;
output;
         LINES="&Label: "||&tab||"\";     output;
           if  (not missing(DATASET_)) then
LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat"||&tab||"
\";
           else  LINES=" ";
         output;
         LINES=&EndLine||"# <END> &Label_Comment =="||&EndLine;output;
      end;
      else if (_N_= 1) then do;
         LINES="# <BEGIN> &Label_Comment ========="||&EndLine; output;
         LINES="&Label: "||&tab||"\";      output;
           if  (not missing(DATASET_)) then
LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat"||&tab||"
\";
           else  LINES=" ";
           output;
      end;
      else if (Last_Line) then do;
           if  (not missing(DATASET_)) then
LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat"||&tab||"
\";
```

```
            else  LINES=" ";
            output;
         LINES=&EndLine||"# <END> &Label_Comment==="||&EndLine;output;
      end;
      else do;
           if  (not missing(DATASET_)) then
LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat"||&tab||"
\";
            else  LINES=" ";
         output;
      end;
   run;

   data _null_;
      set LIBDATA_LIST;
      file MFile mod;
      put LINES;
   run;

   *** RAW+Other / SDTM / ADAM  run;
   data LIBDATA_RUN;
      set work.DATA_(where=(FlagN=&flag_ ))  END=Last_Line ;
      attrib LINES length=$200;
      if ( N = 1 and Last_Line) then do;
         LINES="# <BEGIN> RUN: &Label_Comment====="||&EndLine;  output;
         LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat
:"||&tab||"\";     output;
           if (not missing(RELATIONS_DATA)) then
LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
","sas7bdat%str(   $)"))||".sas7bdat \";
           else     LINES="\"; output;
         LINES=strip(RELATIONS_PROG); output;
         if (not missing(COMMAND_1)) then do;
           LINES=&tab||"mv "||
strip(PATH_)||"/"||strip(DATASET_)||".sas7bdat "||
strip(PATH_)||"/"||"old_"||strip(DATASET_)||".sas7bdat";  output;
           LINES=&tab||strip(COMMAND_1); output;
           LINES=&tab||"sas -sysparm '"|| strip(PATH_)||"
"||strip(DATASET_)||"' &DiffCheckFile";  output;
         end;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
         LINES=&EndLine||"# <END> RUN: &Label_Comment==="||&EndLine;
output;
      end;
      else if (_N_ = 1) then do;
         LINES="# <BEGIN> RUN: &Label_Comment====="||&EndLine; output;
         LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat
:"||&tab||"\";   output;
           if (not missing(RELATIONS_DATA)) then
LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
","sas7bdat%str(   $)"))||".sas7bdat \";
           else       LINES="\";      output;
         LINES=strip(RELATIONS_PROG); output;
```

```
         if (not missing(COMMAND_1)) then do;
           LINES=&tab||"mv "||
strip(PATH_)||"/"||strip(DATASET_)||".sas7bdat "||
strip(PATH_)||"/"||"old_"||strip(DATASET_)||".sas7bdat";  output;
           LINES=&tab||strip(COMMAND_1); output;
           LINES=&tab||"sas -sysparm '"|| strip(PATH_)||"
"||strip(DATASET_)||"' &DiffCheckFile";  output;
         end;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
      end;
      else if (Last_Line) then do;
         LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat
:"||&tab||"\";   output;
           if (not missing(RELATIONS_DATA)) then
LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
",".sas7bdat%str(   $)"))||".sas7bdat \";
           else         LINES="\";    output;
         LINES=strip(RELATIONS_PROG); output;
         if (not missing(COMMAND_1)) then do;
           LINES=&tab||"mv "||
strip(PATH_)||"/"||strip(DATASET_)||".sas7bdat "||
strip(PATH_)||"/"||"old_"||strip(DATASET_)||".sas7bdat";  output;
           LINES=&tab||strip(COMMAND_1); output;
           LINES=&tab||"sas -sysparm '"|| strip(PATH_)||"
"||strip(DATASET_)||"' &DiffCheckFile";  output;
         end;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
         LINES=&EndLine||"# <END> RUN: &Label_Comment===="||&EndLine;
output;
      end;
      else do;
         LINES="$"||strip(LIBNAME_)||"/"||strip(DATASET_)||".sas7bdat
:"||&tab||"\"||&EndLine;  output;
           if (not missing(RELATIONS_DATA)) then
LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
",".sas7bdat%str(   $)"))||".sas7bdat \";
           else      LINES="\";     output;
         LINES=strip(RELATIONS_PROG);   output;
         if (not missing(COMMAND_1)) then do;
           LINES=&tab||"mv "||
strip(PATH_)||"/"||strip(DATASET_)||".sas7bdat "||
strip(PATH_)||"/"||"old_"||strip(DATASET_)||".sas7bdat";  output;
           LINES=&tab||strip(COMMAND_1); output;
           LINES=&tab||"sas -sysparm '"|| strip(PATH_)||"
"||strip(DATASET_)||"' &DiffCheckFile";  output;
         end;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
```

```
      end;
   run;

   data _null_;
      set LIBDATA_RUN;
      file MFile mod;
      put LINES;
   run;

   %IF %sysfunc(exist(LIBDATA))%THEN %DO;
      proc datasets library=work;
         delete LIBDATA_LIST LIBDATA_RUN;
      run;
   %END;
%mend List_and_Run_datasets;

%List_and_Run_datasets(flag_=1 , Label=RAWs_and_Other ,
Label_Comment=RAW+Other datasets );
%List_and_Run_datasets(flag_=2 , Label=SDTMs , Label_Comment=SDTM
datasets );
%List_and_Run_datasets(flag_=3 , Label=ADAMs , Label_Comment=ADAM
datasets );


%macro List_and_Run_TLGs(flag_= , Label= , Label_Comment= );
   *** RAW+Other / SDTM / ADAM definition;
   data TLG_LIST;
      set work.TLG_(where=(FlagN=&flag_))  END=Last_Line ;
      attrib LINES length=$200;
      if (_N_= 1 and Last_Line) then do;
         LINES="# <BEGIN>&Label_Comment============"||&EndLine;
         output;
         LINES="&Label: "||&tab||"\";
         output;
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
"||&tab||"\";
         output;
         LINES=&EndLine||"# <END> &Label_Comment ======"||&EndLine;
         output;
      end;
      else if (_N_= 1) then do;
         LINES="# <BEGIN> &Label_Comment====="||&EndLine; output;
         LINES="&Label: "||&tab||"\";
         output;
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
"||&tab||"\";
         output;
      end;
      else if (Last_Line) then do;
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
"||&tab||"\";
         output;
         LINES=&EndLine||"# <END> &Label_Comment==="||&EndLine;
output;
      end;
      else do;
```

```
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
"||&tab||"\";
         output;
      end;
   run;

   data _null_;
      set TLG_LIST;
      file MFile mod;
      put LINES;
   run;

   *** RAW+Other / SDTM / ADAM run ;
   data TLG_RUN;
      set work.TLG (where=(FlagN=&flag_ ))  END=Last_Line ;
      attrib LINES length=$200;
      if (_N_ = 1 and Last_Line) then do;
         LINES="# <BEGIN> RUN: &Label_Comment="||&EndLine; output;
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
:"||&tab||"\";
         output;

LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
","_sas7bdat%str(   $)"))||"_sas7bdat \";    output;
         LINES=strip(RELATIONS_PROG);     output;
         LINES=&tab||strip(COMMAND_1);     output;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
         LINES=&EndLine||"# <END> RUN:
&Label_Comment=======:"||&EndLine; output;
      end;
      else if (_N_ = 1) then do;
         LINES="# <BEGIN> RUN: &Label_Comment========="||&EndLine;
output;
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
:"||&tab||"\";
         output;

LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
","_sas7bdat%str(   $)"))||"_sas7bdat \";    output;
         LINES=strip(RELATIONS_PROG);     output;
         LINES=&tab||strip(COMMAND_1);      output;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
      end;
      else if (Last_Line) then do;
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
:"||&tab||"\";
         output;

LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
","_sas7bdat%str(   $)"))||"_sas7bdat \";    output;
```

```
         LINES=strip(RELATIONS_PROG);   output;
         LINES=&tab||strip(COMMAND_1);  output;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
         LINES=&EndLine||"# <END> RUN:&Label_Comment="||&EndLine;
output;
      end;
      else do;
         LINES="$"||strip(LIBNAME_)||"/"||strip(OUTPUT_)||"
:"||&tab||"\";
         output;

LINES="$"||strip(tranwrd(tranwrd(strip(RELATIONS_DATA),".","/"),"
","_sas7bdat%str(   $)"))||"_sas7bdat \";  output;
         LINES=strip(RELATIONS_PROG);  output;
         LINES=&tab||strip(COMMAND_1); output;
         if(not missing(COMMAND_QC)) then do;
            LINES=&tab||strip(COMMAND_QC)||&EndLine;
         end;
         output;
      end;
   run;

   data _null_;
      set TLG_RUN;
      file MFile mod;
      put LINES;
   run;
%mend List_and_Run_TLGs;

%List_and_Run_TLGs(flag_=4 , Label=Tables   , Label_Comment=TLG:
Tables   );
%List_and_Run_TLGs(flag_=5 , Label=Listings , Label_Comment=TLG:
Listingd );
%List_and_Run_TLGs(flag_=6 , Label=Graphs   , Label_Comment=TLG:
Graphs   );
```