PharmaSUG 2019 - Paper PO-221

Why waiting longer to check log file when SAS program in Execution? Let's Find bugs Early!!

Thamarai Selvan Palanisamy, Quartesian Clinical Research;
Prakash Subramaniam, Quartesian Clinical Research;

ABSTRACT

Prompt error alerts through emails while an error is encountered, will help user to save the run time and restrict the creation of empty datasets from the subsequent codes. It also helps in monitoring the status of the SAS® programs submitted without being in front of the computer. While running the program, if SAS finds any error in the current line, it will skip the current statement and will start executing the next statement and the process will be complete only after the execution of the end statement of the program. There are programs which takes more than a day to complete. In such cases, user has to open the log file in read only mode very frequently to check for errors, warnings and unexpected notes. The user will have to terminate the execution of program manually if any such messages are identified else the user will be notified with the errors in the log file only at the end of the execution.

Our proposal is to run the parallel utility program along with the production program to check the log file of the current SAS program and to notify the user through an email while encountering an error, warning or unexpected notes in the log file. Also the execution can be terminated automatically and the user can be notified if any potential messages are identified.

INTRODUCTION

Most of the SAS users are using either utility program to identify errors, warnings and unexpected notes or getting notified through an email to find the errors only after the full execution of the SAS job. As of now, most of these programs will help a user to check the log file only after the full execution of the SAS job.

The idea of this paper is to check the log file while the program is in the execution mode by running the utility macro in parallel. Using this utility macro any one of the following could be done:

- User may receive the error messages through an email and if required, the user shall terminate the job manually.
- User may receive the error messages through an email and auto terminate option can be chosen in the utility program.
- If email option is not available, User can skip the notification through email and can auto terminate the execution.

THE NEED FOR PARALLEL RUN AND THE POWER OF SYSTASK STATEMENT

Checking the log file through an automated utility macro makes it easier for the programmer to ensure that the program ran as expected. In common these kind of utility macros are used only on the completed tasks. To trace the log file of a running SAS program, we need to execute another SAS utility macro in parallel. Through the utility macro, log file of the main program can be traced out during the execution. By default, SAS does not allow any user to execute two SAS programs in parallel, to do so some special statements need to be called.

In SAS, we have a special statement called the SYSTASK. The SYSTASK statement is used to execute the system specific commands through SAS. SYSTASK runs the commands as asynchronous tasks, which means that these tasks execute independently of all other tasks that are currently running. In our case, SYSTASK statement is used to run the chasethelog.sas in parallel so that the log traceability will start while the execution of main code starts.

IMPORTANCE OF EMAIL COMMUNICATION

Email communication plays a vital role in every industry. SAS enables us to send e-mail by way of a DATA step, SAS procedure, or SCL. The setup and the interfaces depend on the organization and their

server operating system. In practice, email communication in SAS software is used to send successful completion notification, the generated reports as an attachment, and so on. In our case, email option is used to send the auto termination message if errors, warnings or unexpected notes are found in the log during the execution. This kind of alert helps the user to terminate the execution manually if the error/warning/unexpected notes would cause any problem to the final output.

%CHASETHELOG MACRO

The following lines of code should be placed at the beginning of every long running program:

```
%macro chasethelog(emailid=, EndProcess=);
data _null_;
    file 'chasethelog.sas';
    put '%let StartTime = ' "%sysfunc(putn(%sysfunc(datetime()),
datetime16.));";
    put '%let User= '"%upcase(&SYSUSERID);";
    put '%let PgmName='"%scan(%scan(&SYSPROCESSNAME,2,' '),1,'.');";
    put '%let emailid=' "&emailid;";
    put '%let procid='"&SYSJOBID;";
    put '%let EndProcess=' "&EndProcess;";
    put '%include "'chasethelog_main.sas";';
run;
systask command "chasethelog.sas";
%mend chasethelog;
```

This macro creates new macro variables for Execution Start Time, User ID, Main Program Name, Unix Job ID, Email ID and auto termination process. Along with the creation of macro variables, chasethelog_main.sas code (Refer Appendix at the end of the paper) will be called as an include file, whereas chasethelog_main.sas has an algorithm to perform the auto termination and email notification.

FUNCTIONALITY OF %CHASETHELOG MACRO:

As soon as the execution of main program starts, chasethelog macro creates the new program chasethelog.sas and starts executing using the systask statement. This code reads the main program's log file line by line. Initially it will read log file from the first line to the number of lines available at the time of execution and these observations are saved in a dataset. A flag variable is set on the last record to read the log file and identify the number of observation read on the last execution. The dataset will be checked for the errors, warnings and unexpected notes. If an error is found and the auto termination had been set to yes, then this code will kill the job using the current job ID. If the email ID has been set in the macro then auto termination message will be sent as a notification mail. If only warnings or unexpected messages are found then only a notification email will be sent to the user. Termination of the job happens only if error messages are found, even if the auto termination has been set to yes. If auto termination is set to No and email ID has been set in the macro, only notification email will be sent to the user and user can take the action based on his requirement. If macro dose not find any issue, it will read the next set of observation from the log file. From second time onwards the log file will be read from the last record read on the log file during the previous observation to the observation currently available. The last record can be found using the flag variable set during the previous execution. If the main program does not encounter any issue until the execution completes then chasethelog macro will stop executing without any action. But a successful notification will be sent to the user on completion.

In any software, the way of coding, the usage and the method of execution is based on the operating system and environment setup, SAS is not an exception. In this paper the code written for %chasethelog_main works fine on unix environment with server setup. Little tweaking of the code may be required while working on the other environment.

PROGRAM EXECUTION FLOW:

The program will be executed as per the flow given in the Figure 1. The %Chasethelog macro should be placed at the top of the long running program. The %chasethelog_main.sas should be modified as per

requirement and placed in the same folder where the main program exists. When the initial setup is done, then the main program can be executed in the batch submission mode. As soon as the execution starts the %Chasethelog macro creates the new program called chasethelog.sas and this program will be saved in the folder where the long running program exists. The %Chasethelog macro will start the execution as soon as systask statement executes and the code present below %Chasethelog macro will be executed as usual. If any hazardous messages were found, then the action will be taken based on %Chasethelog macro call. Execution ends normal if no hazardous messages were found.

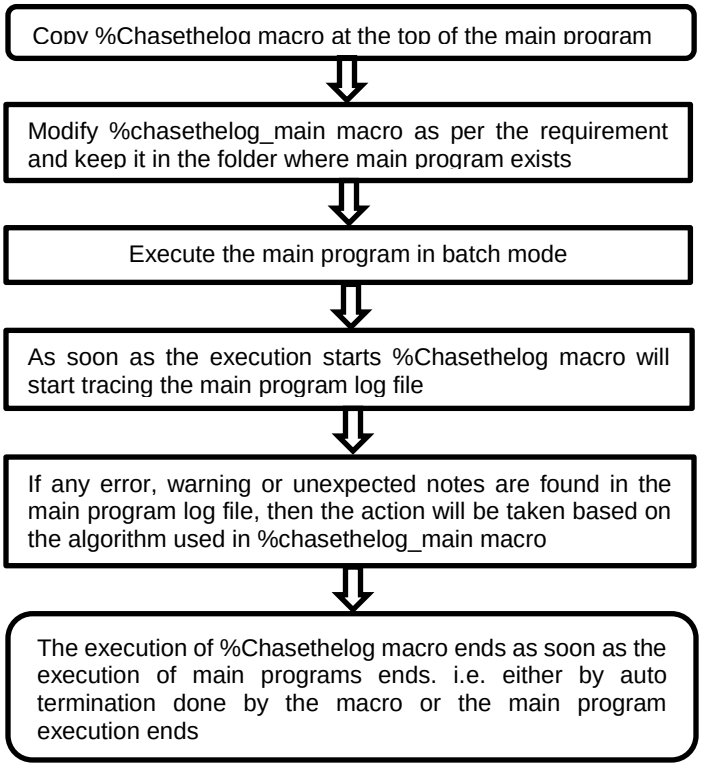

Figure 1. Execution flow diagram

USAGE OF CHASETHE LOG MACRO AND ITS SAMPLE CALLS:

Chasethelog macro can be called in three ways based on the requirement. Each way of calling the macro has different functionality, which are as follows:

1) To receive only error notification through an email:

```
%chasethelog(emailid=aaa@bbb.com, EndProcess=N);
```

2) To terminate the execution of SAS program as soon as the error message has been encountered in log file and receive the notification through an email:

```
%chasethelog(emailid=aaa@bbb.com, EndProcess=Y);
```

3) To auto terminate the SAS execution as the error message has been encountered in log file:

```
%chasethelog(emailid=, EndProcess=Y);
```

In both the first and second case, we receive other notification through an email.

BENEFITS:

- User is alerted of an error through an email even when one is not in front of the computer.
- The user can take immediate action once he receives the email with the error/warning message.
- SAS Program continues running even after it encounters "Unexpected Notes" and "Warnings" thus occupying server space by creating empty datasets, this utility can help identify the errors and thereby user can terminate the program automatically instead of letting the program run completely thereby increasing the server space utilization

LIMITATIONS

Every application or utility have some limitation and ours is not an exception. Our utility macro is applicable only while running the SAS program in BATCH mode. If either email id or auto termination of the process is not requested, then this utility does not serve any purpose

ACKNOWLEDGMENTS

Our Special thanks to Quartesian Clinical Research for the entire support on submitting this paper.

TRADEMARK

CONTACT INFORMATION

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

Name: Thamarai Selvan Palanisamy,
Phone: +91-9176624274
+91-9578579383
Email: mail2info.selvan@gmail.com
thamaraiselvan.p@quartesian.com

Name: Prakash Subramaniam,
Phone: +91-9787488158
+91-9986430203
Email: prakashkanur@yahoo.co.in
Prakash.subramaniam@quartesian.com

APPENDIX:

```
%macro chasethelog_main;
      %let JobStatus=1;
      %let Toread=1;
      %let EndJob=0;
      %let TraceWarn=0;
      %let TraceError=0;
      %let Iteration=0;
      %do %until(&JobStatus=0);
            %let SendMail=0;
            %let Track=0;
            %let WarnFlag=0;
            %let ErrorFlag=0;

            filenamerunlog  pipe "cat &PgmName..log";

            data a;
                  infilerunloglrecl=32000 length=linelength end=eof %if
&Iteration>0 %then firstobs=&toread;;
                  input logline $varying700. linelength ;
                  if index(logline,'ERROR:') or
upcase(substr(left(logline),1,8))='WARNING:' or
index(logline,'uninitialized') > 0 or
                        index(logline,'repeats of BY values') > 0 or
index(logline,'W.D format') > 0 or
                        index(logline,'Invalid') > 0 or
index(logline,'Mathematical operations could not') > 0 or
upcase(substr(left(logline),1,10))='USER ERROR' or
                        upcase(substr(left(logline),1,12))='USER WARNING' or
index(logline,'outside the axis range') > 0 or
                        index(logline,'values have been converted') or
index(logline,'Missing') >0
                        then do;
                              callsymputx('SendMail', '1');
                              if index(logline,'ERROR:')=0 then call
symputx('WarnFlag',1);
                              output;
                  end;
                  if index(logline,'ERROR:') then do;
```

```
                        callsymputx('ErrorFlag',1);
                        if "&EndProcess"="Y" and &EndJob=0 then call
symputx('EndJob',1);
                  end;
                  callsymputx ('nrecord',_n_);
            run;
            %let toread=%eval(&nrecord+&toread);

            %let TraceWarn=%eval(&TraceWarn+&WarnFlag);
            %let TraceError=%eval(&TraceError+&ErrorFlag);



            %if &SendMail=1 and &emailid^= %then %do;
                  filenamesendm email to="&emailid"  Subject=%if &ErrorFlag=1
%then "Error Occured in the &PgmName..sas Program"

                        %else "Suspicious lines Occured in the
&PgmName..sas Program";;
data a;
                        set a;
                        output;
                        logline='';
                        output;
                  run;


data _null_;
                        set a;
                        filesendm;
                        put logline;
                  run;
            %end;


            %if &EndJob %then %do;
                  x "kill &procid";
                  %if &emailid^= %then filename Stats email to="&emailid"
Subject="%upcase(&PgmName.).sas Program is done with ERROR"; ;
                  data _null_;
                        file Stats;
                  run;
```

```
            %let JobStatus=0;
            %gotoEndLogCheck;
      %end;

      filenameprc pipe "ps u";

      data b;
            infileprc length=len;
            input process $varying900. len;
            if index(process,"&PgmName") and index(process,"&procid");
            callsymputx('track',scan(process,2,' '));
      run;

      %if &track^=&procid %then %do;
            %let JobStatus=0;
            %if &emailid^= %then %do;
                  filename Status email to="&emailid" %if &TraceError>0
%then Subject="%upcase(&PgmName.).sas Program is done with ERROR";

                    %else %if &TraceWarn>0 %then
Subject="%upcase(&PgmName.).sas Program is done with Warning";

                        %else Subject="%upcase(&PgmName.).sas Program
is with Success";;
                  data _null_;
                        file Status;
                  run;
            %end;
      %end;

      %EndLogCheck:
    %let Iteration=%eval(&Iteration+1);
      filename _all_ clear;
  %end;
%mend chasethelog_main;
```