PharmaSUG China 2022 - Paper AT- 133

Exploration on the Knowledge of Linux to Help Daily Work

Jinzhi Zhou, Jie Liu, BeiGene (Beijing) Co., Ltd.

ABSTRACT

Linux is a very popular operating system in building company server. More and more companies in pharmaceutical industry choose Linux as server OS to install their SAS. Therefore, certain knowledge on Linux is very necessary for us to work more efficiently.

This paper intends to explore how Linux can help us in our daily work and how problems can be solved via Linux through 2 examples. The Linux OS is provided by Red Hat and SAS 9.4 version is used in this paper.

Sometimes we need to re-run our programs periodically for data cleaning or monitoring, which may happen very frequently. It might be time-consuming and a little boring. Is there anything Linux can do to help us get rid of it? The answer is yes. Cron provided by Linux can schedule a task periodically. A simple cron command makes re-running not being a burden anymore.

There will be unexpected issues when we submit a batch of programs to server, like short of memory and block on file server, especially when hundreds of users are using the server. For the batch run programs like CSR programs, we need to re-run all of them even there is only 1 error. It can take a very long time to finish one batch running. Can Linux solve this problem better? The answer is yes, too. We have developed a shell script for batch run which can fetch the error of each program. If a server error is identified after a program is run, the script will not initiate the following program and the current program will be re-run until there is no server error.

INTRODUCTION

Linux is an operating system like Unix, Windows, and Mac OS, created in 1991 by Linus Torvalds. Since more and more companies in pharmaceutical industry have chosen Linux as server OS to install their SAS, it's very necessary for programmers to learn more about Linux and know how to use Linux commands and Linux shell script to help our daily work.

Shell is a program written in C that serves as a bridge for users to use Linux. Shell is both a command language and a programming language. Some companies in pharmaceutical industry have developed Linux shell script to automatically compare results of statistical analyses, which can greatly reduce the cycle time for production to correct any programming errors.

In addition to the shell script, there are many Linux commands that also can be used to improve the efficiency. The cron command-line utility is a job scheduler on Linux operating systems. Users who set up and maintain software environments use cron to schedule jobs (commands or shell scripts), also known as cron jobs to run programs periodically at fixed times, dates, or intervals. Cron is most suitable for scheduling repetitive tasks. In general, we need to re-run some programs periodically in clinical trials and a cron command makes the repetitive work more efficient and not so boring.

To make our daily work more efficient under Linux OS, we try to explore the application of Linux knowledge in daily work and will give two examples to demonstrate how the Linux command and Linux shell script can be applied.

USING CRON TO SCHEDULE A TASK PERIODICALLY

SYNTAX

The actions of cron are driven by a crontab (cron table) file, namely a configuration file that specifies shell commands to run periodically on a given schedule.

Here is the syntax of the contab:

```
crontab [ -u user] {-l | -r | -e}
or
crontab [ -u user] file
```

Table 1. Option of the crontab

Option	Description
-u <user>	Define user
-e	Edit user's crontab
-l	List user's crontab
-r	Delete user's crontab

In general, a crontab file is as below:

```
SHELL=/bin/bash
59 23 * * * printf "" > /home/log/error_log
30 9 * * 1 sas /home/scripts/pgm.sas
0 9 1 * * /home/scripts/mdr.sh
```

Each line of a crontab file represents a job, as below:

```
# ┌──────────── minute (0 - 59)
# │ ┌──────────── hour (0 - 23)
# │ │ ┌──────────── day of the month (1 - 31)
# │ │ │ ┌──────────── month (1 - 12)
# │ │ │ │ ┌──────────── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                   7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <command to execute>
```

The syntax of each line expects a cron expression made of five fields which represent the time to execute the command, followed by a command to execute. The five fields in cron expression can with the following possible values:

1. Minute. The minute of the hour the command will run on, ranging from 0-59.
2. Hour. The hour the command will run at, ranging from 0-23 in the 24-hour notation.
3. Day of the month. The day of the month the user wants the command to run on, ranging from 1-31.
4. Month. The month that the user wants the command to run in, ranging from 1-12, thus representing January-December.
5. Day of the week. The day of the week for a command to run on, ranging from 0-6, representing Sunday-Saturday. In some systems, the value 7 represents Sunday.

EXAMPLE

Below are 4 simple examples demonstrating how to use cron commands to schedule task.

Example 1

```
59 23 * * * printf "" > /home/log/error_log
```

This example clears the error log at 23:59 (11:59 PM) every day.

Example 2

```
*/5 * * * * echo hello world
```

The above command would output "hello world" to the command line every 5 minutes.

Example 3

```
30 9 * * 1 sas /home/scripts/pgm.sas
```

This command above runs a SAS program called pgm.sas at 9:30 AM every Monday.

Example 4

```
30 9 1 * * /home/scripts/mdr.sh
```

This example runs a shell program called mdr.sh at 9:30 AM every first day of this month.

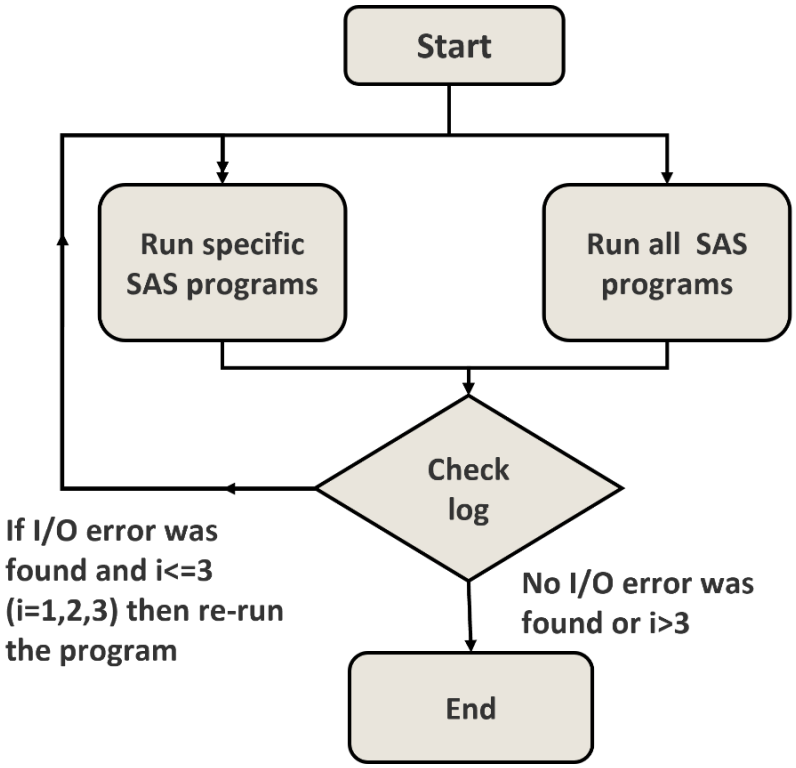
SUMMARY

Based on the examples above, we can find that the syntax of cron command is easy for you to get started even as a freshman in Linux, which also offers several fields to set up time that will make task schedule more flexible.

A SHELL SCRIPT FOR BATCH RUN PROGRAMS AND ERROR FETCHING OF EACH PROGRAM

FLOWCHART

Figure 1. Flowchart of batch run SAS programs and error fetching of each program.



LOGIC

This shell script mainly includes two parts, one for batch-run SAS programs, the other for issue log checking.

Step 1 - Batch run the SAS programs and get the '.log' files, and an option was set to allow the user to include or exclude some SAS programs when not all the code are needed to be run.

Step 2 - Check the issue log by the 'grep' command and concatenate the results into a single text file. When a system I/O error was found, the SAS program will be re-submitted up to three times, and the interval time between each submission is 30 seconds. Finally, it will create an output file which consists of a summary section and a details section.

This shell script was developed in both 'u8' (`pgm_run_u8.sh`) and 'en' (`pgm_run_en.sh`) versions to enable different encodings.

SAMPLE SCRIPT AND OUTPUT

This shell script should be called in other shell script such as demo1.sh.

1. Sample for running all programs by u8 version.

```
alias run_u8="source ./pgm_run_u8.sh"
alias run_en="source ./pgm_run_en.sh"
run_u8 demo1 all
```

2. Sample for running specific programs by u8 version.

```
alias run_u8="source ./pgm_run_u8.sh"
alias run_en="source ./pgm_run_en.sh"
run_u8 demo1 pgm1.sas pgm2.sas pgm3.sas
```

Table 2. Parameter of the shell script

Parameter	Core	Example	Usage
parameter 1	Require	demo1	name of the output log file
parameter 2	Require	'all' or 'pgm1.sas'	if all SAS programs need to be run then the parameter2 should be "all"
parameter 3	Permissible		
parameter N	Permissible		

Note: for Core, 'Require' means this parameter can't be null and 'Permissible' can be null.

3. Sample for summary log file

There are two sections in the final output log file, one is the summary section which will show the information that how many programs were submitted and the number of errors/warnings/notes found in this batch or each program, and the other will show the details of issue log in each program.

Summary

```
****************************************************************
        Summary information of batch demo1
****************************************************************
          3 programs run in demo1
          No ERROR found in demo1
        2 WARNINGS found in demo1
        2 NOTES found in demo1
****************************************************************
        2 WARNINGS found in pgm1.log
        2 NOTES found in pgm3.log
****************************************************************
```

Details

```
----------SEARCHING pgm1.log ----------
WARNING: Apparent symbolic reference OUTNAME not resolved.
WARNING: Apparent symbolic reference OUTNAME not resolved.

----------SEARCHING pgm2.log ----------

No problems found in pgm2.log

----------SEARCHING pgm3.log ----------
NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column).
NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column).
```

SUMMARY

This script provides an efficient way to batch run the SAS code and check issue log. It greatly reduces the time for programmers to check the issue log, which is especially important when they need to re-run the code frequently.

CONCLUSION

Based on the examples in this paper, we could find that the Linux commands and shell script could make some repetitive work not that boring and more efficient, and we believe Linux could help us more in our daily work, so let's further explore Linux in automating our work and improving the efficiency.

REFERENCES

Chen.W.(2018). A Linux Shell Script to Automatically Compare Results of Statistical Analyses. *PharmaSUG Proceedings.* Seattle, WA.

"What is Linux?". Opensource.com. Retrieved May 12, 2020.

https://opensource.com/resources/linux.

"crontab(5): tables for driving cron - Linux man page". Linux.die.net. Retrieved 2013-11-06. https://linux.die.net/man/5/crontab

"Cron Job: a Comprehensive Guide for Beginners 2020". May 24, 2019. https://www.hostinger.com/tutorials/cron-job

ACKNOWLEDGMENTS

Thanks to our managers (Leo Li, Cindy Song and Tony Guo) for their great support on the conference and other colleagues for their comments on this shell script.

CONTACT INFORMATION

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

Jinzhi Zhou
BeiGene (Beijing) Co., Ltd.
jinzhi.zhou@beigene.com

Jie Liu
BeiGene (Beijing) Co., Ltd.
jie1.liu@beigene.com