The Abdus Salam
International Centre for Theoretical Physics

United Nations
Educational, Scientific and
Cultural Organization

IAEA
International Atomic Energy Agency

2057-9

First Workshop on Open Source and Internet Technology for Scientific Environment: with case studies from Environmental Monitoring

7 - 25 September 2009

Shell Programming

Paul Bartholdi

Observatoire de Geneve
Chemin des Maillettes 51
CH-1290 Sauverny
Switzerland

Exercises on Shell Programming

First Workshop on Open Source Software

September 9, 2009

1 Introduction

1.1 Basic information

The goal of this exercise is to get familiar with writing shell scripts and at the same time get something useful. Running solutions are available at the end as an example. Don't copy them without understanding what every line does, and why. Test every thing line by line, try some variants.

Remember: `man <command>` gives you the informations concerning this command, including all options. So it will be up to you to find the necessary informations.

The basic commands we are going to use are:

`cp` make a copy of a file;

`cat` list the content of a file;

`cut` extract a field from a string

`date` find the current date/time in some format;

`echo` write a string

`exit` terminate a script, passing a status number (0 means OK);

`mkdir` create a new directory;

`tr` replace a character by an other one, replace consecutive identical characters by a single copy, or erase a given character;

`uptime` give various information concerning the host, number of users, load on the machine etc.

`‘commands‘` replace the string by the result of executing the commands (try: `‘echo date‘`);

Before writing your scripts, create in your home a directory called `bin` if it does not exist yet. Add this directory to your `PATH` variable in `.bashrc` or `.profile` , and either logout/login or execute `source .profile` or `source .bashrc`.

1.2 Intelligent backup

Using your preferred editor, write a single line script that copy the file given as parameter by adding the current date at the end. Call this file `bck` . Make it executable (`chmod a+x`). Don't forget to specify the "language" you are using (`sh` or `bash`) as a pseudo comment in the first line of your script.

The command:

```
bck MyFile
```

should create the copy called `MyFile_20090909@105612` .

Hints: `date` can format the date really as you want. For example: `date +%Y%m%d@%H%M%S` will produce `20090909@100603`, nice as a suffix for your backuped files. Use backquotes to add this string at the end of the file name.

Put `bck` in your `bin` directory, and verify that you can use it effectively (make a backup of bck itself!).

1.3 Collecting Data concerning your machine

We want to collect automatically every 5 minutes information concerning your host, as the current time, number of users, load and temperature on the cpu. Data should be added to a "unix flat file" having the format (or anything similar. . .):

```
DATE     TIME   LTIME  U L1   L5   L15  TEMP
20090909 092349 206629 3 0.05 0.17 0.09 41
20090909 092408 206648 3 0.04 0.16 0.09 41
```

All time informations are provided by `date`.

Number of users and load by `uptime`.

Eventually, the cpu temperature can be read in the (pseudo) file:

`/proc/acpi/thermal_zone/ATF0/temperature`.

You will have to use heavily the commands `tr` and `cut` to extract the various interesting fields.

`LTIME` is the unix time in seconds from September 7th at 0:00. It is obtained by subtracting the Unix time on September 7th at 0:00 to the current time, all expressed in seconds.

Develop you script one command line at a time, verifying that you get what you want by echoing every results. You can even start by executing the command in interactive mode, to be sure that you understand the options correctly, then pass the line into the editor. These `echo` commands can be commented when every thing is alright.

Your script will contain:

1. Test if the result file (`cpu.info`) exists. If not create it by echoing the head line into it.
2. get the date
3. get the time
4. get the time in seconds from September 7th at 0:00
5. get the number of users
6. get the load for the last minute, last 5 minutes and last 15 minutes
7. get the temperature
8. write every thing at the end of `cpu.info`

Make your script executable, and test it line by line, including the creation of `cpu.info` (erase it to see that it is recreated etc.)

1.4 Running the script every 5 minutes

The system program `cron` will do the job.

First define the editor you want to use into the variable `EDITOR`. For instence, type

```
export EDITOR emacs
```

You can also set it in your `.profile` or `.bashrc` to have it set automatically at login. It is used by `crontab` to fix the parameters of `cron`.

Then run `crontab -e` to edit the list of regular programs, and add a single line in the form:

```
2,7,12,17,22,27,32,37,42,47,52,57 * * * * /home/pb/bin/ex.sh
```

The last field is the script/program (full path) that must be executed regularly (replace my username with yours!). The first field gives the minutes to start the script, the second the hours, third the days of the month, the fourth the month and the last the days of the week. An "$*$" indicates every instances. In our case, every month, every day, every hour, but only on minutes 2, 7, 12, etc.

`crontab -l` will print the list of regular programs controled by `cron` .

After some time verify that new lines have been added correctly.

1.5 Possible Solutions

```
#!/bin/sh
# Make a copy of a file, adding the current time at the end of the file name

if [ ! -f $1 ] ; then
  echo "*** Which file should be backuped?  Do it again but right!"
  exit 1
fi

cp $1 $1_‘date +%y%m%d@%H%M%S‘

exit 0

#!/bin/sh
# Collect information as date/time, nb users, lcpu load and  temperature
# into a flat file (/home/pb/cpu.info).
# If the file is not found it is created first.
# The LTIME is counted in seconds from September 7th at 0:00

# ICTP Open Source Workshop, Trieste - September 9th, Paul Bartholdi

if [ ! -f /home/pb/cpu.info ] ; then
  echo "DATE     TIME   LTIME  U L1   L5   L15  TEMP" > /home/pb/cpu.info
fi

DATE=‘date +%Y%m%d‘
#echo $DATE

TIME=‘date +%H%M%S‘
#echo $TIME

LTIME=$((‘date +%s‘-‘date -d "2009-09-07" +%s‘))
#echo $LTIME

NBUSER=‘uptime | tr -s " " | cut -d"," -f2 | cut -d" " -f2‘
#echo $NBUSER

LOAD1=‘uptime  | tr -s " " | cut -d"," -f3 | cut -d" " -f4‘
LOAD5=‘uptime  | tr -s " " | cut -d"," -f4 | cut -d" " -f2‘
LOAD15=‘uptime | tr -s " " | cut -d"," -f5 | cut -d" " -f2‘
#echo $LOAD1 $LOAD5 $LOAD15

TEMPERATURE=‘cat /proc/acpi/thermal_zone/ATF0/temperature \
               | tr -s " " \
       | cut -d" " -f2‘
#echo $TEMPERATURE

echo "$DATE $TIME $LTIME $NBUSER $LOAD1 $LOAD5 $LOAD15 $TEMPERATURE" \
        >> /home/pb/cpu.info

exit 0
```