Exercise 22: Writing Scripts - Part 3

I. Prepare the environment

II. Looping with for, while and define your own function

1. Login to the CentOS server with student
1. Write a shell script to report the login capability of the first 10 users in the /etc/passwd file. The output have the format as the following

   There are # users can login to the system, they are:
   <user1>, <user2>, <user3>, … ← list all the user that can login to the system (default shell is /bin/bash)

   There are # users cannot login to the system, they are:
   <user1>, <user2>, <user3>, … ← list all the user that can not login to the system (default shell is not /bin/bash)

2. Write a script to use while loop to read the user name from a file named usr.lst. With each user read from usr.lst, display the appropriate information from /etc/passwd.
   The content of usr.lst as below
   root
   student
   student1

3. Re-write the above script but define a function to display the user information as follows:
   ```
   ==========================
   USERNAME: <user name>
   Home directory: <home directory>
   Default shell: <default shell>
   ==========================
   ```

   Use this function to display the information of users in the usr.lst.

III. Schedule script with at and cron table

4. Create a simple script to display a message to the screen as follows:
   "I'm running…"
   Using at command to schedule this script to run in the next 1, 5 and 10 minutes
5. After the first job completed, using atq and atrm to list and remove all the jobs one by one
6. Add a job to cron table to run a script at every minutes of the Thursday of every weeks. You could use the script in step 5.

Exercise Instructions

I. **Prepare the environment**
II. **Looping with for, while and define your all function**

1. Login to the CentOS server with student
2. Write a shell script to report the login capability of the first 10 users in the /etc/passwd file. The output have the format as the following

   There are # users can login to the system, they are:
   <user1>, <user2>, <user3>, … ← list all the user that can login to the system (default shell is /bin/bash)

   There are # users cannot login to the system, they are:
   <user1>, <user2>, <user3>, … ← list all the user that can not login to the system (default shell is not /bin/bash)

```
#!/bin/bash

nusr=0

nologin=0

nusr_name=""

nologin_name=""

for i in $(head -10 /etc/passwd)

do

    if [ $(echo $i|cut -d: -f 7) = "/bin/bash" ]

    then
            nusr=$[$nusr + 1]
            nusr_name="$(echo $i|cut -d: -f 1), $nusr_name"
    else
            nologin=$[$nologin + 1]
            nologin_name="$(echo $i|cut -d: -f 1), $nologin_name"
    fi

done

echo "There are $nusr users can login to the system, they are:"

echo " $nusr_name"
echo -e "\n"

echo "There are $nologin users can not login to the system, they are:"

echo " $nologin_name"
```

3. Write a script to use while loop to read the user name from a file named usr.lst. With each user read from usr.lst, display the appropriate information from /etc/passwd.
   The content of usr.lst as below
   root
   student
   student1

```
#!/bin/bash

while read line

do

    echo "$(grep ^$line: /etc/passwd)"

done </home/student/usr.lst
```

4. Re-write the above script but define a function to display the user information as follows:
   =========================
   USERNAME: <user name>
   Home directory: <home directory>
   Default shell: <default shell>
   =========================

   Use this function to display the information of users in the usr.lst.

```
#!/bin/bash

usrinfo() {

echo "========================="

echo "USERNAME: $(grep ^$1: /etc/passwd |cut -d: -f 1)"

echo "Home directory: $(grep ^$1: /etc/passwd|cut -d: -f 6)"

echo "Default shell: $(grep ^$1: /etc/passwd|cut -d: -f 7)"

echo "========================="
}

while read line

do

    usrinfo $line

    #echo $line

done </home/student/usr.lst
```

III. Schedule script with at and cron table

5. Create a simple script to display a message to the screen as follows:
   "I'm running…"
   Using at command to schedule this script to run in the next 1, 5 and 10 minutes

   $ at now + 1 minutes -f <path to your script>
   $ at now + 5 minutes -f <path to your script>
   $ at now + 10 minutes -f <path to your script>

6. After the first job completed, using atq and atrm to list and remove all the jobs one by one

   $ atq

   $ atrm <the first at job id>
   $ atrm <the second at job id>

7. Add a job to cron table to run a script at every minutes of the Thursday of every weeks. You could use the script in step 5.
   $ export EDITOR=vi
   $ crontab -e
   *** * * * 4 <path to your script>**
   :wq!