Lab 3

Scripts

3.1 Introduction

Usually operating systems administration tasks should be repeated again and again, so the administrator must enter new orders, sometimes changing only one input parameter. Doing these tasks manually, not only implies a considerable investment of time, but exposes the system to errors when repeating a command in a wrong way. The automation of these tasks using scripting languages improves system efficiency as these are done without human intervention; increases reliability because the commands are executed in the same way each time and also ensures consistency in the execution beacuse these tasks can be easily programmed to run periodically.

Although the automation could be made in any programming language, there are some languages, known as scripting languages, which allow to combine easily system commands with the expressions of the scripting language itself. Additionally they facilitate the manipulation of text files, lists, and directories and other useful tasks for system administration. There are many scripting languages available: the associated to the shell (like Bash or C shell) and other features, such as Perl or Python, with more advaned functionality.

3.2 Objective

Learn how to automate common system administration tasks using scripting, in this lab we will use `Bash`.

3.3 Before you start

Review basic programming with Bash shell-scripts.

This lab consists of two parts: the first consist of anlyzing a sample scripts for the detection of ”unnecessary” users in the system. The second part consist of making a script for managing disk space.

Scripts can be done in any shell scripting language but we strongly suggest you use `Bash`. In addition to proposed scripts a final list of problems is presennted for practicing outside the laboratory.

Keep in mind all the time properties of a good script [1]:

1. A script should run without errors.
2. It must perform the task for which it is intended.
3. The program logic must be clearly defined.
4. A script should not do unnecessary work.
5. Scripts should be reusable.

3.4 Script to detect invaliad users

You are asked to make a script that determines which users in `/etc/passwd` are invalid. A user is invalid if he/she is in the `passwd` file but did not have any presence in the system (ie. it has no files).

Also, there are users that have no files, but that are used to run system daemons. Add an option to declare valid users those that have a running process (`-p` flag).

3.4.1 Description of the desired results

An example of script's output without and with the `-p` flag is presented below:

```
$./BadUsers.sh
daemon
bin
sys
sync
games
lp
mail
news
rserral
emorancho
proxy
backup

$./BadUsers.sh -p
bin
sync
games
lp
news
rserral
proxy
backup
```

3.4.2 BASH version of the script

now you have the BASH version of the script. Fill-in the empty spaces with the appropriate language constructs.

```
#!/bin/bash
p=0

function print_help
{
  echo "Usage: $1 [options] "
  echo "Possible options:"
  echo "-p validate users with running process"
}

if [ $# -lt 1 ]; then
  print_help $0
  exit
fi

while [ $# -gt 0 ]; do
  case $1 in
    "-p")
      p=1
```

```
20        shift;;
21      *) echo "Error: not valid option: $1"
22        exit 1;;
23    esac
24  done
25
26  for user in _____________; do
27    home=$(cat /etc/passwd | grep "^$user\>" | cut -d: -f6)
28    if [ -d $home ]; then
29      num_fich=$(find "$home" -type f -user $user | wc -l)
30    else
31      num_fich=0
32    fi
33
34    if [ $num_fich -eq 0 ]; then
35      if [ $p -eq 1 ]; then
36        user_proc=__________________
37        if [ $user_proc -eq 0 ]; then
38          echo $user
39        fi
40      else
41        echo "The user $user has no files in $home"
42      fi
43    fi
44  done
```

What is the purpose of the shift command in line 20?

What is the mening of the grep command in line 27?

3.4.3 Detection of unactive users

Now extend the previous script to detect inactive users. An inactive user is defined as someone who do not have any running process, that long ago have not login (see commands `last` and `lastlogin`), and that long ago have not changed any of their files (see time options of `find`). The period of inactivity should be indicated through a parameter:

```
$./BadUsers.sh -t 2d (indicates 2 days)
alvarez
aduran
xavim
marcg
```

```
 $./BadUsers.sh -t 4m (indicates 4 months)
xavim
marcg
```

Modify the script to include the support for the new option in order to detect inactive users

3.5 Script for disk space management

Make a scripts that computes the disk space used by each user on the whole system. If the amount of disk space exceeds certain space that is passes as a parameter, then wirte a message to the in question to inform him/her to d delete or compress some files. Specifically, the syntax of the program should be the following:

```
$ disk-usage.sh <space_limit>

Per exemple:
$ ./ocupacio.sh 600M
       root        567 MB
       alvarez     128 KB
       aduran      120 MB
       xavim       23 MB
       ( ... )
```

After this extend the script to add an option for groups: `-g`: With this option the script must return the total disk usage for the each one of the users in the specified group, the total disk usage of the whole group, and put a message to users that exceeds the defined limit.

Therefore, the syntax of the final program will be:

```
$ ocupacio.sh [-g grup] max_permes
Per example:
$ ./ocupacio.sh -g users 500K
alvarez            128 KB
xavim              23 MB
( ... )
```

Note: The message should be left on the `.bash_profile` file. The user must be able to identify and delte the messahe withou problem. This means that alongside the message it should get instructions to remove it without trouble.

Bibliography

[1] M. Garrels, *Bash Guide for Beginners.* The Linux Documentation Project. TLDP. [Online]. Available: http://tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf

[2] R. L. Schwartz, T. Phoenix, and brian d foy, *Learning Perl*, 4th ed. Sebastopol, USA: O'Reilly Media, July 2005.

[3] M. Cooper, *Advanced Bash-Scripting Guide. An in-depth exploration of the art of shell scripting.* The Linux Documentation Project. TLDP. [Online]. Available: http://tldp.org/LDP/abs/abs-guide.pdf