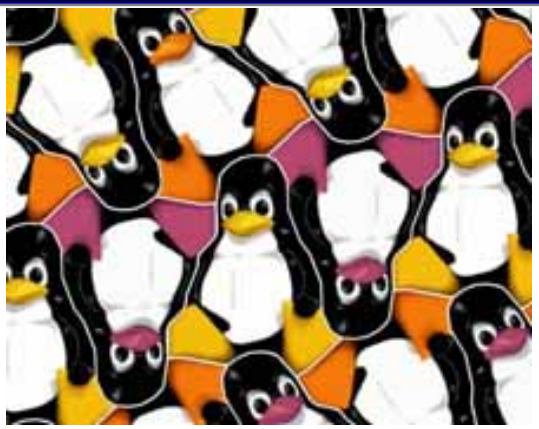
UNIVERSITY OF THE WEST of SCOTLAND

COMP09024 Unix System Administration

Laboratory 3: User Management



Filesystem Hierarchy / User Management

Learning Outcomes

- **Filesystem Hierarchy:** `/, /root, /bin, /usr, /dev, /etc, /lib, /var, /proc`
- **Commands:** `grep, find, du, hostname, sort, ps`
- **Creating and deleting user accounts**

In this lab we will start by reviewing the structure of the Linux filesystem. The filesystem is central to the success and convenience of Unix. It is a good example of the 'keep it simple' philosophy, showing the power achieved by careful implementation of a few well-chosen ideas. In the second part of the lab we will tackle one of the foremost tasks of system management: the creation and deletion of user accounts.

Filesystem Hierarchy

Let's begin by comparing the Linux FS with its Windows counterpart as many new users more familiar with Windows find the differences confusing. Linux does things differently. In Linux there is only a single hierarchical directory structure. Everything starts from the root directory, represented by '`/`', and then expands into sub-directories. Where DOS/Windows has various partitions and then directories under those partitions, Linux places all the partitions under the root directory by 'mounting' them under specific directories. Closest to root under Windows would be `C:`. Under Windows, the various partitions are detected at boot time and assigned a drive letter. Under Linux, unless you mount a partition or a device, the system does not know of its existence. The assignment of the mount points is done by **`/etc/fstab`** at mount time. This might not seem the easiest way of doing things, but it offers great flexibility when mounting various devices.

This approach, known as the unified filesystem, offers several advantages over that used in Windows. Let's take the example of the **/usr** directory. This directory off the root directory contains most of the system executables. Under Linux, you can choose to mount it off another partition or even off another machine over a network. The underlying system need not know the difference because **/usr** appears to be a local directory! How many times have you wished to move executables and data under Windows, only to run into registry and system errors? Another difference, is the use of the forwardslash '/' rather than the backslash '\' in pathnames. Linux is not going against convention here, as this was the standard long before Windows came along. Furthermore, Linux also chooses to be case sensitive, as different cases for the same character are represented by different ASCII codes. It therefore matters whether a command or a variable is written in upper case capitals or lower case letters or a mixture of both (not advisable). E.g. **'echo'** is not the same as **'ECHO'** or **'EcHo'** for that matter. This one feature probably causes the most problems for people learning Linux.

Log in as the super user on your machine. The root directory is given by the forward slash: /. So it does not actually have a name at all. That's a little confusing at the beginning, especially since the home directory of the root-user is called **/root** in Linux. However, remember **/** and **/root** are different directories! In order to distinguish between them, we will refer to the **/** as the root directory and to **/root** always as the home directory of the root user:

/ = root directory
/root = home directory of the root user

Now let's set in the root directory:

```
[root@UWS ~]# cd /
```

Use the **ls -l** command to find out which directories lie directly under the root directory. If you use the **-l** qualifier a so-called long listing of files will appear. The first letter of each entry actually indicates what type of file you are dealing with. If an entry starts with a **d** the corresponding file is a directory. If it starts with a minus sign **-** it is referring to a regular file (other types will be introduced later). One way to show only directories within a file system is to forward the **ls -l** output to the **grep** command. The **grep** command looks for the occurrence of a specific regular expression pattern within a text (ASCII) file. In combination with other Unix commands it is especially powerful.

In order to specify that the first entry of the output (from **ls -l**) has to start with a **d** (for directory) one would use the regular expression **'^d'**. The ^ sign is called 'caret' and indicates that a line (of entry) HAS to start with **d**. Piping (|) the output of the **ls** command to the input of the **grep** command, viz:

ls -l | grep '^d' will display only the lines starting with **d**.

Question

- **Q3.1)** Name the different directories under the **/** directory

Now let's try to set one directory higher by typing in the following:

```
[root@UWS /]# cd ..
```

Then check the directory contents again by typing:

```
[root@UWS /]# ls -l
```

Question

- **Q3.2)** Explain why there is no difference. (Tip: the root directory is the uppermost!)

Unix also has the concept of hidden files. These are normally files of somewhat minor or let's say hidden importance that have to be in a certain directory and it would be a nuisance if they were displayed every time you invoked the **`ls`** command. All of these hidden files start with a dot (**`.`**) . The **`ls`** command itself ignores all files by default that start with a dot. In order to display these files we have to use the **`–a`** qualifier.

Questions

- **Q3.3)** Are there any hidden files in the root directory? If yes: What are their names?
- **Q3.4)** Explain the meaning of the **`.`** ('dot') and the **`..`** ('double dot') in the command lines **`cd .`** and **`cd ..`**

Now let's investigate several of the standard directories. We will begin with the **`/bin`** directory.

`/bin`

The **`/bin`** *directory contains standard Unix commands that are used by both the system administrator as well as non-privileged users. For example, this directory usually contains shells like* **`bash`**, **`csh`** *etc. as well as indispensable commands like* **`cp`**, **`mv`**, **`rm`**, **`cat`**, **`ls`**.

Now set in the **`/bin`** directory

```
[root@UWS /]# cd /bin
```

This directory contains elementary Unix commands and programs. Use the **`ls –l | grep '^d'`** command to find out whether there are any subdirectories in the **`/bin`** directory.

Question

- **Q3.5)** Are there any subdirectories in **`/bin`**?

Now let's find out *approximately* how many commands are in **`/bin`**. Type:

```
[root@UWS bin]# ls  –l   |   wc  -l
```

This is only approximate because **`wc –l`** also counts the initial line beginning 'total'.

Question

- **Q3.6)** How many commands are in **`/bin`**? Write down and explain two commands that you already know.

Note that there are other directories that also contain standard commands, e.g. **`/usr/sbin`** (for non-essential user binaries) and **`/opt`** (for add-on packages).

Questions

- **Q3.7)** Which of the four directories **`/bin`** , **`/usr/bin`** , **`/usr/sbin`** , **`/opt`** contains the most commands?
- **Q3.8)** Which of the four directories contains a large set of gnome-desktop related applications?

`/usr`

This is one of the most important directories in the system as it contains almost all of the user binaries. X (the portable, network-transparent window system) and its supporting libraries can be found here as well as other prominent binaries.

Question

- **Q3.9)** Can you locate the **chroot** binary within **`/usr/sbin`**? Where does it point to? (remember the sign: **`->`** indicates a symbolic link aka shortcut). Which is the abosulte path (from the root **/**)?

After this set once again in the root directory. To understand the next question, we need to know a little more about the Unix filesystem. Almost every entry within the Unix system is treated as a file. Even a directory itself is just a (special) file that contains information about the various files 'contained' within it. Directory files are so special, that even the super user is not allowed to edit them directly! If you update a file system in any way, the kernel will update the entries in the relevant directory files silently. The directory file itself contains information about each associated filename, its file size, its permission, creation date, etc. Directory-files are written in binary format (to keep them small) and can be accessed by appropriate commands such as **`ls.`** The file size of a given directory file is proportional to the files that are elements of that directory. For this reason you can deduce the number of files that are within a given directory by getting the size of the directory from the **`ls –l`** command, where this information is given in the 5th column.

The actual size of a directory CAN'T be obtained in this way (but can be found by using the **`du`** command). Set in the root directory and type **`ls –l`**. Making use of the foregoing information, answer the following questions:

Questions

- **Q3.10)** What is the smallest size allocated to a directory file in **`/usr`**?
  (type: **`ls –l`**)

- **Q3.11)** Why might **`/usr/lib`** be so large?

To find the actual size of the system we have to use the **du** (disk usage command). If you use a command like **du** at a high level within the directory structure without any qualifiers it will descend through all sub-ordinate directories and give voluminous information. We therefore have to tell the command NOT to descend for more than one directory level. In the case of the **du** command this is done with a specific qualifier that is introduced with two minus signs: **`--max-depth=1`**. This qualifier tells **du** to only go down one (**1**) directory from the current location. So in order to find out the respective total sizes of the standard directories type:

```
[root@UWS ~]# cd /
[root@UWS /]# du -k --max-depth=1
```

(Note: -**k** qualifier, displays the contents in kilobytes; **–h** qualifier shows human-readable results.)

Questions

- **Q3.12)** Which is the biggest standard directory in our system?
- **Q3.13)** What is the total size of our current system? (Tip: displayed on the last line of the output in units of kbyte)

`/dev`

This is an interesting directory that once again highlights that everything in a Linux filesystem is essentially a file. Look through this directory and you should see **`sda1`, `sda2`** *etc., which represent the various partitions on the first master drive of the system.* **`/dev/cdrom`** *and* **`/dev/fd0`** *represent your CDROM drive and your floppy drive. This may seem strange but it will make sense if you compare the characteristics of files to that of your hardware. Both can be read from and written to. Take* **`/dev/stout`**, *for instance. This file represents your standard output. So any data written to this file will be re-directed to your terminal as this is the allocated standard output.*

Questions

- **Q3.14)** Try the command, **`cat /etc/passwd > /dev/stdout`**. (The **`cat`** command displays (concatenates) the contents of a file to an output device such as the screen…) Explain why you see the contents of the file displayed (Tip: this is explained above!)

- **Q3.15)** If **`stderr`** is the standard channel for displaying error messages. Where is **`stderr`** directed to? (Tip: use the same command as above, changing **`stdout`** to **`stderr`**).

`/etc`

This directory contains numerous system configuration files. Such as: **`hosts`**, **`resolv.conf`** *and* **`fstab`**. *The* **`/etc/rcx.d`** *directories, holding various startup scripts, are also contained in this*

directory. It is therefore advisable to backup **/etc** *regularly, as this could save a lot of re-configuration work if you ever need to perform a re-install for any reason.*

The usefulness of this directory is best illustrated by an example. Let's assume that you want to change the IP-address of your system and the only thing that you know is that your current IP-address starts with **141.191.200.201**.
In order to quickly identify all files, that contain this address you might set in the **/etc** directory and look for all files that reference this specific number sequence. This can be done using the **find** and **grep** commands.

When using the **find** command, you can filter for specific files, using various qualifiers including **–name, -size** and **–type**, and you can then execute any Unix command on the resulting group of files. To do so you invoke the **–exec** at the end of the **find** command. The group of files selected using **find** is always represented by juxtaposed curly brackets **{}**.The general form to execute any Unix command is:

```
-exec command {} \;
```

Thus:

-exec ls {} \;	(lists all files found by the **find** command)
-exec grep '146.191.*' {} \;	(checks for '146.191' in all files that were found)
-exec {} \;	(executes the files, if they are executables)

the **\;** (backslash semicolon) is the closure sign, that terminates input to the find command. For the above example let us first set in the **/etc** directory and use the **hostname** command to discover the currently assigned system IP address (don't worry if it is just the *loopback address*).

```
[root@UWS ~]# ifconfig
[root@UWS ~]# find /etc -type f -exec grep  '127.0.0.1' {} \;
```

Ensure that you do not leave any space between the curly brackets above! Now, have a look at the output. It should show you ALL entries in any files that contain the information about your IP-address. Now in order to clarify which files actually contain these entries we will extend the **grep** command with the **-l** (not one, but L) qualifier:

```
[root@UWS ~]# find /etc -type f -exec grep -l '127.0.0.1' {} \;
```

Questions

- **Q3.16)** What is the meaning of the -**l** qualifier in the **grep** command?
- **Q3.17)** Which files reference the IP-address?

In the above example, once you have found all files that reference your IP-address, you are just one step away from changing it. In essence Linux allows you to quickly check for ALL entries within ANY directory just by combining the **find** and **grep** commands. The strictly hierarchical filesystem structure GUARANTEES that you are not missing out on crucial files. This approach to accessing and editing critical system features and settings is utilised by a lot of professionals.

Question

- **Q3.18)** How would you make absolutely sure, you are examining the whole filesystem for e.g. the occurrence of a pattern like **`127.0.0.1`**?

Before moving on, set in the root directory and summarize your knowledge of the find command by answering the following questions.

Questions

- **Q3.19)** Locate the Linux commands **`fsck`** and **`find`** in the filesystem.
- **Q3.20)** Locate the standar C library represented by the file **`libc.so.6`**
- **Q3.21)** Locate and run the application **`dmesg`** using only the find command. Write down the full (`find`) command line that you used in your log-book.

`/home`

Linux is a multi-user environment, so each user is also assigned a specific directory, which is accessible only to them and the system administrator. These are known as home directories and can be found under **`/home/username`**. *These directories also contain user specific settings for programs like IRC, X etc.*

Use the **`du -k`** command to find out the current disk usage of the **`/home`** directory and all of its individual subdirectories.

Question

- **Q3.22)** What is the full size of the **`/home`** directory? (Tip: Experiment with the command-lines:

  ```
  # du -k /home  | sort
  ```

  and

  ```
  # du -k /home | sort -nr
  ```

  to help you answer this question). What are the two biggest subdirectories within **`/home`**? (Use the **`man du`** command to find out about the display unit sizes).

`/lib`

The **`/lib`** *directory contains all the shared libraries that are used by the Linux OS.*

With your previously gained knowledge, try to answer the following question:

Question

- **Q3.23)** Determine the number of shared libraries in **`/lib`**. Shared libraries end in **`.so.*`** (Where * represents a wildcard).

`/var`

This directory contains spooling data like mail and also the output from the printer daemon. The system logs are also kept here in **`/var/log/messages`**. *You will also find a database for various information systems relating to computer networking in here.*

Question

- **Q3.24)** Find out whether *libreoffice* has an entry in **`/var`**.

The **`/var`** filesystem has a tendency to fill up fairly quickly. This is mainly due to log files that keep track of **`ftp`** requests and such like. In order to avoid this, some shell-scripts automatically delete old log-files after a specified period of time. Next critically evaluate the contents of some important files within **`/var`**.

```
[root@UWS ~]# cd /var/log
[root@UWS log]# more messages
[root@UWS log]# more lastlog
```

Question

- **Q3.25)** Deduce the role of **`/var/log/messages`** by reviewing its contents.

Now let us dummy an unsuccessful system hack! A normal user (student) wants to login as root, but fails to provide the right password. The file **`messages`** keeps a record of such attempts and allows you to easily identify the culprit! To do so type:

```
[root@UWS ~]# su -l student     (to login as student)
[student@UWS ~]# su -l root     (to simulate an unsuccessful root login)
```

In response to the password request, deliberately provide the wrong root password e.g. : **`abcd`**
Next exit the student shell by:

```
[student@UWS ~]# exit
```

Then check the contents of **`/var/log/messages`** by:

```
[root@UWS ~]# tail /var/log/messages
```

Check the last two entries in this file.

Question

- **Q3.26)** What information is given about the unsuccessful login attempt? Could you identify the hacking culprit at once?
- **Q3.27)** How does the **`tail`** command compare to the **`more`** command?

/proc

The name **`/proc`** *stands for process information pseudo filesystem. It's a part of the memory that contains information about running processes and is constantly updated by the kernel.*

This filesystem was introduced to allow easy access to crucial data pertaining to running processes. A lot of standard programs dealing with the handling of processes actually rely entirely on **`/proc`**.

Set into the directory **/proc** by:

```
[root@UWS ~]# cd /proc
```

then type:

```
[root@UWS proc]# ls -l
```

Question

- **Q3.28)** What is the name and the size of the biggest filesystem entry of **`/proc`**? Do you have any idea what it may represent?

To examine live processes we shall use the **`ps`** command. Invoking the command as root with the qualifiers: **`ps -ef`**, gives a listing of all currently running processes. A special number called the process identification number or PID is used to uniquely identify each process. This number is displayed in the second column of the resulting output. Additional information displayed includes: process running time, process owner and the program that originally invoked the process.

To understand the interplay between the **`/proc`** directory and the **`ps`** command type:

```
[root@UWS proc]# ps -ef
```

then again (in the **`/proc`**) directory:

```
[root@UWS proc]# ls  -l  |  grep '^d'
```

Questions

- **Q3.29)** What is the link between the PID of the running processes and the directory names within **`/proc`**?

- **Q3.30)** What do you think will happen to each directory in **`/proc`** after the associated process has been killed?

Let us examine the contents of a specific directory within **`/proc`**. To do so:

```
[root@UWS proc]# cd  /proc/1                    (subdirectory ONE)
[root@UWS 1]# ls
```

Question

- **Q3.31)** What is written in the **`cmdline`** file (use: **`more cmdline`** to see the contents displayed)? Does this agree with the information as given by the command, **`ps -ef`**?

Now check the file **`status`** as follows:

```
[root@UWS 1]# more /proc/1/status
```

Question

- **Q3.32)** What is the status (check the contents of **`/proc/1/status`**) and the memory size (`VmSize`) that is used by the **`init`** process?

The concept of a pseudo filesystem may seem bizarre. However the benefits are quite considerable: instant access to kernel data, without interrupting and slowing down the system!

`/media`

This is a generic mount point under which you mount external filesystems 'aka' devices. (in some systems it may also be called `/mnt`*) Mounting is the process by which you make a filesystem available to the OS. After mounting, your files become accessible under the mount-point. This directory usually contains mount points or sub-directories where you may mount floppies, CDs and other media. There is no limitation to creating a mount-point anywhere on your system but it makes sense to avoid indiscriminately littering the file system with mount-points. You can create additional mount-points here if you wish and name these as you please. This is a very different, strategy to MS-Windows, where mount-points are hard-coded, such as Floppy=A.*

Follow the directions given in the laboratory 2 to mount a **virtual** floppy.

Question

- **Q3.33)** Name the different directories that are present in the **`/media`** directory.

Now set in **/media/floppy** (having a formatted floppy with some data on it inserted).

```
[root@UWS media]# cd /media/floppy
[root@UWS floppy]# echo 'A dummy file' > testfile.dat
[root@UWS floppy]# ls -l
```

Question

- **Q3.34)** What will be displayed?

Once a file system is correctly mounted, you can disregard the underlying hardware implementation. It doesn't matter whether your file system is a magnetic drive, or a magnetic tape with serial access or even a part of system memory! Always keep in mind that one of Unix's philosophical principles is that: **all files are treated equal.**

User Management

User management within the Linux OS revolves around a specific file that can be found in the **/etc** directory. To find out more type:

```
[root@UWS ~]# more /etc/passwd
```

All users are recorded in this file (so never delete it!). During installation, Linux pre-defines a large number of users, including, of course, **root.** Root should be the 1st entry in the file. The meaning of the colon delimited entries is as follows:

```
username:password:userID:groupID:user_info:home_directory:login_shell.
```

This is all the information the machine needs to know about any user at login time. Let us use the **grep** command to look for all occurrences of the name **root** within the file **/etc/passwd**.

```
[root@UWS ~]# cat /etc/passwd | grep 'root'
```

Now use the **grep** command to help answer the following questions:

Questions

- **3.35)** Why is the actual password depicted as **x**, although the password is not **x**?
- **3.36)** What is the user identification (UID), home directory and login shell of the root-user?
- **3.37)** What is the UID of the daemon called **uucp**?
- **3.38)** Determine the UID of **student** in the Debian system.
- **3.39)** How many users use the **/bin/sh** shell as their login shell? (Tip: pipe the output of **cat /etc/passwd | grep '/bin/sh'** into the word count command **wc -l**). What does **wc -l** actually do? (Use the man page for **wc** to find the answer to that question)

When you add (or delete) a user you are essentially just adding or deleting entries in the **/etc/passwd** (and the **/etc/shadow**) file. Linux provides several commands that indirectly edit this file. At the command line we will use the **useradd** command of the form:

adduser -h homedir -s loginshell login_name.

Let us create a new user on the system:

[root@UWS ~]# adduser -D -h /home/guest1 -s /bin/bash guest1

This will create a guest account for a login name called guest1. His/her home directory will be **/home/guest1** and his/her login shell will be the bash shell. Of course the user guest also needs a password. The appropriate command is:

[root@UWS ~]# passwd guest1

This command will prompt for the password to be given. **(**Make sure you don't forget to type **guest1** after the **passwd** command!!!)

Having created a user account for guest1, assign the password: **only_guest**. After finishing check for success with:

[root@UWS ~]# more /etc/passwd | grep 'guest1'

and by checking the existence of a home directory in **/home.** Now issue the commands:

[root@UWS ~]# adduser -D -H guest2

and check for the existence of **guest2** and **guest3** directories on **/home**. Set the same password for the new users called **guest2** and **guest3,** continue with:

so_comp-e113a-01# cat /etc/passwd | grep 'guest2'

Questions

- **Q3.40)** Which UIDs were given to the user **guest1** and **guest2**?
- **Q3.41)** Why do you think you don't have to supply the UIDs yourself?
- **Q3.42)** Is there any difference between the commands **adduser -H** and **adduser**? (Check for the existence of **guest1** and **guest2** home directories)
- **Q3.43)** Which directory is assigned by default to **guest1** by the command **adduser** (tip: examine and interpret the corresponding line in **/etc/passwd**

Now review the **man** page for **adduser** to find out which file contains the information about the UID settings in our Linux distribution (Tip: this file will also be in **/etc**, it is the last file mentioned at the end of the man page for **adduser**, underneath the heading FILES). Once you have found the name of the file, display its contents and try to figure out the answer to the next question:

Question

- **Q3.44)** What is the maximum number of users that the virtualized Linux system can automatically deal with (Tip: Look at the minimum and maximum settings for the UID-variable in the file that you just found)?

- **Q3.45)** How could you create more users?

Now, delete your newly created guest users and their home directories with the **`deluser`** command. Use **`man userdel`** to figure out which qualifier will allow you to delete a user and their home directory at the same time. If you are uncertain, ask the teaching staff for confirmation beforehand. Attention: the command **`rm`** will not do this! (You should not use **`rm`**, until explicitly advised to do so!)

Question

- **Q3.46)** What is the command that deletes a specific user and their home directory at the same time?

– END OF LAB –