Introduction

The purpose of this experiment is to get familiar with the environment that will be used in the following labs. We use Linux, a Unix-like operating system for PC's. Linux is an open source operating system that offers a great flexibility for teaching and making experiments.

Directory Structure

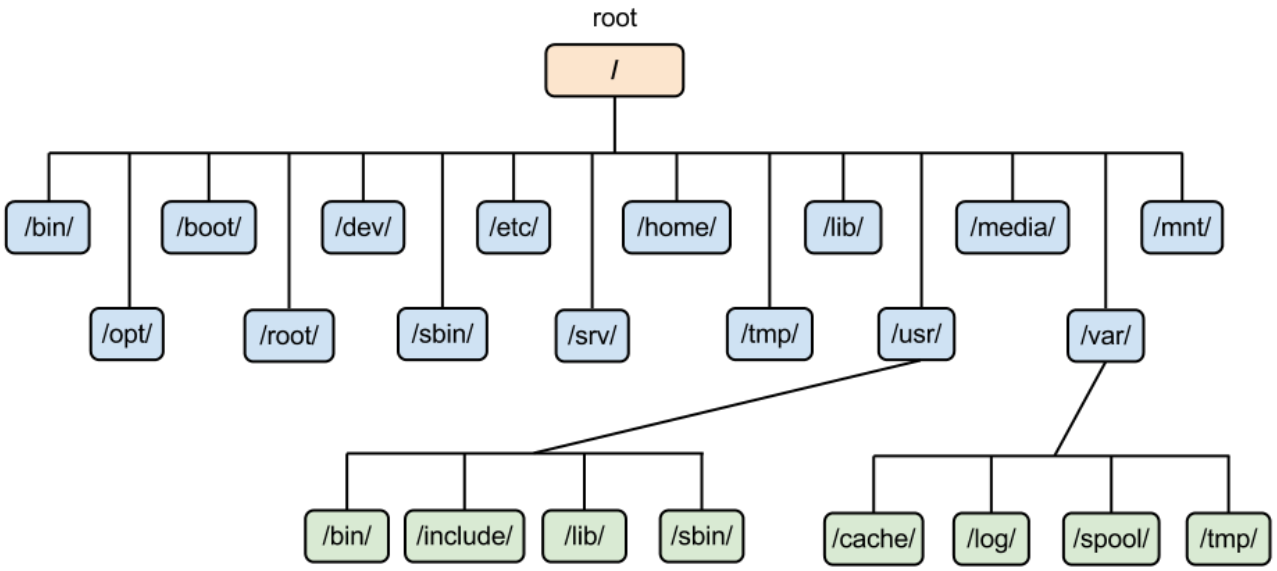

Figure 1: Linux directory hierarchy, source

In Linux, there are no drive letters. A single device is designated as the "root". This device has the special responsibility of containing directory *mount* points. Directory *mount* points are just regular directories in which drives and partitions are mounted. All drives and partitions (floppy disks, CD-ROMs, hard disk, removable USB drives…. etc) are configured to be *mounted* on one of the directories on the file system.
Most mount-points are put into a directory called *"/mnt"*. This is merely a convention. Thus the cd-rom is mounted at the location */mnt/cdrom*, the floppy can usually be found at the location */mnt/floppy*. Most Linux distributions will automatically detected when a CD-ROM, floppy, or other removable media are inserted, and automatically mount to an appropriate location. Before the media can be ejected, *unmount* must be performed (often the GUI will do the unmount for you automatically).
Each user has a home directory "/home/user_name" where he can keep his personal data.

Check this tutorial to find out what are the contents of each directory in the hierarchy

Linux Commands

Students should understand and practice using the following set of basic command[1].
Command syntax <command> <space> <options>

Bash Commands		Directory Operations	
uname -a	Show system & kernel	pwd	Show current directory
head -n1 /etc/issue	Show distribution	mkdir <dir>	Make directory dir
date	Show system date	rmdir <dir>	Remove empty directory dir
uptime	Show uptime	rm -r <dir>	Remove directory dir
whoami	Show your username	cd <dir>	Change directory to dir
who	Show logged users	cd ..	Go up a directory
man <*command*>	Show manual page for command	ls	List files in current dir
echo <*msg*>ppp	Print msg to the output	ls <dir>	List the give dir
seq <first> <step> <last>	Print numbers from first till last incremented or decremented by step value	IO Redirection	
sort <stream>	sorts a text stream or file ascending or descending	cmd < <file>	Input of cmd from file
uniq	Remove duplicates from sorted stream	cmd > <file>	Write output to file
paste <file1> <file2>	Merge different files into a single, multi-column file	cmd > /dev/null	Discard output
tr	Translate all occurrences of one character to another	cmd >> <file>	Append output to file
grep <pattern>	pattern file Search the target file(s) for occurrences of pattern, where pattern may be literal text or regular expression	<cmd1> | <cmd2>	Output of cmd1 as input to cmd2
wc	Count the number of newline, word, and byte counts	File Operations	
clear	Clear	touch <file1>	Create file1

chmod -R <file> <permissions>	change access permission of file one to the new permissions	cat <file1> <file2>	Concatenate files and output
chown	change ownership of file	cat <file1>	View file1 content
sudo	super user do	more <file1>	View and paginate file1
gedit file &	opens the file in the gedit program in the background.	less <file1>	Like more but allow movement in forward & backward direction
Process Management		file <file1>	Get type of file1
ps	Show snapshot of processes	cp <file1> <file2>	Copy file1 to file2
top	Show real time processes	mv <file1> <file2>	Move file1 to file2
kill -9 <pid>	Kill process with id pid	rm <file1>	Delete file1
pkill -9 <name>	Kill process with name name	head <file1>	Show first 10 lines of file1
killall -9 <name>	Kill processes beginning with name	tail <file1>	file1 Show last 10 lines of file1

grep patterns

- ^ (Caret) = match expression at the start of a line, as in ^A.
- $ (dollar sign) = match expression at the end of a line, as in A$.
- \ (Back Slash) = turn off the special meaning of the next character, as in \^.
- [ ] (Brackets) = match any one of the enclosed characters, as in [aeiou].
- Use Hyphen "-" for a range, as in [0-9].
- [^] = match any one character except those enclosed in [ ], as in [^0-9].
- . (Period) = match a single character of any value, except end of line.
- * (Asterisk) = match zero or more of the preceding character or expression.
- \{x,y\} = match x to y occurrences of the preceding.
- \{x\} = match exactly x occurrences of the preceding.
- \{x,\} = match x or more occurrences of the preceding.

Examples

`$ls | wc -l`	it counts the number of contents in the current directory
`$cd /home/`	it shows the home directories of all users
`$echo pwd > file`	write the current path in file (if file doesn't exist it creates it, if it does it clear its content)
`$seq 8 -1 2`	prints 8 7 6 5 4 3 2 , each number in a new line
`$man cp | head -5`	it shows the first 5 lines of manual page for cp
`$cat file1 file2 >> file3`	it concatenates the three files in a third file (file3 then file1 then file2 , because the operator >> appends)
`$ls /`	it shows the linux structure
`$file ~/Desktop`	it shows the type of the folder/file , in this case it is a folder
`cat words.txt | tr '[A-Z]' '[a-z]'`	Replace letters capital with small ones
`tr < words.txt '[:upper:]' '[:lower:]'`	Replace letters capital with small ones
`cat words.txt | tr -d 'i'`	Delete all occurrences of letter "i"
`grep "AI" words.txt`	Search for lines containing "AI"
`grep -i "ai" words.txt`	Search for lines containing "ai", ignore case
`grep -n "AI" words.txt`	Display line number
`grep "^A" words.txt`	Search for lines starting with "A"
`grep "s$" *.txt`	Search for lines ending with "s" in all files with extension .txt in the current directory
`grep " " words.txt`	Search for lines containing spaces
`grep "Data[a-zA-Z]*" words.txt`	Search for lines with the word "Data" followed by zero or more small or capital letters
`grep "Data[ a-zA-Z] \{1,3\} " words.txt`	Search for lines with the word "Data"followed by 1-3 small letters, capital letters or spaces.
`grep -i -n "^this.*toy$"`	Search for lines starting with the word "this" and ending with "toy" ignoring case and show the line number

File Systems Security (Access Rights)

Try typing the following command in your terminal:

```
% ls -l
```

(l for a long listing!)

You will see that you now get lots of details about the contents of your directory, similar to the example below:

```
-rw-rw-r-- 1 joe students 1920 Sep 29 18:01 extern.c
```

What do these fields mean?

- -rwxr-xr-- → permissions
- 1 → number of linked hard-links
- joe → owner of the file
- students → to which group this file belongs to
- 1920 → File size
- Sep 29 18:01 →modification/creation date and time
- extern.c →file/directory name

Who is permitted to use these files/directories? (user types)

- u - the user who owns the file
- g - The group that the file belongs to
- o - other users not in the file's group
- a - all users

What are the permissions?

Each user type have 3 permissions in the following order;

- r - Read
- w - Write
- x - Execute

each permission has a bit, Example:

- A user that has a read permission only has a rwx value = 100 (Binary) = 4 in decimal
- A user that has a write permission only has a rwx value =010 (Binary) = 2 in decimal
- A user that has a read & write & execute has a rwx value = 111 (Binary) = 7 in decimal

Let's understand the extern.c file permissions. The file permission has 9 places;

1. First place indicates the type of the file:
    a. I this specifies symbolic links (shortcut)
    b. d stands for directory (folder)
    c. c stands for character file
    d. - stands for a regular file
2. The next three places are the rwx permissions for the file owner (u) → in this example the user has a read and write permissions only (rw-)
3. The next three places are the rwx permissions for the file group (g) → in this example the group has a read and write permissions only (rw-)
4. The next three places are the rwx permissions for the file group (a) → in this example everyone else has a read permission only (r--)

How to give/remove permission?

- + Add permissions
- - Remove permissions

Example:

- Give the user a read and execute permissions for file extern.c:
    - chmod u+rx extern.c
- Give the user a read and execute permissions, and read only for the rest, for file extern.c:
    - chmod 544 extern.c,
    - chmod u+rx extern.c; chmod gu+r extern.c
- Remove wx permissions from the user
    - chmod u-wx extern.c

Requirement

Hints:

- All commands you need are already given .
- You might need options that we didn't mention, so use the 'man' command to get help and to know what other options are there.
- In order to get the full mark you need to do all the steps using commands only and each step in one line.
- If you get stuck in a command, pass it (do it using gui) , and continue to do the rest , then try it at the end.

Script Steps:

- Open the terminal from the desktop, make sure that you're in Desktop path, type this command "gedit myfirstscript.sh &" → this will open a new file on the desktop in a GUI editor in the background.
- First line of the script should direct the terminal into which interpreter to use to interpret your file, here we use the **bash interpreter** and it's located under /bin. The **#!** symbol is the directive.
  Type: "**#! /bin/bash**" at the beginning of your file.
- In the line after it, use **echo command** to write your name to the terminal
- From the terminal, change the permissions of the file to be executable, as mentioned, to make a file executable, use the chmod command → check the example above
- Now run your script from the terminal by typing your script name and hitting enter, Voila!

Now your script is ready for the real requirement, understand the requirement and add the command that executes what is required to the file: (Hint, run your file after each command)

Requirement Steps:

1. Delete the directory Lab1 if it exists.
2. Create a new directory called Lab1 on Desktop.
3. Copy the files "**words"** & "**numbers"** inside Lab1. (The two files should be placed initially on the Desktop).
4. Create a new file that contains the content of the words & numbers merged **side by side** and save it to a file called "**MergedContent**".
5. Display the first **three** lines in the file **MergedContent.**
6. Sort the **MergedContent** and save your result in the file **SortedMergedContent** in the same folder.
7. Display on the screen the following message "**The sorted file is :**"
8. Display the **SortedMergedContent.**
9. Prevent **anyone** from reading the **SortedMergedContent.**
10. Display the contents of the **MergedContent** after removing the duplicate lines. (You do not need to remove the duplicate lines from the actual file. Just display them without duplicates).
11. Try to replace all small letters with capital ones in **SortedMergedContent.**
12. Display a message on the screen explaining what happened in the last step and why?
13. Solve the problem in (12) and re-run (11) again.
14. Find the line numbers starting with "w" and ending with a number in **MergedContent.**
15. Replace every occurrence of "**i**" (as in **i**ron) in **MergedContent** with "o" and save it to **NewMergedContent** in the same folder.
16. Display both files MergeContent & NewMergeContent side by side on the terminal.