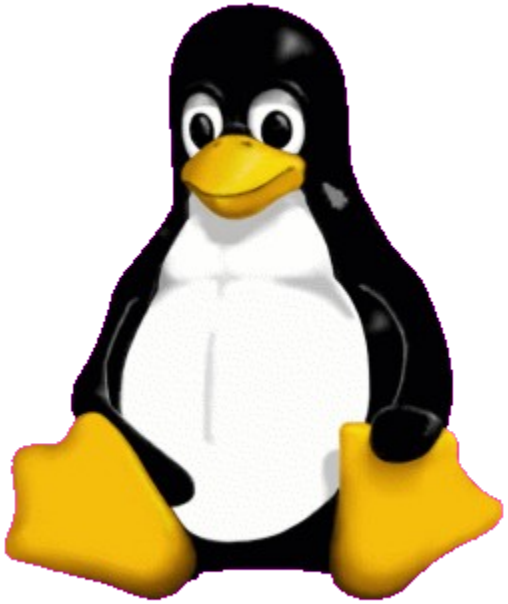
Lesson 3

GNU/Linux

Working with files

By Dr. Amin



Just a reminder!

1- In Linux everything is a file
2- File names in Linux are case sensitive

File1 ≠ file1

file

The file utility determines the file type.

```
am@am-UBOX  ~ $ file pic33.png
pic33.png: PNG image data, 3840 x 1200, 8-bit/color RGBA, non-interlaced
```

Exercise:

Try this in a terminal window

file /etc/passwd

File

-s

file -s for special files like those in /dev and /proc.

```
am@am-UBOX  ~ $ file /proc/cpuinfo
/proc/cpuinfo: empty
am@am-UBOX  ~ $ file -s /proc/cpuinfo
/proc/cpuinfo: ASCII C++ program text
```

Exercise:

Try this in a terminal window

file –s /proc/cpuinfo

touch

One easy way to create an empty file is with touch.

```
am@am-UBOX  ~ $ touch file42
am@am-UBOX  ~ $ touch file33
am@am-UBOX  ~ $ ls -l
total 0
-rw-r--r-- 1 paul paul 0 Oct 15 08:57 file33
-rw-r--r-- 1 paul paul 0 Oct 15 08:56 file42
```

Exercise:

Create two files in your home directory as:

touch file1 file2

Touch -t

Create a file with time and date stamp using touch -t.

```
am@am-UBOX  ~ $ touch –t 200505050000 SinkoDeMayo
total 0
-rw-r--r-- 1 am am 0 May 5 2005 SinkoDeMayo
```

Exercise:

Create a file called a1 created on 21 March 2014

touch -t 201403210000 a1

rm

When you no longer need a file, use rm to remove it.

```
am@am-UBOX  ~ $ ls
BigBattle.txt     file33     file42    SinkoDeMayo
am@am-UBOX  ~ $ rm BigBattle.txt
am@am-UBOX  ~ $ ls
File33    file42    SinkoDeMayo
```

Exercise:

Remove the file a1:

remove a1

Rm -i

To prevent yourself from accidentally removing a file, you can type rm -i.

```
am@am-UBOX  ~ $ ls
file33    file42    SinkoDeMayo
am@am-UBOX  ~ $ rm -i file33
rm: remove regular empty file `file33'? yes
```

Exercise:

Create two file called a321 b321

touch a321 b321

Check if it is there, then remove a321

rm a321

Rm -rf

The rm -rf statement is famous because it will erase anything (providing that you have the permissions to do so)

```
am@am-UBOX  ~ $ mkdir test
am@am-UBOX  ~ $ rm test
rm: cannot remove `test': Is a directory
am@am-UBOX  ~ $ rm -rf test
am@am-UBOX  ~ $ ls test
ls: cannot access test: No such file or directory
```

Exercise:

Create a directory : mkdir called samples

mkdir samples

Then remove it by: rm –rf samples

cp

To copy a file, use cp with a source and a target argument.

```
am@am-UBOX  ~ $ cp file42 file42.copy
am@am-UBOX  ~ $ ls
file42    file42.copy SinkoDeMayo
```

Exercise:

Copy file b321 to MyB321

cp b321 MyB321

cp

If the target is a directory, then the source files are copied to that target directory

```
am@am-UBOX  ~ $ mkdir dir42
am@am-UBOX  ~ $ cp SinkoDeMayo dir42
am@am-UBOX  ~ $ ls dir42/
SinkoDeMayo
```

Exercise:

Make a directory called MYB321

mkdir MyB321

Copy MyB321 to MyDir

cp b321 MyB321

cp -r

To copy complete directories, use cp -r (the -r option forces recursive copying of all files

```
am@am-UBOX  ~ $ ls
dir42    file42    file42.copy      SinkoDeMayo
am@am-UBOX  ~ $ cp -r dir42/ dir33
am@am-UBOX  ~ $ ls
dir33    dir42    file42    file42.copy SinkoDeMayo
am@am-UBOX  ~ $ ls dir33/
SinkoDeMayo
```

Exercise:

Self practice

cp -i

To prevent cp from overwriting existing files, use the -i (for interactive) option.

```
am@am-UBOX  ~ $ cp SinkoDeMayo file42
am@am-UBOX  ~ $ cp SinkoDeMayo file42
am@am-UBOX  ~ $ cp -i SinkoDeMayo file42
cp: overwrite `file42'? n
am@am-UBOX  ~ $
```

Exercise:

Self practice

mv

Use mv to rename a file or to move the file to another directory.

```
am@am-UBOX  ~ $ ls
Dir33    dir42   file42   file42.copy SinkoDeMayo
am@am-UBOX  ~ $ mv file42 file33
am@am-UBOX  ~ $ ls
dir33    dir42   file33    file42.copySinkoDeMayo
```

Exercise:

Self practice

mv

The same mv command can be used to rename directories.

```
am@am-UBOX  ~ $ mv dir33 backup
```

Exercise:

Self practice

mv

-i

The mv also has a -i switch similar to cp and rm.

```
am@am-UBOX  ~ $ mv -i file33 SinkoDeMayo
mv: overwrite `SinkoDeMayo'? no
am@am-UBOX  ~ $
```

Exercise:

Self practice

rename

The rename command is one of the rare occasions where the Linux Fundamentals book has to make a distinction between Linux distributions.

Practice

1. List the files in the /bin directory

```
ls /bin
```

Practice

2. Display the type of file of /bin/cat, /etc/passwd and /usr/bin/passwd.

```
file /bin/cat /etc/passwd /usr/bin/passwd
```

Practice

3. Create a directory ~/touched and enter it.

```
mkdir ~/touched ; cd ~/touched
```

Practice

4. Create the files today.txt and yesterday.txt in touched.

```
touch today.txt yesterday.txt
```

Practice

5. Change the date on yesterday.txt to match yesterday's date.

```
touch -t 201509211405 yesterday.txt
```

Practice

6. Copy yesterday.txt to copy.yesterday.txt

```
cp yesterday.txt copy.yesterday.txt
```

Practice

7. Rename copy.yesterday.txt to kim

```
mv copy.yesterday.txt kim
```

Practice

8. Create a directory called ~/testbackup and copy all files from ~/touched into it.

```
mkdir ~/testbackup ; cp -r ~/touched ~/testbackup/
```

Practice

9. Use one command to remove the directory ~/testbackup and all files into it.

```
rm -rf ~/testbackup
```

Practice

11. Create a directory ~/etcbackup and copy all *.conf files from /etc into it. Did you include all subdirectories of /etc ?

```
cp -r /etc/*.conf ~/etcbackup
```

head

You can use head to display the first ten lines of a file.

```
am@am-UBOX  ~ $ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
```

head -c

The head command can also display the first n lines of a file.

```
am@am-UBOX  ~ $ head -c14 /etc/passwd
root:x:0:0:roo am@am-UBOX  ~ $
```

head

And head can also display the first n bytes.

```
am@am-UBOX  ~ $ head -4 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
```

tail (-n)

Similar to head, the tail command will display the last ten lines of a file.

```
am@am-UBOX  ~ $ head -4 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
```

cat

The cat command is one of the most universal tools, yet all it does is copy standard input to standard output. First, you can use cat to display a file on the screen

```
am@am-UBOX  ~ $ cat /etc/resolv.conf
domain linux-training.be
search linux-training.be
nameserver 192.168.1.42
```

cat

cat is short for concatenate. One of the basic uses of cat is to concatenate files into a bigger (or complete) file.

```
am@am-UBOX  ~ $ cat
/etc/resolv.conf
am@am-UBOX  ~ $ echo one
>part1
am@am-UBOX  ~ $ echo two
>part2
am@am-UBOX  ~ $ echo three
>part3
am@am-UBOX  ~ $ cat part1
one
am@am-UBOX  ~ $ cat part2
two
am@am-UBOX  ~ $ cat part3
three
am@am-UBOX  ~ $ cat part1 part2
part3
one
two
three
am@am-UBOX  ~ $ cat part1 part2
```

Cat >

You can use cat to create flat text files.

The Ctrl d key combination will send an EOF (End of File) to the running process ending

```
am@am-UBOX  ~ $ cat > winter.txt
It is very cold today!
am@am-UBOX  ~ $ cat winter.txt
It is very cold today!
am@am-UBOX  ~ $
```

Cat > ...

You can choose an end marker for cat with << as is shown in this screenshot. This construction is called a here directive and will end the cat

```
am@am-UBOX  ~ $ cat > hot.txt <<stop
> It is hot today!
> Yes it is summer.
> stop
am@am-UBOX  ~ $ cat hot.txt
It is hot today!
Yes it is summer.
```

cat

In the third example you will see that cat can be used to copy files.

```
am@am-UBOX  ~ $  cat winter.txt
It is very cold today!
am@am-UBOX  ~ $ cat winter.txt > cold.txt
am@am-UBOX  ~ $ cat cold.txt
It is very cold today!
```

more, less

The more command is useful for displaying files that take up more than one screen. More will allow you to see the contents of the file page by page. Use the space bar to see the next page, or q to quit. Some people prefer the less command to more.

String

With the strings command you can display readable ascii strings found in (binary) files.
This example locates the ls binary then displays readable strings in the binary file (output is truncated).

Practice - working with files

1. Display the first 12 lines of /etc/services.

```
head -12 /etc/services
```

Practice

2- Display the last line of /etc/passwd.

```
tail -1 /etc/passwd
```

Practice

3- Use cat to create a file named count.txt that looks like this:

```
cat > count.txt
One
Two
Three
Four
Five (followed by Ctrl-d)
```

Practice

4. Use cp to make a backup of this file to cnt.txt.

```
cp count.txt cnt.txt
```

Practice

5. Use cat to make a backup of this file to catcnt.txt.

```
cat count.txt > catcnt.txt
```

Practice

6. Display catcnt.txt, but with all lines in reverse order (the last line first).

```
tac catcnt.txt
```

Practice

7. Use more to display /etc/services.

```
more /etc/services
```

Practice

8. Display the readable character strings from the /usr/bin/passwd command.

```
strings /usr/bin/passwd
```

Practice

9. Use ls to find the biggest file in /etc.

```
ls -lrS /etc
```

Practice

10. Use cat to create a file named tailing.txt that contains the contents of tailing.txt followed
by the contents of /etc/passwd.

```
cat /etc/passwd >> tailing.txt
```

Practice

7. Rename copy.yesterday.txt to kim

```
mv copy.yesterday.txt kim
```

Practice

12. Use cat to create a file named tailing.txt that contains the contents of tailing.txt preceded
by the contents of /etc/passwd.

```
mv tailing.txt tmp.txt ; cat /etc/passwd tmp.txt > tailing.txt
```

Linux file system hierarchy

‘/’ root

/bin

The /bin directory contains binaries for use by all users.

/sbin

/sbin contains binaries to configure the operating system. Many of the system binaries
require root privilege to perform certain tasks.

/lib

Binaries found in /bin and /sbin often use shared libraries located in /lib. Below is a screenshot of the partial contents of /lib.

/lib/modules

Typically, the Linux kernel loads kernel modules from /lib/modules/$kernel-version/.
This directory is discussed in detail in the Linux kernel chapter.

/opt

The purpose of /opt is to store optional software. In many cases this is software from outside the distribution repository. You may find an empty /opt directory on many systems.

/boot

The /boot directory contains all files needed to boot the computer. These files don't change
very often.

/etc

All of the machine-specific configuration files should be located in /etc. Historically /etc stood for etcetera, today people often use the Editable Text Configuration backronym.

/home

Users can store personal or project data under /home. It is common (but not mandatory by
the fhs) practice to name the users home directory after the user name in the format /home/$USERNAME.

/root

On many systems /root is the default location for personal data and profile of the root user.
If it does not exist by default, then some administrators create it.

/srv

You may use /srv for data that is served by your system. The FHS allows locating cvs,
rsync, ftp and www data in this location.

/media

The /media directory serves as a mount point for removable media devices such as CD-ROM's, digital cameras, and various usb attached devices. Since /media is rather new in the Unix world, you could very well encounter systems running without this directory.

/mnt

The /mnt directory should be empty and should only be used for temporary mount points
(according to the FHS).

/tmp

Applications and users should use /tmp to store temporary data when needed. Data stored
in /tmp may use either disk space or RAM.