Unit 11: LINUX

Linux is an open source, UNIX-like operating system kernel designed to work on many different platforms. It has an active community developing it, so many distros (Linux distributions) are constantly evolving. It is impossible to count the available distos to try out, as people are free to modify existing source code and compile their own, though a quick search for "Linux distros" on *DistroWatch*, a popular database of open source distros, returns over 800 results.

The most common distros are a particular flavour of Linux, called GNU/Linux, which is a combination of the Linux kernel and an assortment software from the GNU project, a collaborative development project with the goal to give people the freedom and encourage them to modify and share software. Richard Stallman, the founder of this project, wrote the first GNU General Public License (GPL) which enforces the aims of the the project by requiring that any adaptations and redistributions are bound by the same license. Licenses with this ruling are given the name *Copyleft*, as they impose restrictions which are protected by copyright law, but still allow for the modification and sharing of the work, something that is normally prohibited outside of "fair-use" for most copyrighted material.

Open Source

Open source is a term used to describe software that has publicly available source code, meaning that developers/enthusiastasts can modify and compile the package themselves. Software licensed under an open source license can usually be modified and redistributed by anyone who has acquired a copy of the software.

Here is the definition currently on the OSI's (Open Source Initiative) website:

> Generally, open source software is software that can be freely accessed, used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the The Open Source Definition.

Though Linux is not OSI licensed, it is licensed under GPL v2, a license written by the Free Software Foundation (FSF), an organisation also founded by Richard Stallman.

Using Linux

Linux is praised for having the ability to be incredibly lightweight, in the way that you can find fully-usable distributions that only use a minimal amount of space (*Damn Small Linux (DSL) is a distro of only 50MB*) and in the sense that it doesn't have to consume a lot of computer resources. Some Linux distros don't even include a GUI (Graphical user interface), and are completely CLI-based (Command-line interface), which means that they are controlled entirely by typing text-based commands and writing scripts. Even for distros with a user-friendly graphical interface, in many cases people find it easier to do certain tasks using commands, so terminal emulators exist in virtually every GUI-based distro so that you can still use the command-line interface. An example of a Linux installation using the MATE desktop environment is pictured below.

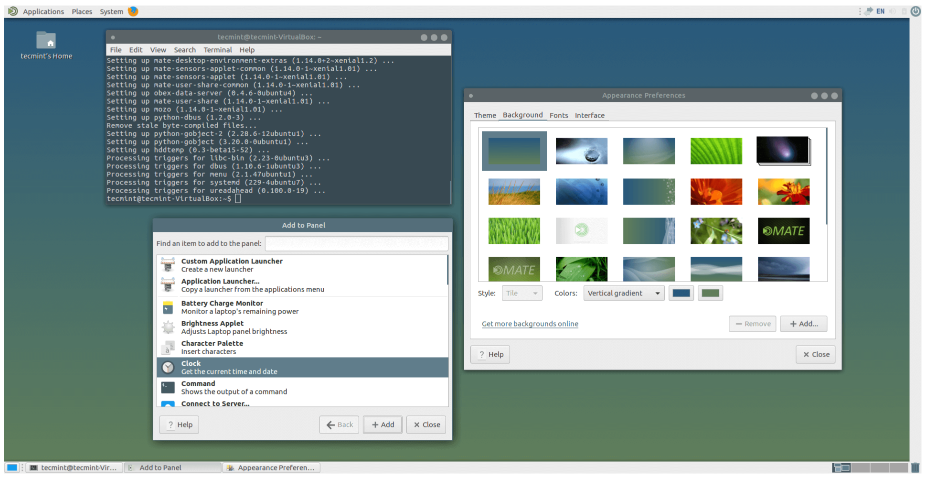
```
Setting up mate-desktop-environment-extras (1.14.0+2~xenial1.2) ...
Setting up mate-sensors-applet-common (1.14.0-1~xenial1.01) ...
Setting up mate-sensors-applet (1.14.0-1~xenial1.01) ...
Setting up mate-user-share-common (1.14.0-1~xenial1.01) ...
Setting up obex-data-server (0.4.6-0ubuntu4) ...
Setting up mate-user-share (1.14.0-1~xenial1.01) ...
Setting up mozo (1.14.0-1~xenial1.01) ...
Setting up python-dbus (1.2.0-3) ...
Remove stale byte-compiled files...
Setting up python-gobject-2 (2.28.6-12ubuntu1) ...
Setting up python-gobject (3.20.0-0ubuntu1) ...
Setting up hddtemp (0.3-beta15-52) ...
Processing triggers for libc-bin (2.23-0ubuntu3) ...
Processing triggers for dbus (1.10.6-1ubuntu3) ...
Processing triggers for menu (2.1.47ubuntu1) ...
Processing triggers for systemd (229-4ubuntu7) ...
Processing triggers for ureadahead (0.100.0-19) ...
tecmint@tecmint-VirtualBox:~$
```
Applications
Places
System
tecmint's Home
tecmint@tecmint-VirtualBox: ~
File Edit View Search Terminal Help
Add to Panel
Find an item to add to the panel:
Custom Application Launcher
Create a new launcher
Application Launcher...
Copy a launcher from the applications menu
Battery Charge Monitor
Monitor a laptop's remaining power
Brightness Applet
Adjusts Laptop panel brightness
Character Palette
Insert characters
Clock
Get the current time and date
Command
Shows the output of a command
Connect to Server...
Help
Back
Add
Close
Appearance Preferences
Theme Background Fonts Interface
Style: Tile
Colors: Vertical gradient
Get more backgrounds online
Remove
Add...
Help
Close
tecmint@tecmint-Vir...
Add to Panel
Appearance Preferen...

Figure 1: Linux running the MATE desktop environment

Useful commands

Here I wont explain everything that these commands can do as there is already enough documentation out there available to read. But I will cover the basic commands I use on a daily basis to get around Linux.

cd

Changes the Current Working Directory (CWD) to the given path.

ls

Lists files in the given path (or ./ by default). Using the `-a` argument also shows hidden files and the `-l` arguement shows files in a list format. Other useful arguements include:

- `-h`: Shows human-readable file sizes (rather than diplaying file sizes in bytes)
- `-d`: Shows directories only

- `-s`: Sorts files by filesize
- `-R`: lists files recursively, meaning that it lists files that are contained in any subfolders of the directory.

For example:

```
~ >>> cd Downloads/
~/Downloads >>> ls
file1.txt  file2.mp4  my_book.epub
~/Downloads >>> ls -a
.  ..  .hidden_file  file1.txt  file2.mp4  my_book.epub
~/Downloads >>> ls -l
total 0
-rw-rw-r-- 1 sysadmin sysadmin 0 Mar  1 20:00 file1.txt
-rw-rw-r-- 1 sysadmin sysadmin 0 Mar  1 20:00 file2.mp4
-rw-rw-r-- 1 sysadmin sysadmin 0 Mar  1 20:00 my_book.epub
```

Note: on Linux, `~` *means your home directory,* `.` *is the current directory, and* `..` *refers to the parent directory*

touch

Creates a new file with the given name.

```
~/Downloads >>> touch new_file.md
~/Downloads >>> ls
file1.txt  file2.mp4  my_book.epub  new_file.md
```

mkdir

Creates a directory, given a name and a desired path.

```
~/Downloads >>> cd ..
~ >>> ls
Desktop  Downloads
~ >>> mkdir ./my_folder
~ >>> ls
Desktop  Downloads  my_folder
```

cp

Copies a given file to the given destination. If you are copying a directory, you can use the `-r` flag to recusively copy the contents too.

```
~ >>> cp ./Downloads/new_file.md ./copied_file.md
~ >>> ls
Desktop  Downloads  my_folder  copied_file.md
```

mv

The same as cp but deletes the original file. You can technically use it to rename a file too.

```
~ >>> mv copied_file.md my_folder/moved_file.md
~ >>> ls
Desktop  Downloads  my_folder
~ >>> ls my_folder
moved_file.md
```

ln

This command is used to create file links, which are similar to what shorcuts are in the Windows OS. There are type types of links you can create with this command: Hard links (the default option) which all point to the same file on disk, whereas soft links (sometimes called symbolic links, created with the `-s` argument), effectively create a new file which is a pointer to the original file.

```
~ >>> ln  my_file.md ./Downloads -s
~ >>> ls -l /Downloads
-rw-rw-r-- 2 sysadmin sysadmin    0 Mar  6 20:50 my_file.md -> my_file.md
```

rm

Removes (deletes) a given file. The `-r` (recursive) flag can be used to remove directories.

```
~ >>> rm -r my_folder
~ >>> ls
Desktop  Downloads
```

alias

Assigns a given command/script to a given alias.

```
~ >>> alias m mkdir
~ >>> m my_new_folder
~ >>> ls
Desktop  Downloads  my_new_folder
```

tree

This script looks recursively through child folders and draws an easy to read folder structure diagram.

```
~ >>> tree .
.
|-- Desktop
|-- Downloads
|   |-- file1.txt
|   |-- file2.mp4
|   `-- my_book.epub
`-- my_new_folder

3 directories, 3 files
```

find

Searches recursively from the given directory for list of files that satisfy a given query, and can execute them all, given the `-exec` flag. One useful way to find files is by name. So by using the `-name` flag, you can pass the script a pattern to look for in the names of files. You can also find files by date, size, owner and many other attributes.

```
~ >>> find . -name "*book*"
./Downloads/my_book.epub
~ >>> find . -name "*b*"
./.bash_logout
./.bashrc
./Downloads/my_book.epub
```

cat

`cat`, short for concatenate, displays the contents of text files.

```
~ >>> cat Downloads/file1.txt
one two
three four
five six
seven eight
dog can
six elephants
lamp keyboard
```

grep

Returns a list of lines, given an input, that match a given pattern. It searches for lines inside given files, so in combination with find, you can easily search recursively for files containing specific strings, which can be very useful.

```
~ >>> cat Downloads/file1.txt | grep six
five six
```

```
six elephants
~ >>> grep six Downloads/file1.txt
five six
six elephants
```

Note: The pipe (|) character in the example above passes the output of the first command (`cat`) to the grep command

man

Opens up a documentation page for a given command. These pages are generally not `info` is a similar command included in the GNU project, which was intended to encourage a more long-form type of documentation. It has support for hyperlinks for referencing different chapters or files from within the docs. Many commands also have a `-h` or –help arguement which shows you a quick summary of the main functions of tool.

Compressing/Archiving Files

Inactive or old data, for example logs or files from completed projects, can quickly build up on disks. So these files often need to compressed or archived, to free up some space for new data. I explained a little more about this in Unit 2.

Compression on Linux

`gzip`, is a GNU tool for lossless file compression and decompression. Lossless refers to the type of compression is uses, which means that the original uncompressed file can be reproduced exaclty from the compressed `gzip` (`gzip` is also the name of the file format). Lossy compression is when the some of the information from the original file is lost in the compression process, usually as the cost of further reducing the compressed file's size. This type of compression might not be completely suitable for plaintext files, for obvious reasons, but is very useful for compressing image data, which, in many cases, doesn't need to be so precise.

How easy it is compress a file with `gzip`:

```
~ >>> ls
Desktop  Downloads  hasans_file.txt  my_folder
~ >>> gzip hasans_file.txt
~ >>> ls
Desktop  Downloads  hasans_file.txt.gz  my_folder
~ >>> gzip -l hasans_file.txt.gz
         compressed        uncompressed  ratio uncompressed_name
               5632               47044  88.1% hasans_file.txt
```

And we can uncompress it just as easily, by using the `gunzip` (g- unzip) command:

```
~ >>> gunzip hasans_file.txt.gz
~ >>> ls
Desktop  Downloads  hasans_file.txt  my_folder
```

Linux also comes with another compression tool, called `bzip2`, which is used in the same way as `gzip`, but uses more complex algorithms, meaning that it can take a little longer to compress/uncompress data, but can also reduce it to a smaller size than is possible with `gzip`. `bunzip2` works in the same way as `gunzip` for uncompressing files. The conventional file extentions for files compressed with `bzip2` are `.bz` and `.bz2`.

Archival

Archives combines multiple individual files into one single file. This could be useful if you wanted to compress this data together, as you would only have to compress this one file, which, depending on the algorithms used, might even be more time efficient than compressing them seperately. The standard UNIX utility for this is `tar` (short for tape-archive). In Linux there is a GNU rewrite which replaces this, also called `tar`.

You can use tar to create, extract and browse tar files (sometimes called tarballs). To create a tar file, you have to use the `-c` argument to enable create "mode" and the `-f` argument to tell it to write to a given filename.

Here I create a new archive containing my text files:

```
~/my_folder >>> ls
hasans_file.txt    hasans_file_2.txt  hasans_file_4.txt
hasans_file_1.txt  hasans_file_3.txt  hasans_file_5.txt
~/my_folder >>> tar -cf my_archive.tar ./*
~/my_folder >>> ls -lh
total 568K
-rw-rw-r-- 1 sysadmin sysadmin  46K Mar  2 18:28 hasans_file.txt
-rw-rw-r-- 1 sysadmin sysadmin  46K Mar  2 19:02 hasans_file_1.txt
-rw-rw-r-- 1 sysadmin sysadmin  46K Mar  2 19:03 hasans_file_2.txt
-rw-rw-r-- 1 sysadmin sysadmin  46K Mar  2 19:03 hasans_file_3.txt
-rw-rw-r-- 1 sysadmin sysadmin  46K Mar  2 19:03 hasans_file_4.txt
-rw-rw-r-- 1 sysadmin sysadmin  46K Mar  2 19:03 hasans_file_5.txt
-rw-rw-r-- 1 sysadmin sysadmin 280K Mar  2 19:32 my_archive.tar
```

You can see here that the archive is equal (roughly) to the sum of the files it was created from's individual sizes. This is because this archive is completely uncompressed. We could use the `gzip` command if we wanted to compress it, or we could use `tar`'s `-z` flag, which creates a gzip-compressed tar archive. The `-j` argument exists for `bzip2` compression too. The convention for compressed individual files is to append the `.gz`/`.bz` file extension to the end of the file. Let me show you an example:

```
~/my_folder >>> tar -czf my_archive.tar.gz ./*
~/my_folder >>> ls -lh my_archive.tar.gz
-rw-rw-r-- 1 sysadmin sysadmin 8.2K Mar  2 19:49 my_archive.tar.gz
```

This created a vastly smaller filer as a result. The `gzip`-compressed tar is 34 times smaller than the original, the only tradeoff being that you have to uncompress it before you can read it, which is not a major issue if these are files aren't expecting much use. You can uncompress and read them with the `tar` command too, with the `-t` flag to enable "browse files" mode and the `-z` flag again to uncompress the data first with `gunzip`:

```
~/my_folder >>> tar -tzf my_archive.tar.gz
./hasans_file.txt
./hasans_file_1.txt
./hasans_file_2.txt
./hasans_file_3.txt
./hasans_file_4.txt
./hasans_file_5.txt
```

Security

Linux adopts UNIX's "Everything is a file" feature, meaning that data from devices and periphirals such as printers, keyboards, disk drives and networked machines are all accessible via the file system. This is done by using virtual filesystems which are all mounted so that they are visible as one file heirarchy.

Users/Groups

User and group permissions are used to control access in Linux. To create a new user, you can use the `useradd` and `passwd` commands:

```
~ >>> useradd hasan -d /home/hasan
```

The `-d` flag sets the home (`~`) directory.

```
~ >>> passwd hasan
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Creating new groups and adding a user to them is just as simple, using the `groupadd` and `usermod` commands:

```
~ >>> useradd
~ >>> groupadd gods
~ >>> groupadd kings
~ >>> usermod -G gods,kings hasan
```

And, finally, we can see what groups a user is assigned to by using the `id` command.

```
~ >>> id hasan
uid=1002(hasan) gid=0(root) groups=1002(gods),1003(kings)
```

The default user, `root`, automatically has access to every file, with no regard for permissions.

File Permissions

Every file in Linux file system has its own properties, one of which being its own permission levels, which is used to restrict access to certain files and directories. There are three types of access you can to a file: read, write and execute. An easy way to see a file's attributes is to use the command `ls`-l"""".

```
~ >>> ls -lh
-rw-r--r-- 1 root root 2.3K Mar  5 20:49 hasans_medical_records.csv
```

Here we can see multiple columns each with different data about the file. In order, these columns describe, in this order:

- Filetype + Permission level.
- Number of hard links pointing to file.
- File's owner.
- File's group.
- File size.
- Last edit: date and time.

The file type + permission level is not immediately intuitive, so I will explain what it means. The first character dictates the type of file that this is. In most cases, this will be either `-`, for standard files, `d` for directories and `l` for links. The second part is made up of nine permission bits, with 3 bits controlling read, write and execute access for the file owner, group and other users.

There are two common ways for these permissions to be written. One, as you can see in the `ls` examples above, is to use `r`, `w` and `x` (read, write and execute) in place of their respective bits to show that the access that has been granted, and `-` in place of bits that are not enabled. With this example, a file with `rwxr--r--` has read, write and execute access for the owner, but anyone else will only have read access. Another way to write these permissions would be to use a number in place of each set of 3 bits (`rwx`). 111, in binary, is equal to 7 in denary, so we can use the digits 0-7 to represent every possible `rwx` combination. With this logic, 220 would equal `-w--w----` and 775 would equal `rwxrwxr-x`.

The `chmod` command can be used to modify a files permissions, if you happen to have write access to the file. The 0-7 digit method is the easiest method to use with this command.

```
~ >>> chmod 000
~ >>> ls -l
---------- 1 root root 2.3K Mar  5 20:49 hasans_medical_records.csv
~ >>> chmod 777
~ >>> ls -l
-rwxrwxrwx 1 root root 2.3K Mar  5 20:49 hasans_medical_records.csv
```