

A Basic Linux Workshop

By: Ziad Matni, UCSB

CONTENT

1. Directory structures in Linux
 - a. Absolute paths vs. Relative paths
 - b. `.` vs `..`

2. Linux file management commands
 - a. **man**
 - b. **cd**
 - c. **pwd**
 - d. **ls** and its options
 - e. **cp** and its options
 - f. **rm** and its options
 - g. **mv**
 - h. **du**

3. Linux text file manipulation commands
 - a. **cat**
 - b. **more** and **less**
 - c. **wc** and its options
 - d. Directing output with `>` and `>>`
 - e. Doing multiple commands at once using `;`
 - f. **head** and **tail**

4. Linux networking commands
 - a. **ssh**
 - b. **scp**

A Basic Linux Workshop

1. Directory Structures in Linux

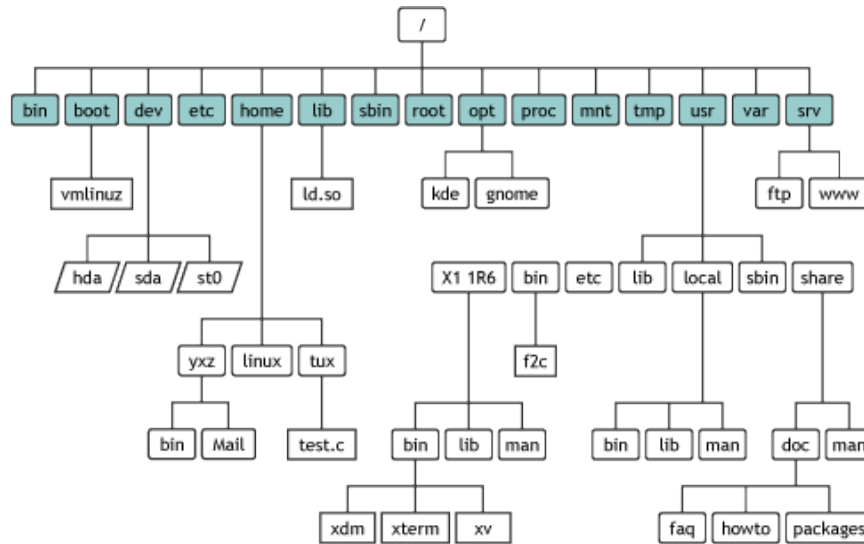


Figure 1: An example of a Linux directory tree

Files are logical collections of data stored in a computer’s secondary storage (i.e. its hard drive, or on a “thumb” drive, etc.). **Directories** (also called **folders**) are constructs where multiple files can reside. Other directories can also be found inside directories.

Think of the organization of Linux directories as a tree with the aptly named **root directory** (symbol: /).

In this directory reside other directories (sometimes called **sub-directories** or **sub-folders**) or stand-alone files. These, in turn, also have directories and files “beneath” them.

We sometimes call these “**child**” and the directories above them “**parent**” directories.

Absolute pathnames

When referring to the location of a file or directory, it is useful to “spell out” the entire “path” to them, starting from the root directory. So, for example, in **Figure 1**, there is a file depicted in the directory tree labeled “**test.c**”. The absolute pathname of this file would be:

`/home/tux/test.c`

A Basic Linux Workshop

Regardless of where a file or a directory resides, an absolute pathname will tell us where it is based on the ultimate reference: the root directory. The example above tells us that, from the root directory, you should go into the **home** folder, then the **tux** folder, which is where you will find the file **test.c**.

If your absolute pathname refers to a *folder*, then the convention is to add a final `/` to it. For example, if referring to the aforementioned folder “**tux**”, you’d say:

`/home/tux/`

Relative pathnames

When you want to indicate the location of a file or a folder *relative* to where you are in a directory, you can use the symbols `.` (for current directory) or `..` (for parent directory – i.e. the directory “right above” where I am).

So if you were in a folder `bin` inside `xyz`, which is in the home directory (i.e. in **`/home/xyz/bin/`**) and wanted to refer to the aforementioned **test.c** file in a relative way, you’d say:

`../../tux/test.c`

That is, go “up” to the parent directory from where I am now, and then do that again, then go down the “`tux`” directory and that’s where you’ll find “`test.c`”.

If you wanted to execute a binary file (e.g. an already compiled file), you often need to specify where that file is, even if it resides in the current directory (i.e. where you currently are). For example, a binary file called “**program**” can be executed using:

`./program`

A Basic Linux Workshop

2. Linux file management commands

In Linux the following commands are some of the most commonly used to manage files and folders. Here's a handy reference. The \$ symbol indicates a Linux prompt.

a) **man** (show manual)

We'll start with this command because you can use it to learn about all other commands!

\$ *man linux_command* show me the manual for *linux_command*

b) **cd** (change directory)

\$ *cd* change directory back to */home*

\$ *cd directory_name* change directory to *directory_name*

\$ *cd path_name* change directory to *path_name*

\$ *cd ..* change directory to the parent directory

c) **pwd** (print working directory)

\$ *pwd* tell me where I am (absolute pathname)

d) **ls** and its options (list)

\$ *ls* list this directory's files and folders

\$ *ls -a* list, incl. hidden files and folders

\$ *ls -l* list, in "long" format (showing more details)

\$ *ls -al* list, in "long" format, incl. hidden stuff

\$ *ls -lrt* list, in "long" format, sorted by modified time-stamp, in reverse order

Many other options exist for *ls*. Use *man ls* for more info.

e) **cp** and its options (copy file)

\$ *cp file_s file_d* create a copy of file *file_s* and call it *file_d*

\$ *cp -r dir_s dir_d* create a full copy of directory *dir_s* and call it *dir_d*

See *man cp*

A Basic Linux Workshop

f) **rm** and its options (remove – or delete – file)

\$ **rm** *file* remove the file

\$ **rm -r** *dir* remove the directory

See *man rm*

g) **mv** (move – or rename – file)

\$ **mv** *file_or_dir* move file or directory

See *man mv*

h) **du** (check disk use)

\$ **du -h** *file_or_dir* check file or directory size

See *man du*

3. Linux text file manipulation commands

a) **cat** (concatenate file)

\$ **cat** *file* print to screen the contents of the text file

(no stopping)

See *man cat*

b) **more**

\$ **more** *file* print to screen the contents of the text file

(stops at each screen length and waits for user to press the space bar)

See *man more* and *man less* (a similar command)

c) **wc** and its options (word count)

\$ **wc** *file* print the number of words in the text file

\$ **wc -l** *file* print the number of lines in the text file

See *man wc*

A Basic Linux Workshop

d) Directing output with >

You can direct the outcome of a Linux command to a text file (either from scratch or as an append to an existing one). For example, if you did:

```
$ ls -l > longlist.txt
```

You'd create a new text file called "**longlist.txt**" that would contain the long format of the list of the current directory.

To append to the end of a file, instead of creating a new one, use >> instead of >

e) Multiple commands using ;

You can do multiple commands at once using ;

For example:

```
$ ls ; more fileABC.txt
```

This will list the current directory and then issue a more command on the text file **fileABC.txt**.

f) **head** and **tail** and its options

```
$ head -n 10 file      print the first 10 lines in the text file
```

```
$ head -c 10 file     print the first 10 bytes in the text file
```

```
$ tail -n 10 file     print the last 10 lines in the text file
```

```
$ tail -c 10 file     print the last 10 bytes in the text file
```

See man head; man tail

A Basic Linux Workshop

4. Linux networking commands – or – How to Access Your CSIL Account From a Terminal Program

Terminal programs can be found natively on UNIX/Linux OS machines, the Mac OS (called “Terminal”) and on Windows 10 (not earlier versions, plus you have to activate this program first, called the “bash” terminal).

Your computer is called “local” and the CSIL computer (or any other computer that’s not yours and that you want to connect to) is called “remote”.

a) Logging in

- 1) From a terminal prompt, type: `$ ssh yourname@csil-XX.cs.ucsb.edu`
where *yourname* is, of course, your CSIL/SoE user name and *XX* is a 2-digit number between 01 and 48 (pick any one). Be careful with spelling mistakes!
- 2) You will be asked to put in your password, after which you will be logged in.

b) “Remote” copying a file from your CSIL folders to your own computer’s hard drive

- 1) Do all of this from your computer (i.e. don’t do the ssh first).
- 2) From a terminal prompt, prepare for where on your computer’s local drive you want to copy to the remote CSIL folder.

```
$ cd local_directory
```

Where *local_directory* is the location, for example: Desktop/csFolder, or
C:/BobbySue/MyCS_Stuff

A Basic Linux Workshop

3) Then type:

```
$ scp yourname@csil-XX.cs.ucsb.edu:~/remote_directory/filename .
```

Note the ~ character and the dot (.) at the end of the command – you must have those in there.

“remote_directory/filename” is called the directory path to your filename. For example, it could be cs8/lab5/program.py or cs16/lab2/boxes.cpp.

For example:

```
$ scp jimbo@csil-11.cs.ucsb.edu:~/cs16/lab03/change.cpp .
```

c) “Remote” copying a file to your CSIL folders from your own computer’s hard drive

- 1) Do all of this from your computer (i.e. don’t do the ssh first).
- 2) From a terminal prompt, prepare for where on your computer’s drive is the local file that you want to copy the remote CSIL folder.

```
$ cd local_directory
```

Where *local_directory* is the location of the file you want to copy over, for example: Desktop/csFolder, or C:/BobbySue/MyCS_Stuff

3) Then type:

```
$ scp local_directory/filename
```

```
yourname@csil-XX.cs.ucsb.edu:~/remote_directory/
```

“remote_directory” is the remote directory for where you want to copy your filename to. For example, it could be cs8/lab5 or cs16/lab2. Again, **XX** is a 2-digit number between 01 and 48 (pick any one).

For example:

```
$ scp ./myPrograms/change.cpp jimbo@csil-11.cs.ucsb.edu:~/cs16/lab03/
```