Linux Essentials

Programming and Data Structures Lab

M Tech CS – First Year, First Semester

Adapted from PDS Lab 2014 and 2015

Login, Logout, Password

```
$ ssh mtc16XX@192.168.---.---
$ ssh –X mtc16XX@192.168.---.---
```

Secure shell login to your account on the server

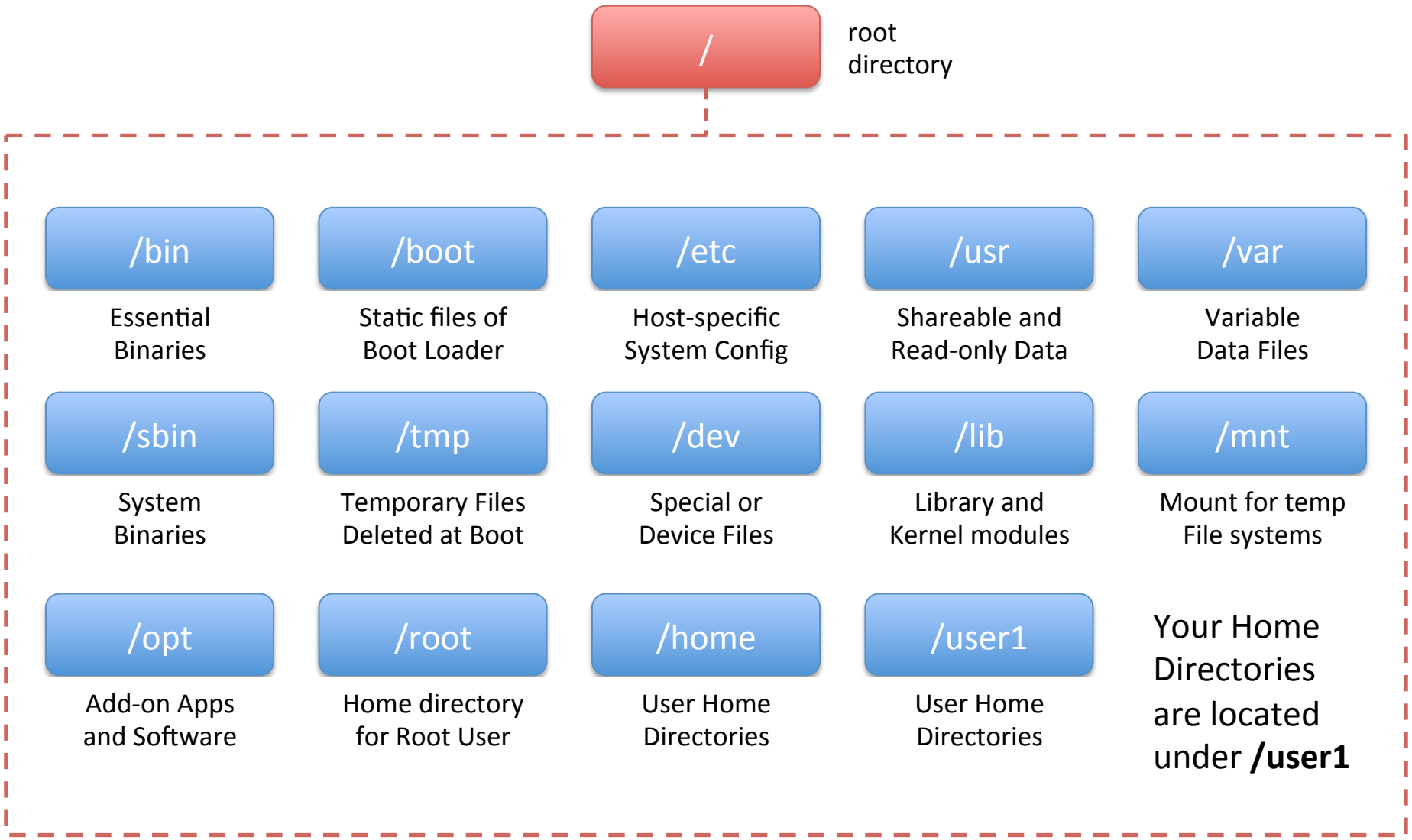
```
$ logout, exit, ^d
```

Logout from your account

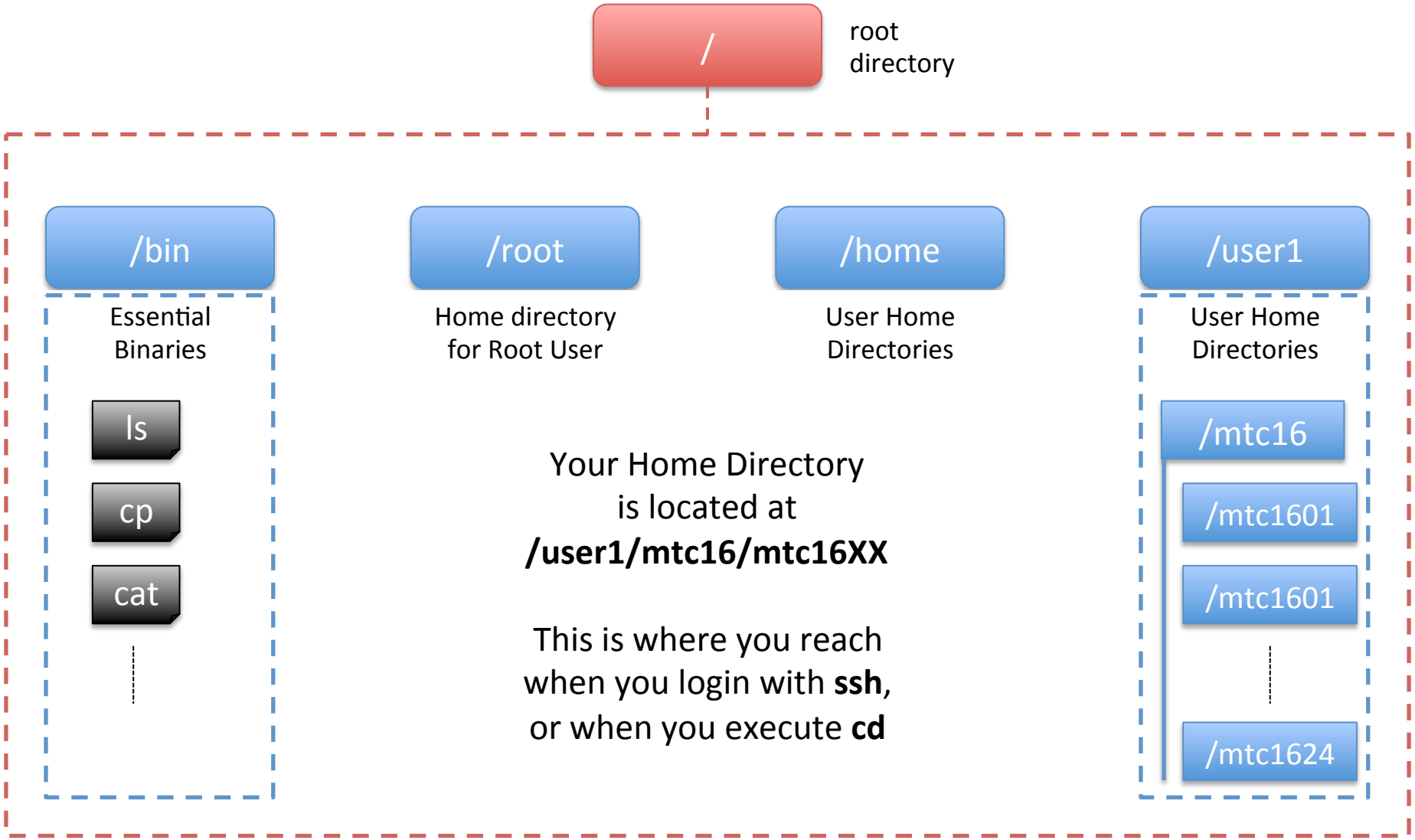
```
$ passwd or yppasswd
```

Change password of your account

Files and Directories



Files and Directories



Create and Remove Directories

```
$ mkdir dfslab
```
MAKE the directory dfslab under your home

```
$ rmdir dfslab
```
REMOVE the directory dfslab from your home
You cannot REMOVE a non-empty directory

```
$ touch hello.txt
```
CREATE the file "hello.txt" under your home – touch does not put any content

```
$ rm hello.txt
```
REMOVE the file – avoid if possible, unless you are really sure about your decision

```
$ rm –i, rm –f
```
REMOVE interactively, or by FORCE – it is way safer to use the interactive mode

Files and Directories

```
$ mkdir dfslab
```
Create directory dfslab and then the directory structure as on the website

```
$ pwd
```
Present working directory

```
dfslab/…
```
Relative path to your dfslab directory from home

```
/user1/mtc16/mtc16XX/dfslab/…
```
Absolute path to your dfslab directory

```
~/dfslab/…
```
~ acts a shorthand for your home at /user1/mtc16/mtc16XX

Files and Directories

`.`

The current directory

`..`

The parent directory

Use of relative path

```
$ cd ~/dfslab/pds20160720
```

Check what happens if you execute

```
$ cd pds20160720
$ cd ./pds20160720
$ cd ../pds20160720
```

Files and Directories

```
$ ls
```

List the contents of the current directory

```
$ ls -l
Total 11
drwxr-xr-x    8 deb   staff    272 Jul 11 11:34 Applications
drwxr-xr-x    5 deb   staff    170 Jun 26 23:23 Backup
drwx------+   4 deb   staff    136 Jul 15 11:04 Desktop
drwx------+   6 deb   staff    204 Jun 30 13:25 Documents
drwx------+  46 deb   staff   1564 Jul 31 12:42 Downloads
drwx------@  17 deb   staff    578 Jul 31 13:23 Dropbox
drwx------@   8 deb   staff    272 Jul 30 12:07 Google Drive
drwx------@  54 deb   staff   1836 Jul 21 12:48 Library
drwxrwxrwx   23 deb   staff    782 Jul 26 17:23 Softwares
drwx------+   3 deb   staff    102 Jun 26 13:04 Movies
drwx------+   3 deb   staff    102 Jun 26 13:04 Music
```

access permissions | owner | group | last modified on | file or directory name

Permissions

```
drwxr-xr-x   4 mtc1601  501   23 Jul 20 15:30 dfslab
-rwxr--r--   1 mtc1601  501   12 Jun 20 15:50 hello.txt
```

There are 3 entities – Owner, Group and Others
There are 3 permissions – Read, Write and Execute
Thus, there are 9 bits to control the permissions

`drwxr-xr-x`	`-rwxr--r--`
d denotes Directory	*- denotes File*
Owner rwx 111 = 7	*Owner rwx 111 = 7*
Group r-x 101 = 5	*Group r-- 100 = 4*
Other r-x 101 = 5	*Other r-- 100 = 4*

```
$ chmod 700 dfslab
```

Change the permission of dfslab to 700 from 755 – prevents access of anyone else
Common permissions – 755 (directories), 744 (files), 700

Read Contents of a File

```
$ more hello.txt
$ less hello.txt
```
Shows the contents of the file -- less is less work ☺

```
$ cat hello.txt
```
Concatenate – prints the whole content onto the screen

```
$ head hello.txt
$ head –n 7 hello.txt
```
Head of the file – default first 10 lines – your may specify

```
$ tail hello.txt
$ tail –n 7 hello.txt
```
Tail of the file – default last 10 lines – you may specify

Copy or Move a File

```
$ cp hello.txt helloCopy.txt
```
Copy the content of the file – creates a new file

```
$ mv hello.txt helloCopy.txt
```
Rename or move the file – simply changes the pointer

```
$ cp fileName anotherFile
$ mv fileName anotherFile
```
Which one should take less time for big files?

```
$ cp fileName directoryName
```
Creates a copy the file in the directory – provided the directory exists

```
$ mv fileName directoryName
```
Moves the file to the directory – provided the directory exists

What am I executing?

```
$ ls
```

Where is this ls binary – can you find out?

```
$ which ls
/usr/bin/ls
```

/usr/bin is in the PATH – hence ls executes

```
$ prog1.o
$ ./prog1.o
```

If the current directory, that is "." (dot), is in the PATH, then the first command will work. If not, use the second one, which specifies the location of prog1.o.

Note : To ensure that you are executing the program in your current directory, typically the one that you have compiled, and not something else, specify ./

Input and Output

```
$ echo “Hi this is me!”
```
Prints “Hi this is me!” on the terminal (stdout)

```
$ echo “Hi this is me!” > test.txt
```
Redirects the output (stdout) to the file. If the file exists, it is overwritten.

```
$ ./prog1.o > output.txt 2> error.txt
```
Redirects output (stdout) to output.txt and error (stderr) to error.txt

```
$ ./prog1.o < in.txt > out.txt 2> err.txt
```
Takes input from in.txt, outputs to out.txt, and records errors in err.txt

```
$ echo “1 2 3 4 5” | ./prog1.o
```
Prints “1 2 3 4 5” and uses (pipes) that as the input for the program prog1.o

Unix Commands

Useful References:
http://www.ucs.cam.ac.uk/docs/leaflets/u5
http://mally.stanford.edu/~sr/computing/basic-unix.html
http://www.math.utah.edu/lab/unix/unix-commands.html

Unix/Bash Cheat-sheets:
https://ubuntudanmark.dk/filer/fwunixref.pdf
http://cli.learncodethehardway.org/bash_cheat_sheet.pdf

```
$ man <command>
```
System manual pages to describe the command in detail

Later ...

```
$ grep
$ find

$ diff
$ sort

$ ps
$ top

$ sed
$ awk
```