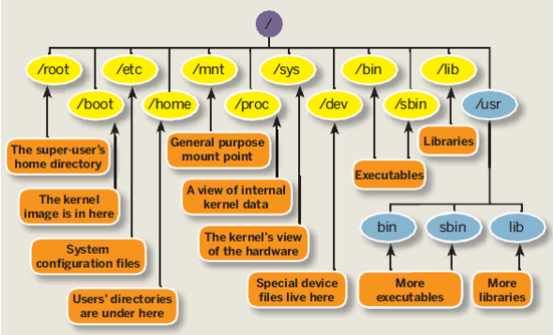
Handout: Introduction to Bash Shell

1 Linux Directory Structures

The linux filesystem looks like the following picture.



Each user has a home directory located in `/home`. The user's home directory is denoted by `/home/username` or `~`. Linux allows hidden files and directories; their names should start with a (.) dot.

2 Linux Shell

Most of us are accustomed to and comfortable with the graphical user interface of modern operating systems. Almost all popular desktop OS like Windows, Mac OS, or Linux come with different flavours of graphical user interfaces. In fact, for handheld computing devices like smartphone or tablets, graphical interfaces are part of the experience.
While the graphical interfaces are nice to look at and comfortable to work with, they are **SLOW**, even for mundane tasks. I shall try to illustrate with the following example. As there are 420 students in this module, I wish to create 420 directories for saving the assignment solutions. Unfortunately, the option of right click and selecting new directory menu for 420 times is prohibitively time consuming. Rather we shall use an interface which is much faster. *Meet the Shell.*

```
rishi@pop-os:~$ echo "hello, welcome to $SHELL"
hello, welcome to /bin/bash
rishi@pop-os:~$
```

- Open the linux terminal.
- Execute the command `echo "hello, welcome to $SHELL"` and find out the installed shell.

3 Shell Commands

In order to perform tasks the shell takes text-based commands, rather than click. Commands are programs designed to perform specific tasks. The shell comes with some basic commands. We list some of them below.

Creating directories

To create directory in the current directory: `mkdir directoryname`

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ mkdir folder1
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls
folder1
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls -l
total 4
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:37 folder1
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

Making multiple directories: `mkdir directoryname1 directoryname2`

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls
folder1
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ mkdir folder2 folder3
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls
folder1  folder2  folder3
```

Making 10 directories at one go: `mkdir directoryname1{0..9}`

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls
folder1  folder2  folder3
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ mkdir folder1{0..9}
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls
folder1   folder11  folder13  folder15  folder17  folder19  folder3
folder10  folder12  folder14  folder16  folder18  folder2
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

Navigating directories

To move inside a directory: `cd <directoryaddress>`

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ cd folder1
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples/folder1$
```

In unix terminology, the parent directory is dubbed as `..`, double dots, and the current directory is dubbed as `.`, a single dot. So, to move back to the parent directory, the command `cd ..` needs to be executed.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples/folder1$ cd ..
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

Content of a directory

The command `ls` lists the content of a directory. See the examples above. We can extract detailed information about each file/directory via the `ls -l` command.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls -l
total 16
-rw-rw-r-- 1 rishi rishi   32 Feb 12 11:03 content
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:37 folder1
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:41 folder2
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:41 folder3
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

The above command does not leak anything about the hidden files and directories though. The `ls -a` command lists all the files and directories including any hidden ones. Any files or directories that have a period at the start of their name are hidden and will not show in a GUI file browser or with `ls` without the `-a` flag. For detailed information, we should use the `ls -la` command.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls -a
.  ..  content  folder1  folder2  folder3  .this_is_a_hidden_folder
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls -la
total 28
drwxrwxr-x 6 rishi rishi 4096 Feb 16 19:44 .
drwxrwxr-x 4 rishi rishi 4096 Feb 12 10:37 ..
-rw-rw-r-- 1 rishi rishi   32 Feb 12 11:03 content
drwxrwxr-x 2 rishi rishi 4096 Feb 12 11:45 folder1
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:41 folder2
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:41 folder3
drwxrwxr-x 2 rishi rishi 4096 Feb 16 19:44 .this_is_a_hidden_folder
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

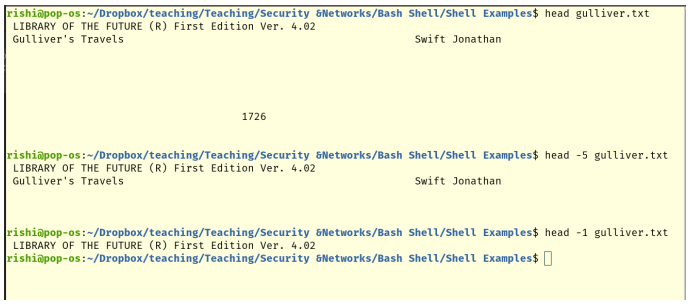
Creating Files

The simplest way to create a file is via redirection. `>` allows the output of a command to be written in a file. To append at the end of the file use `»`. See the example below.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ echo "hello"> test
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ cat test
hello
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ echo "appending a text in new line" >> test
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ cat test
hello
appending a text in new line
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

Viewing Files

As illustrated above, we can view the content of the file using the command `cat`. In fact, `cat` is arguably the most popular and simplest way to view the content of a file. However, the primary purpose of the *cat* command is to concatenate files (see `man cat` for details.) The `less` command views the file one page at a time; very useful when reading a large text. Figure 3.1, the last image of this handout, was the result of the command `less gulliver.txt`.
If we are interested in reading the first few lines (number of lines can be specified), we use the command `head`. It is very useful for extracting file headers.



Copying Files

To copy a file, use the command `cp <source> <destination>`.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ cp test copiedtest
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls
content  copiedtest  folder1  folder2  folder3  test
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ cat copiedtest
hello
appending a text in new line
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

As you may have guessed, when the source or the destination is in a different directory, we need to provide the full path (relative to current directory or /) of the file.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ cp test ./folder1/copiedtest
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls ./folder1
copiedtest
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ cat ./folder1/copiedtest
hello
appending a text in new line
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

Copying File to/from remote computer

To copy files to/from and between remote computers, use the command `scp`. The syntax of the command is similar to `cp`, except we need to provide the credentials required to access the remote computers

```
scp [[user]@source:]sourcefile [[user]@destination:]destinationfile
```

For example, consider the problem of transferring files to/from the virtual machine. Let us first figure out the ipv4 address of the installed vm. The `ip a` command gives us the ip address.

```
employee427@ics22:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP grou
p default qlen 1000
    link/ether 3a:75:4f:e5:b0:86 brd ff:ff:ff:ff:ff:ff
    inet 172.22.124.1/16 brd 172.22.255.255 scope global dynamic eth0
       valid_lft 3117sec preferred_lft 3117sec
    inet6 fe80::3875:4fff:fee5:b086/64 scope link
       valid_lft forever preferred_lft forever
employee427@ics22:~$ ls
bcprov-jdk15on-151.jar  EncTool.java  mail           Templates
CryptoDemoCode          ex1CCM.enc    Music          theFirstToken1
ctr.enc                 ex1CTR.enc    myKeyStore     theFirstToken2
Desktop                 ex1RSA.enc    myScript.java  Videos
Documents               gpg           Pictures
Downloads               index.html    Public
employee427@ics22:~$
```

Now we can download files from the virtual machine using scp. As discussed above `./` denotes the current directory.

```
jkhenes@workbook ❯ scp employee427@172.22.124.1:~/EncTool.java ./          # ~/Documents/myFolder
employee427@172.22.124.1's password:
EncTool.java                                                100%   11KB   9.2MB/s   00:00
jkhenes@workbook ❯ ls                                                     # ~/Documents/myFolder
EncTool.java  myScript.java
jkhenes@workbook ❯                                                        # ~/Documents/myFolder
```

Similarly, we can transfer a file from the local host to the vm.

```
jkhenes@workbook ❯ scp myScript.java employee427@172.22.124.1:~           # ~/Documents/myFolder
The authenticity of host '172.22.124.1 (172.22.124.1)' can't be established.
ED25519 key fingerprint is SHA256:zPJ7mXmlpjEe/7RKpxTbkhlOvJrKlAAi/rR50JWo/8Y.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:16: 192.168.64.2
    ~/.ssh/known_hosts:43: 172.22.125.162
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.22.124.1' (ED25519) to the list of known hosts.
employee427@172.22.124.1's password:
myScript.java                                               100%    0     0.0KB/s   00:00
jkhenes@workbook ❯ _                                                      # ~/Documents/myFolder
```

File Permissions

Linux considers everything as files. File permissions describe who, which, and what activities could be performed with a specific file (and whether it should be considered as a directory). Each file has an owner and a group. The operating system keeps track of permission access rights for three different classes of users.

- Owner
- Group
- Others

There are three permission types for each class of user.

- read
- write
- execute

Each file or directory has a tag of 10 characters describing the file type and these permissions. The first character denotes the file type. It could a directory (denoted by 'd'), or a regular file (denoted by '-'), a symbolic link (denoted by 'l') etc. The next nine characters list out the permissions. They are represented in three sets of three flags. The first three characters denote the read (denoted by 'r'), write (denoted by 'w'), and execute (denoted by 'x') permissions associated with the owner. The second and the third set of three characters represent the read, write, and execute permission of the group and the others. The character '-' implies the corresponding permission to be withheld.
For a concrete example, let us take another look at the screenshot illustrating the `ls -l` command.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls -l
total 16
-rw-rw-r-- 1 rishi rishi   32 Feb 12 11:03 content
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:37 folder1
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:41 folder2
drwxrwxr-x 2 rishi rishi 4096 Feb 12 10:41 folder3
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$
```

The entry for the file "content" starts with the string `-rw-rw-r-`. This shows, the os considers "content" as a regular file. The owner has the permission to read and write, but the owner can not execute. The permission is same for anyone in the group. The other users (everyone outside the group) can only read the file, they can not write to the file or execute it as a program.

Executing programs

A file is executable by a user if the corresponding execute flag is set.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ls -l
total 712
-rwxr--r-- 1 rishi rishi    472 Dec 17  2020 basic_operations.sh
-rw-rw-r-- 1 rishi rishi     32 Feb 12 11:03 content
drwxrwxr-x 2 rishi rishi   4096 Feb 12 11:45 folder1
drwxrwxr-x 2 rishi rishi   4096 Feb 12 10:41 folder2
drwxrwxr-x 2 rishi rishi   4096 Feb 12 10:41 folder3
-rw-rw-r-- 1 rishi rishi 708064 Feb 16 20:39 gulliver.txt
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell/Shell Examples$ ./basic_operations.sh
```

If we look at the permission of "basic_operations.sh" file, we see that the owner has permission to execute. To file is executed using the command `./basic_operations.sh`. Remember, **to execute a file**, or in simple words, to run a file as a program,

- Go to the folder and then write `./filename`

Conclusion

There are many more built-in commands available in the bash shell. Internet has some wonderful pages, search for "useful linux commands". To know more about a particular command, the manual-page could be of great help. You just need to type via `man command-name`. Go ahead, and try with `man scp`.

```
rishi@pop-os:~/Dropbox/teaching/Teaching/Security &Networks/Bash Shell$ man scp
```

Finally, it is worth mentioning that like modern IDEs, bash allows *tab* to autocomplete a command.

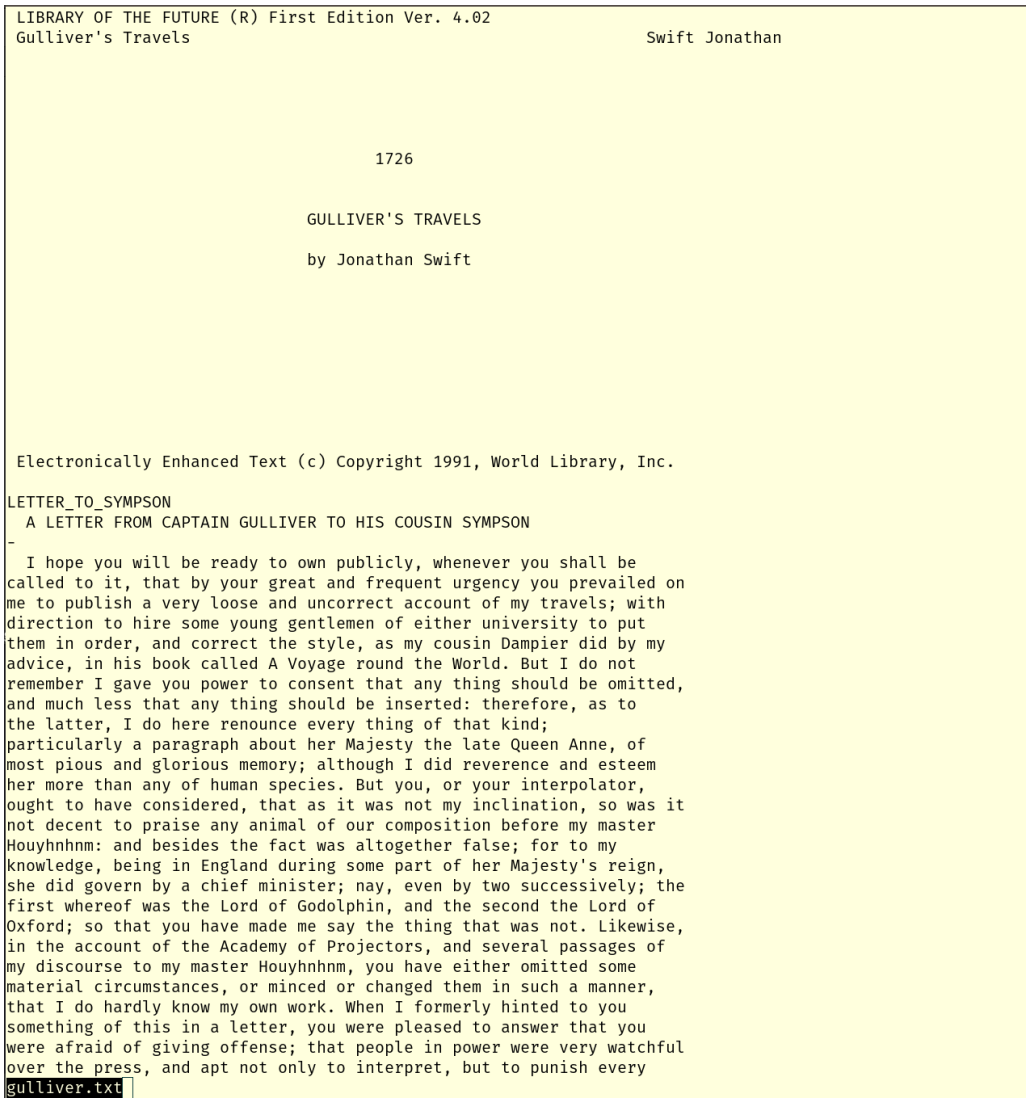


Figure 3.1: Output of `less gulliver.txt`