

## Task 1

**Task 1:** Experiment Unix commands (files directory, data manipulation, network communication etc)

**Scp:** used for copying files.

**Syntax:** \$cp [options] source\_file destination-file

Example: \$cp f1 f2

**OUTPUT:**

\$cat f1

This is GRIET

\$cat f2

This is GRIET

It will copy the contents of f1 to f2

**Options:**

a)-f: Force copy by removing the destination file if needed.

Syntax: \$cp -f source\_file destination-file

Example:\$cp -f f1 f2

OUTPUT:\$cat f1

This is CSE

\$cat f2

This is GRIET

b)-i: Ask the confirmation to overwrite.

Syntax: \$cp -i source\_file destination-file

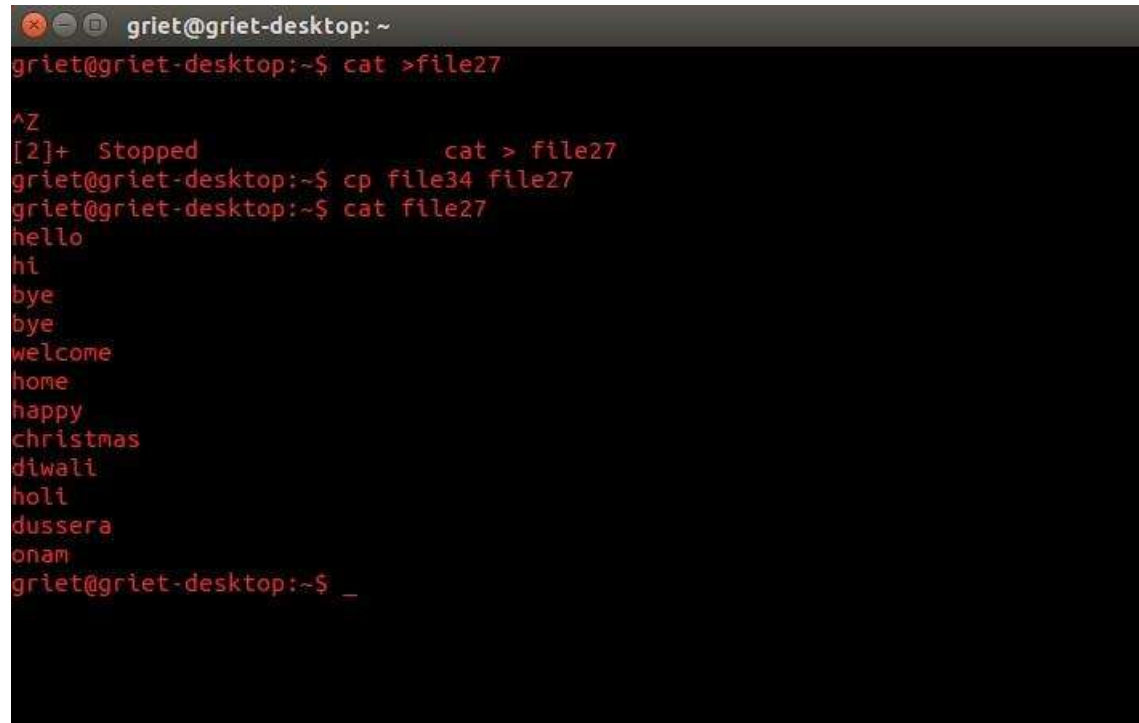
Example:\$cp -i f1 f2

c)-b:It creates backup files before overriding.

Syntax: \$cp -b source\_file destination-file

Example:\$cp -b f1 f2

**Output :**

A terminal window titled 'griet@griet-desktop: ~' with a dark background and red text. The user enters 'cat >file27', followed by a Ctrl-Z signal (^Z) and '[2]+ Stopped cat > file27'. Then, they enter 'cp file34 file27' and 'cat file27', which outputs a list of words: 'hello', 'hi', 'bye', 'bye', 'welcome', 'home', 'happy', 'christmas', 'diwali', 'holi', 'dussera', and 'onam'. The prompt returns to 'griet@griet-desktop:~\$ \_'.

**\$rm:** Used to remove files (or) directories

**Syntax:** \$rm [options] filename

Example:\$rm f1

OUTPUT:

f1 is deleted

**Options:**

a)-f: ignores non existing files, never prompt

Syntax: \$rm -f filename

Example: \$rm -f myfile.txt

OUTPUT:Removes file myfile.txt

b)-r: Removes all files in directory and directory itself

Syntax: `$rm -r filename`

Example: `$rm -r mydirectory`

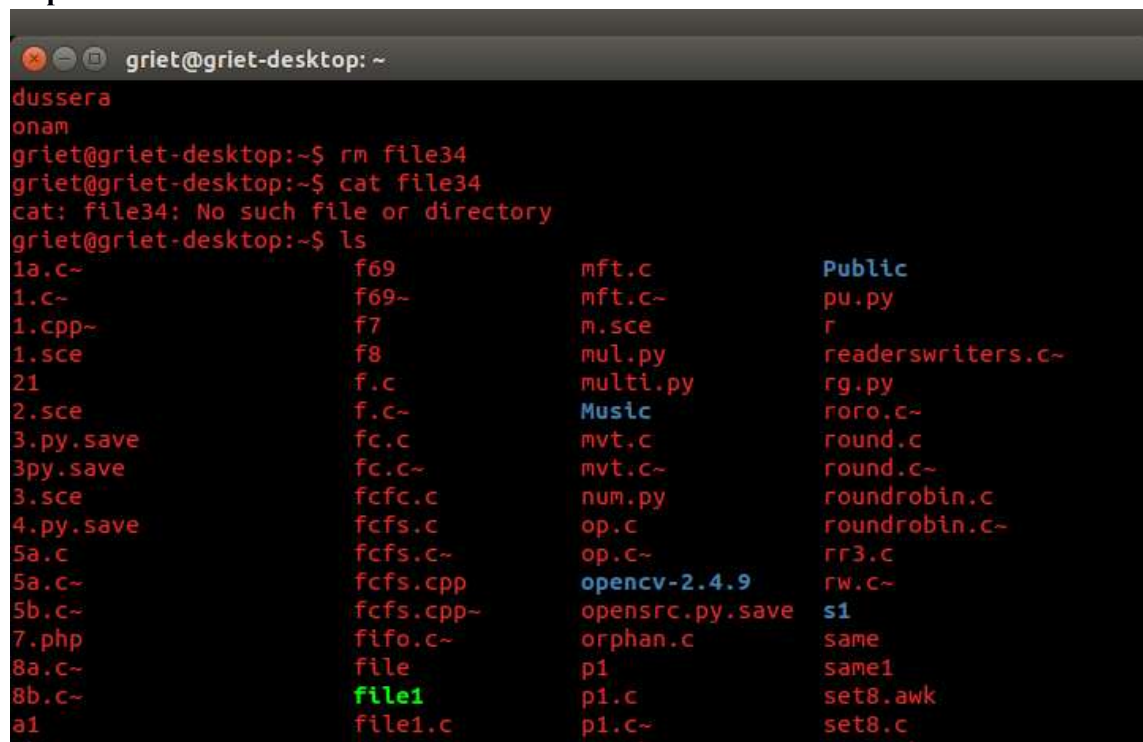
OUTPUT: Removes directory mydirectory and all files in it.

c)-i: prompts before every removal.

Syntax: `$rm -i filename`

Example: `$rm -i bak.c`

**Output:**



```
griet@griet-desktop: ~
dussera
onam
griet@griet-desktop:~$ rm file34
griet@griet-desktop:~$ cat file34
cat: file34: No such file or directory
griet@griet-desktop:~$ ls
1a.c~      f69      mft.c      Public
1.c~       f69~     mft.c~     pu.py
1.cpp~     f7       m.sce      r
1.sce      f8       mul.py     readerswriters.c~
21         f.c      multi.py   rg.py
2.sce      f.c~     Music      roro.c~
3.py.save  fc.c     mvt.c      round.c
3py.save   fc.c~    mvt.c~     round.c~
3.sce      fcfc.c   num.py     roundrobin.c
4.py.save  fcfs.c   op.c       roundrobin.c~
5a.c      fcfs.c~  op.c~      rr3.c
5a.c~     fcfs.cpp opencv-2.4.9  rw.c~
5b.c~     fcfs.cpp~ opencv.py.save s1
7.php     fifo.c~  orphan.c   same
8a.c~     file     p1         same1
8b.c~     file1    p1.c      set8.awk
a1        file1.c  p1.c~     set8.c
```

**\$mv:** mv stands for move. mv is used to move one or more files or directories from one place to another in file system like UNIX. It has two distinct functions:

(i) It rename a file or folder.

(ii) It moves group of files to different directory.

No additional space is consumed on a disk during renaming. This command normally works silently means no prompt for confirmation.

**Syntax:**

mv [Option] source destination

```
griet@griet-desktop:~$ mv file27 file54
griet@griet-desktop:~$ cat file54
hello
hi
bye
bye
welcome
hone
happy
christmas
diwali
holi
dussera
onam
griet@griet-desktop:~$
```

**\$chmod:** To change directory permissions in Linux, use the following:

1. chmod +rwx filename to add permissions.
2. chmod -rwx directoryname to remove permissions.
3. chmod +x filename to allow executable permissions.
4. chmod -wx filename to take out write and executable permissions.

**Output:**

```
griet@griet-desktop: ~
chmod: cannot access 'file27': No such file or directory
griet@griet-desktop:~$ chmod 777 file22
griet@griet-desktop:~$ ls -long
total 956
-rw-rw-r-- 1 950 Feb 16 2019 1a.c~
-rw-rw-r-- 1 1047 Jan 31 2019 1.c~
-rw-rw-r-- 1 265 Feb 12 2019 1.cpp~
-rw-rw-r-- 1 115 Oct 26 2019 1.sce
```

```

griet@griet-desktop: ~
rW-rW-r-- 1 735 Dec 17 2019 fc.c
rW-rW-r-- 1 735 Dec 17 2019 fc.c~
rW-rW-r-- 1 498 May 8 2019 fcfc.c
rW-rW-r-- 1 802 May 8 2019 fcfs.c
rW-rW-r-- 1 802 May 8 2019 fcfs.c~
rW-rW-r-- 1 851 May 6 2019 fcfs.cpp
rW-rW-r-- 1 850 May 6 2019 fcfs.cpp~
rW-rW-r-- 1 583 Apr 9 2019 fifo.c~
rW-rW-r-- 1 0 Mar 12 13:32 file
rwxrwxrwx 1 72 Nov 6 2019 file1
rW-rW-r-- 1 513 Feb 25 14:11 file1.c
rW-rW-r-- 1 511 Feb 25 14:09 file1.c~
rW-rW-r-- 1 1 Nov 4 2019 file1.tst
rW-rW-r-- 1 18 Nov 4 2019 file1.txt
rwxrwxrwx 1 72 Mar 12 12:44 file22
rW-rW-r-- 1 16 Nov 4 2019 file2.txt
rW-rW-r-- 1 5 Jul 3 2019 file3
rW-rW-r-- 1 18 Nov 4 2019 file3.txt
rW-rW-r-- 1 72 Mar 12 12:46 file44
rW-rW-r-- 1 26 Nov 4 2019 file4.txt
rW-rW-r-- 1 72 Mar 12 13:35 file54
rW-rW-r-- 1 85 Sep 26 2019 filename.py
rW-rW-r-- 1 83 Sep 26 2019 file.py
rW-rW-r-- 1 82 Sep 26 2019 file.py.save

```

## \$ps(Process Status):

This command is used to display the attributes of a process.

**Syntax:** \$ps

**Example:** \$ps

Output: PID	TTY	TIME	CMD
644	01	10:30:00	bash
643	02	10:31:00	ps

**Options:**

**-f:** detailed listing which shows parent of every process,use(-f)->(full) option.

**Example:** \$ps -f

**Output:**

UID	PID	PPID	C	STIME	TTY	TIME	CMD
Sumid	291	1	0	10:24:36	console	0:00	-bash

**-u:**it displays processes of a user.

**Example:** \$ps -u sumit

Output:	PID	TTY	TIME	CMD
	378	?	00:05	xsun
	403	?	00:00	xsession

**-a:** displaying all user processes.

**Example:** \$ps -a

Output :	PID	TTY	TIME	CMD
	662	pts/01	00:00:00	ksh
	705	pts/02	00:00:00	sh

**Output:**

```
griet@griet-desktop: ~
griet@griet-desktop:~$ ps -e
  PID TTY          TIME CMD
   1 ?            00:00:01 init
   2 ?            00:00:00 kthreadd
   3 ?            00:00:00 ksoftirqd/0
   5 ?            00:00:00 kworker/0:0H
   7 ?            00:00:02 rcu_sched
   8 ?            00:00:00 rcu_bh
   9 ?            00:00:00 migration/0
  10 ?            00:00:00 watchdog/0
  11 ?            00:00:00 watchdog/1
  12 ?            00:00:00 migration/1
  13 ?            00:00:00 ksoftirqd/1
  15 ?            00:00:00 kworker/1:0H
  16 ?            00:00:00 watchdog/2
  17 ?            00:00:00 migration/2
  18 ?            00:00:00 ksoftirqd/2
  20 ?            00:00:00 kworker/2:0H
  21 ?            00:00:00 watchdog/3
  22 ?            00:00:00 migration/3
  23 ?            00:00:00 ksoftirqd/3
  25 ?            00:00:00 kworker/3:0H
  26 ?            00:00:00 kdevtmpfs
```

**\$kill:** This command is used to kill the process i.e; stop or terminate a process.(by administrator)  
**Syntax:** \$kill <pid>

**Example:** \$kill 644

**Output:** The process gets terminated.

## Task 2

**Task:** Write programs using shell programming and use of vi editor

### Program:

#### Essential Vi Commands

---

- Open a file:

```
vi filename
```

- To go into edit mode:

press ESC and type I

- To go into command mode:

press ESC

- To save a file

press ESC and type :w fileName

- To save a file and quit:

press ESC and type :wq

OR

press ESC and type :x

- To jump to a line:

press ESC and type :the line number

- To Search for a string:

Press ESC and type /wordToSearch

- To quit vi:

Press ESC and type :q

Save the following into a file called hello.sh:

```
#!/bin/bash
echo "Hello, World!"
echo "Knowledge is power."
```

Save and close the file. You can run the script as follows:

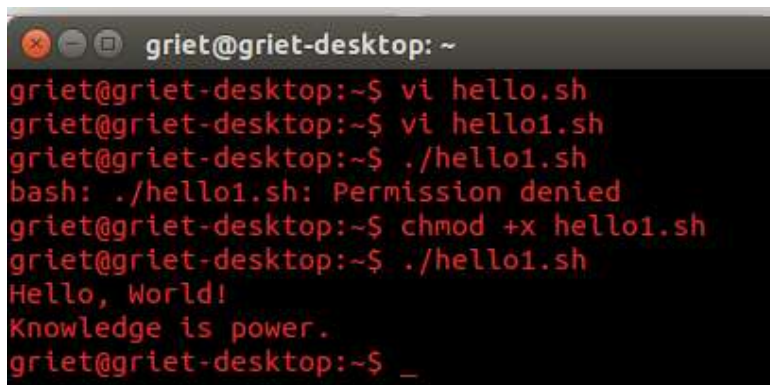
```
./hello.sh
```

### Saving and Running Your Script

The command `./hello.sh` displayed an error message on the screen. It will not run script since you've not set execute permission for your script `hello.sh`. To execute this program, type the following command:

```
chmod +x hello.sh
./hello.sh
```

### Output:

A terminal window titled 'griet@griet-desktop: ~' showing the following commands and output:

```
griet@griet-desktop:~$ vi hello.sh
griet@griet-desktop:~$ vi hello1.sh
griet@griet-desktop:~$ ./hello1.sh
bash: ./hello1.sh: Permission denied
griet@griet-desktop:~$ chmod +x hello1.sh
griet@griet-desktop:~$ ./hello1.sh
Hello, World!
Knowledge is power.
griet@griet-desktop:~$ _
```

A terminal window titled 'griet@griet-desktop: ~' showing the content of the script file:

```
#!/bin/bash
echo "Hello, World!"

echo "Knowledge is power."
```