

# CDAC MUMBAI

## Concepts of Operating System

### Assignment 2 Answers

Name: Amey Mahesh Sonawane

PG-DAC, CDAC Mumbai

## Part A

- echo "Hello, World!"

Answer: This command will simply print the given string in the terminal.



```
cdac@Amey: ~  
cdac@Amey:~$ echo "Hello, World!"  
Hello, World!  
cdac@Amey:~$
```

- name = "Productive"

Answer: This command will assign the string "Productive" inside the variable "name". No output will be displayed though.



```
cdac@Amey: ~  
cdac@Amey:~$ name="Productive"  
cdac@Amey:~$ |
```

- touch file.txt

Answer: This command will create a new file named "file.txt" or if it already exists, it will update the file's last modified date and time. No output as such.



```
cdac@Amey: ~  
cdac@Amey:~$ touch file.txt  
cdac@Amey:~$ |
```

- ls -a

Answer: This command will list all the files and directories in the current directory.

```
cdac@AmeY: ~  
cdac@AmeY:~$ ls -la  
.          .bashrc   .lesshst  .sudo_as_admin_successful data.txt  input.txt  numbers.txt  
..         .cache    .local    .viminfo  duplicate.txt  new        output.txt  
.bash_history .landscape .motd_shown LinuxAssignment file.txt  new.txt  
.bash_logout .lesshstQ .profile  abc       fruits.txt  new1.txt  
cdac@AmeY:~$ |
```

- rm file.txt

Answer: This command will remove the file name file.txt from the current directory. No output as such.

```
cdac@AmeY: ~  
cdac@AmeY:~$ rm file.txt  
cdac@AmeY:~$ ls  
LinuxAssignment abc data.txt duplicate.txt fruits.txt input.txt new new.txt new1.txt numbers.txt output.txt  
cdac@AmeY:~$ |
```

- cp file1.txt file2.txt

Answer: This command will copy the content of file1.txt to file2.txt. If file2.txt does not exist, it will create the same.

```
cdac@AmeY: ~/Assgn2  
cdac@AmeY:~/Assgn2$ cat file1.txt  
Hello  
How are you  
Thank you  
Goodbye  
cdac@AmeY:~/Assgn2$ cp file1.txt file2.txt  
cdac@AmeY:~/Assgn2$ cat file2.txt  
Hello  
How are you  
Thank you  
Goodbye  
cdac@AmeY:~/Assgn2$ |
```

- mv file.txt /path/to/directory/

Answer: This command will move file.txt to specified directory.

```
cdac@AmeY: ~/NewFolder  
cdac@AmeY:~$ touch newfile.txt  
cdac@AmeY:~$ mv newfile.txt NewFolder  
cdac@AmeY:~$ cd NewFolder  
cdac@AmeY:~/NewFolder$ ls  
file.txt  files.txt  newfile.txt  
cdac@AmeY:~/NewFolder$
```

- `chmod 755 script.sh`

Answer: This command changes the permission of the file `script.sh`. It sets the owner permission to Read, Write and Execute. The group permission to Read and Execute and the permission for others to Read and Execute as well.

```
cdac@Amei: ~/NewFolder
cdac@Amei:~/NewFolder$ ls
file.txt  files.txt  newfile.txt
cdac@Amei:~/NewFolder$ chmod 755 files.txt
cdac@Amei:~/NewFolder$ ls -l
total 0
-rwxrwxrwx 1 777 cdac 0 Mar 2 10:19 file.txt
-rwxr-xr-x 1 cdac cdac 0 Mar 2 10:27 files.txt
-rw-r--r-- 1 cdac cdac 0 Mar 2 10:29 newfile.txt
cdac@Amei:~/NewFolder$
```

- `grep "pattern" file.txt`

Answer: This command searches for the word “pattern” inside the specified file.txt.

```
cdac@Amei: ~/NewFolder
cdac@Amei:~/NewFolder$ cat file.txt
New File Here
File Is Created
For Assignment File
cdac@Amei:~/NewFolder$ grep "File" file.txt
New File Here
File Is Created
For Assignment File
cdac@Amei:~/NewFolder$ |
```

- `kill PID`

Answer: This command kills a specified running process, when we provide the process id for it in place of “PID”.

```
cdac@Amei: ~/NewFolder
cdac@Amei:~/NewFolder$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3 21668 13084 ?        Ss   10:19   0:00 /sbin/init
root         2  0.0  0.0  2784  1928 ?        Sl   10:19   0:00 /init
root         6  0.0  0.0  2776   132 ?        Sl   10:19   0:00 plan9 --control-socket 7 --log-level 4 --server-fd 8
root        54  0.0  0.5 66824 19736 ?        S<s  10:19   0:00 /usr/lib/systemd/systemd-journald
root        94  0.0  0.1 23960  6060 ?        Ss   10:19   0:00 /usr/lib/systemd/systemd-udevd
systemd+  109  0.0  0.3 21452 12000 ?        Ss   10:19   0:00 /usr/lib/systemd/systemd-resolved
systemd+  110  0.0  0.1 91020  6404 ?        Ssl  10:19   0:00 /usr/lib/systemd/systemd-timesyncd
root       151  0.0  0.0  4236  2680 ?        Ss   10:19   0:00 /usr/sbin/cron -f -P
message+  152  0.0  0.1  9620  4760 ?        Ss   10:19   0:00 @dbus-daemon --system --address=systemd: --nofork --n
root       179  0.0  0.2 17976  8392 ?        Ss   10:19   0:00 /usr/lib/systemd/systemd-logind
root       186  0.0  0.5 1756096 20176 ?       Ssl  10:19   0:00 /usr/libexec/wsl-pro-service -vv
root       197  0.0  0.0   3160  1148 hvc0    Ss+  10:19   0:00 /sbin/agetty -o -p -- \u --noclear --keep-baud - 1152
syslog    203  0.0  0.1 222508  7484 ?       Ssl  10:19   0:00 /usr/sbin/rsyslogd -n -iNONE
root      212  0.0  0.0   3116  1148 tty1    Ss+  10:19   0:00 /sbin/agetty -o -p -- \u --noclear - linux
root      248  0.0  0.5 107008 22660 ?       Ssl  10:19   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unatt
root      336  0.0  0.0   2788   212 ?        Ss   10:19   0:00 /init
root      339  0.0  0.0   2788   216 ?        S   10:19   0:00 /init
cdac      348  0.0  0.1  6204  5504 pts/0    Ss   10:19   0:00 -bash
root      350  0.0  0.1  6688  4628 pts/1    Ss   10:19   0:00 /bin/login -f
cdac      445  0.0  0.2 20056 11240 ?       Ss   10:19   0:00 /usr/lib/systemd/systemd --user
cdac      446  0.0  0.0  21144  1724 ?        S   10:19   0:00 (sd-pam)
cdac      461  0.0  0.1  6072  5224 pts/1    S+   10:19   0:00 -bash
cdac      671  0.0  0.1  8280  4208 pts/0    R+   10:43   0:00 ps aux
cdac@Amei:~/NewFolder$ kill 248
-bash: kill: (248) - Operation not permitted
cdac@Amei:~/NewFolder$ sudo kill 248
[sudo] password for cdac:
cdac@Amei:~/NewFolder$ |
```

- `mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt`

Answer: This command is a combination of various commands. Firstly

`mkdir mydir`: This will create a directory named “mydir” in current directory.

`&& cd mydir`: This will open the directory “mydir” in the terminal.

`&& touch file.txt`: This will create a file named file.txt inside the “mydir” directory.

`&& echo “Hello, World!” > file.txt`: This will redirect the output “Hello, World!” into file.txt, storing the text inside the same.

`&& cat file.txt`: This will finally display the content of the file.txt on the terminal.

```
cdac@Amei: ~/NewFolder/m x + v
cdac@Amei:~/NewFolder$ mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
Hello, World!
cdac@Amei:~/NewFolder/mydir$ |
```

- `ls -l | grep ".txt"`

Answer: This command generates a detailed list of all the files in the current directory. And then this output is pipelined with `grep “.txt”` command, which will sort the files in the list which have “.txt” in their name.

```
cdac@Amei: ~ x + v
cdac@Amei:~$ ls -l
total 52
drwxrwxrwx 2 cdac cdac 4096 Feb 28 15:44 Desktop
drwxr-xr-x 4 cdac cdac 4096 Feb 27 14:08 LinuxAssignment
drwxr-xr-x 3 cdac cdac 4096 Mar 2 10:46 NewFolder
-rw-r--r-- 1 cdac cdac 101 Feb 26 11:04 abc
-rw-r--r-- 1 cdac cdac 163 Feb 27 12:47 data.txt
-rw-r--r-- 1 cdac cdac 116 Feb 27 13:18 duplicate.txt
-rw-r--r-- 1 cdac cdac 95 Feb 27 13:24 fruits.txt
-rw-r--r-- 1 cdac cdac 36 Feb 27 13:13 input.txt
-rw-r--r-- 1 cdac cdac 16 Feb 28 11:24 new
-rw-r--r-- 1 cdac cdac 10 Feb 27 09:38 new.txt
-rw-r--r-- 1 cdac cdac 10 Feb 27 09:39 new1.txt
-rw-r--r-- 1 cdac cdac 51 Feb 27 12:51 numbers.txt
-rw-r--r-- 1 cdac cdac 36 Feb 27 13:14 output.txt
cdac@Amei:~$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac 163 Feb 27 12:47 data.txt
-rw-r--r-- 1 cdac cdac 116 Feb 27 13:18 duplicate.txt
-rw-r--r-- 1 cdac cdac 95 Feb 27 13:24 fruits.txt
-rw-r--r-- 1 cdac cdac 36 Feb 27 13:13 input.txt
-rw-r--r-- 1 cdac cdac 10 Feb 27 09:38 new.txt
-rw-r--r-- 1 cdac cdac 10 Feb 27 09:39 new1.txt
-rw-r--r-- 1 cdac cdac 51 Feb 27 12:51 numbers.txt
-rw-r--r-- 1 cdac cdac 36 Feb 27 13:14 output.txt
cdac@Amei:~$ |
```

- `cat file1.txt file2.txt | sort | uniq`

Answer: This command will display the content of both the files file1.txt and file2.txt together, and this output is pipelined with the sort command which will alphabetically sort the content of the files. In the end uniq will display only those lines which are distinct, omitting all the repeated lines in the file.

```
cdac@Amey: ~/OsAssgn2
cdac@Amey:~/OsAssgn2$ nano file1.txt
cdac@Amey:~/OsAssgn2$ nano file2.txt
cdac@Amey:~/OsAssgn2$ cat file1.txt file2.txt
India
New Zealand
Australia
Pakistan
West Indies
England
Afghanistan
Sri Lanka
India
Bangladesh
Argentina
Portugal
Germany
Brazil
Uruguay
Ecuador
Pakistan
Bangladesh
cdac@Amey:~/OsAssgn2$ cat file1.txt file2.txt | sort | uniq
Afghanistan
Argentina
Australia
Bangladesh
Brazil
Ecuador
England
Germany
India
New Zealand
Pakistan
Portugal
Sri Lanka
Uruguay
West Indies
cdac@Amey:~/OsAssgn2$
```

- `ls -l | grep "^d"`

Answer: This command will display a detailed list of all the directories in the current directory. Because the `grep "^d"` command filters out the files, as those lines won't have `d` as the first character in them.

```
cdac@Amey: ~
cdac@Amey:~$ ls -l
total 56
drwxrwxrwx 2 cdac cdac 4096 Feb 28 15:44 Assgn2
drwxr-xr-x 4 cdac cdac 4096 Feb 27 14:08 LinuxAssignment
drwxr-xr-x 3 cdac cdac 4096 Mar  2 10:46 NewFolder
drwxr-xr-x 2 cdac cdac 4096 Mar  2 11:03 OsAssgn2
-rw-r--r-- 1 cdac cdac 101 Feb 26 11:04 abc
-rw-r--r-- 1 cdac cdac 163 Feb 27 12:47 data.txt
-rw-r--r-- 1 cdac cdac 116 Feb 27 13:18 duplicate.txt
-rw-r--r-- 1 cdac cdac 95 Feb 27 13:24 fruits.txt
-rw-r--r-- 1 cdac cdac 36 Feb 27 13:13 input.txt
-rw-r--r-- 1 cdac cdac 16 Feb 28 11:24 new
-rw-r--r-- 1 cdac cdac 10 Feb 27 09:38 new.txt
-rw-r--r-- 1 cdac cdac 10 Feb 27 09:39 new1.txt
-rw-r--r-- 1 cdac cdac 51 Feb 27 12:51 numbers.txt
-rw-r--r-- 1 cdac cdac 36 Feb 27 13:14 output.txt
cdac@Amey:~$ ls -l | grep "^d"
drwxrwxrwx 2 cdac cdac 4096 Feb 28 15:44 Assgn2
drwxr-xr-x 4 cdac cdac 4096 Feb 27 14:08 LinuxAssignment
drwxr-xr-x 3 cdac cdac 4096 Mar  2 10:46 NewFolder
drwxr-xr-x 2 cdac cdac 4096 Mar  2 11:03 OsAssgn2
cdac@Amey:~$
```

- `grep -r "pattern" /path/to/directory/`

Answer: This command will display the lines of files in a certain specified directory where the word “pattern” exists. Also, the search will be recursive, as in all the subdirectories and the files inside them will also be checked for the “pattern”.

```
cdac@Amey: ~  
cdac@Amey:~$ ls  
NewFolder  abc      duplicate.txt  input.txt  new.txt  numbers.txt  
LinuxAssignment  OsAssgn2  data.txt  fruits.txt  new      new1.txt  output.txt  
cdac@Amey:~$ grep -r "pattern" /home  
/home/cdac/input.txt:pattern here  
/home/cdac/new1.txt:pattern here  
/home/cdac/new:pattern here  
/home/cdac/data.txt:pattern here  
/home/cdac/.bashrc:# If set, the pattern "*" used in a pathname expansion context will  
cdac@Amey:~$ |
```

- `cat file1.txt file2.txt | sort | uniq -d`

Answer: This command will display all the duplicate lines inside the files file1.txt and file2.txt in the current folder.

```
cdac@Amey: ~/OsAssgn2  
cdac@Amey:~/OsAssgn2$ cat file1.txt file2.txt  
India  
New Zealand  
Australia  
Pakistan  
West Indies  
England  
Afghanistan  
Sri Lanka  
India  
Bangladesh  
Argentina  
Portugal  
Germany  
Brazil  
Uruguay  
Ecuador  
Pakistan  
Bangladesh  
cdac@Amey:~/OsAssgn2$ cat file1.txt file2.txt | sort | uniq -d  
Bangladesh  
India  
Pakistan  
cdac@Amey:~/OsAssgn2$ |
```

- `chmod 644 file.txt`

Answer: This command will set the permissions for groups and others to read only, and read and write only for owner, for file.txt

```
cdac@Amey: ~  
cdac@Amey:~$ ls -l file.txt  
-r--r--r-- 1 cdac cdac 0 Mar  2 11:25 file.txt  
cdac@Amey:~$ chmod 644 file.txt  
cdac@Amey:~$ ls -l file.txt  
-rw-r--r-- 1 cdac cdac 0 Mar  2 11:25 file.txt  
cdac@Amey:~$ |
```

- `cp -r source_directory destination_directory`

Answer: This command will copy all the files and folders of the source directory to the destination directory.

```
cdac@Amey: ~/destnFolder/s × + v
cdac@Amey:~$ cd sourceFolder
cdac@Amey:~/sourceFolder$ ls
file1.txt file2.txt file3.txt
cdac@Amey:~/sourceFolder$ cd ..
cdac@Amey:~$ cp -r sourceFolder destnFolder
cdac@Amey:~$ cd destnFolder
cdac@Amey:~/destnFolder$ ls
sourceFolder
cdac@Amey:~/destnFolder$ cd sourceFolder
cdac@Amey:~/destnFolder/sourceFolder$ ls
file1.txt file2.txt file3.txt
cdac@Amey:~/destnFolder/sourceFolder$ |
```

- `find /path/to/search -name "*.txt"`

Answer: This command will search in all the files and directories in the specified path for files with .txt in their name.

```
cdac@Amey: ~ × + v
cdac@Amey:~$ ls
Assgn2  OsAssgn2  destnFolder  fruits.txt  new.txt  output.txt
LinuxAssignment  abc  duplicate.txt  input.txt  new1.txt  sourceFolder
NewFolder  data.txt  file.txt  new  numbers.txt
cdac@Amey:~$ find /home -name "*.txt"
/home/cdac/numbers.txt
/home/cdac/NewFolder/mydir/file.txt
/home/cdac/NewFolder/files.txt
/home/cdac/NewFolder/file.txt
/home/cdac/NewFolder/newfile.txt
/home/cdac/fruits.txt
/home/cdac/input.txt
/home/cdac/Assgn2/file1.txt
/home/cdac/Assgn2/file2.txt
/home/cdac/destnFolder/sourceFolder/file1.txt
/home/cdac/destnFolder/sourceFolder/file2.txt
/home/cdac/destnFolder/sourceFolder/file3.txt
/home/cdac/output.txt
/home/cdac/new.txt
/home/cdac/sourceFolder/file1.txt
/home/cdac/sourceFolder/file2.txt
/home/cdac/sourceFolder/file3.txt
/home/cdac/OsAssgn2/file1.txt
/home/cdac/OsAssgn2/file2.txt
/home/cdac/new1.txt
/home/cdac/duplicate.txt
/home/cdac/data.txt
/home/cdac/file.txt
/home/cdac/LinuxAssignment/UnzippedDocsHere/docs/file2.txt
/home/cdac/LinuxAssignment/file1.txt
/home/cdac/LinuxAssignment/docs/file2.txt
cdac@Amey:~$ |
```

- `chmod u+x file.txt`

Answer: This command allows the user (owner) with the permission to execute the file named "file.txt".

```
cdac@Amey: ~  
cdac@Amey:~$ ls -l file.txt  
-rw-r--r-- 1 cdac cdac 0 Mar  2 14:03 file.txt  
cdac@Amey:~$ chmod u+x file.txt  
cdac@Amey:~$ ls -l file.txt  
-rwxr--r-- 1 cdac cdac 0 Mar  2 14:03 file.txt  
cdac@Amey:~$ |
```

- echo \$PATH

Answer: This command displays the system's environment variable PATH, which defines directories where the shell looks for executable commands.

```
cdac@Amey: ~  
cdac@Amey:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Program Files/Common Files/Oracle/Java/javapath:/mnt/c/Windows/system32:/mnt/c/Windows:/mnt/c/Windows/System32/Wbem:/mnt/c/Windows/System32/WindowsPowerShell/v1.0:/mnt/c/Windows/System32/OpenSSH/:/mnt/c/Program Files (x86)/NVIDIA Corporation/PhysX/Common:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0:/mnt/c/WINDOWS/System32/OpenSSH:/mnt/c/Program Files/Java/jdk-23/bin:/mnt/c/Program Files/NVIDIA Corporation/NVIDIA app/NvDLISR:/mnt/c/Program Files/dotnet:/mnt/c/Program Files/Git/cmd:/mnt/c/Program Files/Git/bin:/mnt/c/Users/Amey/AppData/Local/Microsoft/WindowsApps:/mnt/c/Users/Amey/AppData/Local/Programs/Microsoft VS Code/bin:/snap/bin  
cdac@Amey:~$ |
```

## Part B

Identify True or False:

1. ls is used to list files and directories in a directory.  
- True.

```
cdac@Amey: ~  
cdac@Amey:~$ ls  
destn2      OsAssgn2  destnFolder  fruits.txt  new.txt      output.txt  
LinuxAssignment abc       duplicate.txt input.txt   new1.txt     sourceFolder  
NewFolder  data.txt  file.txt     new         numbers.txt  
cdac@Amey:~$ |
```

2. mv is used to move files and directories.  
- True

```
cdac@Amey: ~/NewFolder  
cdac@Amey:~$ ls  
destn2      NewFolder  abc          destnFolder  file.txt     input.txt   new1.txt     output.txt  
LinuxAssignment OsAssgn2  data.txt    duplicate.txt fruits.txt   new.txt     numbers.txt  sourceFolder  
cdac@Amey:~$ mv abc NewFolder  
cdac@Amey:~$ cd NewFolder  
cdac@Amey:~/NewFolder$ ls  
abc  file.txt  files.txt  mydir  newfile.txt  
cdac@Amey:~/NewFolder$ |
```



3. cd is used to copy files and directories.

- False.

Reason: The command “cp” is used to copy files and directories instead of “cd”. “cd” is used to change the current directory in the terminal.

4. pwd stands for "print working directory" and displays the current directory.

- True.

```
cdac@Amey: ~/NewFolder
cdac@Amey:~/NewFolder$ pwd
/home/cdac/NewFolder
cdac@Amey:~/NewFolder$ |
```

5. grep is used to search for patterns in files.

- True.

```
cdac@Amey: ~
cdac@Amey:~$ cat new1.txt
Amey
Amey
pattern here
cdac@Amey:~$ grep "Amey" new1.txt
Amey
Amey
cdac@Amey:~$ |
```

6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.

- True.

```
cdac@Amey: ~
cdac@Amey:~$ ls -l data.txt
-r--r--r-- 1 cdac cdac 176 Mar  2 11:15 data.txt
cdac@Amey:~$ chmod 755 data.txt
cdac@Amey:~$ ls -l data.txt
-rwxr-xr-x 1 cdac cdac 176 Mar  2 11:15 data.txt
cdac@Amey:~$ |
```

7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

- True.

```
cdac@Amey: ~/directory1
cdac@Amey:~$ mkdir -p directory1/directory2
cdac@Amey:~$ ls
Assign2      OsAssgn2      directory1     fruits.txt    new1.txt      sourceFolder
LinuxAssignment  data.txt      duplicate.txt  input.txt    numbers.txt
NewFolder    destnFolder   file.txt      new.txt      output.txt
cdac@Amey:~$ cd directory1
cdac@Amey:~/directory1$ ls
directory2
cdac@Amey:~/directory1$ |
```

8. `rm -rf file.txt` deletes a file forcefully without confirmation.
- True.

```
cdac@Amei:~$ ls
Assign2      OsAssgn2      directory1     fruits.txt     new1.txt       sourceFolder
LinuxAssignment data.txt       duplicate.txt   input.txt      numbers.txt
NewFolder    destnFolder    file.txt       new.txt        output.txt
cdac@Amei:~$ rm -rf file.txt
cdac@Amei:~$ ls
Assign2      OsAssgn2      directory1     input.txt      numbers.txt
LinuxAssignment data.txt       duplicate.txt   new.txt        output.txt
NewFolder    destnFolder    fruits.txt     new1.txt       sourceFolder
cdac@Amei:~$ |
```

Identify the Incorrect Commands:

1. `chmodx` is used to change file permissions.
  - Correct

If the “x” mentioned at the end of the command `chmod` is considered as a command line option, then yes, it is correct. `chmod 755` gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.
2. `cpy` is used to copy files and directories.
  - Incorrect

`cpy` command does not copy files and directories. The task can be performed with the command `cp` followed by file name.
3. `mkfile` is used to create a new file.
  - Incorrect

`mkfile` command does not exist in linux. To create a new file, one can use “`touch file.name`” instead.
4. `catx` is used to concatenate files.
  - Correct

If the “x” mentioned at the end of the command is considered as a command line option then yes, it is correct. `cat` command is used to concatenate and display content of a file. It can be followed by `-n` to show numbered output lines, or by `-b` to show only non-empty lines.
5. `rn` is used to rename files.
  - Incorrect

To rename a file, the `mv` command is used. There is no `rn` command in existence.

## Part C

Question 1: Write a shell script that prints "Hello, World!" to the terminal.

Answer: The command is as follows: -

```
echo "Hello, World!"
```

A terminal window titled 'cdac@Ameey: ~' with standard window controls. The prompt is 'cdac@Ameey:~\$'. The user enters 'echo "Hello, World"', and the terminal outputs 'Hello, World' on the next line. The prompt returns to 'cdac@Ameey:~\$'.

Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

Answer: The command is as follows: -

```
name="CDAC Mumbai"
```

```
echo "The name is: $name"
```

A terminal window titled 'cdac@Ameey: ~' with standard window controls. The prompt is 'cdac@Ameey:~\$'. The user enters 'name="CDAC Mumbai"', and the prompt returns. The user then enters 'echo "The name is: \$name"', and the terminal outputs 'The name is: CDAC Mumbai' on the next line. The prompt returns to 'cdac@Ameey:~\$'.

Question 3: Write a shell script that takes a number as input from the user and prints it.

Answer: The command is as follows: -

```
read -p "Enter a number: " num
```

```
echo "You entered: $num"
```

A terminal window titled 'cdac@Ameey: ~' with standard window controls. The prompt is 'cdac@Ameey:~\$'. The user enters 'read -p "Enter a number: " num', and the terminal outputs 'Enter a number: 54' on the next line. The user then enters 'echo "You entered: \$num"', and the terminal outputs 'You entered: 54' on the next line. The prompt returns to 'cdac@Ameey:~\$'.

Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

Answer: The command will be as follows: -

```
num1=5
```

```
num2=3
```

```
sum=$((num1 + num2))
```

```
echo "The sum is: $sum"
```

A terminal window titled 'cdac@Ameiy: ~' with standard window controls. It shows the execution of a shell script: 'cdac@Ameiy:~\$ num1=5', 'cdac@Ameiy:~\$ num2=6', 'cdac@Ameiy:~\$ sum=\$((num1+num2))', and 'cdac@Ameiy:~\$ echo "Sum is \$sum"'. The output is 'Sum is 11'. The prompt returns to 'cdac@Ameiy:~\$' with a cursor.

```
cdac@Ameiy:~$ num1=5
cdac@Ameiy:~$ num2=6
cdac@Ameiy:~$ sum=$((num1+num2))
cdac@Ameiy:~$ echo "Sum is $sum"
Sum is 11
cdac@Ameiy:~$ |
```

Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

Answer: The command will be as follows: -

```
read -p "Enter a number: " num
```

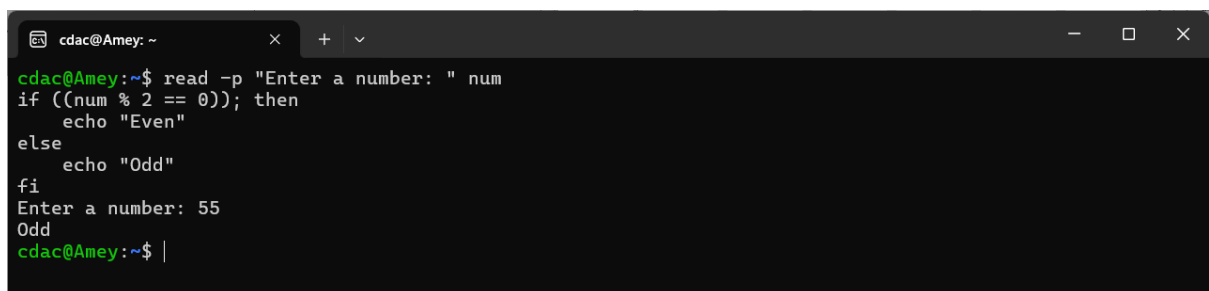
```
if ((num % 2 == 0)); then
```

```
    echo "Even"
```

```
else
```

```
    echo "Odd"
```

```
fi
```

A terminal window titled 'cdac@Ameiy: ~' with standard window controls. It shows the execution of a shell script: 'cdac@Ameiy:~\$ read -p "Enter a number: " num', 'cdac@Ameiy:~\$ if ((num % 2 == 0)); then', 'cdac@Ameiy:~\$ echo "Even"', 'cdac@Ameiy:~\$ else', 'cdac@Ameiy:~\$ echo "Odd"', 'cdac@Ameiy:~\$ fi', 'cdac@Ameiy:~\$ Enter a number: 55', 'cdac@Ameiy:~\$ Odd'. The output is 'Odd'. The prompt returns to 'cdac@Ameiy:~\$' with a cursor.

```
cdac@Ameiy:~$ read -p "Enter a number: " num
cdac@Ameiy:~$ if ((num % 2 == 0)); then
cdac@Ameiy:~$     echo "Even"
cdac@Ameiy:~$ else
cdac@Ameiy:~$     echo "Odd"
cdac@Ameiy:~$ fi
cdac@Ameiy:~$ Enter a number: 55
cdac@Ameiy:~$ Odd
cdac@Ameiy:~$ |
```

Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

Answer: The command is as follows: -

```
for i in {1..5}; do
```

```
    echo "$i"
```

```
done
```

A terminal window titled 'cdac@Amey: ~' showing a shell script execution. The script uses a 'for' loop to iterate over the set {1..5}. The output shows the numbers 1 through 5 printed on separate lines.

```
cdac@Amey:~$ for i in {1..5}; do
echo "$i"
done
1
2
3
4
5
cdac@Amey:~$ |
```

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

Answer: The command is as follows: -

```
i=1
```

```
while [ $i -le 5 ]; do
```

```
    echo "$i"
```

```
    ((i++))
```

```
done
```

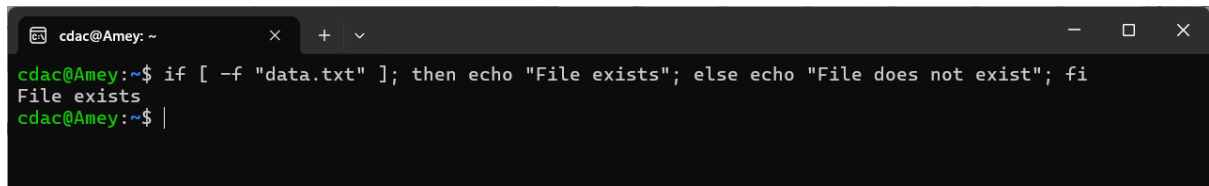
A terminal window titled 'cdac@Amey: ~' showing a shell script execution. The script initializes 'i' to 1 and uses a 'while' loop to print numbers from 1 to 5. The output shows the numbers 1 through 5 printed on separate lines.

```
cdac@Amey:~$ i=1
while [ $i -le 5 ]; do
echo "$i"
((i++))
done
1
2
3
4
5
cdac@Amey:~$ |
```

Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".

Answer: The command is as follows: -

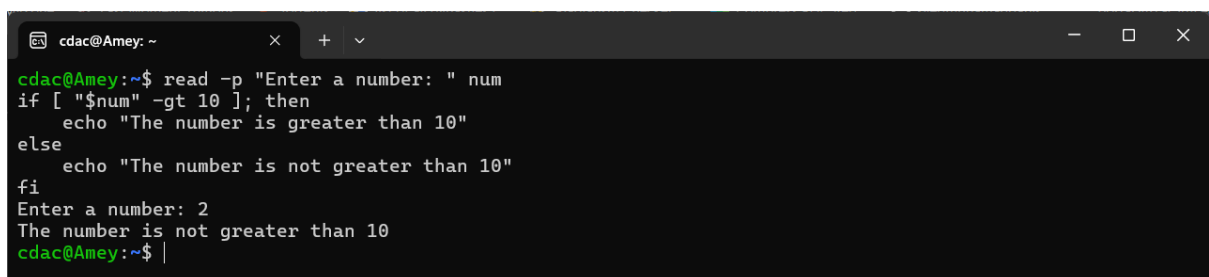
```
if [ -f "file.txt" ]; then  
  
    echo "File exists"  
  
else  
  
    echo "File does not exist"  
  
fi
```

A terminal window titled 'cdac@Amey: ~' with standard window controls. The prompt is 'cdac@Amey:~\$'. The user enters the command 'if [ -f "data.txt" ]; then echo "File exists"; else echo "File does not exist"; fi'. The output is 'File exists'. The prompt returns to 'cdac@Amey:~\$'.

Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.

Answer: The command is as follows: -

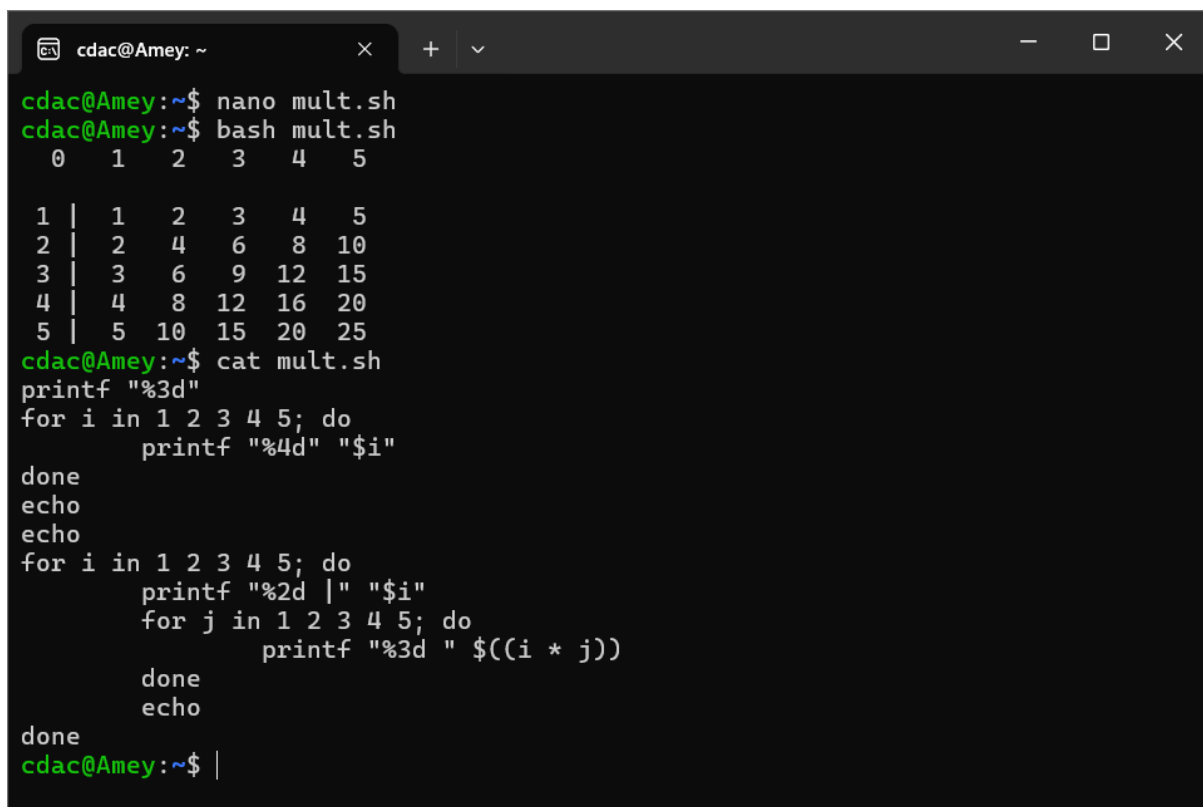
```
read -p "Enter a number: " num  
  
if [ "$num" -gt 10 ]; then  
  
    echo "The number is greater than 10"  
  
else  
  
    echo "The number is not greater than 10"  
  
fi
```

A terminal window titled 'cdac@Amey: ~' with standard window controls. The prompt is 'cdac@Amey:~\$'. The user enters the command 'read -p "Enter a number: " num'. The prompt changes to 'Enter a number: '. The user enters '2'. The prompt returns to 'cdac@Amey:~\$'. The user then enters the command 'if [ "\$num" -gt 10 ]; then echo "The number is greater than 10"; else echo "The number is not greater than 10"; fi'. The output is 'The number is not greater than 10'. The prompt returns to 'cdac@Amey:~\$'.

Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

Answer: The shell script will be as follows: -

```
printf "%3d"
for i in 1 2 3 4 5; do
    printf "%4d" "$i"
done
echo
echo
for i in 1 2 3 4 5; do
    printf "%2d|" "$i"
    for j in 1 2 3 4 5; do
        printf "%3d " $((i * j))
    done
    echo
done
```

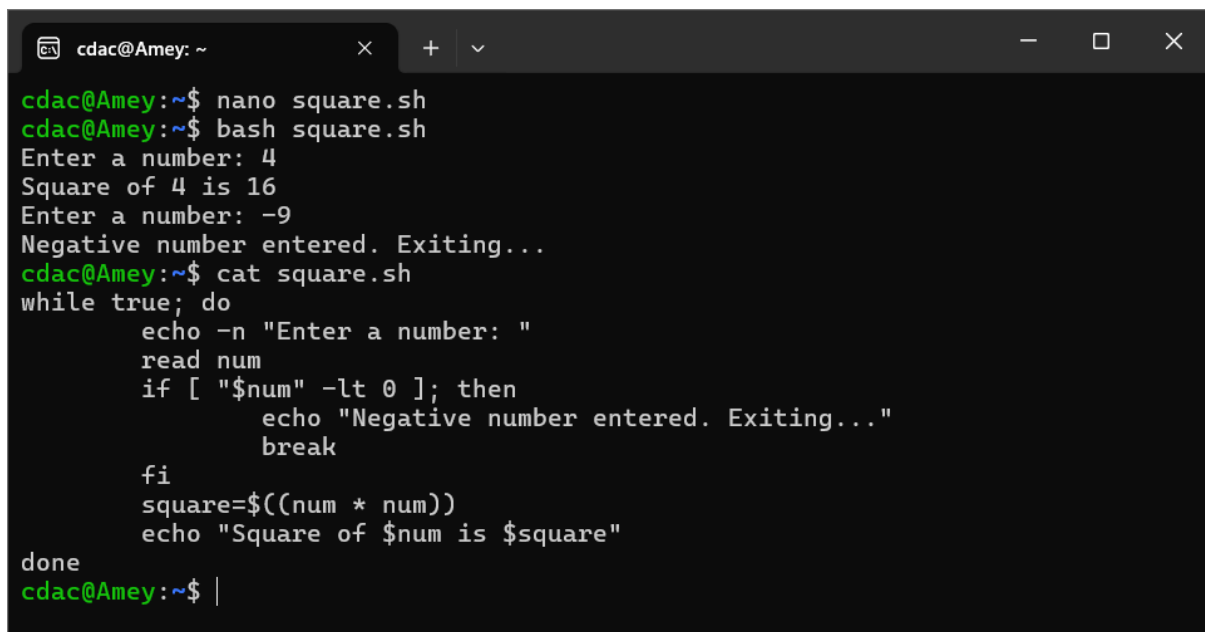


```
cdac@Amey: ~  
cdac@Amey:~$ nano mult.sh  
cdac@Amey:~$ bash mult.sh  
0 1 2 3 4 5  
  
1 | 1 2 3 4 5  
2 | 2 4 6 8 10  
3 | 3 6 9 12 15  
4 | 4 8 12 16 20  
5 | 5 10 15 20 25  
cdac@Amey:~$ cat mult.sh  
printf "%3d"  
for i in 1 2 3 4 5; do  
    printf "%4d" "$i"  
done  
echo  
echo  
for i in 1 2 3 4 5; do  
    printf "%2d|" "$i"  
    for j in 1 2 3 4 5; do  
        printf "%3d " $((i * j))  
    done  
    echo  
done  
cdac@Amey:~$ |
```

Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.

Answer: The shell script will be as follows: -

```
while true; do
    echo -n "Enter a number: "
    read num
    if [ "$num" -lt 0 ]; then
        echo "Negative number entered. Exiting..."
        break
    fi
    square=$((num * num))
    echo "Square of $num is $square"
done
```

A terminal window titled 'cdac@Amey: ~' with standard window controls. The terminal shows the following sequence of commands and output:  
cdac@Amey:~\$ nano square.sh  
cdac@Amey:~\$ bash square.sh  
Enter a number: 4  
Square of 4 is 16  
Enter a number: -9  
Negative number entered. Exiting...  
cdac@Amey:~\$ cat square.sh  
while true; do  
 echo -n "Enter a number: "  
 read num  
 if [ "\$num" -lt 0 ]; then  
 echo "Negative number entered. Exiting..."  
 break  
 fi  
 square=\$((num \* num))  
 echo "Square of \$num is \$square"  
done  
cdac@Amey:~\$ |



## **Part E**

1. Consider the following processes with arrival times and burst times:

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

Solution:

Gantt Chart:

P1	P2	P3	
0	5	8	14

Response time is time taken by a process to respond to the CPU call.

So, we can infer the response times for the processes, from the gantt chart as these: -

Process ID	Arrival Time	Burst Time	Response Time
P1	0	5	0
P2	1	3	5
P3	2	6	8

Now waiting time can be calculated with the formula,

Waiting time = Response time – Arrival time (for non-pre-emptive

According the following table can be formed

Process ID	Arrival Time	Burst Time	Response Time	Waiting Time
P1	0	5	0	$0 - 0 = 0$
P2	1	3	5	$5 - 1 = 4$
P3	2	6	8	$8 - 2 = 6$

So, the average waiting time for these processes using FCFS algorithms is equal to

$$= (0 + 4 + 6) / 3$$

$$= 3.33 \text{ units.}$$

2. Consider the following processes with arrival times and burst times:

Process ID	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

Solution:

Gantt Chart:

P1	P1	P3	P1	P4	P2	
0	1	2	3	4	8	13

Waiting time is the time spent by the process in the waiting queue,

For process P1, waiting time = 1

For process P2 =  $8 - 1 = 7$

For process P3 = 0

For process P4 =  $4 - 3 = 1$

therefore, the table can be formed like this

Process ID	Arrival Time	Burst Time	Waiting Time
P1	0	3	1
P2	1	5	7
P3	2	1	0
P4	3	4	1

And with waiting time available for each process, turn-around time can be calculated with the formula

Turn-Around Time = Waiting Time + Burst Time

Hence the table will look like this with added turn-around times of processes

Process ID	Arrival Time	Burst Time	Waiting Time	Turn-Around Time
P1	0	3	1	4
P2	1	5	7	12
P3	2	1	0	1
P4	3	4	1	5

So, the average turn-around time =  $(4 + 12 + 1 + 5) / 4$   
 $= 5.5$  units

The average turn-around time for these processes is equal to 5.5 units.

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.

Solution:

By using priority scheduling,

Gantt Chart:

P1	P2	P2	P2	P2	P4	P1	P3	
0	1	2	3	4	5	7	13	20

Waiting time is time spent by the process in the queue before it gets executed completely.

So, waiting times for each of the processes can be inferred as,

For process P1, waiting time =  $0 + 7 = 7$

For P2 =  $1 - 1 = 0$

For P3 =  $13 - 2 = 11$

And for P4 =  $5 - 3 = 2$

Accordingly, average waiting time can be calculated as

$$= (P1 + P2 + P3 + P4) / 4$$

$$= 20 / 4$$

$$= 5$$

Average waiting time of these processes is 5 units.

4. Consider the following processes with arrival times and burst times, and the time quantum for

Round Robin scheduling is 2 units:

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turnaround time using Round Robin scheduling.

Solution:

By round robin scheduling algorithm,

Gantt Chart:

P1	P2	P3	P4	P1	P2	P4	P2	
0	2	4	6	8	10	12	13	14

Waiting time is equal to the total time a process spends waiting for completion of execution.

So, from the Gantt chart, waiting times can be inferred as: -

$$P1 = 0 + 6 = 6$$

$$P2 = (2 + 6 + 1) - 1 = 8$$

$$P3 = 4 - 2 = 2$$

$$\text{And } P4 = (6 + 4) - 3 = 7$$

Using these values, we can calculate the turn-around time of each of the processes as well

The formula is,

$$\text{Turn-Around Time} = \text{Waiting Time} + \text{Burst Time}$$

$$\text{Turn-Around Time for P1} = 6 + 4 = 10$$

$$\text{For P2} = 5 + 8 = 13$$

$$\text{For P3} = 2 + 2 = 4$$

$$\text{And for P4} = 3 + 7 = 10$$

In conclusion, the average turn-around time would be equal to

$$= (P1 + P2 + P3 + P4) / 4$$

$$= 37 / 4$$

$$= 9.25 \text{ units}$$

Thus, the average turn-around time for these processes is 9.25 units.

5. Consider a program that uses the `fork()` system call to create a child process. Initially, the parent process has a variable `x` with a value of 5. After forking, both the parent and child processes increment the value of `x` by 1.

What will be the final values of `x` in the parent and child processes after the `fork()` call?

Answer:

The `fork()` is a system call in operating systems to generate a child process from an original process or parent process. Such different instances of a process allow much more efficient process management environment. With child processes in existence, there can be parallel execution of tasks. This increases the speed of processing considerably.

What exactly happens when a duplicate copy of parent process is created, is that the child process gets a duplicate copy of the parent process's memory space, including any initialized variables. After the `fork()` is called, both child and parent process run independently, which means changing anything in parent process won't affect anything in child process.

Given situation is,

Variable `x = 5` in parent process

`fork()` is called, so a child process is generated.

And finally, both parent and child increment `x` by 1

So, in parent process, `x = 5` will become `x = 5 + 1`

That is `x` becomes `x = 6`

Child process remains unaffected and simply increments `x` by 1 too

So `x = 5 + 1 = 6`

So we can conclude that since parent and child process are independent, the final values of `x` in parent process will be `x = 6` and in the child process will also be `x = 6`.