OS - Assignment 2

PART-A

- What will the following commands do?

- **echo "Hello, World!"**

  Displays line of text. It has few options like,

  -n : do not output the trailing newline

```
cdac@kalyanidivyani:~$ echo -n Hello from me
Hello from mecdac@kalyanidivyani:~$ echo -n Hello from me
Hello from mecdac@kalyanidivyani:~$ echo Hello
Hello
cdac@kalyanidivyani:~$ echo hi
hi
cdac@kalyanidivyani:~$ echo -n hiiee
hiieecdac@kalyanidivyani:~$
```

  -e enable interpretation of backslash escapes

```
cdac@kalyanidivyani:~$ echo -e "Hello\nworld"
Hello
world
```

  -E disable interpretation of backslash escapes (default)

```
cdac@kalyanidivyani:~$ echo -E "Hello\nworld"
Hello\nworld
```

Option	Behavior
-e	Enables interpretation of backslash escapes
-E	Disables interpretation of backslash escapes (default)

- **name="Productive"**
  It assigns string “Prodcutive” to the variable name. You can use this variable in the code. There are no spaces around the = sign; otherwise, it will cause an error. The string should be enclosed in quotes if it contains spaces or special characters.

```
cdac@kalyanidivyani:~$ name= "Productive"
Productive: command not found
cdac@kalyanidivyani:~$ name ="Productive"
Command 'name' not found, did you mean:
  command 'mame' from snap mame (mame0270)
  command 'nam' from deb nam (1.15-6)
  command 'nvme' from deb nvme-cli (2.8-1ubuntu0.1)
  command 'named' from deb bind9 (1:9.18.28-0ubuntu0.24.04.1)
  command 'lame' from deb lame (3.100-6)
  command 'namei' from deb util-linux (2.39.3-9ubuntu6.1)
  command 'nama' from deb nama (1.216-2)
  command 'mame' from deb mame (0.261+dfsg.1-1)
  command 'uname' from deb coreutils (9.4-2ubuntu2)
See 'snap info <snapname>' for additional versions.
cdac@kalyanidivyani:~$ name="Productive"
cdac@kalyanidivyani:~$ echo name
name
cdac@kalyanidivyani:~$ echo $name
Productive
```

- **touch file.txt**
  touch – Change file timestamp. Update the access and modification times of each FILE to the current time. A FILE argument that does not exist is created empty, unless -c or -h is supplied.

```
cdac@kalyanidivyani:~/LinuxAssignment$ ls
Assignment2  data.txt  docs-zip.zip   file1.txt  fruit.txt  output.txt       unzipped-docs
Numbers.txt  docs      duplicate.txt  file2.txt  input.txt  shellProgram.sh
cdac@kalyanidivyani:~/LinuxAssignment$ ls -l file1.txt
-rw-r--r-- 1 cdac cdac 59 Feb 27 13:15 file1.txt
cdac@kalyanidivyani:~/LinuxAssignment$ touch file1.txt
cdac@kalyanidivyani:~/LinuxAssignment$ ls -l file1.txt
-rw-r--r-- 1 cdac cdac 59 Mar  2 09:43 file1.txt
```

  Mandatory arguments to long options are mandatory for short options too.

-a change only the access time

```
cdac@kalyanidivyani:~/LinuxAssignment$ stat data.txt
  File: data.txt
  Size: 99              Blocks: 8          IO Block: 4096   regular file
Device: 8,32    Inode: 37517       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    cdac)   Gid: ( 1000/    cdac)
Access: 2025-02-27 13:30:30.103788822 +0000
Modify: 2025-02-27 13:30:18.277788870 +0000
Change: 2025-02-27 13:30:18.277788870 +0000
 Birth: 2025-02-27 13:30:18.268788870 +0000
cdac@kalyanidivyani:~/LinuxAssignment$ touch -a data.txt
cdac@kalyanidivyani:~/LinuxAssignment$ stat data.txt
  File: data.txt
  Size: 99              Blocks: 8          IO Block: 4096   regular file
Device: 8,32    Inode: 37517       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    cdac)   Gid: ( 1000/    cdac)
Access: 2025-03-02 10:03:26.252788715 +0000
Modify: 2025-02-27 13:30:18.277788870 +0000
Change: 2025-03-02 10:03:26.252788715 +0000
 Birth: 2025-02-27 13:30:18.268788870 +0000
```

-c, --no-create

do not create any files

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ ls
code10  code11  code3  code4  code5  code6  code7  code8  code9  fib  fibonacci  greatst3  primeno  sumodd
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ touch -c code12
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ ls
code10  code11  code3  code4  code5  code6  code7  code8  code9  fib  fibonacci  greatst3  primeno  sumodd
```

-d, --date=STRING

parse STRING and use it instead of current time

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ ls -l code11
-rw-r--r-- 1 cdac cdac 253 Feb 28 19:13 code11
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ touch -d "2023-01-01 12:30:00" code11
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ ls -l code11
-rw-r--r-- 1 cdac cdac 253 Jan  1  2023 code11
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ touch -d "yesterday" code
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ touch -d "yesterday" code11
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ ls -l code11
-rw-r--r-- 1 cdac cdac 253 Mar  1 15:28 code11
```

-f (ignored)

This option is present for compatibility with older systems, but it does nothing in modern Linux versions. It behaves exactly like touch myfile.txt.

-h, --no-dereference

Used when working with symbolic links. Normally, touch updates the timestamp of the target file a symlink points to. With -h, touch updates the symlink itself, not the target. Updates the timestamp of link.txt (the symlink itself), not original.txt.

- **ls -a**
  list directory contents
  -a, --all
  do not ignore entries starting with .
  -A, --almost-all
  do not list implied . and ..
  --author
  with -l, print the author of each file
  -b, --escape
  print C-style escapes for nongraphic characters

- **rm file.txt**
  remove files or directories
  -f, --force
  ignore nonexistent files and arguments, never prompt
  -i prompt before every removal
  -I prompt once before removing more than three files, or when removing recursively;
  less intrusive than

- **cp file1.txt file2.txt**
  copy files and directories
  -a, --archive
  same as -dR --preserve=all
  --attributes-only
  don't copy the file data, just the attributes
  --backup[=CONTROL]
  make a backup of each existing destination file
  -b like --backup but does not accept an argument
  --copy-contents
  copy contents of special files when recursive

- **mv file.txt /path/to/directory/**
  move (rename) files
  -b like --backup but does not accept an argument
  --debug
  explain how a file is copied. Implies -v
  -f, --force
  do not prompt before overwriting
  -i, --interactive
  prompt before overwrite

- **chmod 755 script.sh**
  - change file mode bits
  - chmod changes the file mode bits of each given file ac-cording to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.
  - The format of a symbolic mode is [ugoa...][[-+=][perms...]...], where perms is either zero or more letters from the set rwxXst, or a single letter from the set ugo. Multiple symbolic modes can be given, separated by commas.
  - A combination of the letters ugoa controls which users' access to the file will be changed: the user who owns it (u), other users in the file's group (g), other users not in the file's group (o), or all users (a). If +none of these are given, the effect is as if (a) were given, but bits that are set in the umask are not affected.
  - The operator + causes the selected file mode bits to be added to the existing file mode bits of each file; - causes them to be removed; and = causes them to be added and causes unmentioned bits to be removed except that a directory's unmentioned set user and group ID bits are not affected.

- **grep "pattern" file.txt**
  - grep, egrep, fgrep, rgrep - print lines that match patterns
  - It includes the variant programs egrep, fgrep and rgrep. These programs are the same as grep -E, grep -F, and grep -r, respectively. These variants are deprecated upstream, but Debian provides for backward compatibility. For portability reasons, it is recommended to avoid the variant programs, and use grep with the related option instead.
- **kill PID**
  - kill - send a signal to a process
  - kill PID is used to terminate a process with a specific Process ID (PID).

- **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**
  - mkdir mydir, it will create mydir directory.
  - cd mydir, change the directory to mydir.
  - touch file.txt, create a file in mydir directory.
  - echo prints Hello World!
  - '>' will move input to file.txt
  - cat file.txt will print Hello World! in console.

- **ls -l | grep ".txt"**
  Lists files and directories in the current directory in long format (showing details like permissions, owner, size, and modification date). grep ".txt" filters the output to show only files containing .txt in their name.

- **cat file1.txt file2.txt | sort | uniq**
  Displays (concatenates) the contents of file1.txt and file2.txt. Sorts the combined output alphabetically. Removes duplicate lines (only works on adjacent duplicates, so sorting is necessary).

- **ls -l | grep "^d"**
  Lists all files and directories in long format (showing permissions, owner, size, and modification date). grep "^d" → Filters only directories from the output.

- **grep -r "pattern" /path/to/directory/**
  grep searches for specific pattern. '-r' in recursively subdirectories also. "Pattern" text to search. And the path where it searches.

- **cat file1.txt file2.txt | sort | uniq –d**
  Combines (concatenates) the contents of file1.txt and file2.txt. Then sorts the combined content alphabetically. Uniq will display only the duplicate lines (i.e., lines that appear in both files).
- **chmod 644 file.txt**
  change mod will change the permissions of User, Group & Others. 4- read, 2-write, 1-execute. User- read & write, Group – read only, Others – read only.

- **cp -r source_directory destination_directory**
  Copies files and directories. -r (Recursive) → Copies all files and subdirectories from source_directory to destination_directory.

- **find /path/to/search -name "*.txt"**
  find searches for files and directories recursively. Path the starting directory where the search begins. "*.txt" searchs all txt files.

- **chmod u+x file.txt**
  Change the mode of user permission to execute for the file file.txt.

- **echo $PATH**
  Displays the system's executable search paths (directories where the shell looks for commands).

PART B

Identify True or False

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

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

3. **cd is used to copy files and directories.**
   - False
   - cd (change directory) is used to navigate between directories, not to copy files, cp is used to copy files and directories.

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

5. **grep is used to search for patterns in files.**
   - True

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

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

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

Identify the Incorrect Commands

1. **chmodx is used to change file permissions.**
   - chmod is used to change file permissions, not chmodx.

2. **cpy is used to copy files and directories.**
   - cp is used to copy files & directories, not cpy.

3. **mkfile is used to create a new file.**
   - Linux, use: touch filename, also nano filename creates a file and allows directly to edit it.
   - macOS, mkfile is a valid command to create a file of a specific size.

4. **catx is used to concatenate files.**
   - cat is used to concatenate files, not catx.

5. **rn is used to rename files.**
   - mv is used to rename file.

PART C

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

```
cdac@kalyanidivyani:~$ echo Hello World!
Hello World!
```

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

```
cdac@kalyanidivyani:~$ name="CDAC Mumbai"
cdac@kalyanidivyani:~$ $name
CDAC: command not found
cdac@kalyanidivyani:~$ echo $name
CDAC Mumbai
```

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

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code3
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code3
Enter a number :
12
Entered number : 12
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code3
echo Enter a number :
read num
echo Entered number : $num
```

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

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code4
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code4
Enter a number :
12
Enter a number :
3
The sum of 12 and 3 is 15
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code4
echo Enter a number :
read num1
echo Enter a number :
read num2
sum=`expr $num1 + $num2`
echo The sum of $num1 and $num2 is $sum
```

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

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code5
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code5
Enter number to check even or odd :
12
The 12 is even.
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code5
Enter number to check even or odd :
3
The 3 is odd.
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code5
        echo Enter number to check even or odd :
read num
if [ $((num % 2)) -eq 0 ]
then
        echo The $num is even.
else
        echo The $num is odd.
fi
```

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

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code7
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code7
1
2
3
4
5
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code7
no=1
for no in {1..5}
do
echo $no
done
```

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

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code6
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code6
1
2
3
4
5
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code6
no=1
while [ $no -lt 6 ]
do
echo $no
no=`expr $no + 1`
done
```

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".

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code8
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code8
File don't exist
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ ls
code3  code4  code5  code6  code7  code8  file.txt
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code8
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code8
File Exists
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ ls
code3  code4  code5  code6  code7  code8  file.txt
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ rm file.txt
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code8
File don't exist
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code8
file="file.txt"
if [ -e "$file" ]
then
        echo "File Exists"
else
        echo "File don't exist"
fi
```

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.

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code9
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code9
Enter any number:
9
The 9 is not greater than 10.
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code9
Enter any number:
12
The 12 is greater than 10.
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code9
echo Enter any number:
read no
if [ $no -gt 10 ]
then
echo The $no is greater than 10.
else
echo The $no is not greater than 10.
fi
```

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.

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code10
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code10
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
```

```
4 * 9 = 36
4 * 10 = 40

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code10
for i in {1..5}
do
for j in {1..10}
do
echo $i "*" $j = $(( i * j ))
done
echo
done
```

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.

```
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ nano code11
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ bash code11
Enter numbers(to stop loop enter negative number):
2
Square of 2 is 4
Want to exit enter negative number..
3
Square of 3 is 9
Want to exit enter negative number..
4
Square of 4 is 16
Want to exit enter negative number..
0
Square of 0 is 0
Want to exit enter negative number..
-1
entered negative number..exiting..
cdac@kalyanidivyani:~/LinuxAssignment/Assignment2$ cat code11
echo Enter numbers"(to stop loop enter negative number)":
while true
do
read no
if [ $no -lt 0 ]
then
echo entered negative number..exiting..
break
fi
square=`expr $((no*no))`
echo Square of $no is $square
echo Want to exit enter negative number..
done
```