VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Name: Apurba Koirala
Reg no: 22BCE3799
Subject Code: BCSE303P
Course Title: Operating Systems Lab

Lab Slot: L39 + L40

Guided By: Dr. Anto S

Lab Assessment 1.

Exercise 1 – Operating System Basics

Question – I

Demonstrate various Linux commands

ls: ls lists files and directories. The -l flag provides detailed information, including permissions, owner, size, and modification date.

cd: cd changes the current directory to the specified path.

mkdir: mkdir creates a new directory.

rmdir: the rmdir command removes empty directories

touch: touch creates a new file or updates the timestamp of an existing file

cp: cp copies files to a given location

mv: mv command moves or renames files or directories

rm: rm deletes files

pwd: gives full directory path of the current location in terminal

pushd: pushd command save the current directory on a stack and then changes to a new directory

popd: popd command returns the previous directory saved on the stack by pushd command

find: find command is used to search for files or directories within the filesystem

echo: echo command is used to display a line of text or a variable's value on the terminal

cat: cat concatenates and displays the content of files

whereis: whereis command is used to locate the binary, source, and manual page files for a command

printenv: displays environment variables of the current shell

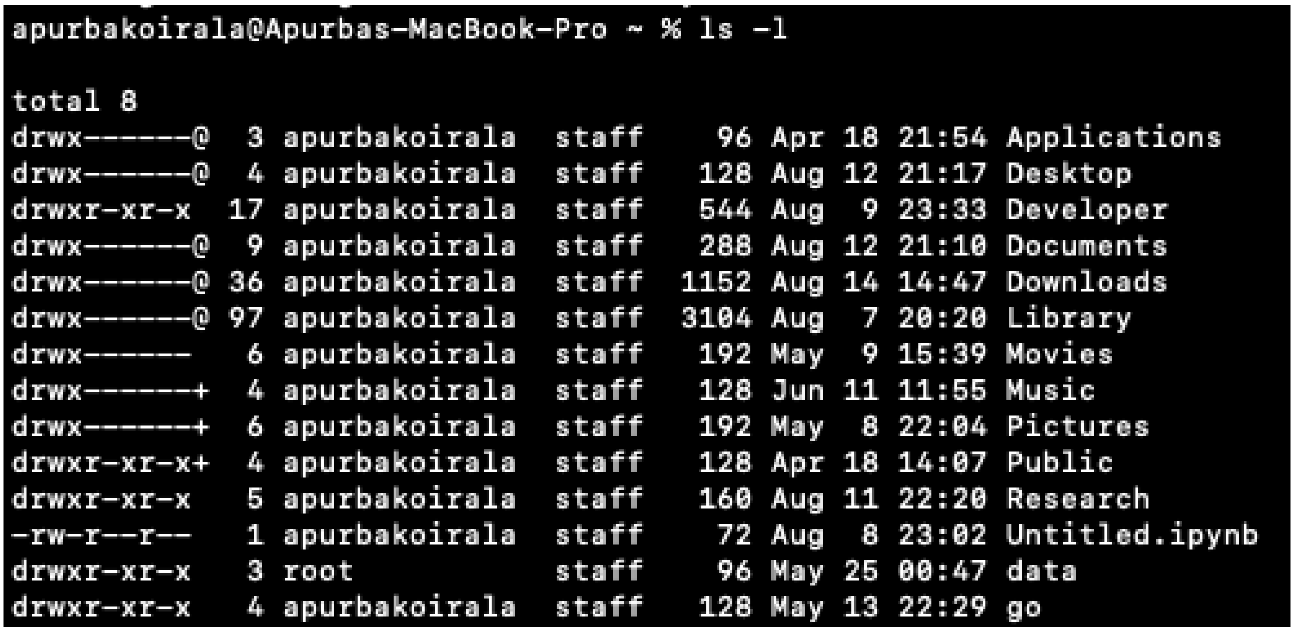
grep: is used to search for specific pattern within files or output

head: displays first few lines of a file

tail: displays last few lines of a file

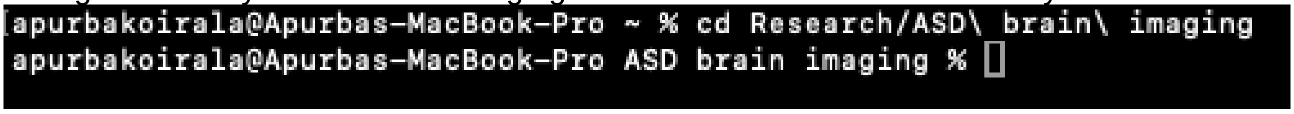
OUTPUT SCREEN SHOT:

ls:
lists all directories



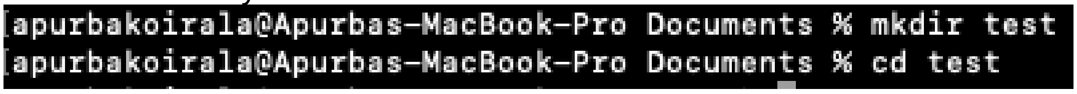
```
apurbakoirala@Apurbas-MacBook-Pro ~ % ls -l

total 8
drwx------@  3 apurbakoirala  staff    96 Apr 18 21:54 Applications
drwx------@  4 apurbakoirala  staff   128 Aug 12 21:17 Desktop
drwxr-xr-x  17 apurbakoirala  staff   544 Aug  9 23:33 Developer
drwx------@  9 apurbakoirala  staff   288 Aug 12 21:10 Documents
drwx------@ 36 apurbakoirala  staff  1152 Aug 14 14:47 Downloads
drwx------@ 97 apurbakoirala  staff  3104 Aug  7 20:20 Library
drwx------   6 apurbakoirala  staff   192 May  9 15:39 Movies
drwx------+  4 apurbakoirala  staff   128 Jun 11 11:55 Music
drwx------+  6 apurbakoirala  staff   192 May  8 22:04 Pictures
drwxr-xr-x+  4 apurbakoirala  staff   128 Apr 18 14:07 Public
drwxr-xr-x   5 apurbakoirala  staff   160 Aug 11 22:20 Research
-rw-r--r--   1 apurbakoirala  staff    72 Aug  8 23:02 Untitled.ipynb
drwxr-xr-x   3 root           staff    96 May 25 00:47 data
drwxr-xr-x   4 apurbakoirala  staff   128 May 13 22:29 go
```

cd:
changes directory to ASD brain imaging which is inside Research directory



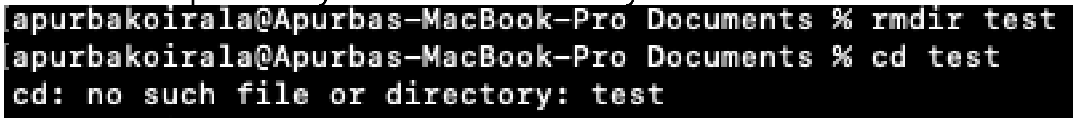
```
apurbakoirala@Apurbas-MacBook-Pro ~ % cd Research/ASD\ brain\ imaging
apurbakoirala@Apurbas-MacBook-Pro ASD brain imaging %
```

mkdir
makes a directory named test



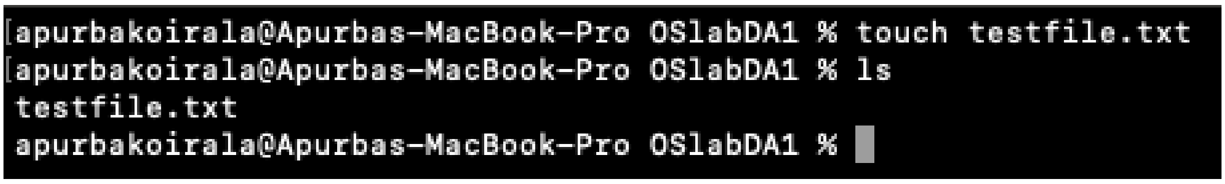
```
apurbakoirala@Apurbas-MacBook-Pro Documents % mkdir test
apurbakoirala@Apurbas-MacBook-Pro Documents % cd test
```

rmdir:
removes the previously created test directory



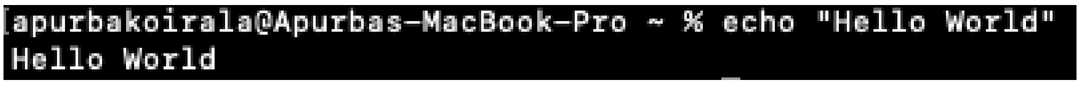
```
apurbakoirala@Apurbas-MacBook-Pro Documents % rmdir test
apurbakoirala@Apurbas-MacBook-Pro Documents % cd test
cd: no such file or directory: test
```

touch:
creates file named testfile.txt



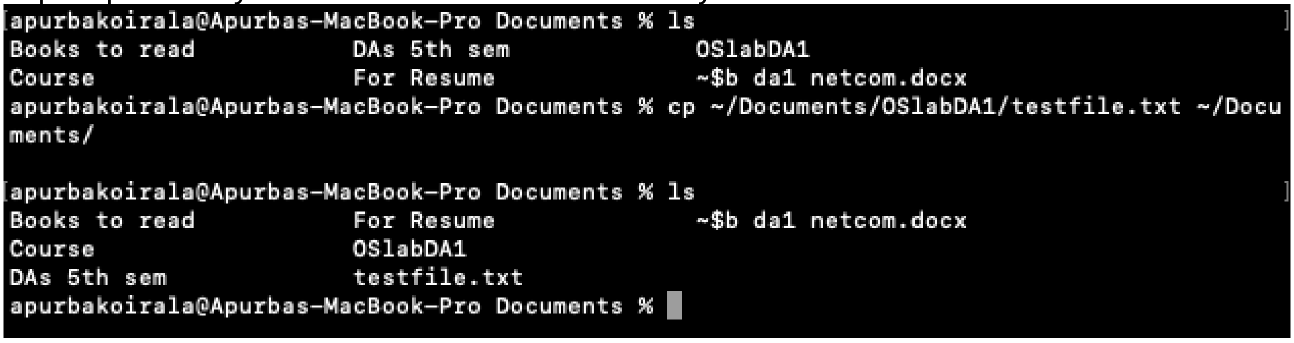
```
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % touch testfile.txt
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ls
testfile.txt
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

echo: a text is displayed on terminal window



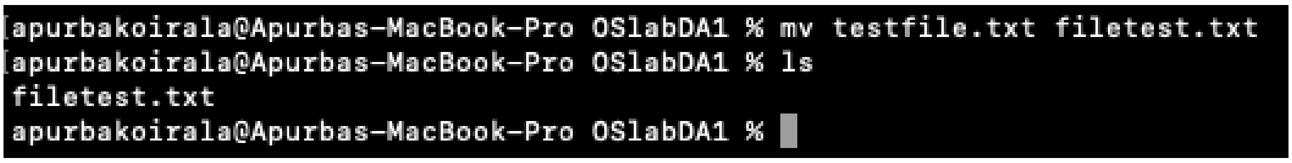
```
apurbakoirala@Apurbas-MacBook-Pro ~ % echo "Hello World"
Hello World
```

cp:
copies previously created testfile.txt from directory OSlabDA1 to Documents



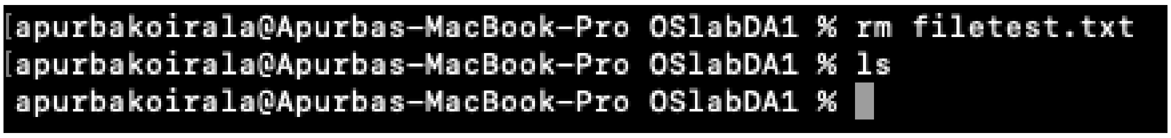
```
[apurbakoirala@Apurbas-MacBook-Pro Documents % ls                                    ]
Books to read           DAs 5th sem             OSlabDA1
Course                  For Resume              ~$b da1 netcom.docx
apurbakoirala@Apurbas-MacBook-Pro Documents % cp ~/Documents/OSlabDA1/testfile.txt ~/Docu
ments/

[apurbakoirala@Apurbas-MacBook-Pro Documents % ls                                    ]
Books to read           For Resume              ~$b da1 netcom.docx
Course                  OSlabDA1
DAs 5th sem             testfile.txt
apurbakoirala@Apurbas-MacBook-Pro Documents %
```

mv:
renames testfile.txt to filetest.txt



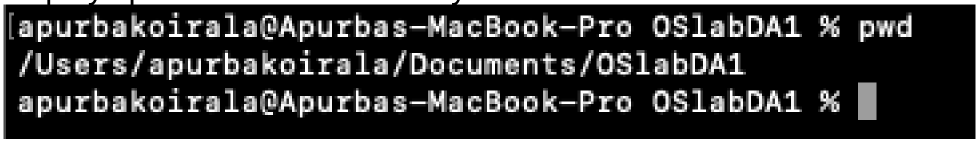
```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % mv testfile.txt filetest.txt
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ls
filetest.txt
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

rm:
removes filetest.txt



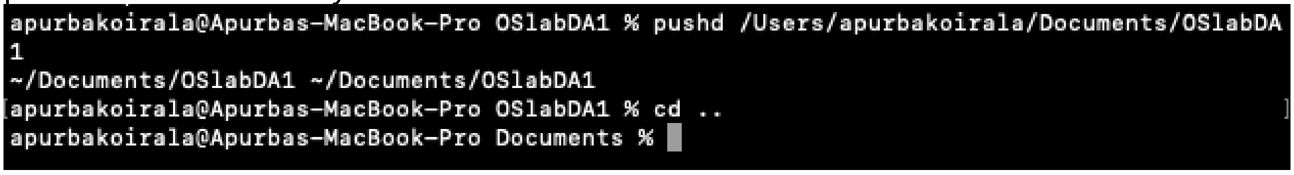
```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % rm filetest.txt
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ls
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

pwd:
displays path to current directory



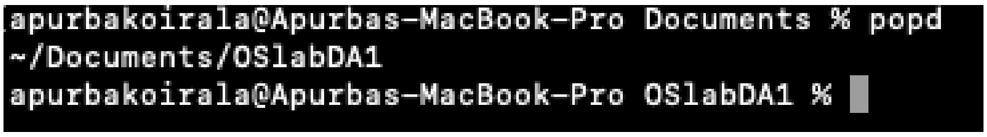
```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % pwd
/Users/apurbakoirala/Documents/OSlabDA1
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

pushd:
pushes current directory to the stack



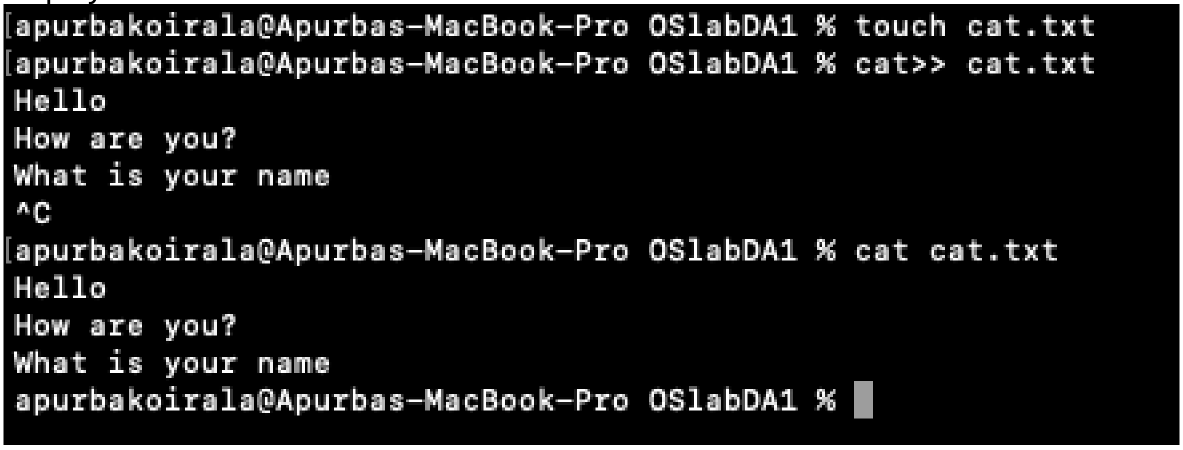
```
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % pushd /Users/apurbakoirala/Documents/OSlabDA
1
~/Documents/OSlabDA1 ~/Documents/OSlabDA1
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % cd ..                                  ]
apurbakoirala@Apurbas-MacBook-Pro Documents %
```

popd: directory is changed back to the previously pushed directory as it is popped



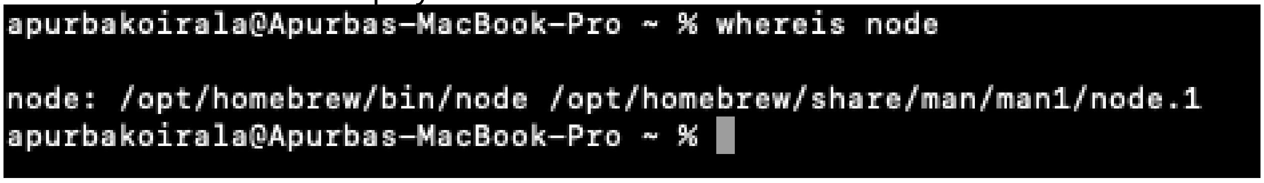
```
[apurbakoirala@Apurbas-MacBook-Pro Documents % popd
~/Documents/OSlabDA1
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

cat: a new file named cat.txt is made, text is entered using the cat command and displayed.



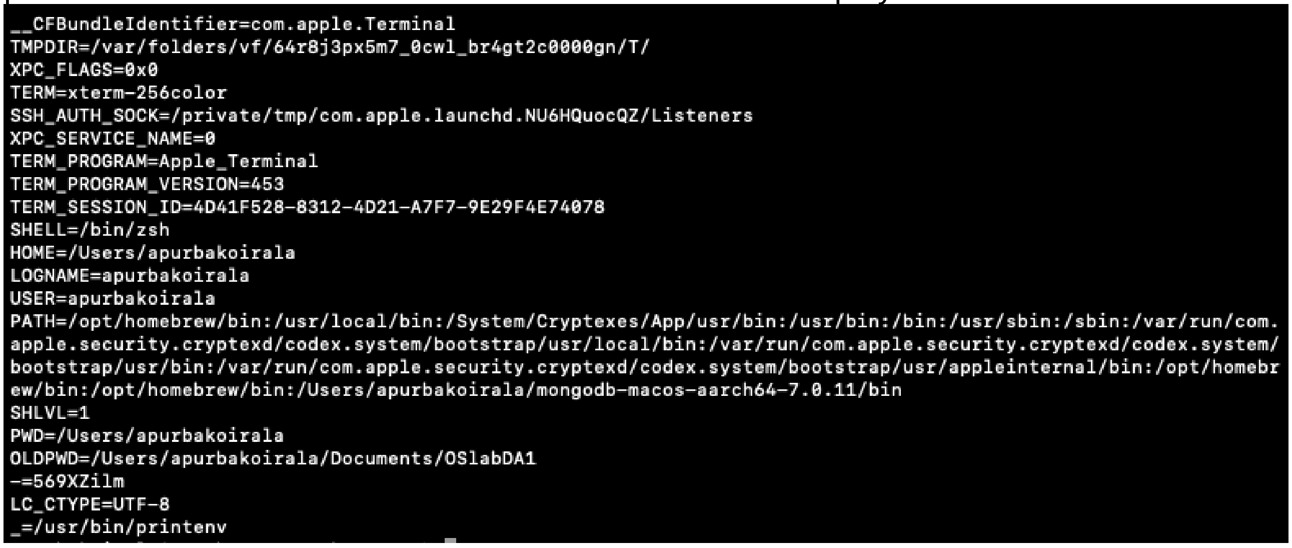
```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % touch cat.txt
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % cat>> cat.txt
Hello
How are you?
What is your name
^C
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % cat cat.txt
Hello
How are you?
What is your name
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

whereis: node location displayed



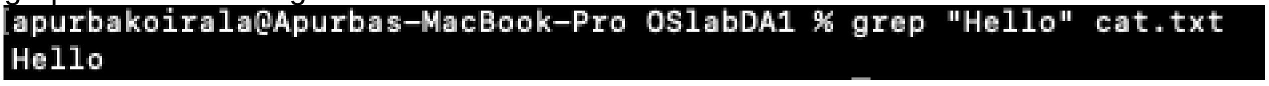
```
apurbakoirala@Apurbas-MacBook-Pro ~ % whereis node

node: /opt/homebrew/bin/node /opt/homebrew/share/man/man1/node.1
apurbakoirala@Apurbas-MacBook-Pro ~ %
```

printenv: environment variables of current shell session displayed



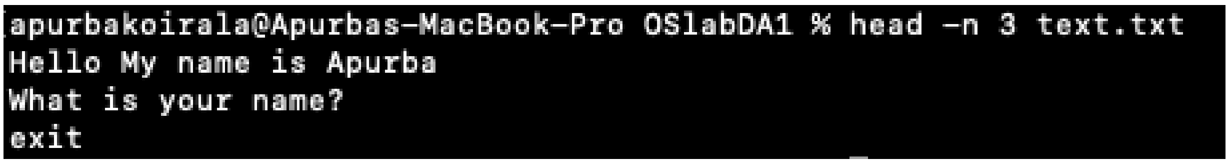
```
__CFBundleIdentifier=com.apple.Terminal
TMPDIR=/var/folders/vf/64r8j3px5m7_0cwl_br4gt2c0000gn/T/
XPC_FLAGS=0x0
TERM=xterm-256color
SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.NU6HQuocQZ/Listeners
XPC_SERVICE_NAME=0
TERM_PROGRAM=Apple_Terminal
TERM_PROGRAM_VERSION=453
TERM_SESSION_ID=4D41F528-8312-4D21-A7F7-9E29F4E74078
SHELL=/bin/zsh
HOME=/Users/apurbakoirala
LOGNAME=apurbakoirala
USER=apurbakoirala
PATH=/opt/homebrew/bin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.
apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/
bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/homebr
ew/bin:/opt/homebrew/bin:/Users/apurbakoirala/mongodb-macos-aarch64-7.0.11/bin
SHLVL=1
PWD=/Users/apurbakoirala
OLDPWD=/Users/apurbakoirala/Documents/OSlabDA1
-=569XZilm
LC_CTYPE=UTF-8
_=/usr/bin/printenv
```

grep: lines containing Hello in cat.txt are shown



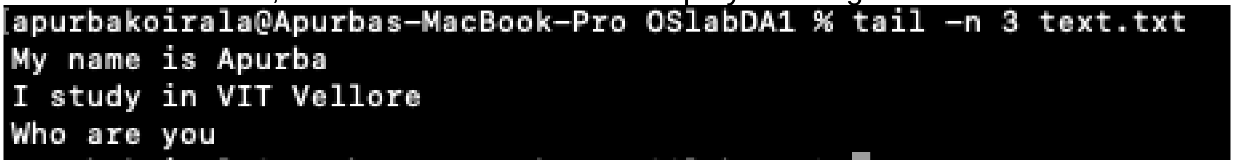
```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % grep "Hello" cat.txt
Hello
```

head: more text added to text.txt. The first three lines displayed using the head command



```
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % head -n 3 text.txt
Hello My name is Apurba
What is your name?
exit
```

tail: similar to head, last three lines of text.txt displayed using the tail command



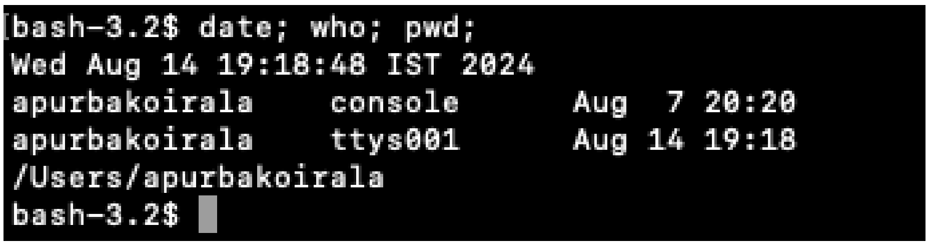
```
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % tail -n 3 text.txt
My name is Apurba
I study in VIT Vellore
Who are you
```

QUESTION– II – Shell Scripts

1. Write Script to see current date, time, username, and current directory
SOURCE CODE:

```
date; who; pwd;
```

OUTPUT :



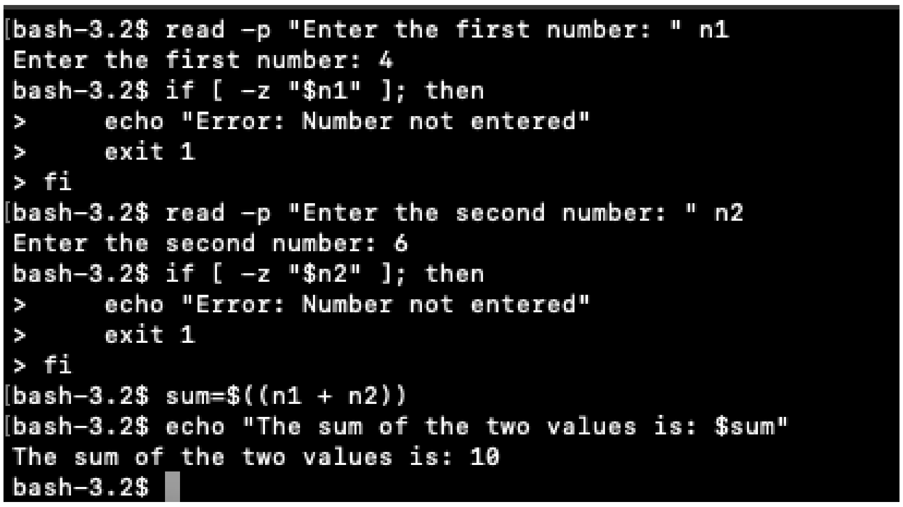
```
bash-3.2$ date; who; pwd;
Wed Aug 14 19:18:48 IST 2024
apurbakoirala    console      Aug  7 20:20
apurbakoirala    ttys001      Aug 14 19:18
/Users/apurbakoirala
bash-3.2$
```

2. How to write shell script that will add two numbers, which are supplied as command line argument, and if this two numbers are not given show error and its usage.

SOURCE CODE:

```
read -p "Enter the first number: " n1
if [ -z "$n1" ]; then
   echo "Error: Number not entered"
   exit 1
fi
read -p "Enter the second number: " n2
if [ -z "$n2" ]; then
   echo "Error: Number not entered"
   exit 1
fi
sum=$((n1 + n2))
echo "The sum of the two values is: $sum"
```

OUTPUT :

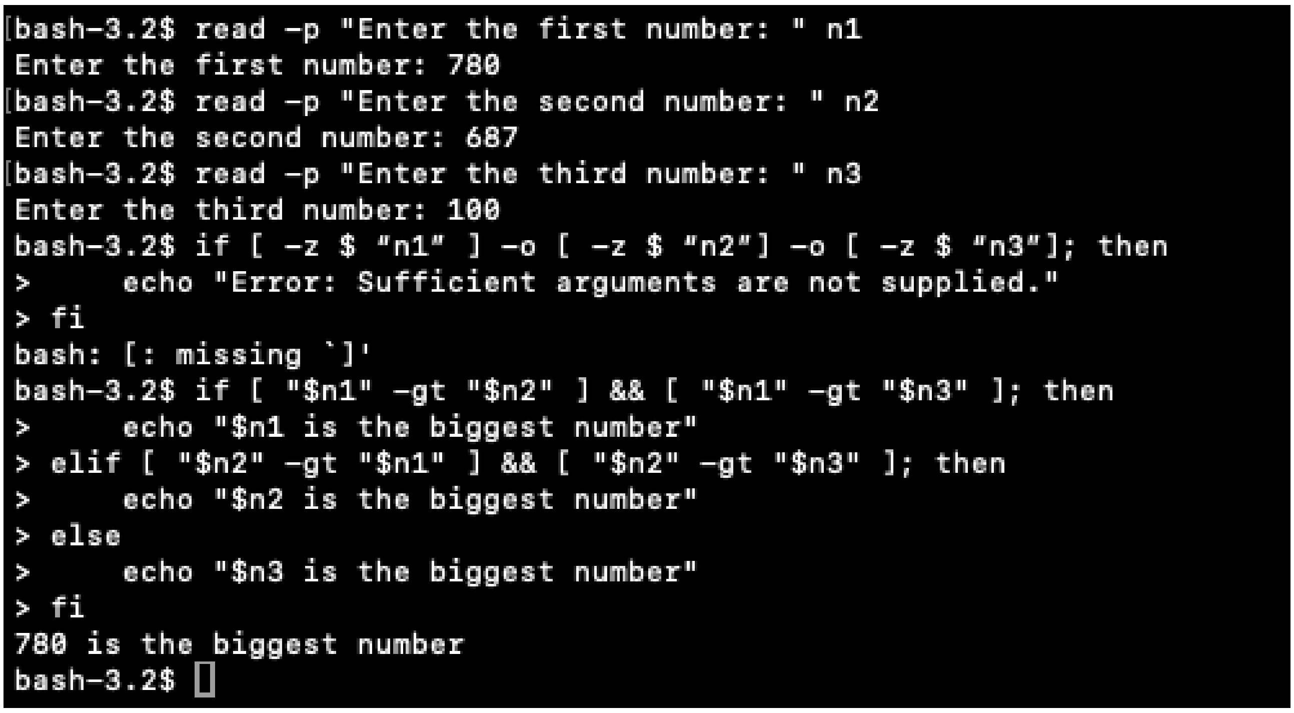

```
[bash-3.2$ read -p "Enter the first number: " n1
Enter the first number: 4
bash-3.2$ if [ -z "$n1" ]; then
>     echo "Error: Number not entered"
>     exit 1
> fi
[bash-3.2$ read -p "Enter the second number: " n2
Enter the second number: 6
bash-3.2$ if [ -z "$n2" ]; then
>     echo "Error: Number not entered"
>     exit 1
> fi
[bash-3.2$ sum=$((n1 + n2))
[bash-3.2$ echo "The sum of the two values is: $sum"
The sum of the two values is: 10
bash-3.2$
```

3. Write Script to find out biggest number from given three nos. Numbers are supplied as command line argument. Print error if sufficient arguments are not supplied.

SOURCE CODE:

```
#!/bin/bash

read -p "Enter the first number: " n1
read -p "Enter the second number: " n2
read -p "Enter the third number: " n3
if [ -z $ “n1” ] -o [ -z $ “n2”] -o [ -z $ “n3”]; then
   echo "Error: Sufficient arguments are not supplied."
fi
if [ "$n1" -gt "$n2" ] && [ "$n1" -gt "$n3" ]; then
   echo "$n1 is the biggest number"
elif [ "$n2" -gt "$n1" ] && [ "$n2" -gt "$n3" ]; then
   echo "$n2 is the biggest number"
else
   echo "$n3 is the biggest number"
fi
```

OUTPUT :



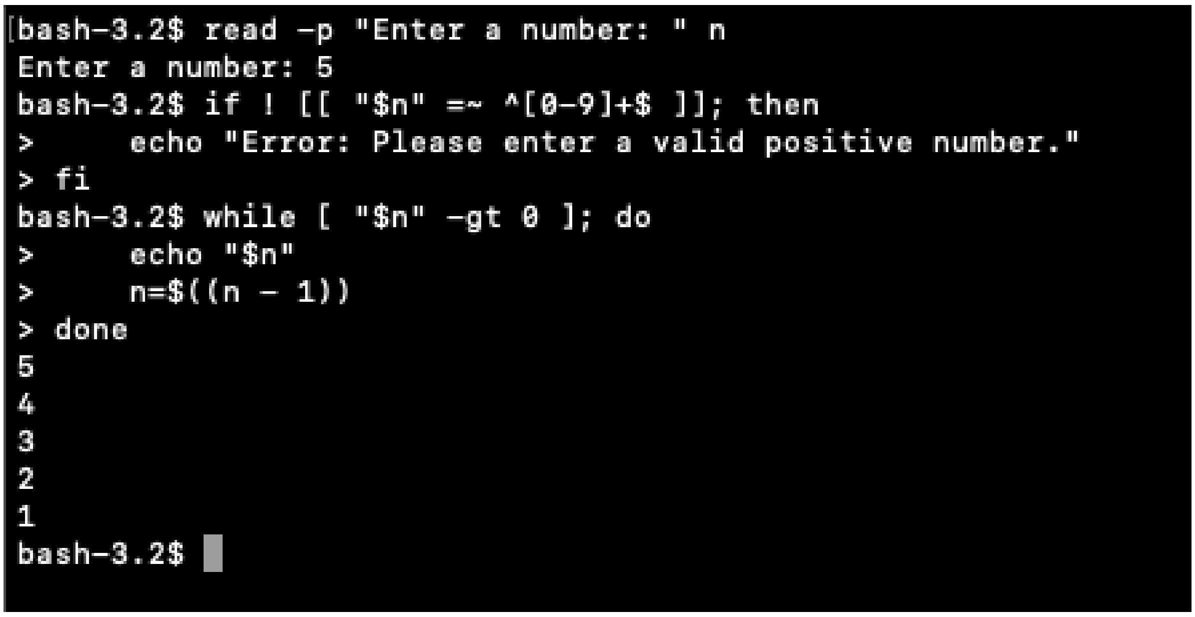
```
[bash-3.2$ read -p "Enter the first number: " n1
Enter the first number: 780
[bash-3.2$ read -p "Enter the second number: " n2
Enter the second number: 687
[bash-3.2$ read -p "Enter the third number: " n3
Enter the third number: 100
bash-3.2$ if [ -z $ “n1” ] -o [ -z $ “n2”] -o [ -z $ “n3”]; then
>     echo "Error: Sufficient arguments are not supplied."
> fi
bash: [: missing `]'
bash-3.2$ if [ "$n1" -gt "$n2" ] && [ "$n1" -gt "$n3" ]; then
>     echo "$n1 is the biggest number"
> elif [ "$n2" -gt "$n1" ] && [ "$n2" -gt "$n3" ]; then
>     echo "$n2 is the biggest number"
> else
>     echo "$n3 is the biggest number"
> fi
780 is the biggest number
bash-3.2$ 
```

4. Write script to print the following numbers as 5,4,3,2,1 using while loop.

SOURCE CODE:

```
read -p "Enter a number: " n
if ! [[ "$n" =~ ^[0-9]+$ ]]; then
   echo "Error: Please enter a valid positive number."
   exit 1
fi
while [ "$n" -gt 0 ]; do
   echo "$n"
   n=$((n - 1))
done
```

OUTPUT :



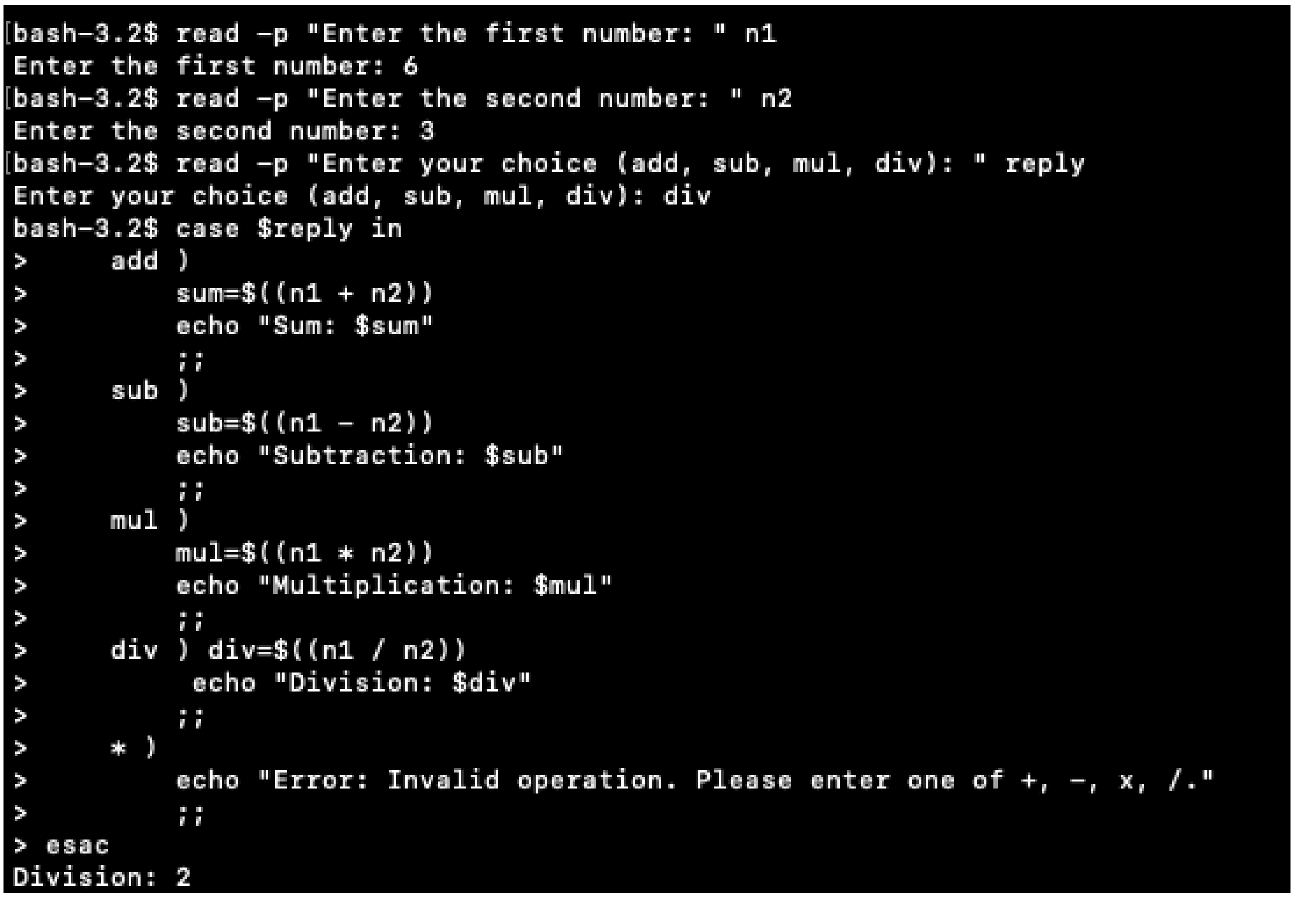
```
[bash-3.2$ read -p "Enter a number: " n
Enter a number: 5
bash-3.2$ if ! [[ "$n" =~ ^[0-9]+$ ]]; then
>     echo "Error: Please enter a valid positive number."
> fi
bash-3.2$ while [ "$n" -gt 0 ]; do
>     echo "$n"
>     n=$((n - 1))
> done
5
4
3
2
1
bash-3.2$ 
```

5. Write Script, using case statement to perform basic math operation as follows: + addition, - subtraction, x multiplication, / division

SOURCE CODE:

```
read -p "Enter the first number: " n1
read -p "Enter the second number: " n2
read -p "Enter your choice (add, sub, mul, div): " reply
case $reply in
   add )
       sum=$((n1 + n2))
       echo "Sum: $sum"
       ;;
   sub )
       sub=$((n1 - n2))
       echo "Subtraction: $sub"
       ;;
   mul )
       mul=$((n1 * n2))
       echo "Multiplication: $mul"
       ;;
   div ) div=$((n1 / n2))
        echo "Division: $div"
       ;;
   * )
       echo "Error: Invalid operation. Please enter one of +, -, x, /."
       ;;
esac
```

OUTPUT :



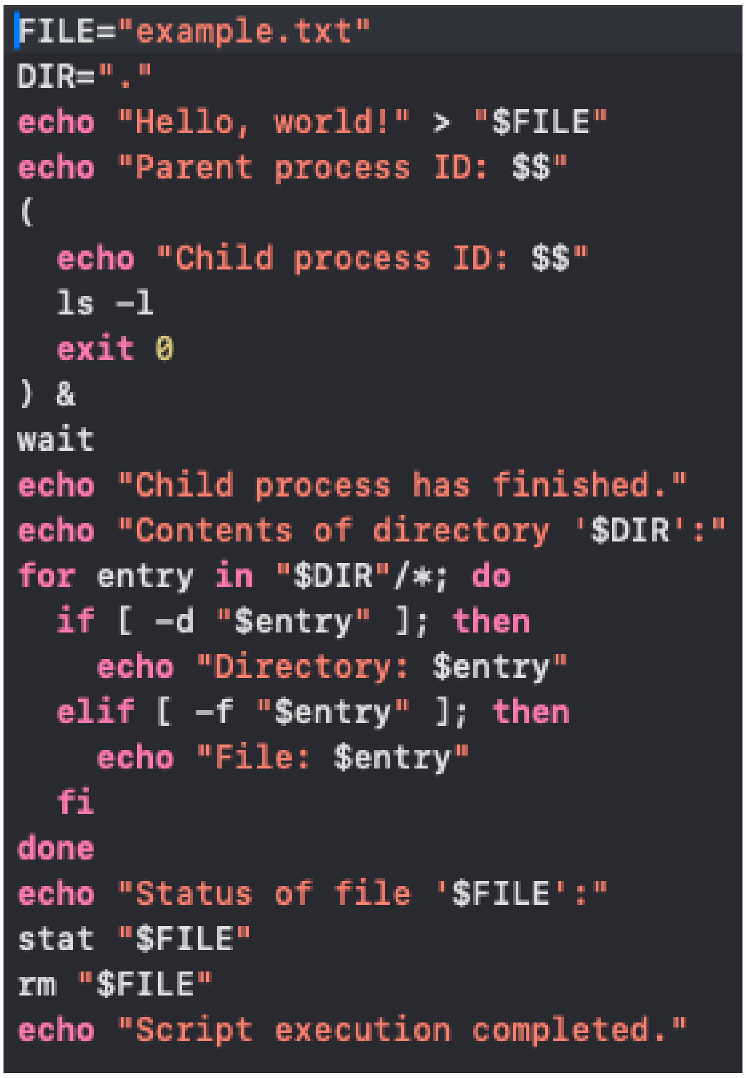
```
[bash-3.2$ read -p "Enter the first number: " n1
Enter the first number: 6
[bash-3.2$ read -p "Enter the second number: " n2
Enter the second number: 3
[bash-3.2$ read -p "Enter your choice (add, sub, mul, div): " reply
Enter your choice (add, sub, mul, div): div
bash-3.2$ case $reply in
>     add )
>         sum=$((n1 + n2))
>         echo "Sum: $sum"
>         ;;
>     sub )
>         sub=$((n1 - n2))
>         echo "Subtraction: $sub"
>         ;;
>     mul )
>         mul=$((n1 * n2))
>         echo "Multiplication: $mul"
>         ;;
>     div ) div=$((n1 / n2))
>          echo "Division: $div"
>         ;;
>     * )
>         echo "Error: Invalid operation. Please enter one of +, -, x, /."
>         ;;
> esac
Division: 2
```

QUESTION 3 – System Calls
1. Write Programs using the following system calls of LINUX operating system:
a. fork, exec, getpid, exit, wait, close, opendir, readdir, stat

SOURCE CODE:
FILE="example.txt"

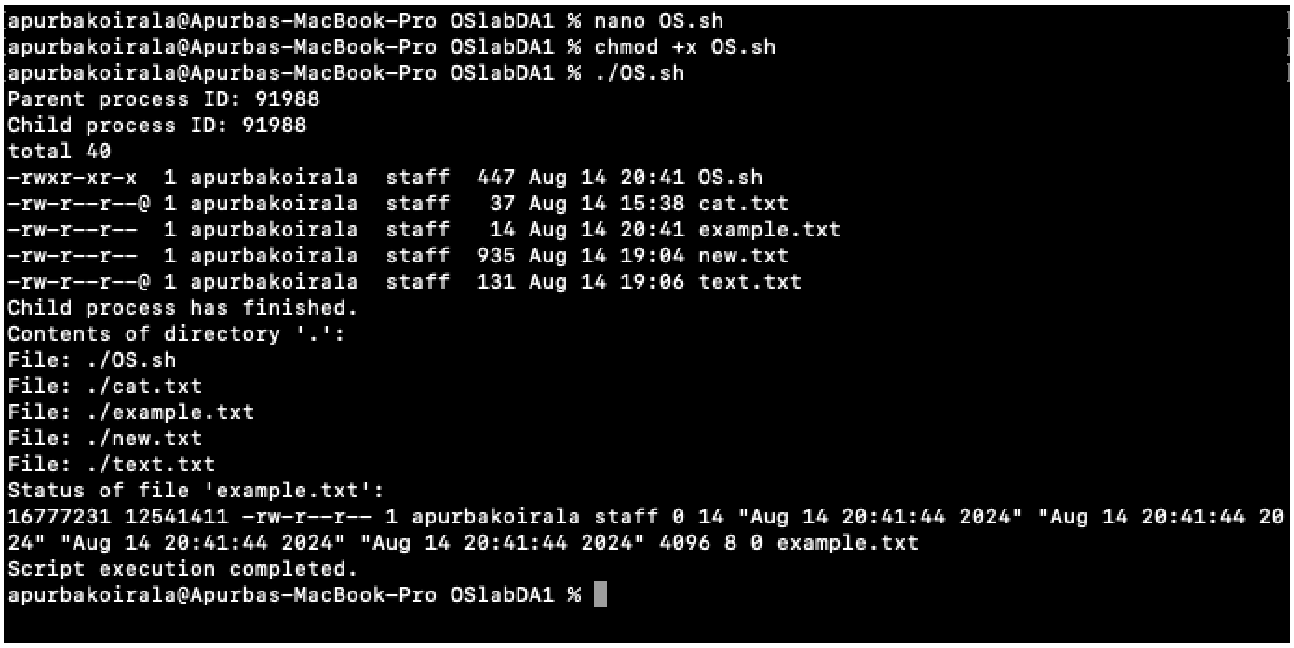
```
DIR="."
echo "Hello, world!" > "$FILE"
echo "Parent process ID: $$"
(
  echo "Child process ID: $$"
  ls -l
  exit 0
) &
wait
echo "Child process has finished."
echo "Contents of directory '$DIR':"
for entry in "$DIR"/*; do
  if [ -d "$entry" ]; then
          echo "Directory: $entry"
  elif [ -f "$entry" ]; then
          echo "File: $entry"
  fi
done
echo "Status of file '$FILE':"
stat "$FILE"
rm "$FILE"
echo "Script execution completed."
```

OUTPUT :



```
FILE="example.txt"
DIR="."
echo "Hello, world!" > "$FILE"
echo "Parent process ID: $$"
(
  echo "Child process ID: $$"
  ls -l
  exit 0
) &
wait
echo "Child process has finished."
echo "Contents of directory '$DIR':"
for entry in "$DIR"/*; do
  if [ -d "$entry" ]; then
    echo "Directory: $entry"
  elif [ -f "$entry" ]; then
    echo "File: $entry"
  fi
done
echo "Status of file '$FILE':"
stat "$FILE"
rm "$FILE"
echo "Script execution completed."
```

RESULTS:

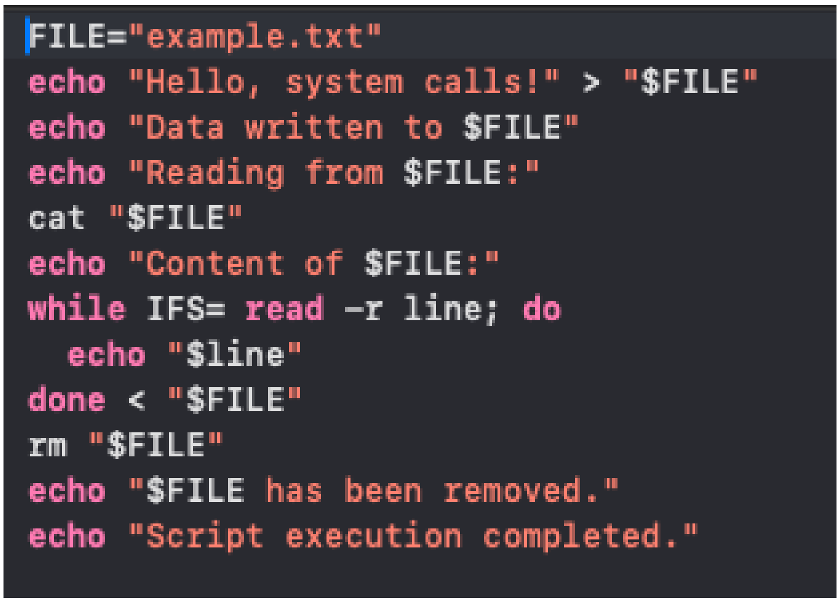


```
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % nano OS.sh
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % chmod +x OS.sh
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ./OS.sh
Parent process ID: 91988
Child process ID: 91988
total 40
-rwxr-xr-x  1 apurbakoirala  staff  447 Aug 14 20:41 OS.sh
-rw-r--r--@ 1 apurbakoirala  staff   37 Aug 14 15:38 cat.txt
-rw-r--r--  1 apurbakoirala  staff   14 Aug 14 20:41 example.txt
-rw-r--r--  1 apurbakoirala  staff  935 Aug 14 19:04 new.txt
-rw-r--r--@ 1 apurbakoirala  staff  131 Aug 14 19:06 text.txt
Child process has finished.
Contents of directory '.':
File: ./OS.sh
File: ./cat.txt
File: ./example.txt
File: ./new.txt
File: ./text.txt
Status of file 'example.txt':
16777231 12541411 -rw-r--r-- 1 apurbakoirala staff 0 14 "Aug 14 20:41:44 2024" "Aug 14 20:41:44 20
24" "Aug 14 20:41:44 2024" "Aug 14 20:41:44 2024" 4096 8 0 example.txt
Script execution completed.
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

2. Write Programs using I/O system calls of LINUX operating system:
open, read, write etc.

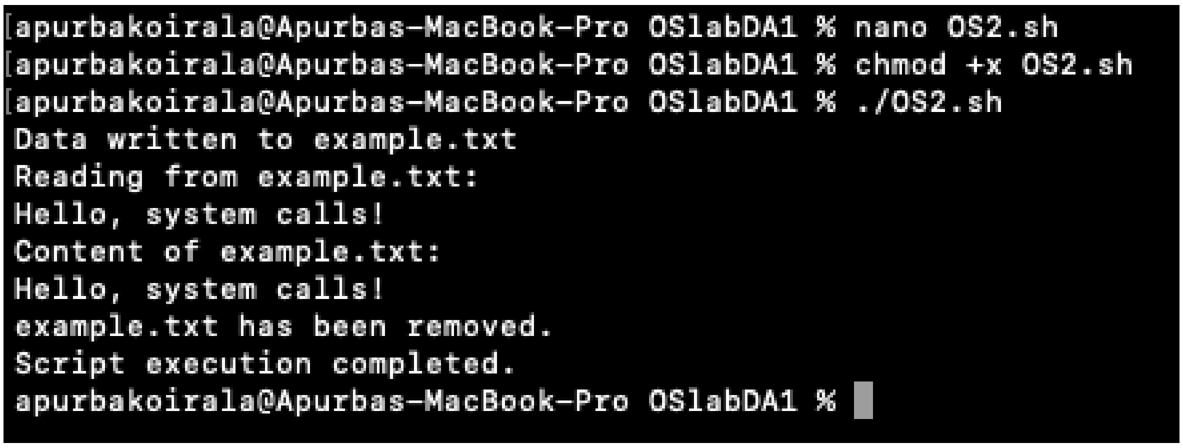
SOURCE CODE:

```
FILE="example.txt"
echo "Hello, system calls!" > "$FILE"
echo "Data written to $FILE"
echo "Reading from $FILE:"
cat "$FILE"
echo "Content of $FILE:"
while IFS= read -r line; do
  echo "$line"
done < "$FILE"
rm "$FILE"
echo "$FILE has been removed."
echo "Script execution completed."
```



```
FILE="example.txt"
echo "Hello, system calls!" > "$FILE"
echo "Data written to $FILE"
echo "Reading from $FILE:"
cat "$FILE"
echo "Content of $FILE:"
while IFS= read -r line; do
  echo "$line"
done < "$FILE"
rm "$FILE"
echo "$FILE has been removed."
echo "Script execution completed."
```

OUTPUT :
RESULTS:



```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % nano OS2.sh
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % chmod +x OS2.sh
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ./OS2.sh
Data written to example.txt
Reading from example.txt:
Hello, system calls!
Content of example.txt:
Hello, system calls!
example.txt has been removed.
Script execution completed.
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

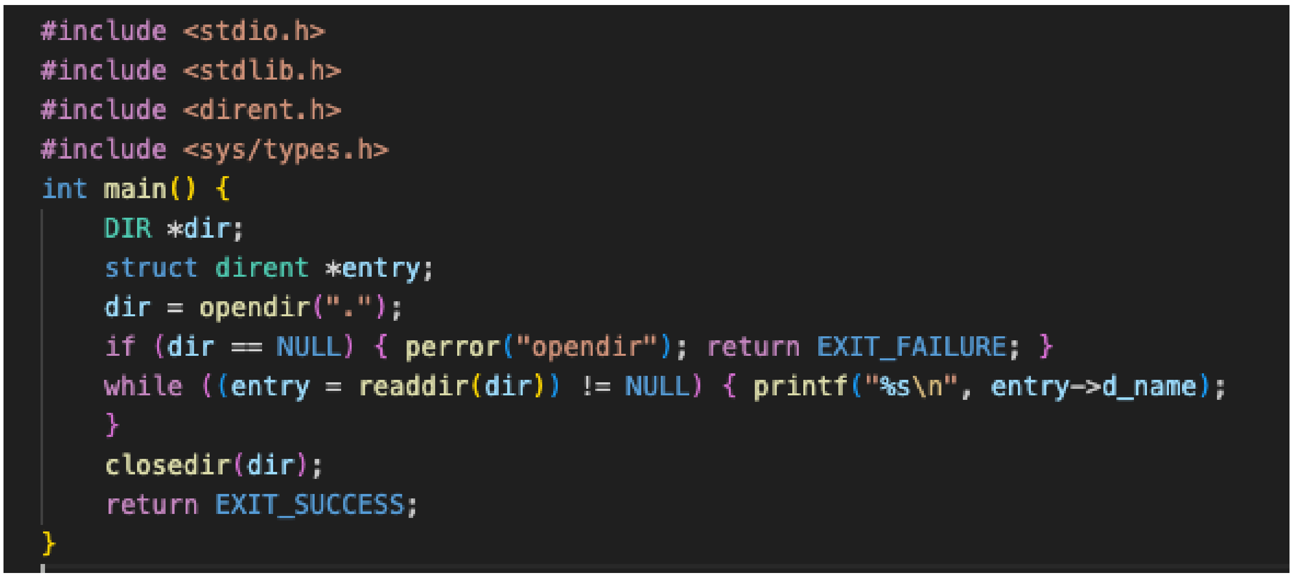
3. Write Programs using C to simulate LINUX commands

a. ls command,

SOURCE CODE:

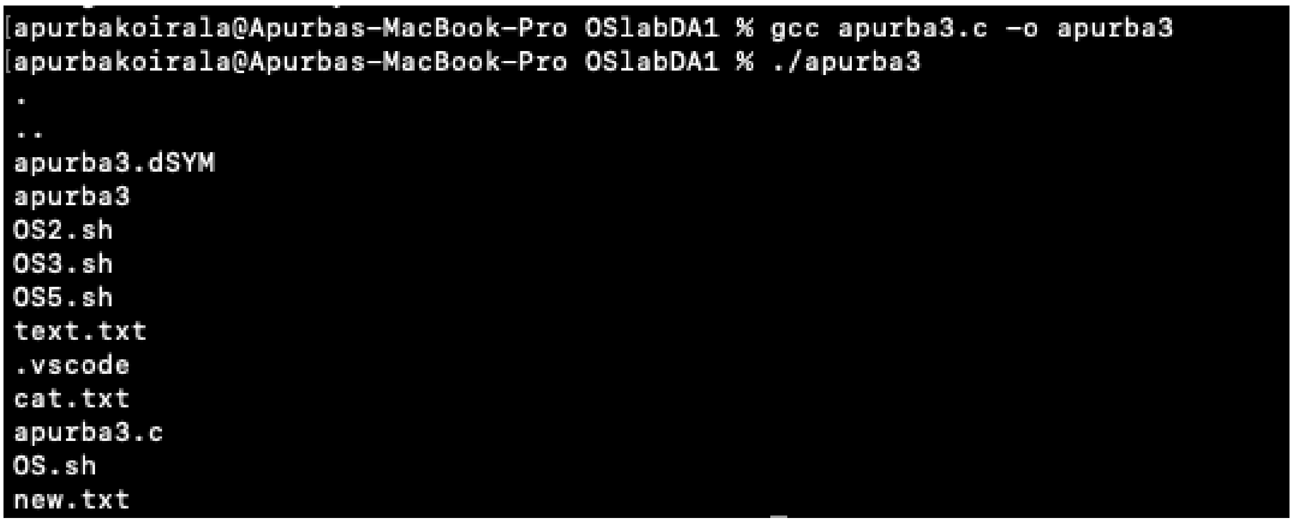
```c
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
int main() {
   DIR *dir;
   struct dirent *entry;
   dir = opendir(".");
   if (dir == NULL) { perror("opendir"); return EXIT_FAILURE; }
   while ((entry = readdir(dir)) != NULL) { printf("%s\n", entry->d_name);
   }
   closedir(dir);
   return EXIT_SUCCESS;
}
```

OUTPUT :



```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
int main() {
    DIR *dir;
    struct dirent *entry;
    dir = opendir(".");
    if (dir == NULL) { perror("opendir"); return EXIT_FAILURE; }
    while ((entry = readdir(dir)) != NULL) { printf("%s\n", entry->d_name);
    }
    closedir(dir);
    return EXIT_SUCCESS;
}
```

RESULTS:



```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % gcc apurba3.c -o apurba3
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ./apurba3
.
..
apurba3.dSYM
apurba3
OS2.sh
OS3.sh
OS5.sh
text.txt
.vscode
cat.txt
apurba3.c
OS.sh
new.txt
```

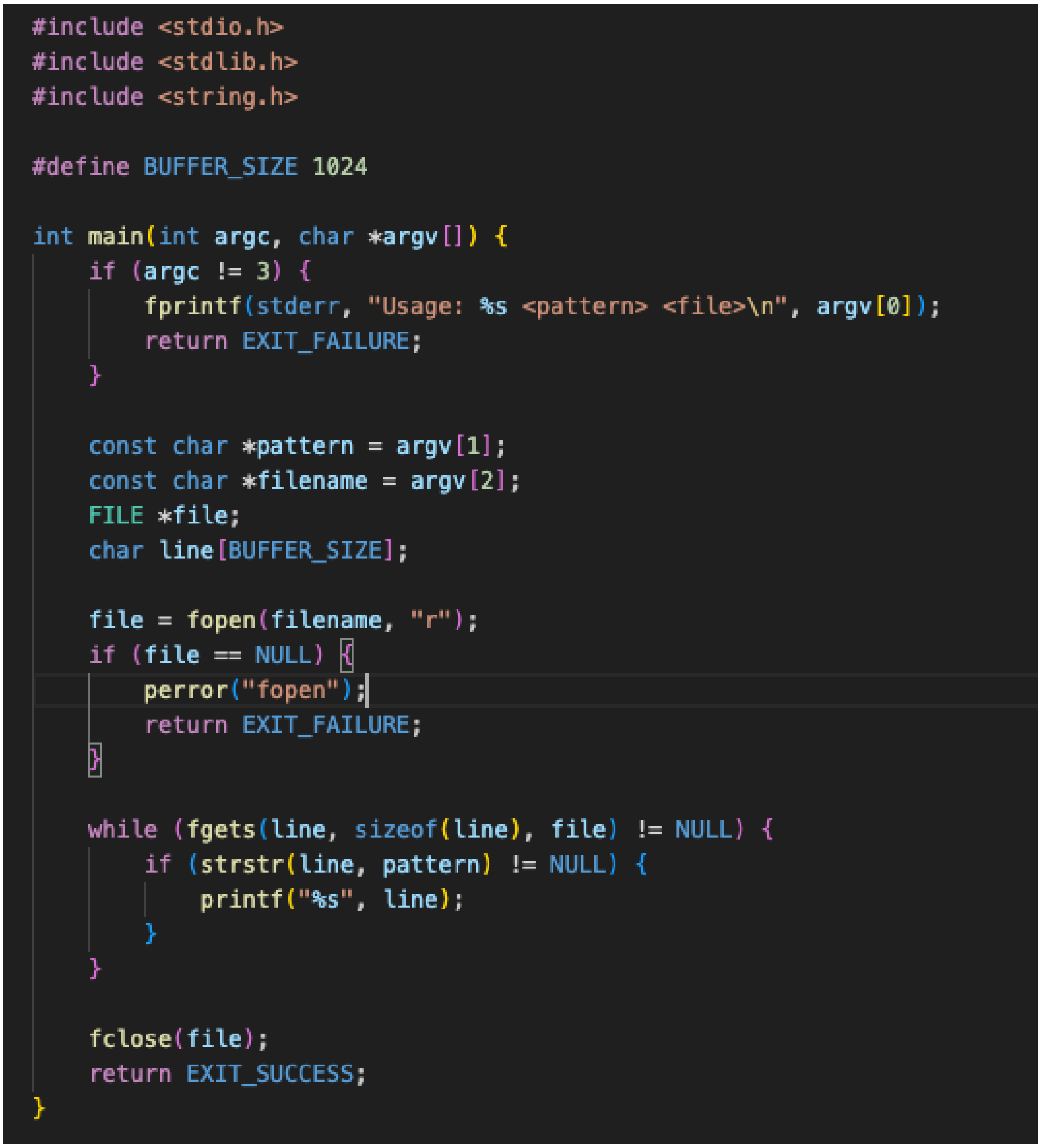
b. grep command

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 1024

int main(int argc, char *argv[]) {
   if (argc != 3) {
      fprintf(stderr, "Usage: %s <pattern> <file>\n", argv[0]);
      return EXIT_FAILURE;
   }

   const char *pattern = argv[1];
   const char *filename = argv[2];
   FILE *file;
   char line[BUFFER_SIZE];
```

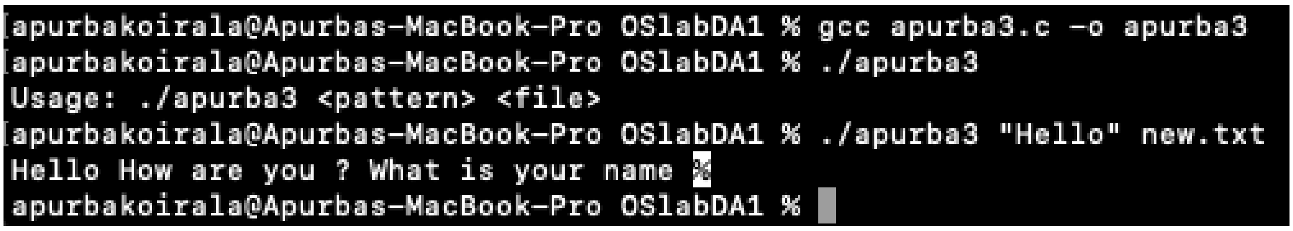
```
    file = fopen(filename, "r");
    if (file == NULL) {
        perror("fopen");
        return EXIT_FAILURE;
    }

    while (fgets(line, sizeof(line), file) != NULL) {
        if (strstr(line, pattern) != NULL) {
            printf("%s", line);
        }
    }

    fclose(file);
    return EXIT_SUCCESS;
}
```

OUTPUT:



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 1024

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <pattern> <file>\n", argv[0]);
        return EXIT_FAILURE;
    }

    const char *pattern = argv[1];
    const char *filename = argv[2];
    FILE *file;
    char line[BUFFER_SIZE];

    file = fopen(filename, "r");
    if (file == NULL) {
        perror("fopen");
        return EXIT_FAILURE;
    }

    while (fgets(line, sizeof(line), file) != NULL) {
        if (strstr(line, pattern) != NULL) {
            printf("%s", line);
        }
    }

    fclose(file);
    return EXIT_SUCCESS;
}
```

RESULTS:



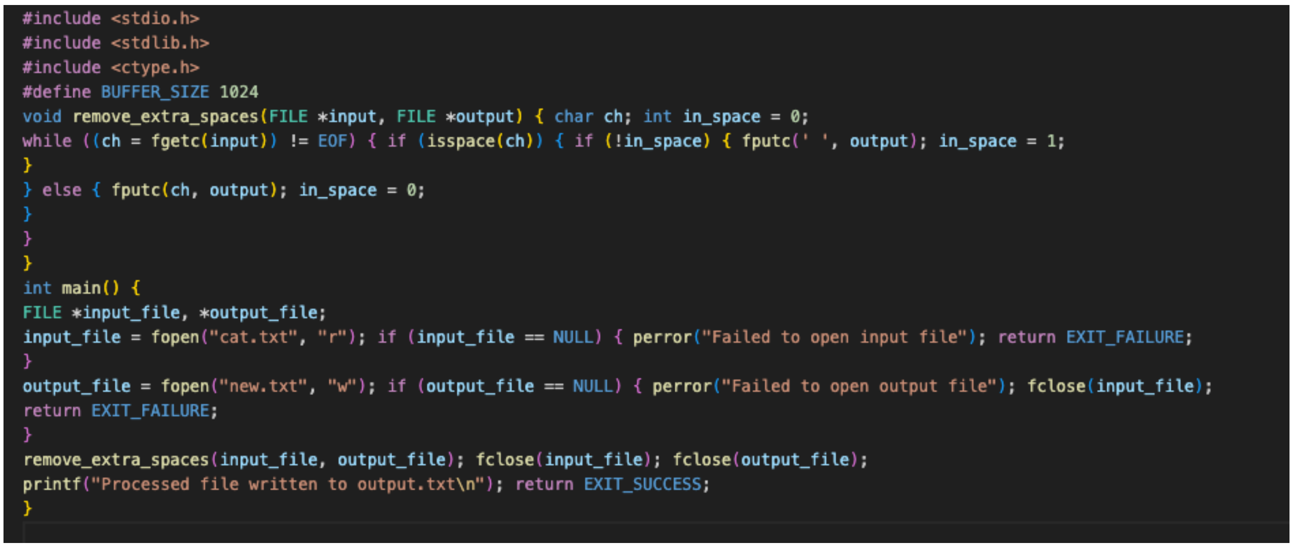
```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % gcc apurba3.c -o apurba3
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ./apurba3
Usage: ./apurba3 <pattern> <file>
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ./apurba3 "Hello" new.txt
Hello How are you ? What is your name %
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```

4. Create a file with few lines, write a C program to read the file and delete the spaces more than one in the file.

SOURCE CODE:

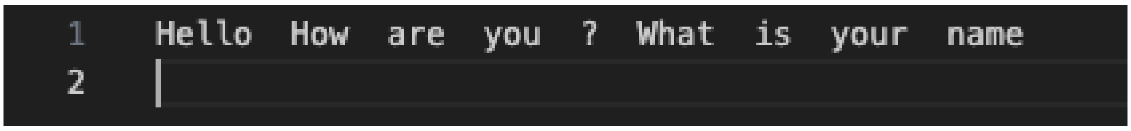
```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define BUFFER_SIZE 1024
void remove_extra_spaces(FILE *input, FILE *output) { char ch; int in_space = 0;
while ((ch = fgetc(input)) != EOF) { if (isspace(ch)) { if (!in_space) { fputc(' ', output); in_space = 1;
}
} else { fputc(ch, output); in_space = 0;
}
}
}
int main() {
FILE *input_file, *output_file;
input_file = fopen("cat.txt", "r"); if (input_file == NULL) { perror("Failed to open input file"); return EXIT_FAILURE;
}
output_file = fopen("new.txt", "w"); if (output_file == NULL) { perror("Failed to open output file"); fclose(input_file);
return EXIT_FAILURE;
}
remove_extra_spaces(input_file, output_file); fclose(input_file); fclose(output_file);
printf("Processed file written to output.txt\n"); return EXIT_SUCCESS;
}
```

OUTPUT :



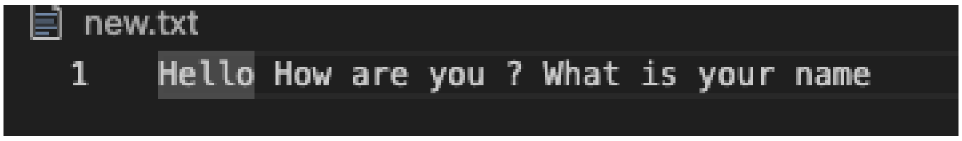
```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define BUFFER_SIZE 1024
void remove_extra_spaces(FILE *input, FILE *output) { char ch; int in_space = 0;
while ((ch = fgetc(input)) != EOF) { if (isspace(ch)) { if (!in_space) { fputc(' ', output); in_space = 1;
}
} else { fputc(ch, output); in_space = 0;
}
}
}
int main() {
FILE *input_file, *output_file;
input_file = fopen("cat.txt", "r"); if (input_file == NULL) { perror("Failed to open input file"); return EXIT_FAILURE;
}
output_file = fopen("new.txt", "w"); if (output_file == NULL) { perror("Failed to open output file"); fclose(input_file);
return EXIT_FAILURE;
}
remove_extra_spaces(input_file, output_file); fclose(input_file); fclose(output_file);
printf("Processed file written to output.txt\n"); return EXIT_SUCCESS;
}
```

Cat.txt



```
Hello   How   are   you   ?   What   is   your   name

```

New.txt



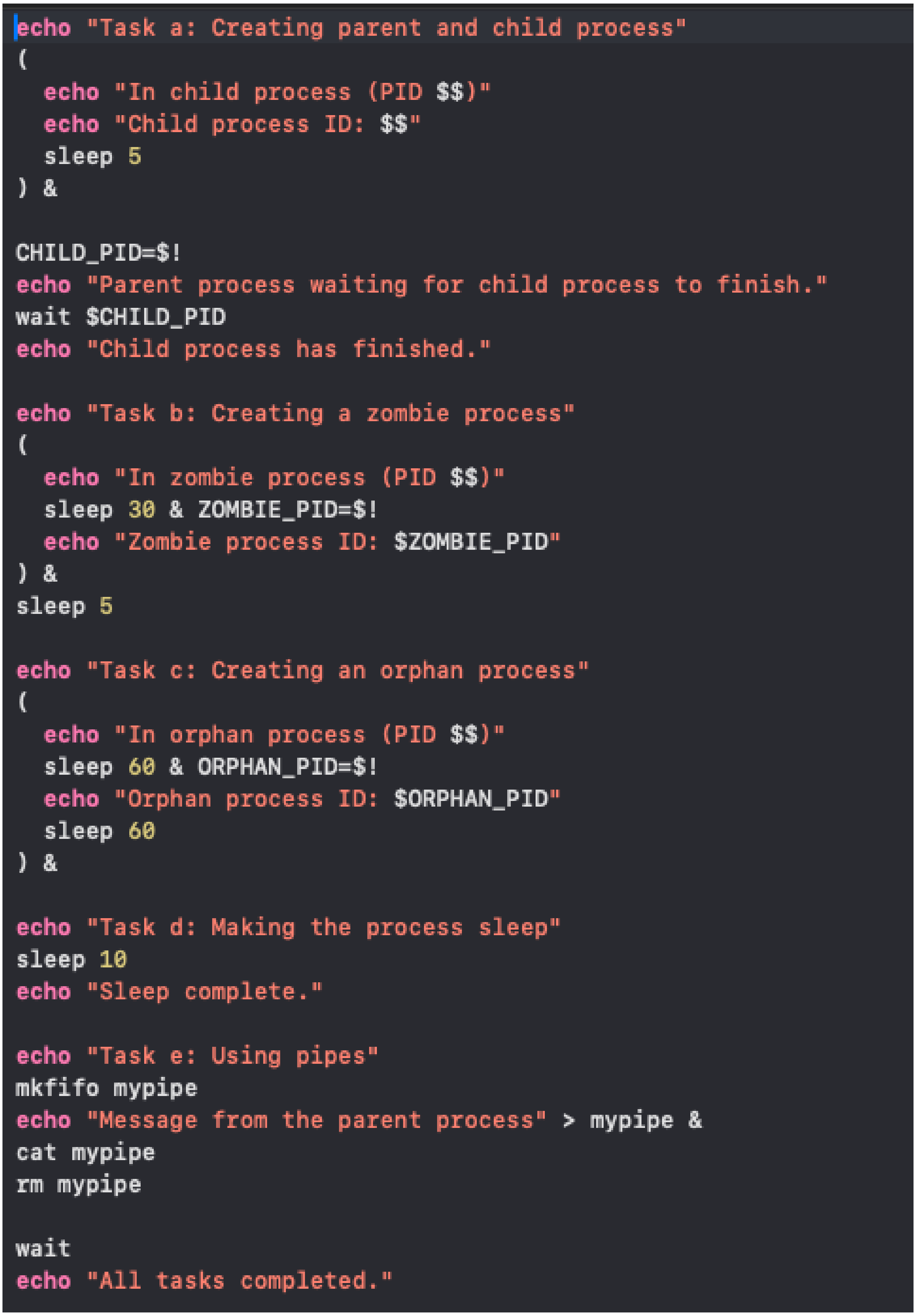
```
new.txt
Hello How are you ? What is your name
```

5. Write a program:
a. To create parent & child process and print their id.
b. To create a zombie process.
c. To create orphan process.

d. To make the process to sleep for few seconds.
e. Implement the program to pass messages using pipes.

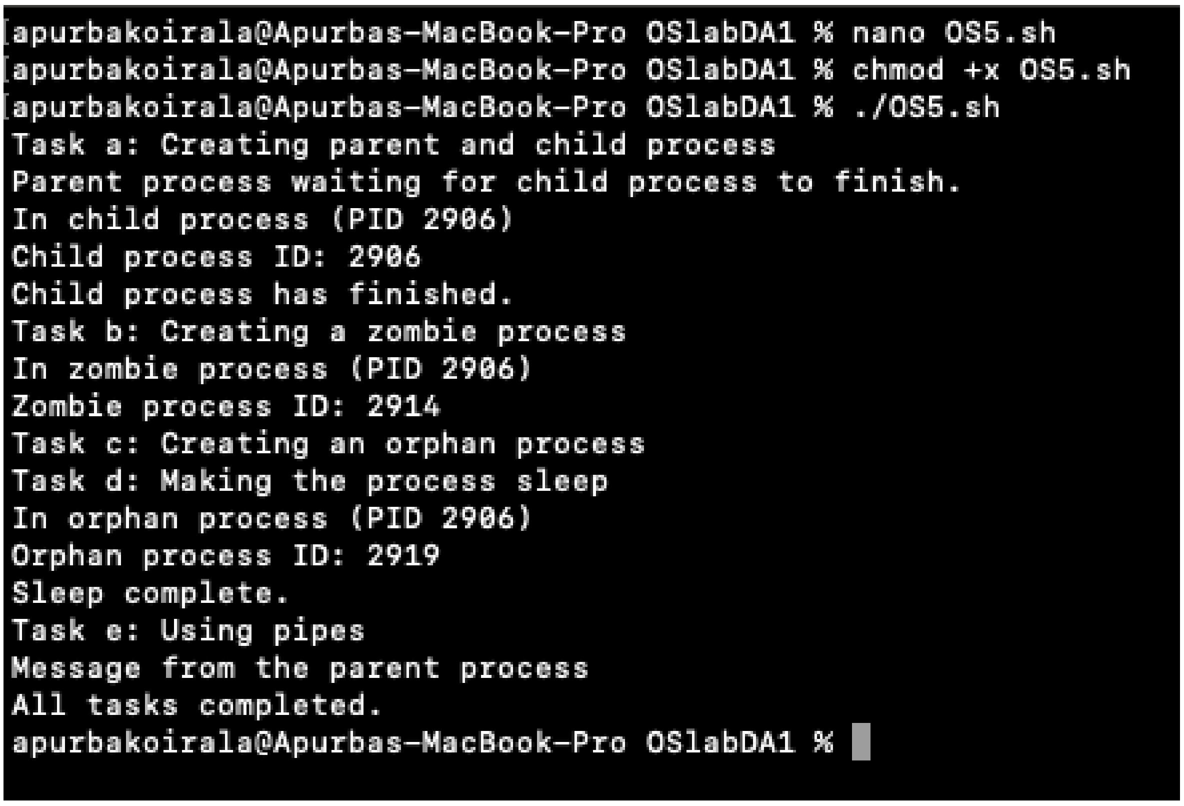
SOURCE CODE:
```
echo "Task a: Creating parent and child process"

  echo "In child process (PID     )"
  echo "Child process ID:     "
          5



echo "Parent process waiting for child process to finish."

echo "Child process has finished."

echo "Task b: Creating a zombie process"

  echo "In zombie process (PID     )"
          30
  echo "Zombie process ID:                        "

        5

echo "Task c: Creating an orphan process"

  echo "In orphan process (PID     )"
          60
  echo "Orphan process ID:                          "
          60


echo "Task d: Making the process sleep"
        10
echo "Sleep complete."

echo "Task e: Using pipes"

echo "Message from the parent process"




echo "All tasks completed."
```

OUTPUT :



```
echo "Task a: Creating parent and child process"
(
  echo "In child process (PID $$)"
  echo "Child process ID: $$"
  sleep 5
) &

CHILD_PID=$!
echo "Parent process waiting for child process to finish."
wait $CHILD_PID
echo "Child process has finished."

echo "Task b: Creating a zombie process"
(
  echo "In zombie process (PID $$)"
  sleep 30 & ZOMBIE_PID=$!
  echo "Zombie process ID: $ZOMBIE_PID"
) &
sleep 5

echo "Task c: Creating an orphan process"
(
  echo "In orphan process (PID $$)"
  sleep 60 & ORPHAN_PID=$!
  echo "Orphan process ID: $ORPHAN_PID"
  sleep 60
) &

echo "Task d: Making the process sleep"
sleep 10
echo "Sleep complete."

echo "Task e: Using pipes"
mkfifo mypipe
echo "Message from the parent process" > mypipe &
cat mypipe
rm mypipe

wait
echo "All tasks completed."
```

RESULTS:



```
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % nano OS5.sh
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % chmod +x OS5.sh
[apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 % ./OS5.sh
Task a: Creating parent and child process
Parent process waiting for child process to finish.
In child process (PID 2906)
Child process ID: 2906
Child process has finished.
Task b: Creating a zombie process
In zombie process (PID 2906)
Zombie process ID: 2914
Task c: Creating an orphan process
Task d: Making the process sleep
In orphan process (PID 2906)
Orphan process ID: 2919
Sleep complete.
Task e: Using pipes
Message from the parent process
All tasks completed.
apurbakoirala@Apurbas-MacBook-Pro OSlabDA1 %
```