TY(IT)

Unix Operating System

Probable Assignment List for External Practical Exam

TY(IT)_UOS_Lab_ESE

(https://docs.google.com/document/d/1_QGqhkS8pat1Rx6HWe8MbHvTdbo7Ki5hEGZLgxic0dE/edit)

1. **Write a program to use fork system call to create 5 child processes and assign 5 operations to childs.**

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>


void performOperation(int operationId) {

    switch (operationId) {

    case 1:

        printf("Child %d: Performing operation 1\n", getpid());

        // Add your code for operation 1 here

        break;

    case 2:

        printf("Child %d: Performing operation 2\n", getpid());

        // Add your code for operation 2 here
```

```
        break;
    case 3:
        printf("Child %d: Performing operation 3\n", getpid());
        // Add your code for operation 3 here
        break;
    case 4:
        printf("Child %d: Performing operation 4\n", getpid());
        // Add your code for operation 4 here
        break;
    case 5:
        printf("Child %d: Performing operation 5\n", getpid());
        // Add your code for operation 5 here
        break;
    default:
        printf("Child %d: Invalid operation ID\n", getpid());
    }
}

int main() {
    int i;
    pid_t childPID;
```

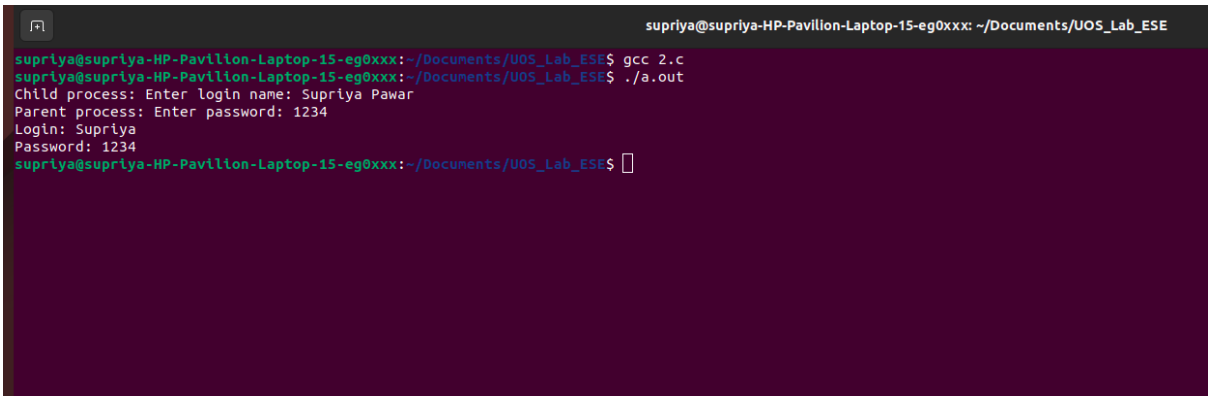
```
    // Create 5 child processes
    for (i = 1; i <= 5; i++) {
    childPID = fork();

    if (childPID == -1) {
        perror("fork() failed");
        exit(EXIT_FAILURE);
    } else if (childPID == 0) {
        // Child process
        performOperation(i);
        exit(EXIT_SUCCESS);
    }
    }

    // Parent process
    for (i = 1; i <= 5; i++) {
    wait(NULL);
    }
    return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~$ cd Documents
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents$ cd UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 1.c
1.c: In function ‘main’:
1.c:52:9: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   52 |         wait(NULL);
      |         ^~~~
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Child 14707: Performing operation 1
Child 14708: Performing operation 2
Child 14710: Performing operation 4
Child 14709: Performing operation 3
Child 14711: Performing operation 5
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

2. **Write a program to use vfork system call(login name by child and password by parent)**

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

int main() {
        pid_t childPID;
        char login[100];
        char password[100];

        childPID = vfork();

        if (childPID == -1) {
        perror("vfork() failed");
        exit(EXIT_FAILURE);
        } else if (childPID == 0) {
        // Child process
        printf("Child process: Enter login name: ");
        scanf("%s", login);
        exit(EXIT_SUCCESS);
        } else {
        // Parent process
        printf("Parent process: Enter password: ");
        scanf("%s", password);
```

```
        printf("Login: %s\n", login);
        printf("Password: %s\n", password);
        }

        return 0;
}
```



3. Write a program to open any application using fork system call.

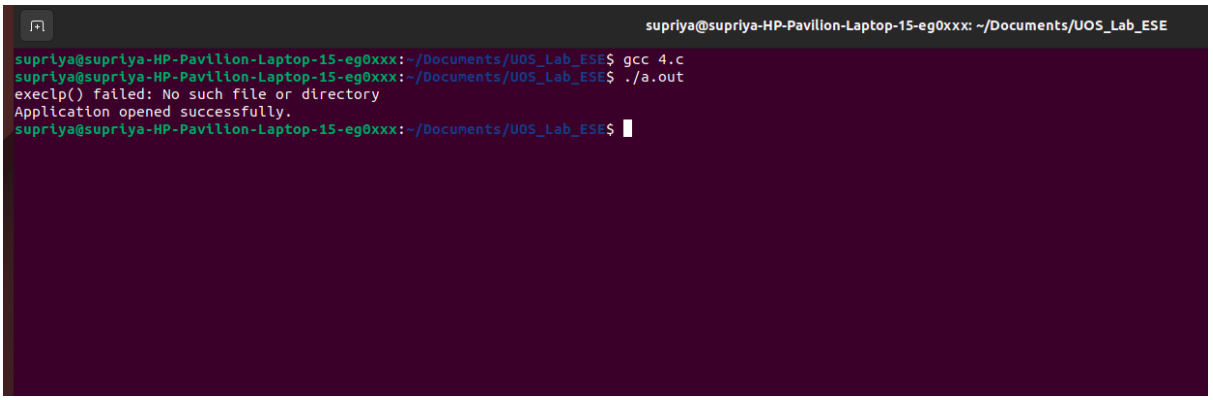
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
        pid_t childPID;

        childPID = fork();

        if (childPID == -1) {
        perror("fork( ) failed");
        exit(EXIT_FAILURE);
        } else if (childPID == 0) {
        // Child process
        char *args[] = {"<path_to_application>", NULL};  // Replace
<path_to_application> with the actual path to the application
        execvp(args[0], args);
```

```
        perror("execvp() failed");  // This line will be executed only if
execvp fails
        exit(EXIT_FAILURE);
        } else {
        // Parent process
        printf("Application opened successfully.\n");
        }

        return 0;
}
```

4. Write a program to open any application using the vfork system call.

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

int main() {
        pid_t childPID;

        childPID = vfork();

        if (childPID == -1) {
        perror("vfork() failed");
        exit(EXIT_FAILURE);
        } else if (childPID == 0) {
        // Child process
```

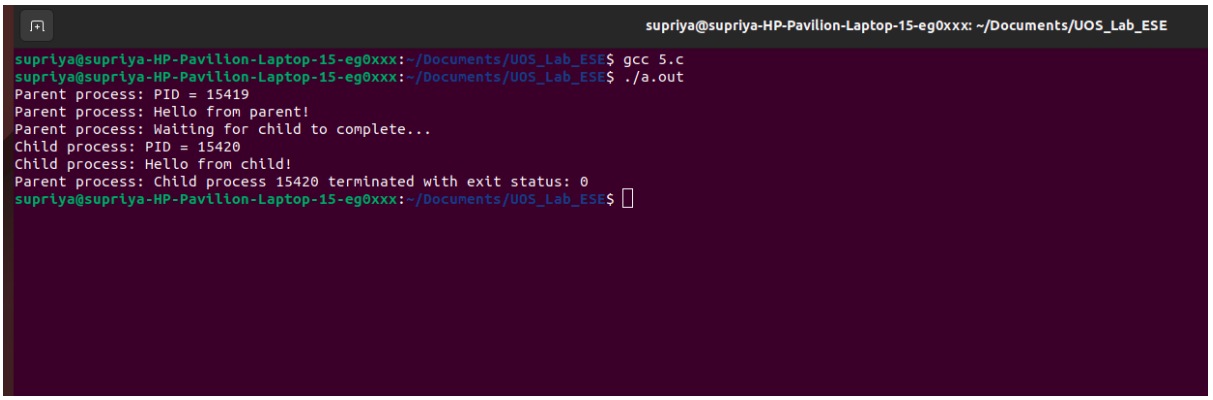
```
        execlp("<application_name>", "<application_name>", NULL);  // Replace <application_name> with the actual application name or path
        perror("execlp() failed");  // This line will be executed only if execlp fails
        exit(EXIT_FAILURE);
        } else {
        // Parent process
        printf("Application opened successfully.\n");
        }

        return 0;
}
```



5. Write a program to demonstrate the wait use with fork sysem call.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
        pid_t childPID;

        childPID = fork();

        if (childPID == -1) {
```

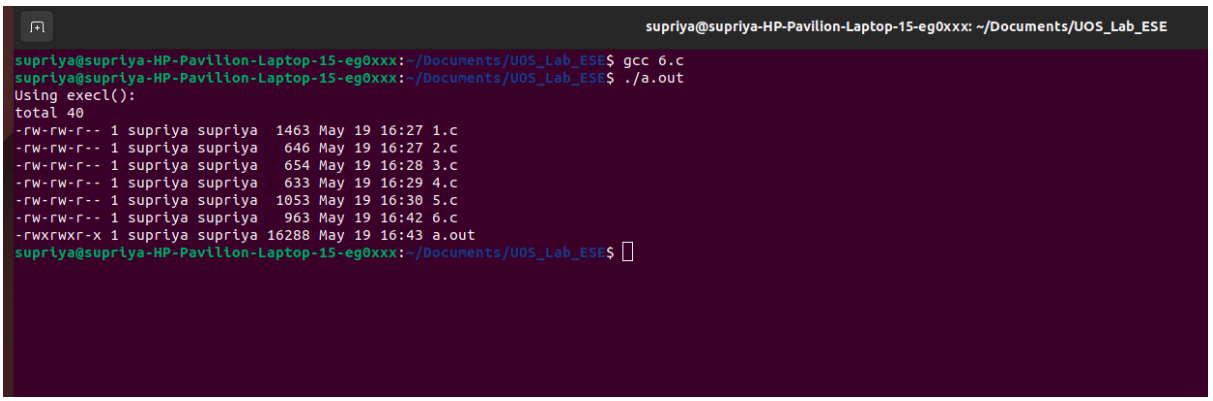
```
        perror("fork() failed");
        exit(EXIT_FAILURE);
        } else if (childPID == 0) {
        // Child process
        printf("Child process: PID = %d\n", getpid());
        printf("Child process: Hello from child!\n");
        exit(EXIT_SUCCESS);
        } else {
        // Parent process
        printf("Parent process: PID = %d\n", getpid());
        printf("Parent process: Hello from parent!\n");
        printf("Parent process: Waiting for child to complete...\n");

        int status;
        pid_t terminatedChild = wait(&status);

        if (terminatedChild == -1) {
        perror("wait() failed");
        exit(EXIT_FAILURE);
        }

        if (WIFEXITED(status)) {
        printf("Parent process: Child process %d terminated with exit status:
%d\n", terminatedChild, WEXITSTATUS(status));
        }
        }

        return 0;
}
```



6. **Write a program to demonstrate the variations exec system call.**

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

int main() {
	// Using execl()
	printf("Using execl():\n");
	execl("/bin/ls", "ls", "-l", NULL);

	// Using execv()
	printf("\nUsing execv():\n");
	char *args[] = {"/bin/ls", "-l", NULL};
	execv(args[0], args);

	// Using execle()
	printf("\nUsing execle():\n");
	char *env[] = {"HOME=/home/user", "PATH=/bin:/usr/bin",
NULL};
	execle("/bin/ls", "ls", "-l", NULL, env);

	// Using execve()
	printf("\nUsing execve():\n");
	char *args2[] = {"/bin/ls", "-l", NULL};
	execve(args2[0], args2, env);

	// Using execlp()
	printf("\nUsing execlp():\n");
	execlp("ls", "ls", "-l", NULL);

	// Using execvp()
	printf("\nUsing execvp():\n");
	char *args3[] = {"ls", "-l", NULL};
	execvp(args3[0], args3);

	// If any of the exec() functions are successful, this line will not be
executed.
```

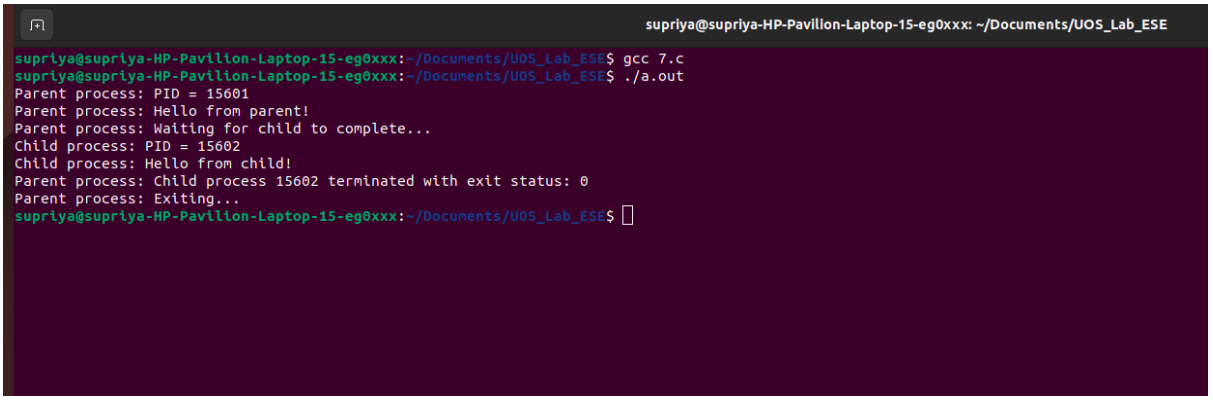
```
        perror("exec() failed");
        exit(EXIT_FAILURE);
}
```



7. Write a program to demonstrate the exit system call use with wait & fork sysem call.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
        pid_t childPID;

        childPID = fork();

        if (childPID == -1) {
        perror("fork() failed");
        exit(EXIT_FAILURE);
        } else if (childPID == 0) {
        // Child process
        printf("Child process: PID = %d\n", getpid());
        printf("Child process: Hello from child!\n");
        exit(EXIT_SUCCESS);
```

```
        } else {
        // Parent process
        printf("Parent process: PID = %d\n", getpid());
        printf("Parent process: Hello from parent!\n");
        printf("Parent process: Waiting for child to complete...\n");

        int status;
        pid_t terminatedChild = wait(&status);

        if (terminatedChild == -1) {
        perror("wait() failed");
        exit(EXIT_FAILURE);
        }

        if (WIFEXITED(status)) {
        printf("Parent process: Child process %d terminated with exit status:
%d\n", terminatedChild, WEXITSTATUS(status));
        }

        printf("Parent process: Exiting...\n");
        exit(EXIT_SUCCESS);
        }

        return 0;

}
```



8. **Write a program to demonstrate the kill system call to send signals between unrelated processes.**

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>

int main() {
   pid_t childPID;

   childPID = fork();

   if (childPID == -1) {
      perror("fork() failed");
      exit(EXIT_FAILURE);
   } else if (childPID == 0) {
      // Child process
      printf("Child process: PID = %d\n", getpid());
      sleep(2);
      printf("Child process: Sending SIGUSR1 signal to parent
process...\n");
      kill(getppid(), SIGUSR1);
      exit(EXIT_SUCCESS);
   } else {
      // Parent process
      printf("Parent process: PID = %d\n", getpid());
      printf("Parent process: Waiting for signal from child process...\n");

      // Signal handler for SIGUSR1
      void sigusr1_handler(int signum) {
         printf("Parent process: Received SIGUSR1 signal from child
process.\n");
      }

      // Register the signal handler
      signal(SIGUSR1, sigusr1_handler);

      while(1) {
```

```
        // Do other work in the parent process
        // ...
    }
}

return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 8.c
/usr/bin/ld: warning: /tmp/ccsbOFQI.o: requires executable stack (because the .note.GNU-stack section is executable)
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Parent process: PID = 15943
Parent process: Waiting for signal from child process...
Child process: PID = 15944
Child process: Sending SIGUSR1 signal to parent process...
Parent process: Received SIGUSR1 signal from child process.
```

9. Write a program to demonstrate the kill system call to send signals between related processes(fork).

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>

int main() {
    pid_t childPID;

    childPID = fork();

    if (childPID == -1) {
        perror("fork() failed");
        exit(EXIT_FAILURE);
    } else if (childPID == 0) {
        // Child process
        printf("Child process: PID = %d\n", getpid());

        // Sleep for 2 seconds
```

```
        sleep(2);

        printf("Child process: Sending SIGUSR1 signal to parent
process...\n");
        kill(getppid(), SIGUSR1);
        exit(EXIT_SUCCESS);
    } else {
        // Parent process
        printf("Parent process: PID = %d\n", getpid());
        printf("Parent process: Waiting for signal from child process...\n");

        // Signal handler for SIGUSR1
        void sigusr1_handler(int signum) {
            printf("Parent process: Received SIGUSR1 signal from child
process.\n");
        }

        // Register the signal handler
        signal(SIGUSR1, sigusr1_handler);

        int status;
        pid_t terminatedChild = wait(&status);

        if (terminatedChild == -1) {
            perror("wait() failed");
            exit(EXIT_FAILURE);
        }

        printf("Parent process: Child process %d terminated.\n",
terminatedChild);
    }

    return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 9.c
/usr/bin/ld: warning: /tmp/ccMFnlr1.o: requires executable stack (because the .note.GNU-stack section is executable)
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Parent process: PID = 16192
Parent process: Waiting for signal from child process...
Child process: PID = 16193
Child process: Sending SIGUSR1 signal to parent process...
Parent process: Received SIGUSR1 signal from child process.
Parent process: Child process 16193 terminated.
```

10. Write a program to use alarm and signal system call(check i/p from user within time)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

int timed_out = 0;

void handleTimeout(int signum) {
    timed_out = 1;
}

int main() {
    int input;

    signal(SIGALRM, handleTimeout);

    printf("Enter a number within 5 seconds: ");

    alarm(5);  // Set an alarm for 5 seconds

    if (scanf("%d", &input) == 1) {
        printf("You entered: %d\n", input);
    } else {
        printf("Invalid input!\n");
    }

    alarm(0);  // Disable the alarm

    if (timed_out) {
        printf("Time is up! You took too long to enter input.\n");
    }

    return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 10.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Enter a number within 5 seconds: 34
You entered: 34
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Enter a number within 5 seconds: 0
You entered: 0
Time is up! You took too long to enter input.
```

11. Write a program for alarm clock using alarm and signal system call.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>

void handleAlarm(int signum) {
    printf("Alarm! Time is up!\n");
    exit(EXIT_SUCCESS);
}

int main() {
    int seconds;

    printf("Enter the number of seconds for the alarm: ");
    scanf("%d", &seconds);

    signal(SIGALRM, handleAlarm);

    printf("Setting the alarm for %d seconds...\n", seconds);
    alarm(seconds);

    printf("Waiting for the alarm...\n");
    pause();  // Wait until a signal is received

    printf("Exiting the program.\n");

    return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 11.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Enter the number of seconds for the alarm: 2
Setting the alarm for 2 seconds...
Waiting for the alarm...
Alarm! Time is up!
```

12. Write a program to give statistics of a given file using stat system call.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include<time.h>

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    char *filename = argv[1];
    struct stat fileStat;

    // Call stat() to get file statistics
    if (stat(filename, &fileStat) == -1) {
        perror("stat() failed");
        exit(EXIT_FAILURE);
    }

    // Print file statistics
    printf("File: %s\n", filename);
    printf("Size: %lld bytes\n", (long long)fileStat.st_size);
    printf("Inode number: %ld\n", (long)fileStat.st_ino);
    printf("Permissions: %o\n", fileStat.st_mode);
    printf("Owner UID: %d\n", fileStat.st_uid);
    printf("Group GID: %d\n", fileStat.st_gid);
    printf("Last accessed: %s", ctime(&fileStat.st_atime));
    printf("Last modified: %s", ctime(&fileStat.st_mtime));
```

```
    printf("Last status change: %s", ctime(&fileStat.st_ctime));

    return 0;
}
```

13. Write a program to give statistics of a given file using fstat system call.

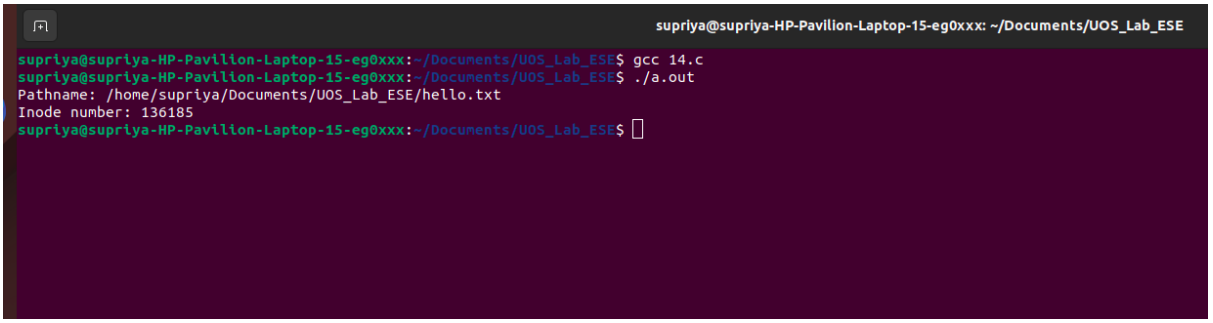
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    char *filename = argv[1];
    int fd;
    struct stat fileStat;

    // Open the file
    if ((fd = open(filename, O_RDONLY)) == -1) {
        perror("open() failed");
        exit(EXIT_FAILURE);
    }

    // Call fstat() to get file statistics
    if (fstat(fd, &fileStat) == -1) {
        perror("fstat() failed");
        close(fd);
        exit(EXIT_FAILURE);
    }
```

```
    // Print file statistics
    printf("File: %s\n", filename);
    printf("Size: %lld bytes\n", (long long)fileStat.st_size);
    printf("Inode number: %ld\n", (long)fileStat.st_ino);
    printf("Permissions: %o\n", fileStat.st_mode);
    printf("Owner UID: %d\n", fileStat.st_uid);
    printf("Group GID: %d\n", fileStat.st_gid);
    printf("Last accessed: %s", ctime(&fileStat.st_atime));
    printf("Last modified: %s", ctime(&fileStat.st_mtime));
    printf("Last status change: %s", ctime(&fileStat.st_ctime));

    // Close the file
    close(fd);

    return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 13.c
13.c: In function ‘main’:
13.c:33:33: warning: implicit declaration of function ‘ctime’ [-Wimplicit-function-declaration]
   33 |     printf("Last accessed: %s", ctime(&fileStat.st_atime));
      |                                 ^~~~~
13.c:7:1: note: ‘ctime’ is defined in header ‘<time.h>’; did you forget to ‘#include <time.h>’?
    6 | #include <unistd.h>
  +++ |+#include <time.h>
    7 |
13.c:33:29: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘int’ [-Wformat=]
   33 |     printf("Last accessed: %s", ctime(&fileStat.st_atime));
      |                            ~^   ~~~~~~~~~~~~~~~~~~~~~~~~
      |                             |   |
      |                             |   int
      |                             char *
      |                            %d
13.c:34:29: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘int’ [-Wformat=]
   34 |     printf("Last modified: %s", ctime(&fileStat.st_mtime));
      |                            ~^   ~~~~~~~~~~~~~~~~~~~~~~~~
      |                             |   |
      |                             |   int
      |                             char *
      |                            %d
13.c:35:34: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘int’ [-Wformat=]
   35 |     printf("Last status change: %s", ctime(&fileStat.st_ctime));
      |                                 ~^   ~~~~~~~~~~~~~~~~~~~~~~~~
      |                                  |   |
      |                                  |   int
      |                                  char *
      |                                 %d
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
File: /home/supriya/Documents/UOS_Lab_ESE/hello.txt
Size: 26 bytes
Inode number: 136185
Permissions: 100664
Owner UID: 1000
Group GID: 1000
Segmentation fault (core dumped)
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

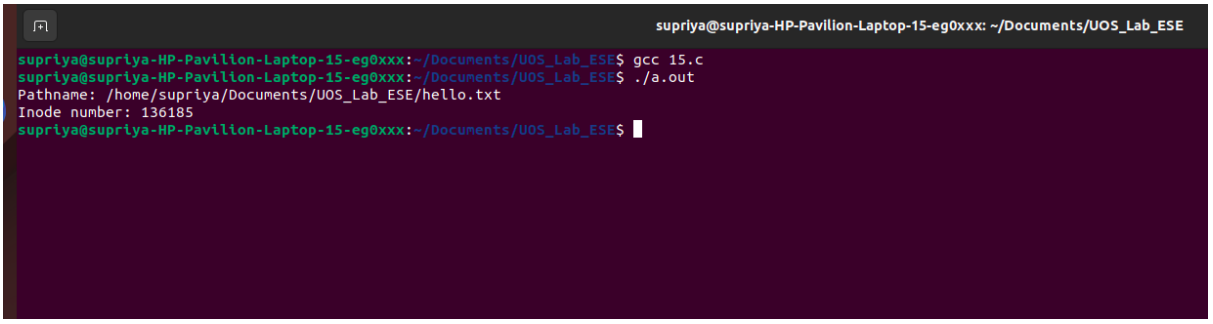
14. Write a program to convert pathname to Inode using stat system call

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s <pathname>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    char *pathname = argv[1];
    struct stat fileStat;

    // Call stat() to get file statistics
    if (stat(pathname, &fileStat) == -1) {
        perror("stat() failed");
        exit(EXIT_FAILURE);
    }

    // Print inode number
    printf("Pathname: %s\n", pathname);
    printf("Inode number: %ld\n", (long)fileStat.st_ino);

    return 0;
}
```



15. Write a program to convert pathname to Inode using 'ls' command.

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

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s <pathname>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    char command[256];
    snprintf(command, sizeof(command), "ls -i %s", argv[1]);

    FILE *fp = popen(command, "r");
    if (fp == NULL) {
        perror("popen() failed");
        exit(EXIT_FAILURE);
    }

    char output[256];
    if (fgets(output, sizeof(output), fp) != NULL) {
        char *token = strtok(output, " ");
        if (token != NULL) {
            printf("Pathname: %s\n", argv[1]);
            printf("Inode number: %s\n", token);
        }
    }

    pclose(fp);

    return 0;
}
```



16. Write a multithreaded program in JAVA for chatting.

Server

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class ChatServer {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Server started. Waiting for clients...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
clientSocket.getInetAddress().getHostAddress());

                ClientHandler clientHandler = new ClientHandler(clientSocket);
                clientHandler.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

class ClientHandler extends Thread {
    private Socket clientSocket;
    private BufferedReader inputReader;
    private PrintWriter outputWriter;

    public ClientHandler(Socket clientSocket) throws IOException {
        this.clientSocket = clientSocket;
        inputReader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
```

```
        outputWriter = new PrintWriter(clientSocket.getOutputStream(),
true);
    }

    @Override
    public void run() {
        try {
            String clientMessage;
            while ((clientMessage = inputReader.readLine()) != null) {
                System.out.println("Client says: " + clientMessage);

                // Echo back the message to the client
                outputWriter.println("Server says: " + clientMessage);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                clientSocket.close();
                inputReader.close();
                outputWriter.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Client

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ChatClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 12345);
            System.out.println("Connected to server.");
```

```
        BufferedReader inputReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter outputWriter = new
PrintWriter(socket.getOutputStream(), true);

        BufferedReader consoleReader = new BufferedReader(new
InputStreamReader(System.in));

        String userInput;
        while ((userInput = consoleReader.readLine()) != null) {
            outputWriter.println(userInput);

            String serverResponse = inputReader.readLine();
            System.out.println("Server says: " + serverResponse);
        }

        socket.close();
        inputReader.close();
        outputWriter.close();
        consoleReader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

17. Write a program to create 3 threads, first thread printing even no, second thread printing odd no. and third thread printing prime no.

```
public class NumberPrinter {
  public static void main(String[] args) {
    Thread evenThread = new Thread(new EvenNumberPrinter());
    Thread oddThread = new Thread(new OddNumberPrinter());
    Thread primeThread = new Thread(new PrimeNumberPrinter());
```

```
        evenThread.start();
        oddThread.start();
        primeThread.start();
    }
}

class EvenNumberPrinter implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i <= 20; i += 2) {
            System.out.println("Even Number: " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class OddNumberPrinter implements Runnable {
    @Override
    public void run() {
        for (int i = 1; i <= 20; i += 2) {
            System.out.println("Odd Number: " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class PrimeNumberPrinter implements Runnable {
    @Override
    public void run() {
        for (int i = 2; i <= 20; i++) {
            if (isPrime(i)) {
                System.out.println("Prime Number: " + i);
                try {
                    Thread.sleep(500);
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

    private boolean isPrime(int num) {
        if (num <= 1) {
            return false;
        }

        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) {
                return false;
            }
        }

        return true;
    }
}
```

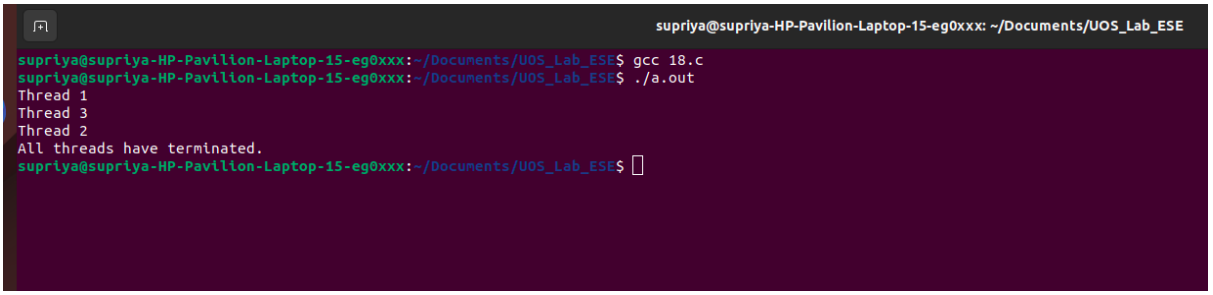
18. Write a multithreaded program in linux to use the pthread library.

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

#define NUM_THREADS 3

void *printMessage(void *arg) {
    char *message = (char *)arg;
    printf("%s\n", message);
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_THREADS];
```

```
    char *messages[NUM_THREADS] = {
        "Thread 1",
        "Thread 2",
        "Thread 3"
    };

    int i;
    for (i = 0; i < NUM_THREADS; i++) {
        int status = pthread_create(&threads[i], NULL, printMessage, (void
*)messages[i]);
        if (status != 0) {
            fprintf(stderr, "Error creating thread %d: %d\n", i, status);
            exit(EXIT_FAILURE);
        }
    }

    for (i = 0; i < NUM_THREADS; i++) {
        int status = pthread_join(threads[i], NULL);
        if (status != 0) {
            fprintf(stderr, "Error joining thread %d: %d\n", i, status);
            exit(EXIT_FAILURE);
        }
    }

    printf("All threads have terminated.\n");

    return 0;
}
```

Compile the program:

gcc -o multithreaded multithreaded.c -pthread



19. Write a multithreaded program for producer-consumer problem in JAVA.

```
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

public class ProducerConsumer {
    public static void main(String[] args) {
        BlockingQueue<Integer> buffer = new ArrayBlockingQueue<>(5);

        Thread producerThread = new Thread(new Producer(buffer));
        Thread consumerThread = new Thread(new Consumer(buffer));

        producerThread.start();
        consumerThread.start();
    }
}

class Producer implements Runnable {
    private BlockingQueue<Integer> buffer;

    public Producer(BlockingQueue<Integer> buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        try {
            for (int i = 1; i <= 10; i++) {
                buffer.put(i);
                System.out.println("Produced: " + i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

class Consumer implements Runnable {
```

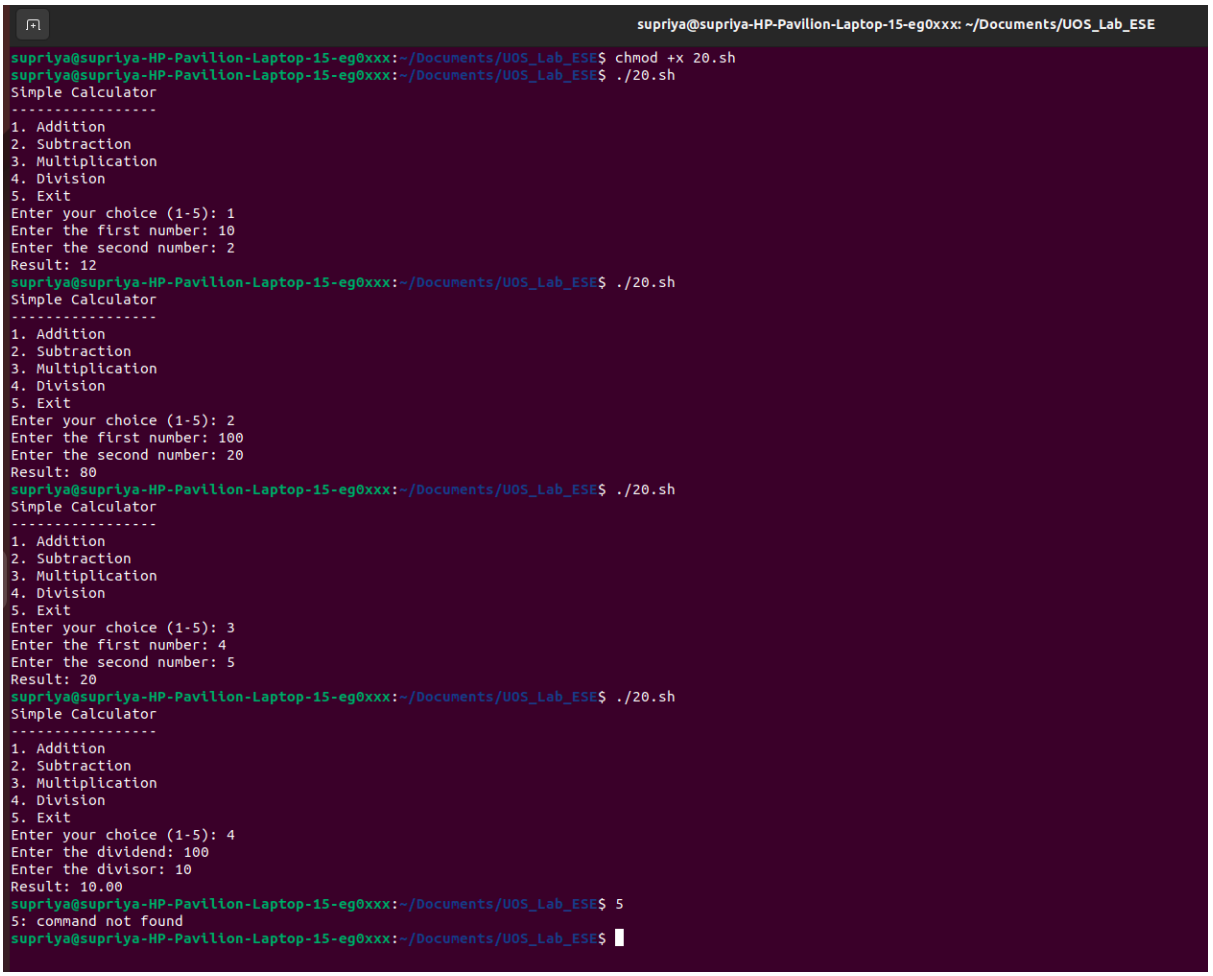
```java
    private BlockingQueue<Integer> buffer;

    public Consumer(BlockingQueue<Integer> buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        try {
            for (int i = 1; i <= 10; i++) {
                int value = buffer.take();
                System.out.println("Consumed: " + value);
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ javac ProducerConsumer.java
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ java ProducerConsumer
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Produced: 4
Consumed: 3
Produced: 5
Produced: 6
Consumed: 4
Produced: 7
Produced: 8
Consumed: 5
Produced: 9
Produced: 10
Consumed: 6
Consumed: 7
Consumed: 8
Consumed: 9
Consumed: 10
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

20. Write a program to implement shell script for calculator.

```
#!/bin/bash

echo "Simple Calculator"
echo "-----------------"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
echo "5. Exit"

read -p "Enter your choice (1-5): " choice

case $choice in
   1)
      read -p "Enter the first number: " num1
      read -p "Enter the second number: " num2
      result=$(echo "$num1 + $num2" | bc)
      echo "Result: $result"
      ;;
   2)
      read -p "Enter the first number: " num1
      read -p "Enter the second number: " num2
      result=$(echo "$num1 - $num2" | bc)
      echo "Result: $result"
      ;;
   3)
      read -p "Enter the first number: " num1
      read -p "Enter the second number: " num2
      result=$(echo "$num1 * $num2" | bc)
      echo "Result: $result"
      ;;
   4)
      read -p "Enter the dividend: " dividend
      read -p "Enter the divisor: " divisor
      result=$(echo "scale=2; $dividend / $divisor" | bc)
      echo "Result: $result"
      ;;
   5)
```

```
        echo "Exiting..."
        exit 0
        ;;
    *)
        echo "Invalid choice. Exiting..."
        exit 1
        ;;
esac
```



21. Write a program to implement a digital clock using shell script.

```
#!/bin/bash

while true; do
    clear
    echo "Digital Clock"
    echo "-------------"
    echo $(date +"%T")
    sleep 1
done
```

22. Write a program to check whether system is in network or not using 'ping' command using shell script.

```
#!/bin/bash

ping -c 1 google.com > /dev/null

if [ $? -eq 0 ]; then
        echo "System is connected to the network."
else
        echo "System is not connected to the network."
fi
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 22.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 22.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 22_digitalClock.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ chmod +x 22_digitalClock.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./22_digitalClock.sh
System is connected to the network.
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

23. Write a program to sort the given 10 numbers in ascending order using shell.

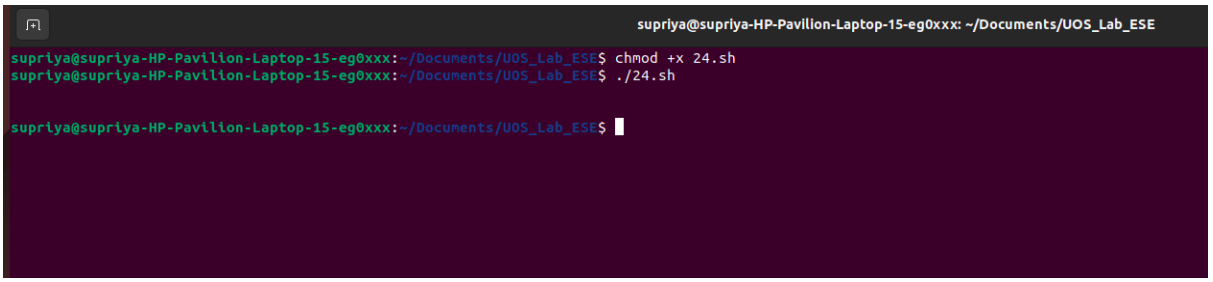
```
#!/bin/bash

echo "Enter 10 numbers:"

# Read the numbers into an array
read -a numbers

# Sort the numbers in ascending order
sorted_numbers=$(printf "%s\n" "${numbers[@]}" | sort -n)

echo "Sorted numbers (ascending order):"
echo "$sorted_numbers"
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 22.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 22.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 22_digitalClock.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ chmod +x 22_digitalClock.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./22_digitalClock.sh
System is connected to the network.
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 23.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 23.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ chmod +x 23.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./23.sh
Enter 10 numbers:
11
Sorted numbers (ascending order):
11
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./23.sh
Enter 10 numbers:
1 2 45 6 11 34 7 52 44 10
Sorted numbers (ascending order):
1
2
6
7
10
11
34
44
45
52
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

24. Write a program to print “Hello World” message in bold, blink effect, and in different colors like red, blue etc.

Chatgpt

```
#!/bin/bash

# Bold effect
bold=$(tput bold)

# Blink effect
blink=$(tput blink)

# Colors
red=$(tput setaf 1)
blue=$(tput setaf 4)
reset=$(tput sgr0)

# Print "Hello World" with effects and colors
echo "${bold}${blink}${red}Hello World${reset}"
echo "${bold}${blink}${blue}Hello World${reset}"
```



Journal

echo -e "\033[1m Hello World"

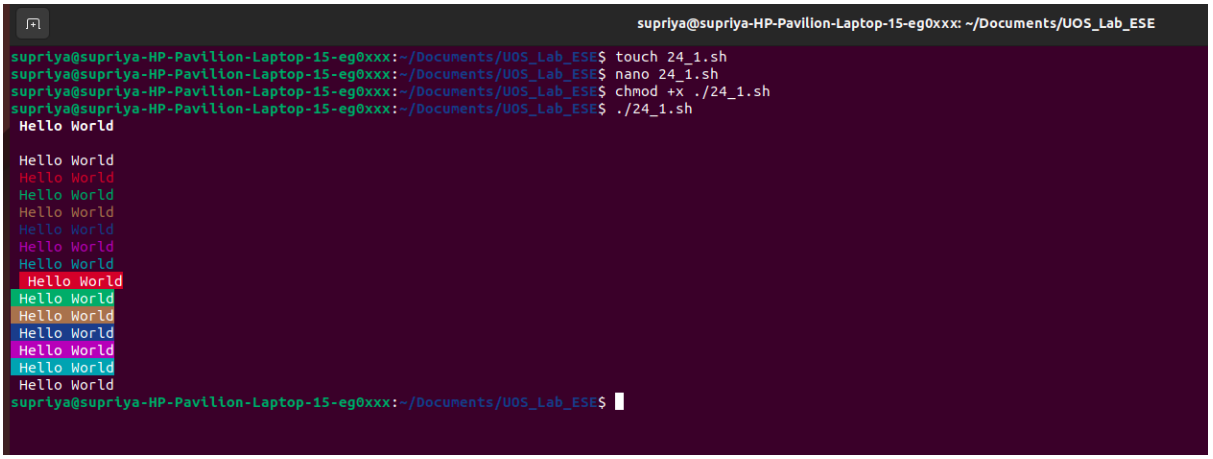
bold effect

echo -e "\033[5m Hello World"

blink effect

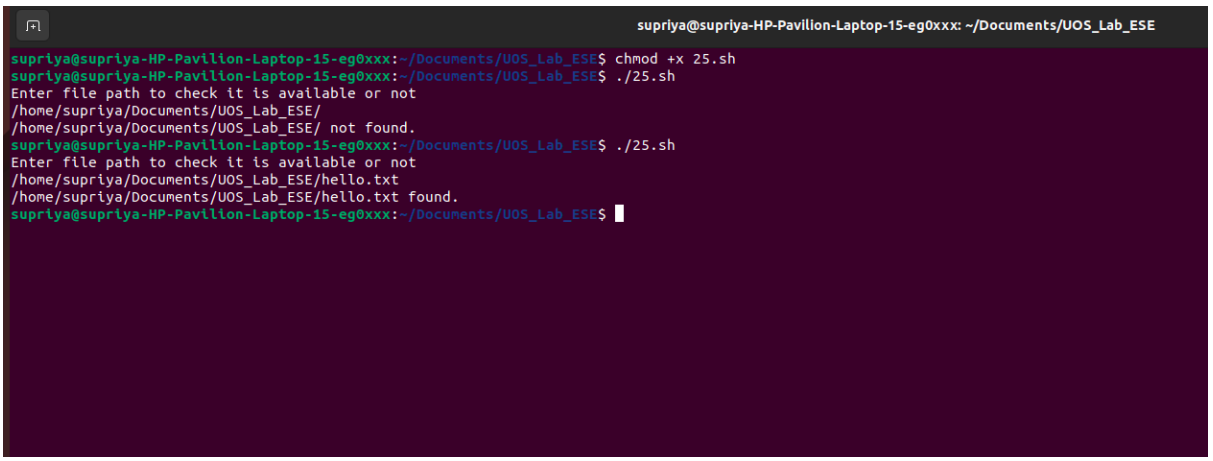
echo -e "\033[0m Hello World"

```
# back to noraml
echo -e "\033[31m Hello World"
# Red color
echo -e "\033[32m Hello World"
# Green color
echo -e "\033[33m Hello World"
# See remaing on screen
echo -e "\033[34m Hello World"
echo -e "\033[35m Hello World"
echo -e "\033[36m Hello World"
echo -e -n "\033[0m "
# back to normal
echo -e "\033[41m Hello World"
echo -e "\033[42m Hello World"
echo -e "\033[43m Hello World"
echo -e "\033[44m Hello World"
echo -e "\033[45m Hello World"
echo -e "\033[46m Hello World"
echo -e "\033[0m Hello World"
# back to normal
```



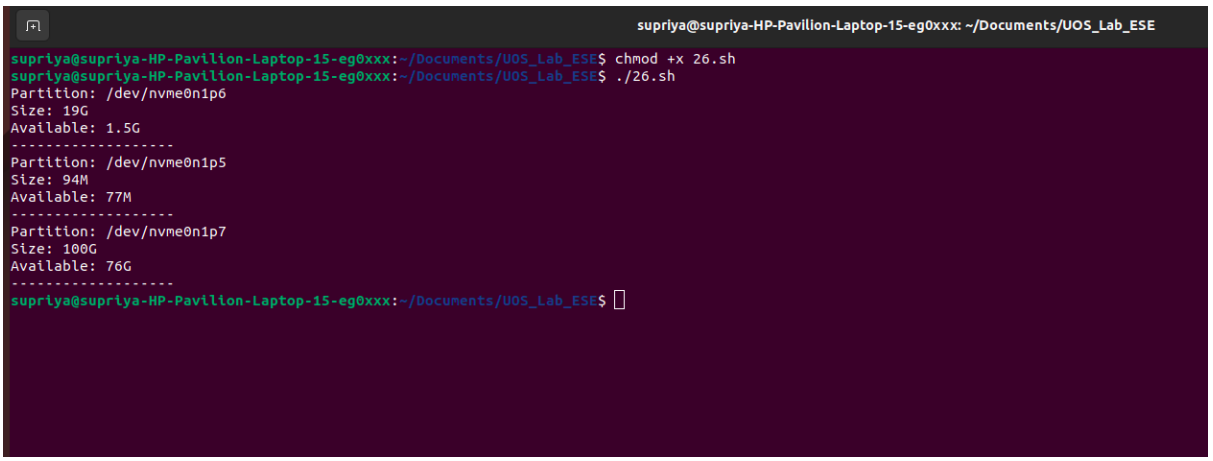
25. Write a shell script to find whether a given file exists or not.

```
echo "Enter file path to check it is available or not";

read file

if [ -f "$file" ]

then

echo "$file found."

else

echo "$file not found."

fi
```



26. Write a shell script to show the disk partitions and their size and disk usage i.e free space.

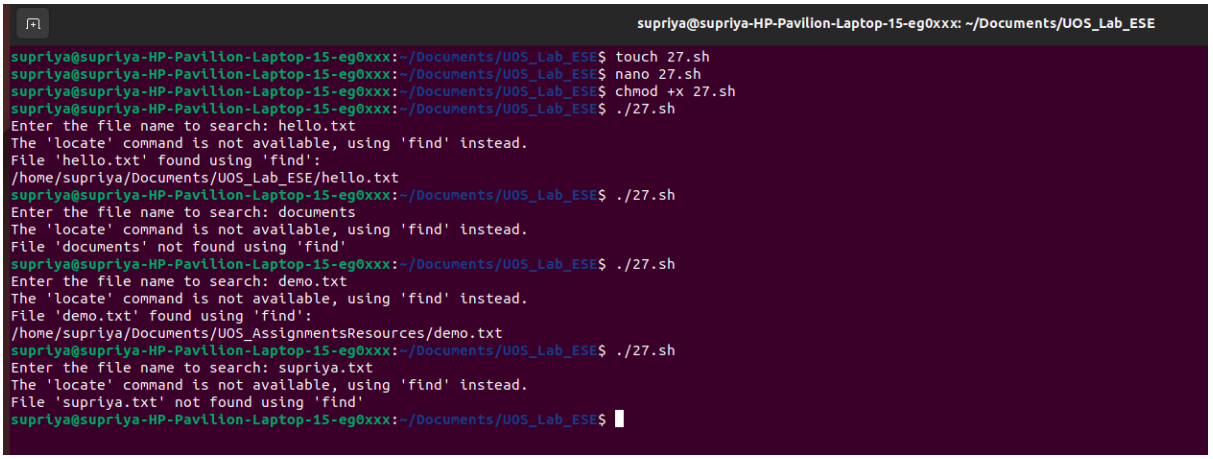
```
#!/bin/bash

# Get the list of disk partitions excluding tmpfs
partitions=$(df -h --exclude-type=tmpfs | awk 'NR>1 {print $1}')

# Iterate over the partitions
for partition in $partitions; do
        # Get the size and available space of the partition
        size=$(df -h $partition | awk 'NR>1 {print $2}')
        available=$(df -h $partition | awk 'NR>1 {print $4}')

        # Print the partition, size, and available space
        echo "Partition: $partition"
        echo "Size: $size"
        echo "Available: $available"
        echo "-------------------"
done
```



27. Write a shell script to find the given file in the system using find or locate command.

```
#!/bin/bash

# Get the file name from the user
read -p "Enter the file name to search: " filename

# Check if the 'locate' command is available
if command -v locate &>/dev/null; then
        # Use 'locate' to find the file
        locate_result=$(locate -b -n 1 "$filename")

        # Check if 'locate' found the file
        if [[ -n "$locate_result" ]]; then
        echo "File '$filename' found using 'locate':"
        echo "$locate_result"
        else
        echo "File '$filename' not found using 'locate'"
        fi
else
        echo "The 'locate' command is not available, using 'find' instead."

        # Use 'find' to search for the file
        find_result=$(find / -name "$filename" 2>/dev/null)

        # Check if 'find' found the file
```

```
        if [[ -n "$find_result" ]]; then
        echo "File '$filename' found using 'find':"
        echo "$find_result"
        else
        echo "File '$filename' not found using 'find'"
        fi
fi
```



28. Write a shell script to download webpage at given url using command(wget)

```
#!/bin/bash

# Get the URL from the user
read -p "Enter the URL of the webpage to download: " url

# Check if 'wget' command is available
if command -v wget &>/dev/null; then
        # Download the webpage using 'wget'
        wget "$url"
        echo "Webpage downloaded successfully."
else
        echo "The 'wget' command is not available. Please install 'wget' to
download webpages."
fi
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ chmod +x 28.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./28.sh
Enter the URL of the webpage to download: https://example.com
--2023-05-24 19:48:43--  https://example.com/
Resolving example.com (example.com)... 2606:2800:220:1:248:1893:25c8:1946, 93.184.216.34
Connecting to example.com (example.com)|2606:2800:220:1:248:1893:25c8:1946|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1256 (1.2K) [text/html]
Saving to: ‘index.html.1’

index.html.1                                        100%[===================================================================================================>]   1.23K  --.-KB/s    in 0s

2023-05-24 19:48:45 (74.7 MB/s) - ‘index.html.1’ saved [1256/1256]

Webpage downloaded successfully.
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

29. Write a shell script to download a webpage from a given URL . (Using wget command).

30. Write a shell script to display the users on the system . (Using finger or who command).

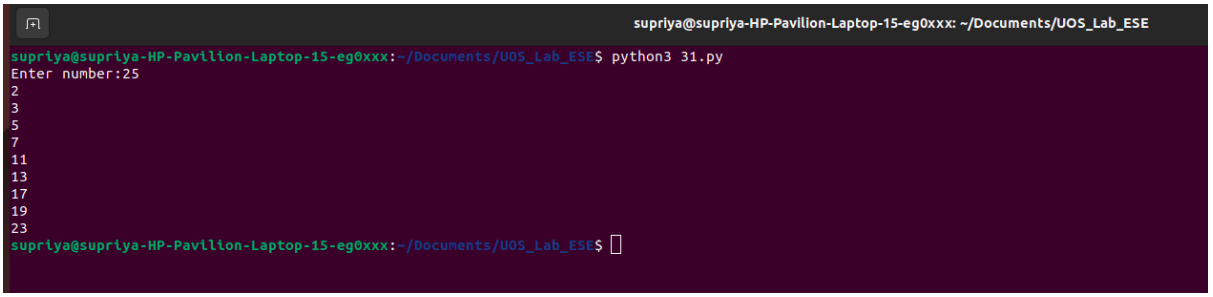
```
#!/bin/bash

#Use who command to retrieve user information
who
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ chmod +x 30.sh
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./30.sh
supriya  :1           2023-05-24 15:36 (:1)
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

31. Write a python recursive function for prime number input limit as a parameter to it.

```
n = int(input("Enter number:"))
```

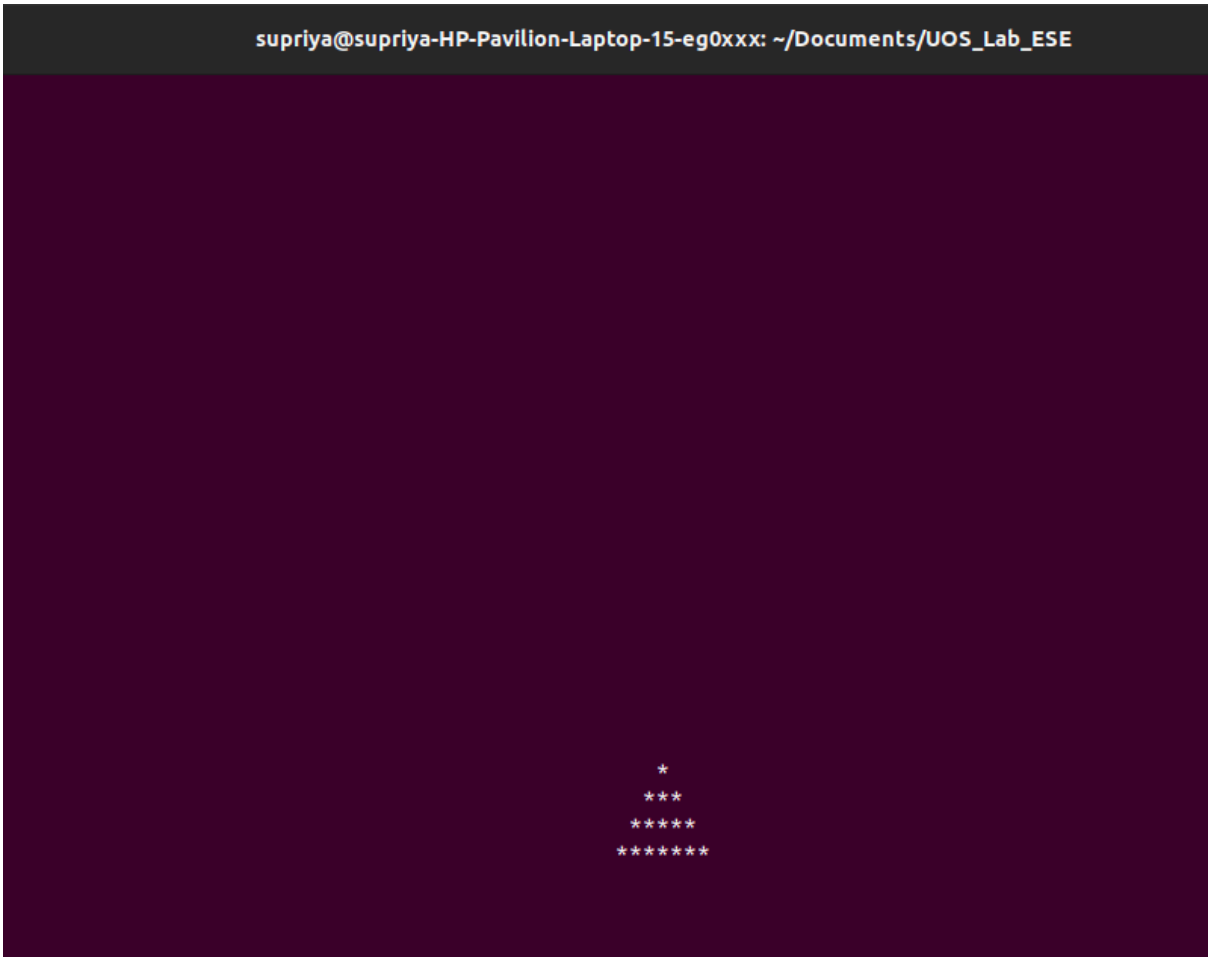
```
for i in range(1, n + 1):
        c = 0

        if i == 1:
        continue
        if i == 2:
        print(2)
        continue

        for j in range(2, i - 1):
        if i % j == 0:
        c = c + 1

        if c == 0:
        print(i)
```



***30. Write a program to display the following pyramid. The number of lines in the pyramid should not be hard-coded. It should be obtained from the user. The pyramid should appear as close to the center of the screen as possible.(Hint: Basics n loops)**

```
import os
 rows, columns = os.popen('stty size', 'r').read().split()
r=int(rows)
c=int(columns)
n = int(input("Enter number of rows:"))
for i in range(int(r/2-n/2)):
print()
for i in range(n):
for k in range(int(c/2)-int(n/2)):
```

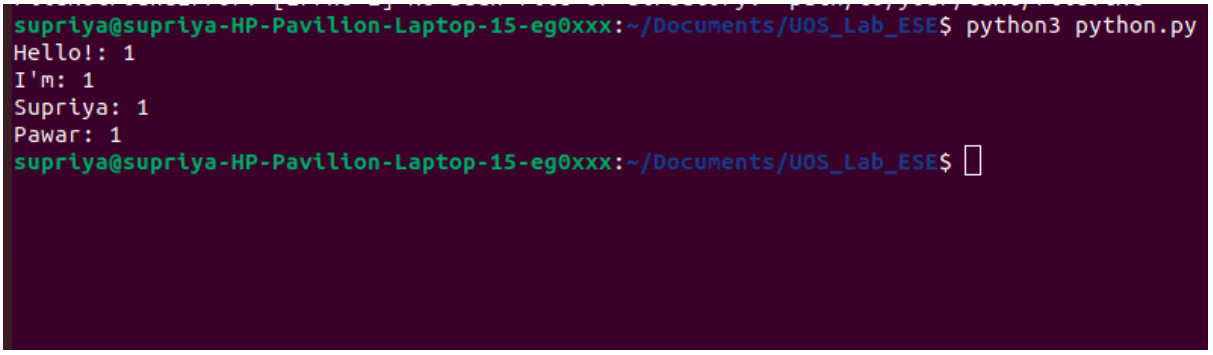
```
print(" ",end="")
for k in range(n-i-1):
print(" ",end="")
for k in range(2*i+1):
print("*",end="")
print("\n",end="")
for i in range(int(r/2-n/2));
print()
```



***31. Take any txt file and count word frequencies in a file.(hint : file handling + basics).**

```
from collections import Counter

def count_word_frequencies(file_path):
    # Read the text file
    with open(file_path, 'r') as file:
```

```
        text = file.read()

    # Split the text into words
    words = text.split()

    # Count the frequencies of each word
    word_counts = Counter(words)

    return word_counts

# Provide the path to your text file
file_path = 'path/to/your/text/file.txt'

# Call the function to count word frequencies
word_frequencies = count_word_frequencies(file_path)

# Display the word frequencies
for word, count in word_frequencies.items():
    print(f'{word}: {count}')
```



***32. Generate frequency list of all the commands you have used, and show the top 5 commands along with their count. (Hint: history command hist will give you a list of all commands used.)**

⇒ This code runs the shell command to retrieve the command history and captures the output. It then splits the output into lines, extracts the last word of each line (which represents the command), and counts the frequencies of each command using a dictionary. Finally, it sorts the command frequencies in descending order and prints the top 5 most frequent commands.

```
#!/bin/bash

# Run the 'history' command and save the output to a temporary file
history > temp_history.txt

# Extract the commands from the history file and count their occurrences
command_counts=$(awk '{print $2}' temp_history.txt | sort | uniq -c | sort -nr)

# Display the top 5 commands along with their count
echo "Top 5 Commands:"
echo "$command_counts" | head -n 5
# Remove the temporary file
rm temp_history.txt
```

32. Write a shell script to download a given file from ftp://10.10.13.16 if it exists on ftp. (use lftp, get and mget commands).

33. Write program to implement producer consumer problem using semaphore.h in C/JAVA

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

#define BUFFER_SIZE 5
#define NUM_ITEMS 20

int buffer[BUFFER_SIZE];
int count = 0;
sem_t full, empty, mutex;
```

```
int itemsProduced = 0;
int itemsConsumed = 0;

void *producer(void *arg) {
        int item = 1;
        while (itemsProduced < NUM_ITEMS) {
        sem_wait(&empty);
        sem_wait(&mutex);

        buffer[count] = item;
        printf("Produced item: %d\n", item);
        count++;

        sem_post(&mutex);
        sem_post(&full);

        item++;
        itemsProduced++;

        // Sleep for some time to simulate production time
        usleep(100000);
        }
}

void *consumer(void *arg) {
        while (itemsConsumed < NUM_ITEMS) {
        sem_wait(&full);
        sem_wait(&mutex);

        int item = buffer[count - 1];
        printf("Consumed item: %d\n", item);
        count--;

        sem_post(&mutex);
        sem_post(&empty);

        itemsConsumed++;
        // Sleep for some time to simulate consumption time
        usleep(150000);
        }
}
```

```
int main() {
	pthread_t producer_thread, consumer_thread;

	// Initialize semaphores
	sem_init(&full, 0, 0);
	sem_init(&empty, 0, BUFFER_SIZE);
	sem_init(&mutex, 0, 1);

	// Create producer and consumer threads
	pthread_create(&producer_thread, NULL, producer, NULL);
	pthread_create(&consumer_thread, NULL, consumer, NULL);

	// Wait for threads to finish
	pthread_join(producer_thread, NULL);
	pthread_join(consumer_thread, NULL);

	// Destroy semaphores
	sem_destroy(&full);
	sem_destroy(&empty);
	sem_destroy(&mutex);

	return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 33.c
33.c: In function ‘producer’:
33.c:32:9: warning: implicit declaration of function ‘usleep’ [-Wimplicit-function-declaration]
   32 |         usleep(100000);
      |         ^~~~~~
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Produced item: 1
Consumed item: 1
Produced item: 2
Consumed item: 2
Produced item: 3
Consumed item: 3
Produced item: 4
Produced item: 5
Consumed item: 5
Produced item: 6
Consumed item: 6
Produced item: 7
Produced item: 8
Consumed item: 8
Produced item: 9
Consumed item: 9
Produced item: 10
Produced item: 11
Consumed item: 11
Produced item: 12
Consumed item: 12
Produced item: 13
Produced item: 14
Consumed item: 14
Produced item: 15
Consumed item: 15
Produced item: 16
Consumed item: 16
Produced item: 17
Consumed item: 17
Produced item: 18
Consumed item: 18
Produced item: 19
Consumed item: 19
Produced item: 20
Consumed item: 20
Consumed item: 13
Consumed item: 10
Consumed item: 7
Consumed item: 4
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

34. Write a program to implement reader-writers problem using semaphore.

```
import java.util.concurrent.Semaphore;
class Reader implements Runnable
{
  private Semaphore mutex;
  private Semaphore wrt;
  private int readerId;
  public Reader(Semaphore mutex, Semaphore wrt, int readerId)
  {
        this.mutex = mutex;
        this.wrt = wrt;
        this.readerId = readerId;
  }
  @Override
  public void run()
  {
```

```
        try
        {
        while (true)
        {
        Thread.sleep((long) (Math.random() * 5000)); // Simulate reading
time
        mutex.acquire(); // Acquire mutex to ensure mutual exclusion
between readers
        System.out.println("Reader " + readerId + " is reading");
        mutex.release(); // Release mutex
// Reading is happening concurrently, so multiple readers can read at the
same time
        Thread.sleep((long) (Math.random() * 5000)); // Simulate
processing time
        }
        }
        catch (InterruptedException e)
        {
        e.printStackTrace();
        }
  }
}

class Writer implements Runnable
{
        private Semaphore mutex;
        private Semaphore wrt;
        private int writerId;
        public Writer(Semaphore mutex, Semaphore wrt, int writerId)
        {
        this.mutex = mutex;
        this.wrt = wrt;
        this.writerId = writerId;
        }
        @Override
        public void run()
        {
        try
        {
        while (true)
        {
                Thread.sleep((long) (Math.random() * 5000)); // Simulate
```

```
writing time
                wrt.acquire(); // Acquire write lock
                System.out.println("Writer " + writerId + " is writing");
                Thread.sleep((long) (Math.random() * 5000)); // Simulate
processing time
                wrt.release(); // Release write lock
        }
        }
        catch (InterruptedException e)
        {
        e.printStackTrace();
        }
        }
}

public class ReaderWriterSemaphore
{
        public static void main(String[] args)
        {
        int numReaders = 3;
        int numWriters = 2;
        Semaphore mutex = new Semaphore(1); // Mutex for reader access
        Semaphore wrt = new Semaphore(1); // Semaphore for write lock
        // Create reader threads
        for (int i = 1; i <= numReaders; i++)
        {
        Thread readerThread = new Thread(new Reader(mutex, wrt, i));
        readerThread.start();
        }
        // Create writer threads
        for (int i = 1; i <= numWriters; i++)
        {
        Thread writerThread = new Thread(new Writer(mutex, wrt, i));
        writerThread.start();
        }
        }
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch ReaderWriterSemaphore.java
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano ReaderWriterSemaphore.java
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ javac ReaderWriterSemaphore.java
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ java ReaderWriterSemaphore
Writer 2 is writing
Writer 2 is writing
Reader 1 is reading
Writer 1 is writing
Writer 2 is writing
Reader 3 is reading
Reader 2 is reading
Writer 1 is writing
Reader 3 is reading
Reader 1 is reading
Writer 2 is writing
Reader 2 is reading
Writer 2 is writing
Writer 1 is writing
Reader 1 is reading
Reader 3 is reading
Writer 1 is writing
Writer 2 is writing
Reader 1 is reading
Reader 2 is reading
Reader 3 is reading
Writer 1 is writing
Reader 1 is reading
Writer 2 is writing
Writer 1 is writing
Reader 1 is reading
Reader 2 is reading
Reader 2 is reading
Reader 1 is reading
Writer 2 is writing
Reader 2 is reading
Reader 2 is reading
Reader 3 is reading
Reader 3 is reading
Writer 1 is writing
Reader 1 is reading
Writer 1 is writing
Reader 2 is reading
Writer 2 is writing
Reader 3 is reading
Reader 1 is reading
Writer 1 is writing
Reader 2 is reading
Reader 3 is reading
Writer 2 is writing
```

35. Write a program for chatting between two/three users to demonstrate IPC using message passing (msgget, msgsnd, msgrcv).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>

#define MAX_MESSAGE_SIZE 100
#define MAX_USERS 3

struct message {
        long mtype;          // Message type
        char mtext[MAX_MESSAGE_SIZE];  // Message data
};
```

```
int main() {
        key_t key;
        int msgid;
        struct message msg;

        // Generate a unique key for the message queue
        key = ftok(".", 'a');

        // Create a message queue
        msgid = msgget(key, IPC_CREAT | 0666);
        if (msgid == -1) {
        perror("msgget");
        exit(1);
        }

        printf("Chat program started!\n");

        // Fork child processes for each user
        for (int user = 1; user <= MAX_USERS; user++) {
        pid_t pid = fork();
        if (pid < 0) {
        perror("fork");
        exit(1);
        } else if (pid == 0) {
        printf("User %d: Ready to chat\n", user);

        // Receive messages from other users
        while (1) {
                if (msgrcv(msgid, &msg, sizeof(msg.mtext), user, 0) == -1) {
                perror("msgrcv");
                exit(1);
                }

                printf("User %d: %s", user, msg.mtext);

                // Terminate the chat if the message contains 'bye'
                if (strncmp(msg.mtext, "bye", 3) == 0)
                break;
        }

        exit(0);
```

```
        }
        }

        // Send messages between users
        int sender, receiver;
        while (1) {
        printf("Enter sender and receiver (0 0 to exit): ");
        scanf("%d %d", &sender, &receiver);

        if (sender == 0 || receiver == 0)
        break;

        if (sender < 1 || sender > MAX_USERS || receiver < 1 || receiver >
MAX_USERS) {
        printf("Invalid sender or receiver.\n");
        continue;
        }

        printf("Enter message (max %d characters): ",
MAX_MESSAGE_SIZE - 1);
        scanf(" %[^\n]", msg.mtext);

        msg.mtype = receiver;

        // Send the message
        if (msgsnd(msgid, &msg, strlen(msg.mtext) + 1, 0) == -1) {
        perror("msgsnd");
        exit(1);
        }

        // Terminate the chat if the message contains 'bye'
        if (strncmp(msg.mtext, "bye", 3) == 0)
        break;
        }

        // Terminate child processes
        for (int i = 0; i < MAX_USERS; i++) {
        wait(NULL);
        }

        // Remove the message queue
        if (msgctl(msgid, IPC_RMID, NULL) == -1) {
```

```
        perror("msgctl");
        exit(1);
        }

        printf("Chat program terminated.\n");

        return 0;
}
```

```
^Csupriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 35.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 35.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 35.c
35.c: In function ‘main’:
35.c:93:9: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   93 |         wait(NULL);
      |         ^~~~
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Chat program started!
Enter sender and receiver (0 0 to exit): User 1: Ready to chat
User 2: Ready to chat
User 3: Ready to chat
1 2
Enter message (max 99 characters): Hello user 2
Enter sender and receiver (0 0 to exit): 2 1
Enter message (max 99 characters): heyy user 1
Enter sender and receiver (0 0 to exit): 0 0
```

36. Write a program to demonstrate IPC using shared memory (shmget, shmat, shmdt). In this, one process will take numbers as input from user and another process will sort the numbers.

37. Write a program in which different processes will perform different operation on shared memory. (using shmget, shmat, shmdt).

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#define SHARED_MEMORY_KEY 1234
```

```
#define MAX_SIZE 100
typedef struct {
int data[MAX_SIZE];
int count;
} SharedData;
void writeData(SharedData* sharedData) {
printf("Writing data to shared memory...\n");
// Write data to the shared memory
for (int i = 0; i < sharedData->count; i++) {
sharedData->data[i] = i + 1;
}
printf("Data written to shared memory successfully.\n");
}
void readData(SharedData* sharedData) {
printf("Reading data from shared memory...\n");
// Read and display the data from the shared memory
for (int i = 0; i < sharedData->count; i++) {
printf("%d ", sharedData->data[i]);
}
printf("\n");
}
void modifyData(SharedData* sharedData) {
printf("Modifying data in shared memory...\n");
// Modify the data in the shared memory
for (int i = 0; i < sharedData->count; i++) {
sharedData->data[i] += 10;
}
printf("Data modified successfully.\n");
}
int main() {
int shmid;
SharedData* sharedData;
// Create the shared memory segment
shmid = shmget(SHARED_MEMORY_KEY, sizeof(SharedData),
IPC_CREAT | 0666);
if (shmid == -1) {
perror("shmget");
exit(1);
}
// Attach the shared memory segment to the process's address space
sharedData = (SharedData*)shmat(shmid, NULL, 0);
if (sharedData == (SharedData*)-1) {
```

```
perror("shmat");
exit(1);
}
// Create child processes
pid_t childPid1 = fork();
if (childPid1 == -1) {
perror("fork");
exit(1);
}
if (childPid1 == 0) {
// Child process 1 (write data to shared memory)
writeData(sharedData);
// Detach the shared memory segment
shmdt(sharedData);
exit(0);
}
pid_t childPid2 = fork();
if (childPid2 == -1) {
perror("fork");
exit(1);
}
if (childPid2 == 0) {
// Child process 2 (read data from shared memory)
sleep(1); // Wait for the write operation to complete
readData(sharedData);
// Detach the shared memory segment
shmdt(sharedData);
exit(0);
}
pid_t childPid3 = fork();
if (childPid3 == -1) {
perror("fork");
exit(1);
}
if (childPid3 == 0) {
// Child process 3 (modify data in shared memory)
sleep(2); // Wait for the read operation to complete
modifyData(sharedData);
// Detach the shared memory segment
shmdt(sharedData);
exit(0);
}
```

```
// Wait for all child processes to complete
wait(NULL);
wait(NULL);
wait(NULL);
// Detach and remove the shared memory segment
shmdt(sharedData);
shmctl(shmid, IPC_RMID, NULL);
return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 37.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 37.c
37.c: In function 'main':
37.c:92:1: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
   92 | wait(NULL);
      | ^~~~
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Writing data to shared memory...
Data written to shared memory successfully.
Reading data from shared memory...

Modifying data in shared memory...
Data modified successfully.
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

38. Write programs to simulate linux commands cat, ls, cp, mv, head etc.

39. Write a program to ensure that function f1 should executed before executing function f2 using semaphore. (Ex. Program should ask for username before entering password).

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 39.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 39.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Enter username: supriya_pawar
Function f1 executed successfully.
Enter password: supriya@06
Function f2 executed successfully.
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

40. Write a program using OpenMP library to parallelize the for loop in a sequential program of finding prime numbers in a given range.

41. Using OpenMP library write a program in which master thread count the total no. of threads created, and others will print their thread numbers.

42. Implement the program for IPC using MPI library ("Hello world" program).

43. Write a 2 programs that will both send and messages and construct the following dialog between them
(Process 1) Send the message "Are you hearing me?"
(Process 2) Receives the message and replies "Loud and Clear".
(Process 1) Receives the reply and then says "I can hear you too".
IPC:Message Queues:msgget, msgsnd, msgrcv.

44. Write a program for TCP to demonstrate the socket system calls

45. **Write a program for UDP to demonstrate the socket system calls**

46. **Implement echo server using TCP in iterative/concurrent logic.**

47. **Implement echo server using UDP in iterative/concurrent logic.**

48. **Write a program using PIPE, to Send data from parent to child over a pipe. (unnamed pipe)**

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

#define BUFFER_SIZE 256
```

```
int main() {
        int pipefd[2]; // Pipe file descriptors
        char buffer[BUFFER_SIZE];
        pid_t pid;

        // Create a pipe
        if (pipe(pipefd) == -1) {
        perror("pipe");
        exit(1);
        }

        // Fork a child process
        pid = fork();
        if (pid < 0) {
        perror("fork");
        exit(1);
        } else if (pid == 0) {
        // Child process

        // Close the write end of the pipe since child will only read
        close(pipefd[1]);

        // Read the data from the pipe
        ssize_t bytesRead = read(pipefd[0], buffer, BUFFER_SIZE);
        if (bytesRead == -1) {
        perror("read");
        exit(1);
        }

        // Null-terminate the received data
        buffer[bytesRead] = '\0';

        printf("Child: Received data from parent: %s\n", buffer);

        // Close the read end of the pipe
        close(pipefd[0]);

        exit(0);
        } else {
        // Parent process
```

```
        // Close the read end of the pipe since parent will only write
        close(pipefd[0]);

        printf("Parent: Enter data to send to child: ");
        fgets(buffer, BUFFER_SIZE, stdin);

        // Write the data to the pipe
        ssize_t bytesWritten = write(pipefd[1], buffer, BUFFER_SIZE);
        if (bytesWritten == -1) {
        perror("write");
        exit(1);
        }

        printf("Parent: Sent data to child.\n");

        // Close the write end of the pipe
        close(pipefd[1]);

        // Wait for the child process to complete
        wait(NULL);

        exit(0);
        }

        return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 48.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 48.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 48.c
48.c: In function ‘main’:
48.c:68:9: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   68 |         wait(NULL);
      |         ^~~~
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Parent: Enter data to send to child: Hello..
Parent: Sent data to child.
Child: Received data from parent: Hello..

supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

49. Write a program using FIFO, to Send data from parent to child over a pipe. (named pipe)

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

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define FIFO_NAME "myfifo"
#define BUFFER_SIZE 256

int main() {
        int fd;
        char buffer[BUFFER_SIZE];
        pid_t pid;

        // Create a named pipe (FIFO)
        if (mkfifo(FIFO_NAME, 0666) == -1) {
        perror("mkfifo");
        exit(1);
        }

        // Fork a child process
        pid = fork();
        if (pid < 0) {
        perror("fork");
        exit(1);
        } else if (pid == 0) {
        // Child process

        // Open the named pipe for reading
        fd = open(FIFO_NAME, O_RDONLY);
        if (fd == -1) {
        perror("open");
        exit(1);
        }

        // Read the data from the pipe
        ssize_t bytesRead = read(fd, buffer, BUFFER_SIZE);
        if (bytesRead == -1) {
        perror("read");
        exit(1);
        }
```

```
        // Null-terminate the received data
        buffer[bytesRead] = '\0';

        printf("Child: Received data from parent: %s\n", buffer);

        // Close the pipe
        close(fd);

        exit(0);
        } else {
        // Parent process

        // Open the named pipe for writing
        fd = open(FIFO_NAME, O_WRONLY);
        if (fd == -1) {
        perror("open");
        exit(1);
        }

        printf("Parent: Enter data to send to child: ");
        fgets(buffer, BUFFER_SIZE, stdin);

        // Write the data to the pipe
        ssize_t bytesWritten = write(fd, buffer, BUFFER_SIZE);
        if (bytesWritten == -1) {
        perror("write");
        exit(1);
        }

        printf("Parent: Sent data to child.\n");

        // Close the pipe
        close(fd);

        // Wait for the child process to complete
        wait(NULL);

        exit(0);
        }

        return 0;
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ touch 49.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ nano 49.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 49.c
49.c: In function ‘main’:
49.c:79:9: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
   79 |         wait(NULL);
      |         ^~~~
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Parent: Enter data to send to child: hello
Parent: Sent data to child.
Child: Received data from parent: hello

supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```

50. Write a program using PIPE, to Send files from parent to child over a pipe. (unnamed pipe)

<u>SAME PROGRAM as 48</u>

51. Write a program using FIFO, to Send files from parent to child over a pipe. (named pipe)

<u>SAME PROGRAM as 49</u>

52. Write a program using PIPE, to convert uppercase to lowercase filter to read command/ from file

53. Write a program to illustrate the semaphore concept. Use fork so that 2 process running simultaneously and communicate via semaphore.

Hello Supriya 🙂

54. Write 3 programs separately, 1st program will initialize the semaphore and display the semaphore ID. 2nd program will perform the P operation and print message accordingly. 3rd program will perform the V operation print the message accordingly for the same semaphore declared in the 1st program.

55. Write a program to demonstrate the lockf system call for locking.

56. Write a program to demonstrate the flock system call for locking.

```
#include <stdio.h>
        #include <stdlib.h>
        #include <errno.h>
        #include <fcntl.h>
        #include <unistd.h>

        int main(int argc, char *argv[])
        {
                        /* l_type   l_whence  l_start  l_len  l_pid   */
        struct flock fl = { F_WRLCK, SEEK_SET, 0,       0,      0 };
        int fd;

        fl.l_pid = getpid();
```

```
if (argc > 1)
        fl.l_type = F_RDLCK;

if ((fd = open("lockdemo.c", O_RDWR)) == -1) {
        perror("open");
        exit(1);
}

printf("Press <RETURN> to try to get lock: ");
getchar();
printf("Trying to get lock...");

if (fcntl(fd, F_SETLKW, &fl) == -1) {
        perror("fcntl");
        exit(1);
}

printf("got lock\n");
printf("Press <RETURN> to release lock: ");
getchar();

fl.l_type = F_UNLCK;  /* set to unlock same region */

if (fcntl(fd, F_SETLK, &fl) == -1) {
        perror("fcntl");
        exit(1);
}

printf("Unlocked.\n");

close(fd);
}
```

```
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx: ~/Documents/UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~$ cd Documents
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents$ cd UOS_Lab_ESE
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ gcc 56.c
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$ ./a.out
Press <RETURN> to try to get lock: RETURN
Trying to get lock...got lock
Press <RETURN> to release lock: Unlocked.
supriya@supriya-HP-Pavilion-Laptop-15-eg0xxx:~/Documents/UOS_Lab_ESE$
```