Graphic Era HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

Term-Work

Of

Operating System Lab (PCS-506)

Submitted in partial fulfilment of the requirement for the V semester

Bachelor of Technology

By

Navneet Bhatt

2261383

5th Sem

Under the Guidance of

Faculty-in-Charge

Mr. Ansh Dhingra

Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI

2024-2025

Graphic Era HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

STUDENT'S DECLARATION

I Navneet Bhatt hereby declare the work which is being presented in the term-work, entitled "Operating System Lab" in partial fulfilment of the requirement for the award of the degree **B-Tech** in the session **2024-25**, is an authentic record of my own work carried out under the supervision of Mr. Ansh Dhingra. The matter embodied in this term-work has not been submitted by me for the award of any other degree.

Date: Navneet Bhatt

Graphic Era HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

ACKNOWLEDGEMENT

We take immense pleasure in thanking Honorable **"Mr. Ansh Dhingra"** (**Assistant Professor, CSE, GEHU Bhimtal Campus**) to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me everything that we need. We again want to extend thanks to our President **"Prof. (Dr.) Kamal Ghanshala"** for providing us all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to Professor **"Col. Anil Kumar"** (Director Gehu Bhimtal),

other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this thesis.

Finally, yet importantly, we would like to express my heartiest thanks to our beloved parents,

for their moral support, affection and blessings. We would also like to pay our sincere thanks to all our friends and well-wishers for their help and wishes for the successful completion of this research.

Navneet Bhatt

Graphic Era HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

INDEX

S.NO.	OBJECTIVE	DATE	SIGN
1.	Demonstration of FORK() System Call		
2.	Parent Process Computes the SUM OF EVEN and Child Process Computes the sum of ODD NUMBERS using fork		
3.	Demonstration of WAIT() System Call		
4.	Implementation of ORPHAN PROCESS & ZOMBIE PROCESS		
5.	Implementation of PIPE		

1. Demonstration of FORK() System Call

```
#include<stdio.h>
#include<unistd.h> int
main()
{
        pid_t pid = fork();
        if(pid<0)
        {
                printf("fork failed\n");
        }
        else if(pid==0)
        {
                printf("this is the child process\n");
        }
        else{
                printf("this is the parent process with child pid %d\n",pid);
        }
        return 0;
}
```

2. Parent Process Computes the SUM OF EVEN and Child Process Computes the sum of ODD NUMBERS using fork

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h> #include<sys/types.h>
int main()
{
        pid_t pid=fork();
        if(pid<0)
        {
                perror("fork failed\n");
                exit(1);
        }
        else if(pid==0)
        {
                int sum_odd=0;
                int i;
                for(i=1;i<=10;i++)
                {
                        if(i%2 !=0)
                        {
                          sum_odd = sum_odd + i;
                    }
                }
                printf("child process : sum of odd numbers is %d\n",sum_odd);
        }
        else
        {
                int sum_even = 0;               int i;
                for(i=1;i<=10;i++)
                {
                        if(i%2==0)
                        {
                                sum_even=sum_even+i;
                        }
                }
                printf("parent process : sum of even numbers is %d\n",sum_even);
        }
        return 0;
}
```

3. Demonstration of WAIT() System Call

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

int main() {
   pid_t p = fork(); // Create a new process

   if (p < 0) {
      // Error handling if fork fails          perror("Fork failed");
      exit(1);
   }
   else if (p == 0) {         // Child
process
      printf("Child process: My process ID is %d\n", getpid());
printf("Child process: Doing some work...\n");          sleep(2); // Simulate
work with sleep         printf("Child process: Work done, exiting now.\n");
      exit(0); // Exit the child process
   }    else {
      // Parent process
      printf("Parent process: Waiting for child process to complete...\n");
wait(NULL); // Wait for the child process to complete          printf("Parent process:
Child process has completed.\n");          printf("Parent process: My process ID is
%d\n", getpid());
   }

   return 0;
}
```

4. Implementation of ORPHAN PROCESS & ZOMBIE PROCESS

a) ORPHAN PROCESS :

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

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        // Error handling if fork fails          perror("Fork failed");
        exit(1);
    }
    else if (pid == 0) {          // Child
process
        sleep(3);  // Simulate some work
        printf("Child process: My parent has terminated, I am now an orphan. My PID is
%d\n", getpid());
        printf("Child process: My new parent process ID is %d\n", getppid());  // Adopted by
init/systemd
    }     else {
        // Parent process
        printf("Parent process: Terminating before child completes. My PID is %d\n", getpid());
        exit(0);  // Parent process terminates
    }

    return 0;
}
```

b) ZOMBIE PROCESS :

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

int main() {     pid_t
pid = fork();

   if (pid < 0) {
       // Error handling if fork fails
perror("Fork failed");
       exit(1);
   }
   else if (pid == 0) {
// Child process
       printf("Child process: My process ID is %d\n", getpid());
printf("Child process: Exiting now.\n");
       exit(0);  // Child process terminates
   }
else {
       // Parent process
       printf("Parent process: Child process created. My PID is %d\n", getpid());
sleep(5);  // Parent sleeps, keeping the child as a zombie
       printf("Parent process: Checking on the child process. It should be a zombie
now.\n");

       // Wait for the child process to complete (reap the zombie)
wait(NULL);
       printf("Parent process: Child process has been reaped.\n");
   }

   return 0;
}
```

5. . Implementation of PIPE

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

int main() {
   int fd[2];  // fd[0] is for reading, fd[1] is for writing
pid_t pid;
   char write_msg[] = "Hello from parent to child!";
   char read_msg[100];

   // Create a pipe      if
(pipe(fd) == -1) {
perror("Pipe failed");
       return 1;
   }
   // Fork a new process
pid = fork();

   if (pid < 0) {      // Error
in fork           perror("Fork
failed");
       return 1;
   }
   else if (pid == 0) {
       // Child process: Reads from pipe           close(fd[1]);  //
Close the write end of the pipe in child           read(fd[0],
read_msg, sizeof(read_msg));  // Read from pipe
printf("Child process received message: %s\n", read_msg);
close(fd[0]);  // Close the read end of the pipe in child
   }
else {
       // Parent process: Writes to pipe
       close(fd[0]);  // Close the read end of the pipe in parent
write(fd[1], write_msg, strlen(write_msg) + 1);  // Write to pipe
printf("Parent process sent message: %s\n", write_msg);
close(fd[1]);  // Close the write end of the pipe in parent

       // Parent waits for child to finish
wait(NULL);
   }

   return 0;
}
```