DRONACHARYA College of Engineering

LABORATORY MANUAL

B.Tech. Semester- IV

OPERATING SYSTEM LAB
Subject code: LC-CSE-212G

Prepared by:	**Checked by:**	**Approved by:**
Prof. Pankaj Kumari	Prof. Yashwardhan Soni	Name : Prof. (Dr.) Isha Malhotra
Sign.: …………………….	**Sign.:** ………………….	**Sign.:** ………………….

DEPARTMENT OF CSE/CSIT/IT/IOT
DRONACHARYA COLLEGE OF ENGINEERING
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)

Table of Contents

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objectives (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (COs)
8. CO- PO and CO-PSO mapping
9. Course Overview
10. List of Experiments
11. DOs and DON'Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Lab Experiments

Vision and Mission of the Institute

Vision:

“Empowering human values and advanced technical education to navigate and address global challenges with excellence.”

Mission:

- M1 - Seamlessly integrate human values with advanced technical education.
- M2 - Supporting the cultivation of a new generation of innovators who are not only skilled but also ethically responsible.
- M3 - Inspire global citizens who are equipped to create positive and sustainable impact, driving progress towards a more inclusive and harmonious world.

Vision and Mission of the Department

Vision:

- “Preparing technologists with in-depth insights into information technology, and embedding ethics via focused technical training.

Mission:

- Empower technologists to excel in information technology through rigorous training and hands-on experience.
- Foster a culture of integrity and responsibility by instilling ethical principles in every aspect of technical education.
- Encourage technologists with new ideas and good leadership in the tech world, training to possess strong values.

Programme Educational Objectives (PEOs)

- Demonstrate technical competence with analytical and critical thinking to understand and meet the requirements of Industry, academia and research.
- Exhibit leadership, team skills and entrepreneurship skills to provide solutions to real world problems.
- Work in multi-disciplinary industries with social and environmental responsibility, work ethics and adaptability to address engineering and social problems.

PROGRAM OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

P11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

P12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological chan

PROGRAM SPECIFIC OUTCOMES (PSOs)

- Have proficiency in programming skills to design, develop and apply appropriate techniques, for solving engineering problems.
- Have knowledge to build, automate and manage business solutions using advanced technologies.
- Have pleasure towards research in applied computer technologies.

University Syllabus

Course code	LC-CSE-212G					
Category	Lab Course					
Course title	Operating System Lab					
Scheme and Credits	L	T	P	Credits	Semester = 4	
	0	0	4	2		
Classwork	25 Marks					
Exam	25 Marks					
Total	50 Marks					
Duration of Exam	03 Hours					

1. Introduction to UNIX File System.

2. File and Directory Related Commands in UNIX.

3. Essential UNIX Commands for working in UNIX environment.

4. I/O Redirection and Piping

5. Introduction to VI Editors.

6. Introduction of Processes in UNIX

7. Communication in UNIX and AWK.

8. Introduction of the concept of Shell Scripting.

9. Decision and Iterative Statements in Shell Scripting.

10. Writing the Shall Scripts for unknown problems

Course Outcomes (COs)

Upon successful completion of the course, the students will be able to:

CO1: Understand the structure and architectural components of UNIX Operating System to analyze and design the problem. Moreover, students would be able to know the Basic Introduction of UNIX Operating System.

CO2: Basic Introduction of UNIX Commands that are used for operating the UNIX.

CO3: Introduction of Shell Scripting and VI Editor.so that the students get familiar with writing the UNIX scripts in UNIX editor.

CO4: Students will establish themselves as effective professionals by solving real problems with UNIX Shell Scripting knowledge and with attention to teamwork, critical thinking and problem-solving skills by Writing Shell Scrips of unknown problems

CO-PO Mapping

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2	2	-	-	-	-	-	-	-	-	-
CO2	3	2	2	-	-	-	-	-	-	-	-	-
CO3	3	2	2	-	-	-	-	-	-	-	-	-
CO4	3	2	2	-	-	-	-	-	-	-	-	-

CO-PSO Mapping

CO	PSO1	PSO2	PSO3	PSO4
CO1	3	2	1	-
CO2	3	2	1	-
CO3	3	2	1	-
CO4	3	2	1	-

Course Overview

Operating System Lab is a course that teaches students about the fundamentals of operating systems. The lab sessions are designed to give students hands-on experience with operating system concepts, such as process management, memory management, file systems, and inter-process communication.

The topics covered in Operating System Lab include:

* Process management

* Memory management

* File systems

* Inter-process communication

* Scheduling

* Virtual memory

* Distributed operating systems

List of Experiments mapped with COs

Program No	List of Programs	CO
1.	Study of BASIC UNIX COMMANDS	CO1, CO2
2.	Study of BASIC DOS COMMANDS	CO2
3.	PROGRAM TO FIND THE LARGEST AMONG THREE NUMBERS.	CO1
4.	PROGRAM TO FIND THE ADDITION OF TWO NUMBER'S	CO1
5.	PROGRAM TO PRINT THE NUMBERS FROM 5 TO 1	CO1
6.	PROGRAM TO PRINT THE CURRENT DATE.	CO2
7.	PROGRAM TO DISPLAY THE BIODATA.	CO2
8.	PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEMS (OPENDIR, READDIR, CLOSEDIR)	CO3
9.	PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEM	CO3
10.	TO WRITE A C PROGRAM FOR IMPLEMENTATION OF FCFS AND SJF SCHEDULING ALGORITHMS.	CO4

DOs and DON'Ts

DOs

1. Login-on with your username and password.
2. Log off the Computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.
2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.
3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.
4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.
2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible, isolate the burning equipment from the other surrounding equipment.
3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.
4. Call security and emergency department immediately:

Emergency : Reception
Security: Front Gate

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

1) All files must contain a title page followed by an index page. ***The files will not be signed by the faculty without an entry in the index page.***

2) Student's Name, roll number and date of conduction of experiment must be written on all pages.

3) For each experiment, the record must contain the following
    (i) Aim/Objective of the experiment
    (ii) Pre-experiment work (as given by the faculty)
    (iii) Lab assignment questions and their solutions
    (iv) Test Cases (if applicable to the course)
    (v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.
2. Students must ensure that their lab record is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	**Exemplary (4)**	**Competent (3)**	**Needs Improvement (2)**	**Poor (1)**
<u>AC1:</u> **Pre-Lab written work (this may be assessed through viva)**	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
<u>AC2:</u> **Program Writing/ Modeling**	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
<u>AC3:</u> **Identification & Removal of errors/ bugs**	Able to identify errors/ bugs and remove them	Able to identify errors/ bugs and remove them with little bit of guidance	Is dependent totally on someone for identification of errors/ bugs and their removal	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out
<u>AC4:</u> **Execution & Demonstration**	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
<u>AC5:</u> **Lab Record Assessment**	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done	

	variants of input and output	with all variants of input and output	with all variants of input and output	

LAB EXPERIMENTS

PROGRAM NO. 1

PRE-EXPERIMENT QUESTIONS:

1. What is your prior experience with Unix or Unix-like operating systems?
2. Have you used any basic Unix commands before? If so, which ones?
3. Are you familiar with the command-line interface and comfortable working in a terminal?

BASIC UNIX COMMANDS

Date command:

Unix maintains a system clock. As for now you can simply display the current date with the date command, which shows the

date and time for the nearest second as shown:

$ date: Thur oct 4 11:23:52 IST 1999

$ date +%m : This command shows current month(in digits) [eg. 08]

$ date +%h : This command shows current month(in words) [eg. Aug]

Cal command:

This command is used for printing the calendar of any particular month or the entire year.

$ cal 2000: This command will show the calender of 2000.

$ Cal 09 2007- This command will display the calender of September,2007

The head command:

The head command is used to display the top few records of the file. The syntax of the

command is as follows:

$ head -5 filename

This command will display the top 5 lines of filename.

The tail command:

The tail command displays the end few records of the file. If no line count is given, the tail command displays the last ten lines of the file.

$ tail –3 emp.lst

This command will display the last 3 lines of emp.lst .

To create a file:

```
$  cat  >seema
is  a  good  girl
```

To save a file:

```
ctrl+d
```

To display a file:

```
$ cat seema
is a good girl
```

ls:

This command is used to list all the files or directories.

```
$ ls – x
```

This command will display all the files column wise.

```
$ ls –a
```

This command will display all the hidden files

To copy one file into another:

```
$ cp dog jam
```

it will copy the content of dog file into jam file.

To rename a file:

```
$ mv  fish whale
```

it will rename a file name fish to file name whale.

To remove a file:

```
$ rm whale
```

it remove the file whale.

To show the current directory in which you are working:

```
$ pwd
```

To clear the screen:

```
$ tput clear
```

To make a directory:

```
$ mkdir school
```

This command will create a directory name school.

```
$ mkdir apple grapes banana
```

This command will create all the above directories in one go.

To remove a directory:

```
$ rmdir school
```

To use the calculator:

```
$ bc
12+5
```

This command will display the result 17.

```
12+5;12-5;12*5
```

This command will display the result 17 7 60.

```
Scale=2
14/3
```

here Scale is used to display the result upto 2 decimal places.

To come out from the UNIX:

```
$ exit
```

POST-EXPERIMENT QUESTIONS:

1. What is the purpose of the "ls" command in UNIX? How does it differ from "ls -l" and "ls -a"?
2. How do you create a new directory using the "mkdir" command? Can you create multiple directories at once?
3. What is the purpose of the "cd" command? How do you navigate to a specific directory using this command?

PROGRAM NO-2

PRE-EXPERIMENT QUESTIONS:

1. What is the purpose of the "dir" command in DOS?
2. How can you change the current directory using the "cd" command?
3. What is the syntax for creating a new directory using the "mkdir" command?

BASIC DOS COMMANDS

1. File commands.

(a) Copy

C:\>copy source-path_name destination-path_name

This command will copy source file into destination file.

(b) Delete

C:\>del abc

This command will delete the abc file.

(c) Rename

C:\>ren abc xyz

This command will rename the file name abc to file name xyz.

(d) Type

C:\>type xyz

This command will display the contents of file xyz.

2. Directory command

(a) DIR

This command is used to list the files under a directory.

C:\>Dir

This command will show all directories

Dir\w :- width wise

Dir\p :- page file

(b) CHDIR

This command changes working directory to directory you specify.

C:\> cd e:

This command changes C directory to E.

E:\>

C:\>cd..

This command make the parent directory the root directory

C:\> cd\

This command makes the root directory as working directory.

(c) MKDIR (MD)

This command is used to create a sub-directory.

C:\> md java

This command will create a directory java.

3. MISCELLANEOUS COMMANDS

(a) DATE

C:\> Date

To display current date and enter the new date if you want to change

(b)Time

C:\>Time

To display the time currently and enter new time.

(c) VERSION

C:\>ver

This command displays the version of MS- DOS you are working upon.

(d) CLEAR SCREEN

```
C:\>cls
```

This command clears the screen.

(e) PATH

```
C:\>path c:\>java\bin
```

This command will set the above path.

POST-EXPERIMENT QUESTIONS:

1. What is the purpose of the "dir" command in DOS, and how does it work?
2. How can you create a new directory using the "mkdir" command in DOS? Can you provide an example?
3. What is the purpose of the "cd" command in DOS, and how can you use it to navigate through directories?

PROGRAM NO-3

PRE-EXPERIMENT QUESTIONS:

1. How was the data collected during the experiment?
2. Were there any control variables or control groups used in the experiment?
3. What were the main findings or results of the experiment?

1.PROGRAM TO FIND THE LARGEST AMONG THREE NUMBERS.

```
echo largest among three numbers

echo Enter number1

read a

echo Enter number2

read b

echo Enter number3

read c

if [ $a -gt $b ] && [ $a -gt $c ]

then

echo $a is greater

elif [ $b -gt $a ] && [ $b -gt $c ]

then echo $b is big

else echo $c is big

fi

~

~

"large" 33L, 268C                              12,1          All
```

OUTPUT

```
[user9@localhost user9]$ sh large

largest among three numbers

Enter number1

34

Enter number2

73

Enter number3

44

73 is big

[user9@localhost user9]$
```

POST-EXPERIMENT QUESTIONS:

1. How was the data collected during the experiment?
2. Were there any control variables or control groups used in the experiment?
3. What were the main findings or results of the experiment?

PROGRAM NO-4

PRE-EXPERIMENT QUESTIONS:

1. What is the purpose of creating this program?
2. Who will be the intended users of this program? Will it be designed for general users, students, or specific professionals?
3. How will you handle invalid input or errors?

PROGRAM TO FIND THE ADDITION OF TWO NUMBERS

```
echo addition of two numbers

echo Enter num1

read a

echo Enter num2

read b

c=`expr $a + $b`

echo addition is $c

~

~


~

~

~

"add" 7L, 113C                    6,1        All
```

OUTPUT

```
[user9@localhost user9]$ sh add

addition of two numbers

Enter num1

4

Enter num2

5

addition is 9

[user9@localhost user9]$
```

POST-EXPERIMENT QUESTIONS:

1. How did you design and implement the program to find the addition of two numbers?
2. What programming language did you use for the implementation?
3. Did you encounter any challenges or difficulties during the coding process? If so, how did you overcome them?

PROGRAM NO-5

PRE-EXPERIMENT QUESTIONS:

1. What programming language will you be using to implement this program?
2. Will the program be executed in a console/terminal or as a graphical application?
3. Should the program print the numbers in ascending or descending order?

3.PROGRAM TO PRINT TNE NUMBERS FROM 5 TO 1

```
i=5

echo sequence is

while [ $i != 0 ]

do

    echo $i

    i=`expr $i - 1`

done

~

~

~

"print" 8L, 80C                                   8,1         All
```

OUTPUT

```
[user9@localhost user9]$ sh print
sequence is
5
4
3
2
1
[user9@localhost user9]$
```

POST-EXPERIMENT QUESTIONS:

1. How does the **for** loop work in this program?
2. What is the significance of the values used in the **range** function?
3. What would happen if you change the step size in the **range** function to a positive value?

PROGRAM NO-6

PRE-EXPERIMENT QUESTIONS:

1. What programming language will you be using for this program?
2. Are there any specific requirements or constraints for the program?
3. Should the program output the date in a specific format? If yes, what is the desired format?

4. PROGRAM TO PRINT THE CURRENT DATE.

```
echo current date is `date`

echo user is `who am i`

echo current dir `pwd`



~

~

~

~

"date" 4L, 76C                                   3,1          All
```

OUTPUT

```
[user9@localhost user9]$ sh date

current date is Wed Nov 14 11:33:58 IST 2007

user is localhost.localdomain!user9 pts/3 Nov 14 11:24 (192.168.1.49)

current dir /home/user9

[user9@localhost user9]$
```

POST-EXPERIMENT QUESTIONS:

1. How did you implement the program to retrieve the current date?
2. Did you encounter any challenges while developing the program? If so, what were they, and how did you overcome them?
3. Is the current date displayed in a specific format? If yes, what format did you choose and why?

PROGRAM NO-7

PRE-EXPERIMENT QUESTIONS:

1. How will the program retrieve and store the biodata? Will it read from a file or a database, or will the data be hardcoded into the program itself?
2. What format should the biodata be displayed in? Should it be a plain text output, or would you prefer a formatted layout with headings and sections?
3. Will the program have any filtering or sorting capabilities? For example, can users search for specific individuals based on criteria like age, education, or skills?

PROGRAM TO DISPLAY THE BIODATA.

```
echo "Enter Your name"

read a

echo "Enter your age"

read b
```

```
echo "Enter your father's name"
read c
echo "Enter your mother's name"
read d
echo "Enter your address"
read e
echo your name is $a
echo your age is $b
echo your father's name is $c
echo your mother's name is $d
echo your address is $e
~
~
~
"biodata" 15L, 297C                    15,1      All
```

OUTPUT

```
biodata" 15L, 297C written
```

```
[user9@localhost user9]$ sh biodata

Enter Your name

Seema

Enter your age

20

Enter your father's name

Mr.Satyapal  yadav

Enter your mother's name

Mrs.Sushila yadav

Enter your address

Delhi

your name is Seema

your age is 20

your fathers name is Mr.Satyapal yadav

echo your mothers name is Mrs. Sushila yadav

your address is Delhi

[user9@localhost user9]$
```

POST-EXPERIMENT QUESTIONS:

1. What specific operating system(s) was the program designed to run on?
2. What programming language was used to develop the program?
3. How was the program executed or launched on the operating system?

PROGRAM NO-8

PRE-EXPERIMENT QUESTIONS:

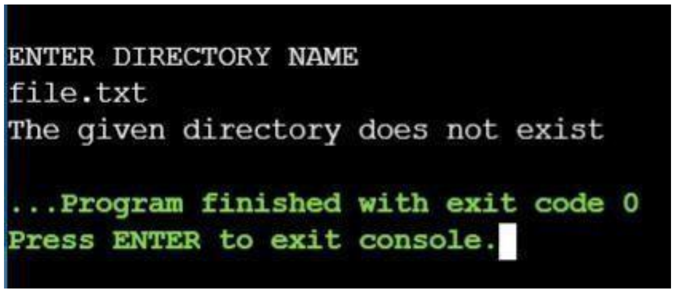
1. How does the **opendir** system call handle errors?
2. What are some potential use cases for the **opendir** system call?
3. What is the difference between **opendir** and **open** system calls?

PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEMS (OPENDIR, READDIR, CLOSEDIR)

```
#include<stdio.h>

#include<dirent.h>

struct dirent *dptr;

int main(int argc, char *argv[])

{
```

```
char buff[100]; DIR

*dirp;

printf("\n\n ENTER DIRECTORY NAME"); scanf("%s", buff); if((dirp=opendir(buff))==NULL)

{

printf("The given directory does not exist");

exit(1); }

while(dptr=readdir(dirp))

{

printf("%s\n",dptr->d_name);

}

closedir(dirp);

}
```

OUTPUT:



```
ENTER DIRECTORY NAME
file.txt
The given directory does not exist

...Program finished with exit code 0
Press ENTER to exit console.
```

POST-EXPERIMENT QUESTIONS:

1. How does the **readdir** function relate to the **opendir** system call?
2. What happens if a directory is modified or deleted while a directory stream is open?
3. Are there any security considerations to keep in mind when using the **opendir** system call?

PROGRAM NO-9

PRE-EXPERIMENT QUESTIONS:

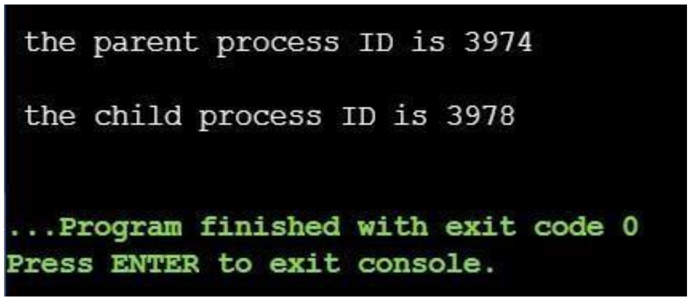
1. What is the purpose of the **fork** system call?
2. How does the **fork** system call work? What does it return?
3. What is the purpose of the **getpid** system call?

PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEM (fork, getpid, exit)

PROGRAM:

```
#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/types.h>

int main() {  int pid, pid1, pid2; pid = fork();
if (pid == -1) {

printf("ERROR IN PROCESS CREATION \n");

exit(1);  }  if

(pid != 0) {

pid1 = getpid();

printf("\n the parent process ID is %d\n", pid1);

} else {    pid2

= getpid();

printf("\n the child process ID is %d\n", pid2);

}

return 0;

}
```

OUTPUT:



```
 the parent process ID is 3974

 the child process ID is 3978


...Program finished with exit code 0
Press ENTER to exit console.
```

POST-EXPERIMENT QUESTIONS:

1. What happens when the **exit** system call is executed? How does it affect the process and the overall system?
2. Describe the value that is typically returned by a process when it calls the **exit** system call. What does this value represent?
3. Can a process exit without calling the **exit** system call? If so, how does this happen?

PROGRAM NO-10

PRE-EXPERIMENT QUESTIONS:

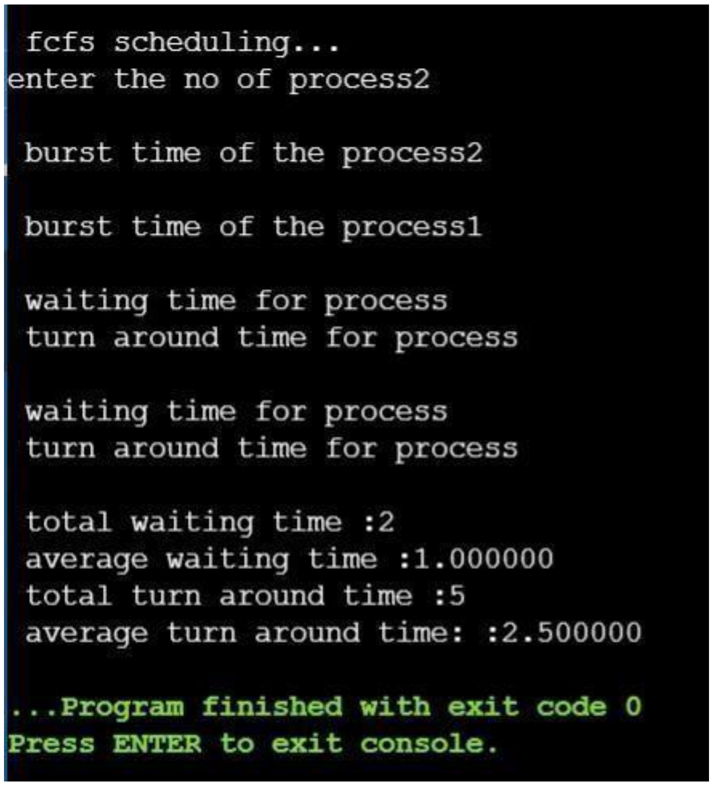
1. What is the purpose of the FCFS and SJF scheduling algorithms?
2. What is the basic idea behind the FCFS scheduling algorithm?
3. What is the basic idea behind the SJF scheduling algorithm?

To write a C program for implementation of FCFS and SJF scheduling algorithms.

```
#include<stdio.h>
#include<stdlib.h>
struct fcfs
{ int pid;
int btime;
int
```

```
wtime;
int ttime;
} p[10];int main() { int i,n;
 int
 towtwtime=0,totttime=0;
 printf("\n fcfs scheduling...\n");
 printf("enter the no of
 process"); scanf("%d",&n);
 for(i=0;i<n;i++)
 {
 p[i].pid=1;
 printf("\n burst time of the process”);
 scanf("%d",&p[i].btime);
 }
p[0].wtime=0;
p[0].ttime=p[0].btime;
totttime+=p[i].ttime;
for(i=0;i<n;i++)
{
p[i].wtime=p[i-1].wtime+p[i-1].btim
p[i].ttime=p[i].wtime+p[i].btime;
totttime+=p[i].ttime;
towtwtime+=p[i].wtime;
}
for(i=0;i<n;i++)
{{
printf("\n waiting time for process”); printf("\n turn around time for process”); printf("\n"); }}
printf("\n total waiting time :%d", totwtime ); printf("\n average waiting
time:%f",(float)totwtime/n); printf("\n total turn around time
:%d",totttime); printf("\n average turn around time:
:%f",(float)totttime/n);
```

OUTPUT:



```
 fcfs scheduling...
enter the no of process2

 burst time of the process2

 burst time of the process1

 waiting time for process
 turn around time for process

 waiting time for process
 turn around time for process

 total waiting time :2
 average waiting time :1.000000
 total turn around time :5
 average turn around time: :2.500000

...Program finished with exit code 0
Press ENTER to exit console.
```

POST-EXPERIMENT QUESTIONS:

1. What are the advantages and disadvantages of using the FCFS scheduling algorithm?
2. What are the advantages and disadvantages of using the SJF scheduling algorithm?
3. What are the key data structures and variables required to implement the FCFS scheduling algorithm in a C program?

This lab manual has been updated by

Prof. Pankaj Kumari (pankaj.kumari@ggnindia.dronacharya.info)

Crosschecked By

HOD CSE

Please spare some time to provide your valuable feedback.