

Operating Systems

Lab Manual

Submitted in the partial fulfillment of the
requirements for the award of Degree of

Bachelor of Engineering

in

Computer Science and Engineering

By

NAME: Sriharini Margapuri

ROLL NO: 1005-21-733065



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (A)
Osmania University, Hyderabad – 500 007 2022-2023

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING (A)
Osmania University, Hyderabad – 500 007**

CERTIFICATE

This is to certify that Sriharini Margapuri bearing
Roll no: 1005-21-733065 studying B.E. V Semester has successfully completed
”Operating Systems Lab” for the academic year 2023-24.

Internal Examiner

External Examiner

TABLE OF CONTENTS

| Programs | Page No. |
|---|-----------------|
| Program 1: Write a program to demonstrate open(), close(), read(), write() system calls. | 1 |
| Program 2: Write a program to demonstrate fork() system call. | 2 |
| Program 3: Write a program to demonstrate multiple fork(). | 3 |
| Program 4: Write a program to demonstrate pipe: A. Parent Writing, Child Reading B. Child Writing, Parent Reading | 5 |
| Program 5: Write a program for FCFS (non-preemptive) scheduling algorithm. | 7 |
| Program 6: Write a program for FCFS (preemptive) scheduling algorithm. | 8 |
| Program 7: Write a program for SJF (non-preemptive) algorithm. | 10 |
| Program 8: Write a program for SJF (preemptive) algorithm. | 12 |
| Program 9: Write a program for Priority (non-preemptive) algorithm. | 14 |
| Program 10: Write a program for Priority (preemptive) algorithm. | 15 |
| Program 11: Write a program for Round Robin scheduling algorithm. | 17 |
| Program 12: Write a program to demonstrate: A. First Fit B. Best Fit C. Worst Fit | 19 |
| Program 13: Write a program for FIFO page replacement algorithm. | 24 |
| Program 14: Write a program for Optimal page replacement algorithm. | 26 |
| Program 15: Write a program for LRU page replacement algorithm. | 28 |
| Program 16: Write a program to demonstrate producer-consumer problem. | 30 |
| Program 17: Write a program to demonstrate readers-writers problem. | 32 |
| Program 18: Write a program to demonstrate dining-philosophers problem. | 34 |
| Program 19: Write a program to demonstrate file reading and writing. | 37 |

TABLE OF CONTENTS

| Shell Script | Page No. |
|--|-----------------|
| 1. Write a shell script to show basic Linux commands. | 38 |
| 2. Write an interactive shell script that takes one input. | 38 |
| 3. Write an interactive shell script that takes multiple user inputs. | 39 |
| 4. Write a shell script that takes file names as input and copies first file into the second file. | 39 |
| 5. Write shell script which will accept 5 numbers as parameters and display their sum. Also display the contents of the different variables in the script. | 40 |
| 6. Write a shell script which will accept different numbers and finds their sum. The number of parameters can vary. | 40 |
| 7. Write a shell script that takes user's answers and returns whether it is correct using case. | 41 |
| 8. Write a shell script that takes user's answers and keeps asking for inputs until user gets the answer right using while. | 41 |
| 9. Write a shell script that asks for user login name using until. | 42 |
| 10. Write a shell script that takes 3 numbers from the user and determines the largest of those 3 numbers using if. | 43 |
| 11. Write a shell script to demonstrate arithmetic operators. | 44 |
| 12. Write a shell script to demonstrate do-while. | 45 |
| 13. Write a shell script to demonstrate if-else. | 46 |
| 14. Write a shell script to demonstrate case. | 47 |
| 15. Write a shell script to demonstrate logical operators. | 48 |

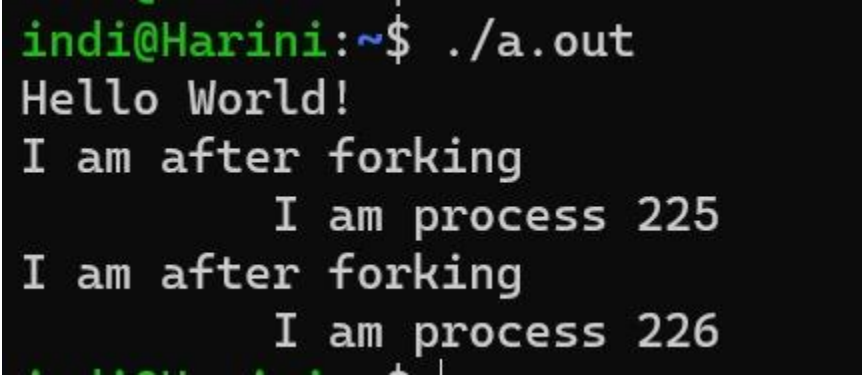
Program 1: Write a program to demonstrate open(), close(), read(), write() system calls.

```
1 #include<fcntl.h>
2 #include<sys/types.h>
3 #include<sys/stat.h>
4 #include<stdio.h>
5 #include<stdlib.h>
6
7 static char message[]="Hello world";
8
9 int main(){
10     int fd;
11     char *buffer[80];
12     fd=open("datafile.dat",O_RDWR|O_CREAT|O_EXCL,S_IREAD|S_IWRITE);
13     if(fd!=-1){
14         printf("File is opened for read/write access\n");
15         write(fd,message,sizeof(message));
16         close(fd);
17         fd=open("datafile.dat",O_RDWR,S_IREAD|S_IWRITE);
18         if(read(fd,buffer,sizeof(message))==sizeof(message)){
19             printf("\n  %s\n  was written to datafile.dat\n",buffer);
20         }else{
21             printf("*** error reading datafile.dat ***");
22         }
23         close(fd);
24     }else{
25         printf("datafile.dat already exists\n");
26         fd=open("datafile.dat",O_RDWR,S_IREAD|S_IWRITE);
27         if(read(fd,buffer,sizeof(message))==sizeof(message)){
28             printf("\n  %s\n  was written to datafile.dat\n",buffer);
29         }else{
30             printf("*** Error reading datafile.dat ***");
31         }
32         close(fd);
33     }
34     exit(0);
35 }
```

```
● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Use
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc systemcall.c -o systemcall }
File is opened for read/write access
" Hello world" was written to datafile.dat
```

Program 2: Write a program to demonstrate fork() system call.

```
1 #include<stdio.h>
2 #include<unistd.h>
3
4 int main(void){
5     printf("Hello World!\n");
6     fork();
7     printf("I am after forking\n");
8     printf("\t I am process %d \n", getpid());
9 }
```



A terminal window with a black background and green text. The prompt is 'indi@Harini:~\$'. The user has entered './a.out'. The output shows 'Hello World!' followed by two lines of 'I am after forking' with indented process IDs. The first line shows 'I am process 225' and the second line shows 'I am process 226'. A cursor is visible at the end of the second line of output.

```
indi@Harini:~$ ./a.out
Hello World!
I am after forking
        I am process 225
I am after forking
        I am process 226
```

Program 3: Write a program to demonstrate multiple fork().

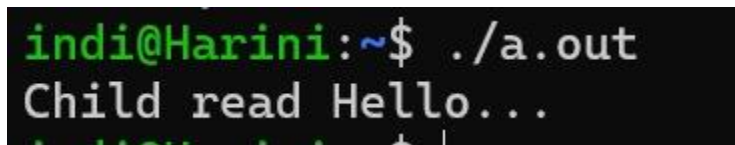
```
1 #include<stdio.h>
2 #include<unistd.h>
3
4 int main(void){
5     printf("Here I'm just before 1st forking statement\n");
6     fork();
7     printf("Here I'm just after 1st forking statement\n");
8     fork();
9     printf("Here I'm just after 2nd forking statement\n");
10    fork();
11    printf("Here I'm just after 3rd forking statement\n");
12    printf("\tHello world from process %d\n", getpid());
13}
```

```
indi@Harini:~$ cc multiplefork.c
indi@Harini:~$ ./a.out
Here I'm just before 1st forking statement
Here I'm just after 1st forking statement
Here I'm just after 1st forking statement
Here I'm just after 2nd forking statement
Here I'm just after 2nd forking statement
Here I'm just after 2nd forking statement
Here I'm just after 2nd forking statement
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
        Hello world from process 234
        Hello world from process 233
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
        Hello world from process 237
        Hello world from process 235
        Hello world from process 238
Here I'm just after 3rd forking statement
Here I'm just after 3rd forking statement
        Hello world from process 239
        Hello world from process 236
Here I'm just after 3rd forking statement
        Hello world from process 240
```


Program 4: Write a program to demonstrate pipe:

A. Parent Writing, Child Reading

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4
5 #define SIZE 1024
6
7 int main(){
8     int pfd[2],nread,pid;
9     char buf[SIZE];
10    if(pipe(pfd)==-1){
11        perror("Pipe failed");
12        exit(1);
13    }
14    if((pid=fork())<0){
15        perror("Fork failed");
16        exit(2);
17    }
18    if(pid==0){
19        close(pfd[1]);
20        while((nread=read(pfd[0],buf,SIZE))!=0)
21            printf("Child read %s\n",buf);
22        close(pfd[0]);
23    }else{
24        close(pfd[0]);
25        strcpy(buf,"Hello...");
26        write(pfd[1],buf,strlen(buf)+1);
27        close(pfd[1]);
28    }
29}
```



```
indi@Harini:~$ ./a.out
Child read Hello...
```

B. Child Writing, Parent Reading

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4
5 #define READ 0
6 #define WRITE 1
7
8 char *phrase="Welcome to OS lab";
9
10 int main(){
11     int fd[2],bytesread;
12     char message[100];
13     pipe(fd);
14     if(fork()==0){
```

```
15     close(fd[READ]);
16     write(fd[WRITE],phrase,strlen(phrase)+1);
17     close(fd[WRITE]);
18 }else{
19     close(fd[WRITE]);
20     bytesread=read(fd[READ],message,100);
21     printf("Read %d bytes: %s\n",bytesread,message);
22     close(fd[READ]);
23 }
24}
```

A terminal window with a black background and green text. The prompt is 'indi@Harini:~\$'. The user has entered './a.out'. The output is 'Read 18 bytes: Welcome to OS lab'. The prompt is now 'indi@Harini:~\$' with a cursor at the end.

```
indi@Harini:~$ ./a.out
Read 18 bytes: Welcome to OS lab
indi@Harini:~$
```

Program 5: Write a program for FCFS (non-preemptive) scheduling algorithm.

```

1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int i, n;
6     float sumw = 0, sumt = 0;
7     int w[10], b[10], t[10], p[10];
8     float avgw, avgt;
9     printf("Enter the number of processes: ");
10    scanf("%d", &n);
11    for(i=0; i<n; i++){
12        printf("Enter the burst time for process %d: ", (i+1));
13        scanf("%d", &b[i]);
14        p[i] = i+1;
15    }
16    w[0]=0;
17    printf("\tProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");
18    for(i=0; i<n; i++){
19        t[i] = b[i] + w[i];
20        printf("\t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], w[i],
t[i]);
21        w[i+1] = w[i] + b[i];
22        sumw += w[i];
23        sumt += t[i];
24    }
25    avgw = sumw/n;
26    avgt = sumt/n;
27    printf("Average waiting time: %f\n", avgw);
28    printf("Average turnaround time: %f\n", avgt);
29}

```

Philosopher 3 is thinking.

```

● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab
\Osmania University\Classes\Sem 5\OS\Lab\> if ($?) { gcc fcfs.c -o fcfs }
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the burst time for process 2: 3
Enter the burst time for process 3: 3

```

| Process | Burst Time | Waiting Time | Turnaround Time |
|---------|------------|--------------|-----------------|
| 1 | 24 | 0 | 24 |
| 2 | 3 | 24 | 27 |
| 3 | 3 | 27 | 30 |

```

Average waiting time: 17.000000
Average turnaround time: 27.000000

```

Program 6: Write a program for FCFS (preemptive) scheduling algorithm.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main(){
4     int i, j, n, temp;
5     float sumw = 0, sumt = 0;
6     int w[10], b[10], t[10], p[10], a[10];
7     float avgw, avgt;
8     printf("Enter the number of processes: ");
9     scanf("%d", &n);
10    for(i=0; i<n; i++){
11        printf("Enter the burst time for process %d: ", (i+1));
12        scanf("%d", &b[i]);
13        printf("Enter the arrival time for process %d: ", (i+1));
14        scanf("%d", &a[i]);
15        p[i] = i+1;
16    }
17    for(i=0; i<n; i++){
18        for(j=i+1; j<n; j++){
19            if(a[i] > a[j]){
20                temp = b[i];
21                b[i] = b[j];
22                b[j] = temp;
23
24                temp = p[i];
25                p[i] = p[j];
26                p[j] = temp;
27
28                temp = a[i];
29                a[i] = a[j];
30                a[j] = temp;
31            }
32        }
33    }
34    w[0]=0;
35    printf("\tProcess \t Burst Time \t Arrival Time \t Waiting Time \t
Turnaround Time\n");
36    for(i=0; i<n; i++){
37        t[i] = b[i] + w[i]-a[i];
38        printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i],
a[i], w[i], t[i]);
39        w[i+1] = w[i] + b[i] - a[i];
40        sumw += w[i];
41        sumt += t[i];
42    }
43    avgw = sumw/n;
44    avgt = sumt/n;
45    printf("Average waiting time: %f\n", avgw);
46    printf("Average turnaround time: %f\n", avgt);
47}
```

```

Average turnaround time: 89500284.000000
● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\s
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc fcfsp.c -o fcfsp } ; if ($?) { .\
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the arrival time for process 1: 2
Enter the burst time for process 2: 3
Enter the arrival time for process 2: 0
Enter the burst time for process 3: 3
Enter the arrival time for process 3: 1
    Process      Burst Time    Arrival Time    Waiting Time    Turnaround Time
        2             3           0             0             3
        3             3           1             3             5
        1            24           2             5            27
Average waiting time: 2.666667
Average turnaround time: 11.666667

```

Program 7: Write a program for SJF (non-preemptive) algorithm.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main(){
4     int i, j, n, temp;
5     float sumw = 0, sumt = 0;
6     int w[10], b[10], t[10], p[10];
7     float avgw, avgt;
8     printf("Enter the number of processes: ");
9     scanf("%d", &n);
10    for(i=0; i<n; i++){
11        printf("Enter the burst time for process %d: ", (i+1));
12        scanf("%d", &b[i]);
13        p[i] = i+1;
14    }
15    for(i=0; i<n; i++){
16        for(j=i+1; j<n; j++){
17            if(b[i] > b[j]){
18                temp = b[i];
19                b[i] = b[j];
20                b[j] = temp;
21
22                temp = p[i];
23                p[i] = p[j];
24                p[j] = temp;
25            }
26        }
27    }
28    w[0]=0;
29    printf("\tProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");
30    for(i=0; i<n; i++){
31        t[i] = b[i] + w[i];
32        printf("\t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], w[i],
t[i]);
33        w[i+1] = w[i] + b[i];
34        sumw += w[i];
35        sumt += t[i];
36    }
37    avgw = sumw/n;
38    avgt = sumt/n;
39    printf("Average waiting time: %f\n", avgw);
40    printf("Average turnaround time: %f\n", avgt);
41}
```

```

● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab>
  \Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc sjfnp.c -o sjfnp }
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the burst time for process 2: 3
Enter the burst time for process 3: 3
      Process      Burst Time      Waiting Time      Turnaround Time
          2          3             0             3
          3          3             3             6
          1         24             6            30
Average waiting time: 3.000000
Average turnaround time: 13.000000

```

Program 8: Write a program for SJF (preemptive) algorithm.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int i, j, n;
6     float sumw = 0, sumt = 0;
7     int w[10], b[10], a[10], r[10], t[10];
8     float avgw, avgt;
9     printf("Enter the number of processes: ");
10    scanf("%d", &n);
11    int time, remain = n;
12    b[9] = 9999;
13    for(i=0; i<n; i++){
14        printf("Enter the burst time for process %d: ", (i+1));
15        scanf("%d", &b[i]);
16        printf("Enter the arrival time for process %d: ", (i+1));
17        scanf("%d", &a[i]);
18        r[i] = b[i];
19    }
20
21    printf("\tProcess \t Burst Time \t Arrival Time \t Waiting Time \t
Turnaround Time\n");
22    for(time = 0; remain != 0; time++){
23        int s = 9;
24        for(i=0; i<n; i++){
25            if(a[i] <= time && b[i] <= b[s] && r[i] > 0){
26                s=i;
27            }
28        }
29        r[s]--;
30        if(r[s] == 0){
31            remain--;
32            w[s] = time + 1 - a[s] - b[s];
33            t[s] = time + 1 - a[s];
34            printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", (s+1),
b[s], a[s], w[s], t[s]);
35            sumw += w[s];
36            sumt += t[s];
37        }
38    }
39    avgw = sumw/n;
40    avgt = sumt/n;
41    printf("Average waiting time: %f\n", avgw);
42    printf("Average turnaround time: %f\n", avgt);
43}
```



```

PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\sriha\
● \Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc sjfp.c -o sjfp } ; if ($?) { .\sjfp }
Enter the number of processes: 3
Enter the burst time for process 1: 24
Enter the arrival time for process 1: 0
Enter the burst time for process 2: 3
Enter the arrival time for process 2: 2
Enter the burst time for process 3: 3
Enter the arrival time for process 3: 1

```

| Process | Burst Time | Arrival Time | Waiting Time | Turnaround Time |
|---------|------------|--------------|--------------|-----------------|
| 3 | 3 | 1 | 0 | 3 |
| 2 | 3 | 2 | 2 | 5 |
| 1 | 24 | 0 | 6 | 30 |

```

Average waiting time: 2.666667
Average turnaround time: 12.666667

```

Program 9: Write a program for Priority (non-preemptive) algorithm.

```

1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int i, n, temp;
6     float sumw = 0, sumt = 0;
7     int w[10], b[10], t[10], p[10];
8     float avgw, avgt;
9     printf("Enter the number of processes: ");
10    scanf("%d", &n);
11    for(i=1; i<=n; i++){
12        printf("Enter the priority for process %d: ", i);
13        scanf("%d", &temp);
14        printf("Enter the burst time for process %d: ", i);
15        scanf("%d", &b[temp]);
16        p[temp] = i;
17    }
18    w[1]=0;
19    printf("\tProcess \t Burst Time \t Priority \t Waiting Time \t
Turnaround Time\n");
20    for(i=1; i<=n; i++){
21        t[i] = b[i] + w[i];
22        printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], i,
w[i], t[i]);
23        w[i+1] = w[i] + b[i];
24        sumw += w[i];
25        sumt += t[i];
26    }
27    avgw = sumw/n;
28    avgt = sumt/n;
29    printf("Average waiting time: %f\n", avgw);
30    printf("Average turnaround time: %f\n", avgt);
31}

```

```

PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc pnp.c -o pnp } ; if ($?) { .\pnp }
Enter the number of processes: 5
Enter the priority for process 1: 3
Enter the burst time for process 1: 10
Enter the priority for process 2: 1
Enter the burst time for process 2: 1
Enter the priority for process 3: 4
Enter the burst time for process 3: 2
Enter the priority for process 4: 5
Enter the burst time for process 4: 1
Enter the priority for process 5: 2
Enter the burst time for process 5: 5

```

| Process | Burst Time | Priority | Waiting Time | Turnaround Time |
|---------|------------|----------|--------------|-----------------|
| 2 | 1 | 1 | 0 | 1 |
| 5 | 5 | 2 | 1 | 6 |
| 1 | 10 | 3 | 6 | 16 |
| 3 | 2 | 4 | 16 | 18 |
| 4 | 1 | 5 | 18 | 19 |

```

Average waiting time: 8.200000
Average turnaround time: 12.000000

```

Program 10: Write a program for Priority (preemptive) algorithm.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int i, j, n;
6     float sumw = 0, sumt = 0;
7     int w[10], b[10], a[10], p[10], r[10], t[10];
8     float avgw, avgt;
9     printf("Enter the number of processes: ");
10    scanf("%d", &n);
11    int time, remain = n;
12    b[9] = 9999;
13    for(i=1; i<=n; i++){
14        printf("Enter the priority for process %d: ", i);
15        scanf("%d", &p[i]);
16        printf("Enter the burst time for process %d: ", i);
17        scanf("%d", &b[i]);
18        printf("Enter the arrival time for process %d: ", i);
19        scanf("%d", &a[i]);
20        r[i] = b[i];
21    }
22
23    printf("\tProcess \t Burst Time \t Arrival Time \t Priority \t Waiting
Time \t Turnaround Time\n");
24    for(time = 0; remain != 0; time++){
25        int s = 9;
26        for(i=1; i<=n; i++){
27            if(a[i] <= time && p[i] <= p[s] && r[i] > 0){
28                s=i;
29            }
30        }
31        r[s]--;
32        if(r[s] == 0){
33            remain--;
34            w[s] = time + 1 - a[s] - b[s];
35            t[s] = time + 1 - a[s];
36            printf("\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n",
(s+1), b[s], a[s], p[s], w[s], t[s]);
37            sumw += w[s];
38            sumt += t[s];
39        }
40    }
41    avgw = sumw/n;
42    avgt = sumt/n;
43    printf("Average waiting time: %f\n", avgw);
44    printf("Average turnaround time: %f\n", avgt);
45}
```

```

Average turnaround time: 12.800000
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc pp.c -o pp } ; if ($?) { .\pp }
Enter the number of processes: 5
Enter the priority for process 1: 3
Enter the burst time for process 1: 10
Enter the arrival time for process 1: 0
Enter the priority for process 2: 1
Enter the burst time for process 2: 1
Enter the arrival time for process 2: 1
Enter the priority for process 3: 4
Enter the burst time for process 3: 2
Enter the arrival time for process 3: 2
Enter the priority for process 4: 5
Enter the burst time for process 4: 1
Enter the arrival time for process 4: 3
Enter the priority for process 5: 2
Enter the burst time for process 5: 5
Enter the arrival time for process 5: 4

```

| Process | Burst Time | Arrival Time | Priority | Waiting Time | Turnaround Time |
|---------|------------|--------------|----------|--------------|-----------------|
| 3 | 1 | 1 | 1 | 0 | 1 |
| 6 | 5 | 4 | 2 | 0 | 5 |
| 2 | 10 | 0 | 3 | 6 | 16 |
| 4 | 2 | 2 | 4 | 14 | 16 |
| 5 | 1 | 3 | 5 | 15 | 16 |

```

Average waiting time: 7.000000
Average turnaround time: 10.800000

```

Program 11: Write a program for Round Robin scheduling algorithm.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int i, n, flag = 0, tq, time, remain;
6     float sumw = 0, sumt = 0;
7     int w[10], b[10], t[10], p[10], r[10];
8     float avgw, avgt;
9     printf("Enter the number of processes: ");
10    scanf("%d", &n);
11    remain = n;
12    printf("Enter time slice: ");
13    scanf("%d", &tq);
14    for(i=0; i<n; i++){
15        printf("Enter the burst time for process %d: ", (i+1));
16        scanf("%d", &b[i]);
17        r[i] = b[i];
18        p[i] = i+1;
19    }
20    printf("\tProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");
21    for(time=0, i= 0; remain !=0;){
22        if(r[i] <= tq && r[i] > 0){
23            time += r[i];
24            r[i] = 0;
25            flag = 1;
26        }else if(r[i] > 0){
27            r[i] -= tq;
28            time += tq;
29        }
30        if(r[i] == 0 && flag == 1){
31            remain --;
32            w[i] = time - b[i];
33            t[i] = time;
34            printf("\t\t %d \t\t %d \t\t %d \t\t %d\n", p[i], b[i], w[i],
35            t[i]);
36            sumw += w[i];
37            sumt += t[i];
38            flag = 0;
39        }
40        if(i == n-1){
41            i=0;
42        }else{
43            i++;
44        }
45        avgw = sumw/n;
46        avgt = sumt/n;
47        printf("Average waiting time: %f\n", avgw);
48        printf("Average turnaround time: %f\n", avgt);
49    }
```

```

cd "c:\Users\
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc rr.c -o rr } ; if ($?) { .\rr }
Enter the number of processes: 3
Enter time slice: 1
Enter the burst time for process 1: 24
Enter the burst time for process 2: 3
Enter the burst time for process 3: 3

```

| Process | Burst Time | Waiting Time | Turnaround Time |
|---------|------------|--------------|-----------------|
| 2 | 3 | 5 | 8 |
| 3 | 3 | 6 | 9 |
| 1 | 24 | 6 | 30 |

```

Average waiting time: 5.666667
Average turnaround time: 15.666667

```

Program 12: Write a program to demonstrate:

A. First Fit

```
1 #include<stdio.h>
2
3 int main(){
4     int i, j, n, flag;
5     printf("Enter the number of files: ");
6     scanf("%d", &n);
7     int files[100], holes[100];
8     printf("Enter the size of each file.\n");
9     for(i=0; i<n; i++){
10         scanf("%d", &files[i]);
11     }
12     printf("Enter the size of each hole.\n");
13     for(i=0; i<n; i++){
14         scanf("%d", &holes[i]);
15     }
16     printf("File No. \t File Size \t Hole Size \t Memory Wasted \n");
17     for(i=0; i<n; i++){
18         flag = 0;
19         for(j=0; j<n; j++){
20             if(files[i] <= holes[j]){
21                 flag = 1;
22                 printf("%d \t\t %d \t\t %d \t\t %d \n", i+1, files[i],
23 holes[j], holes[j] - files[i]);
24                 holes[j] = 0;
25                 break;
26             }
27             if(flag == 0){
28                 printf("%d \t\t %d \t\t Unable to allocate memory.\n", i+1,
29 files[i]);
30             }
31 }
```

```
Number of page faults is: 15
● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc firstfit.c -o firstfit }
Enter the number of files: 5
Enter the size of each file.
6 29 2 256 1
Enter the size of each hole.
500 250 6 7 199
File No.      File Size      Hole Size      Memory Wasted
1             6             500           494
2             29            250           221
3             2             6             4
4             256          Unable to allocate memory.
5             1             7             6
```

B. Best Fit

```
1 #include<stdio.h>
2
3 int main(){
4     int i, j, n, temp, flag;
5     printf("Enter the number of files: ");
6     scanf("%d", &n);
7     int files[100], holes[100];
8     printf("Enter the size of each file.\n");
9     for(i=0; i<n; i++){
10         scanf("%d", &files[i]);
11     }
12     printf("Enter the size of each hole.\n");
13     for(i=0; i<n; i++){
14         scanf("%d", &holes[i]);
15     }
16     for(i=0; i<n; i++){
17         for(j=i+1; j<n; j++){
18             if(holes[i] > holes[j]){
19                 temp = holes[j];
20                 holes[j] = holes[i];
21                 holes[i] = temp;
22             }
23         }
24     }
25     printf("File No. \t File Size \t Hole Size \t Memory Wasted \n");
26     for(i=0; i<n; i++){
27         flag = 0;
28         for(j=0; j<n; j++){
29             if(files[i] <= holes[j]){
30                 flag = 1;
31                 printf("%d \t\t %d \t\t %d \t\t %d \n", i+1, files[i],
32 holes[j], holes[j] - files[i]);
33                 holes[j] = 0;
34                 break;
35             }
36             if(flag == 0){
37                 printf("%d \t\t %d \t\t Unable to allocate memory.\n", i+1,
38 files[i]);
39             }
40 }
```



```

● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab>
ty\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc bestfit.c -o bestfit } ; if ($?) {
Enter the number of files: 5
Enter the size of each file.
6 29 2 256 1
Enter the size of each hole.
500 250 6 7 199
File No.      File Size      Hole Size      Memory Wasted
1             6             6             0
2             29          199          170
3             2             7             5
4            256          500          244
5             1          250          249

```

C. Worst fit

```
1 #include<stdio.h>
2
3 int main(){
4     int i, j, n, temp, flag;
5     printf("Enter the number of files: ");
6     scanf("%d", &n);
7     int files[100], holes[100];
8     printf("Enter the size of each file.\n");
9     for(i=0; i<n; i++){
10         scanf("%d", &files[i]);
11     }
12     printf("Enter the size of each hole.\n");
13     for(i=0; i<n; i++){
14         scanf("%d", &holes[i]);
15     }
16     for(i=0; i<n; i++){
17         for(j=i+1; j<n; j++){
18             if(holes[i] < holes[j]){
19                 temp = holes[j];
20                 holes[j] = holes[i];
21                 holes[i] = temp;
22             }
23         }
24     }
25     printf("File No. \t File Size \t Hole Size \t Memory Wasted \n");
26     for(i=0; i<n; i++){
27         flag = 0;
28         for(j=0; j<n; j++){
29             if(files[i] <= holes[j]){
30                 flag = 1;
31                 printf("%d \t\t %d \t\t %d \t\t %d \n", i+1, files[i],
holes[j], holes[j] - files[i]);
32                 holes[j] = 0;
33                 break;
34             }
35         }
36         if(flag == 0){
37             printf("%d \t\t %d \t\t Unable to allocate memory.\n", i+1,
files[i]);
38         }
39     }
```

```
HELLO WORLD was written to datafile.dat
PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "c:\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc worstfit.c -o worstfit }
Enter the number of files: 5
Enter the size of each file.
6 29 2 256 1
Enter the size of each hole.
500 250 6 7 199
File No.      File Size      Hole Size      Memory Wasted
1             6             500           494
2             29           250           221
3             2           199           197
4            256      Unable to allocate memory.
5             1             7             6
```

Program 13: Write a program for FIFO page replacement algorithm.

```
1 #include<stdio.h>
2
3 int main(){
4     int i, j, n, m, point = 0, flag, fault = 0;
5     printf("Enter the number of pages:");
6     scanf("%d", &n);
7     int pages[100];
8     printf("Enter the reference string: ");
9     for (i=0; i<n; i++){
10         scanf("%d", &pages[i]);
11     }
12     printf("Enter the number of frames.");
13     scanf("%d", &m);
14     int frames[100], max[100];
15     for(j=0; j<m; j++){
16         frames[j] = -1;
17         max[j] = 9999;
18     }
19     for(i=0; i<n; i++){
20         flag = 0;
21         for(j=0; j<m; j++){
22             if(frames[j] == pages[i]){
23                 flag = 1;
24             }
25         }
26         if(flag == 0){
27             frames[point] = pages[i];
28             if(point == m-1){
29                 point = 0;
30             }else{
31                 point++;
32             }
33         }
34         if(flag == 0){
35             fault++;
36             printf("Page Fault: ");
37             for(j=0; j<m; j++){
38                 printf("%d ", frames[j]);
39             }
40             printf("\n");
41         }
42     }
43     printf("Number of page faults is: %d ", fault);
44 }
```

```
Average Turnaround Time: 27.000000
● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc fifo.c -o fifo }
Enter the number of pages:20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames:3
Page Fault: 7 -1 -1
Page Fault: 7 0 -1
Page Fault: 7 0 1
Page Fault: 2 0 1
Page Fault: 2 3 1
Page Fault: 2 3 0
Page Fault: 4 3 0
Page Fault: 4 2 0
Page Fault: 4 2 3
Page Fault: 0 2 3
Page Fault: 0 1 3
Page Fault: 0 1 2
Page Fault: 7 1 2
Page Fault: 7 0 2
Page Fault: 7 0 1
Number of page faults is: 15
```

Program 14: Write a program for Optimal page replacement algorithm.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int i, j, k, n, m, repl, flag, point, fault = 0;
6     printf("Enter the number of pages:");
7     scanf("%d", &n);
8     int pages[100];
9     printf("Enter the reference string: ");
10    for (i=0; i<n; i++){
11        scanf("%d", &pages[i]);
12    }
13    printf("Enter the number of frames.");
14    scanf("%d", &m);
15    int frames[100], max[100];
16    for(j=0; j<m; j++){
17        frames[j] = -1;
18        max[j] = 9999;
19    }
20    for(i=0; i<n; i++){
21        flag = 0;
22        for(j=0; j<m; j++){
23            if(frames[j] == pages[i]){
24                flag = 1;
25            }
26        }
27        if(flag == 0 && i >= m){
28            for(j=0; j<m; j++){
29                point = 0;
30                for(k = i+1; k<n; k++){
31                    if(pages[k] != frames[j]){
32                        point++;
33                    }else{
34                        max[j] = point;
35                        break;
36                    }
37                }
38            }
39            repl = 0;
40            for(j=0; j<m; j++){
41                if(max[repl] < max[j]){
42                    repl = j;
43                }
44            }
45            frames[repl] = pages[i];
46        }
47        if(flag == 0){
48            if(i < m){
49                frames[i] = pages[i];
50            }
51            fault++;
52            printf("Page Fault: ");
53            for(j=0; j<m; j++){
54                max[j] = 9999;
```

```

55         printf("%d ", frames[j]);
56     }
57     printf("\n");
58 }
59 }
60 printf("Number of page faults is: %d ", fault);
61}

```

```

● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab> cd "
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc optimal.c -o optimal }
Enter the number of pages:20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames.3
Page Fault: 7 -1 -1
Page Fault: 7 0 -1
Page Fault: 7 0 1
Page Fault: 2 0 1
Page Fault: 2 0 3
Page Fault: 2 4 3
Page Fault: 2 0 3
Page Fault: 2 0 1
Page Fault: 7 0 1
Number of page faults is: 9

```

Program 15: Write a program for LRU page replacement algorithm.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5     int i, j, k, n, m, repl, flag, point, fault = 0;
6     printf("Enter the number of pages:");
7     scanf("%d", &n);
8     int pages[100];
9     printf("Enter the reference string: ");
10    for (i=0; i<n; i++){
11        scanf("%d", &pages[i]);
12    }
13    printf("Enter the number of frames.");
14    scanf("%d", &m);
15    int frames[100], max[100];
16    for(j=0; j<m; j++){
17        frames[j] = -1;
18        max[j] = 9999;
19    }
20    for(i=0; i<n; i++){
21        flag = 0;
22        for(j=0; j<m; j++){
23            if(frames[j] == pages[i]){
24                flag = 1;
25            }
26        }
27        if(flag == 0 && i >= m){
28            for(j=0; j<m; j++){
29                point = 0;
30                for(k = i-1; k>=0; k--){
31                    if(pages[k] != frames[j]){
32                        point++;
33                    }else{
34                        max[j] = point;
35                        break;
36                    }
37                }
38            }
39            repl = 0;
40            for(j=0; j<m; j++){
41                if(max[j] > max[repl]){
42                    repl = j;
43                }
44            }
45            frames[repl] = pages[i];
46        }
47        if(flag == 0){
48            if(i < m){
49                frames[i] = pages[i];
50            }
51            fault++;
52            printf("Page Fault: ");
53            for(j=0; j<m; j++){
54                max[j] = 9999;
```



```

55         printf("%d ", frames[j]);
56     }
57     printf("\n");
58 }
59 }
60 printf("Number of page faults is: %d ", fault);
61}

```

```

Collect2.exe: error: ld returned 1 exit status
● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab\> gcc lru.c -o lru
Enter the number of pages:20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames:3
Page Fault: 7 -1 -1
Page Fault: 7 0 -1
Page Fault: 7 0 1
Page Fault: 2 0 1
Page Fault: 2 0 3
Page Fault: 4 0 3
Page Fault: 4 0 2
Page Fault: 4 3 2
Page Fault: 0 3 2
Page Fault: 1 3 2
Page Fault: 1 0 2
Page Fault: 1 0 7
Number of page faults is: 12
● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS\Lab\>

```

Program 16: Write a program to demonstrate producer-consumer problem.

```
1 #include<stdio.h>
2 #include<pthread.h>
3 #include<sys/types.h>
4 #include<unistd.h>
5 #include<stdlib.h>
6 #include<semaphore.h>
7
8 sem_t empty, full, mutex;
9 char buf[10];
10
11 void *producer(void* arg){
12     int i;
13     for(i=0; i<10; i++){
14         sem_wait(&empty);
15         sem_wait(&mutex);
16         buf[i] = i;
17         printf("Item produced is %d\n", buf[i]);
18         sem_post(&mutex);
19         sem_post(&full);
20         sleep(1);
21     }
22     pthread_exit("Producer\n");
23 }
24
25 void *consumer(void* arg){
26     int j;
27     for(j=0; j<10; j++){
28         sem_wait(&full);
29         sem_wait(&mutex);
30         j = buf[j];
31         printf("Item consumed is %d\n", buf[j]);
32         sem_post(&mutex);
33         sem_post(&empty);
34         sleep(5);
35     }
36     pthread_exit("Consumer\n");
37 }
38
39 int main(){
40     pthread_t pid1, pid2;
41     sem_init(&empty, 0, 10);
42     sem_init(&full, 0, 0);
43     sem_init(&mutex, 1, 1);
44     void *status;
45     pthread_create(&pid1, NULL, producer, NULL);
46     pthread_create(&pid2, NULL, consumer, NULL);
47     pthread_join(pid1, &status);
48     printf("The exited status of 1st is %s\n", (char*)status);
49     pthread_join(pid2, &status);
50     printf("The exited status %s\n", (char*)status);
51     return 0;
52 }
```

```
● PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS
  \Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc pc.c -o pc }
Item produced is 0
Item consumed is 0
Item produced is 1
Item produced is 2
Item produced is 3
Item produced is 4
Item consumed is 1
Item produced is 5
Item produced is 6
Item produced is 7
Item produced is 8
Item produced is 9
Item consumed is 2
The exited status of 1st is Producer

Item consumed is 3
Item consumed is 4
Item consumed is 5
Item consumed is 6
Item consumed is 7
Item consumed is 8
Item consumed is 9
The exited status Consumer
```

Program 17: Write a program to demonstrate readers-writers problem.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<pthread.h>
4 #include<semaphore.h>
5
6 int data=0, rdcnt=0;
7
8 sem_t mutex, writeblock;
9
10 void * reader(void * no){
11     printf("\n\tReader %d is executing ", (int)no);
12     sem_wait(&mutex);
13     printf("\n\tWait to mutex by %d reader", (int)no);
14     rdcnt++;
15     if(rdcnt==1){
16         sem_wait(&writeblock);
17         printf("\n\tWait to writerblock by %d reader", (int)no);
18     }
19     printf("\n\t***Reader %d read data = %d ", (int)no, data);
20     if(rdcnt==1){
21         sem_post(&writeblock);
22         printf("\n\tSignal to writerblock by %d reader", (int)no);
23     }
24     sem_post(&mutex);
25     printf("\n\tSignal to mutex by %d reader\n", (int)no);
26 }
27
28 void * writer(void * no){
29     printf("\n\tWriter %d is executing ", (int)no);
30     sem_wait(&writeblock);
31     printf("\n\tWait to writerblock by %d writer", (int)no);
32     data+=5;
33     printf("\n\t***Writer %d write data = %d ", (int)no, data);
34     sem_post(&writeblock);
35     printf("\n\tSignal to writer by %d writer\n", (int)no);
36 }
37
38 int main(){
39     int no, i, ir=0, iw=0, ch;
40     sem_init(&mutex, 0, 1);
41     sem_init(&writeblock, 0, 1);
42     printf("\nEnter no of readers and writers to create: ");
43     scanf("%d", &no);
44     pthread_t r[no], w[no];
45     do{
46         printf("\n\t1. Reader\n\t2. Writer\n\t3. Terminate\n\tYour choice:
47 ");
48         scanf("%d", &ch);
49         switch(ch){
50             case 1: pthread_create(&r[ir], NULL, reader, (void *)ir);
51                     pthread_join(r[ir++], NULL);
52                     break;
53             case 2: pthread_create(&w[iw], NULL, writer, (void *)iw);
54                     pthread_join(w[iw++], NULL);
```

```
54             break;
55         }
56     }while(ch!=3);
57
58     sem_destroy(&mutex);
59     sem_destroy(&writeblock);
60 }
```

Enter no of readers and writers to create: 2

1. Reader
2. Writer
3. Terminate
Your choice: 1

Reader 0 is executing
Wait to mutex by 0 reader
Wait to writerblock by 0 reader
***Reader 0 read data = 0
Signal to writerblock by 0 reader
Signal to mutex by 0 reader

1. Reader
2. Writer
3. Terminate
Your choice: 2

Writer 0 is executing
Wait to writerblock by 0 writer
***Writer 0 write data = 5
Signal to writer by 0 writer

1. Reader
2. Writer
3. Terminate
Your choice: 3

Program 18: Write a program to demonstrate dining-philosophers problem.

```
1 #include<stdio.h>
2 #include<pthread.h>
3 #include<semaphore.h>
4
5 #define N 5
6 #define THINKING 0
7 #define HUNGRY 1
8 #define EATING 2
9 #define LEFT (ph_num+4)%N
10#define RIGHT (ph_num+1)%N
11
12sem_t mutex;
13sem_t S[N];
14
15void * philosopher(void *num);
16void take_fork(int);
17void put_fork(int);
18void test(int);
19
20int state[N];
21int phil_num[N] = {0, 1, 2, 3, 4};
22
23int main(){
24    int i;
25    pthread_t thread_id[N];
26    sem_init(&mutex, 0, 1);
27    for(i=0; i<N; i++){
28        pthread_create(&thread_id[i], NULL, philosopher, &phil_num[i]);
29        printf("Philosopher %d is thinking.\n", i+1);
30    }
31    for(i=0; i<N; i++){
32        pthread_join(thread_id[i], NULL);
33    }
34}
35
36void *philosopher(void *num){
37    while(1){
38        int *i = num;
39        sleep(1);
40        take_fork(*i);
41        sleep(0);
42        put_fork(*i);
43    }
44}
45
46void take_fork(int ph_num){
47    sem_wait(&mutex);
48    state[ph_num] = HUNGRY;
49    printf("Philosopher %d is hungry.\n", ph_num+1);
50    test(ph_num);
51    sem_post(&mutex);
52    sem_wait(&S[ph_num]);
53    sleep(1);
```

```
54}
55
56void test(int ph_num){
57    if(state[ph_num] == HUNGRY && state[LEFT] != EATING && state[RIGHT] !=
EATING){
58        state[ph_num] = EATING;
59        sleep(2);
60        printf("Philosopher %d takes chopsticks %d and %d.\n", ph_num+1,
LEFT+1, ph_num+1);
61        printf("Philosopher %d is eating.\n", ph_num+1);
62        sem_post(&S[ph_num]);
63    }
64}
65
66void put_fork(int ph_num){
67    sem_wait(&mutex);
68    state[ph_num] = THINKING;
69    printf("Philosopher %d puts chopstick %d and %d down.\n", ph_num+1,
LEFT+1, ph_num+1);
70    printf("Philosopher %d is thinking.\n", ph_num+1);
71    test(LEFT);
72    test(RIGHT);
73    sem_post(&mutex);
74}
```

```

PS C:\Users\sriha\OneDrive\Documents\Osmania University\Classes\Sem 5\OS
\Osmania University\Classes\Sem 5\OS\Lab\" ; if ($?) { gcc dp.c -o dp }
dp.c: In function 'philosopher':
dp.c:39:9: warning: implicit declaration of function 'sleep' [-Wimplicit
 39 |         sleep(1);
    |         ^~~~~~
Philosopher 1 is thinking.
Philosopher 2 is thinking.
Philosopher 3 is thinking.
Philosopher 4 is thinking.
Philosopher 5 is thinking.
Philosopher 1 is hungry.
Philosopher 1 takes chopsticks 5 and 1.
Philosopher 1 is eating.
Philosopher 4 is hungry.
Philosopher 4 takes chopsticks 3 and 4.
Philosopher 4 is eating.
Philosopher 2 is hungry.
Philosopher 5 is hungry.
Philosopher 3 is hungry.
Philosopher 1 puts chopstick 5 and 1 down.
Philosopher 1 is thinking.
Philosopher 2 takes chopsticks 1 and 2.
Philosopher 2 is eating.
Philosopher 2 puts chopstick 1 and 2 down.
Philosopher 2 is thinking.
Philosopher 1 takes chopsticks 5 and 1.
Philosopher 1 is eating.
Philosopher 4 puts chopstick 3 and 4 down.
Philosopher 4 is thinking.
Philosopher 3 takes chopsticks 2 and 3.
Philosopher 3 is eating.
Philosopher 5 puts chopstick 4 and 5 down.
Philosopher 5 is thinking.

```


Program 19: Write a program to demonstrate file reading and writing.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #define BUFFER_SIZE 1024
5 typedef struct {
6     FILE *input_file;
7     FILE *output_file;
8 } ThreadData;
9 void *copyFile(void *arg) {
10     ThreadData *data = (ThreadData *)arg;
11     char buffer[BUFFER_SIZE];
12     size_t bytesRead;
13     while ((bytesRead = fread(buffer, 1, BUFFER_SIZE, data->input_file)) >
14 0) {
15         fwrite(buffer, 1, bytesRead, data->output_file);
16     }
17     fclose(data->input_file);
18     fclose(data->output_file);
19     return NULL;
20 }
21 int main() {
22     pthread_t thread;
23     ThreadData data;
24     data.input_file = fopen("input.txt", "rb");
25     if (data.input_file == NULL) {
26         perror("Error opening input file");
27         exit(EXIT_FAILURE);
28     }
29     data.output_file = fopen("output.txt", "wb");
30     if (data.output_file == NULL) {
31         perror("Error opening output file");
32         fclose(data.input_file);
33         exit(EXIT_FAILURE);
34     }
35     if (pthread_create(&thread, NULL, copyFile, &data) != 0) {
36         perror("Error creating thread");
37         fclose(data.input_file);
38         fclose(data.output_file);
39         exit(EXIT_FAILURE);
40     }
41     if (pthread_join(thread, NULL) != 0) {
42         perror("Error joining thread");
43         fclose(data.input_file);
44         fclose(data.output_file);
45         exit(EXIT_FAILURE);
46     }
47     printf("File copied successfully.\n");
48     return 0;
49 }
```

```
indi@Harini:~$ ./a.out
File copied successfully.
```

SHELL SCRIPT:

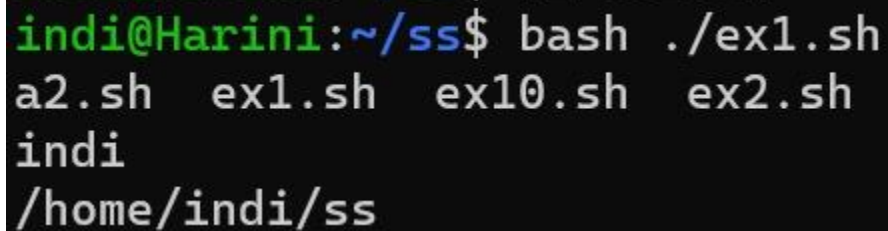
1. Write a shell script to show basic Linux commands.

SS1

ls

who

pwd

A terminal window with a black background and green text. The prompt is 'indi@Harini:~/ss\$'. The user has entered 'bash ./ex1.sh'. The output shows 'a2.sh', 'ex1.sh', 'ex10.sh', and 'ex2.sh' on the next line, followed by 'indi' and '/home/indi/ss' on the following lines.

```
indi@Harini:~/ss$ bash ./ex1.sh
a2.sh  ex1.sh  ex10.sh  ex2.sh
indi
/home/indi/ss
```

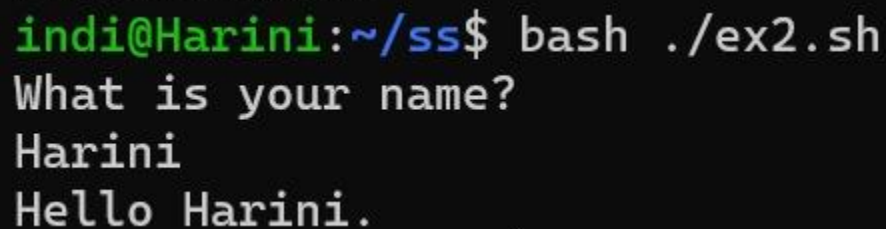
2. Write an interactive shell script that takes one input.

SS2

echo What is your name?

read name

echo Hello \$name.

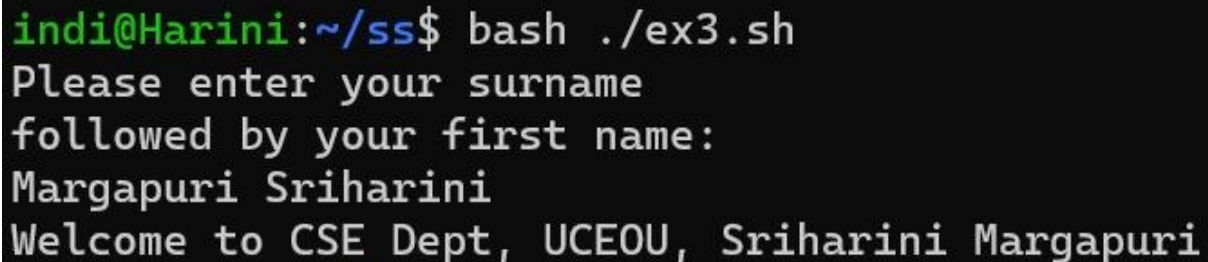
A terminal window with a black background and green text. The prompt is 'indi@Harini:~/ss\$'. The user has entered 'bash ./ex2.sh'. The output shows 'What is your name?' on the next line, followed by 'Harini' and 'Hello Harini.' on the following lines.

```
indi@Harini:~/ss$ bash ./ex2.sh
What is your name?
Harini
Hello Harini.
```

3. Write an interactive shell script that takes multiple user inputs.

SS3

```
echo "Please enter your surname"
echo "followed by your first name: "
read name1 name2
echo "Welcome to CSE Dept, UCEOU, $name2 $name1"
```

A terminal window showing the execution of the SS3 script. The prompt is 'indi@Harini:~/ss\$'. The user enters 'bash ./ex3.sh'. The script prompts for a surname, then for a first name. The user enters 'Margapuri Sriharini'. The script then outputs 'Welcome to CSE Dept, UCEOU, Sriharini Margapuri'.

4. Write a shell script that takes to file names as input and copies first file into the second file.

SS4

```
echo "Please Enter source file name :"
```

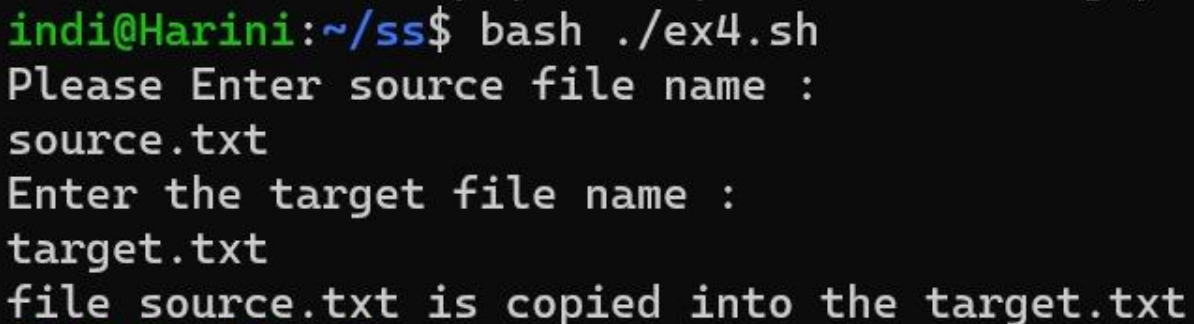
read source

```
echo "Enter the target file name :"
```

read target

```
cp $source $target
```

```
echo file $source is copied into the $target
```

A terminal window showing the execution of the SS4 script. The prompt is 'indi@Harini:~/ss\$'. The user enters 'bash ./ex4.sh'. The script prompts for a source file name, then for a target file name. The user enters 'source.txt' and 'target.txt'. The script then outputs 'file source.txt is copied into the target.txt'.

5. Write shell script which will accept 5 numbers as parameters and display their sum. Also display the contents of the different variables in the script.

SS5

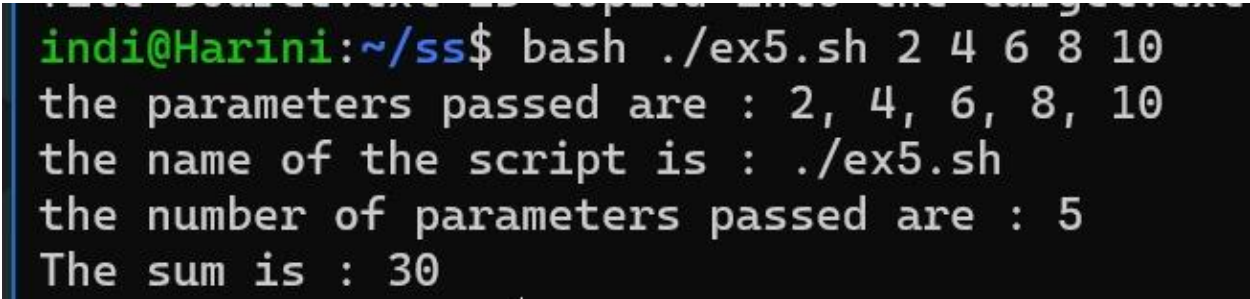
echo the parameters passed are : \$1, \$2, \$3, \$4, \$5

echo the name of the script is : \$0

echo the number of parameters passed are : \$#

sum=`expr \$1 + \$2 + \$3 + \$4 + \$5` echo

The sum is : \$sum



```
indi@Harini:~/ss$ bash ./ex5.sh 2 4 6 8 10
the parameters passed are : 2, 4, 6, 8, 10
the name of the script is : ./ex5.sh
the number of parameters passed are : 5
The sum is : 30
```

6. Write a shell script which will accept different numbers and finds their sum. The number of parameters can vary.

#SS6

sum=0

while [\$# -gt 0]

do

sum=`expr \$sum + \$1`

shift

done

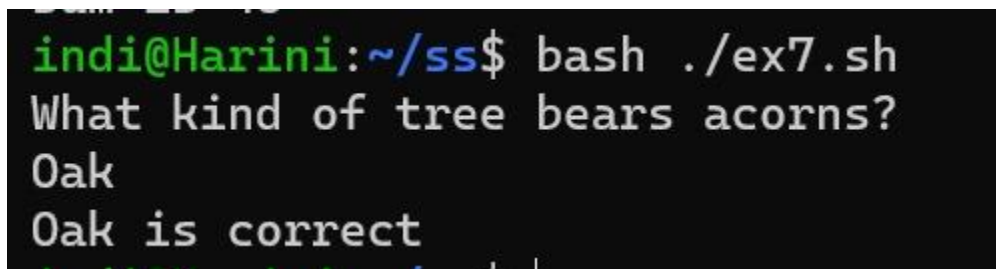
echo sum is \$sum



```
indi@Harini:~/ss$ bash ./ex6.sh 1 2 3 4 5 6 7 8 9 0
sum is 45
```

7. Write a shell script that takes user's answers and returns whether it is correct using case.

```
#SS7
echo What kind of tree bears acorns?
read response
case $response in
[Oo][Aa][Kk]) echo $response is correct ;;
*) echo Sorry, response is wrong
esac
```



```
indi@Harini:~/ss$ bash ./ex7.sh
What kind of tree bears acorns?
Oak
Oak is correct
```

8. Write a shell script that takes user's answers and keeps asking for inputs until user gets the answer right using while.

```
#SS8
echo What is the Capital of Saudi Arabia? R
read answer
while test $answer != Riyadh
do echo No, Wrong please try again.
read answer
done
echo This is correct.
```

```
What is the Capital of Saudi Arabia?  
Dubai  
No, Wrong please try again.  
Riyadh  
This is correct  
indi@Harini:~/ss$ |
```

9. Write a shell script that asks for user login name using until.

```
#SS9  
echo "Please Enter the user login name: "  
read login_name  
until who | grep $login_name  
do  
sleep 30  
done  
echo The user $login_name has logged in
```

```
Please Enter the user login name:  
indi  
indi  
The user indi has logged in  
indi@Harini:~/ss$ |
```

10. Write a shell script that takes 3 numbers from the user and determines the largest of those 3 numbers using if.

```
#SS10
echo "Enter the first number : "
read num1
echo "Enter the second number : "
read num2
echo "Enter the third number : "
read num3
if test $num1 -gt $num2
then
    if test $num1 -gt $num3
    then
        echo $num1 is the largest
    else
        echo $num3 is the largest
    fi
else
    if test $num2 -gt $num3
    then
        echo $num2 is largest
    else
        echo $num3 is the largest
    fi
fi
```

```
Enter the first number :  
2  
Enter the second number :  
5  
Enter the third number :  
8  
8 is the largest  
indi@Harini:~/ss$ |
```

11. Write a shell script to demonstrate arithmetic operators.

```
echo Enter the first number:  
read num1  
echo Enter the second number:  
read num2  
  
sum=expr $num1 + $num2  
dif=expr $num1 - $num2  
pro=expr $num1 \* $num2  
quo=expr $num1 / $num2  
  
echo $num1 + $num2 = $sum  
echo $num1 - $num2 = $dif  
echo $num1 \* $num2 = $pro  
echo $num1 / $num2 = $quo
```



```
indi@Harini:~/ss$ ./a1.sh
Enter the first number:
10
Enter the second number:
5
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
```

12. Write a shell script to demonstrate do-while.

```
echo Enter a number:
read num
i=1
while [ $i -le 10 ]
do
    pro=expr $i \* $num
    echo $i x $num = $pro
    i=expr $i + 1
done
```

```
indi@Harini:~/ss$ bash ./a2.sh
Enter a number:
3
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
4 x 3 = 12
5 x 3 = 15
6 x 3 = 18
7 x 3 = 21
8 x 3 = 24
9 x 3 = 27
10 x 3 = 30
```

13. Write a shell script to demonstrate if-else.

```
echo Enter a number:
read num
rem=expr $num % 2
if [ $rem -eq 0 ]
then
    echo $num is even.
else
    echo $num is odd.
fi
```

```

indi@Harini:~/ss$ bash ./a3.sh
Enter a number:
2
2 is even.
indi@Harini:~/ss$ bash ./a3.sh
Enter a number:
3
3 is odd.

```

14. Write a shell script to demonstrate case.

```

echo Enter the first number:
read num1
echo Enter the second number:
read num2

echo 1. Addition
echo 2. Difference
echo 3. Multiplication
echo 4. Division

echo "Enter an operation (1-4)"
read op

case $op in
1) sum=expr $num1 + $num2
    echo $num1 + $num2 = $sum;;
2) dif=expr $num1 - $num2
    echo $num1 - $num2 = $dif;;
3) pro=expr $num1 \* $num2
    echo $num1 \* $num2 = $pro;;
4) quo=expr $num1 / $num2
    echo $num1 / $num2 = $quo;;
*) echo Terminating...;;
esac

```

```

indi@Harini:~/ss$ bash ./a4.sh
Enter the first number:
10
Enter the second number:
5
1. Addition
2. Difference
3. Multiplication
4. Division
Enter an operation (1-4)
1
10 + 5 = 15

```

15. Write a shell script to demonstrate logical operators.

```

read -p 'Enter a : ' a
read -p 'Enter b : ' b

if(( $a == "true" && $b == "true" ))
then
    echo Both are true.
else
    echo Both are not true.
fi

if(( $a == "true" || $b == "true" ))
then
    echo Atleast one of them is true.
else
    echo None of them is true.
fi

if(( ! $a == "true" ))
then
    echo "a" was initially false.
else
    echo "a" was initially true.
fi

```

```
indi@Harini:~/ss$ bash ./a5.sh
Enter a : true
Enter b : false
Both are true.
Atleast one of them is true.
a was initially true.
```