

# Operating System Lab Observation

**1a**

**Program:**

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

int main()
{
    int fd;
    char buffer[50];

    /* open() – create and open a file */
    fd = open("sample.txt", O_CREAT | O_RDWR, 0644);

    if (fd < 0)
    {
        printf("File opening failed\n");
        return 1;
    }

    /* write() – write data into file */
    write(fd, "Hello OS Lab", 12);

    /* move file pointer to beginning */
    lseek(fd, 0, SEEK_SET);
```

```
/* read() – read data from file */

read(fd, buffer, 12);

/* display data */

printf("Data read from file: %s\n", buffer);

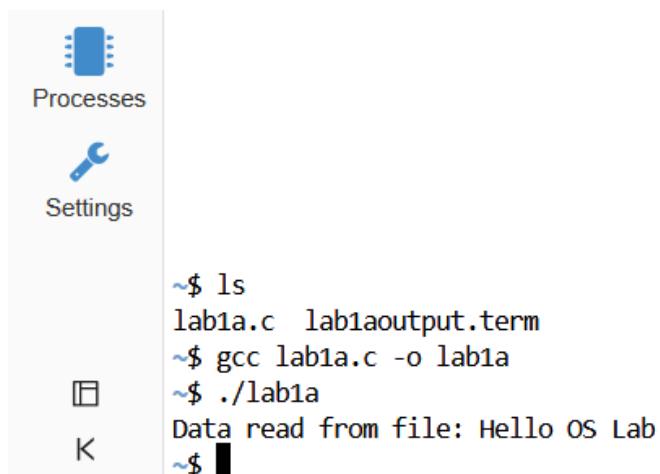
/* close() – close the file */

close(fd);

return 0;

}
```

#### Output:



```
$ ls
lab1a.c  lab1aoutput.term
$ gcc lab1a.c -o lab1a
$ ./lab1a
Data read from file: Hello OS Lab
$
```

**1b:**

**Program:**

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

int main()
{
    DIR *d;
    struct dirent *entry;

    /* open current directory */
    d = opendir(".");
    if (d == NULL)
    {
        printf("Directory cannot be opened\n");
        return 1;
    }

    /* read directory entries */
    printf("Directory contents:\n");
    while ((entry = readdir(d)) != NULL)
    {
        printf("%s\n", entry->d_name);
    }

    /* close directory */
    closedir(d);
    return 0;
}
```

## Output:

```
Hello ⌂ Buy a license or redeem a voucher.  
Prices start at a few $ per month. Apply your license to this project. ⓘ Free projects...  
lab1a.x lab1ac.x lab1b.x lab1bc.x lab1c.x lab1cc.x lab2.c.x lab2ou.x lab3.c.x  
Assistant Server ~ Terminal File Edit View Go Help  
$ gcc lab1b.c -o lab1b  
$ ./Lab1b  
Directory contents:  
.  
..  
lab1aoutput.term  
lab1b  
.ssh  
lab1a  
.snapshots  
lab1b.c  
lab1boutput.term  
.bash_profile  
.bashrc  
sample.txt  
lab1a.c  
.cache  
.smc  
$
```

1c:

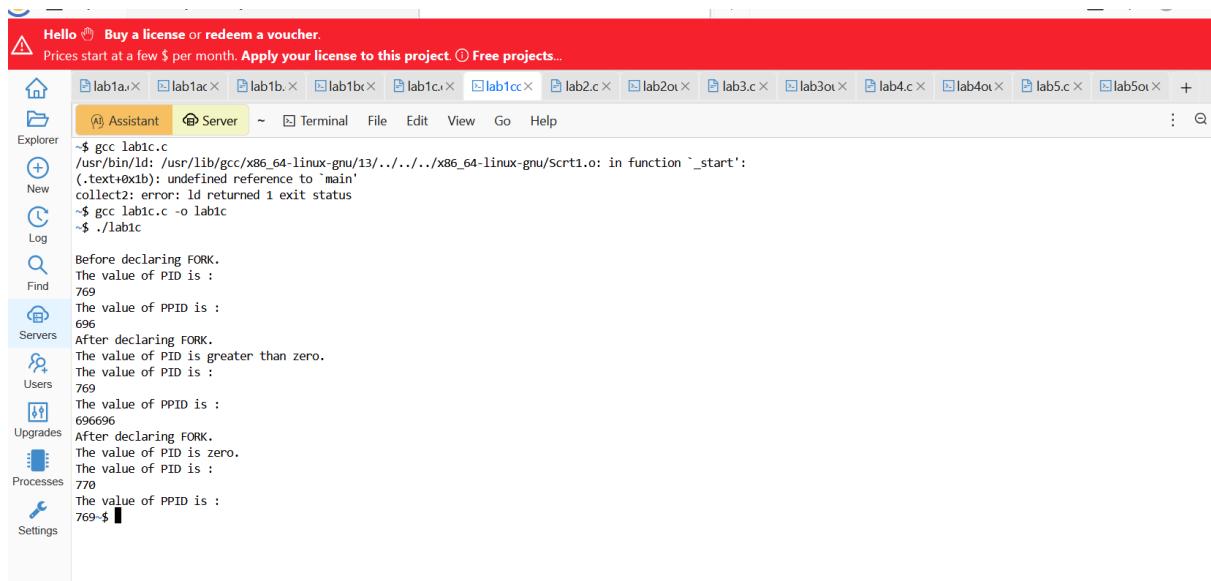
## Program:

```
#include<stdio.h>  
  
#include<unistd.h>  
  
#include<sys/types.h>  
  
void main()  
{  
    printf("\nBefore declaring FORK.");  
    printf("\nThe value of PID is : ");  
    printf("\n%d",getpid());  
    printf("\nThe value of PPID is : ");  
    printf("\n%d",getppid());
```

```
pid_t pid=fork();//Used to create child process  
if(pid==0)//Child Process  
{  
printf("\nAfter declaring FORK.");  
printf("\nThe value of PID is zero.");  
printf("\nThe value of PID is : ");  
printf("\n%d",getpid());  
printf("\nThe value of PPID is : ");  
printf("\n%d",getppid());  
}  
  
else if(pid>0)//Parent Process  
{  
printf("\nAfter declaring FORK.");  
printf("\nThe value of PID is greater than zero.");  
printf("\nThe value of PID is : ");  
printf("\n%d",getpid());  
printf("\nThe value of PPID is : ");  
printf("\n%d",getppid());  
}  
  
else//Error  
{  
printf("\nAfter declaring FORK.");  
printf("\nThe value of PID is less than zero.");  
printf("\nThe value of PID is : ");  
printf("\n%d",getpid());  
printf("\nThe value of PPID is : ");  
printf("%d",getppid());  
}
```

}

## Output:



```
Hello ⚡ Buy a license or redeem a voucher.
Prices start at a few $ per month. Apply your license to this project. Free projects...
$ gcc lab1c.c
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/13/../../../../x86_64-linux-gnu/Scrt1.o: in function `__start':
(.text+0xb): undefined reference to `main'
collect2: error: ld returned 1 exit status
$ gcc lab1c.c -o lab1c
$ ./lab1c

Before declaring FORK.
The value of PID is :
769
The value of PPID is :
696
After declaring FORK.
The value of PID is greater than zero.
The value of PID is :
769
The value of PPID is :
696696
After declaring FORK.
The value of PID is zero.
The value of PID is :
770
The value of PPID is :
769-$
```

## Lab2:

### Program:

```
#include <stdio.h>

// FCFS Scheduling

void fcfs(int n, int bt[]) {

int wt[10], tat[10];

wt[0] = 0;

for(int i = 1; i < n; i++)

wt[i] = wt[i-1] + bt[i-1];

printf("\nFCFS Scheduling:\n");

printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

for(int i = 0; i < n; i++) {
```

```

tat[i] = wt[i] + bt[i];

printf("P%d\t%d\t%d\t%d\t%d\n", i+1, bt[i], wt[i], tat[i]);

}

}

// Round Robin Scheduling

void roundRobin(int n, int bt[], int tq) {

int rem_bt[10], wt[10], tat[10];

int t = 0;

for(int i = 0; i < n; i++) {

rem_bt[i] = bt[i];

wt[i] = 0;

}

while(1){

int done = 1;

for(int i = 0; i < n; i++) {

if(rem_bt[i] > 0) {

done = 0;

if(rem_bt[i] > tq) {

t += tq;

rem_bt[i] -= tq;

} else {

t += rem_bt[i];

wt[i] = t - bt[i];

rem_bt[i] = 0;

}

}

}

if(done == 1)

```

```
        break;
    }

printf("\nRound Robin Scheduling (Time Quantum = %d):\n", tq);
printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

for(int i = 0; i < n; i++) {
    tat[i] = bt[i] + wt[i];
    printf("P%d\t%d\t%d\t%d\n", i+1, bt[i], wt[i], tat[i]);
}

int main() {
    int n, tq;
    int bt[10];

    printf("Enter number of processes: ");
    scanf("%d", &n);

    printf("Enter burst times:\n");
    for(int i = 0; i < n; i++) {
        printf("P%d: ", i+1);
        scanf("%d", &bt[i]);
    }

    fcfs(n, bt);

    printf("\nEnter Time Quantum for Round Robin: ");
    scanf("%d", &tq);

    roundRobin(n, bt, tq);

    return 0;
}
```

}

## Output:

The screenshot shows a terminal window with the following session:

```
Hello ☀ Buy a license or redeem a voucher.  
⚠ Prices start at a few $ per month. Apply your license to this project. ⓘ Free projects...  
Assistant Server Terminal File Edit View Go Help  
New Log Find Servers Users Upgrades Processes Settings  
$ gcc lab2.c o- lab  
gcc: error: -E or -x required when input is from standard input  
$ gcc lab2.c -o lab2  
$ ./lab2  
bash: ./: Is a directory  
$ gcc lab2.c -o lab2  
$ ./lab2  
Enter number of processes: 3  
Enter burst times:  
P1: 1  
P2: 2  
P3: 5  
FCFS Scheduling:  
Process Burst Time Waiting Time Turnaround Time  
P1 1 0 1  
P2 2 1 3  
P3 5 3 8  
Enter Time Quantum for Round Robin: 6  
Round Robin Scheduling (Time Quantum = 6):  
Process Burst Time Waiting Time Turnaround Time  
P1 1 0 1  
P2 2 1 3  
P3 5 3 8  
$ 9  
bash: 9: command not found  
$
```

## Lab3:

### Program:

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

```
int main() {  
    int n, i, j;  
    int bt[10], pr[10], p[10];  
    int wt[10], tat[10], temp;  
  
    printf("Enter number of processes: ");  
    scanf("%d", &n);
```

```

for(i = 0; i < n; i++) {
    p[i] = i + 1;
    printf("\n");
}

Enter Burst Time and Priority for P%d: ", p[i]);
scanf("%d %d", &bt[i], &pr[i]);
}

// Sort by Priority
for(i = 0; i < n; i++) {
    for(j = i + 1; j < n; j++) {
        if(pr[i] > pr[j]) {
            temp = pr[i]; pr[i] = pr[j]; pr[j] = temp;
            temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
            temp = p[i]; p[i] = p[j]; p[j] = temp;
        }
    }
}

wt[0] = 0;

for(i = 1; i < n; i++)
    wt[i] = wt[i - 1] + bt[i - 1];
for(i = 0; i < n; i++)
    tat[i] = wt[i] + bt[i];
printf("\nProcess\tBT\tPriority\tWT\tTAT\n");
for(i = 0; i < n; i++)
    printf("P%d\t%d\t%d\t%d\t%d\t%d\n", p[i], bt[i], pr[i], wt[i], tat[i]);

```

```
return 0;
```

```
}
```

## Output:

The screenshot shows a web-based terminal interface on a browser window. The address bar indicates the project ID: cocalc.com/projects/322f7eb4-00b2-4c59-a231-31b615d591ce/files/OS%20LAB/exno4.term. The terminal window title is "Server". The terminal content shows the execution of a C program named "priority.c". The program uses the FCFSRR scheduling algorithm to print processes' burst times and priorities. The output includes the command "gcc priority.c -o priority", the execution of the binary "priority", and the resulting table of processes (P1-P4) with their burst times (BT), priorities, waiting time (WT), and turnaround time (TAT).

```
priority.c exno4.term
Hello Buy a license or redeem a voucher.
Prices start at a few $ per month. Apply your license to this project. Free projects...
~/OS LAB$ gcc priority.c -o priority
~/OS LAB$ ./priority
Enter number of processes: 4
Enter Burst Time and Priority for P1: 2 1
Enter Burst Time and Priority for P2: 3 2
Enter Burst Time and Priority for P3: 4 3
Enter Burst Time and Priority for P4: 4 4
Process BT Priority WT TAT
P1 2 1 0 2
P2 3 2 2 5
P3 4 3 5 9
P4 4 4 9 13
~/OS LAB$
```

## Lab4:

### Program:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define N 5
sem_t forks[N];
pthread_t philosophers[N];
void* philosopher(void* num){
    int id = *(int*)num;
    printf("Philosopher %d is thinking\n", id);
```

```

sleep(1);

sem_wait(&forks[id]);

sem_wait(&forks[(id + 1) % N]);

printf("Philosopher %d is eating\n", id);

sleep(1);

sem_post(&forks[id]);

sem_post(&forks[(id + 1) % N]);

printf("Philosopher %d finished eating\n", id);

return NULL;

}

int main() {

    int i, id[N];

    for (i = 0; i < N; i++)
        sem_init(&forks[i], 0, 1);

    for (i = 0; i < N; i++) {
        id[i] = i;
        pthread_create(&philosophers[i], NULL, philosopher, &id[i]);
    }

    for (i = 0; i < N; i++)
        pthread_join(philosophers[i], NULL);
}

return 0;
}

```

**Output:**

The screenshot shows a software interface with a terminal window at the top displaying compilation errors:

```
cc1: fatal error: dining.c: No such file or directory
compilation terminated.
$ ./a.out
bash: ./a.out: No such file or directory
$ gcc lab4.c -lpthread
$ ./a.out
```

Below the terminal is a log window showing the execution of a philosopher program:

```
Philosopher 0 is thinking
Philosopher 3 is thinking
Philosopher 1 is thinking
Philosopher 4 is thinking
Philosopher 2 is thinking
Philosopher 0 is eating
Philosopher 3 is eating
Philosopher 0 finished eating
Philosopher 3 finished eating
Philosopher 2 is eating
Philosopher 4 is eating
Philosopher 2 finished eating
Philosopher 4 finished eating
Philosopher 1 is eating
Philosopher 1 finished eating
```

## Lab5: Program:

```
#include<stdio.h>

void firstFit(int blockSize[], int m, int processSize[], int n)
{
    int allocation[n];
    for(int i=0;i<n;i++)
        allocation[i]=-1;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            if(blockSize[j] >= processSize[i])
            {
                allocation[i]=j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }
}
```

```
    }
}
}

prin ("\nFIRST FIT\nProcess No.\tProcess Size\tBlock No.\n");
for(int i=0;i<n;i++)
{
    if(alloca on[i]!=-1)
        prin ("%d\t%d\t%d\n",i+1,processSize[i],alloca on[i]+1);
    else
        prin ("%d\t%d\tNot Allocated\n",i+1,processSize[i]);
}
}
```

```
void bestFit(int blockSize[], int m, int processSize[], int n)
```

```
{
    int alloca on[n];
    for(int i=0;i<n;i++)
        alloca on[i]=-1;

    for(int i=0;i<n;i++)
    {
        int bestIdx=-1;
        for(int j=0;j<m;j++)
        {
            if(blockSize[j]>=processSize[i])
            {
                if(bestIdx==-1 || blockSize[j]<blockSize[bestIdx])

```

```
    bestIdx=j;  
}  
}  
  
if(bestIdx!=-1)  
{  
    alloca on[i]=bestIdx;  
    blockSize[bestIdx]-=processSize[i];  
}  
}  
  
prin ("\nBEST FIT\nProcess No.\tProcess Size\tBlock No.\n");  
for(int i=0;i<n;i++)  
{  
    if(alloca on[i]!=-1)  
        prin ("%d\t%d\t%d\n",i+1,processSize[i],alloca on[i]+1);  
    else  
        prin ("%d\t%d\tNot Allocated\n",i+1,processSize[i]);  
}  
}  
  
int main()  
{  
    int blockSize[] = {100, 500, 200, 300, 600};  
    int processSize[] = {212, 417, 112, 426};  
  
    int m = 5;  
    int n = 4;
```

```

int block1[5], block2[5];

for(int i=0;i<m;i++)

{

    block1[i]=blockSize[i];

    block2[i]=blockSize[i];

}

firstFit(block1,m,processSize,n);

bestFit(block2,m,processSize,n);

return 0;
}

```

### **Output:**

```

Hello Buy a license or redeem a voucher
Prices start at a few $ per month. Apply your license to this project. Free projects...
EX NO 5.term
priority.c
memoryalloc.c
EX NO 5.term
priority.c
sample.txt
syscalls
~/OS LAB$ ls
1b.c 1c.c 2026-01-20-file-1.term 2026-01-20-file-3.term 2026-01-27-file-2.term 2FCFSRR.C 'EX NO 5.term' exno4.term memoryalloc.c priority.c syscalls
~/OS LAB$ pwd
/home/user/OS LAB
~/OS LAB$ gcc memoryalloc.c -o memoryalloc
~/OS LAB$ ./memoryalloc

FIRST FIT
Find Process No. Process Size Block No.
Servers
1 212 2
2 417 5
3 112 2
4 426 Not Allocated

Users
Upgrades
Processes
~/OS LAB$ 


```