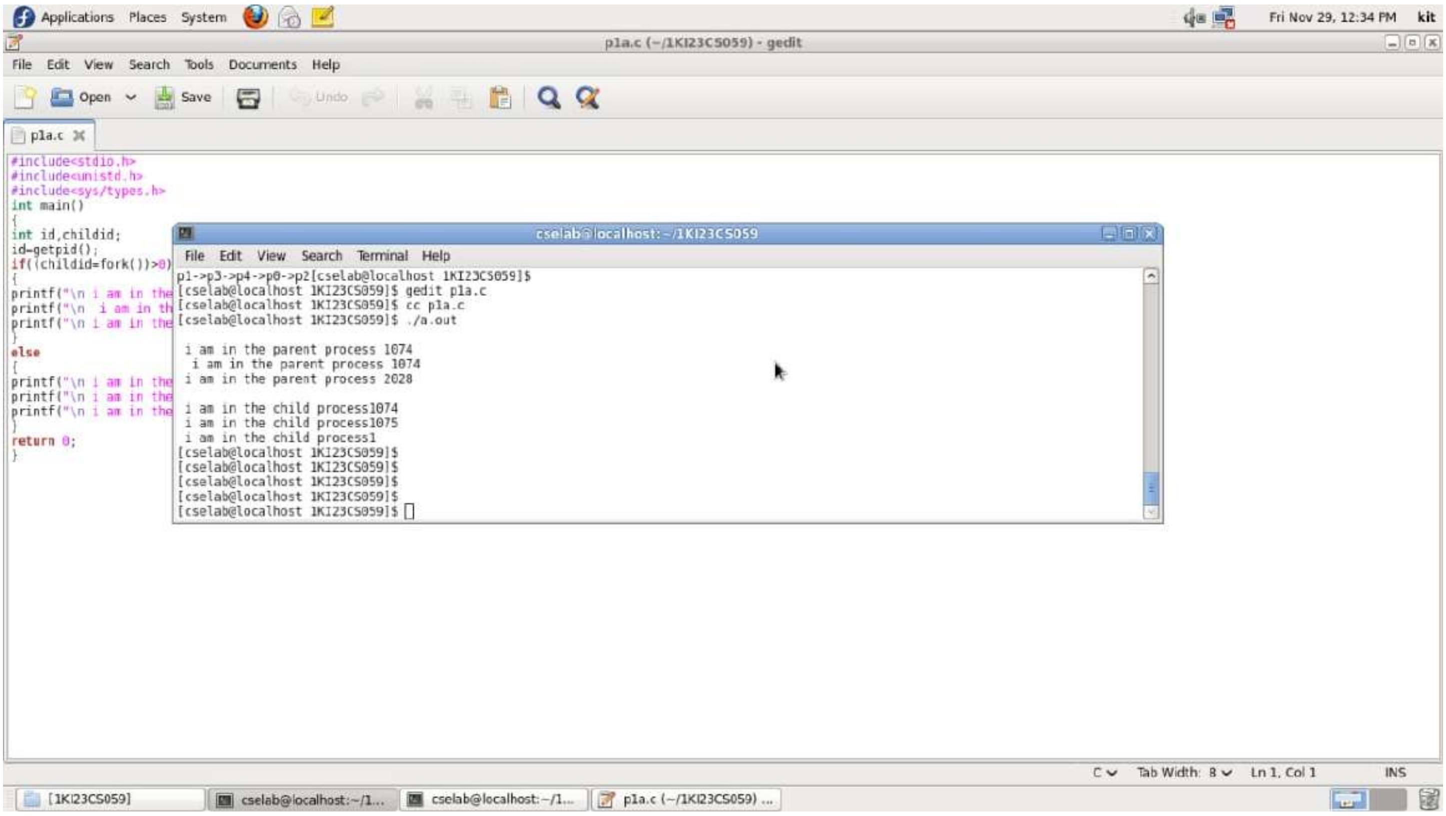
1. Develop a C program to implement the process system calls (fork(), exec(), wait(), create process, terminate process)

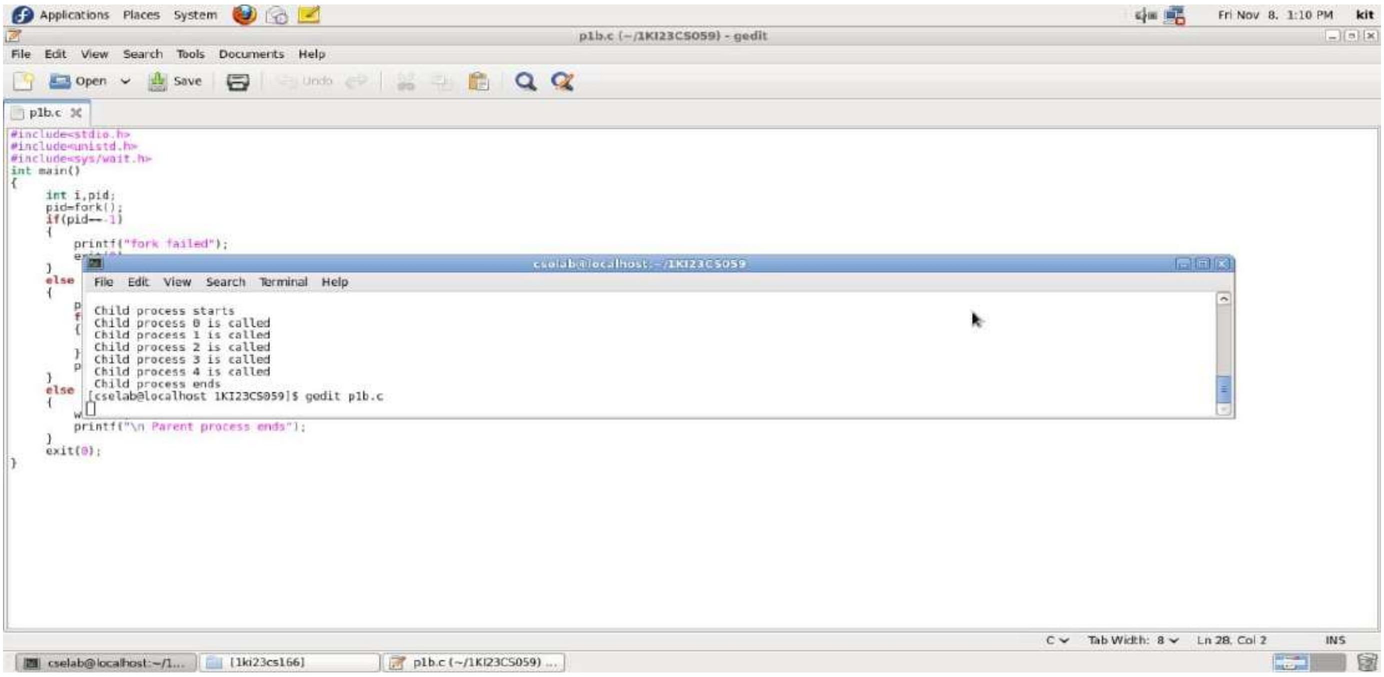
fork() system call

```
#include<stdio.h>
#include <unistd.h>
#include<sys/types.h>
int main()
{
      int id, childid;
      id=getpid();
      if((childid=fork())>0)
      {
            printf("\n I am in the parent process %d",id);
            printf("\n I am in the parent process %d",getpid());
            printf("\n I am in the parent process %d\n",getppid());
      }
      else
      {
            printf("\n I am in child process %d",id);
            printf("\n I am in the child process %d",getpid());
            printf("\n I am in the child process %d",getppid());
      }
      return 0;

}
```



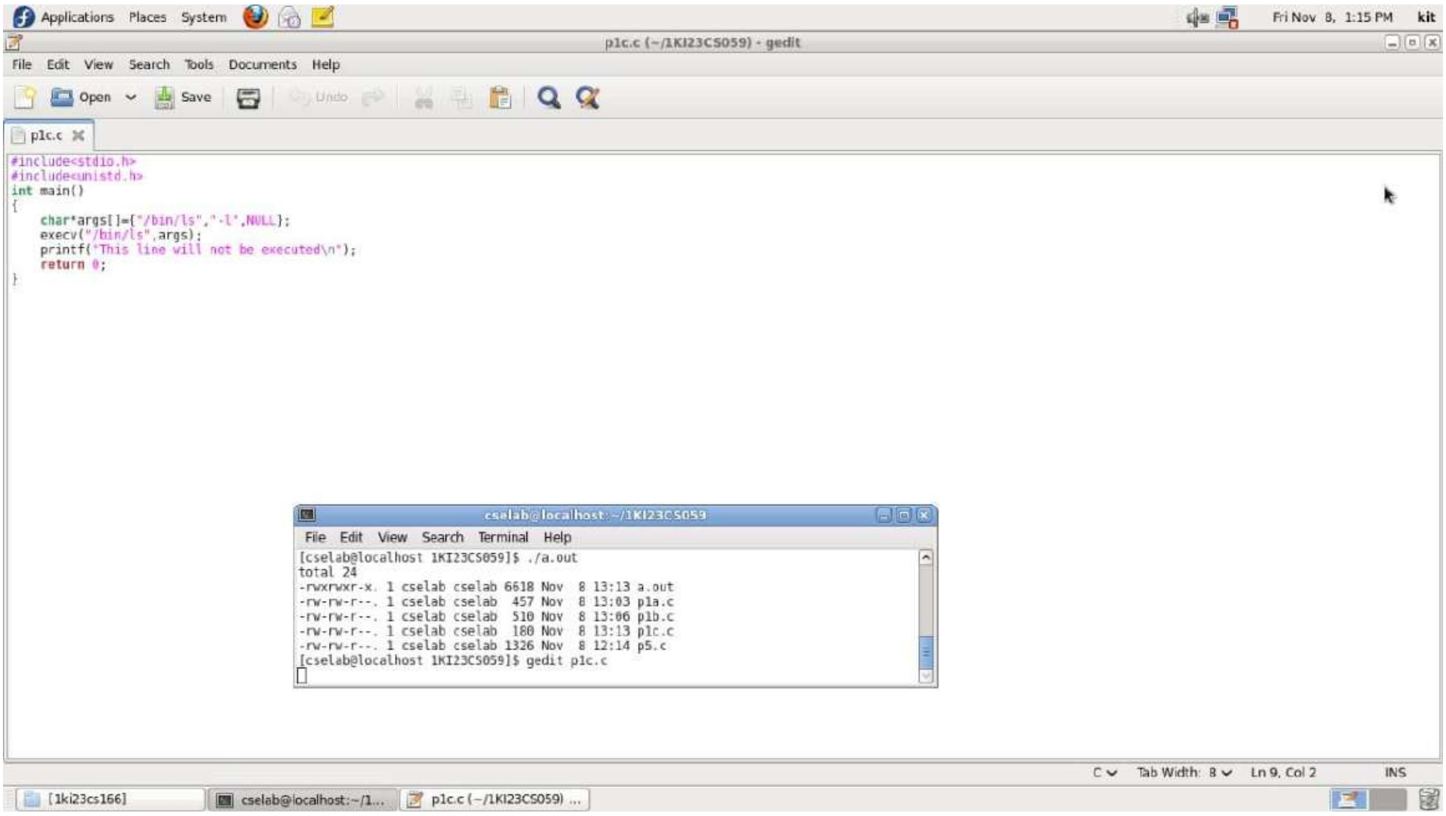
wait() system call

```
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
int main( )
{
      int i, pid;
      pid=fork( );
      if(pid== -1)
      {
            printf("fork failed");
            exit(0);
      }
      else if(pid==0)
      {
            printf("\n Child process starts");
            for(i=0; i<5; i++)
            {
                  printf("\n Child process %d is called", i);
            }
            printf("\n Child process ends");
      }
      else
      {
            wait(0);
            printf("\n Parent process ends");
      }
      exit(0);
 }
```



exec() system call

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

int main() {
    char *args[] = {"/bin/ls", "-l", NULL};
    execv("/bin/ls", args);
    printf("This line will not be executed\n");
    return 0;
}
```

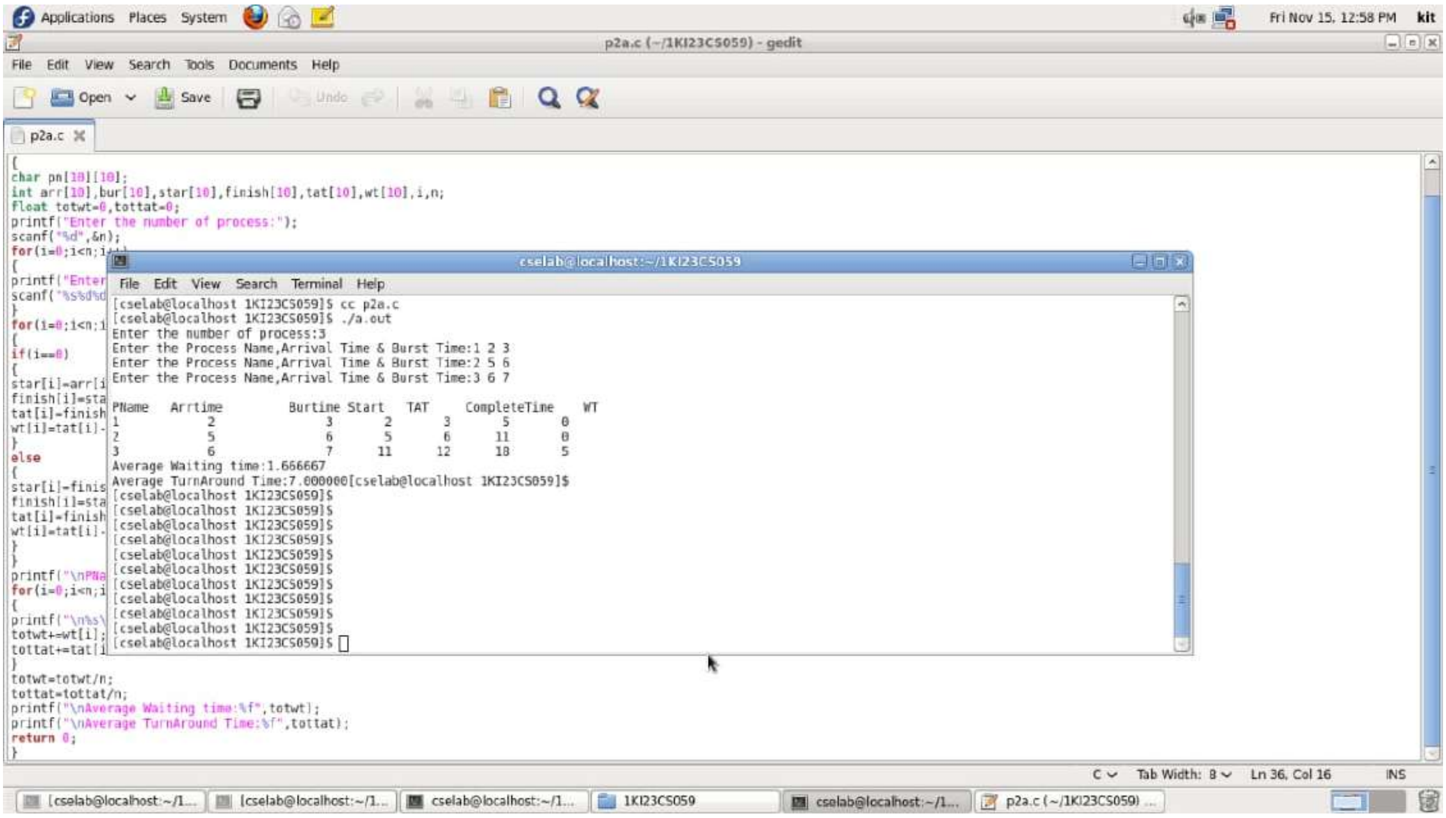


2. Simulate the following CPU scheduling algorithms to find turnaround time and waiting time a) FCFS
b) SJF c) Round Robin d) Priority.

//First Come First Serve (FCFS) Scheduling Algorithm

```
#include<stdio.h>
int main()
{
	char pn[10][10];
	int arr[10],bur[10],star[10],finish[10],tat[10],wt[10],i,n;
	float totwt=0,tottat=0;
	printf("Enter the number of processes:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("Enter the Process Name, Arrival Time & Burst Time:");
		scanf("%s%d%d",&pn[i],&arr[i],&bur[i]);
	}
	for(i=0;i<n;i++)
	{
		if(i==0)
		{
			star[i]=arr[i];
			finish[i]=star[i]+bur[i];
			tat[i]=finish[i]-arr[i];
			wt[i]=tat[i]-bur[i];
		}
		else
		{
			star[i]=finish[i-1];
			finish[i]=star[i]+bur[i];
			tat[i]=finish[i]-arr[i];
			wt[i]=tat[i]-bur[i];
		}
	}
```

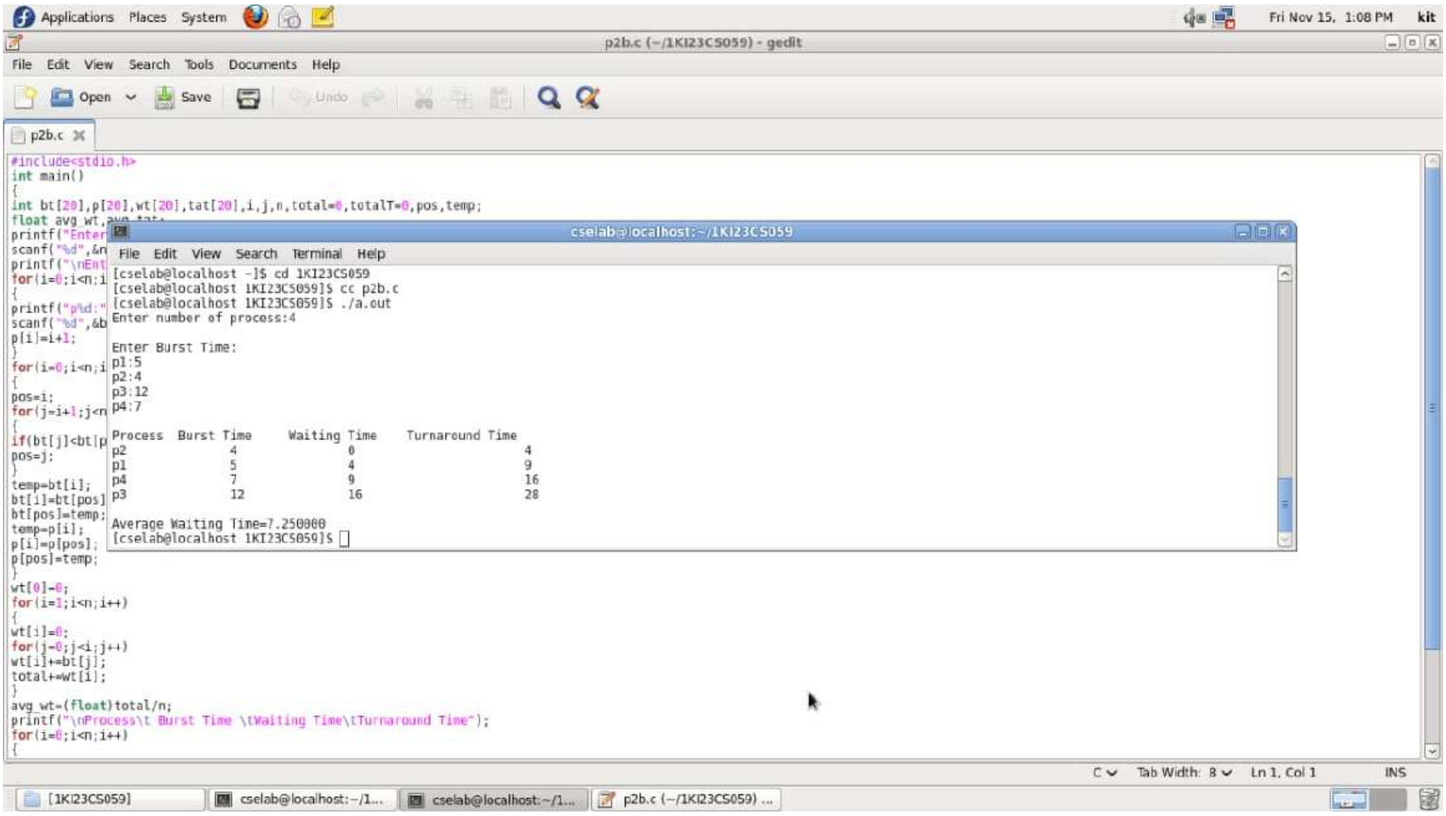
```c
	printf("\nPName\tArrtime \tBurtime\tStart
\tTAT\tCompleteTime\tWT");
	for(i=0;i<n;i++)
	{
		printf("\n%s\t%6d\t\t%6d\t%6d\t%6d\t%6d\t\t%6d",pn[i],arr[i]
	, bur[i],star[i],tat[i],finish[i],wt[i]);
		totwt+=wt[i];
		tottat+=tat[i];
	}
	totwt=totwt/n;
	tottat=tottat/n
	;
	printf("\nAverage Waiting time:%f",totwt);
	printf("\nAverage Turn Around Time:%f",tottat);
}
```



// Shortest Job First (SJF) Scheduling Algorithm

```
#include<stdio.h>
int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,totalT=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time:\n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }

    //sorting of burst times
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[j]<bt[pos])
                pos=j;
        }

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;

    //finding the waiting time of all the processes
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
             //individual WT by adding BT of all previous completed
processes
            wt[i]+=bt[j];

        //total waiting time
        total+=wt[i];
    }
```

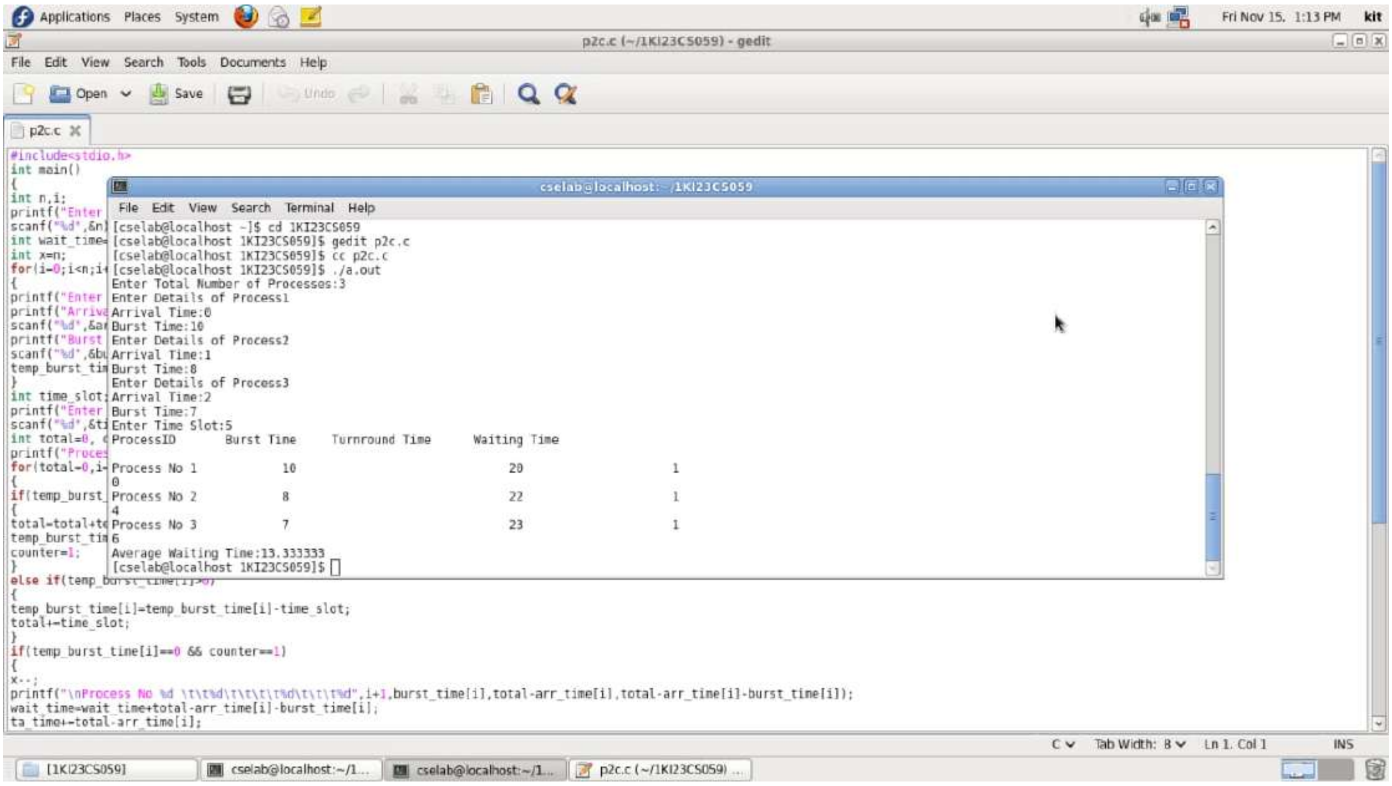
```
    //average waiting time
    avg_wt=(float)total/n;

    printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        //turnaround time of individual processes
        tat[i]=bt[i]+wt[i];

        //total turnaround time
        totalT+=tat[i];
        printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    //average turnaround time
    avg_tat=(float)totalT/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f",avg_tat);
}
```



// Round Robin Scheduling algorithm

```
 #include<stdio.h>
 int main()
{
    //Input no of processed
    int  n;
    printf("Enter Total Number of Processes:");
    scanf("%d", &n);
    int wait_time = 0, ta_time = 0, arr_time[n], burst_time[n],
temp_burst_time[n];
    int x = n;

    //Input details of processes
    for(int i = 0; i < n; i++)
    {
        printf("Enter Details of Process %d \n", i + 1);
        printf("Arrival Time:  ");
        scanf("%d", &arr_time[i]);
        printf("Burst Time:   ");
        scanf("%d", &burst_time[i]);
        temp_burst_time[i] = burst_time[i];
    }

    //Input time slot
    int time_slot;
    printf("Enter Time Slot:");
    scanf("%d", &time_slot);

    //Total indicates total time
    //counter indicates which process is executed
    int total = 0,  counter = 0,i;
    printf("Process ID       Burst Time       Turnaround Time
Waiting Time\n");
    for(total=0, i = 0; x!=0; )
    {
        // define the conditions
        if(temp_burst_time[i] <= time_slot && temp_burst_time[i] > 0)
        {
            total = total + temp_burst_time[i];
            temp_burst_time[i] = 0;
            counter=1;
        }
        else if(temp_burst_time[i] > 0)
        {
            temp_burst_time[i] = temp_burst_time[i] - time_slot;
            total  += time_slot;
        }
        if(temp_burst_time[i]==0 && counter==1)
        {
            x--; //decrement the process no.
```

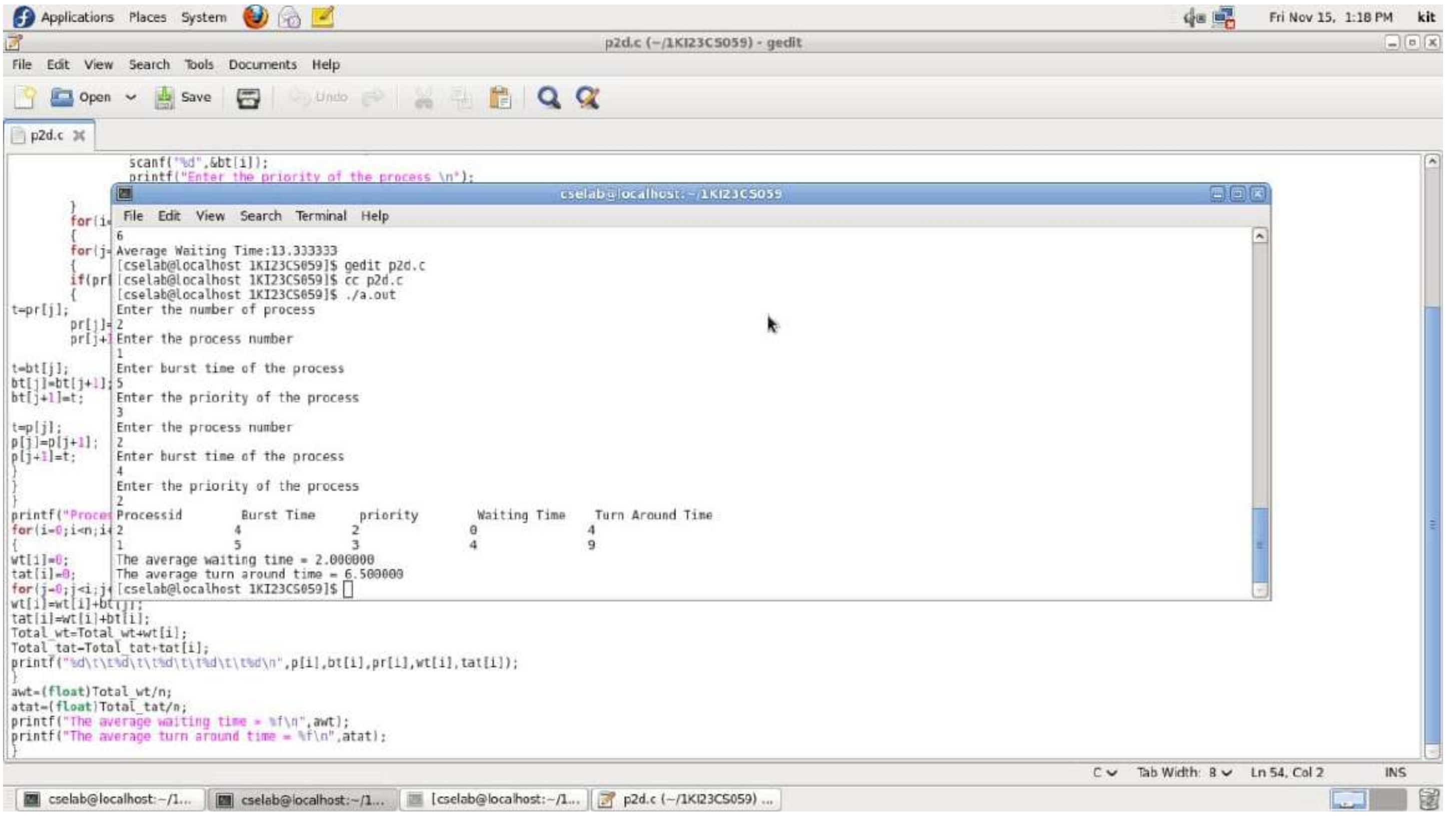
```
            printf("\nProcess No %d  \t\t %d\t\t\t\t %d\t\t\t %d", i+1,
burst_time[i],
                     total-arr_time[i], total-arr_time[i]-burst_time[i]);
            wait_time = wait_time+total-arr_time[i]-burst_time[i];
            ta_time += total -arr_time[i];
            counter =0;
        }
        if(i==n-1)
        {
            i=0;
        }
        else if(arr_time[i+1]<=total)
        {
            i++;
        }
        else
        {
            i=0;
        }
    }
    float average_wait_time = wait_time * 1.0 / n;
    float average_turnaround_time = ta_time * 1.0 / n;
    printf("\nAverage Waiting Time:%f", average_wait_time);
    printf("\nAvg Turnaround Time:%f", average_turnaround_time);
    return 0;
}
```



// Priority scheduling algorithm

```
#include <stdio.h>
#define max 5
int main()
{
    int
i,j,n,t,p[max],bt[max],pr[max],wt[max],tat[max],Total_wt=0,Total_tat=0;
    float awt=0,atat=0;
    printf("Enter the number of processes\n");
    scanf("%d",&n);
    //Enter the processes according to their arrival times
    for(i=0;i<n;i++)
   {
     printf("Enter the process number\n");
          scanf("%d",&p[i]);
     printf("Enter the burst time of the process\n");
      scanf("%d",&bt[i]);
     printf("Enter the priority of the process\n");
      scanf("%d",&pr[i]);
   }
//Apply the bubble sort technique to sort the processes according to
their priorities times
for(i=0;i<n;i++)
{
 for(j=0;j<n-i-1;j++)
 {
  if(pr[j]>pr[j+1])
  {
  // Sort according to priorities
t=pr[j];
  pr[j]=pr[j+1];
  pr[j+1]=t;

  // Sorting burst times
t=bt[j];
  bt[j]=bt[j+1];
  bt[j+1]=t;
// Sorting Process numbers
  t=p[j];
  p[j]=p[j+1];
  p[j+1]=t;
  } //if
 } //for
} //for
printf("Processid \t Burst Time\t Priority\tWaiting Time\t Turn Around
Time\n");
for(i=0;i<n;i++)
{
 wt[i]=0;
 tat[i]=0;
 for(j=0;j<i;j++)
```

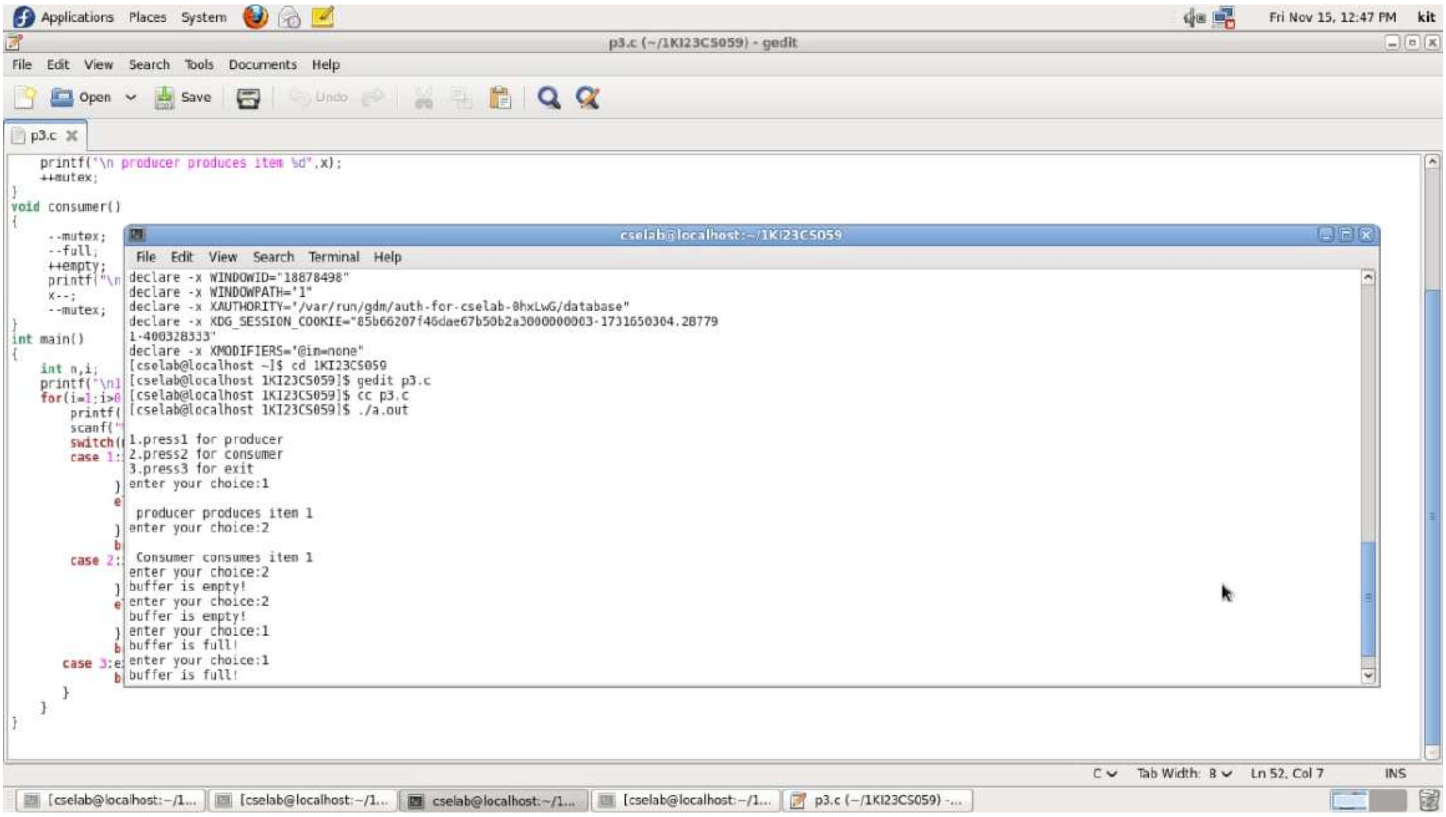
```
  wt[i]=wt[i]+bt[j];
 tat[i]=wt[i]+bt[i];
 Total_wt=Total_wt+wt[i];
 Total_tat=Total_tat+tat[i];
 printf("%d\t\t %d\t\t%d\t\t %d\t\t
%d\n",p[i],bt[i],pr[i],wt[i],tat[i]);
}
awt=(float)Total_wt/n;
atat=(float)Total_tat/n;
printf("The average waiting time =  %f\n",awt);
printf("The average turn aroud time = %f\n",atat);
return 0;
}
```



3. Develop a C program to simulate producer-consumer problem using semaphores.

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

// Initialize a mutex to 1
int mutex = 1;

// Number of full slots as 0
int full = 0;

// Number of empty slots as size of buffer
int empty = 10, x = 0;

// Function to produce an item and add it to the buffer
void producer()
{
	// Decrease mutex value by 1
	--mutex;
	// Increase the number of full
	// slots by 1
	++full;
	// Decrease the number of empty
	// slots by 1
	--empty;
	// Item produced
	x++;
	printf("\nProducer produces item %d", x);
	// Increase mutex value by 1
	++mutex;
}
// Function to consume an item and
// remove it from buffer
void consumer()
{
	// Decrease mutex value by 1
	--mutex;
	// Decrease the number of full slots by 1
	--full;
	// Increase the number of empty slots by 1
	++empty;
	printf("\nConsumer consumes item %d", x);
	x--;
	// Increase mutex value by 1
	++mutex;
}
// Driver Code
int main()
{
	int n, i;
	printf("\n1. Press 1 for Producer \n2. Press 2 for Consumer \n3.
Press 3 for Exit");
```

```
	for (i = 1; i > 0; i++) {
		printf("\nEnter your choice:");
		scanf("%d", &n);
		// Switch Cases
		switch (n) {
		case 1:
			// If mutex is 1 and empty is non-zero, then it is
possible to produce
			if ((mutex == 1)
				&& (empty != 0)) {
				producer();
			}
			// Otherwise, print buffer  is full
			else {
				printf("Buffer is full!");
			}
			break;

		case 2:
				// If mutex is 1 and full is non-zero, then it is
				//possible to consume
			if ((mutex == 1)
				&& (full != 0)) {
				consumer();
			}
			// Otherwise, print Buffer is empty
			else {
				printf("Buffer is empty!");
			}
			break;
		// Exit Condition
		case 3:
			exit(0);
			break;
		}
	}
	return 0;
}
```



Applications Places System
Fri Nov 15, 12:47 PM kit
p3.c (~/1KI23CS059) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
p3.c
printf("\n producer produces item %d",x);
++mutex;
}
void consumer()
{
--mutex;
--full;
++empty;
x--;
--mutex;
}
int main()
{
int n,i;
cselab@localhost:~/1KI23CS059
File Edit View Search Terminal Help
declare -x WINDOWID="18878498"
declare -x WINDOWPATH="1"
declare -x XAUTHORITY="/var/run/gdm/auth-for-cselab-0hxLwG/database"
declare -x XDG_SESSION_COOKIE="85b66207f46dae67b50b2a3000000003-1731650304.28779
1-400328333"
declare -x XMODIFIERS="@im=none"
[cselab@localhost ~]$ cd 1KI23CS059
[cselab@localhost 1KI23CS059]$ gedit p3.c
[cselab@localhost 1KI23CS059]$ cc p3.c
[cselab@localhost 1KI23CS059]$ ./a.out
1.press1 for producer
2.press2 for consumer
3.press3 for exit
enter your choice:1
producer produces item 1
enter your choice:2
Consumer consumes item 1
enter your choice:2
buffer is empty!
enter your choice:2
buffer is empty!
enter your choice:1
buffer is full!
enter your choice:1
buffer is full!
C Tab Width: 8 Ln 52, Col 7 INS

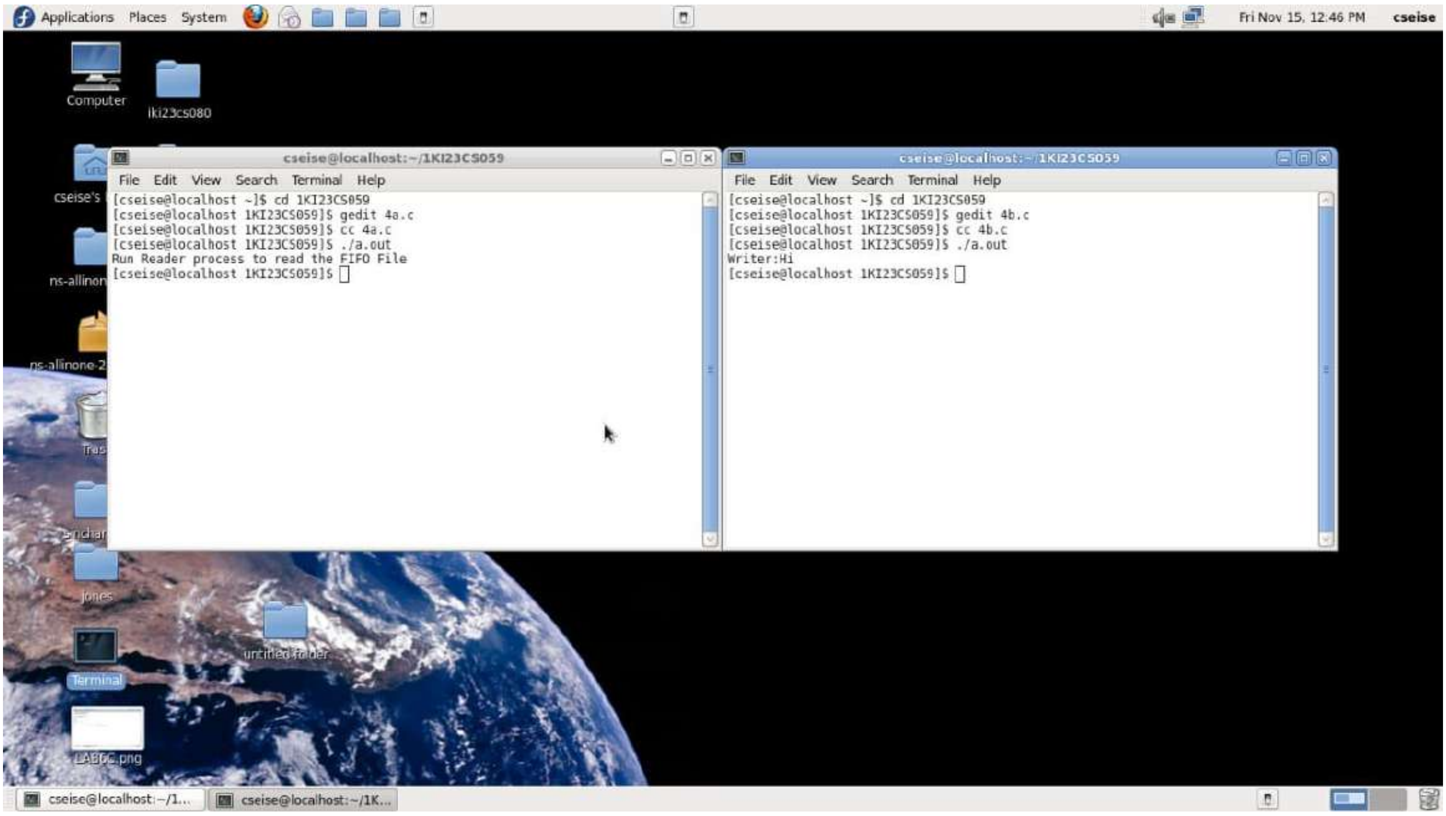
4. Develop a C program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.

/*Writer Process*/

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
 #include <sys/types.h>
#include <unistd.h>
 int main()
{
      int fd;
      char buf[1024];
      /* create the FIFO (named pipe) */
      char * myfifo = "/tmp/myfifo";
      mkfifo(myfifo, 0666);
      printf("Run Reader process to read the FIFO File\n");
      fd = open(myfifo, O_WRONLY);
      write(fd,"Hi", sizeof("Hi"));
      /* write "Hi" to the FIFO */
      close(fd);
      unlink(myfifo);
      /* remove the FIFO */
      return 0;
}
```

/*Reader Process*/

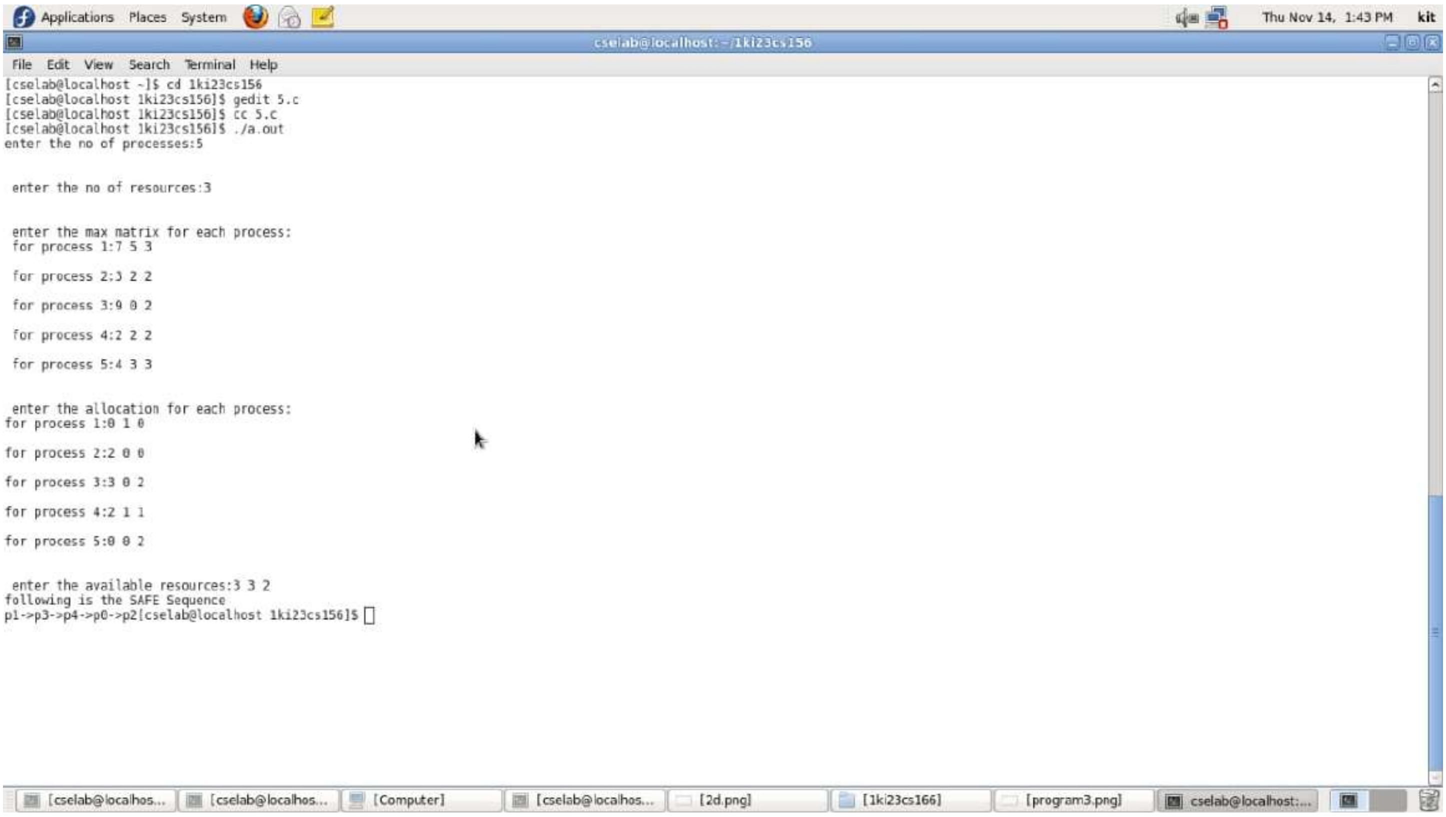
```
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
 #define MAX_BUF 1024
int main()
{
      int fd;
      /* A temp FIFO file is not created in reader */
      char *myfifo = "/tmp/myfifo";
      char buf[MAX_BUF];
      /* open, read, and display the message from the FIFO */
      fd = open(myfifo, O_RDONLY);
      read(fd, buf, MAX_BUF);
      printf("Writer: %s\n", buf);
      close(fd);
       return 0;
}
```


Applications Places System
Fri Nov 15, 12:46 PM
cseise
Computer
iki23cs080
cseise@localhost:~/1KI23CS059
File Edit View Search Terminal Help
[cseise@localhost ~]$ cd 1KI23CS059
[cseise@localhost 1KI23CS059]$ gedit 4a.c
[cseise@localhost 1KI23CS059]$ cc 4a.c
[cseise@localhost 1KI23CS059]$ ./a.out
Run Reader process to read the FIFO File
[cseise@localhost 1KI23CS059]$
cseise@localhost:~/1KI23CS059
File Edit View Search Terminal Help
[cseise@localhost ~]$ cd 1KI23CS059
[cseise@localhost 1KI23CS059]$ gedit 4b.c
[cseise@localhost 1KI23CS059]$ cc 4b.c
[cseise@localhost 1KI23CS059]$ ./a.out
Writer:Hi
[cseise@localhost 1KI23CS059]$
untitled folder
Terminal
LAB6C.png
cseise@localhost:~/1...
cseise@localhost:~/1K...

5. Develop a C program to simulate Bankers Algorithm for DeadLock Avoidance.

```
#include <stdio.h>
int main()
{
         int n, m, i, j, k,ind=0,y=0, flag=0;
         int max[10][10], avail[10], alloc[10][10], need[10][10], f[10],
         ans[10];
         //read number of processes
         printf("Enter the no of processes : ");
         scanf("%d", &n);
         //read number of resources
         printf("\n\nEnter the no of resources : ");
         scanf("%d", &m);
         //read maximum matrix
         printf("\n\nEnter the Max Matrix for each process : ");
    for(i = 0; i < n; i++)
    {
         printf("\nFor process %d : ", i + 1);
         for(j = 0; j < m; j++)
              scanf("%d", &max[i][j]);
    }
    //read allocation matrix
    printf("\n\nEnter the allocation for each process : ");
    for(i = 0; i < n; i++)
    {
         printf("\nFor process %d : ",i + 1);
         for(j = 0; j < m; j++)
              scanf("%d", &alloc[i][j]);
    }
    //read available vector
    printf("\n\nEnter the Available Resources : ");
    for(i = 0; i < m; i++)
         scanf("%d", &avail[i]);
    //initialize finish status of processes to zero
    for (k = 0; k < n; k++) {
        f[k] = 0;
    }
    //calculate need matrix
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    }
    //driver code - if need > available then can't allocate resources to
//that process else we allocate and that process executes
    for (k = 0; k < 5; k++) {
        for (i = 0; i < n; i++) {
            if (f[i] == 0) {
                flag = 0;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]){
                        flag = 1;
```

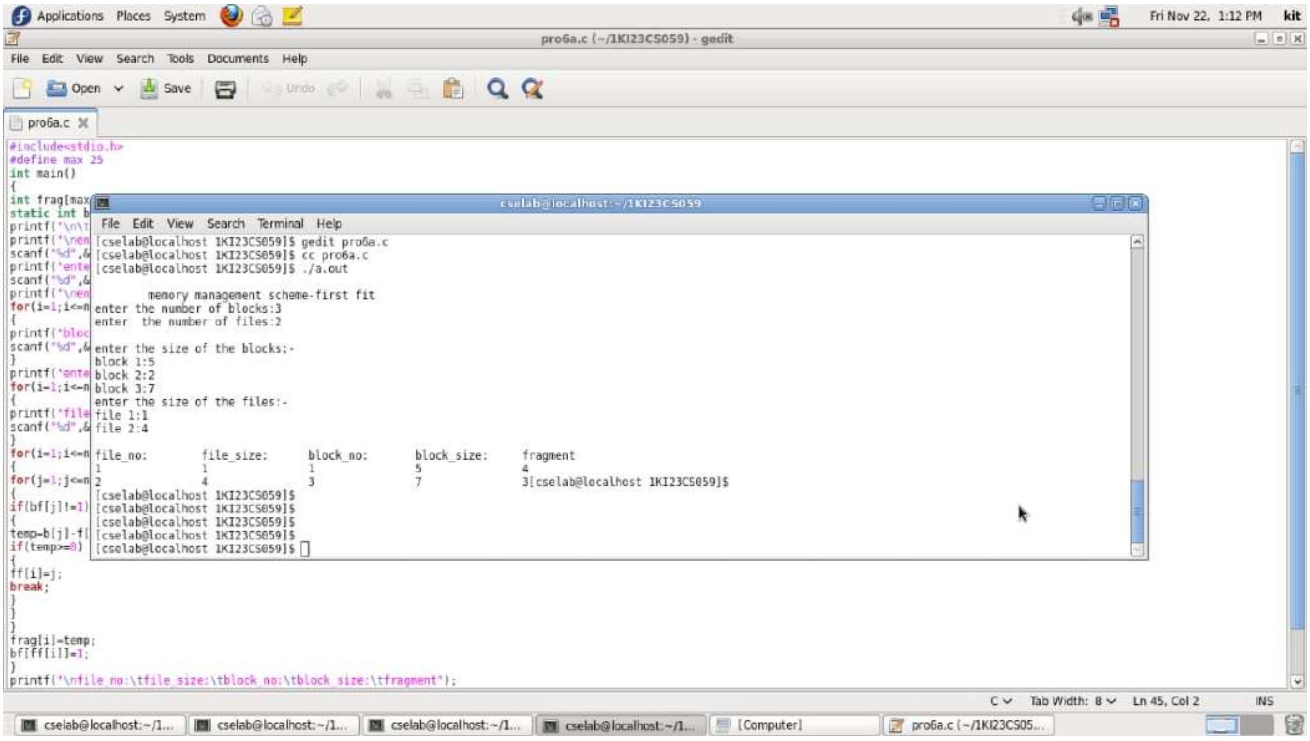
```
                        break;
                    }
                }

                if (flag == 0) {
                    ans[ind++] = i;
                  //if process finishes execution, it releases the
                  //allocated resources and available vector is updated
                    for (y = 0; y < m; y++)
                        avail[y] += alloc[i][y];
                    f[i] = 1;
                }
            }
        }
    }

      flag = 1;
      //display unsafe status
      for(i=0;i<n;i++)
    {
      if(f[i]==0)
      {
        flag=0;
         printf("The following system is not safe");
        break;
      }
    }
    //display safe state with sequence
      if(flag==1)
    {
      printf("Following is the SAFE Sequence\n");
      for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
      printf(" P%d", ans[n - 1]);
    }
    return 0;
}
```


Applications Places System
Thu Nov 14, 1:43 PM
kit
cselab@localhost:~/1ki23cs156
File Edit View Search Terminal Help
[cselab@localhost ~]$ cd 1ki23cs156
[cselab@localhost 1ki23cs156]$ gedit 5.c
[cselab@localhost 1ki23cs156]$ cc 5.c
[cselab@localhost 1ki23cs156]$ ./a.out
enter the no of processes:5
enter the no of resources:3
enter the max matrix for each process:
for process 1:7 5 3
for process 2:3 2 2
for process 3:9 0 2
for process 4:2 2 2
for process 5:4 3 3
enter the allocation for each process:
for process 1:0 1 0
for process 2:2 0 0
for process 3:3 0 2
for process 4:2 1 1
for process 5:0 0 2
enter the available resources:3 3 2
following is the SAFE Sequence
p1->p3->p4->p0->p2[cselab@localhost 1ki23cs156]$
[cselab@localhos...
[cselab@localhos...
[Computer]
[cselab@localhos...
[2d.png]
[1ki23cs166]
[program3.png]
cselab@localhost:...

6. Develop a C program to simulate the following contiguous memory allocation Techniques: a) Worst fit b) Best fit c) First fit.

a) First Fit

```
#include<stdio.h>
#define max 25
int main()
{
	int frag[max],b[max],f[max],i,j,nb,nf,temp;
	static int bf[max],ff[max];
	printf("\n\tMemory Management Scheme - First Fit");
	printf("\nEnter the number of blocks:");
	scanf("%d",&nb);
	printf("Enter the number of files:");
	scanf("%d",&nf);
	printf("\nEnter the size of the blocks:-\n");
	for(i=1;i<=nb;i++)
	{
		printf("Block %d:",i);
		scanf("%d",&b[i]);
	}
	printf("Enter the size of the files :-\n");
	for(i=1;i<=nf;i++)
	{
		printf("    File %d:",i);
		scanf("%d",&f[i]);
	}
	for(i=1;i<=nf;i++)
	{
		for(j=1;j<=nb;j++)
		{
			if(bf[j]!=1)
			{
				temp=b[j]-f[i];
				if(temp>=0)
				{
					ff[i]=j;
					break;
				}
			}
		}
		frag[i]=temp;
		bf[ff[i]]=1;
	}
	printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
	for(i=1;i<=nf;i++)
		printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
	return 0;
}
```


[cselab@localhost 1KI23CS059]$ gedit pro6a.c
[cselab@localhost 1KI23CS059]$ cc pro6a.c
[cselab@localhost 1KI23CS059]$ ./a.out
memory management scheme-first fit
enter the number of blocks:3
enter the number of files:2
enter the size of the blocks:-
block 1:5
block 2:2
block 3:7
enter the size of the files:-
file 1:1
file 2:4
file_no: file_size: block_no: block_size: fragment
1 1 1 5 4
2 4 3 7 3[cselab@localhost 1KI23CS059]$
[cselab@localhost 1KI23CS059]$
printf("\nfile_no:\tfile_size:\tblock_no:\tblock_size:\tfragment");

b) Best-fit

```
#include<stdio.h>
#define max 25
int main()
{
	int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
	static int bf[max],ff[max];
	printf("\nEnter the number of blocks:");
	scanf("     %d",&nb);
	printf("Enter the number of files:");
	scanf("%d",&nf);
	printf("\nEnter the size of the blocks:-\n");
	for(i=1;i<=nb;i++)
	{
		printf("Block %d:",i);
		scanf("%d",&b[i]);
	}
	printf("Enter the size of the files :-\n");
	for(i=1;i<=nf;i++)
	{
		printf("File %d:",i);
		scanf("%d",&f[i]);
	}
	for(i=1;i<=nf;i++)
	{
		for(j=1;j<=nb;j++)
		{
			if(bf[j]!=1)
			{
				temp=b[j]-f[i];
				if(temp>=0)
					if(lowest>temp)
					{
						ff[i]=j;
						lowest=temp;
					}
			}
		}
		frag[i]=lowest;
		bf[ff[i]]=1;
		lowest=10000;
	}
	printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
	for(i=1;i<=nf && ff[i]!=0;i++)
		printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
}
```

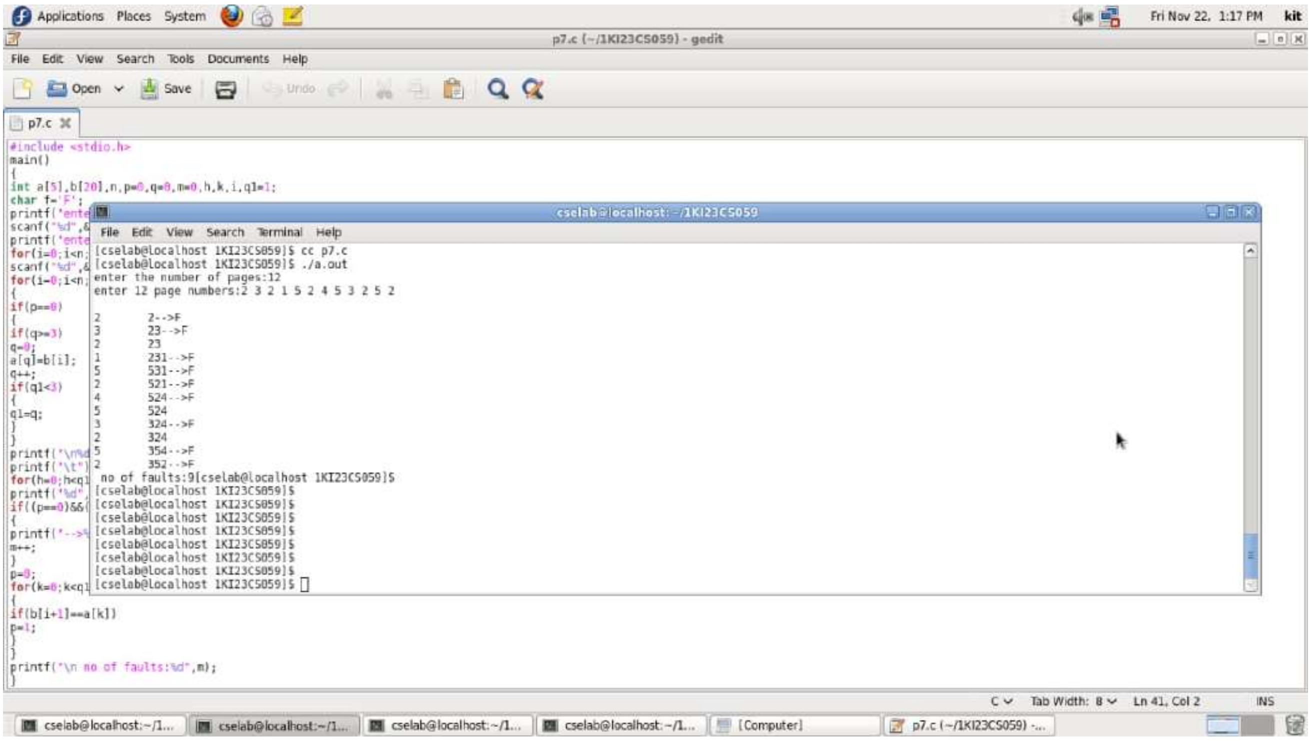
c) Worst Fit

```
#include<stdio.h>
#define max 25
int main()
{
	int frag[max],b[max],f[max],i, j, nb, nf, temp, highest=0;
	static int bf[max],ff[max];
	printf("\n\tMemory Management Scheme - Worst Fit");
	printf("\nEnter the number of blocks:");
	scanf("%d",&nb);
	printf("Enter the number of files:");
	scanf("%d",&nf);
	printf("\nEnter the size of the blocks:-\n");
	for(i=1;i<=nb;i++)
	{
		printf("Block %d:",i);
		scanf("%d",&b[i]);
	}
	printf("Enter the size of the files :-\n");
	for(i=1;i<=nf;i++)
	{
		printf("File %d:",i);
		scanf("%d",&f[i]);
	}
	for(i=1;i<=nf;i++)
	{
		for(j=1;j<=nb;j++)
		{
			if(bf[j]!=1) //if bf[j] is not allocated
			{
				temp=b[j]-f[i];
				if(temp>=0)
					if(highest<temp)
					{
						ff[i]=j;
						highest=temp;
					}
			}
		}
		frag[i]=highest;
		bf[ff[i]]=1;
		highest=0;
	}
	printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
	for(i=1;i<=nf;i++)
		printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
}
```

7. Develop a C program to simulate page replacement algorithms: a) FIFO b) LRU

a) FIFO

```
#include<stdio.h>
main()
{
	int a[5],b[20],n,p=0,q=0,m=0,h,k,i,q1=1;
	char f='F';
	printf("Enter the Number of Pages:");
	scanf("%d",&n);
	printf("Enter %d Page Numbers:",n);
	for(i=0;i<n;i++)
		scanf("%d",&b[i]);
	for(i=0;i<n;i++)
	{
		if(p==0)
		{
			if(q>=3)
				q=0;
			a[q]=b[i];
			q++;
			if(q1<3)
			{
				q1=q;
			}
		}
		printf("\n%d",b[i]);
		printf("\t");
		for(h=0;h<q1;h++)
			printf("%d",a[h]);
		if((p==0)&&(q<=3))
		{
			printf("-->%c",f);
			m++;
		}
		p=0;
		for(k=0;k<q1;k++)
		{
			if(b[i+1]==a[k])
				p=1;
		}
	}
	printf("\nNo of faults:%d",m);
}
```



Applications Places System
Fri Nov 22, 1:17 PM
p7.c (~/1KI23CS059) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
p7.c
#include <stdio.h>
main()
{
int a[5],b[20],n,p=0,q=0,m=0,h,k,i,q1=1;
char f='F';
cselab@localhost:~/1KI23CS059
File Edit View Search Terminal Help
[cselab@localhost 1KI23CS059]$ cc p7.c
[cselab@localhost 1KI23CS059]$ ./a.out
enter the number of pages:12
enter 12 page numbers:2 3 2 1 5 2 4 5 3 2 5 2
2 2-->F
3 23-->F
2 23
1 231-->F
5 531-->F
2 521-->F
4 524-->F
5 524
3 324-->F
2 324
5 354-->F
2 352-->F
no of faults:9[cselab@localhost 1KI23CS059]$
[cselab@localhost 1KI23CS059]$
if(b[i+1]==a[k])
p=1;
}
}
printf("\n no of faults:%d",m);
}
Tab Width: 8 Ln 41, Col 2 INS
cselab@localhost:~/1...
[Computer]
p7.c (~/1KI23CS059) -...

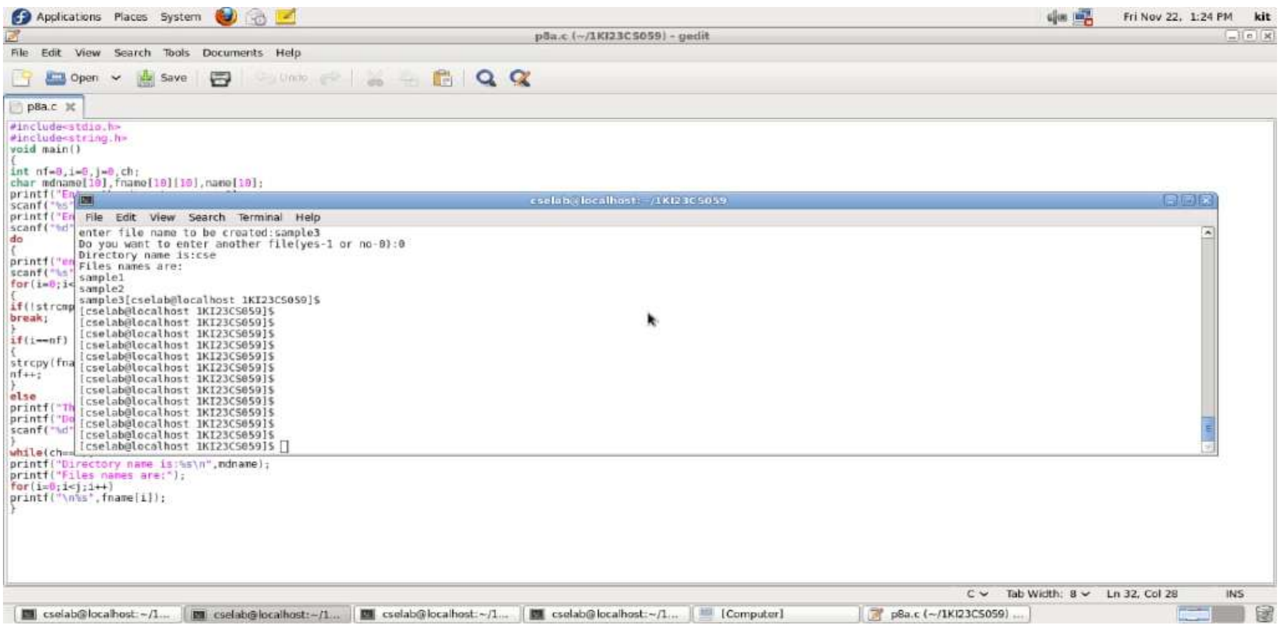
b) LRU

```
#include<stdio.h>
main()
{
	int a[5],b[20],p=0,q=0,m=0,h,k,i,q1=1,j,u,n;
	char f='F';
	printf("Enter the number of pages:");
	scanf("%d",&n);
	printf("Enter %d Page Numbers:",n);
	for(i=0;i<n;i++)
		scanf("%d",&b[i]);
	for(i=0;i<n;i++)
	{
		if(p==0)
		{
			if(q>=3)
				q=0;
			a[q]=b[i];
			q++;
			if(q1<3)
			{
				q1=q;
			}
		}
		printf("\n%d",b[i]);
		printf("\t");
		for(h=0;h<q1;h++)
			printf("%d",a[h]);
			if((p==0)&&(q<=3))
			{
				printf("-->%c",f);
				m++;
			}
			p=0;
			if(q1==3)
			{
				for(k=0;k<q1;k++){
					if(b[i+1]==a[k])
						p=1;
				}
				for(j=0;j<q1;j++){
					u=0;
					k=i;
					while(k>=(i-1)&&(k>=0)) {
						if(b[k]==a[j])
							u++;
						k--;
					}
					if(u==0)
						q=j;
				}
			}
			else
			{
				for(k=0;k<q;k++) {
					if(b[i+1]==a[k])
						p=1;
				}
			}
	}
	printf("\nNo of faults:%d",m);
}
```

8. Simulate following File Organization Techniques a) Single level directory b) Two level directory.

a). Single level directory

```
#include<stdio.h>
#include<string.h>
void main()
{
	int nf=0,i=0,j=0,ch;
	char mdname[10],fname[10][10],name[10];
	printf("Enter the directory name:");
	scanf("%s",mdname);
	printf("Enter the number of files:");
	scanf("%d",&nf);
	do
	{
		printf("Enter file name to be created:");
		scanf("%s",name);
		for(i=0;i<nf;i++)
		{
			if(!strcmp(name,fname[i]))
				break;
		}
		if(i==nf)
		{
			strcpy(fname[j++],name);
			nf++;
		}
		else
			printf("There is already %s\n",name);
		printf("Do you want to enter another file(yes - 1 or no - 0):");
		scanf("%d",&ch);
	}while(ch==1);
	printf("Directory name is:%s\n",mdname);
	printf("Files names are:");
	for(i=0;i<j;i++)
		printf("\n%s",fname[i]);
}
```

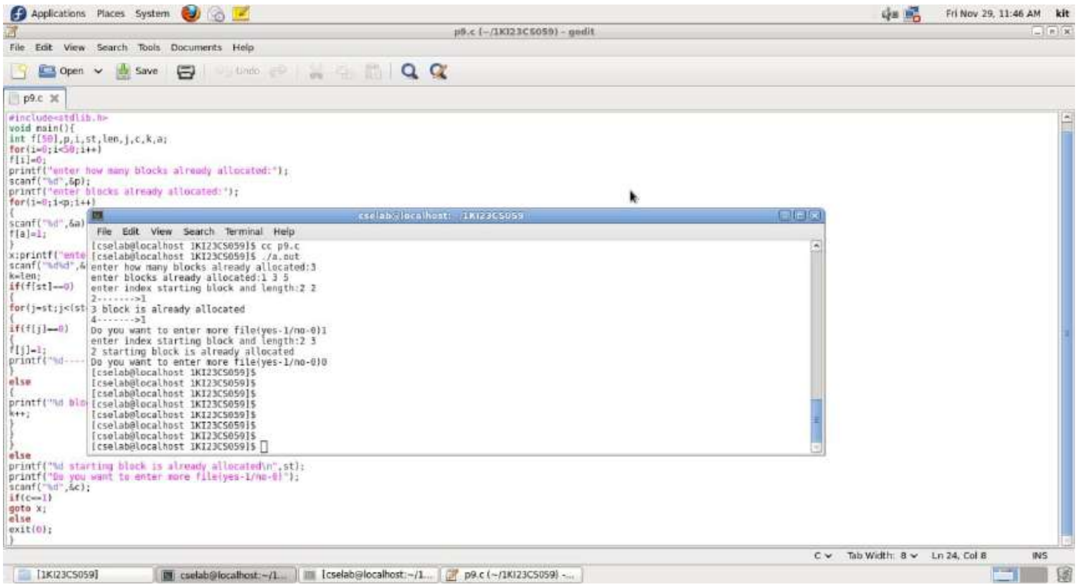


b). Two Level Directory

```
#include<stdio.h>
struct st
{
	char dname[10];
	char sdname[10][10];
	char fname[10][10][10];
	int ds,sds[10];
}dir[10];
void main()
{
	int i,j,k,n;
	printf("enter number of directories:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("enter directory %d names:",i+1);
		scanf("%s",&dir[i].dname);
		printf("enter size of directories:");
		scanf("%d",&dir[i].ds);
		for(j=0;j<dir[i].ds;j++)
		{
			printf("enter subdirectory name and size:");
			scanf("%s",&dir[i].sdname[j]);
			scanf("%d",&dir[i].sds[j]);
			for(k=0;k<dir[i].sds[j];k++)
			{
				printf("enter file name:");
				scanf("%s",&dir[i].fname[j][k]);
			}
		}
	}
	printf("\ndirname\t\tsize\tsubdirname\tsize\tfiles");
	printf("\n*****************************************************\n");
	for(i=0;i<n;i++)
	{
		printf("%s\t\t%d",dir[i].dname,dir[i].ds);
		for(j=0;j<dir[i].ds;j++)
		{
			printf("\t%s\t\t%d\t",dir[i].sdname[j],dir[i].sds[j]);
			for(k=0;k<dir[i].sds[j];k++)
				printf("%s\t",dir[i].fname[j][k]);
			printf("\n\t\t");
		}
		printf("\n");
	 }
	printf("\n");
}
```

9. Develop a C program to simulate the Linked file allocation strategies.

```
#include<stdio.h>
#include<stdlib.h>
void main(){
	int f[50], p,i, st, len, j, c, k, a;
	for(i=0;i<50;i++)
		f[i]=0;
	printf("Enter how many blocks already allocated: ");
	scanf("%d",&p);
	printf("Enter blocks already allocated: ");
	for(i=0;i<p;i++)
	{
		scanf("%d",&a);
		f[a]=1;
	}
	x: printf("Enter index starting block and length: ");
	scanf("%d%d", &st,&len);
	k=len;
	if(f[st]==0)
	{
		for(j=st;j<(st+k);j++)
		{
			if(f[j]==0)
			{
				f[j]=1;
				printf("%d-------->%d\n",j,f[j]);
			}
			else
			{
				printf("%d Block is already allocated \n",j);
				k++;
			}
		}
	}
	else
		printf("%d starting block is already allocated \n",st);
	printf("Do you want to enter more file(Yes - 1/No - 0)");
	scanf("%d", &c);
	if(c==1)
		goto x;
	else
		exit(0);
}
```



10. Develop a C program to simulate SCAN disk scheduling algorithm.

```
#include<stdio.h>
void main()
{
		int
queue[20],n,head,i,j,k,seek=0,max,diff,temp,queue1[20],queue2[20],temp1=0,temp2=0;
		float avg;
	printf("Enter the max range of disk\n");
		scanf("%d",&max);
		printf("Enter the initial head position\n");
		scanf("%d",&head);
		printf("Enter the size of queue request\n");
		scanf("%d",&n);
		printf("Enter the queue of disk positions to be read\n");
		for(i=1;i<=n;i++)
		{
				scanf("%d",&temp);
				if(temp>=head)
				{
						queue1[temp1]=temp;
						temp1++;
				}
				else
				{
						queue2[temp2]=temp;
						temp2++;
				}
		}
		for(i=0;i<temp1-1;i++)
		{
				for(j=i+1;j<temp1;j++)
				{
						if(queue1[i]>queue1[j])
						{
								temp=queue1[i];
								queue1[i]=queue1[j];
								queue1[j]=temp;
						}
				}
		}
		for(i=0;i<temp2-1;i++)
		{
				for(j=i+1;j<temp2;j++)
				{
						if(queue2[i]<queue2[j])
						{
								temp=queue2[i];
								queue2[i]=queue2[j];
								queue2[j]=temp;
						}
				}
		}
		for(i=1,j=0;j<temp1;i++,j++)
		queue[i]=queue1[j];
		queue[i]=max;
		for(i=temp1+2,j=0;j<temp2;i++,j++)
		queue[i]=queue2[j];
		queue[i]=0;
		queue[0]=head;
		for(j=0;j<=n+1;j++)
		{
				diff=abs(queue[j+1]-queue[j]);
```

```
                                seek+=diff;
                                printf("Disk head moves from %d to %d with seek
%d\n",queue[j],queue[j+1],diff);
                }
                printf("Total seek time is %d\n",seek);
                avg=seek/(float)n;
                printf("Average seek time is %f\n",avg);
}
```