OS Manual

Program 1: Write a program to read data from the standard input device and write it on the screen(using read()/write() system calls)

```
#include int main()
{
        int nread;
        char buff[20];
        nread=read(0,buff,10); //read 10 bytes from standard input device(keyboard) and store
                                                it in buffer(buff)
         write (1,buff,nread); //print 10 bytes from the buffer on the screen
}
```

Commands:
cc filename.c

./a.out

Pesce mandya
Pesce mandya

Program 2: Write a program to print 10 characters starting from the 10th character from a file(lseek() system call)

//Let the contents of the file F1 be “1234567890abcdefghijxxxxxxxx”. This means we want the output to be “abcdefghij”.
//Note: the first character ‘1’ is at 0th position

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
int main()
{
int n,f,f1;
char buff[10];
f=open("seeking",O_RDWR);
```

```
f1=lseek(f,10,SEEK_SET);
printf("Pointer is at %d position\n",f1);
read(f,buff,10);
write(1,buff,10);
}
```

Program 3: Write a program to implement IPC using shared memory.

Shared Memory for Writer Process

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

int main()
{
 int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);
  //creates shared memory segment with key 2345, having size 1024 bytes. IPC_CREAT
  //is used to create the shared segment if it does not exist. 0666 are the permisions on the
  //shared segment.
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0); //process attached to shared memory segment
printf("Process attached at %p\n",shared_memory);
  //this prints the address where th segment is attached with this process
printf("Enter some data to write to shared memory\n");
read(0,buff,100);     //get some input from user
strcpy(shared_memory,buff);    //data written to shared memory
printf("You wrote : %s\n",(char *)shared_memory);
}
```

Output:

```
Key of shared memory is 0
Process attached at x7ffe04fb000
Enter some data to write to shared memory
```

Hello World
You wrote: Hello World

Shared Memory for Reader Process

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

int main()
{
int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345, 1024, 0666);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0);   //process attached to shared memory segment
printf("Process attached at %p\n",shared_memory);
printf("Data read from shared memory is : %s\n",(char *)shared_memory);
}
```

Output:
Key of Shared memory is 0
Process attached at 0x7f76b4292999
Data read from shared memory is: Hello World

Program 4: Implement the Producer & consumer Problem (Semaphore)

```
#include <stdio.h>

void main()
{
        int buffer[10], bufsize, in, out, produce, consume, choice=0;
        in = 0; out = 0; bufsize = 10;
        while(choice !=3)
```

```
{
        printf("\n 1. Produce \t 2. Consume \t3. Exit ");
        printf("\n Enter your choice: ");
        scanf("%d", &choice);
        switch(choice)
        {
                case 1:
                        if((in+1)%bufsize==out)
                                printf("\n Buffer is Full");
                        else
                        {
                                printf("\n Enter the value: ");
                                scanf("%d", &produce);
                                buffer[in] = produce;
                                in = (in+1)%bufsize;
                        }
                        break;
                case 2:
                        if(in == out)
                                printf("\n Buffer is Empty");
                        else
                        {
                                consume = buffer[out];
                                printf("\n The consumed value is %d", consume);
                                out = (out+1)%bufsize;
                        }
                        break;
                }
        }
}
```

Program 5: Implement the solution to dining philosopher's problem using monitors.

```
#include <stdio.h>
#include <stdlib.h>
int one( );
int two( ) ;
int tph, philname[20], status[20], howhung, hu[20], cho;
```

```
int main()
{
	int i;
	printf("\n\nDINING PHILOSOPHER PROBLEM");
	printf("\nEnter the total no. of philosophers: ");
	scanf("%d",&tph);
	for(i=0;i<tph;i++)
	{
		philname[i] = (i+1); status[i]=1;
	}
	printf("How many are hungry : ");
	scanf("%d", &howhung);
	if(howhung==tph)
	{
		printf("\nAll are hungry..\nDead lock stage will occur");
		printf("\nExiting..");
	}
	else
	{
		for(i=0;i<howhung;i++)
		{
			printf("Enter philosopher %d position: ",(i+1));
			scanf("%d", &hu[i]);
			status[hu[i]]=2;
		}
 do {
	printf("1.One can eat at a time\t2.Two can eat at a time\t3.Exit\nEnter your choice:");
	scanf("%d", &cho);
	switch(cho)
	{
		case 1:
			one(); break;
		case 2:
			two(); break;
		case 3:
			exit(0);
		default:
			printf("\nInvalid option..");
	}
 }
 while(1);
```

```
 }
}
int one()
{
        int pos=0, x, i;
        printf("\nAllow one philosopher to eat at any time\n");

        for(i=0;i<howhung; pos++)
        {
                Printf("\nP %d is granted to eat", philname[hu[pos]]);
                for(x=pos; x<howhung; x++)
                        printf("\nP %d is waiting", philname[hu[x]]);
        }
}

int two()
{
        int i, j, s=0, t, r, x;
        printf("\n Allow two philosophers to eat at same time\n");
        for(i=0;i<howhung; i++)
        {
            for(j=i+1;j<howhung; j++)
            {
                if(abs(hu[i]-hu[j])>=1&& abs(hu[i]-hu[j])!=4)
                {
                 printf("\n\ncombination %d \n", (s+1));
                 t=hu[i];
                 r=hu[j];
                 s++;
                 printf("\nP %d and P %d are granted to eat", philname[hu[i]], philname[hu[j]]);
                for(x=0;x<howhung;x++)
                {
                        if((hu[x]!=t)&&(hu[x]!=r))
                                printf("\nP %d is waiting", philname[hu[x]]);
                }
         }
   }
  }
}
```

Program 6: Implement the FCFS CPU Scheduling Algorithms

```
#include <stdio.h>
#include <conio.h>
main()
{
	int bt[20], wt[20], tat[20], i, n;
	float wtavg, tatavg;
	clrscr();
	printf("\nEnter the number of processes -- ");
	scanf("%d", &n);
	for(i=0;i <n;i++)
	{
		printf("\nEnter Burst Time for Process %d -- ", i);
		scanf("%d", &bt[i]);
	}
	wt[0] = wtavg = 0;
	tat[0] = tatavg = bt[0];
	for(i=1;i <n;i++)
	{
		wt[i] = wt[i-1] +bt[i-1];
		tat[i] = tat[i-1] +bt[i];
		wtavg = wtavg + wt[i];
		tatavg = tatavg + tat[i];
	}
	printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
	}
	for(i=0;i<n;i++)
		printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
	printf("\nAverage Waiting Time -- %f", wtavg/n);
	printf("\nAverage Turnaround Time -- %f", tatavg/n);
	getch();
}
```

Program 7: Implement Bankers Algorithm for Deadlock Avoidance

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

int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void input();
void show();
void cal();

int main()
{
        int i,j;
        printf("********** Banker's Algorithm ************\n");
        input();
        show();
        cal();
        getch();
        return 0;
}
void input()
{
        int i,j;
        printf("Enter the no of Processes\t");
        scanf("%d",&n);
        printf("Enter the no of resources instances\t");
        scanf("%d",&r);
        printf("Enter the Max Matrix\n");
        for(i=0; i<n;i++)
        {
           for(j=0;j<r;j++)
           {
                scanf("%d",&max[i][j]);
            }
        }
printf("Enter the Allocation Matrix\n");
for(i=0;i<n;i++)
```

```
{
  for(j=0;j<r;j++)
   { scanf("%d",&alloc[i][j]);
  }
}
printf("Enter the available Resources\n");
for(j=0;j <r;j++)
{
        scanf("%d",&avail[j]);
        }
}

void show()
{
        int i,j;
        printf("Process\t Allocation\t Max\t Available\t");
        for(i=0;i<n;i++)
        {
                printf("\nP%d\t ",i+1);
                for(j=0;j <r;j++)
                {   printf("%d ",alloc[i][j]); }
                printf("\t\t");
                for(j=0;j<r;j++)
                {         printf("%d ",max[i][j]); }
                printf("\t\t");
        if(i==0)
        {
        for(j=0;j <r;j++)
        printf("%d ",avail[j]);
        }
        printf("\t\t");
        }
        printf("\n");
 }
void cal()
{
        int finish[100],temp,need[100][100],flag=1,k,c1=0;
        int safe[100];
        int i,j;
        for(i=0;i<n;i++)
        {         finish[i]=0; }
```

```
            //find need matrix
            for(i=0; i<n;i++)
            {
              for(j=0;j<r;j++)
              {
                        need[i][j]=max[i][j]-alloc[i][j];
            }
}
printf("\n");
//print need matrix
printf("----Need Matrix-------\n");
for(i=0;i<n;i++)
{
for(j=0;j <r;j++)
{           printf("%d ",need[i][j]);
}
printf("\n");
}
printf("\n");

while(flag)
 {
   flag=0;
   for(i=0;i<=n; i++)
  {    int c=0;
       for(j=0; j<r; j++)
       {     if ((finish[i]==0) && (need[i][j]<=avail[j]))
          {
             c++;
        if(c==r)
        {
      for(k=0;k<r; k++)
    {
            avail[k]+=alloc[i][j];
            finish[i]=1;
            flag=1;
}
printf("P%d->",i);
if(finish[i]==1)
{           i=n;       }
} }
```

```
} }
}

printf("\n\n");
for(i=0;i<n;i++)
{
 if(finish[i]==1)
        { c1++; }
else
        { printf("P%d->",i); }
}
if(c1==n)
{       printf("\n The system is in safe state"); }
else
{ printf("\n Process are in dead lock");
printf("\n System is in unsafe state");
}
}
```

Program 8: Implement the following Memory Allocation Methods for fixed partition

a) First Fit b) Worst Fit

A. First Fit

```
#include <stdio.h>
#include <conio.h>
#define max 25
void main()
{
        int frag[max],b[max],f[max],i,j,nb,nf,temp;
        static int bf[max],ff[max];
        clrscr();
        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]);
getch();
}
```

B. Worst Fit

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

#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
clrscr();
```

```
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]);
getch();
}
```

Program 9: Implement the following Page Replacement Algorithms

a) FIFO b) LRU

A. FIFO

```
#include <stdio.h>
#include <conio.h>
main()
{
        int i, j, k, f, pf=0, count=0, rs[25], m[10], n;
        clrscr();
        printf("\n Enter the length of reference string -- ");
        scanf("%d",&n);
        printf("\n Enter the reference string -- ");
        for(i=0;i<n;i++)
          scanf("%d",&rs[i]);
        printf("\n Enter no. of frames -- ");
        scanf("%d",&f);
        for(i=0;i<f;i++)
                m[i]=-1;
        printf("\n The Page Replacement Process is -- \n");
        for(i=0;i<n;i++)
        {
          for(k=0;k<f;k++)
          {
             if(m[k]==rs[i]) break; }
           if(k==f)
            { m[count++]=rs[i]; pf++; }
        for(j=0;j <f;j++)
        printf("\t%d",m[j]);
        if(k==f)
          printf("\tPF No. %d",pf);
         printf("\n");
        if(count==f)
           count=0;
        }
        printf("\n The number of Page Faults using FIFO are %d",pf);
        getch();
```

```
}
```

B. LRU

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

main()
{
        int i, j , k, min, rs[25], m[10], count[10], flag[25], n, f, pf=0, next=1;
        clrscr();
        printf("Enter the length of reference string -- ");
        scanf("%d",&n);
        printf("Enter the reference string -- ");
        for(i=0;i<n;i++)
        {
           scanf("%d",&rs[i]);
           flag[i]=0;
        }
        printf("Enter the number of frames -- ");
        scanf("%d",&f);
        for(i=0;i<f;i++)
        {          count[i]=0; m[i]=-1; }
        printf("\nThe Page Replacement process is -- \n");
        for(i=0;i<n;i++)
        {
        for(j=0;j<f;j++)
        {   if(m[j]==rs[i])
           { flag[i]=1; count[j]=next; next++; }
        }

        if(flag[i]==0)
        {
          if(i<f)
          {  m[i]=rs[i]; count[j]=next; next++;
           }
        else
        {
          min=0;
            for(j=1;j<f;j++)
```

```
                if(count[min]> count[j])
                   min=j; m[min]=rs[i];
          count[min]=next;
          next++;
   }
pf++;
}
for(j=0;j<f;j++)
  printf("%d\t", m[j]);
     if(flag[i]==0)
          printf("PF No. -- %d" , pf);
 printf("\n");
}
printf("\nThe number of page faults using LRU are %d",pf);
getch();
}
```

Program 10 Implement the following Disk Scheduling Algorithms:
a) SSTF Scheduling b) SCAN Scheduling

A. SSTF Scheduling

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main()
{
          int queue[100],t[100],head,seek=0,n,i,j,temp;
          float avg;
          clrscr();
          printf("*** SSTF Disk Scheduling Algorithm ***\n");
          printf("Enter the size of Queue\t");
          scanf("%d",&n);
          printf("Enter the Queue\t");
          for(i=0;i<n;i++)
          {          scanf("%d",&queue[i]);          }
          printf("Enter the initial head position\t");
```

```
scanf("%d",&head);
for(i=1;i<n;i++)
        t[i]=abs(head-queue[i]);
for(i=0;i<n;i++)
{
   for(j=i+1;j<n;j++)
   {
        if(t[i]>t[j])
        {
                temp=t[i]; t[i]=t[j]; t[j]=temp; temp=queue[i];
                queue[i]=queue[j]; queue[j]=temp;
        }
 } }
for(i=1;i<n-1;i++)
{       seek=seek+abs(head-queue[i]);
        head=queue[i];
}
printf("\nTotal Seek Time is%d\t",seek);
avg=seek/(float)n;
printf("\nAverage Seek Time is %f\t",avg);
return 0;
}
```

OUTPUT:

```
*** SSTF Disk Scheduling Algorithm ***
Enter the size of Queue 5
Enter the Queue 10 17 2 15 4
Enter the initial head position 3
Total Seek Time is14
Average Seek Time is 2.800000
```

RESULT:

Thus the program was executed and verified successfully.

B. SCAN Scheduling

SCAN DISK SCHEDULING ALGORITHM

```
#include <stdio.h>
main()
{
        int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0;
        clrscr();
```

```
                printf("enter the no of tracks to be traveresed");
                scanf("%d'",&n);
                printf("enter the position of head");
                scanf("%d",&h);
                t[0]=0;t[1]=h;
                printf("enter the tracks");
                for(i=2;i<n+2;i++)
                        scanf("%d",&t[i]);
                for(i=0;i<n+2;i++)
                {
                  for(j=0;j<(n+2)-i-1;j++)
                  {
                        if(t[j]>t[j+1])
                        { temp=t[j]; t[j]=t[j+1]; t[j+1]=temp; }
                  }
                }
                for(i=0;i<n+2;i++)
                if(t[i]==h)
                        j=i; k=i; p=0;
                while(t[j]!=0)
                {
                         atr[p]=t[j];
                        j--;
                        p++;
                }
                atr[p]=t[j];
                for(p=k+1;p<n+2;p++,k++)
                        atr[p]=t[k+1];
                for(j=0;j<n+1;j++)
                {
                  if(atr[j]>atr[j+1])
                        d[j]=atr[j]-atr[j+1];
                  else
                        d[j]=atr[j+1]-atr[j];
                        sum+=d[j];
                }
                printf("\nAverage header movements:%f",(float)sum/n);
                getch();
}
```

INPUT:

Enter no. of tracks:9
Enter track position:55 58 60 70 18 90 150 160 184

OUTPUT:

Tracks Traversed	Difference Between tracks
150	50
160	10
184	24
90	94
70	20
60	10
58	2
55	3
18	37