Program 1 Shell Script :

Os Assignment 1 - Part B

#! /bin/bash opt=1

while [ "$opt" -lt 6 ] do

echo -e "Choose one of the Following\n1. Create a New Address Book\n2. View Records\n3. Insert new Record\n4. Delete a Record\n5. Modify a Record\n6. Exit"

# echo -e, enables special features of echo to use \n \t \b etc.

read opt case $opt in

1)

echo "Enter filename" read fileName

if [ -e $fileName ] ; then # -e to check if file exists, if exits remove the file

rm $fileName

fi

cont=1 echo

"NAME\tNUMBER\t\tADDRESS\n===============================\n"

| cat >> $fileName while [ "$cont" -gt 0 ] do

echo "\nEnter Name" read name

echo "Enter Phone Number of $name" read number

echo "Enter Address of $name" read address

echo "$name\t$number\t\t$address" | cat >> $fileName

echo "Enter 0 to Stop, 1 to Enter next"

read cont

done

;;

2)

cat $fileName

;;

3)

echo "\nEnter Name" read name

echo "Enter Phone Number of $name" read number echo "Enter Address of $name"

read address

echo "$name\t$number\t\t$address" | cat >> $fileName

;;

4)

echo "Delete record\nEnter Name/Phone Number"

read pattern temp="temp" grep -v $pattern $fileName | cat >> $temp rm $fileName cat $temp | cat >> $fileName rm $temp

;;

5)

echo "Modify record\nEnter Name/Phone Number"

read pattern temp="temp" grep -v $pattern $fileName | cat >> $temp rm $fileName cat $temp | cat >> $fileName rm $temp echo "Enter Name" read name echo "Enter Phone Number of $name" read number echo "Enter Address of $name"

read address

echo -e "$name\t$number\t$address" | cat >> $fileName

;;

esac done

Output:

(base) rmdstic@rmdstic-OptiPlex-3010:~$ ./1.b.sh

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

1 Enter filename employee \nEnter Name

Apeksha

Enter Phone Number of Apeksha

7854855457

Enter Address of Apeksha

XYZ

Enter 0 to Stop, 1 to Enter next

1

\nEnter Name

Samrudhi

Enter Phone Number of Samrudhi

8754755474

Enter Address of Samrudhi

PQR

Enter 0 to Stop, 1 to Enter next

1

\nEnter Name

Sankalp

Enter Phone Number of Sankalp

2385485647

Enter Address of Sankalp

LMN

Enter 0 to Stop, 1 to Enter next

0

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

2

NAME\tNUMBER\t\tADDRESS\n===============================\n

Apeksha\t7854855457\t\tXYZ

Samrudhi\t8754755474\t\tPQR

Sankalp\t2385485647\t\tLMN

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

3

\nEnter Name

Ajay

Enter Phone Number of Ajay

6587435675

Enter Address of Ajay

ABC

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

2

NAME\tNUMBER\t\tADDRESS\n===============================\n

Apeksha\t7854855457\t\tXYZ

Samrudhi\t8754755474\t\tPQR

Sankalp\t2385485647\t\tLMN

Ajay\t6587435675\t\tABC

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

4

Delete record\nEnter Name/Phone Number Ajay

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

2

NAME\tNUMBER\t\tADDRESS\n===============================\n

Apeksha\t7854855457\t\tXYZ

Samrudhi\t8754755474\t\tPQR

Sankalp\t2385485647\t\tLMN

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

5

Modify record\nEnter Name/Phone Number

Apeksha

Enter Name

Apeksha

Enter Phone Number of Apeksha

8754785574

Enter Address of Apeksha

QRT

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

2

NAME\tNUMBER\t\tADDRESS\n===============================\n

Samrudhi\t8754755474\t\tPQR

Sankalp\t2385485647\t\tLMN

Apeksha 8754785574 QRT

Choose one of the Following

1. Create a New Address Book
2. View Records
3. Insert new Record
4. Delete a Record
5. Modify a Record
6. Exit

6

(base) rmdstic@rmdstic-OptiPlex-3010:~$ ^C

(base) rmdstic@rmdstic-OptiPlex-3010:~$ ^C

Parent Child Process using Fork and Wait

# include<stdio.h>

# include <stdlib.h>

# include<sys/types.h>

# include<unistd.h>

# include<sys/wait.h>

int split ( int[], int , int ); void quickSort(int\* ,int, int);

void mergeSort(int arr[],int low,int mid,int high)

{

int i,j,k,l,b[20];

l=low; i=low;

j=mid+1;

while((l<=mid)&&(j<=high))

{

if(arr[l]<=arr[j])

{ b[i]=arr[l]; l++;

}

else { b[i]=arr[j];

j++; } i++;

}

if(l>mid)

{

for(k=j;k<=high;k++)

{ b[i]=arr[k];

i++;

}

}

else {

for(k=l;k<=mid;k++)

{

b[i]=arr[k];

i++; }

}

for(k=low;k<=high;k++)

{

arr[k]=b[k];

}

}

void partition(int arr[],int low,int high)

{

int mid; if(low<high) {

double temp; mid=(low+high)/2; partition(arr,low,mid); partition(arr,mid+1,high);

mergeSort(arr,low,mid,high);

}

}

void display(int a[],int size)

{

int i;

for(i=0;i<size;i++)

{

printf("%d\t\t",a[i]);

}

printf("\n");

}

int main() {

int pid, child\_pid;

int size,i,status;

printf("Enter the number of Integers to

Sort::::\t"); scanf("%d",&size); int a[size]; int pArr[size]; int cArr[size];

for(i=0;i<size;i++){

printf("Enter number %d:",(i+1)); scanf("%d",&a[i]);

pArr[i]=a[i]; cArr[i]=a[i];

}

printf("Your Entered Integers for Sorting\n"); display(a,size);

pid=getpid();

printf("Current Process ID is : %d\n",pid); printf("[ Forking Child Process ... ] \n");

child\_pid=fork(); if( child\_pid < 0){ printf("\nChild Process Creation Failed!!!!\n");

exit(-1);

}

else if( child\_pid==0) { printf("\nThe Child Process\n"); printf("\nchild process is %d",getpid()); printf("\nparent of child process is %d",getppid()); printf("Child is sorting the list of Integers by QUICK SORT::\n");

quickSort(cArr,0,size-1); printf("The sorted List by Child::\n");

display(cArr,size);

printf("Child Process Completed ...\n");

sleep(10);

printf("\nparent of child process is %d",getppid());

}

else {

printf("parent process %d started\n",getpid()); printf("Parent of parent is %d\n",getppid());

sleep(30);

printf("The Parent Process\n");

printf("Parent %d is sorting the list of Integers by MERGE SORT\n",pid); partition(pArr,0,size-1); printf("The sorted List by Parent::\n");

display(pArr,size); wait(&status);

printf("Parent Process Completed ...\n");

}

return 0;

} int split ( int a[ ], int lower, int upper )

{ int i, p, q, t ; p = lower + 1 ; q = upper ; i = a[lower] ;

while ( q >= p )

{

while ( a[p] < i ) p++ ; while ( a[q] > i )

q-- ;

if ( q > p )

{

t = a[p] ; a[p] = a[q] ; a[q] = t ;

}

}

t = a[lower] ; a[lower] = a[q] ;

a[q] = t ; return

q ;

}

void quickSort(int a[],int lower, int upper)

{ int i ;

if ( upper > lower )

{

i = split ( a, lower, upper ) ; quickSort ( a, lower, i - 1 ) ;

quickSort ( a, i + 1, upper ) ;

}

}

EXECVE Command (Cheat No 3)

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> #include <string.h> void main(int argc, char \*argv[])

{

int val[10],ele; pid\_t pid; char\* cval[10]; char \*newenviron[] = {NULL }; int i,j,n,temp; printf("\nEnter the size for an array: ");

scanf("%d",&n); printf("\nEnter %d elements: ", n);

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

scanf("%d",&val[i]); printf("\nEntered

elements are: ");

for(i=0;i<n;i++) printf("\t%d",val[i]);

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

{

for(j=0;j<n-1;j++)

{

if(val[j]>val[j+1])

{

temp=val[j];

val[j]=val[j+1];

val[j+1]=temp;

}

}

}

printf("\nSorted elements are: "); for(i=0;i<n;i++) printf("\t%d",val[i]);

printf("\nEnter element to search: "); scanf("%d",&ele); val[i] = ele; for (i=0; i < n+1; i++) { char a[sizeof(int)];

snprintf(a, sizeof(int), "%d", val[i]); cval[i] = malloc(sizeof(a));

strcpy(cval[i], a);

}

cval[i]=NULL; pid=fork();

printf("\nstart process and its child process ID is :%d and parent id is: %d", getpid(),

getppid());

if(pid==0)

{

execve(argv[1], cval, newenviron); perror("Error in execve call...");

}

}

**Code: Child code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

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

{ int i,j,c,ele; int arr[argc];

for (j = 0; j < argc-1; j++)

{

int n=atoi(argv[j]);

arr[j]=n;

}

ele=atoi(argv[j] ); i=0; j=argc-1; c=(i+j)/2;

while(arr[c]!=ele && i<=j)

{

if(ele > arr[c]) i = c+1; else j = c-1; c = (i+j)/2;

} if(i<=j)

printf("\nElement Found in the given Array...!!!\n");

else

printf("\nElement Not Found in the given Array...!!!\n");

}

Shortest Job First

#include<stdio.h> int main()

{

int

bt[20],p[20],wt[20],tat[20],i,j,n,total=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;

}

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

{

wt[i]=0;

for(j=0;j<i;j++) wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=(float)total/n;

total=0; printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time"); for(i=0;i<n;i++)

{

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

total+=tat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

avg\_tat=(float)total/n;

printf("\n\nAverage Waiting Time=%f",avg\_wt); printf("\nAverage Turnaround Time=%f\n",avg\_tat);

}

Round robin With Different Arrival Time

#include<stdio.h>

#include <sys/types.h> #include <sys/wait.h> int main()

{

int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];

float avg\_wt, avg\_tat;

printf(" Total number of process in the system: ");

scanf("%d", &NOP); y = NOP;

for(i=0;i<NOP; i++)

{

printf("\n Enter the Arrival and Burst time of the Process[%d]\n",

i+1); printf(" Arrival time is: \t"); scanf("%d", &at[i]); printf("

\nBurst time is: \t"); scanf("%d", &bt[i]); temp[i] = bt[i];

}

printf("Enter the Time Quantum for the process: \t");

scanf("%d", &quant);

printf("\n Process No\t\t Burst Time \t\t TAT \t\t Waiting Time ");

for(sum=0, i = 0; y!=0; )

{

if(temp[i] <= quant && temp[i] > 0)

{

sum = sum + temp[i];

temp[i] = 0;

count=1;

}

else if(temp[i] > 0)

{

temp[i] = temp[i] - quant; sum = sum + quant;

}

if(temp[i]==0 && count==1)

{ y--

;

printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum- at[i]-

bt[i]); wt = wt+sum-at[i]-bt[i]; tat = tat+sum-at[i]; count =0;

}

if(i==NOP-1)

{

i=0;

}

else if(at[i+1]<=sum)

{

i++;

}

else

{

i=0;

}

}

avg\_wt = wt \* 1.0/NOP;

avg\_tat = tat \* 1.0/NOP;

printf("\n Average Turn Around Time: \t%f",

avg\_wt); printf("\n Average Waiting Time: \t%f", avg\_tat);

}

Thread Synchronization Using Counting Semaphore(Producer , Consumer Problem)

#include<stdio.h>

#include<semaphore.h>

#include<sys/types.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>

#define BUFFER\_SIZE

10 pthread\_mutex\_t mutex; sem\_t empty,full; int buffer[BUFFER\_SIZE];

int counter; pthread\_t

tid;

void \*producer(); void \*consumer(); void insert\_item(int);

int remove\_item(); void initilize()

{

pthread\_mutex\_init(&mutex,NULL); sem\_init(&full,0,0); sem\_init(&empty,0,BUFFER\_SIZE);

}

void \*producer()

{

int item,wait\_time; wait\_time=rand()%5; sleep(wait\_time)%5; item=rand()%10; sem\_wait(&empty); pthread\_mutex\_lock(&mutex); printf("Producer produce

%d\n\n",item); insert\_item(item); pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

}

void \*consumer()

{

int item,wait\_time;

wait\_time=rand()%5;

sleep(wait\_time); sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

item=remove\_item(); printf("Consumer consume %d\n\n",item); pthread\_mutex\_unlock(&mutex); sem\_post(&empty);

}

void insert\_item(int item)

{

buffer[counter++]=item;

}

int remove\_item()

{

return buffer[--counter];

}

int main()

{

int n1,n2; int i; printf("Enter number of Producers: "); scanf("%d",&n1); printf("Enter number of Consumers: ");

scanf("%d",&n2);

initilize(); for(i=0;i<n1;i++)

pthread\_create(&tid,NULL,producer,NULL); for(i=0;i<n2;i++)

pthread\_create(&tid,NULL,consumer,NULL); sleep(5);

exit(0);

}

Thread Synchrnization [Reader , Write Problem]

#include<stdio.h>

#include<pthread.h>

#include<unistd.h>

pthread\_mutex\_t wr, mutex; int a=10, readcount=0; void \* reader(void \*arg){ long int num; num=(long int) arg;

pthread\_mutex\_lock(&mutex

); readcount++; pthread\_mutex\_unlock(&mutex); if(readcount==1) pthread\_mutex\_lock(&wr);

printf("Reader %ld is in critical section", num); printf("Reader %ld is reading data %d", num a); sleep(1); pthread\_mutex\_lock(&mutex); readcount--;

pthread\_mutex\_unlock(&mutex); if(readcount==0) pthread\_mutex\_lock(&wr);

printf("\n Reader %ld left the critical section", num);

}

void \* writer (void \*arg){ long int num; num=(long int) arg; pthread\_mutex\_lock(&wr); printf("\n Writer %ld is in critical section", num); printf("\n Writer %ld has written dataas %d", num

++a); sleep(1);

pthread\_mutex\_unlock(&wr);

printf("\n Writer %ld is in critical section", num);

}

int main() { pthread\_t

r[10],w[10]; long int

i,j;

int nor, now;

pthread\_mutex\_init(&wr,NULL); pthread\_mutex\_init(&mutex,NULL); printf("Enter number of reader and writter"); scanf("%d %d",&nor,&now); for(i=0;i<nor;i++)

{

pthread\_create(&r[i],NULL,reader,(void \*)i); }

for(j=0;j<now;j++)

{

pthread\_create(&w[j],NULL,reader,(void \*)j);

}

for(i=0;i<nor;i++)

{

pthread\_join(r[i],NULL);

}

for(j=0;j<now;j++)

{

pthread\_join(w[j],NULL);

}

return 0;

}

Deadlock Avoidance Bankers Algorithm :

#include <stdio.h> int main()

{ int n, m, i, j, k; n = 5; m = 3;

int alloc[5][3] = { { 0, 1, 0 },

{ 2, 0, 0 },

{ 3, 0, 2 },

{ 2, 1, 1 }, { 0, 0, 2 } };

int max[5][3] = { { 7, 5, 3 },

{ 3, 2, 2 },

{ 9, 0, 2 },

{ 2, 2, 2 }, { 4, 3, 3 } }; int avail[3] = { 3, 3, 2

};

int f[n], ans[n], ind = 0; for (k = 0; k < n; k++) { f[k] = 0;

}

int need[n][m];

for (i = 0; i < n; i++) { for (j = 0; j < m; j++)

need[i][j] = max[i][j] - alloc[i][j];

}

int y = 0; for (k = 0; k < 5; k++) { for (i = 0; i < n; i++) { if (f[i] == 0) {

int flag = 0;

for (j = 0; j < m; j++) { if (need[i][j] > avail[j]){ flag = 1; break;

}

}

if (flag == 0) { ans[ind++] = i; for (y = 0; y < m; y++) avail[y] += alloc[i][y];

f[i] = 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);

}

FCFS [Page Replacement Algorithm]

#include <stdio.h> int main() { int referenceString[10], pageFaults = 0, m, n, s, pages, frames; printf("\nEnter the number of Pages:\t"); scanf("%d", & pages); printf("\nEnter reference string values:\n"); for (m = 0; m < pages; m++)

{

printf("Value No. [%d]:\t", m + 1); scanf("%d", & referenceString[m]);

}

printf("\n What are the total number of frames:\t");

{ scanf("%d", & frames);

}

int temp[frames]; for (m = 0; m < frames; m++) { temp[m]

= -1;

}

for (m = 0; m < pages; m++)

{ s = 0; for (n = 0; n < frames; n++)

{

if (referenceString[m] == temp[n])

{ s++; pageFaults--;

}

}

pageFaults++;

if ((pageFaults <= frames) && (s == 0))

{

temp[m] = referenceString[m];

}

else if (s == 0)

{

temp[(pageFaults - 1) % frames] = referenceString[m];

}

printf("\n"); for (n = 0; n < frames; n++)

{

printf("%d\t", temp[n]);

}

}

printf("\nTotal Page Faults:\t%d\n", pageFaults);

return 0;

}

LRU :

#include<stdio.h>

int findLRU(int time[], int n)

{

int i, minimum = time[0], pos = 0; for (i = 1; i < n; ++i) {

if (time[i] < minimum)

{

minimum = time[i];

pos = i;

}

}

return pos;

}

int main() {

int no\_of\_frames, no\_of\_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i,

j, pos, faults = 0;

printf("Enter number of frames: "); scanf("%d", & no\_of\_frames); printf("Enter number of pages: ");

scanf("%d", & no\_of\_pages); printf("Enter reference string: ");

for (i = 0; i < no\_of\_pages; ++i)

{

scanf("%d", & pages[i]);

}

for (i = 0; i < no\_of\_frames; ++i)

{

frames[i] = -1;

for (i = 0; i < no\_of\_pages; ++i)

{

flag1 = flag2 = 0; for (j = 0; j < no\_of\_frames; ++j)

{

if (frames[j] == pages[i])

{

counter++; time[j] = counter; flag1 = flag2 = 1; break;

}

}

if (flag1 == 0) { for (j = 0; j < no\_of\_frames; ++j)

{

if (frames[j] == -1)

{

counter++; faults++; frames[j] = pages[i]; time[j]

= counter; flag2 =

1; break;

}

}

}

if (flag2 == 0) { pos = findLRU(time, no\_of\_frames); counter++; faults++; frames[pos] = pages[i]; time[pos] = counter;

}

printf("\n"); for (j = 0; j < no\_of\_frames; ++j)

{

printf("%d\t", frames[j]);

}

}

printf("\n\nTotal Page Faults = %d", faults); return 0;

Optimal For Frame Size As Minimum Tree :

#include<stdio.h> int main() {

int no\_of\_frames, no\_of\_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j,

k, pos, max, faults = 0;

printf("Enter number of frames: "); scanf("%d", & no\_of\_frames); printf("Enter number of pages: "); scanf("%d", & no\_of\_pages); printf("Enter page reference string: ");

for (i = 0; i < no\_of\_pages; ++i)

{

scanf("%d", & pages[i]);

}

for (i = 0; i < no\_of\_frames; ++i)

{

frames[i] = -1;

}

for (i = 0; i < no\_of\_pages; ++i)

{

flag1 = flag2 = 0; for (j = 0; j <

no\_of\_frames; ++j)

{

if (frames[j] == pages[i])

{

flag1 = flag2 = 1; break; }

}

if (flag1 == 0)

{

for (j = 0; j < no\_of\_frames; ++j)

{

if (frames[j] == -1)

{

faults++; frames[j] = pages[i]; flag2 =

1; break;

}

}

if (flag2 == 0)

{

flag3 = 0; for (j = 0; j < no\_of\_frames; ++j)

{

temp[j] = -1; for (k = i + 1; k < no\_of\_pages; ++k)

{ if (frames[j] == pages[k])

{

temp[j] = k; break; }

}

}

for (j = 0; j < no\_of\_frames; ++j)

{

if (temp[j] == -1)

{

pos = j; flag3 = 1; break;

}

}

if (flag3 == 0)

{

max = temp[0]; pos = 0; for (j = 1; j < no\_of\_frames; ++j)

{

if (temp[j] > max)

{

max = temp[j]; pos = j;

}

}

}

frames[pos] = pages[i]; faults++;

}

printf("\n"); for (j = 0; j < no\_of\_frames; ++j)

{

printf("%d\t", frames[j]);

}

}

printf("\n\nTotal Page Faults = %d", faults);

return 0;

}

**Full duplex communication between two independent processes.**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h> #define max\_buf 100 int main()

{

char \* myfifo1 = "myfifo1", \* myfifo2 = "myfifo2"; char buf[50]; FILE \* fp; int i = 0, words = 0, lines = 0;

mkfifo(myfifo2, 0777);

int fd, fd1;

fd = open(myfifo1, O\_RDWR);

read(fd, buf, max\_buf);

printf("\nMessage received is: %s", buf);

while (buf[i] != '\0')

{

while (buf[i] == ' ')

{ words++, i++;

}

if (buf[i] == '.' || buf[i] == '?' || buf[i] == '!')

{

lines++, i++;

}

i++;

}

printf("\n Total no. of characters:%d", i);

fp = fopen("abc.txt", "w+"); fprintf(fp, "Total characters=%d", i); printf("\n Total no. of words:%d", words); fp = fopen("abc.txt", "w+");

fprintf(fp, "Total characters=%d", words);

printf("\n Total no. of characters:%d", lines); fp = fopen("abc.txt", "w+");

fprintf(fp, "Total no. of lines=%d", lines); fclose(fp);

unlink(myfifo1);

fd1 = open(myfifo2, O\_RDWR); write(fd1, & i, sizeof(i)); write(fd1, & words, sizeof(words)); write(fd1, & lines, sizeof(lines)); close(fd1); return 0;

}

**. Inter-process Communication using Shared Memory using System**

#include <sys/ipc.h>

#include <sys/shm.h>

#define PROJECT\_ID 220

#define READ\_BY\_CLIENT 0

#define WRITTEN\_BY\_SERVER 1 #define ARRAY\_LENGTH 5

typedef struct SharedMemory {

int status;

int array[ARRAY\_LENGTH];

}

SharedMemory; key\_t getKey() { return ftok(".", PROJECT\_ID);

} int shm\_init() { return shmget(getKey(), sizeof(SharedMemory),

IPC\_CREAT | 0666);

}

SharedMemory \* attach(int shm\_id) { return (SharedMemory \* ) shmat(shm\_id, NULL, 0);

}

int detach(SharedMemory \* shm) {

return shmdt((void \* ) shm);

}

**// reader.c**

#include <stdio.h>

#include <stdlib.h> #include "SharedMemory.c" int main() { int shm\_id, i; if ((shm\_id = shm\_init()) == -1) { perror("Error occured while initialising Shared Memory\n"); exit(-1);

}

SharedMemory \* mSharedMemory = attach(shm\_id); if

(mSharedMemory -> status == READ\_BY\_CLIENT) {

printf("Server hasn't written value yet\n"); exit(-1);

}

printf("Printing %d Numbers\n", ARRAY\_LENGTH); for (i = 0; i < ARRAY\_LENGTH; i++) { printf("%d\n", mSharedMemory -> array[i]);

}

mSharedMemory -> status = READ\_BY\_CLIENT; if (detach(mSharedMemory) == -1) { perror("Error occured while detaching Shared memory\n"); exit(-1);

}

}

**//Writer.cc**

#include <stdio.h>

#include <stdlib.h> #include "SharedMemory.c" int main() { int shm\_id, i; if ((shm\_id = shm\_init()) == -1) { perror("Error occured while initialising Shared Memory\n"); exit(-1);

}

SharedMemory \* mSharedMemory = attach(shm\_id); if (mSharedMemory -> status == WRITTEN\_BY\_SERVER) {

printf("Client hasn't read value yet\n");

exit(-1);

}

printf("Enter %d Numbers\n", ARRAY\_LENGTH); for (i = 0; i < ARRAY\_LENGTH; i++) { scanf("%d", & mSharedMemory -> array[i]);

}

mSharedMemory -> status = WRITTEN\_BY\_SERVER; if (detach(mSharedMemory) == -1) { perror("Error occured while detaching Shared memory\n"); exit(-1);

} char c; printf("Press any key to exit\n"); scanf(" %c", &

c); } Disk Sceduling : SSTF

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

int main()

{

int RQ[100], i, n, TotalHeadMoment = 0, initial, count = 0; printf("Enter the number of Requests\n"); scanf("%d", & n); printf("Enter the Requests sequence\n"); for (i = 0; i < n; i++) scanf("%d", & RQ[i]); printf("Enter initial head position\n"); scanf("%d", & initial); while (count != n)

{

int min = 1000, d, index; for (i = 0; i < n; i++)

{

d = abs(RQ[i] - initial);

if (min > d)

{

min = d; index = i;

}

}

TotalHeadMoment = TotalHeadMoment + min;

initial = RQ[index];

RQ[index] = 1000; count++;

}

printf("Total head movement is %d", TotalHeadMoment); return 0;

}

Scan Algorithm :

#include<stdio.h> int main()

{

int i, j, sum = 0, n; int d[20]; int disk; int temp, max; int dloc; printf("enter number of location\t"); scanf("%d", & n); printf("enter position of head\t"); scanf("%d", & disk);

printf("enter elements of disk queue\n"); for (i = 0; i < n; i++)

{

scanf("%d", & d[i]);

}

d[n] = disk; n = n + 1; for (i = 0; i < n; i++)

{ for (j = i; j < n; j++) { if (d[i] > d[j]) { temp = d[i]; d[i] = d[j]; d[j] = temp;

}

}

}

max = d[n];

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

{

if (disk == d[i]) {

dloc = i;

break;

}

}

for (i = dloc; i >= 0; i--)

{

printf("%d -->", d[i]);

}

printf("0 -->");

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

{

printf("%d-->", d[i]);

}

sum = disk + max;

printf("\nmovement of total cylinders %d", sum);

return 0;

}

C – LOOK Disk Sceduling :

#include<stdio.h> #include<stdlib.h> int main() {

int RQ[100], i, j, n, TotalHeadMoment = 0, initial, size,

move; printf("Enter the number of Requests\n"); scanf("%d", & n);

printf("Enter the Requests sequence\n"); for (i = 0; i < n; i++)

scanf("%d", & RQ[i]);

printf("Enter initial head

position\n"); scanf("%d", & initial); printf("Enter total disk size\n"); scanf("%d", & size);

printf("Enter the head movement direction for high 1 and for low 0 \n");

scanf("%d", & move);

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

{ for (j = 0; j < n - i - 1; j++)

{

if (RQ[j] > RQ[j + 1])

{

int temp; temp = RQ[j]; RQ[j] = RQ[j + 1];

RQ[j + 1] = temp;

}

}

}

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

{ if (initial < RQ[i])

{

index = i; break;

}

}

if (move == 1)

{

for (i = index; i < n; i++)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial); initial = RQ[i];

}

for (i = 0; i < index; i++)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial); initial = RQ[i];

}

}

else { for (i = index - 1; i >= 0; i--)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial); initial = RQ[i];

}

for (i = n - 1; i >= index; i--)

{

TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial); initial = RQ[i];

}

}

printf("Total head movement is %d", TotalHeadMoment); return 0;

}