BUBT | BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Committed to Academic Excellence

Lab Assignment-3

Course Code: CSE 310

Course Title: Operating Systems

Submitted to:	Submitted by:
Name: Suman Saha	Name: Syeda Nowshin Ibnat
Assistant Professor	ID: 17183103020
Dept. of CSE	Intake: 39
at Bangladesh University of Business and Technology.	Section: 01
	Program: B.Sc. in CSE
	Semester: Fall 20-21

Date of Submission: 19/03/2021

1 No question Answer

Q: Write down a shell program to implement Shortest Job First Algorithm.

Source code:

```
#!/bin/bash
clear ;
n=0
#enter number of process
read -p "Echo number of process: " n;
declare -a bt[n];
declare -a p[n];
declare -a wt[n];
declare -a tat[n];
bt[0]=0
# get burst time
echo "Enter Burst time: ";
for((i=0; i<n; i++)); do
((N=i+1))
read -p "process $N: "  bt[i];
(( p[$i]=i+1 ))
done
# echo "printing the bt[] array: ";
# declare -p bt
# echo ;
# echo ;
btn=( $( printf "%s\n" "${bt[@]}" | sort -n ) )
```

```
(( wt[0]=0 ))
for (( i = 1; i < n; i++ )); do
(( wt[i]=0 ))
for(( j = 0; j < i; j++ )); do
(( wt[i]+=btn[j] ))
done
(( total+=wt[i] ))
done
(( avg_wt=total/n ))
(( avg_wt+=total%n ))
(( total = 0))
echo "Process    Burst Time    Waiting Time Turnaround Time";
for(( i = 0; i < n; i++ )) {
(( tat[i]=btn[i]+wt[i] ))
(( total+=tat[i] ))
echo "${p[i]}           ${btn[i]}              ${wt[i]}                ${tat[i]}" }
(( avg_tat=total/n ))
(( avg_tat+=total%n ))
echo "Average Waiting Time= " $avg_wt;
echo "Average Turnaround Time= " $avg_tat;
echo ;
echo ;
# read -p "Enter Burst time: " burst;
```

2 No question Answer

Q: Write down a program to implement the Reader Writers Problem.

Source code:

```
/*

This program provides a possible solution for first readers writers problem using mutex and semaphore.

I have used 10 readers and 5 producers to demonstrate the solution. We can always play with these values.

*/

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

sem_t wrt;

pthread_mutex_t mutex;

int cnt = 1;

int numreader = 0;

void *writer(void *wno)

{  sem_wait(&wrt);

cnt = cnt*2;

printf("Writer %d modified cnt to %d\n",(*((int *)wno)),cnt);

sem_post(&wrt); }

void *reader(void *rno) {

// Reader acquire the lock before modifying numreader

pthread_mutex_lock(&mutex);

numreader++;

if(numreader == 1) {
```

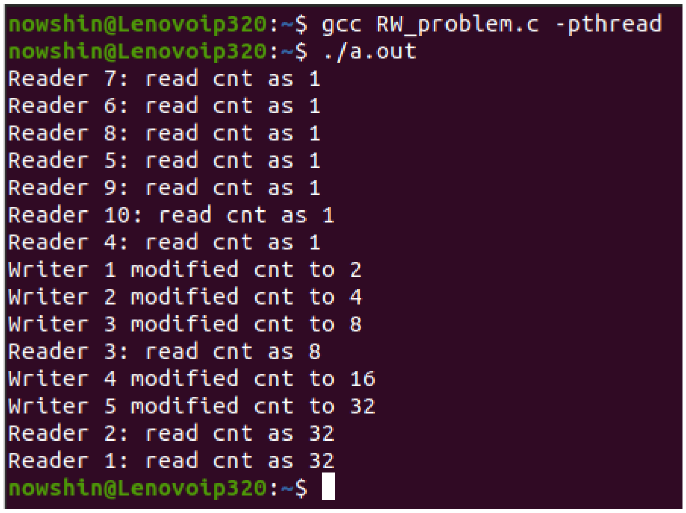
```
sem_wait(&wrt); // If this id the first reader, then it will block the writer }
pthread_mutex_unlock(&mutex);
// Reading Section
printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);
// Reader acquire the lock before modifying numreader
pthread_mutex_lock(&mutex);
numreader--;
if(numreader == 0) {
sem_post(&wrt); // If this is the last reader, it will wake up the writer. }
pthread_mutex_unlock(&mutex); }
int main() {
pthread_t read[10],write[5];
pthread_mutex_init(&mutex, NULL);
sem_init(&wrt,0,1);
int a[10] = {1,2,3,4,5,6,7,8,9,10}; //Just used for numbering the producer and consumer
for(int i = 0; i < 10; i++) {
pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]); }
for(int i = 0; i < 5; i++) {
pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]); }
for(int i = 0; i < 10; i++) {
pthread_join(read[i], NULL); }
for(int i = 0; i < 5; i++) {
pthread_join(write[i], NULL); }

pthread_mutex_destroy(&mutex);
```

sem_destroy(&wrt);

return 0; }

Sample Output:



```
nowshin@Lenovoip320:~$ gcc RW_problem.c -pthread
nowshin@Lenovoip320:~$ ./a.out
Reader 7: read cnt as 1
Reader 6: read cnt as 1
Reader 8: read cnt as 1
Reader 5: read cnt as 1
Reader 9: read cnt as 1
Reader 10: read cnt as 1
Reader 4: read cnt as 1
Writer 1 modified cnt to 2
Writer 2 modified cnt to 4
Writer 3 modified cnt to 8
Reader 3: read cnt as 8
Writer 4 modified cnt to 16
Writer 5 modified cnt to 32
Reader 2: read cnt as 32
Reader 1: read cnt as 32
nowshin@Lenovoip320:~$ 
```

3 No question Answer

Q: Write down a program to implement the Segmentation memory management technique and calculate external fragmentation occur there.

Source code:

#include<stdio.h>

#include<stdlib.h>

struct list {

int seg;

int base;

int limit;

struct list *next;

} *p;

void insert(struct list *q,int base,int limit,int seg) {

if(p==NULL) {

p=malloc(sizeof(Struct list));p->limit=limit;

```
p->base=base;
p->seg=seg;
p->next=NULL; }
else {
while(q->next!=NULL)  {
Q=q->next;
Printf(“yes”) }
q->next=malloc(sizeof(Struct list));
q->next ->limit=limit;
q->next ->base=base;
q->next ->seg=seg;
q->next ->next=NULL; } }
int find(struct list *q,int seg) {
while(q->seg!=seg) {
q=q->next; }
return q->limit; }
int search(struct list *q,int seg) {
while(q->seg!=seg) {
q=q->next; }
return q->base; }
main() {
p=NULL;
int seg,offset,limit,base,c,s,physical;
printf("Enter segment table/n");
printf("Enter -1 as segment value for termination\n"); do {
```

```
printf("Enter segment number");
scanf("%d",&seg);
if(seg!=-1) {
printf("Enter base value:");
scanf("%d",&base);
printf("Enter value for limit:");
scanf("%d",&limit);
insert(p,base,lmit,seg); } }
while(seg!=-1)
printf("Enter offset:");
scanf("%d",&offset);
printf("Enter bsegmentation number:");
scanf("%d",&seg);
c=find(p,seg);
s=search(p,seg);
if(offset<c) {
physical=s+offset;
printf("Address in physical memory %d\n",physical); }
else {
printf("error"); }}
```

4 No question Answer

Q: Write down a program to implement the Paging memory management technique and calculate internal fragmentation occur there.

Source code:

```
#include<stdio.h>

#include<math.h>

#define MAX 50

int main(){

int page[MAX],i,n,f,ps,off,pno,ProcessSize, internalFrag;

int choice=0;

printf("\nEnter page size: ");

scanf("%d",&ps);

printf("\nEnter process size: ");

scanf("%d",&ProcessSize);

n = (ceil(ProcessSize / (ps*1.0))) ;

internalFrag = ps - (ProcessSize % ps);

printf("Internal fragmentation: \n");

printf("\nEnter no of frames: ");

scanf("%d",&f);

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

page[i]=-1;

printf("\nEnter the page table\n");

printf("(Enter frame no as -1 if that page is not present in any frame)\n\n");

printf("\npageno\tframeno\n-------\t-------");

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

printf("\n\n%d\t\t",i);
scanf("%d",&page[i]); }
do{
printf("\n\nEnter the logical address(i.e,page no & offset):");
scanf("%d%d",&pno,&off);
if(page[pno]==-1)
printf("\n\nThe required page is not available in any of frames");
else
printf("\n\nPhysical address(i.e,frame no & offset):%d,%d",page[pno],off);
printf("\nDo you want to continue(1/0)?:");
scanf("%d",&choice);}
while(choice==1);
return 0; }
```

5 No question Answer

Q: Write down a program to implement optimal page replacement algorithm.

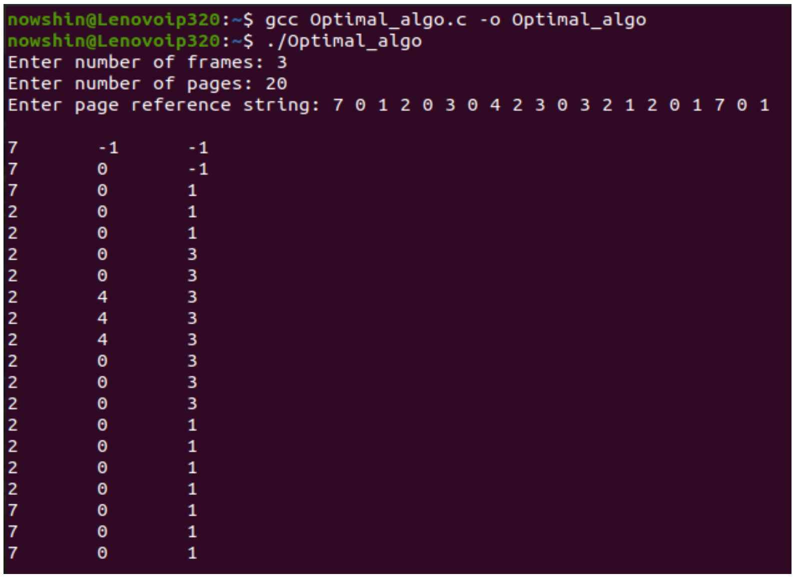
Source Code:

```
#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; }
```

Sample Output:



```
nowshin@Lenovoip320:~$ gcc Optimal_algo.c -o Optimal_algo
nowshin@Lenovoip320:~$ ./Optimal_algo
Enter number of frames: 3
Enter number of pages: 20
Enter page reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7       -1      -1
7       0       -1
7       0       1
2       0       1
2       0       1
2       0       3
2       0       3
2       4       3
2       4       3
2       4       3
2       0       3
2       0       3
2       0       3
2       0       1
2       0       1
2       0       1
2       0       1
7       0       1
7       0       1
7       0       1
```