CS 2257 - OPERATING SYSTEMS LABORATORY

LABORATORY MANUAL

FOR IV SEMESTER B.E / CSE

ACADEMIC YEAR: 2013 - 2014

(FOR PRIVATE CIRCULATION ONLY)

ANNA UNIVERSITY, CHENNAI

NAME :…………………………………………

REG.NO :…………………………………………

BATCH :…………………………………………

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DR.NAVALAR NEDUNCHEZHIAYN COLLEGE OF ENGINEERING,

THOLUDUR-606303, CUDDALORE DIST.

GENERAL INSTRUCTIONS FOR LABORATORY CLASSES

DO'S

- Without Prior permission do not enter into the Laboratory.
- While entering into the LAB students should wear their ID cards.
- The Students should come with proper uniform.
- Students should sign in the LOGIN REGISTER before entering into the laboratory.
- Students should come with observation and record note book to the laboratory.
- Students should maintain silence inside the laboratory.
- After completing the laboratory exercise, make sure to shutdown the system properly.

DONT'S

- Students bringing the bags inside the laboratory..
- Students wearing slippers/shoes insides the laboratory.
- Students using the computers in an improper way.
- Students scribbling on the desk and mishandling the chairs.
- Students using mobile phones inside the laboratory.
- Students making noise inside the laboratory.

HARDWARE REQUIREMENTS:

Processors	-	2.0 GHz or Higher
RAM	-	256 MB or Higher
Hard Disk	-	20 GB or Higher

SOFTWARE: Linux:

Ubuntu / OpenSUSE / Fedora / Red Hat / Debian / Mint OS

Linux could be loaded in individual PCs.

UNIVERSITY PRACTICAL EXAMINATION

ALLOTMENT OF MARKS

Internal assessment	**-**	**20 marks**
Practical assessment	**-**	**80 marks**
Total	**-**	**100 marks**

INTERNAL ASSESSMENT (20 marks)

Staff should maintain the assessment Register and the Head of the Department should monitor it.

SPLIT UP OF INTERNAL MARKS

Record Note	**-**	**10 marks**
Model Exam	**-**	**5 marks**
Attendance	**-**	**5 marks**
Total	**-**	**20 marks**

UNIVERSITY EXAMINATION

The exam will be conducted for 100 marks. Then the marks will be calculated to 80 marks.

SPLIT UP OF PRACTICAL EXAMINATION MARKS

Aim and Algorithm	**-**	**20 marks**
Program	**-**	**40 marks**
Output	**-**	**20 marks**
Result	**-**	**10 marks**
Viva-voce	**-**	**10 marks**
Total	**-**	**100 marks**

ANNA UNIVERSITY CHENNAI

CS 2257 – OPERATING SYSTEMS LABORATORY

(Implement the following on LINUX or other Unix like platform. Use C for high level language implementation)

LIST OF EXPERIMENTS

1. Write programs using the following system calls of UNIX operating system:

   Fork, exec, getpid, exit, wait, close, stat, opendir, readdir
2. Write programs using the I/O system calls of UNIX operating system (open, read, write, etc)
3. Write C programs to simulate UNIX commands like ls, grep, etc.
4. Given the list of processes, their CPU burst times and arrival times, display/print the Gantt chart for FCFS and SJF. For each of the scheduling policies, compute and print the average waiting time and average turnaround time. (2 sessions)
5. Given the list of processes, their CPU burst times and arrival times, display/print the Gantt chart for Priority and Round robin. For each of the scheduling policies, compute and print the average waiting time and average turnaround time. (2 sessions)
6. Developing Application using Inter Process Communication (using shared memory, pipes or message queues)
7. Implement the Producer – Consumer problem using semaphores (using UNIX system calls).
8. Implement some memory management schemes – I
9. Implement some memory management schemes – II
10. Implement any file allocation technique (Linked, Indexed or Contiguous)

Total: 45

Example for exercises 8 and 9:

Free space is maintained as a linked list of nodes with each node having the starting byte address and the ending byte address of a free block. Each memory request consists of the process-id and the amount of storage space required in bytes. Allocated memory space is again maintained as a linked list of nodes with each node having the process-id, starting byte address and the ending byte address of the allocated space. When a process finishes (taken as input) the appropriate node from the allocated list should be deleted and this free disk space should be added to the free space list. [Care should be taken to merge contiguous free blocks into one single block. This results in deleting more than one node from the free space list and changing the start and end address in the appropriate node]. For allocation use first fit, worst fit and best fit.

CONTENTS

Exercise Number: 1

Title of the Exercise : PROCESS SYSTEM CALLS

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write c program to implement the Process system calls.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program.
2	Declare the pid and get the pid by using the getpid() method.
3	Create a child process by calling the fork() system call.
4	Check if(pid==0) then print the child process id and if(pid==1) then print the parent process value. Otherwise print fork failed.
5	Check if pid==0 then execute the ‘ls’ command and display the list of files and dir.

c) Program

```
#include<stdio.h>
#include<sys/types.h>
void main(){
pid_t pid;
pid =getpid();
printf("before fork %d",pid);
pid=fork();
if(pid==0)
printf("\nthisline from child,\nthe child process id %d\n",getpid());
else if(pid==1)
printf("this line is from parent,value=%d\n");
else if(pid<1){
printf("\n fork failed");
exit(1);
}
if(pid==0){
execl("/bin/ls","ls","-1",(char *)0);
}
if(pid>0)
wait((int *)0)
}
```

d) Output:

```
before fork 4258
thisline from child,
the child process id 4259
a.out
ch.c
come.c
data
file1.c
file2.c
first.c
fit.c
sysytem.c
before fork 4258
```

e) Result:

Thus the process system call program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. What is an operating system?

Operating system is an important part of almost every computer system

2. What are the components present in computer?

A computer system can be divided roughly into four components, the operating system, the application programs, and the user.

3. What is symmetric multiprocessing?

Each processor runs an identical copy of the operating system, and these copies communicate with one another as needed.

Exercise Number: 2

Title of the Exercise : I/O SYSTEM CALLS

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a 'c' program for I/O system calls.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program.
2	pen a file for O_RDWR for R/W,O_CREATE for creating a file ,O_TRUNC for truncate a file.
3	Using getchar(), read the character and stored in the string[] array.
4	The string [] array is write into a file close it.
5	Then the first is opened for read only mode and read the characters and displayed it and close the file.
6	Stop the program.

c) Program

```
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
int main(){
int n,i=0;
int f1,f2;
char c,strin[100];
f1=open("data",O_RDWR|O_CREAT|O_TRUNC);
while((c=getchar())!='\n'){
strin[i++]=c;
}
strin[i]='\0';
write(f1,strin,i);
close(f1);
f2=open("data",O_RDONLY);
read(f2,strin,0);
printf("\n%s\n",strin);
close(f2);
return 0;
}
```

d) Output:

```
hai

hai
```

e) Result:

Thus the I/O system call program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. **List out the types in mainframe systems**
   i) Batch system
   ii) Multiprogrammed systems
   iii) Time-sharing system

2. **What is file-server systems?**
   File-server system provides a file system interface where clients can create, update, read, and delete files

3. **What is job scheduling?**
   If several jobs are ready to be brought in to memory, and if there is not enough room for all of them, then the system must choose among them. Making this decision is job scheduling.

Exercise Number: 3

Title of the Exercise : FIRST COME FIRST SERVE SCHEDULING
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write the program to implement CPU & scheduling algorithm for first come first serve scheduling.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program.
2	Get the number of processes and their burst time.
3	Initialize the waiting time for process 1 and 0.
4	The waiting time for the other processes are calculated as follows:
5	for(i=2;i<=n;i++),wt.p[i]=p[i-1]+bt.p[i-1].
6	The waiting time of all the processes is summed then average value time is calculated.
7	The waiting time of each process and average times are displayed.
8	Stop the program.

c) Program

```
#include<stdio.h>
struct process{
int pid;          int bt;          int wt,tt;
}p[10];
int main(){
int i,n,totwt,tottt,avg1,avg2;
clrscr();
printf("enter the no of process \n");
scanf("%d",&n);
for(i=1;i<=n;i++){
p[i].pid=i;
printf("enter the burst time \n");
scanf("%d",&p[i].bt);
}
p[1].wt=0;
p[1].tt=p[1].bt+p[1].wt;
i=2;
while(i<=n){
p[i].wt=p[i-1].bt+p[i-1].wt;
p[i].tt=p[i].bt+p[i].wt;
i++;
}
i=1;
totwt=tottt=0;
printf("\n processid \t bt\t wt\t tt\n");
while(i<=n){
printf("\n\t%d \t%d \t%d \t%d",p[i].pid,p[i].bt,p[i].wt,p[i].tt);
totwt=p[i].wt+totwt;
tottt=p[i].tt+tottt;
i++;
}
avg1=totwt/n;          avg2=tottt/n;
printf("\navg1=%d \t avg2=%d \t",avg1,avg2);
getch();
return 0;
}
```

d) Output:

```
enter the no of process        3
enter the burst time           2
enter the burst time           4
enter the burst time           6
Process sid     bt     wt     tt
          1     2      0      2
          2     4      2      6
          3     6      6      12
avg1=2                 avg2=6
```

e) Result:

Thus the FCFS program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. Define non-preemptive scheduling

First come first serve scheduling (fcfs)
Shortest job first scheduling

2. Define dispatcher and its functions?

i. Switching context ii.Switching to user mode
iii. Jumping to the proper location in the user program to restarts the program

3. Define dispatch latency?

The time taken for the dispatcher to stop one process and start another running process is known as dispatch latency.

4. What is mean by FCFS scheduling?

FCFS also known as First-in-first-out(FIFO)which is the simplest scheduling policy. arriving jobs are inserted into the tail(rear)of the ready queue.

Exercise Number: 4

Title of the Exercise : SHORTEST JOB FIRST SCHEDULING
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement cpu scheduling algorithm for shortest job first scheduling.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program. Get the number of processes and their burst time.
2	Initialize the waiting time for process 1 as 0.
3	The processes are stored according to their burst time.
4	The waiting time for the processes are calculated a follows: for(i=2;i<=n;i++).wt.p[i]=p[i=1]+bt.p[i-1].
5	The waiting time of all the processes summed and then the average time is calculated.
6	The waiting time of each processes and average time are displayed.
7	Stop the program.

c) Program

```
#include<stdio.h>
#include<conio.h>
struct process{
int pid;          int bt;          int wt;          int tt;
}p[10],temp;
int main(){
int i,j,n,totwt,tottt;
float avg1,avg2;          clrscr();
printf("\nEnter the number of process:\t");
scanf("%d",&n);
for(i=1;i<=n;i++){
p[i].pid=i;
printf("\nEnter the burst time:\t");
scanf("%d",&p[i].bt);
}
for(i=1;i<n;i++){
for(j=i+1;j<=n;j++){
if(p[i].bt>p[j].bt){
temp.pid=p[i].pid;               p[i].pid=p[j].pid;
p[j].pid=temp.pid;               temp.bt=p[i].bt;
p[i].bt=p[j].bt;                 p[j].bt=temp.bt;
}}}
p[1].wt=0;                       p[1].tt=p[1].bt+p[1].wt;
i=2;
while(i<=n){
p[i].wt=p[i-1].bt+p[i-1].wt;       p[i].tt=p[i].bt+p[i].wt;
i++;
}
i=1;
totwt=tottt=0;
printf("\nProcess id \tbt \twt \ttt");
while(i<=n){
printf("\n\t%d \t%d \t%d \t%d\n",p[i].pid,p[i].bt,p[i].wt,p[i].tt);
totwt=p[i].wt+totwt;
tottt=p[i].tt+tottt;
i++;
}
avg1=totwt/n;                avg2=tottt/n;
printf("\nAVG1=%f\t AVG2=%f",avg1,avg2);
getch();             return 0;
}
```

Output:

```
enter the number of process    3
enter the burst time:  2
enter the burst time:  4
enter the burst time:  6
process    id      bt      wt      tt
           1        2       0       2
           2        4       2       6
           3        6       6       12
        AVG1=2.000000                  AVG2=6.000000
```

d) Result:

Thus the SJF program was executed and verified successfully

QUESTIONS AND ANSWERS

1. **Define CPU scheduling.**
   CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multi programmed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.
2. **What is a Dispatcher?**
   The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves: • Switching context • Switching to user mode
   • Jumping to the proper location in the user program to restart that program.
3. **What is turnaround time?**
   Turnaround time is the interval from the time of submission to the time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

Exercise Number: 5

Title of the Exercise : PRIORITY SCHEDULING
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a 'C' program to perform priority scheduling.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program.
2	Read burst time, waiting time, turn the around time and priority.
3	Initialize the waiting time for process 1 and 0.
4	Based up on the priority process are arranged.
5	The waiting time for other processes are calculated based on priority.
6	The waiting time of all the processes is summed and then the average waiting time is calculated.
7	The waiting time of each process and average waiting time are displayed based on the priority.
8	Stop the program.

c) Program

```
#include<stdio.h>
#include<conio.h>
struct process{
int pid;        int bt;        int wt;        int tt;        int prior;
}
p[10],temp;
int main(){
int i,j,n,totwt,tottt,arg1,arg2;
clrscr();
printf("enter the number of process");
scanf("%d",&n);
for(i=1;i<=n;i++){
p[i].pid=i;
printf("enter the burst time");
scanf("%d",&p[i].bt);
printf("\n enter the priority");
scanf("%d",&p[i].prior);
}
for(i=1;i<n;i++){
for(j=i+1;j<=n;j++){
if(p[i].prior>p[j].prior){
temp.pid=p[i].pid;
p[i].pid=p[j].pid;
p[j].pid=temp.pid;
temp.bt=p[i].bt;
p[i].bt=p[j].bt;
p[j].bt=temp.bt;
temp.prior=p[i].prior;
p[i].prior=p[j].prior;
p[j].prior=temp.prior;
}
}
}
p[i].wt=0;
p[1].tt=p[1].bt+p[1].wt;
i=2;
while(i<=n){
p[i].wt=p[i-1].bt+p[i-1].wt;
p[i].tt=p[i].bt+p[i].wt;
i++;
}
i=1;
totwt=tottt=0;
printf("\n process to \t bt \t wt \t tt");
while(i<=n){
printf("\n%d\t %d\t %d\t %d\t",p[i].pid,p[i].bt,p[i].wt,p[i].tt);
totwt=p[i].wt+totwt;
tottt=p[i].tt+tottt;
i++;
}
arg1=totwt/n;
arg2=tottt/n;
printf("\n arg1=%d \t arg2=%d\t",arg1,arg2);
getch();
return 0;
}
```

Output:

```
    enter the no of process:3
    enter the burst time:2
    enter the priority:3
    enter the burst time:4
    enter the priority:1
    enter the burst time:6
    enter the priority:2

            process to    bt    wt    tt
            1      4      0      4     4
            1      6      4      10    14
            1      2      10      12   22
  avg1=4
  avg2=8
```

d) Result:

Thus the priority scheduling program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. **What are privileged instructions?**
   Some of the machine instructions that may cause harm to a system are designated as privileged instructions. The hardware allows the privileged instructions to be executed only in monitor mode.
2. **How can a user program disrupt the normal operations of a system?**
   A user program may disrupt the normal operation of a system by
   • Issuing illegal I/O operations
   • by accessing memory locations within the OS itself
   • Refusing to relinquish the CPU
3. **How is the protection for memory provided?**
   The protection against illegal memory access is done by using two registers. The base register and the limit register. The base register holds the smallest legal physical address; the limit register contains the size of the range. The base and limit registers can be loaded only by the OS using special privileged instructions.

Exercise Number: 6

Title of the Exercise : ROUND ROBIN SCHEDULING

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement cpu scheduling for Round Robin Scheduling.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Get the number of process and their burst time.
2	Initialize the array for Round Robin circular queue as ‘0’.
3	The burst time of each process is divided and the quotients are stored on the round robin array.
4	According to the array value the waiting time for each process and the average time are calculated as line the other scheduling.
5	The waiting time for each process and average times are displayed.
6	Stop the program.

c) Program

```
#include<stdio.h>
#include<conio.h>
struct process{
int pid,bt,tt,wt;
};
int main(){
struct process x[10],p[30];
int i,j,k,tot=0,m,n;
float wttime=0.0,tottime=0.0,a1,a2;
clrscr();
printf("\nEnter the number of process:\t");
scanf("%d",&n);
for(i=1;i<=n;i++){
x[i].pid=i;
printf("\nEnter the Burst Time:\t");
scanf("%d",&x[i].bt);
tot=tot+x[i].bt;
}
printf("\nTotal Burst Time:\t%d",tot);
p[0].tt=0;
k=1;
printf("\nEnter the Time Slice:\t");
scanf("%d",&m);
for(j=1;j<=tot;j++){
for(i=1;i<=n;i++){
if(x[i].bt!=0){
p[k].pid=i;
if(x[i].bt-m<0){
p[k].wt=p[k-1].tt;
p[k].bt=x[i].bt;
p[k].tt=p[k].wt+x[i].bt;
x[i].bt=0;
k++;
}
else{
p[k].wt=p[k-1].tt;
p[k].tt=p[k].wt+m;
x[i].bt=x[i].bt-m;
k++;
}           }                     }
}
printf("\nProcess id \twt \ttt");
for(i=1;i<k;i++){
printf("\n\t%d \t%d \t%d",p[i].pid,p[i].wt,p[i].tt);
wttime=wttime+p[i].wt;
tottime=tottime+p[i].tt;
a1=wttime/n;
a2=tottime/n;
}
printf("\n\nAverage Waiting Time:\t%f",a1);
printf("\n\nAverage TurnAround Time:\t%f",a2);
getch();
return 0;
}
```

Output:

```
enter the no of process3

enter the burst time3

enter the burst time5

enter the burst time7

total burst time :     15
enter the time slice:  2

process id       wt      tt
   1              0      2
   2              2      4
   3              4      6
   1              6      7
```

```
2          7    9
3          9    11
2          11   12
3          12   14
3          14   15

avg waiting time:      21.666666
avg turnaround time:   26.666666
```

Result:

Thus the Round Robin scheduling program was executed and verified successfully.

QUESTIONS AND ANSWERS

a. **What is turnaround time?**
Turnaround time is the interval from the time of submission to the time of completion of a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

b. **Define race condition.**
When several process access and manipulate same data concurrently, then the outcome of the execution depends on particular order in which the access takes place is called race condition. To avoid race condition, only one process at a time can manipulate the shared variable.

c. **Define entry section and exit section.**
The critical section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section. The critical section is followed by an exit section. The remaining code is the remainder section

Exercise Number: 7

Title of the Exercise : PIPE PROCESSING
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program for pipe processing.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net,UNIX OS,	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program.
2	Declare the file pointer fp, f1, f2.
3	Read the choice.
4	In case ‘1’ open a file and display the list of files and directories.
5	In case ‘2’ open a file and ptr bin\pwd for read mode it displays the personnel working directory.
6	In case ‘3’ open a file in write mode.
7	In case ‘4’ Exit command is execute.
8	Stop the program.

c) Program

```
#include<stdio.h>
#include<sys/stat.h>
#include<string.h>
int main(){
FILE *fp,*f1,*f2;
int opt;
char line[100];
printf("enter ur choice\n");
scanf("%d",&opt);
switch(opt){
case 1:
if((fp=popen("/bin/ls","r"))==NULL){
printf("pipeline error");
}
printf("list command\n");
printf("*************\n");
while(fgets(line,80,fp)){
printf("%s",line);
}
break;
case 2:
printf("PWD comman\n");
printf("*********\n");
if((f1=popen("/bin/pwd","r"))==NULL){
printf("PWD function error\n");
}
while(fgets(line,80,f1)){
printf("%s",line);
}
break;
case 3:
printf("cat commond\n");
printf("***********\n");
if((f2=fopen("ch.c","w"))==NULL){
printf("Cat function error\n");
}
while(fgets(line,80,f2)){
printf("%s",line);
}
break;
case 4:
printf("exit\n");
printf("****\n");
exit(0);
break;
}
}
```

d) Output:

```
        enter the choice
1
list command
*********************
a.out
ch.c
come.c
data
file1.c
file2.c

enter the choice
2
PWD command
**********************
enter the choice
3
CAT command
```

```
**********************

enter the choice
4
exit
**********************
```

Result:

Thus the Pipe Processing program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. **What is logical address space and physical address space?**
   The set of all logical addresses generated by a program is called a logical address space; the set of all physical addresses corresponding to these logical addresses is a physical address space.
2. **What is the main function of the memory-management unit?**
   The runtime mapping from virtual to physical addresses is done by a hardware device called a memory management unit (MMU).
3. **Define swapping.**
   A process needs to be in memory to be executed. However a process can be swapped temporarily out of memory to a backing tore and then brought back into memory for continued execution. This process is called swapping.
4. **What do you mean by best fit?**
   Best fit allocates the smallest hole that is big enough. His entire list has to be searched, unless it is sorted by size. His strategy produces the smallest leftover hole.
5. **What do you mean by first fit?**
   First fit allocates the first hole that is big enough. Searching can either start at the beginning of the set of holes or where the previous first-fit search ended. Searching can be stopped as soon as a free hole that is big enough is found.

Exercise Number: 8

Title of the Exercise : IMPLEMENTATION OF PRODUCER CONSUMER PROBLEM USING SEMAPHORE

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To implement producer/consumer problem using semaphore.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Declare variable for producer & consumer as pthread-t-tid produce tid consume.
2	Declare a structure to add items, semaphore variable set as struct.
3	Read number of items to be produced and consumed.
4	Declare and define semaphore function for creation and destroy.
5	Define producer function.
6	Define consumer function.
7	Call producer and consumer.
8	Stop the execution.

c) Program

```
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#define NBUFF 10
int nitems;
struct{
int buff[NBUFF];
sem_t mutex,nempty,nstored;
}shared;
void *produce(void*);
void *consume(void*);
int main(int argc,char**argv){
pthread_t tid_produce,tid_consume;
if(argc!=2){
printf("usage:filename<nitems>");
return 0;
}
printf("\n\nproducer_consumer problem using semaphore \n");
printf("..............................................................................\n");
nitems=atoi(argv[1]);
sem_init(&shared.mutex,0,1);
sem_init(&shared.nempty,0,NBUFF);
sem_init(&shared.nstored,0,0);
pthread_setconcurrency(2);
pthread_create(&tid_produce,NULL,produce,NULL);
pthread_create(&tid_consume,NULL,consume,NULL);
pthread_join(tid_produce,NULL);
pthread_join(tid_consume,NULL);
sem_destroy(&shared.mutex);
sem_destroy(&shared.nempty);
sem_destroy(&shared.nstored);
}
void *produce(void *arg){
int i;
for(i=0;i<nitems;i++){
sem_wait(&shared.nempty);
sem_wait(&shared.mutex);
shared.buff[i%NBUFF]=i;
printf("\tproducer........");
printf("buff[%d]=%d\n\n",i,shared.buff[i%NBUFF]);
sem_post(&shared.mutex);
sem_post(&shared.nstored);
sleep(3);
}
return NULL;
}
void *consume(void *arg){
int i;
for(i=0;i<nitems;i++){
sem_wait(&shared.nstored);
sem_wait(&shared.mutex);
printf("\tconsumer............");
printf("buff[%d]=%d\n\n\n",i,shared.buff[i%NBUFF]);
sem_post(&shared.mutex);
sem_post(&shared.nempty);
sleep(3);
}
return NULL;
}
```

d) Output:

```
            Enter the value of n=4
        Enter the item=1
        Enter the item=2
        Enter the item=3
        Enter the item=5
```

```
Consumer item=1
Consumer item=2
Consumer item=3
Consumer item=5
```

Result:

Thus the producer consumer program was executed and verified successfully

QUESTIONS AND ANSWERS

1. **What is a semaphore? (NOV/DEC 08)**
   A semaphore 'S' is a synchronization tool which is an integer value that, apart from initialization, is accessed only through two standard atomic operations; wait and signal. Semaphores can be used to deal with the n process critical section problem. It can be also used to solve various synchronization problems. The classic definition of 'wait' wait (S) {while (S<=0) ; S-- ;} the classic definition of 'signal' signal (S)
   {S++ ;}
2. **Define busy waiting and spin lock. (APRIL/MAY 10)**
   When a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code. This is called as busy waiting and this type of semaphore is also called a spin lock, because the process while waiting for the lock.
3. **What is a thread? (Nov/DEC 07)**
   A thread otherwise called a lightweight process (LWP) is a basic unit of CPU utilization, it comprises of a thread id, a program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and operating system resources such as open files and signals.

Exercise Number: 9

Title of the Exercise : FIRST FIT AND BEST FIT
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To implement first fit, best fit algorithm for memory management.

REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program.
2	Get the segment size, number of process to be allocated and their corresponding sizes.
3	Get the options. If the option is ‘2’ call first fit function.
4	If the option is ‘1’ call best fit function. Otherwise exit.
5	For first fit, allocate the process to first possible segment which is free and set the slap as ‘1’. So that none of process to be allocated to segment which is already allocated and vice versa.
6	For best fit , do the following steps,
7	Sorts the segments according to their sizes.
8	Allocate the process to the segment which is equal to or slightly greater than the process size and set the flag as the ‘1’ .So that none of the process to be allocated to the segment which is already allocated and vice versa. Stop the program.

c) Program

```
#include<stdio.h>
#include<conio.h>
#define MAXSIZE 25
void printlayout(int[],int);
int firstfit(int[],int,int);
int bestfit(int[],int,int);
void main(){
int i,a[MAXSIZE],n,req,choice,pos,ch;
clrscr();
printf("How many segments");
scanf("%d",&n);
for(i=0;i<n;i++){
printf("Segment size");
scanf("%d",&a[i]);
}
loop:
printf("How much is your memory requirement");
scanf("%d",&req);
printf("\n1)Bestfit\n2)Firstfit\n3)Exit");
printf("\nEnter your choice");
scanf("%d",&choice);
switch(choice){
  case 1:
          pos=bestfit(a,n,req);
          break;
  case 2:
          pos=firstfit(a,n,req);
          break;
}
clrscr();
printf("\tBest fit and First fit algorithm\n");
printf("\t_________________________________\n\n");
printlayout(a,n);
printf("Your memory requiement is:%d\n\n",req);
printf("Alloted memory region is:%d\n\n",a[pos]);
a[pos]=0;
printf("Do you want to continue 1/0");
scanf("%d",&ch);
if(ch==1)
goto loop;
}
void printlayout(int a[],int n){
int i,j;
printf("\t\t Memory free list\n");
printf("\t\t__________________\n\n");
printf("\t\t|~~~~~~|\n");
for(i=0;i<n;i++){
if(a[i]!=0){
for(j=1;j<=(a[i]/100);j++);
printf("\t\t|      |\n");
printf("\t\t|  %d |\n",a[i]);
printf("\t\t|------|\n");
}                  }
printf("\n\n");
}
int firstfit(int a[],int n,int r){
int i;    for(i=0;i<n;i++)
if(a[i]>=r)       break;
return i;
}
int bestfit(int a[],int n,int r){
int b[MAXSIZE],i,j,temp,val;
for(i=0;i<n;i++)
b[i]=a[i];
for(i=0;i<n-1;i++)
for(j=i;j<n-1;j++)
if(b[i]>b[j]){   temp=b[i];      b[i]=b[j];       b[j]=temp;      }
for(i=0;i<n;i++)
if(b[i]>=r)
```

```
break;
val=b[i];
for(i=0;i<n;i++)
if(a[i]==val)
break;
return i;
}
```

d) Output:

```
            How many segments 3
Segment size 212
Segment size 6
Segment size 432
How much ur memory requirement 128

    1.  Best fit algorithm
    2.  First fit algorithm
    3.  exit
    Enter ur choice 2
        Best fit and first algorithm:
        ____________________
            Memory free list
            _______________
            [___________________]
            |  |
            |  |
            |2|2|
            _________________
            |6|
            _________________
            |  |
            |  |
            |  |
            |  |
            |432|
            ____________________
Your memory requirement is 128
allocated memory region is 212
Do you want to continue ..(1/0):1
How much ur memory requirement 128
    1.  Best fit algorithm
    2.  first fit algorithm
    3.  Exit
Enter ur choice :1
        Bestfit and firstfit algorthm:
        ______________________
            Memory free is list
            ___________________
            [_________________________]
            |6|
            _______________
            | |
            | |
                | |
                | |
                |432|
                ______________
UR memory requirement is 128
Allocated memory region is 432
Do you want to continue ..(1/0):1
    1.  Best fit algorithm
    2.  First fit algorithm'
    3.  Exit
    Enter ur choice :1
                Best fit and first fit algorithm'
                _______________________
                        Memory free is list
                        _____________
                        [______________________]
                        |6|
                        _________________
        UR memory requirement is 5
        Allocated memory region is 6
```

e) Result:

Thus the First Bit and Best Fit program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. **What are the common strategies to select a free hole from a set of available holes?**
   The most common strategies are a. First fit b. Best fit c. Worst fit
2. **What do you mean by best fit? (APRIL/MAY 07)**
   Best fit allocates the smallest hole that is big enough. His entire list has to be searched, unless it is sorted by size. His strategy produces the smallest leftover hole.
3. **What do you mean by first fit? (NOV/DEC 09)**
   First fit allocates the first hole that is big enough. Searching can either start at the beginning of the set of holes or where the previous first-fit search ended. Searching can be stopped as soon as a free hole that is big enough is found.

Exercise Number: 10(a)

Title of the Exercise : FILE MANIPULATION-I
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program for file manipulation for displays the file and directory in

UNIX.FACILITIES **FACILITIES REQUIRED AND PROCEDURE**

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program
2	Use the function print return type wid it declare the variable. all with clear.
3	Main function is used to check the file present in the directory or not.
4	Using the file pointer in the file to that the argument is less than a times means print std in.
5	By using if loop check in file, open two means print error
6	Stop the program

c) Program

```
#include<stdio.h>
void print(FILE *infile)
{
char str[80];
while(fgets(str,80,infile))
printf("%s",str);
}
int main(int argc,char *argv[])
{
char str[80];
FILE *infile;
if(argc<2)
print(stdin);
else
{
int i;
for(i=1;i<argc;i++)
{
if(!(infile=fopen(argv[i],"rt")))
perror("argv[i]");
else
print(infile);
fclose(infile);
}
```

Output:

```
[iicse31@csehost iicse31]$ cc fileman 1.c
[iicse31@csehost iicse31]$ ./a.out pattern.sh
echo"enter the pattern name"
read pattern
echo"enter the filename"
read filename
grep-n$patternname $filename

[iicse31@csehost iicse 31]$
```

d) Result:

Thus the File manipulation I program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. **What is a reference string?**
   An algorithm is evaluated by running it on a particular string of memory references and computing the number of page faults. The string of memory reference is called a reference string.
2. **What is a file? (APRIL/MAY 08) (MAY/JUNE 2011)**
   A file is a named collection of related information that is recorded on secondary storage. A file contains either programs or data. A file has certain "structure" based on its type.
   • File attributes: Name, identifier, type, size, location, protection, time, date
   a. • File operations: creation, reading, writing, repositioning, deleting, truncating, appending, renaming
   b. • File types: executable, object, library, source code etc.
3. **List the various file attributes.**
   A file has certain other attributes, which vary from one operating system to another, but typically consist of these: Name, identifier, type, location, size, protection, time, and date and user identification
4. **What is Directory?**
   The device directory or simply known as directory records information-such as name, location, size, and type for all files on that particular partition. The directory can be viewed as a symbol table that translates file names into their directory entries.
5. **Define UFD and MFD.(NOV/DEC 07)**
   In the two-level directory structure, each user has her own user file directory (UFD). Each UFD has a similar structure, but lists only the files of a single user. When a job starts the system's master file directory (MFD) is searched. The MFD is indexed by the user name or account number, and each entry points to the UFD for that user.

Exercise Number: 10(b)

Title of the Exercise : FILE MANIPULATION-II
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to perform file manipulation for creating a new directory.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net,UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Start the program
2	void print c base and pointer
3	print waqe name and get the value of exit ( )
4	Then int and go to if loop
5	print the argument
6	Stop the program

c) Program

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/stat.h>
void printusage(const char *myname)
{
printf("\nUsage;%s<path name>[<mode>]\n\n",myname)
exit(1);
}
int main(int argc,char *argv[])
{
int mode;
if(argc==2)
printusage(argv[0]);
if(argc==2)
mode=0777;
else if(scanf(argv[2],"%0",mode==-1)
{
printf("%s:invalid mode\n",argv[0]);
exit(1);
}
if(mkdir(argv[1],mode==-1)
perror(argv[0]);
else
printf("new directory %s is created\n",argv[1]);
}
```

d) Output:

```
[iicse28@csehost iicse28]$ cc file2.c
[iicse28@csehost iicse 28]$ ./a.out os
new directory os is created
[iicse28@csehost iicse28]$ cd os
[iicse28@csehost os]$
```

e) Result:

Thus the File Manipulation II program was executed and verified successfully.

QUESTIONS AND ANSWERS

1. **Define seek time and latency time.**
   The time taken by the head to move to the appropriate cylinder or track is called seek time. Once the head is at right track, it must wait until the desired block rotates under the read-write head. This delay is latency time.
2. **What are the allocation methods of a disk space?**
   Three major methods of allocating disk space which are widely in use are
   a. Contiguous allocation b. Linked allocation c. Indexed allocation
3. **What are the advantages of Contiguous allocation?**
   The advantages are
   a. Supports direct access b. Supports sequential access c. Number of disk seeks is minimal.
4. **What are the drawbacks of contiguous allocation of disk space?**
   The disadvantages are
   a. Suffers from external fragmentation
   b. Suffers from internal fragmentation
   c. Difficulty in finding space for a new file
   d. File cannot be extended
   e. Size of the file is to be declared in advance
5. **What are the advantages of Linked allocation?**
   The advantages are:
   a. No external fragmentation
   b. Size of the file does not need to be declared

Exercise Number: 11

Title of the Exercise : Even or Odd
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to find whether a number is even or odd

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Read the input number.
2	Perform modular division on input number by 2.
3	If remainder is 0 print the number is even.
4	Else print number is odd.
5	Stop the program.

c) Program

```
echo "enter the number"
read num
echo "enter the number"
read num
if [ `expr $num % 2` -eq 0 ]
then
echo "number is even"
else
echo "number is odd"
fi
```

Output:

```
enter the number: 5
the number is odd.
```

d) Result:

Thus the program has been executed successfully.

QUESTIONS AND ANSWERS

1. Explain the concept of Reentrancy?

It is a useful, memory-saving technique for multiprogrammed timesharing systems. A Reentrant Procedure is one in which multiple users can share a single copy of a program during the same period.Reentrancy has 2 key aspects: The program code cannot modify itself, and the local data

for each user process must be stored separately. Thus, the permanent part is the code, and the temporary part is the pointer back to the calling program and local variables used by that program. Each execution instance is called activation. It executes the code in the permanent part, but has its own copy of local variables/parameters. The temporary part associated with each activation is the activation record.Generally, the activation record is kept on the stack.
Note: A reentrant procedure can be interrupted and called by an interrupting program, and still execute correctly on returning to the procedure.

2. Explain Belady's Anomaly?

Also called FIFO anomaly. Usually, on increasing the number of frames allocated to a process virtual memory, the process execution is faster, because fewer page faults occur. Sometimes, the reverse happens, i.e., the execution time increases even when more frames are allocated to the process. This is Belady's Anomaly. This is true for certain page reference patterns.

3. What is a binary semaphore? What is its use?

A binary semaphore is one, which takes only 0 and 1 as values. They are used to implement mutual exclusion and synchronize concurrent processes.

4. What is thrashing?

It is a phenomenon in virtual memory schemes when the processor spends most of its time swapping pages, rather than executing instructions. This is due to an inordinate number of page faults.

5. List the Coffman's conditions that lead to a deadlock.

1. Mutual Exclusion: Only one process may use a critical resource at a time.
2. Hold & Wait: A process may be allocated some resources while waiting for others.
3. No Pre-emption: No resource can be forcible removed from a process holding it.
4. Circular Wait: A closed chain of processes exist such that each process holds at least one resource needed by another process in the chain.

Exercise Number: 12

Title of the Exercise : Biggest Of Two Numbers
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to find biggest in two numbers.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Read The Two Numbers.
2	If Value Of A Is Greater Than B Is Big.
3	Else Print B Is Big.
4	Stop The Program.

c) Program

```
echo "enter the number"
read a b
if [ $a -gt $b ]
then
echo "A is big"
else
echo "B is big"
fi
```

Output:

```
Enter The Two Number:
23 67
    B is Big.
```

d) Result:

Thus the program has been executed successfully.

QUESTIONS AND ANSWERS

1. What are short, long and medium-term scheduling?

Long term scheduler determines which programs are admitted to the system for processing. It controls the degree of multiprogramming. Once admitted, a job becomes a process. Medium term scheduling is part of the swapping function. This relates to processes that are in a blocked or suspended state. They are swapped out of real-memory until they are ready to execute. The swapping-in decision is based on memory-management criteria. Short term scheduler, also known as a dispatcher executes most frequently, and makes the finest grained decision of which process should execute next. This scheduler is invoked whenever an event occurs. It may lead to interruption of one process by preemption.

2. What are turnaround time and response time?

Turnaround time is the interval between the submission of a job and its completion. Response time is the interval between submission of a request, and the first response to that request.

3. What are the typical elements of a process image?

User data: Modifiable part of user space. May include program data, user stack area, and programs that may be modified. User program: The instructions to be executed.System Stack: Each process has one or more LIFO stacks associated with it. Used to store parameters and calling addresses for procedure and system calls.Process control Block (PCB): Info needed by the OS to control processes.

4. What is the Translation Lookaside Buffer (TLB)?

In a cached system, the base addresses of the last few referenced pages is maintained in registers called the TLB that aids in faster lookup. TLB contains those page-table entries that have been most recently used. Normally, each virtual memory reference causes 2 physical memory accesses- one to fetch appropriate page-table entry, and one to fetch the desired data. Using TLB in-between, this is reduced to just one physical memory access in cases of TLB-hit.

5. What is the resident set and working set of a process?

Resident set is that portion of the process image that is actually in real-memory at a particular instant.Working set is that subset of resident set that is actually needed for execution. (Relate this to the variable-window size method for swapping techniques.)

Exercise Number: 13

Title of the Exercise : Factorial Of Number
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To find a factorial of a number using shell script.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number.
2	Initialize fact as 1.
3	Initialize I as 1.
4	While I is lesser than or equal to no.
5	Multiply the value of I and fact and assign to fact increment the value of I by 1.
6	print the result.
7	Stop the program.

c) Program

```
echo "enter the number"
read n
fact=1
i=1
while [ $i -le $n ]
do
fact=`expr $i \* $fact`
i=`expr $i + 1`
done
echo "the fcatorial number of $ni is $fact
```

d) Output:

```
Enter the number :
4
The factorial of 4 is 24.
```

e) Result:

Thus the program has been executed successfully.

QUESTIONS AND ANSWERS

1. When is a system in safe state?

The set of dispatchable processes is in a safe state if there exists at least one temporal order in which all processes can be run to completion without resulting in a deadlock.

2. What is cycle stealing?

We encounter cycle stealing in the context of Direct Memory Access (DMA). Either the DMA controller can use the data bus when the CPU does not need it, or it may force the CPU to temporarily suspend operation. The latter technique is called cycle stealing. Note that cycle stealing can be done only at specific break points in an instruction cycle.

3. What is meant by arm-stickiness?

If one or a few processes have a high access rate to data on one track of a storage disk, then they may monopolize the device by repeated requests to that track. This generally happens with most common device scheduling algorithms (LIFO, SSTF, C-SCAN, etc). High-density multisurface disks are more likely to be affected by this than low density ones.

4. What are the stipulations of C2 level security?

C2 level security provides for:

1. Discretionary Access Control
2. Identification and Authentication
3. Auditing
4. Resource reuse

Exercise Number: 14

Title of the Exercise : Fibonacci Series
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To find a Fibonacci Series using shell script.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

Title of the Exercise : Factorial Of Number

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To find a factorial of a number using shell script.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Tel net, UNIX OS	-

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number.
2	Initialize fact as 1.
3	Initialize I as 1.
4	While I is lesser than or equal to no.
5	Multiply the value of I and fact and assign to fact increment the value of I by 1.
6	print the result.
7	Stop the program.

c) Program

```
echo "enter the number"
read n
fact=1
i=1
while [ $i -le $n ]
do
fact=`expr $i \* $fact`
i=`expr $i + 1`
done
```

```
echo "the fcatorial number of $ni is $fact
```

d) Output:

```
Enter the number :
4
The factorial of 4 is 24.
```

e) Result:

Thus the program has been executed successfully.

QUESTIONS AND ANSWERS

1. What is busy waiting?

The repeated execution of a loop of code while waiting for an event to occur is called busy-waiting. The CPU is not engaged in any real productive activity during this period, and the process does not progress toward completion.

2. Explain the popular multiprocessor thread-scheduling strategies.

1. Load Sharing: Processes are not assigned to a particular processor. A global queue of threads is maintained. Each processor, when idle, selects a thread from this queue. Note that load balancing refers to a scheme where work is allocated to processors on a more permanent basis.

3. Gang Scheduling: A set of related threads is scheduled to run on a set of processors at the same time, on a 1-to-1 basis. Closely related threads / processes may be scheduled this way to reduce synchronization blocking, and minimize process switching. Group scheduling predated this strategy.

4. Dedicated processor assignment: Provides implicit scheduling defined by assignment of threads to processors. For the duration of program execution, each program is allocated a set of processors equal in number to the number of threads in the program. Processors are chosen from the available pool.

5. Dynamic scheduling: The number of thread in a program can be altered during the course of execution.

MODEL QUESTIONS

1. To write c program to implement the Process system calls
2. To find a factorial of a number using shell script.
3. To find a Fibonacci Series using shell script
4. To write a program to find biggest in two numbers.
5. To write a program to find whether a number is even or odd
6. To write a program for file manipulation for displays the file and directory in
7. To implement first fit, best fit algorithm for memory management.
8. To implement producer/consumer problem using semaphore.
9. To write a program for pipe processing.
10. To write a program to implement cpu scheduling for Round Robin Scheduling.
11. To write a ‘C’ program to perform priority scheduling.
12. To write a program to implement cpu scheduling algorithm for shortest job first scheduling.
13. To write the program to implement CPU & scheduling algorithm for first come first serve scheduling.
14. To write a ‘c’ program for I/O system calls.