

CSA0465 – OPERATING SYSTEMS FOR HANDLING DEADLOCKS

LAB EXPERIMENTS

Name :- D.Siva Prasad Reddy

Reg no :- 192011401

11. Round Robin :-

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

```
int main()
```

```
{
```

```
int count,j,n,time,remain,flag=0,time_quantum;
```

```
int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
```

```
printf("Enter Total Process:\t ");
```

```
scanf("%d",&n);
```

```
remain=n;
```

```
for(count=0;count<n;count++)
```

```
{
```

```
printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
```

```
scanf("%d",&at[count]);
```

```
scanf("%d",&bt[count]);
```

```
rt[count]=bt[count];
```

```
}
```

```
printf("Enter Time Quantum:\t");
```

```
scanf("%d",&time_quantum);
```

```
printf("\n\nProcess\t| Turnaround Time | Waiting Time\n\n");
```

```
for(time=0,count=0;remain!=0;)
```

```
{
```

```
if(rt[count]<=time_quantum && rt[count]>0)
```

```

{
time+=rt[count];
rt[count]=0;
flag=1;
}
else if(rt[count]>0)
{
rt[count]-=time_quantum;
time+=time_quantum;
}
if(rt[count]==0 && flag==1)
{
remain--;
printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
wait_time+=time-at[count]-bt[count];
turnaround_time+=time-at[count];
flag=0;
}
if(count==n-1)
count=0;
else if(at[count+1]<=time)
count++;
else
count=0;
}
printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

```

}

```
if (x + (count + 1) == 0) { f1 = a; }
```

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

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

```
int mutex = 1;
```

```
int full = 0;
```

```
int empty = 10, x = 0;
```

```
void producer()
```

 $\{$

```
++full;
```

```
--empty;
```

```
x++;
```

```
printf("\nProducer produces"
```

x);

```

        ++mutex;
    }

void consumer()
{
    --mutex;
    --full;
    ++empty;
    printf("\nConsumer consumes "
           "item %d",
           x);
    x--;
    ++mutex;
}

int main()
{
    int n, i;

    printf("\n1. Press 1 for Producer"
           "\n2. Press 2 for Consumer"
           "\n3. Press 3 for Exit");

#pragma omp critical
    for (i = 1; i > 0; i++) {
        printf("\nEnter your choice:");
        scanf("%d", &n);
        switch (n) {
            case 1:
                if ((mutex == 1)
                    && (empty != 0)) {

```

```

        producer();
    }
    else {
        printf("Buffer is full!");
    }
    break;
case 2:
        if ((mutex == 1)
            && (full != 0)) {
            consumer();
        }
        else {
            printf("Buffer is empty!");
        }
        break;
case 3:
    exit(0);
    break;
}
}
}

```

The image shows a C program for a Producer-Consumer problem and its execution output. The program is written in a text editor and runs in a command prompt window.

```
#include <stdio.h>
#include <stdlib.h>
int mutex = 1;
int full = 0;
int empty = 10, x = 0;
void producer()
{
    --mutex;
    ++full;
    --empty;
    x++;
    printf("\nProducer produces "
           "item %d",
           x);
    ++mutex;
}
void consumer()
{
    --mutex;
    --full;
    ++empty;
    printf("\nConsumer consumes "
           "item %d",
           x);
    x--;
    ++mutex;
}
int main()
{
    int n, i;
    printf("\n1. Press 1 for Producer"
           "\n2. Press 2 for Consumer"
           "\n3. Press 3 for Exit");
}
```

The execution output shows the following sequence of events:

```
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice:1
Producer produces item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:1
Producer produces item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:3
Process returned 0 (0x0)   execution time : 23.955 s
Press any key to continue.
```

13. Dinning Philosopher :-

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

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

```
#include<pthread.h>
```

```
#include<semaphore.h>
```

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

```
sem_t room;
```

```
sem_t chopstick[5];
```

```
void * philosopher(void *);
```

```
void eat(int);
```

```
int main()
```

```
{
```

```
    int i,a[5];
```

```
    pthread_t tid[5];
```

```

sem_init(&room,0,4);

for(i=0;i<5;i++)
    sem_init(&chopstick[i],0,1);

for(i=0;i<5;i++){
    a[i]=i;
    pthread_create(&tid[i],NULL,philosopher,(void *)&a[i]);
}
for(i=0;i<5;i++)
    pthread_join(tid[i],NULL);
}

void * philosopher(void * num)
{
    int phil=*(int *)num;

    sem_wait(&room);
    printf("\nPhilosopher %d has entered room",phil);
    sem_wait(&chopstick[phil]);
    sem_wait(&chopstick[(phil+1)%5]);

    eat(phil);
    sleep(2);
    printf("\nPhilosopher %d has finished eating",phil);

    sem_post(&chopstick[(phil+1)%5]);
}

```

```

        sem_post(&chopstick[phil]);

        sem_post(&room);
    }

void eat(int phil)
{
    printf("\nPhilosopher %d is eating",phil);
}

```

The screenshot shows a C program named 'program 13.c' and its execution output in a terminal window.

Program Code (program 13.c):

```

for(i=0;i<5;i++){
    a[i]=i;
    pthread_create(&tid[i],NULL,philosopher,(void *)&a[i]);
}
for(i=0;i<5;i++){
    pthread_join(tid[i],NULL);
}

void * philosopher(void * num)
{
    int phil=*(int *)num;

    sem_wait(&room);
    printf("\nPhilosopher %d has entered room",phil);
    sem_wait(&chopstick[phil]);
    sem_wait(&chopstick[(phil+1)%5]);

    eat(phil);
    sleep(2);
    printf("\nPhilosopher %d has finished eating",phil);

    sem_post(&chopstick[(phil+1)%5]);
    sem_post(&chopstick[phil]);
    sem_post(&room);
}

void eat(int phil)
{
    printf("\nPhilosopher %d is eating",phil);
}

```

Execution Output:

```

Philosopher 0 has entered room
Philosopher 1 has entered room
Philosopher 0 is eating
Philosopher 2 has entered room
Philosopher 2 is eating
Philosopher 3 has entered room
Philosopher 0 has finished eating
Philosopher 2 has finished eating
Philosopher 4 has entered room
Philosopher 3 is eating
Philosopher 1 is eating
Philosopher 1 has finished eating
Philosopher 3 has finished eating
Philosopher 4 is eating
Philosopher 4 has finished eating
Process returned 0 (0x0)   execution time : 6.092 s
Press any key to continue.

```

14. Bankers Algorithm :-

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

```

int main()
{
    int n,r,i,j,k,p,u=0,s=0,m;

    int block[10],run[10],active[10],newreq[10];

    int max[10][10],resalloc[10][10],resreq[10][10];

```



```

int totalloc[10],totext[10],simalloc[10];

//clrscr();

printf("Enter the no of processes:");

scanf("%d",&n);

printf("Enter the no ofresource classes:");

scanf("%d",&r);

printf("Enter the total existed resource in each class:");

for(k=1; k<=r; k++)

    scanf("%d",&totext[k]);

printf("Enter the allocated resources:");

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

    for(k=1; k<=r; k++)

        scanf("%d",&resalloc);

printf("Enter the process making the new request:");

scanf("%d",&p);

printf("Enter the requested resource:");

for(k=1; k<=r; k++)

    scanf("%d",&newreq[k]);

printf("Enter the process which are n blocked or running:");

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

{

    if(i!=p)

    {

        printf("process %d:\n",i+1);

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

    }

}

}

```

```

block[p]=0;
run[p]=0;
for(k=1; k<=r; k++)
{

    j=0;
    for(i=1; i<=n; i++)
    {
        totalloc[k]=j+resalloc[i][k];
        j=totalloc[k];
    }
}
for(i=1; i<=n; i++)
{
    if(block[i]==1 || run[i]==1)
        active[i]=1;
    else
        active[i]=0;
}
for(k=1; k<=r; k++)
{
    resalloc[p][k]+=newreq[k];
    totalloc[k]+=newreq[k];
}
for(k=1; k<=r; k++)
{
    if(totext[k]-totalloc[k]<0)

```

```

{
    u=1;
    break;
}
}
if(u==0)
{
    for(k=1; k<=r; k++)
        simalloc[k]=totalloc[k];
    for(s=1; s<=n; s++)
        for(i=1; i<=n; i++)
        {
            if(active[i]==1)
            {
                j=0;
                for(k=1; k<=r; k++)
                {
                    if((totext[k]-simalloc[k])<(max[i][k]-resalloc[i][k]))
                    {
                        j=1;
                        break;
                    }
                }
            }
        }
        if(j==0)

        {

```

```

        active[i]=0;
        for(k=1; k<=r; k++)
            simalloc[k]=resalloc[i][k];
    }
}
m=0;
for(k=1; k<=r; k++)
    resreq[p][k]=newreq[k];
printf("Deadlock willn't occur");
}
else
{
    for(k=1; k<=r; k++)
    {
        resalloc[p][k]=newreq[k];
        totalloc[k]=newreq[k];
    }
    printf("Deadlock will occur");
}
}

```

```
in 12.c × program 13.c × program 14.c ×
1 #include<stdio.h>
2
3 int main()
4 {
5     int n,r,i,j,k,p,u=0,s=0,m;
6     int block[10],run[10],active[10],newreq[10];
7     int max[10][10],resalloc[10][10],resreq[10][10];
8     int totalloc[10],totext[10],simalloc[10];
9     //classx();
10    printf("Enter the no of processes:");
11    scanf("%d",&n);
12    printf("Enter the no of resource classes:");
13    scanf("%d",&r);
14    printf("Enter the total existed resource in each class:");
15    for(k=1; k<=r; k++)
16        scanf("%d",&totext[k]);
17    printf("Enter the allocated resources:");
18    for(i=1; i<=n; i++)
19        for(k=1; k<=r; k++)
20            scanf("%d",&resalloc[i][k]);
21    printf("Enter the process making the new request:");
22    scanf("%d",&p);
23    printf("Enter the requested resource:");
24    for(k=1; k<=r; k++)
25        scanf("%d",&newreq[k]);
26    printf("Enter the process which are n blocked or running:");
27    for(i=1; i<=n; i++)
28    {
29        if(i!=p)
30        {
31            printf("process %d:\n",i+1);
32            scanf("%d%d",&block[i],&run[i]);
33            printf("Enter the no of processes:5\n");
34            printf("Enter the no of resource classes:4\n");
35            printf("Enter the total existed resource in each class:1 1 0 0\n");
36            printf("Enter the allocated resources:8 7 2 3\n");
37            printf("2 6 7 9\n");
38            printf("1 6 7 8\n");
39            printf("2 7 6 5\n");
40            printf("1 2 8 5\n");
41            printf("Enter the process making the new request:4\n");
42            printf("Enter the requested resource:3\n");
43            printf("2 7 6 5\n");
44            printf("Enter the process which are n blocked or running:process 2:\n");
45            printf("1 2 8 5\n");
46            printf("process 3:\n");
47            printf("process 4:\n");
48            printf("1 0 0 0\n");
49            printf("process 6:\n");
50            printf("Deadlock will occur\n");
51            printf("Process returned 0 (0x0)   execution time : 98.843 s\n");
52            printf("Press any key to continue.\n");
53        }
54    }
55 }
```

15. Multi Threading :-

```
#include<string.h>
```

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

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

struct

 $\{$

```
char dname[10],fname[10][10];
```

```
int fcnt;
```

```
}dir[10];
```

```
int main()
```

 $\{$

```
int i,ch,dcnt,k;
```

```
char f[30], d[30];
```

```
dcnt=0;
```

```
while(1)
```

$$\{$$

```
printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");
```

```
printf("\n4. Search File\t5. Display\t6. Exit\tEnter your choice -- ");
```

```

scanf("%d",&ch);
switch(ch)
{
case 1: printf("\nEnter name of directory -- ");
scanf("%s", dir[dcnt].dname);
dir[dcnt].fcnt=0;
dcnt++;
printf("Directory created");
break;
case 2: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
if(strcmp(d,dir[i].dname)==0)
{
printf("Enter name of the file -- ");
scanf("%s",dir[i].fname[dir[i].fcnt]);
printf("File created");
break;
}
if(i==dcnt)
printf("Directory %s not found",d);
break;
case 3: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)

```

```

{
printf("Enter name of the file -- ");
scanf("%s",f);
for(k=0;k<dir[i].fcnt;k++)
{
if(strcmp(f, dir[i].fname[k])==0)
{
printf("File %s is deleted ",f);
dir[i].fcnt--;
strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);
goto jmp;
}
}
printf("File %s not found",f);
goto jmp;
}
}
printf("Directory %s not found",d);
jmp : break;
case 4: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)
{
printf("Enter the name of the file -- ");
scanf("%s",f);

```

```

for(k=0;k<dir[i].fcnt;k++)
{
if(strcmp(f, dir[i].fname[k])==0)
{
printf("File %s is found ",f);
goto jmp1;
}
}
printf("File %s not found",f);
goto jmp1;
}
printf("Directory %s not found",d);
jmp1: break;
case 5: if(dcnt==0)
printf("\nNo Directory's ");
else
{
printf("\nDirectory\tFiles");
for(i=0;i<dcnt;i++)
{
printf("\n%s\t",dir[i].dname);
for(k=0;k<dir[i].fcnt;k++)
printf("\t%s",dir[i].fname[k]);
}
}
break;

```



```
default:exit(0);
```

```
}
```

```
}
```

```
}
```

Output :-

```
1 #include<string.h>
2 #include<stdlib.h>
3 #include<stdio.h>
4 struct
5 {
6     char dname[10],fname[10][10];
7     int fcnt;
8 }dir[10];
9 int main()
10 {
11     int i,ch,dcnt,k;
12     char f[30],d[30];
13     dcnt=0;
14     while(1)
15     {
16         printf("\n1. Create Directory\t2. Create File\t3. Delete File\t4. Search File\t5. Display\t6. Exit\n");
17         scanf("%d",&ch);
18         switch(ch)
19         {
20             case 1: printf("\nEnter name of directory -- ");
21                     scanf("%s", dir[dcnt].dname);
22                     dir[dcnt].fcnt=0;
23                     dcnt++;
24                     printf("Directory created");
25                     break;
26             case 2: printf("\nEnter name of the directory ");
27                     scanf("%s",d);
28                     for(i=0;i<dcnt;i++)
29                     if(strcmp(d,dir[i].dname)==0)
30                     {
31                         printf("Enter name of the file ");
32                         scanf("%s",f);
33                         if(strcmp(f,dir[i].fname[i])!=0)
34                         {
35                             printf("File created");
36                             dir[i].fname[i][0]='\0';
37                             strcpy(dir[i].fname[i],f);
38                             break;
39                         }
40                     }
41                     break;
42             case 3: printf("\nEnter name of the directory -- ");
43                     scanf("%s",d);
44                     for(i=0;i<dcnt;i++)
45                     if(strcmp(d,dir[i].dname)==0)
46                     {
47                         printf("File %s not found",dir[i].fname[i]);
48                         break;
49                     }
50                     break;
51             case 4: printf("\nEnter name of the directory -- ");
52                     scanf("%s",d);
53                     for(i=0;i<dcnt;i++)
54                     if(strcmp(d,dir[i].dname)==0)
55                     {
56                         printf("Files\n");
57                         for(k=0;k<dir[i].fcnt;k++)
58                             printf("%s\t",dir[i].fname[i][k]);
59                         printf("\n");
60                     }
61                     break;
62             case 5: printf("\nEnter name of the directory -- ");
63                     scanf("%s",d);
64                     for(i=0;i<dcnt;i++)
65                     if(strcmp(d,dir[i].dname)==0)
66                     {
67                         printf("Directory\n");
68                         for(k=0;k<dir[i].fcnt;k++)
69                             printf("%s\t",dir[i].fname[i][k]);
70                         printf("\n");
71                     }
72                     break;
73             case 6: exit(0);
74             default: continue;
75         }
76     }
77 }
```

```
"C:\Users\siva8\OneDrive\Documents\OS\Program 15.exe"
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 1
Enter name of directory -- Siva
Directory created
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 2
Enter name of the directory -- Siva
Enter name of the file -- Srinu
File created
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 3
Enter name of the directory -- Srinu
Directory Srinu not found
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 4
Enter name of the directory -- Siva
Enter the name of the file -- Srinu
File Srinu not found
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 5
Directory Files
Siva
1. Create Directory 2. Create File 3. Delete File
4. Search File 5. Display 6. Exit Enter your choice -- 6
Process returned 0 (0x0) execution time : 133.834 s
Press any key to continue.
```