

DAY -2

CSA0465 – OPERATING SYSTEMS FOR HANDLING DEADLOCKS

LAB EXPERIMENTS – Slot B

Name :- Aswini .P

Reg no :- 192011399

6. Producer and Consumer.

Program :-

```
#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");
#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();
}

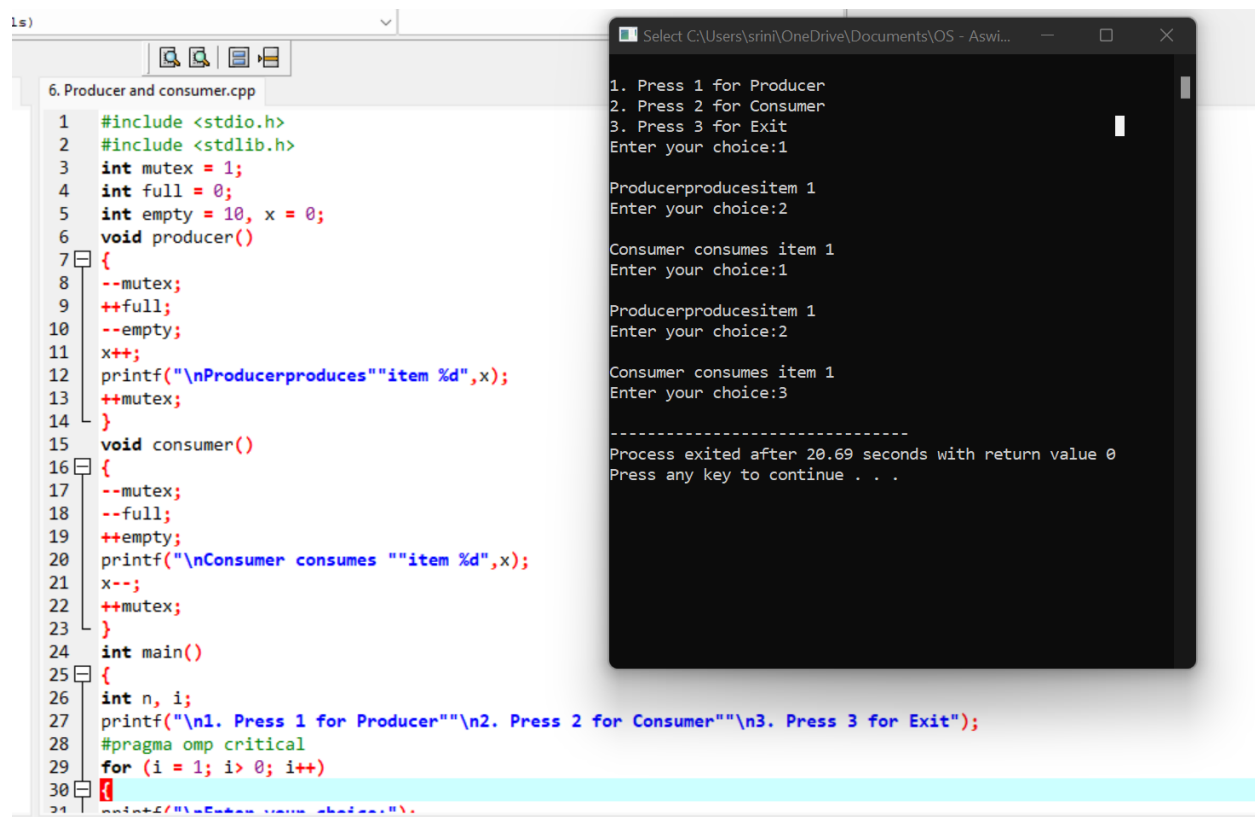
```

```

else {
printf("Buffer is empty!");
}
break;
case 3:
exit(0);
break;
}
}
}

```

Output :-



The screenshot shows a code editor on the left and a terminal window on the right. The code editor displays the source code for a Producer-Consumer program in C, using OpenMP for synchronization. The terminal window shows the program's output, including prompts for user choice, item production and consumption messages, and a final exit message.

```

6. Producer and consumer.cpp
1  #include <stdio.h>
2  #include <stdlib.h>
3  int mutex = 1;
4  int full = 0;
5  int empty = 10, x = 0;
6  void producer()
7  {
8      --mutex;
9      ++full;
10     --empty;
11     x++;
12     printf("\nProducer produces item %d", x);
13     ++mutex;
14 }
15 void consumer()
16 {
17     --mutex;
18     --full;
19     ++empty;
20     printf("\nConsumer consumes item %d", x);
21     x--;
22     ++mutex;
23 }
24 int main()
25 {
26     int n, i;
27     printf("\n1. Press 1 for Producer\n2. Press 2 for Consumer\n3. Press 3 for Exit");
28     #pragma omp critical
29     for (i = 1; i > 0; i++)
30     {
31         printf("\nEnter your choice: ");

```

```

Select C:\Users\sriini\OneDrive\Documents\OS - Aswi...
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 exited after 20.69 seconds with return value 0
Press any key to continue . . .

```

7. Paging – FIFO :-

Program :-

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

```

int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
printf("\n ENTER THE PAGE NUMBER :\n");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("\n ENTER THE NUMBER OF FRAMES :");
scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]= -1;
j=0;
printf("\tref string\t page frames\n");
for(i=1;i<=n;i++)
{
printf("%d\t\t",a[i]);
avail=0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if (avail==0)
{
frame[j]=a[i];
j=(j+1)%no;
count++;
for(k=0;k<no;k++)
printf("%d\t",frame[k]);

```

```

}

printf("\n");

}

printf("Page Fault Is %d",count);

return 0;

}

```

Output :-

The screenshot shows a C program for the FIFO paging algorithm. The code is in a file named '7. Paging FIFO.c'. It prompts the user to enter the number of pages (n) and the number of frames (no). It then displays a table of reference strings and the state of page frames. The output shows that there are 7 page faults.

```

1  #include<stdio.h>
2  int main()
3  {
4      int i,j,n,a[50],frame[10],no,k,avail,count=0;
5      printf("\n ENTER THE NUMBER OF PAGES:\n");
6      scanf("%d",&n);
7      printf("\n ENTER THE PAGE NUMBER :\n");
8      for(i=1;i<=n;i++)
9          scanf("%d",&a[i]);
10     printf("\n ENTER THE NUMBER OF FRAMES :");
11     scanf("%d",&no);
12     for(i=0;i<no;i++)
13         frame[i]= -1;
14     j=0;
15     printf("\tref string\t page frames\n");
16     for(i=1;i<=n;i++)
17     {
18         printf("%d\t\t",a[i]);
19         avail=0;
20         for(k=0;k<no;k++)
21             if(frame[k]==a[i])
22                 avail=1;
23         if (avail==0)
24         {
25             frame[j]=a[i];
26             j=(j+1)%no;
27             count++;
28             for(k=0;k<no;k++)
29                 printf("%d\t",frame[k]);
30         }
31         printf("\n");
32     }
33     printf("Page Fault Is %d",count);
34     return 0;
35 }

```

Output:

```

ENTER THE NUMBER OF PAGES:
9

ENTER THE PAGE NUMBER :
1
2
3
5
6
3
2
4
2

ENTER THE NUMBER OF FRAMES :3
ref string      page frames
1              1      -1      -1
2              1      2      -1
3              1      2      3
5              5      2      3
6              5      6      3
3
2              5      6      2
4              4      6      2
2

Page Fault Is 7
Process returned 0 (0x0)   execution time : 19.512 s
Press any key to continue.

```

8. Paging – LRU :-

Program :-

```

#include<stdio.h>

int main()

{

int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];

printf("Enter no of pages:");

```

```
scanf("%d",&n);
printf("Enter the reference string:");
for(i=0;i<n;i++)
scanf("%d",&p[i]);
printf("Enter no of frames:");
scanf("%d",&f);
q[k]=p[k];
printf("\n\t%d\n",q[k]);
c++;
k++;
for(i=1;i<n;i++)S
{
c1=0;
for(j=0;j<f;j++)
{
if(p[i]!=q[j])
c1++;
}
if(c1==f)
{
c++;
if(k<f)
{
q[k]=p[i];
k++;
for(j=0;j<k;j++)
printf("\t%d",q[j]);
printf("\n");
```

```
}  
else  
{  
for(r=0;r<f;r++)  
{  
c2[r]=0;  
for(j=i-1;j<n;j--)  
{  
if(q[r]!=p[j])  
c2[r]++;  
else  
break;  
}  
}  
for(r=0;r<f;r++)  
b[r]=c2[r];  
for(r=0;r<f;r++)  
{  
for(j=r;j<f;j++)  
{  
if(b[r]<b[j])  
{  
t=b[r];  
b[r]=b[j];  
b[j]=t;  
}  
}  
}
```

```

for(r=0;r<f;r++)
{
if(c2[r]==b[0])
q[r]=p[i];
printf("\t%d",q[r]);
}
printf("\n");
}
}
}
printf("\nThe no of page faults is %d",c);
}

```

Output :-

```

jing FIFO.c  X  *8. Paging LRU.c  X
1  #include<stdio.h>
2  int main()
3  {
4  int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
5  printf("Enter no of pages:");
6  scanf("%d",&n);
7  printf("Enter the reference string:");
8  for(i=0;i<n;i++)
9  scanf("%d",&p[i]);
10 printf("Enter no of frames:");
11 scanf("%d",&f);
12 q[k]=p[k];
13 printf("\n\t%d\n",q[k]);
14 c++;
15 k++;
16 for(i=1;i<n;i++)
17 {
18 c1=0;
19 for(j=0;j<f;j++)
20 {
21 if(p[i]!=q[j])
22 c1++;
23 }
24 if(c1==f)
25 {
26 c++;
27 if(k<f)
28 {
29 q[k]=p[i];
30 k++;
31 for(j=0;j<k;j++)
32 printf("\t%d",q[j]);
33 printf("\n");
34 }
35 else

```

```

"C:\Users\pavan\OneDrive\Documents\OS\8. ...
Enter no of pages:10
Enter the reference string:1
2
3
0
1
2
3
4
1
6
Enter no of frames:3

1      2
1      2      3
0      2      3
0      1      3
0      1      2
3      1      2
3      4      2
3      4      1
6      4      1

The no of page faults is 10
Process returned 0 (0x0)   execution time : 18.584 s
Press any key to continue.

```


9. Paging – Optimal :-

Program :-

```
#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;

}

```

Output :-

The screenshot shows a C program titled "Paging - Optimal.c" running in a terminal. The program implements the LRU page replacement algorithm. It prompts the user for the number of frames (3), the number of pages (8), and a page reference string (1 0 2 0 2 2 0 2 2). The program then displays a table of frame states and calculates 7 total page faults.

```

j FIFO.c x *8. Paging LRU.c x Paging - Optimal.c x
1 #include<stdio.h>
2 int main()
3 {
4     int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k, pos,
5     max, faults = 0;
6     printf("Enter number of frames: ");
7     scanf("%d", &no_of_frames);
8     printf("Enter number of pages: ");
9     scanf("%d", &no_of_pages);
10    printf("Enter page reference string: ");
11    for(i = 0; i < no_of_pages; ++i)
12    {
13        scanf("%d", &pages[i]);
14    }
15    for(i = 0; i < no_of_frames; ++i)
16    {
17        frames[i] = -1;
18    }
19    for(i = 0; i < no_of_pages; ++i)
20    {
21        flag1 = flag2 = 0;
22        for(j = 0; j < no_of_frames; ++j)
23        {
24            if(frames[j] == pages[i])
25            {
26                flag1 = flag2 = 1;
27                break;
28            }
29        }
30        if(flag1 == 0)
31        {
32            for(j = 0; j < no_of_frames; ++j)
33            {
34                if(frames[j] == -1)
35                {
36                    faults++;
37                    frames[j] = pages[i];
38                }
39            }
40        }
41    }
42    printf("\n\nTotal Page Faults = %d", faults);
43    return 0;
44 }

```

Terminal Output:

```

C:\Users\pavan\OneDrive\Documents\OS\Paging...
Enter number of frames: 3
Enter number of pages: 8
Enter page reference string: 1
0
2
0
2
0
2
0
2
2

1      -1      -1
1      0      -1
1      0      2
3      0      2
3      0      2
4      0      2
5      0      2
6      0      2

Total Page Faults = 7
Process returned 0 (0x0)   execution time : 14.211 s
Press any key to continue.

```

10 . Sequential file allocation :-

Program :-

```
#include <stdio.h>

typedef struct
{
    int usn;
    char name[25];
    int m1,m2,m3;
}
STD;
STD s;

void display(FILE *);
int search(FILE *,int);
void main()
{
    int i,n,usn_key,opn;
    FILE *fp;
    printf(" How many Records ? ");
    scanf("%d",&n);
    fp=fopen("stud.dat","w");
    for (i=0;i<n;i++)
    {
        printf("Read the Info for Student: %d (usn,name,m1,m2,m3) \n",i+1);
        scanf("%d%s%d%d%d",&s.usn,s.name,&s.m1,&s.m2,&s.m3);
        fwrite(&s,sizeof(s),1,fp);
    }
    fclose(fp);
    fp=fopen("stud.dat","r");
```

```

do
{
printf("Press 1- Display\t 2- Search\t 3- Exit\t Your Option?");
scanf("%d",&opn);
switch(opn)
{
case 1: printf("\n Student Records in the File \n");
display(fp);
break;
case 2: printf(" Read the USN of the student to be searched ?");
scanf("%d",&usn_key);
if(search(fp,usn_key))
{
printf("Success ! Record found in the file\n");
printf("%d\t%s\t%d\t%d\t%d\n",s.usn,s.name,s.m1,s.m2,s.m3);
}
else
printf(" Failure!! Record with USN %d not found\n",usn_key);
break;
case 3: printf(" Exit!! Press a key . . .");
break;
default: printf(" Invalid Option!!! Try again !!!\n");
break;
}
}
while(opn != 3);
fclose(fp);
}

```

```

/* End of main() */

void display(FILE *fp)
{
    rewind(fp);
    while(fread(&s,sizeof(s),1,fp))
    printf("%d\t%s\t%d\t%d\t%d\n",s.usn,s.name,s.m1,s.m2,s.m3);
}

int search(FILE *fp, int usn_key)
{
    rewind(fp);
    while(fread(&s,sizeof(s),1,fp))
    if( s.usn == usn_key) return 1;
    return 0;
}

```

Output :-

```

1  #include <stdio.h>
2  typedef struct
3  {
4      int usn;
5      char name[25];
6      int m1,m2,m3;
7  }
8  STD;
9  STD s;
10 void display(FILE *);
11 int search(FILE *,int);
12 void main()
13 {
14     int i,n,usn_key,opn;
15     FILE *fp;
16     printf(" How many Records ? ");
17     scanf("%d",&n);
18     fp=fopen("stud.dat","w");
19     for (i=0;i<n;i++)
20     {
21         printf("Read the Info for Student: %d (usn,name,m1,m2,m3) \n",i+1);
22         scanf("%d%s%d%d%d",&s.usn,&s.name,&s.m1,&s.m2,&s.m3);
23         fwrite(&s,sizeof(s),1,fp);
24     }
25     fclose(fp);
26     fp=fopen("stud.dat","r");
27     do
28     {
29         printf("Press 1- Display\t 2- Search\t 3- Exit\t Your Option?");
30         scanf("%d",&opn);
31         switch (opn)
32         {
33             case 1: printf("\n Student Records in the File \n");
34                     display(fp);
35                     break;
36             case 2: printf(" Read the USN of the student to be searched ?");
37                     scanf("%d",&usn_key);

```

```

"C:\Users\pavan\OneDrive\Documents\OS\10. Sequential file allocation.exe"
How many Records ? 3
Read the Info for Student: 1 (usn,name,m1,m2,m3)
1
alpha
56
88
99
Read the Info for Student: 2 (usn,name,m1,m2,m3)
2
beta
86
91
98
Read the Info for Student: 3 (usn,name,m1,m2,m3)
3
gamma
100
95
76
Press 1- Display      2- Search      3- Exit      Your Option?1
Student Records in the File
1  alpha   56   88   99
2  beta    86   91   98
3  gama    100  95   76
Press 1- Display      2- Search      3- Exit      Your Option?2
Read the USN of the student to be searched ?3
Success ! Record found in the file
3  gama    100  95   76
Press 1- Display      2- Search      3- Exit      Your Option?3
Exit!! Press a key . . .
Process returned 0 (0x0)   execution time : 171.144 s
Press any key to continue.

```