

DAY -4

CSA0465 – OPERATING SYSTEMS FOR HANDLING DEADLOCKS

LAB EXPERIMENTS – Slot B

Name :- Aswini .P

Reg no :- 192011399

16. First Fit Memory Allocation :-

Program :-

```
#include<stdio.h>

void main()
{
    int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;
    for(i = 0; i < 10; i++)
    {
        flags[i] = 0;
        allocation[i] = -1;
    }
    printf("Enter no. of blocks: ");
    scanf("%d", &bno);
    printf("\nEnter size of each block: ");
    for(i = 0; i < bno; i++)
        scanf("%d", &bsize[i]);
    printf("\nEnter no. of processes: ");
    scanf("%d", &pno);
    printf("\nEnter size of each process: ");
    for(i = 0; i < pno; i++)
        scanf("%d", &psize[i]);
    for(i = 0; i < pno; i++)        //allocation as per first fit
        for(j = 0; j < bno; j++)
            if(flags[j] == 0 && bsize[j] >= psize[i])
            {
```

```

        allocation[j] = i;

        flags[j] = 1;

        break;

    }

    //display allocation details

    printf("\nBlock no.\tsize\tprocess no.\tsize");

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

    {

        printf("\n%d\t%d\t", i+1, bsize[i]);

        if(flags[i] == 1)

            printf("%d\t\t", allocation[i]+1, psize[allocation[i]]);

        else

            printf("Not allocated");

    }

}

```

Output :-

```

1 #include<stdio.h>
2 void main()
3 {
4     int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;
5     for(i = 0; i < 10; i++)
6     {
7         flags[i] = 0;
8         allocation[i] = -1;
9     }
10    printf("Enter no. of blocks: ");
11    scanf("%d", &bno);
12    printf("\nEnter size of each block: ");
13    for(i = 0; i < bno; i++)
14        scanf("%d", &bsize[i]);
15    printf("\nEnter no. of processes: ");
16    scanf("%d", &pno);
17    printf("\nEnter size of each process: ");
18    for(i = 0; i < pno; i++)
19        scanf("%d", &psize[i]);
20    for(i = 0; i < pno; i++) //allocation as per first fit
21        for(j = 0; j < bno; j++)
22            if(flags[j] == 0 && bsize[j] >= psize[i])
23            {
24                allocation[j] = i;
25                flags[j] = 1;
26                break;
27            }
28    //display allocation details
29    printf("\nBlock no.\tsize\tprocess no.\tsize");
30    for(i = 0; i < bno; i++)
31    {
32        printf("\n%d\t\t", i+1, bsize[i]);
33        if(flags[i] == 1)
34            printf("%d\t\t", allocation[i]+1, psize[allocation[i]]);
35        else
36            printf("Not allocated");

```

Enter no. of blocks: 5
Enter size of each block: 3
15
20
16
19
Enter no. of processes: 5
Enter size of each process: 12
13
18
23
32

Block no.	size	process no.	size
1	3	Not allocated	12
2	15	1	12
3	20	2	13
4	16	Not allocated	
5	19	3	18

Process returned 5 (0x5) execution time : 82.734 s
Press any key to continue.

17. FCFS Disk Scheduling :-

Program :-

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

```

#include<stdlib.h>

int main()
{
int RQ[100],i,n,TotalHeadMoment=0,initial;
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);
for(i=0;i<n;i++)
{
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
}
printf("Total head moment is %d",TotalHeadMoment);
return 0;
}

```

Output :-

The image shows a screenshot of a C program and its execution output. On the left, a code editor window titled 'CFS Disk Scheduling.c' displays the source code. The code is a C program that calculates the total head moment for a disk scheduling algorithm. It includes headers for `stdio.h` and `stdlib.h`, and defines a `main` function. The program prompts the user to enter the number of requests and the requests sequence, then calculates the total head moment by iterating through the requests and summing the absolute differences between consecutive head positions. The final output is printed as 'Total head moment is 252'.

On the right, a terminal window titled '"C:\Users\pavan\OneDrive\Documents\OS\17. FCF..."' shows the execution of the program. The user enters '11' for the number of requests and '0 14 41 53 65 67 98 122 124 183 199' for the requests sequence. The program outputs 'Total head moment is 252' and 'Process returned 0 (0x0) execution time : 133.076 s'. The terminal also shows the prompt 'Press any key to continue.'.

18. SCAN Disk Scheduling :-

Program :-

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

```
#include <math.h>
```

```
int main()
```

```
{
```

```
    int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],  
    queue2[20], temp1 = 0, temp2 = 0;
```

```
    float avg;
```

```
    printf("Enter the max range of disk\n");
```

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

```
    printf("Enter the initial head position\n");
```

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

```
    printf("Enter the size of queue request\n");
```

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

```
    printf("Enter the queue of disk positions to be read\n");
```

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

```
{

scanf("%d", &temp);

if (temp >= head)

{

    queue1[temp1] = temp;

    temp1++;
}

else

{

    queue2[temp2] = temp;

    temp2++;
}

}

for (i = 0; i < temp1 - 1; i++)

{

    for (j = i + 1; j < temp1; j++)
```

```
{  
  
    if (queue1[i] > queue1[j])  
  
    {  
  
        temp = queue1[i];  
  
        queue1[i] = queue1[j];  
  
        queue1[j] = temp;  
    }  
}  
}
```

```
for (i = 0; i < temp2 - 1; i++)  
  
{  
  
    for (j = i + 1; j < temp2; j++)  
  
    {  
  
        if (queue2[i] < queue2[j])  
  
        {  
  
            temp = queue2[i];  
  
            queue2[i] = queue2[j];
```

```
        queue2[j] = temp;
    }
}
}
```

```
for (i = 1, j = 0; j < temp1; i++, j++)
```

```
    queue[i] = queue1[j];
```

```
queue[i] = max;
```

```
for (i = temp1 + 2, j = 0; j < temp2; i++, j++)
```

```
    queue[i] = queue2[j];
```

```
queue[i] = 0;
```

```
queue[0] = head;
```

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

```
{
```

```
    diff = abs(queue[j + 1] - queue[j]);
```

```
    seek += diff;
```

```
    printf("Disk head moves from %d to %d with seek %d\n", queue[j],
queue[j + 1], diff);
```

```
}
```

```
printf("Total seek time is %d\n", seek);
```

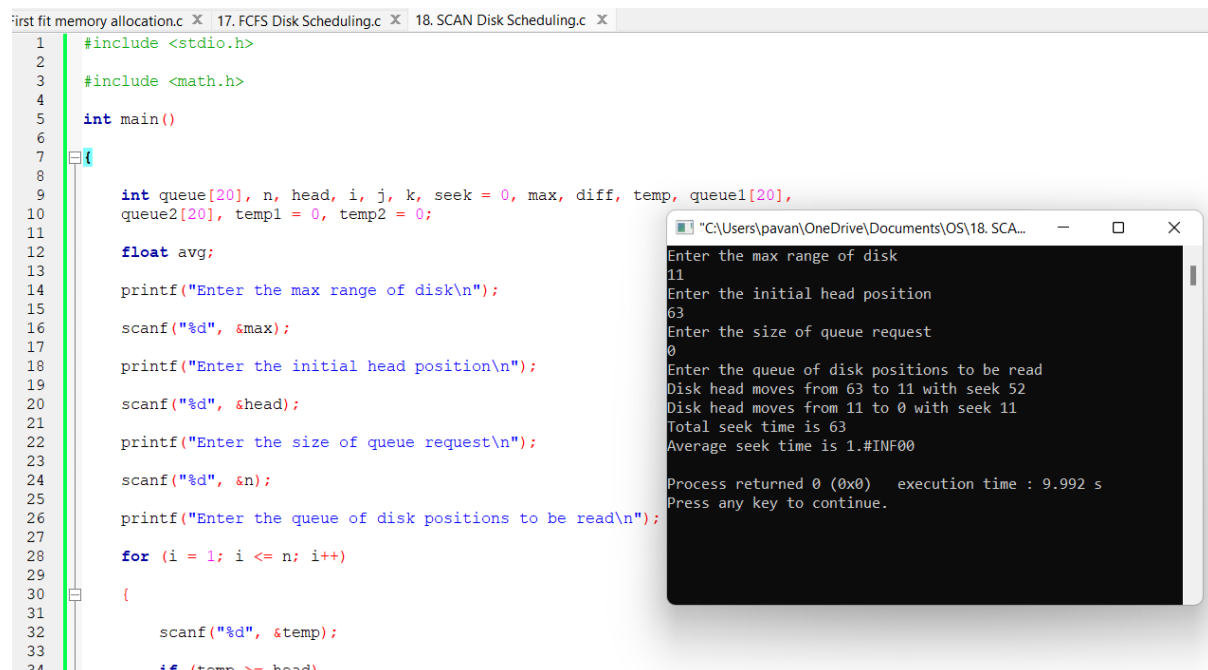
```
avg = seek / (float)n;
```

```
printf("Average seek time is %f\n", avg);
```

```
return 0;
```

```
}
```

Output :-



The screenshot shows a C++ IDE with three tabs: 'first fit memory allocation.c', '17. FCFS Disk Scheduling.c', and '18. SCAN Disk Scheduling.c'. The active tab is '18. SCAN Disk Scheduling.c', which contains the following code:

```
1  #include <stdio.h>
2
3  #include <math.h>
4
5  int main()
6
7  {
8
9      int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],
10     queue2[20], temp1 = 0, temp2 = 0;
11
12     float avg;
13
14     printf("Enter the max range of disk\n");
15
16     scanf("%d", &max);
17
18     printf("Enter the initial head position\n");
19
20     scanf("%d", &head);
21
22     printf("Enter the size of queue request\n");
23
24     scanf("%d", &n);
25
26     printf("Enter the queue of disk positions to be read\n");
27
28     for (i = 1; i <= n; i++)
29     {
30
31         scanf("%d", &temp);
32
33         if (temp >= head)
```

The output window shows the following text:

```
Enter the max range of disk
11
Enter the initial head position
63
Enter the size of queue request
0
Enter the queue of disk positions to be read
Disk head moves from 63 to 11 with seek 52
Disk head moves from 11 to 0 with seek 11
Total seek time is 63
Average seek time is 1.#INF00

Process returned 0 (0x0)   execution time : 9.992 s
Press any key to continue.
```

19. Single level directory :-

Program :-

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

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

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

```
struct
```

```
{
```

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



```

int fcnt;

}dir;

void main()

{
int i,ch;
char f[30];
dir.fcnt = 0;
printf("\nEnter name of directory -- ");
scanf("%s", dir.dname);
while(1)
{
printf("\n\n1. Create File\t2. Delete File\t3. Search File \n 4. Display Files\t5. Exit\nEnter
your choice -- ");
scanf("%d",&ch);
switch(ch)
{
case 1: printf("\nEnter the name of the file -- ");
scanf("%s",dir.fname[dir.fcnt]);
dir.fcnt++;
break;
case 2: printf("\nEnter the name of the file -- ");
scanf("%s",f);
for(i=0;i<dir.fcnt;i++)
{
if(strcmp(f, dir.fname[i])==0)
{
printf("File %s is deleted ",f);
strcpy(dir.fname[i],dir.fname[dir.fcnt-1]); break; } }
if(i==dir.fcnt) printf("File %s not found",f);
else
dir.fcnt--;

```

```

break;

case 3: printf("\nEnter the name of the file -- ");
scanf("%s",f);
for(i=0;i<dir.fcnt;i++)
{
if(strcmp(f, dir.fname[i])==0)
{
printf("File %s is found ", f);
break;
}
}
if(i==dir.fcnt)
printf("File %s not found",f);
break;

case 4: if(dir.fcnt==0)
printf("\nDirectory Empty");
else
{
printf("\nThe Files are -- ");
for(i=0;i<dir.fcnt;i++)
printf("\t%s",dir.fname[i]);
}
break;

default: exit(0);
}
}
}

```

Output :-

The screenshot displays a code editor with a C program titled "19. Single level directory.c - Code::Blocks 20.03". The program implements a menu-driven system for file management. The code includes headers for `stdio.h`, `string.h`, and `stdlib.h`. It defines a `struct` for directory entries with `fname` and `fcnt` fields. The `main` function uses a `do-while` loop to present a menu with five options: Create File, Delete File, Search File, Display Files, and Exit. Depending on the user's choice, the program performs the corresponding action, such as scanning the directory for files, deleting a file, or displaying the current directory contents. The execution output on the right shows the program's behavior as the user interacts with the menu, successfully creating, deleting, and displaying files.

```

27 scanf("%s", f);
28 for(i=0; i<dir.fcnt; i++)
29 {
30     if(strcmp(f, dir.fname[i]) == 0)
31     {
32         printf("File %s is deleted ", f);
33         strcpy(dir.fname[i], dir.fname[dir.fcnt-1]); break; } }
34     if(i==dir.fcnt) printf("File %s not found", f);
35     else
36         dir.fcnt--;
37     break;
38     case 3: printf("\nEnter the name of the file -- ");
39     scanf("%s", f);
40     for(i=0; i<dir.fcnt; i++)
41     {
42         if(strcmp(f, dir.fname[i]) == 0)
43         {
44             printf("File %s is found ", f);
45             break;
46         }
47     }
48     if(i==dir.fcnt)
49         printf("File %s not found", f);
50     break;
51     case 4: if(dir.fcnt == 0)
52         printf("\nDirectory Empty");
53     else
54     {
55         printf("\nThe Files are -- ");
56         for(i=0; i<dir.fcnt; i++)
57             printf("\t%s", dir.fname[i]);
58     }
59     break;
60     default: exit(0);
61 }
62 }
63 }

```

Execution Output:

```

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 1
Enter the name of the file -- Pavani

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 4
The Files are -- Pavani

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 1
Enter the name of the file -- Kalyani

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 4
The Files are -- Pavani Kalyani

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 3
Enter the name of the file -- Pavani
File Pavani is found

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 2
Enter the name of the file -- Kalyani
File Kalyani is deleted

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 4
The Files are -- Pavani

1. Create File 2. Delete File 3. Search File
4. Display Files 5. Exit
Enter your choice -- 5

```

20. Two level directory structure :-

Program :-

```

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

struct

{

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

int fcnt;

}dir[10];

void 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\t",dir[i].dname);
for(k=0;k<dir[i].fcnt;k++)
printf("\t%s",dir[i].fname[k]);
}
}
break;
default:exit(0);
}
}

```

}

Output :-

The image shows a C++ IDE with multiple tabs. The active tab is '20. Two level directory.c'. The code is as follows:

```
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 }
10 void main()
11 {
12     int i,ch,dcnt,k;
13     char f[30],d[30];
14     dcnt=0;
15     while(1)
16     {
17         printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");
18         printf("\n\n4. Search File\t5. Display\t6. Exit\tEnter your choice -- ");
19         scanf("%d",&ch);
20         switch(ch)
21         {
22             case 1: printf("\nEnter name of directory -- ");
23                     scanf("%s", dir[dcnt].dname);
24                     dir[dcnt].fcnt=0;
25                     dcnt++;
26                     printf("Directory created");
27                     break;
28             case 2: printf("\nEnter name of the directory -- ");
29                     scanf("%s",d);
30                     for(i=0;i<dcnt;i++)
31                     if(strcmp(d,dir[i].dname)==0)
32                     {
33                         printf("Enter name of the file -- ");
34                         scanf("%s",dir[i].fname[dir[i].fcnt]);
35                         printf("File created");
36                     }
37         }
38     }
39 }
```

The terminal window shows the execution of the program. The user enters 'Aswini' as the directory name and 'Pavani' as the file name. The program creates the directory and file, and then displays the directory listing.

```
Enter name of directory -- Aswini
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 -- Aswini
Enter name of the file -- Pavani
File created
1. Create Directory    2. Create File    3. Delete File
4. Search File        5. Display      6. Exit Enter your choice -- 5

Directory            Files
Aswini
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 -- Aswini
Enter the name of the file -- Pavani
File Pavani not found
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 : 70.687 s
Press any key to continue.
```