CSA0465 – OPERATING SYSTEMS FOR HANDLING DEADLOCKS

LAB EXPERIMENTS – Slot B

Name :- D. Siva prasad Reddy

Reg no :- 192011401

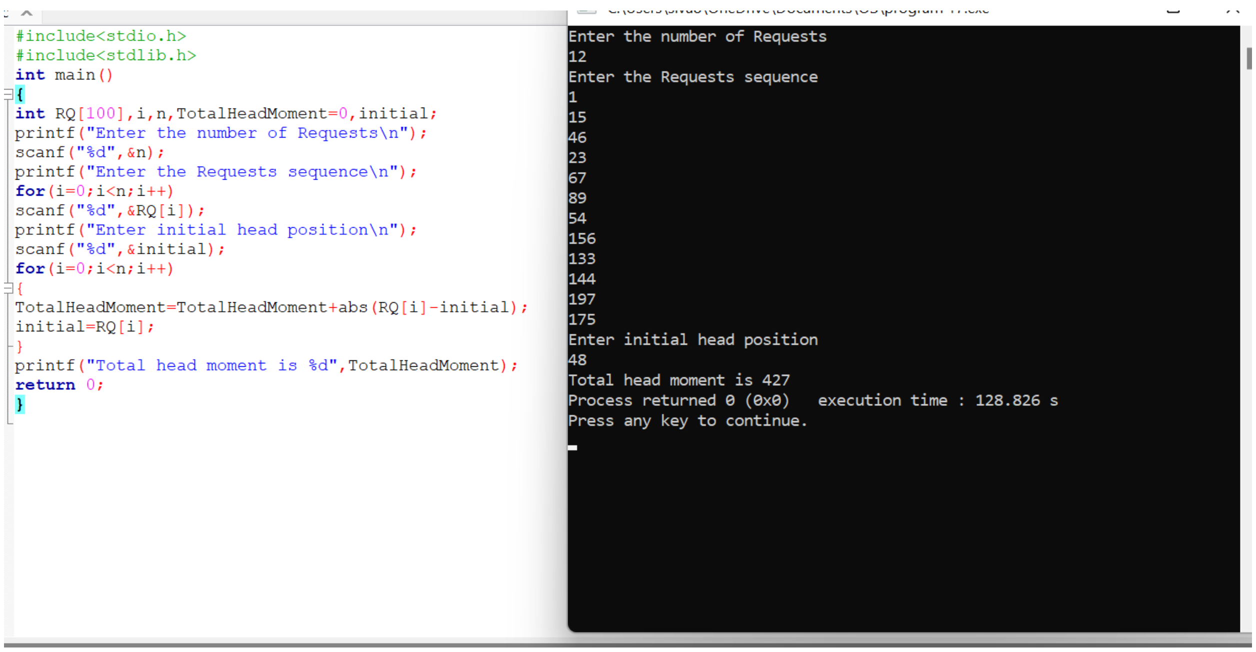
16. First Fit Memory Allocation:-

```
#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++)
```

```
        for(j = 0; j < bno; j++)
            if(flags[j] == 0 && bsize[j] >= psize[i])
            {
                allocation[j] = i;
                flags[j] = 1;
                break;
            }

    printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
    for(i = 0; i < bno; i++)
    {
        printf("\n%d\t\t%d\t\t", i+1, bsize[i]);
        if(flags[i] == 1)
            printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]);
        else
            printf("Not allocated");
    }
}
```

17. Construct a C program to simulate the Least Recently Used paging technique of memory management.

```
#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;

}
```



```
#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;
}
```

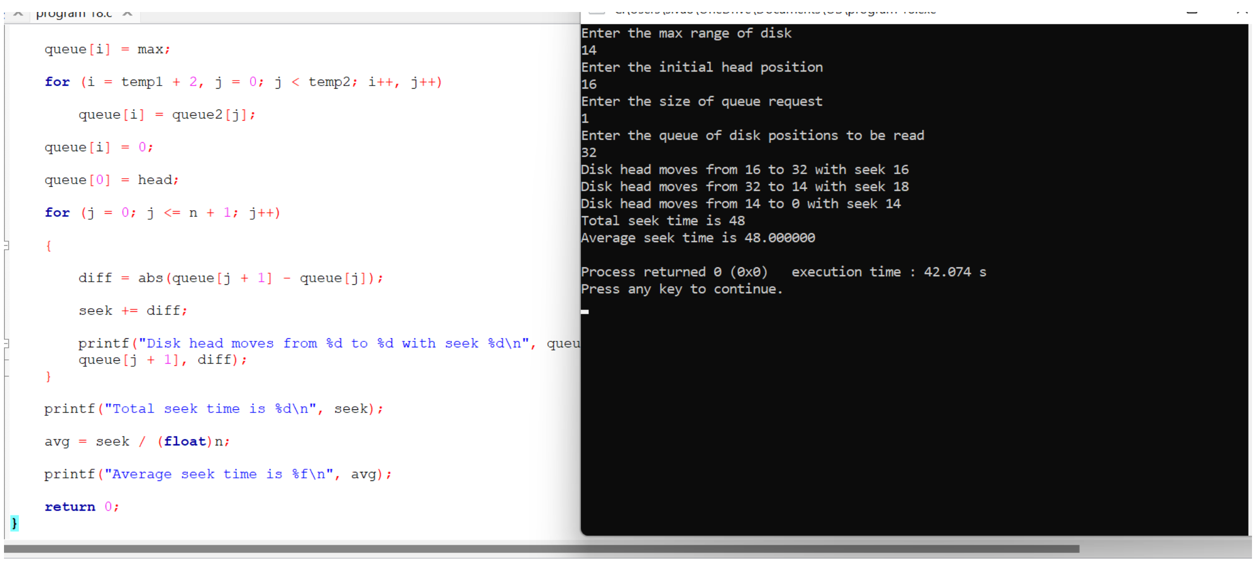
```
Enter the number of Requests
12
Enter the Requests sequence
1
15
46
23
67
89
54
156
133
144
197
175
Enter initial head position
48
Total head moment is 427
Process returned 0 (0x0)   execution time : 128.826 s
Press any key to continue.
```

18. SCAN Disk Scheduling :-

```
#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;

}
```



```
    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", queu
        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;
}
```

```
Enter the max range of disk
14
Enter the initial head position
16
Enter the size of queue request
1
Enter the queue of disk positions to be read
32
Disk head moves from 16 to 32 with seek 16
Disk head moves from 32 to 14 with seek 18
Disk head moves from 14 to 0 with seek 14
Total seek time is 48
Average seek time is 48.000000

Process returned 0 (0x0)   execution time : 42.074 s
Press any key to continue.
```

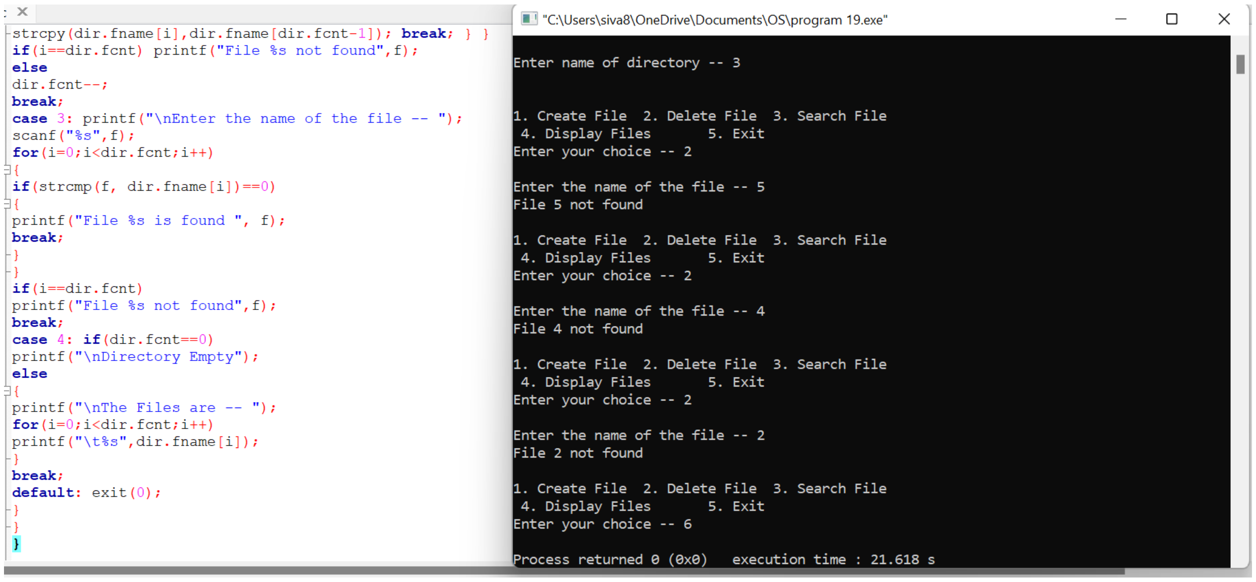
19. Single level directory :-

```
#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);
```

```
}

}

}
```



```
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);
}
}
}
```

"C:\Users\siva8\OneDrive\Documents\OS\program 19.exe"

```
Enter name of directory -- 3


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

Enter the name of the file -- 5
File 5 not 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 -- 4
File 4 not 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 -- 2
File 2 not found

1. Create File  2. Delete File  3. Search File
 4. Display Files       5. Exit
Enter your choice -- 6

Process returned 0 (0x0)   execution time : 21.618 s
```

20. Two level directory structure

```
 #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\t\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);

}

}

}
```