

LAB 8

Write a C program to simulate the following contiguous memory allocation techniques

first fit, best fit, worst fit

16/8/23

Write a C program to simulate deadlock detection
Write a C program to simulate the following
contiguous memory allocation techniques.

a) worst fit
b) best fit
c) first fit

1) worst fit

```
#include <stdio.h>

void worst_fit (int blocksize [], int blocks, int processsize [],
               int process)
{
    int allocation[process];
    int occupied[blocks];
    for (int i=0; i<process; i++) {
        allocation[i] = -1;
    }
    for (int i=0; i<blocks; i++) {
        occupied[i] = 0;
    }
    for (int i=0; i<process; i++) {
        int indexPlaced = -1;
        for (int j=0; j<blocks; j++) {
            if (blocksize[j] >= processsize[i] && !occupied[j])
            {
                if (indexPlaced == -1)
                    indexPlaced = j;
            }
            else if (blocksize[indexPlaced] < blocksize[j])
                indexPlaced = j;
        }
        allocation[i] = indexPlaced;
    }
}
```

```

if (indexPlaced != -1) {
    allocation[i] = indexPlaced;
    occupied[indexPlaced] = 1;
    blockSize[indexPlaced] = processSize[i];
}
printf("In Process No. 1st Process Size 1st block no. \n");
for (int i=0; i < process; i++) {
    printf("%d 1st 1st 1st %d 1st 1st", i+1,
        processSize[i]);
    if (allocation[i] != -1)
        printf("%d \n", allocation[i] + 1);
    else
        printf("Not Allocated \n");
}
}

```

```

int main() {
    int i, blocks, process;
    printf("Enter the no. of blocks: ");
    scanf("%d", &blocks);
    int blockSize[blocks];
    printf("Enter size of each blocks: ");
    for (int i=0; i < blocks; i++)
        scanf("%d", &processSize[i]);
    printf("Enter size of each process: ");
    for (i=0; i < process; i++)
        scanf("%d", &processSize[i]);
    worstFit(blockSize, blocks, processSize, process);
    return 0;
}

```

Output:

Enter no. of blocks : 3
Enter size of each block: 5 & 1
Enter no. of processes : 2
Enter size of each process: 1 & 4
Process No. Process size Block No.
1 1 3
2 4 1

6. // best fit :

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

```
#define MAX 10
```

```
void BestFit (int blocksize[], int blocks, int processsize[],
```

```
int processes, int m) {
```

```
int allocation [processes];
```

```
int occupied [blocks];
```

```
for (int i=0; i<processes; i++) {
```

```
allocation[i] = -1;
```

```
}
```

```
for (int k=0; k<blocks; k++) {
```

```
occupied[k] = 0;
```

```
}
```

```
for (int i=0; i<processes; i++) {
```

```
int indexproce = -1;
```

```
for (int j=0; j<blocks; j++) {
```

```
if (blocksizes[j] >= processsize[i] && !occupied[j]) {
```

```
if (indexPlaced == -1)
```

```
indexPlaced = j;
```

```
else if (blocksizes[j] < blocksizes[indexPlaced])
```

```
indexPlaced = j;
```

```
}
```

```
}
```

```
if (indexPlaced != -1)
```


Date: _____
Page: _____

```

    {
        allocation[i] = indexPlaced;
        occupied[indexPlaced] = 1;
    }
}

printf("In process No. 1 to process size 1 to block no. 1n");
for (int i=0; i<process; i++) {
    printf("%d 1 to 1 to %d 1 to 1", i+1, process[i]);
    if (allocation[i] != -1)
        printf("%d 1n", allocation[i] + 1);
    else
        printf("Not allocated 1n");
}

int main()
{
    int p, m, j;
    printf("Enter the number of processes and blocks:");
    scanf("%d %d", &p, &m);
    int processsize[p], blocksize[m];
    printf("Enter the process size:");
    for (j=0; j<p; j++)
        scanf("%d", &processsize[j]);
    printf("Enter the block size:");
    for (j=0; j<m; j++)
        scanf("%d", &blocksize[j]);
    int blocks = size of (blocksize) / size of (blocksize[0]);
    int process = size of (processsize) / size of (processsize[0]);
    Bestfit(blocksize, blocks, processsize, process, m);
    return 0;
}

```

"C:\Users\ysrmo\OneDrive - Base PU College\Desktop\4thsem\CN\CN_LAB\OS\bin\Debug\OS.exe"

Memory Management Scheme - First Fit

Enter the number of blocks:5

Enter the number of files:5

Enter the size of the blocks:

Block 1:100

Block 2:200

Block 3:300

Block 4:400

Block 5:500

Enter the size of the files:

File 1:150

File 2:200

File 3:300

File 4:450

File 5:500

File_no:	File_size:	Block_no:	Block_size:	Fragment
1	150	2	200	50
2	200	3	300	100
3	300	4	400	100
4	450	5	500	50
5	500	0	7551792	-1