**Jyothiradithyan K**

**S4 CSE, 34.**

**Experiment 6**

**Date:** 19/09/21

**Aim:** Implement fixed partition contiguous memory allocation algorithms.

***Sample Input:-***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Size of blocks in main memory(4 blocks)*** | | | | ***Size of Programs in disk (3 programs)*** | | |
| *Block1* | *Block2* | *Block3* | *Block4* | *Program1* | *Program2* | *Program3* |
| 5 | 2 | 3 | 7 | 1 | 3 | 3 |

***Sample Output:-***

a) **First-fit**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Program No.* | *Program Size* | *Block No.* | *Block Size* | *Fragmentaton* |
| 1  2  3 | 1  3  3 | 1  3  4 | 5  3  7 | 4  0  4 |

b) **Best-fit**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Program No.* | *Program Size* | *Block No.* | *Block Size* | *Fragmentaton* |
| 1  2  3 | 1  3  3 | 2  3  1 | 2  3  5 | 1  0  2 |

c) **Worse-fit**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Program No.* | *Program Size* | *Block No.* | *Block Size* | *Fragmentaton* |
| 1  2  3 | 1  3  3 | 4  1  3 | 7  5  3 | 6  2  0 |

Comment on the performance of the three algorithms.

**Program Code:**

//firstfit

void firstfit(int blockSize[],int m,int processSize[],int n){

int allocation[n],flag[m],fragment[n];

int i,j;

for(i=0;i<n;i++){

allocation[i] = -1;

}

for(i=0;i<m;i++){

flag[i] = 0;

}

for(i=0;i<n;i++){

for(j=0;j<m;j++){

if(blockSize[j] >= processSize[i] && flag[j]==0){

allocation[i] = j;

fragment[i] = blockSize[j] - processSize[i];

flag[j] = 1;

break;

}

}

}

printf(" \nProgram no Program size Blockno Block size Fragmentation\n");

for(i=0;i<n;i++){

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

if(allocation[i] != -1)

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

printf("\t%d\n",fragment[i]);

}

}

//bestfit

void bestfit(int blockSize[],int m,int processSize[],int n){

int allocation[n],flag[n],fragment[n];

int i,j,best;

for(i=0;i<n;i++){

allocation[i] = -1;

}

for(i=0;i<m;i++){

flag[i] = 0;

}

for(i=0;i<n;i++){

best = -1;

for(j=0;j<m;j++){

if(blockSize[j] >= processSize[i] && flag[j]==0){

if(best==-1)

best = j;

else if(blockSize[best] > blockSize[j])

best = j;

}

}

if(best!= -1){

allocation[i] = best;

fragment[i] = blockSize[best] - processSize[i];

flag[best] = 1;

}

}

printf(" \nProgram no Program size Blockno Block size Fragmentation\n");

for(i=0;i<n;i++){

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

if(allocation[i] != -1)

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

printf("\t%d\n",fragment[i]);

}

}

//worstfit

void worstfit(int blockSize[],int m,int processSize[],int n){

int allocation[n],flag[n],fragment[n];

int i,j,worst;

for(i=0;i<n;i++){

allocation[i] = -1;

}

for(i=0;i<m;i++){

flag[i] = 0;

}

for(i=0;i<n;i++){

worst = -1;

for(j=0;j<m;j++){

if(blockSize[j] >= processSize[i] && flag[j]==0){

if(worst==-1)

worst = j;

else if(blockSize[worst] < blockSize[j])

worst = j;

}

}

if(worst!= -1){

allocation[i] = worst;

fragment[i] = blockSize[worst] - processSize[i];

flag[worst] = 1;

}

}

printf(" \nProgram no Program size Blockno Block size Fragmentation\n");

for(i=0;i<n;i++){

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

if(allocation[i] != -1)

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

printf("\t%d\n",fragment[i]);

}

}

void main(){

int m,n,i;

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

scanf("%d",&n);

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

scanf("%d",&m);

int blockSize[m];

int processSize[n];

printf("\nEnter the size of blocks\n");

for(i=0;i<m;i++){

printf("Block %d: ",(i+1));

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

}

printf("\nEnter the size of processes\n");

for(i=0;i<n;i++){

printf("Process %d: ",(i+1));

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

}

printf("\nFirst fit allocation");

firstfit(blockSize,m,processSize,n);

printf("\nBest fit allocation");

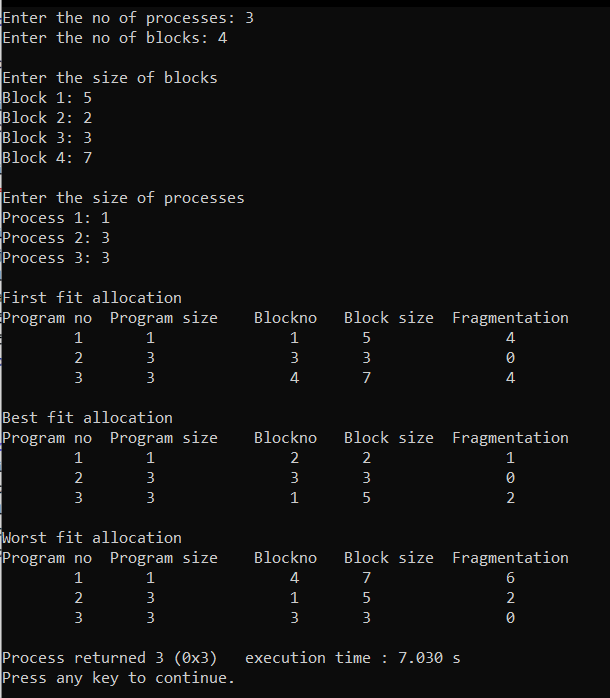
bestfit(blockSize,m,processSize,n);

printf("\nWorst fit allocation");

worstfit(blockSize,m,processSize,n);

}

**Output**



**Comments on the performance**

* First fit and best fit are more popular
* First fit is more faster
* Best fit can save larger blocks which can be later used to service large requests
* Best fit is slower as it has to search the entire list
* Best fit tends to create very small blocks, which are often unusable
* Worst fit prevents the rate of production of small blocks