EXPERIMENT – 9 IMPLEMENTATION OF BEST-FIT MEMORY ALLOCATION TECHNIQUES

AIM:

To write a C program to to simulate Best-fit contiguous memory allocation techniques

ALGORITHM

Step 1: Start

Step 2: Input memory blocks and processes with sizes.

Step 3: Initialize all memory blocks as free.

Step 4: Start by picking each process and find the minimum block size that can be assigned to current

process, if found then assign it to the current process.

Step 5: If not then leave that process and keep checking the further processes.

Step 6: Stop

PROGRAM

#include<stdio.h>

#define MAX 20

struct BLOCK {

int size;

int isallocated;

int pallocated;

int frag;

} block[MAX];

void main() {

int np, nb, psize[MAX], i, j, k, best;

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

scanf("%d",&nb);

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

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

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

scanf("%d",&block[i].size);

block[i].isallocated = 0;

block[i].pallocated = -1;

block[i].frag = block[i].size;

}

printf("\nEnter the no: of processes: ");

scanf("%d",&np);

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

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

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

scanf("%d",&psize[j]);

}

//Best Fit Memory Allocation

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

best = -1;

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

if(block[i].isallocated==0 && block[i].size >= psize[j]) {

if(best == -1)

best = i;

else if(block[i].size < block[best].size)

best = i;

}

}

if(best!=-1) {

printf("\nProcess %d is allocated to block %d", j+1, best+1);

block[best].frag = block[best].size - psize[j];

block[best].isallocated = 1;

block[best].pallocated = j+1;

}

}

printf("\n\nBlock\tIsallocated\tProcess\_allocated\tBlockSize\tPSize\tFragment\n");

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

printf("%d\t%d\t\t%d\t\t\t%d\t\t%d\t%d\n",i+1, block[i].isallocated, block[i].pallocated, block[i].size, psize[block[i].pallocated], block[i].frag);

}

}

OUTPUT

Enter the no: of blocks: 5

Enter the size of the blocks

Block 1: 100

Block 2: 50

Block 3: 30

Block 4: 120

Block 5: 35

Enter the no: of processes: 4

Enter the size of the processes

Process 1: 40

Process 2: 10

Process 3: 30

Process 4: 60

Process 1 is allocated to block 2

Process 2 is allocated to block 3

Process 3 is allocated to block 5

Process 4 is allocated to block 1

Block Isallocated Process\_allocated BlockSize PSize Fragment

1 1 4 100 60 40

2 1 1 50 10 10

3 1 2 30 30 20

4 0 -1 120 -1 120

5 1 3 35 60 5

EXPERIMENT – 10 IMPLEMENTATION OF WORST-FIT MEMORY ALLOCATION TECHNIQUES

Write a C program to implement WORST-FIT MEMORY ALLOCATION

AIM:

To write a C program to to simulate Worst-fit contiguous memory allocation techniques

ALGORITHM:

Step 1: Start

Step 2: Input memory blocks and processes with sizes.

Step 3: Initialize all memory blocks as free.

Step 4: Start by picking each process and find the maximum block size that can be assigned to

current process, if found then assign it to the current process.

Step 5: If not then leave that process and keep checking the further processes.

Step 6: Stop

PROGRAM

#include<stdio.h>

#define MAX 20

struct BLOCK

{

int size;

int isallocated;

int pallocated;

int frag;

} block[MAX];

void main()

{

int np, nb, psize[MAX], i, j, k, worst;

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

scanf("%d",&nb);

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

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

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

scanf("%d",&block[i].size);

block[i].isallocated = 0;

block[i].pallocated = -1;

block[i].frag = block[i].size;

}

printf("\nEnter the no: of processes: ");

scanf("%d",&np);

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

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

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

scanf("%d",&psize[j]);

}

//Worst Fit Memory Allocation

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

worst = -1;

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

if(block[i].isallocated==0 && block[i].size >= psize[j]) {

if(worst == -1)

worst = i;

else if(block[i].size > block[worst].size)

worst = i;

}

}

if(worst!=-1) {

printf("\nProcess %d is allocated to block %d",j+1,worst+1);

block[worst].frag = block[worst].size - psize[j];

block[worst].isallocated = 1;

block[worst].pallocated = j+1;

}

}

printf("\n\nBlock\tIsallocated\tProcess\_allocated\tBlockSize\tPSize\tFragment\n");

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

printf("%d\t%d\t\t%d\t\t\t%d\t\t%d\t%d\n",i+1, block[i].isallocated,block[i].pallocated, block[i].size, psize[block[i].pallocated], block[i].frag);

}

}

OUTPUT

Enter the no: of blocks: 5

Enter the size of the blocks

Block 1: 100

Block 2: 50

Block 3: 30

Block 4: 120

Block 5: 35

Enter the no: of processes: 4

Enter the size of the processes

Process 1: 40

Process 2: 10

Process 3: 30

Process 4: 60

Process 1 is allocated to block 4

Process 2 is allocated to block 1

Process 3 is allocated to block 2

Block Isallocated Process\_allocated BlockSize PSize Fragment

1 1 2 100 10 90

2 1 3 50 30 20

3 0 -1 30 -1 30

4 1 1 120 40 80

5 0 -1 35 -1 35

EXPERIMENT – 11 IMPLEMENTATION OF FIRST-FIT MEMORY ALLOCATION TECHNIQUES

write a C program to implement FIRST-FIT MEMORY ALLOCATION

AIM:

To write a C program to simulate First-fit contiguous memory allocation techniques

ALGORITHM:

Step 1: Start

Step 2: Input memory blocks with size and processes with size.

Step 3: Initialize all memory blocks as free.

Step 4: Start by picking each process and check if it can be assigned to the current block.

Step 5: If size-of-process <= size-of-block if yes then assign and check for next process.

Step 6: If not then keep checking the further blocks.

Step 6: Stop

PROGRAM

#include<stdio.h>

#define MAX 20

struct BLOCK {

int size;

int isallocated;

int pallocated;

int frag;

} block[MAX];

void main() {

int np, nb, i, psize[MAX], j;

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

scanf("%d",&nb);

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

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

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

scanf("%d",&block[i].size);

block[i].isallocated = 0;

block[i].pallocated = -1;

block[i].frag = block[i].size;

}

printf("\nEnter the no: of processes: ");

scanf("%d",&np);

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

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

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

scanf("%d",&psize[j]);

}

//First Fit Memory Allocation

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

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

printf("\nComparing process %d with block %d",j+1 ,i+1);

if(block[i].isallocated==0 && block[i].size >= psize[j]) {

printf("\nProcess %d is allocated to block %d\n",j+1 ,i+1);

block[i].frag = block[i].size - psize[j];

block[i].isallocated = 1;

block[i].pallocated = j+1;

break;

}

}

}

printf("\n\nBlock\tIsallocated\tProcess\_allocated\tBlockSize\tPSize\tFragment\n");

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

printf("%d\t%d\t\t%d\t\t\t%d\t\t%d\t%d\n",i+1, block[i].isallocated, block[i].pallocated, block[i].size, psize[block[i].pallocated], block[i].frag);

}

}

OUTPUT

Enter the no: of blocks: 5

Enter the size of the blocks

Block 1: 20

Block 2: 100

Block 3: 40

Block 4: 200

Block 5: 10

Enter the no: of processes: 4

Enter the size of the processes

Process 1: 90

Process 2: 50

Process 3: 30

Process 4: 40

Comparing process 1 with block 1

Comparing process 1 with block 2

Process 1 is allocated to block 2

Comparing process 2 with block 1

Comparing process 2 with block 2

Comparing process 2 with block 3

Comparing process 2 with block 4

Process 2 is allocated to block 4

Comparing process 3 with block 1

Comparing process 3 with block 2

Comparing process 3 with block 3

Process 3 is allocated to block 3

Comparing process 4 with block 1

Comparing process 4 with block 2

Comparing process 4 with block 3

Comparing process 4 with block 4

Comparing process 4 with block 5

Block Isallocated Process\_allocated BlockSize PSize Fragment

1 0 -1 20 -1 20

2 1 1 100 90 10

3 1 3 40 30 10

4 1 2 200 50 150

5 0 -1 10 -1 10