Experiment 8

Subject: CSL403 Operating System Lab

NAME: GINI CHACKO

ROLL: 8942

CLASS: SE COMPS B

BATCH: B

Aim: Study Memory Management

Objectives: Implement various Memory Management techniques and evaluate their performance.

Problem Statement:

Implement Dynamic Partitioning Placement Algorithms

(a)Best Fit

(b) First-Fit

(c)Worst-Fit

1.Given the number of holes and their sizes, number of blocks to be placed in memory and their sizes, find which algorithm would be resulting in effective utilization of memory.

2.Give the allotment of blocks to holes in each algorithm

Answer:

1.Given the number of holes and their sizes, number of blocks to be placed in memory and their sizes, find which algorithm would be resulting in effective utilization of memory.

Ans :

A.] BEST-FIT

Process_no	Process_size	Block_no	Block_size	Fragment
1	357	2	400	43
2	210	6	250	40
3	468	4	500	32
4	491	3	600	109

B.] FIRST-FIT

Process_no.	Process Size	Block Number	Block Size	Fragment
1	357	2	400	43
2	210	3	600	390
3	468	4	500	32
4	491	1	200	-241

C.] WORST-FIT

Process No.	Process Size	Block Number	Block Size	Fragment
1	357	3	600	243
2	210	4	500	290
3	468	1	200	0
4	491	1	200	0

In case of Best Fit :
Internal Fragmentation = 224

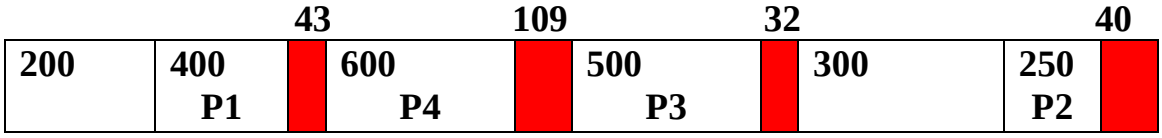
In case of First Fit :
Internal Fragmentation = 465

In case of Worst Fit :
Internal Fragmentation = 535

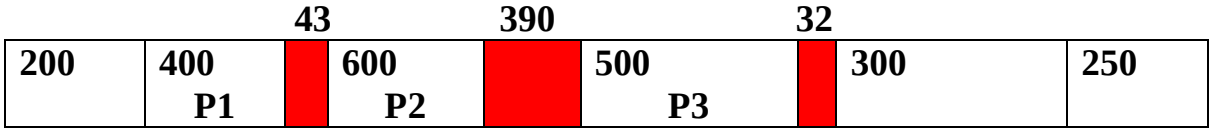
From the above results, after comparing the internal fragmentation of best fit, first fit and worst fit, we can conclude that the best fit would be resulting in effective utilization of memory.

2.Give the allotment of blocks to holes in each algorithm

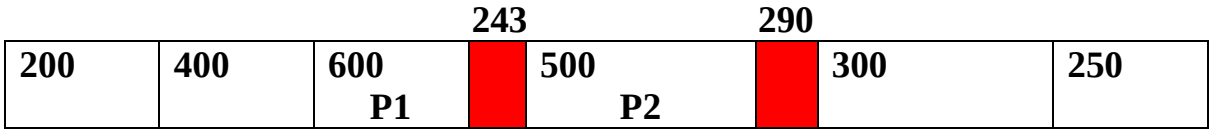
A.] BEST-FIT



B.] FIRST-FIT



C.] WORST-FIT



Program Section:

A.] BEST-FIT

CODE:

```
#include<stdio.h>

void main()
{
        int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999;
        static int barray[20],parray[20];
```

```
        printf("\n****** Memory Management Scheme - Best Fit ******\n");
        printf("\nEnter the number of blocks : ");
        scanf("%d",&nb);
        printf("Enter the number of processes : ");
        scanf("%d",&np);

        printf("\nEnter the size of the blocks : \n");
        for(i=1;i<=nb;i++)
  {
                printf("Block no.%d : ",i);
    scanf("%d",&b[i]);
  }

        printf("\nEnter the size of the processes : \n");
        for(i=1;i<=np;i++)
  {
    printf("Process no.%d : ",i);
    scanf("%d",&p[i]);
  }

        for(i=1;i<=np;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(barray[j]!=1)
                        {
                                temp=b[j]-p[i];
                                if(temp>=0)
                                        if(lowest>temp)
                                        {
                                                parray[i]=j;
                                                lowest=temp;
                                        }
                        }
                }

                fragment[i]=lowest;
                barray[parray[i]]=1;
                lowest=10000;
        }

        printf("\nProcess_no\tProcess_size\tBlock_no\tBlock_size\tFragment\n");
        for(i=1;i<=np && parray[i]!=0;i++)
                printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\n\n",i,p[i],parray[i],b[parray[i]],fragment[i]);
}
```

B.] FIRST-FIT

CODE:

```
#include<stdio.h>

int main()
{
    static int block_arr[10], file_arr[10];
    int fragments[10], blocks[10], files[10];
    int m, n, number_of_blocks, number_of_files, temp;

    printf("\n****** Memory Management Scheme - First Fit ******\n");
    printf("\nEnter the Total Number of Blocks : ");
    scanf("%d", &number_of_blocks);
    printf("Enter the Total Number of Process : ");
    scanf("%d", &number_of_files);
    printf("\nEnter the Size of the Blocks : \n");
    for(m = 0; m < number_of_blocks; m++)
    {
        printf("Block No.[%d] : \t", m + 1);
        scanf("%d", &blocks[m]);
    }
    printf("\nEnter the Size of the Process : \n");
    for(m = 0; m < number_of_files; m++)
    {
        printf("Process No.[%d] : ", m + 1);
        scanf("%d", &files[m]);
    }
    for(m = 0; m < number_of_files; m++)
    {
        for(n = 0; n < number_of_blocks; n++)
        {
            if(block_arr[n] != 1)
            {
                temp = blocks[n] - files[m];
                if(temp >= 0)
                {
                    file_arr[m] = n;
                    break;
                }
            }
        }
        fragments[m] = temp;
        block_arr[file_arr[m]] = 1;
    }
    printf("\nProcess_no.\tProcess Size\tBlock Number\tBlock Size\tFragment\n");
    for(m = 0; m < number_of_files; m++)
    {
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\n\n", m+1, files[m], file_arr[m]+1,  blocks[file_arr[m]],
fragments[m]);
    }
    printf("\n");
    return 0;
}
```

C.] WORST-FIT

CODE:

```
#include<stdio.h>

int main()
{
    int fragments[10], blocks[10], files[10];
    int m, n, number_of_blocks, number_of_files, temp, top = 0;
    static int block_arr[10], file_arr[10];

    printf("\n****** Memory Management Scheme - Worst Fit ******\n");
    printf("\nEnter the Total Number of Blocks : ");
    scanf("%d",&number_of_blocks);
    printf("Enter the Total Number of Process : ");
    scanf("%d",&number_of_files);
    printf("\nEnter the Size of the Blocks : \n");
    for(m = 0; m < number_of_blocks; m++)
    {
        printf("Block No.[%d] : ", m + 1);
        scanf("%d", &blocks[m]);
    }
    printf("\nEnter the Size of the Process : \n");
    for(m = 0; m < number_of_files; m++)
    {
        printf("File No.[%d] : ", m + 1);
        scanf("%d", &files[m]);
    }
    for(m = 0; m < number_of_files; m++)
    {
        for(n = 0; n < number_of_blocks; n++)
        {
            if(block_arr[n] != 1)
            {
                temp = blocks[n] - files[m];
                if(temp >= 0)
                {
                    if(top < temp)
                    {
                        file_arr[m] = n;
                        top = temp;
                    }
                }
            }
            fragments[m] = top;
            block_arr[file_arr[m]] = 1;

        }
        top = 0;
    }
    printf("\nProcess No.\tProcess Size\tBlock Number\tBlock Size\tFragment\n");
```

```
    for(m = 0; m < number_of_files; m++)
    {
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\n\n", m+1, files[m], file_arr[m]+1, blocks[file_arr[m]],
fragments[m]);
    }
    printf("\n");
    return 0;
}
```

Output Section:

A.] BEST-FIT

OUTPUT:

```
gini@gini:~/Practicals/OS_LAB_8$ gcc exp_8a.c
gini@gini:~/Practicals/OS_LAB_8$ ./a.out

****** Memory Management Scheme - Best Fit ******

Enter the number of blocks : 6
Enter the number of processes : 4

Enter the size of the blocks :
Block no.1 : 200
Block no.2 : 400
Block no.3 : 600
Block no.4 : 500
Block no.5 : 300
Block no.6 : 250

Enter the size of the processes :
Process no.1 : 357
Process no.2 : 210
Process no.3 : 468
Process no.4 : 491

Process_no      Process_size    Block_no        Block_size      Fragment

1               357             2               400             43


2               210             6               250             40


3               468             4               500             32


4               491             3               600             109
```

B.] FIRST-FIT

OUTPUT:

```
gini@gini:~/Practicals/OS_LAB_8$ gcc exp_8b.c
gini@gini:~/Practicals/OS_LAB_8$ ./a.out

****** Memory Management Scheme - First Fit ******

Enter the Total Number of Blocks : 6
Enter the Total Number of Process : 4

Enter the Size of the Blocks :
Block No.[1] :  200
Block No.[2] :  400
Block No.[3] :  600
Block No.[4] :  500
Block No.[5] :  300
Block No.[6] :  250

Enter the Size of the Process :
Process No.[1] : 357
Process No.[2] : 210
Process No.[3] : 468
Process No.[4] : 491

Process_no.     Process Size    Block Number    Block Size      Fragment

1               357             2               400             43


2               210             3               600             390


3               468             4               500             32


4               491             1               200             -241
```

C.] WORST-FIT

OUTPUT:

```
gini@gini:~/Practicals/OS_LAB_8$ gcc exp_8c.c
gini@gini:~/Practicals/OS_LAB_8$ ./a.out

****** Memory Management Scheme - Worst Fit ******

Enter the Total Number of Blocks : 6
Enter the Total Number of Process : 4

Enter the Size of the Blocks :
Block No.[1] : 200
Block No.[2] : 400
Block No.[3] : 600
Block No.[4] : 500
Block No.[5] : 300
Block No.[6] : 250

Enter the Size of the Process :
Process No.[1] : 357
Process No.[2] : 210
Process No.[3] : 468
Process No.[4] : 491

Process No.     Process Size    Block Number    Block Size      Fragment

1               357             3               600             243


2               210             4               500             290


3               468             1               200             0


4               491             1               200             0
```