Vidyavardhini's College of Engineering & Technology
Department of Computer Engineering
Academic Year : 2023-24

Experiment No 8

Experiment No 8: First Fit Strategy
Date of Performance:20/03/24
Date of Submission:27/03/24
Name: Jenil Kotadia
Roll No : 19

AIM: IMPLEMENTATION MEMORY STRATEGY FIRST FIT

OBJECTIVE:The objective of implementing the First Fit memory allocation strategy is to efficiently allocate memory blocks to incoming processes or data segments in a computer system. This strategy aims to find the first available memory block that is large enough to accommodate the incoming request, thereby minimizing fragmentation and maximizing memory utilization.

THEORY :

When a new process requests memory allocation, the system searches the available memory starting from the beginning (lowest address) of the memory space. The system allocates memory to the first available block that is large enough to accommodate the process. Once a suitable block is found, the allocation is made, and the process begins execution.

ADVANATGES :

- Simple and easy to implement.
- Efficient in terms of time complexity since it searches for the first available block that satisfies the memory requirement.

DISADVANTAGE:

- May lead to memory fragmentation: As processes are allocated and deallocated, the memory can become fragmented, with small gaps of unused memory scattered throughout. This fragmentation can reduce the efficiency of memory utilization over time.
- Suboptimal space utilization: Since the first available block is chosen, it may not always be the smallest block that satisfies the memory requirement. This can lead to inefficient use of memory space, known as internal fragmentation.

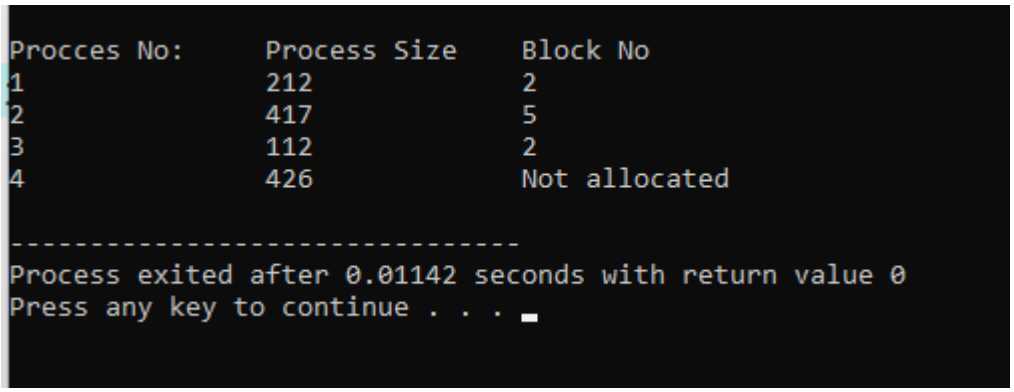
CODE:

```
#include<stdio.h>

void firstfit(int blocksize[],int m,int processsize[],int n){
        int i,j;
        int allocation[n];
        for(i=0;i<n;i++){
                allocation[i]=-1;
        }
                for(i=0;i<n;i++){
                for(j=0;j<m;j++){
```

```
                        if(blocksize[j]>=processsize[i]){
                                allocation[i]=j;
                                blocksize[j]-=processsize[i];
                                break;
                        }
                }
        }
        printf("\nProcces No:\tProcess Size\tBlock No\n");
                for(i=0;i<n;i++){
                printf("%i\t\t",i+1);
                        printf("%i\t\t",processsize[i]);
if(allocation[i]!=-1)
        printf("%i",allocation[i]+1);
else
 printf("Not allocated");
 printf("\n");
}
}
int main(){
        int m;
        int n;
        int blocksize[]={100,500,200,300,600};
        int processsize[]={212,417,112,426};
        m=sizeof(blocksize)/sizeof(blocksize[0]);
        n=sizeof(processsize)/sizeof(processsize[0]);
        firstfit(blocksize,m,processsize,n);
        return 0;
}
```

OUTPUT:



CONCLUSION :

1. Simplicity : The First Fit strategy is easy to understand and implement, making it an attractive choice for memory management systems.
2. Efficiency: It typically offers a quick allocation process by selecting the first available memory block that meets the process's requirements. This efficiency is particularly evident when compared to more complex allocation algorithms.
3. Fragmentation: However, the experiment highlighted a potential drawback of the First Fit strategy—memory fragmentation. As processes are allocated and deallocated, small gaps of unused memory can accumulate, leading to fragmentation. This fragmentation can decrease overall memory utilization efficiency and may require additional memory management techniques to mitigate its effects.
4. Space Utilization: While the First Fit strategy aims to allocate memory as soon as a suitable block is found, it may not always select the most optimal block in terms of size. This can result in suboptimal space utilization, with larger blocks being allocated when smaller ones would suffice.
5. Trade-offs: Like any memory allocation strategy, the First Fit approach involves trade-offs. While it offers simplicity and efficiency, it may struggle to maintain optimal memory usage over extended periods, especially in systems with frequent
process allocations and deallocations.