|  |
| --- |
| Experiment No.8 |
| Implement memory allocation strategy First fit. |
| Date of Performance: 12/04/2024 |
| Date of Submission: 13/04/2024 |

**Aim:**  To study and implement memory allocation strategy First fit.

**Objective:** The objective of memory allocation strategies  is to provide ways to dynamically allocate portions of memory to programs at their request, and free it for reuse when no longer needed as the main memory is limited in size.

**Theory:**

The primary role of the memory management system is to satisfy requests for memory allocation. Sometimes this is implicit, as when a new process is created. At other times, processes explicitly request memory. Either way, the system must locate enough unallocated memory and assign it to the process.

**Partitioning:** The simplest methods of allocating memory are based on dividing memory into areas with fixed partitions.

**Selection Policies:** If more than one free block can satisfy a request, then which one should we pick? There are several schemes that are frequently studied and are commonly used.

**First Fit:** In the first fit approach is to allocate the first free partition or hole large enough which can accommodate the process. It finishes after finding the first suitable free partition.

* **Advantage:** Fastest algorithm because it searches as little as possible.
* **Disadvantage:** The remaining unused memory areas left after allocation become waste if it is too smaller. Thus request for larger memory requirement cannot be accomplished.

## Best Fit: The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed.

## Worst fit: In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

Next Fit: If we want to spread the allocations out more evenly across the memory space, we often use a policy called next fit. This scheme is very similar to the first fit approach, except for the place where the search starts.

**Program :**

#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("\nProcess No. \t Process Size \t Block No.\n");

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

printf(" %i \t \t \t \t \t",i+1);

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

if(allocation[i] != -1)

printf("%i",allocation[i]+1);

else

printf("Not allocated");

printf("\n");

}

}

void main(){

int m;//Number of blocks

int n;//Number of process

int blockSize[]={100,500,200,300,600};

int processSize[]={212,417,112,426};

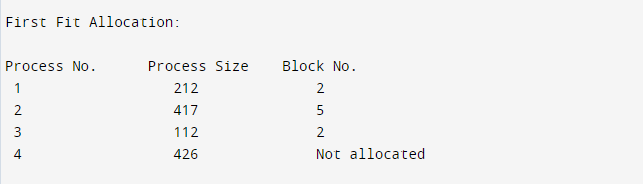
m=sizeof(blockSize)/sizeof(blockSize[0]);

n=sizeof(processSize)/sizeof(processSize[0]);

firstFit(blockSize,m,processSize,n);

}

**Result:**

****

**Conclusion:**

Internal fragmentation occurs when allocated memory to a process is larger than what the process actually needs. It leads to wasted memory space within the allocated block, reducing overall system efficiency. Efficient memory allocation techniques like segmentation and paging can help mitigate internal fragmentation, ensuring optimal use of system resources.