SSN COLLEGE OF ENGINEERING, KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CS8461 - OPERATING SYSTEM LAB

Lab Exercise 10 Implementation of Memory Management Algorithms

Problem :

This problem can be implemented using linked list or arrays.

Free space is maintained as a linked list of nodes with each node having the starting byte address and the ending byte address of a free block. Each memory request consists of the process-id and the amount of storage space required in bytes. Allocated memory space is again maintained as a linked list of nodes with each node having the process id, starting byte address and the ending byte address of the allocated space. When a process finishes (taken as input), the appropriate node from the allocated list should be deleted and this free disk space should be added to the free space list. [Care should be taken to merge contiguous free blocks into one single block. This results in deleting more than one node from the free space list and changing the start and end address in the appropriate node]. For allocation use first fit, worst fit and best fit algorithms.

Steps:

1. Get the algorithm to work with. (First Fit, Best Fit, Worst Fit)

2. Implement this using Linked List / Arrays.

3. Let the options be 1. Entry / Allocate 2. Exit / Deallocate 3. Display (Memory allocated LL, Free Pool LL, Total Physical memory by Merging 2 LL) 4.Coalescing Holes.

4. Create a node using structure with the following details. Starting address, ending address, Size, Status (contains process id or H to denote that it is a hole).

5. Create a Alloted memory Linked list with a collection of nodes (Initially no nodes).

6. Create a Free Pool Linked List with a collection of nodes (Initially all partitions will be in free pool LL).

7. Get the memory representation as input from user. No of partitions, starting address and ending address of each partition. Calculate the Size of each partition. Initially all the partitions will be in free pool LL.

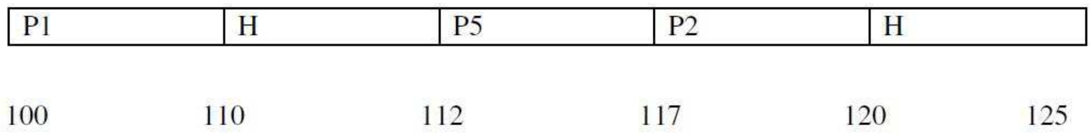
8. Now get whether a process enters / exits.

9. If ”Entry” then get the process id and its size in bytes. Using the algorithm allocate memory for it and store it in alloted memory linked list. Modify in Free pool Linked list also. [delete from free pool LL, insert into memory allocation LL]

10.If “exit” then get the process id and then deall ocate the process from memory allocation linked list and include it in free pool linked list. [delete from memory allocation LL, insert into free pool LL]

11.On display you have to display both LL separately and combined memory structure should be displayed by merging both LL.

Merged Output should be



12.For Coalescing of Holes scan the free pool LL and check for continuous holes. If so merge them as single hole by modifying ending address and size and delete the other hole.

13.For first fit select the first hole that satisfies the memory request.

14.For best fit select the hole that is large enough to accommodate the process but incurs minimum wastage of memory.

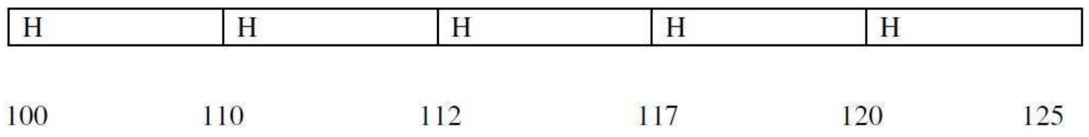
15.For worst fit select the hole that is the largest.

Sample Input and output:

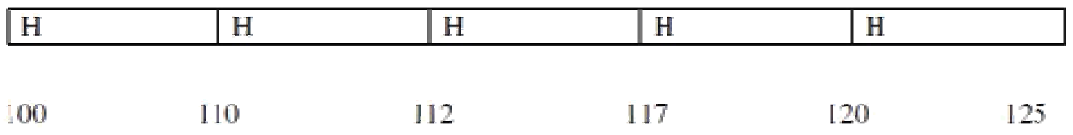
Enter the Memory Representation:
Enter the no. of partitions in memory : 5
Starting and ending address of partition 1: 100 110
Starting and ending address of partition 2: 110 112
....

Alloted Memory LL -----> Null

Free Pool >



Physical Memory--->



Memory Allocation Algorithm:
1.First Fit
2.Best Fit
3.Worst Fit
4. Exit from program
Enter choice : 1

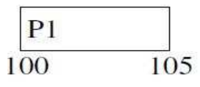
First Fit Memory Allocation Algorithm

1.Entry / Allocate
2. Exit / Deallocate
3. Display
4. Coalescing of Holes
5. Back to Algorithm

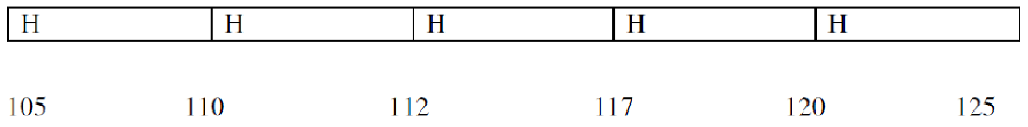
Enter choice : 1

Enter process id : P1
Enter size needed : 5
Memory is alloted to P1

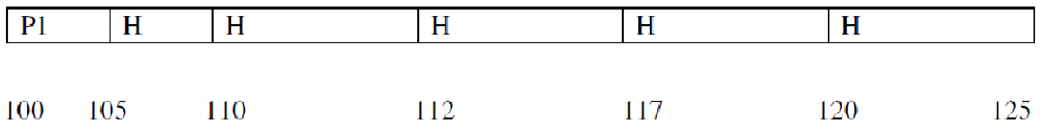
Alloted Memory LL---- >



Free Pool ------>

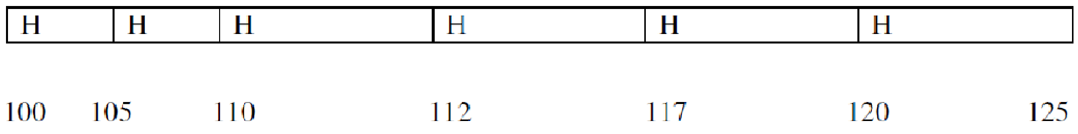


Physical Memory------>

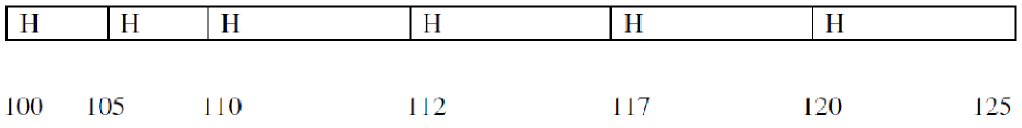


First Fit Memory Allocation Algorithm

```
1.Entry / Allocate
2. Exit / Deallocate
3. Display
4. Coalescing of Holes
5. Back to Algorithm
Enter choice : 2
Enter process id : P1

Memory is deallocated.

Alloted Memory LL ----> Null

Free Pool --
```



```
Physical Memory------>
```


H
H
H
H
H
H
100
105
110
112
117
120
125