DSA
Assignment 02

Name: Aamir Ansari **Batch:** A **Roll no:** 01

Assignment on Recursion and Storage management

Recursion

Recursion is a technique that breaks a problem into one or more sub-problems that are similar to the original problem. A recursive function is defined as a function that calls itself to solve a smaller version of its task until a final call is made which does not require a call to itself. Every recursive solution has two major cases.
They are

1. **Base case**, in which the problem is simple enough to be solved directly without making any further calls to the same function.
2. **Recursive case**, in which first the problem at hand is divided into simpler sub-parts. Second the function calls itself but with sub-parts of the problem obtained in the first step. Third, the result is obtained by combining the solutions of simpler sub-parts

Types of Recursion

1. Direct Recursion: A function is said to be directly recursive if it explicitly calls itself.
2. Indirect Recursion: A function is said to be indirectly recursive if it contains a call to an other function which ultimately calls it.
3. Tail Recursion: A recursive function is said to be tail recursive if no operations are pending to be performed when the recursive function returns to its caller. when the called function returns, the returned value is immediately returned from the calling function. Tail recursive functions are highly desirable because they are much more efficient to use as the amount of information that has to be stored on the system stack is independent of the number of recursive calls.

Sequential Fit Methods

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. This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. The o erating system doesn’t search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

Advantages:

It is fast in processing. As the processor allocates the nearest available memory partition to the job, it is very fast in execution.

Disadvantages:

It wastes a lot of memory. The processor ignores if the size of partition allocated to the job is very large as compared to the size of job or not. It just allocates the memory. As a result, a lot of memory is wasted and many jobs may not get space in the memory, and would have to wait for another job to complete.

Best Fit

The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently.

Advantages:

It is memory efficient. The operating system allocates the job minimum possible space in the memory, making memory management very efficient. To save memory from getting wasted, it is the best method.

Disadvantages:

It is a slow process. Checking the whole memory for each job makes the working of the operating system very slow. It takes a lot of time to complete the work.

Worst fit

In this allocation technique the process traverses the whole memory and always search for largest hole/partition, and then the process is placed in that hole/partition. It is a slow process because it has to traverse the entire memory to search largest hole.

Advantages:

Since this process chooses the largest hole/partition, therefore there will be large internal fragmentation. Now, this internal fragmentation will be quite big so that other small processes can also be placed in that left over partition.

Disadvantages:

It is a slow process because it traverses all the partitions in the memory and then selects the largest partition among all the partitions, which is a time-consuming process.

Fragmentation

Fragmentation is an unwanted problem that deals with memory fragment and occur due to either non-contiguous memory allocation or fixed size memory allocation. As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Types of Fragmentation

There are two types of fragmentation:

Internal Fragmentation:

Internal fragmentation happens when the memory is split into mounted sized blocks. Whenever a method requests for the memory, the mounted sized block is allotted to the method. just in case the memory allotted to the method is somewhat larger than the memory requested, then the distinction between allotted and requested memory is that the Internal fragmentation.

External Fragmentation:

External fragmentation happens when there’s a sufficient quantity of area within the memory to satisfy the memory request of a method. However, the process’s memory request cannot be fulfilled because the memory offered is during a non-contiguous manner. Either you apply first-fit or best-fit memory allocation strategy it’ll cause external fragmentation.