Operating Systems!!!

Contiguous Memory Allocation

The main purpose of computer system is to execute programs. These programs, together with the data the they access, must be at least partially in the main memory during execution. There are two methods to store data in the main memory: 1. Contiguous Memory Allocation 2. Non-Contiguous Memory Allocation.

Today I will be discussing contiguous memory allocation.

Contiguous memory allocation is a classical memory allocation model that assigns a process consecutive memory blocks (that is, memory blocks having consecutive addresses). When a process needs to execute, memory is requested by the process. The size of the process is compared with the amount of contiguous main memory available to execute the process. If sufficient contiguous memory is found, the process is allocated memory to start its execution. Otherwise, it is added to a queue of waiting processes until sufficient free contiguous memory is available.

There are two methods for allocating memory:

1. Fixed partition scheme or Static memory allocation

2. Variable partition scheme or Dynamic memory allocation

Fixed partition Scheme

1. In this scheme, the memory is divided into several fixed size partitions.
2. Each partition may contain exactly one process. Thus, the degree of multiprogramming is bound by the number of partitions.
3. When a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates the partition becomes available for another process.

Variable partition scheme

1. In this scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
2. Initially, all memory is available for user processes and is considered one large block of available memory, a **hole**.
3. The operating system takes into account the memory requirements of each process present in the input queue and the amount of available memory space in determining which processes are allocated memory.
4. When a process arrives, the system searches the set for a hole that is large enough for this process. If the hole is two large, it is split into two parts. One part is allocated to the arriving process; the other is returned to the set holes.
5. When a process terminates, it releases its block of memory, which is then placed back in the set of holes. If the new hole is adjacent to other holes, these adjacent holes are merged together to form one larger hole.

Now, let us see how to satisfy a request of size n from a list of free holes. The three of the most commonly used strategies to solve this problem are: first-fit, best-fit and worst-fit strategies.

1. **First-fit:** Allocate the first hole that is big enough. Searching can either start from the beginning of the set of holes or at the location where the previous search ended. The search ends as soon as we find a free hole that is large enough.
2. **Best-Fit**: Allocate the smallest hole that is big enough. The entire list is searched and the smallest hole that is big enough the store the data is selected.
3. **Worst-Fit:** Allocate the largest hole. The list is searched and the largest hole is selected to store the data.

In Variable size partitioning scheme, worst-fit is better than best-fit, as worst-fit produces holes of larger size that can be used by other processes whereas in best-fit holes are produced of small sizes and cannot be used by other processes.

EXTERNAL FRAGMENTATION

1. Contiguous memory allocation leads to **external fragmentation**.
2. As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when there is enough total memory space to satisfy a request but the available spaces are not contiguous; storage is fragmented into large number of smaller holes.
3. This fragmentation problem can be severe. In the worst case, we could have a block of free memory between every two processes. If all these small pieces of memory were in a big free block instead, we might be able to run several more processes.

INTERNAL FRAGMENTATION

1. It occurs when fixed sized memory blocks are allocated to the processes.
2. When the memory assigned to the process is slightly larger than the memory requested by the process this creates free space in the allocated block causing internal fragmentation.