CSC 224 Principles of OS

6. Memory Management

Memory management

- Main memory enables processes to exist.
- It is within main memory that instructions reside which is interpreted by the processor.
- In addition to holding program code, the memory is also a work space and transient storage repository for various kinds of data.
- The part of the OS that manages the memory is memory manager.

The principle problems to be handled by the OS memory manager are:-

- 1. Provide memory space to enable several processes to be executed at the same time.
- 2. Keep track of which parts of memory are in use and which are not
- 3. Allocate memory to processes when they need it and de-allocate it when they are done.

- 4. Manage swapping between main memory and disk when main memory is too small to hold all the processes.
- 5. Provide a satisfactory level of performance (i.e. process execution speed) for the system users.
- NB: Although several processes may be active and hence occupying memory space at any instant in time only one instruction is being executed.

Process loading and swapping

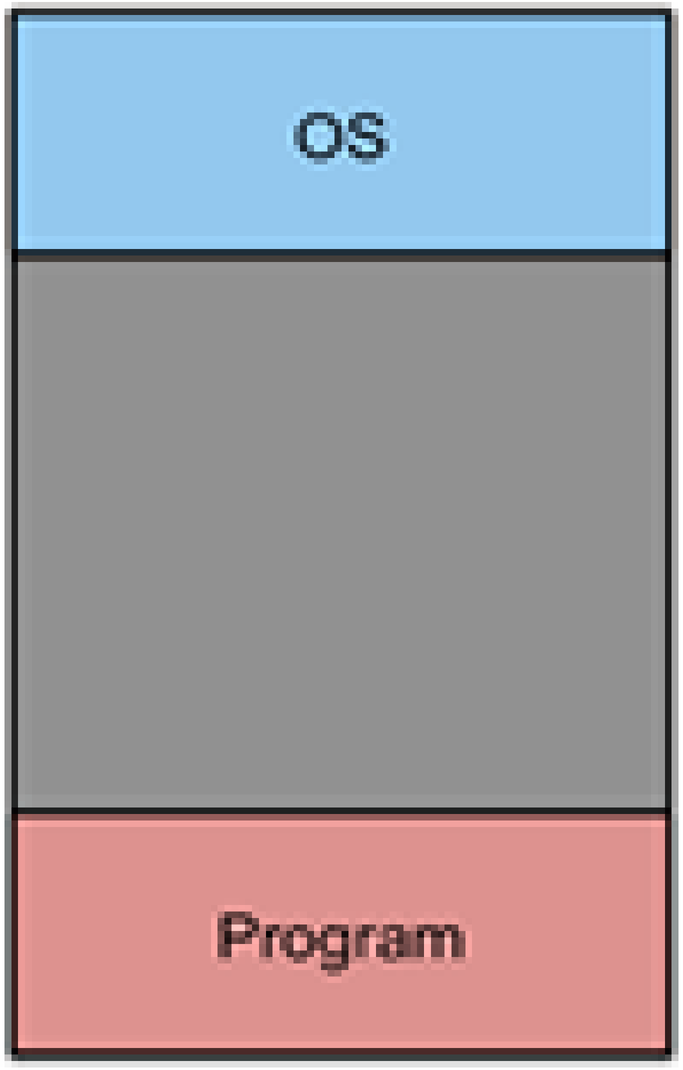
- In order to execute a process, the program code has to be transferred from secondary storage to main memory, this activity is called process **loading**.
- Transfer of a process from main memory to secondary storage in order to accommodate and execute another process is called **swapping.**
- The transfer to secondary storage must preserve the entire state of the process including current data values and the status of all registers.

MEMORY ALLOCATION TECHNIQUES/METHODS

- **1. Single Process System**
- In a computer which is only intended to run one process at a time memory management is simple.
- The process to be executed is loaded into the free space area of the memory. A part of memory space will be unused.

- Only one process can be loaded into memory at any given time, making it limited in capability; thus the rest of the space is wasted.
- The process effectively takes over the memory space of the machine, with a portion of memory devoted to the operating system
- Early computer systems and early microcomputer operating systems adopted this solution. For example, MS-DOS

Single partitioning/Monoprogramming

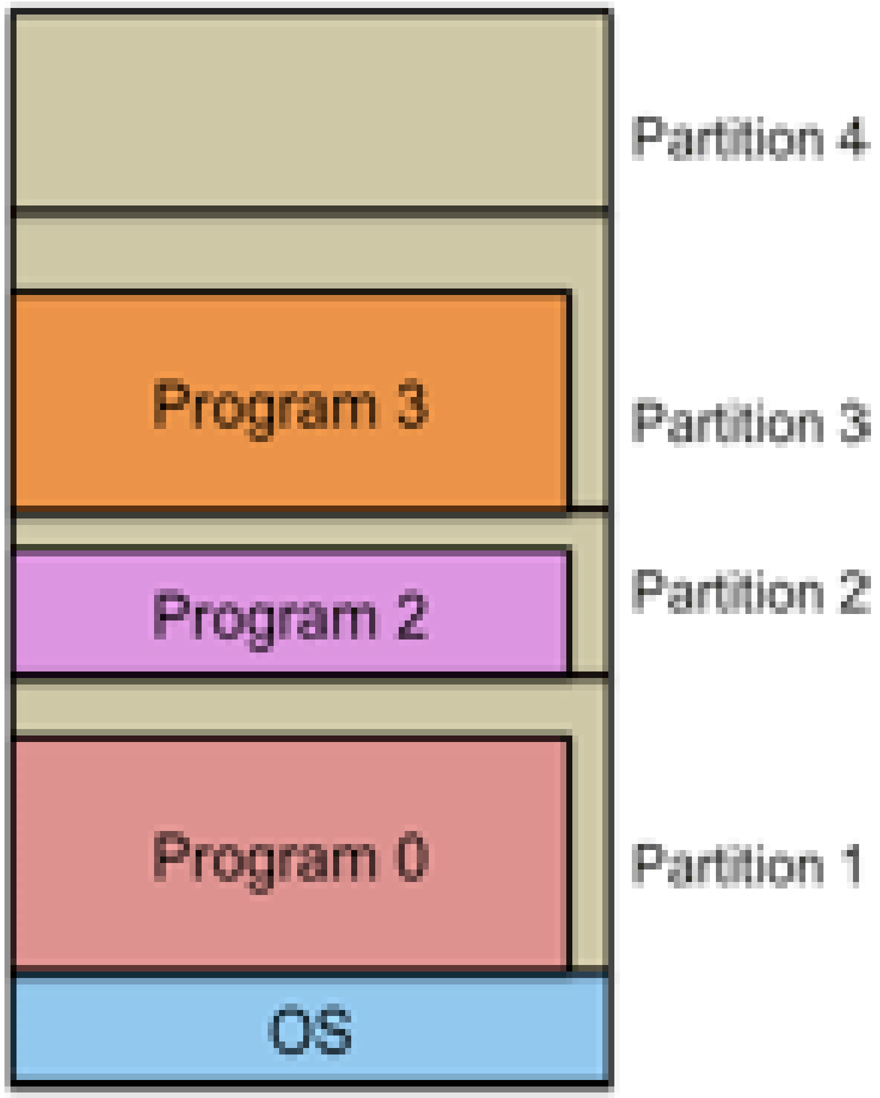


2. **Fixed Partition Memory**

- This scheme allows division of the memory into a number of separate fixed areas.
- Each of which can hold one process the size of each area differ, with each partition containing unused space.
- Thus the total unused space could be a lot. The occurrence of wasted space in this way is referred to as internal fragmentation – means wastage within the space allocated to a process.

- Typically the partitions would be set up with a range of portion sizes so that a mixture of large and small processes should be accommodated.
- Normally the processes will be independent so that a process cannot reference addresses outside its area.
- Since there is now ay to prevent a process generating an invalid address, memory protection has to be implemented by the operating system and I/O hardware.

Multiple fixed partitions



Disadvantages of fixed partition

- The fixed partition sizes can prevent a process being run due to the unavailability of a partition of sufficient size.
- Internal fragmentation wastes space which collectively could accommodate another process.

3. **Variable Partition Memory**

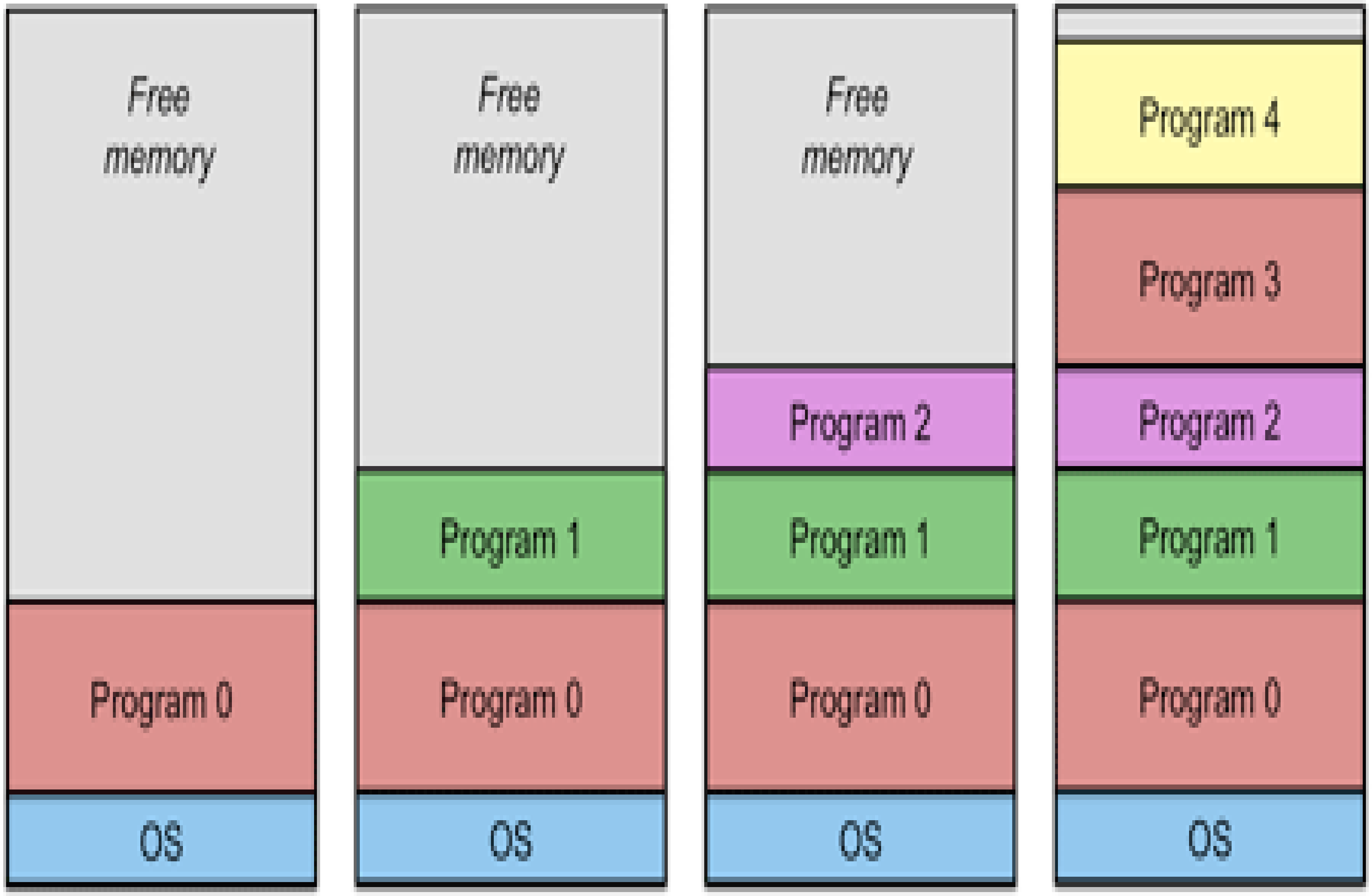
- Cure of fixed partition problems is to allow the partitions to be variable in size at load time i.e. allocate a process the exact amount of memory it requires.
- Processes are loaded into consecutive areas until the memory is filled or more likely the remaining space is too small to accommodate another process.

- When a process terminates, the space occupied is freed and becomes available for the loading of a new process.
- This reveals a flaw because as the processes are terminated and space is freed, the free space appears as a series of holes between the active memory areas.
- The OS must attempt to load an incoming process into a space large enough to accommodate it.

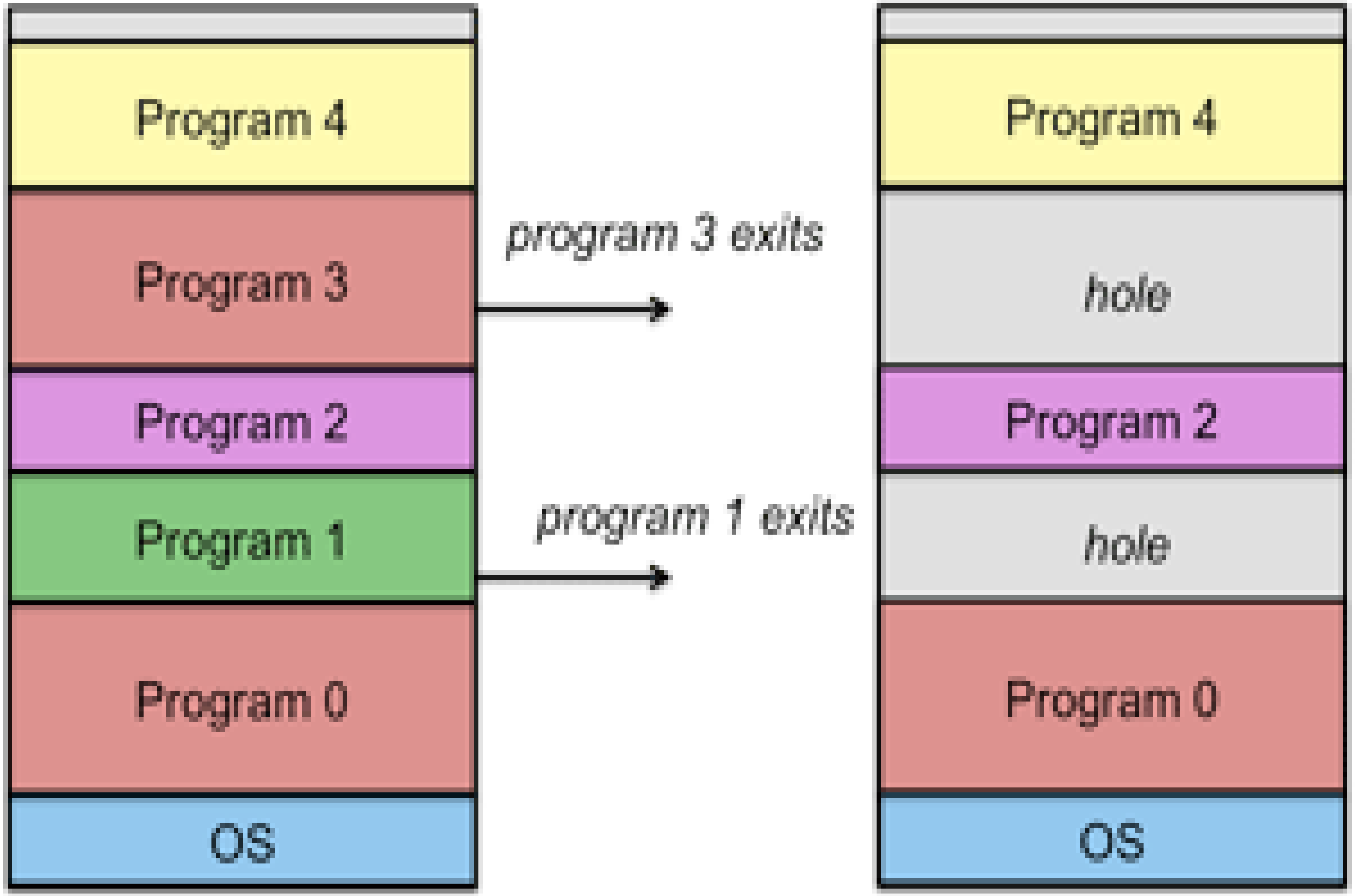
- It can often happen that a new process cannot be started because none of the hole is large enough even though the total free space (collected together) is more than the required size.
- Distribution of the free memory space in this fashion is called **External fragmentation.**

- The main difference between fixed partition and the variable partition is that the number, location and size of the partitions vary dynamically in the latter as processes come and go whereas they are fixed in the former.

Variable


Free memory
Program 0
OS
Free memory
Program 1
Program 0
OS
Free memory
Program 2
Program 1
Program 0
OS
Program 4
Program 3
Program 2
Program 1
Program 0
OS

Holes in variable



Difference between External and Internal Fragmentation

- **Internal Fragmentation** – It occurs in a fixed partition scheme within the space allocated to a process.

- **External Fragmentation** – It occurs in a variable partition scheme between the spaces allocated to a process.

- It will frequently happen that a process adjacent to one or more holes will terminate and free its space allocation.
- This results in two or three adjacent holes, which can then be viewed and utilized as a single hole.
- The effect is called coalescing of holes.
- It is a significant factor in maintaining fragmentation within usable limits.

Allocation algorithms/Storage placement policies

- If more than one region of memory is able to accommodate a segment (or process), the operating system needs to make a decision on how to pick the best segment.
- It is necessary to try and select “best” locations in which to place them i.e. select a series of holes which will provide the maximum overall throughput for the system bearing in mind that an inefficient allocation can delay the loading of a process.
- Several approaches are possible.

First Fit

- Scan segments of memory until you find a hole that fits.
- Break the hole into one segment for the process and another segment that will be a hole containing the leftover memory.
- This algorithm has the advantage of a small search.

Next fit

- This algorithm is just like First Fit except that it searches from where it left off the previous time (and then wraps around).

Best fit

- The Best Fit technique searches the list for the hole that best fits the process.
- It’s slower than First Fit since the entire list must be searched.
- It also produces tiny holes that are often useless.

Worst fit

- Since best fit doesn’t perform well, you would think its inverse would.
- The **Worst Fit** algorithm searches for the largest hole on the assumption that the hole remaining after allocation will be big enough to be useful.
- It also doesn’t yield good performance.

Quick fit

- The Quick Fit algorithm maintains separate lists for common sizes (e.g. 4K holes, 12K holes, …).
- It’s fast but has the drawback that finding neighbors for merging is difficult.

Variable Partition Allocation with Compaction

- Fragmentation problem encountered in the previous method can be tackled by physically moving resident processes about the memory in order to close up the holes and hence bring the free space into a single large block
- This process is referred to as **compaction**

- Compaction will make the total free space more usable by incoming processes.
- This is achieved at the expense of large-scale movement of current processes.
- All processes will need to be suspended while the re-shuffle takes place with attendant updating of process context information such as the load address.