Static and Dynamic Memory Management

Static and Dynamic Memory Management

- Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.
- Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.
- It checks how much memory is to be allocated to processes.
- It decides which process will get memory at what time.
- It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Static and Dynamic Memory Management

Process Address Space:

- The process address space is the set of logical addresses that a process references in its code.
- The operating system takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program.
- There are three types of addresses when memory is allocated:

✔ **Symbolic addresses:** The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.

✔ **Relative addresses:** At the time of compilation, a compiler converts symbolic addresses into relative addresses.

✔ **Physical addresses:** The loader generates these addresses at the time when a program is loaded into main memory.

Static and Dynamic Memory Management

- Virtual and physical addresses are the same in **compile-time** and **load-time** address-binding schemes.
- Virtual and physical addresses differ in **execution-time** address-binding scheme.
- The set of all logical addresses generated by a program is referred to as a **logical address space**.
- The set of all physical addresses corresponding to these logical addresses is referred to as a **physical address space**.
- The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device.

Static vs Dynamic Loading

- The choice between Static or Dynamic Loading is to be made at the time of computer program being developed.
- If you have to load your program statically, then at the time of compilation, the complete programs will be compiled and linked without leaving any external program or module dependency. The linker combines the object program with other necessary object modules into an absolute program, which also includes logical addresses.
- If you are writing a Dynamically loaded program, then your compiler will compile the program and for all the modules which you want to include dynamically, only references will be provided and rest of the work will be done at the time of execution.

Static vs Dynamic Loading

- At the time of loading, with static loading, the absolute program is loaded into memory in order for execution to start.
- If you are using dynamic loading, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program.

Static vs Dynamic Linking

- When static linking is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency.
- When dynamic linking is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking.

Static Memory Allocation	Dynamic Memory Allocation
In the static memory allocation, variables get allocated permanently, till the program executes or function call finishes.	In the Dynamic memory allocation, variables get allocated only if your program unit gets active.
Static Memory Allocation is done before program execution.	Dynamic Memory Allocation is done during program execution.
It uses stack for managing the static allocation of memory	It uses heap for managing the dynamic allocation of memory
In Static Memory Allocation, there is no memory re-usability	In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required
In this, once the memory is allocated, the memory size can not change.	In this allocation, when memory is allocated the memory size can be changed.

Static Memory Allocation	Dynamic Memory Allocation
In this memory allocation scheme, we cannot reuse the unused memory.	This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it.
In this memory allocation scheme, execution is faster than dynamic memory allocation.	In this memory allocation scheme, execution is slower than static memory allocation.
Memory is allocated at compile time.	Memory is allocated at run time.
In this allocated memory remains from start to end of the program.	In this allocated memory can be released at any time during the program.
Example: This static memory allocation is generally used for array.	**Example:** This dynamic memory allocation is generally used for linked list.