Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management –

Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.

In multiprogramming, the OS decides which process will get memory when and how much.

Allocates the memory when a process requests it to do so.

De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management –

Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.

Allocates the processor (CPU) to a process.

De-allocates processor when a process is no longer required.

his section contains the global and static variables.

Program

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. For example, here is a simple program written in C programming language –

```
#include <stdio.h>

int main() {
printf("Hello, World! \n");
return 0;
}
```

A computer program is a collection of instructions that performs a specific task when executed by a computer. When we compare a program with a process, we can conclude that a process is a dynamic instance of a computer program.

A part of a computer program that performs a well-defined task is known as an algorithm. A collection of computer programs, libraries and related data are referred to as a software.

Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

In general, a process can have one of the following five states at a time.

S.N. State & Description
1
Start

This is the initial state when a process is first started/created.

2
Ready

The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.

3
Running

Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

4
Waiting

Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

5
Terminated or Exit

Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

File
A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

Single vs Multithreaded Process
Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

Threads minimize the context switching time.
Use of threads provides concurrency within a process.
Efficient communication.
It is more economical to create and context switch threads.
Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.
Types of Thread
Threads are implemented in following two ways –

User Level Threads – User managed threads.

Kernel Level Threads – Operating System managed threads acting on kernel, an operating system core.

User Level Threads
In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

User level thread
Advantages
Thread switching does not require Kernel mode privileges.
User level thread can run on any operating system.
Scheduling can be application specific in the user level thread.
User level threads are fast to create and manage.
Disadvantages
In a typical operating system, most system calls are blocking.
Multithreaded application cannot take advantage of multiprocessing.
Kernel Level Threads
In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages
Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
Kernel routines themselves can be multithreaded.
Disadvantages
Kernel threads are generally slower to create and manage than the user threads.
Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.
Multithreading Models
Some operating system provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

Many to many relationship.
Many to one relationship.
One to one relationship.
Many to Many Model