

Summer-20 Final Term Assignment

Subject: Operating System Concepts

Note: Please attempt all Questions in sequence. All questions carry equal marks.

(50)

Name	Momin Hussain
ID	14672
Semester	4th Software Engineering

1. Differentiate between a process and thread with example.

Process:

Process means any program is in execution. Process control block controls the operation of any process. Process control block contains information about processes for example Process priority, process id, process state, CPU, register, etc. A process can create other processes which are known as **Child Processes**. Process takes more time to terminate and it is isolated means it does not share memory with any other process.

Thread:

Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process. A thread has 3 states: running, ready, and blocked.

Thread takes less time to terminate as compared to process and like process threads do not isolate.

PROCESS	THREAD
Processes are heavyweight operations	Threads are lighter weight operations
Each process has its own memory space	Threads use the memory of the process they belong to
Inter-process communication is slow as processes have different memory addresses	Inter-thread communication can be faster than inter-process communication because threads of the same process share memory with the process they belong to



2. List and discuss few types of thread.

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, application manages 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 begins with a single thread and begins running in that 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 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 multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within same process requires a mode switch to the Kernel.



3. What is a deadlock? In what situations it occurs in an OS.

Deadlock

In an operating system, a deadlock occurs when a process or thread enters a waiting state because a requested system resource is held by another waiting process, which in turn is waiting for another resource held by another waiting process. If a process is unable to change its state indefinitely because the resources requested by it are being used by another waiting process, then the system is said to be in a deadlock

In an operating system, a deadlock occurs when a process or thread enters a waiting state because a requested system resource is held by another waiting process, which in turn is waiting for another resource held by another waiting process.

Four Necessary and Sufficient Conditions for Deadlock

- mutual exclusion. The resources involved must be unshareable; otherwise, the processes would not be prevented from using the resource when necessary.
- hold and wait or partial allocation. ...
- no pre-emption. ...
- resource waiting or circular wait.

Deadlocks can be avoided by avoiding at least one of the four conditions, because all this four conditions are required simultaneously to cause deadlock.

1. Mutual Exclusion. ...
2. Hold and Wait. ...
3. No Preemption. ...
4. Circular Wait.

Starvation is the name given to the indefinite postponement of a process because it requires some resource before it can run, but the resource, though available for allocation, is never allocated to this process.



4. Discuss a solution to the critical-section problem must satisfy the three requirements

Solution to Critical Section Problem

A solution to the critical section problem must satisfy the following three conditions:

Mutual Exclusion

Out of a group of cooperating processes, only one process can be in its critical section at a given point of time.

Progress

If no process is in its critical section, and if one or more threads want to execute their critical section then any one of these threads must be allowed to get into its critical section.

Bounded Waiting

After a process makes a request for getting into its critical section, there is a limit for how many other processes can get into their critical section, before this process's request is granted. So after the limit is reached, system must grant the process permission to get into its critical section.



5. Differentiate between dynamic loading and dynamic linking with example.

Dynamic Loading

In dynamic loading, a routine of a program is not loaded until it is called by the program. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. Other routines methods or modules are loaded on request. Dynamic loading makes better memory space utilization and unused routines are never loaded.

Dynamic Linking

Linking is the process of collecting and combining various modules of code and data into a executable file that can be loaded into memory and executed. Operating system can link system level libraries to a program. When it combines the libraries at load time, the linking is called static linking and when this linking is done at the time of execution, it is called as dynamic linking. In static linking, libraries linked at compile time, so program code size becomes bigger whereas in dynamic linking libraries linked at execution time so program code size remains smaller.

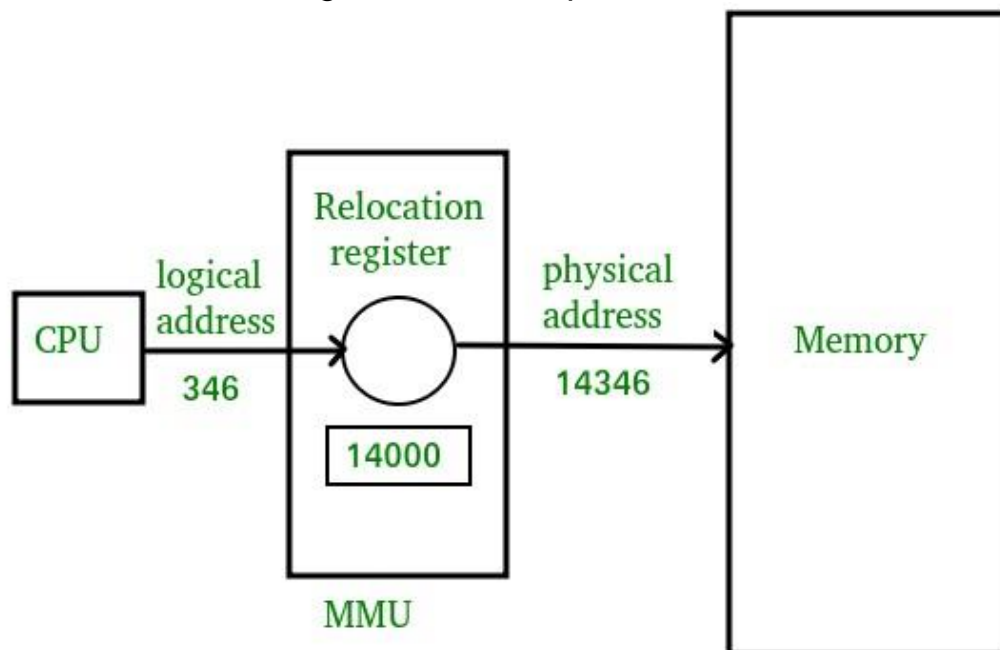


6. Write your understanding about logical Vs Physical address space?

Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective.

The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.



Differences Between Logical and Physical Address in Operating System

1. The basic difference between Logical and physical address is that Logical address is generated by CPU in perspective of a program whereas the physical address is a location that exists in the memory unit.
2. Logical Address Space is the set of all logical addresses generated by CPU for a program whereas the set of all physical address

mapped to corresponding logical addresses is called Physical Address Space.

3. The logical address does not exist physically in the memory whereas physical address is a location in the memory that can be accessed physically.
4. Identical logical addresses are generated by Compile-time and Load time address binding methods whereas they differs from each other in run-time address binding method. Please refer [this](#) for details.
5. The logical address is generated by the CPU while the program is running whereas the physical address is computed by the Memory Management Unit (MMU).



Thanks