OPERATING SYSTEM
IT-41033

CHAPTER – 4
Q&A

4.3 Which of the following components of program state are shared across threads in a multithreaded process?

a. Register values

b. Heap memory

c. Global variables

d. Stack memory

- Heap memory and global variables components of program state are shared across threads in a multithreaded process.

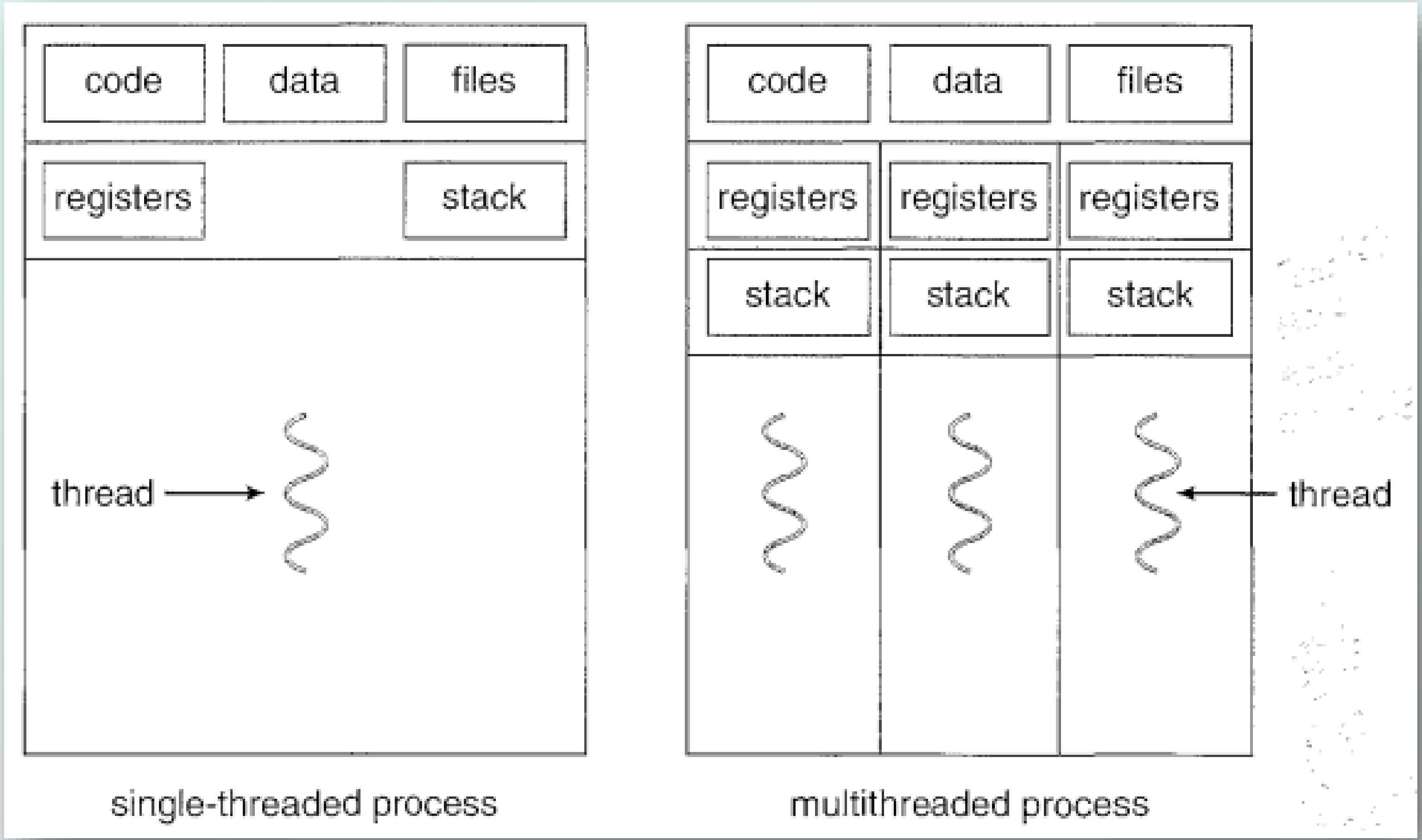
4.6 What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

- User-level threads are unknown by the kernel, whereas the kernel is aware of kernel threads.
- On systems using either many to one or many to many mapping, user threads are scheduled by the thread library and the kernel schedules kernel threads.
- Kernel threads does not be associated with a process whereas every user thread belongs to a process.
- Kernel threads are generally more expensive to maintain than user threads as they must be represented with a kernel data structure.

4.10 What resources are used when a thread is created? How do they differ from those used when a process is created?

- Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation.
- Creating a process requires allocating a process control block (PCB), a rather large data structure.
- The PCB includes a memory map, list of open files, and environment variables.
- Allocating and managing the memory map is typically the most time-consuming activity.
- Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority

Q Illustrate single thread and multithread processes.

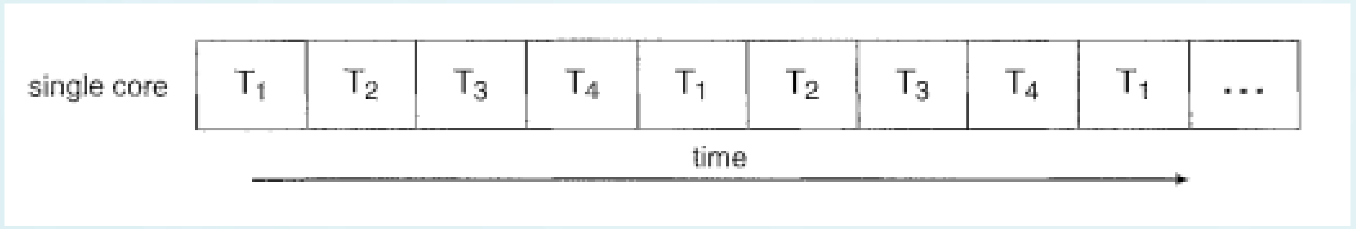


Q Explain Multicore Programming System.

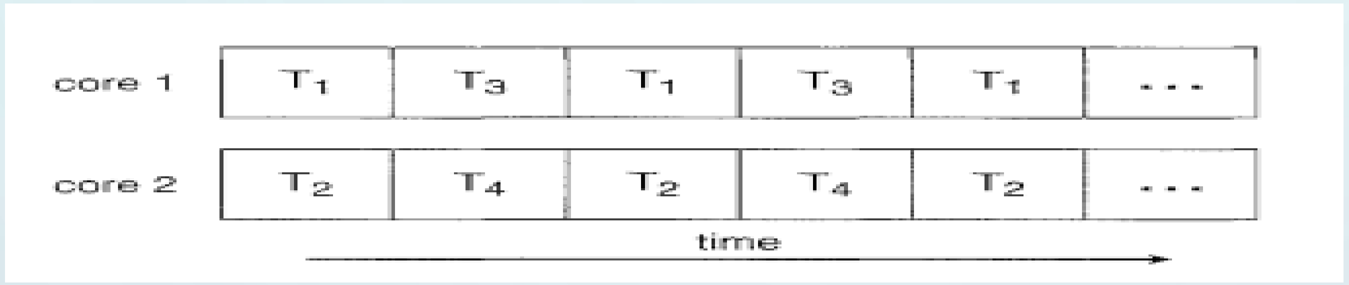
There are five areas present challenges in multicore programming system.

- **Dividing activities**: It is to find areas that can be divided into separate, concurrent tasks and thus can run in parallel on individual cores.
- **Balance**: While identifying tasks that can run in parallel, programmers must also ensure that the tasks perform equal work of equal value.
- **Data Splitting**: In separate tasking, the data accessed and manipulated by the tasks must be divided to run on separate cores.
- **Data Dependency**: The data accessed by the tasks must be examined for dependencies between two or more tasks. In instances where one task depends on data from another, programmers must ensure that the execution of the tasks is synchronized to accommodate the data dependency.
- **Testing and Debugging**: When a program is running in parallel on multiple cores, there are many different execution paths.

Q Explain Multicore Programming System (Cont…)



Concurrent execution on a single-core system



Parallel execution on a multicore system

Q In threading issues, explain about thread cancellation and thread pools.

Thread cancellation

Thread cancellation is the task of terminating a thread before it has completed. If multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be canceled. A thread that is to be canceled is often referred to as the target thread. Cancellation of a target thread may occur in two different scenarios:

- *Asynchronous cancellation*: One thread immediately terminates the target thread.
- *Deferred cancellation*: The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an orderly fashion.

Thread pools

Whereas creating a separate thread is certainly superior to creating a separate process, a multithreaded server nonetheless has potential problems. If we allow all concurrent requests to be serviced in a new thread, we have not placed abound on the number of threads concurrently active in the system.

Thread pools offer these benefits,

- Servicing a request with an existing thread is usually faster than waiting to create a thread.
- A thread pool limits the number of threads that exist at any one point. This is particularly important on systems that cannot support a large number of concurrent threads.