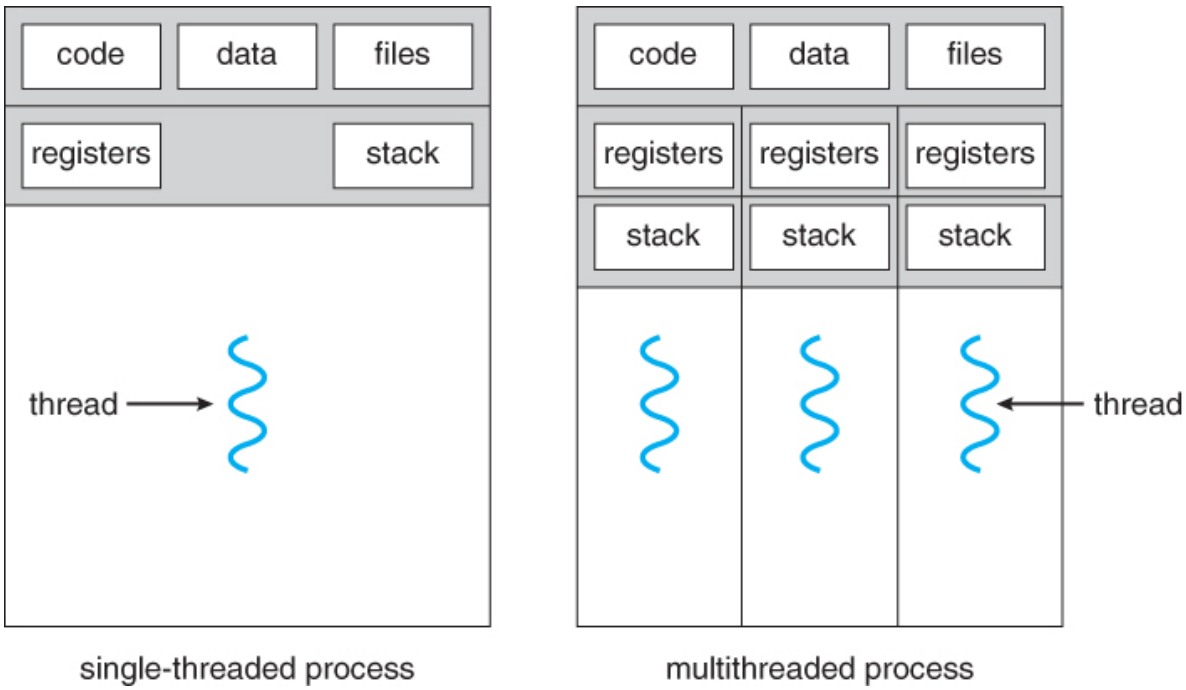
Part I: Theory Questions

1. **What are threads and multithreading?**
   -Thread is a basic unit of CPU usage consisting of a program counter, stack, a thread ID, and a set of registers.A set of registers essentially holds the current working variables and is essentially a stack. includes execution history

   Traditional processors have a single control thread.

   Multithreaded applications have multiple threads within a single process.Each Multithread has its own program counter, stack and register.However, it shares certain structures such as common code, data, and open files.

   You can see this better in the image below.



2. **Why is a thread faster than a process?**

   In a multiprocessor system, multiple threads can run on multiple CPUs simultaneously.Therefore, multi-threaded programs can run much faster than a single-processor system.Also, because threads require fewer resources and create less overhead, they can be faster than a program that uses multiple processes.

3. **What are the advantages of threads in operating systems?**

Threads have many features. Some of them are;

-Reduce context switching.

-Don't need for inter-process communication.

-Threads are dependent on process.

-Threads require less time for creation.

Also, threads are very useful in modern programming when a process has multiple tasks to perform independently of the others.

To give a good example of this;

- For example, in a word processor, a background thread checks spelling and grammar, a foreground thread processes user input (keystrokes), a third thread loads images from the hard drive, and a fourth makes periodic automatic backups of the file being edited.

4. **What is the difference between a process and a thread?**

Here are some of the main differences between a process and a thread;

- Regarding memory sharing, while memory cannot be shared between processors, it is shared between threads.

-Memory footprint takes up a lot of space on processors, while Threads take up less space.

- Processors are optimized for CPU-bound tasks while Threads are optimized for I/O-related tasks

- Processors are slower than threads.

-Child processes on processors can be interrupted, but Threads do not have this problem.

-A process is an instance of a program running on a computer.

-A thread is a unit of execution within a process.

-A process can have one or more threads.

5. **What are the types of threads? Explain the types.**

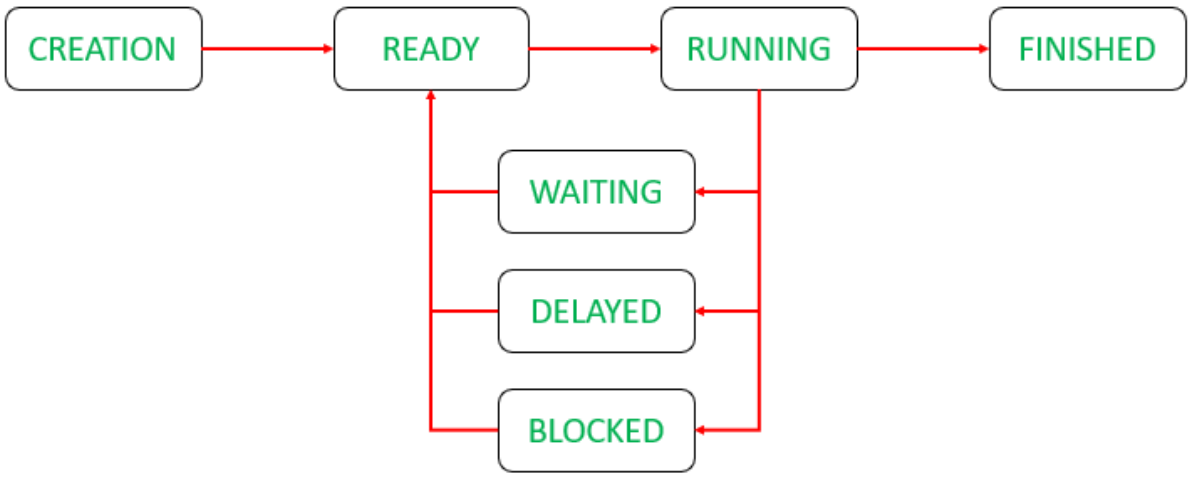
There are two types of threads to be managed in a modern system: User threads and kernel threads.

User threads are implemented in the user level library, they are not created using the system calls. Thread switching does not need to call OS and to cause interrupt to Kernel. Kernel doesn't know about the user level thread and manages them as if they were single-threaded processes.

Kernel threads knows and manages the threads. Instead of thread table in each process, the kernel itself has thread table (a master one) that keeps track of all the threads in the system. In addition kernel also maintains the traditional process table to keep track of the processes. OS kernel provides system call to create and manage threads.

6. What are the states of threads in OS?

There are five thread states in the Operating System.
These are ;
- Ready, Running, Waiting, Delayed, Blocked.



7. **What is the total number of child processes created? Give a detailed explanation for the answer.**

There are three calls to the fork() method. The first iteration, fork will create one child process of the main process. So, We have one child on this call. The second iteration, fork will create two children of the first calls. After this calls We have three child processes after this iteration. The last iteration, fork will create four child processes of the second call. The total number of created child processes is seven child processes. In conclusion, we have 2^3 situations of total calls.

8. **What is the total number of child processes created? Give a detailed explanation for the answer.**

2^n child process will be created by a fork() mechanism. Because the first iteration will create one child process. And the next step,up to 2^i child processes are created.