4.1 PROCESSES AND THREADS

1-

	Process	Thread
Overhead in creation and deletion.	Creating a process is more costly than creating a thread. It requires allocating memory, copy the program code and data and assign resources. Similarly, deleting a process requires deallocating the memory and releasing system resources.	Creating a thread is relatively easier and faster. It does not require allocating a memory space or copying data and program code since it shares the same space with the process. Also, deleting a thread is faster.
Way of communication and its speed.	Communication between processes requires invoking the kernel to manage the communication process which is relatively slow.	In a user-level thread, threads can communicate with each other through the shared memory without invoking the kernel which makes it fast.

4.2 TYPES OF THREADS

2-

	ULT	KLT
Mapping	Each ULT is mapped to one KLTs since the kernel is unaware of user-level threads.	Either one or many kernel-level threads can be mapped to one process.
Dealing with multiprocessor systems.	Since the kernel is unaware of the user-level threads, only one thread within a process can be executed. The kernel will treat all the threads within a process as one process in terms of scheduling and dispatching.	KLTs make use of multiprocessor systems. The kernel is aware of the KLTs and is responsible for scheduling and dispatching of the threads.
Overhead on the kernel	They don’t invoke the kernel for dispatching and scheduling. All theses are done at the program level. Hence, less overhead.	More overhead.
Portability	Programs that make use of ULT are portable.	Not portable.
Dispatching and scheduling	Dispatching and scheduling are performed at the program level by the threads library.	The kernel.

3- Because the kernel is unaware of the threads at the program level. It sees the entire process as one entity that is assigned to one processor. Which is not the case in KLT. In KLTs, the kernel takes care of scheduling and dispatching the different threads within a process.

4-

	ULT	KLT
Advantages	• Thread switching doesn't require invoking the OS which save overhead. • ULTs can run on any OS. • The application and the threads library are responsible of the scheduling and dispatching.	• KLTs make use of multiprocessing systems. • If one thread within a process got blocked, the kernel will dispatch to another thread within the same process.
Disadvantages	• A blocking system call will cause the entire process along with its threads to be blocked. • ULTs don't make use of multiprocessing systems.	• Switching between two threads requires a mode switch to the kernel.

4.3 MULTICORE AND MULTITHREADING

5- Thread switching doesn't require kernel-mode privileges. All of the thread management data structures are within the address space of one process. The process doesn't make a switch to the kernel mode to do thread management. This saves much overhead which makes thread switching faster.

6-

- ULT: if an algorithm uses ULT, it can be more efficient since ULT are lightweight processes and they are easy to create and destroy. In addition, performing multiple independent calculations concurrently can enhance the performance of the software.
- KLT: using KLTs can also be more efficient. The OS will be responsible for scheduling these threads across multiple processors. On the other hand, KLTs are heavyweight. Creating and managing kernel threads can be sometimes costly and degrade the performance.
- Uniprocessors and multiprocessors: an algorithm that makes use of multiple processors in multiprocessing systems can benefit significantly from these processors. The workload will be distributed across the different processes which saves much time.

4.4 WINDOWS PROCESS AND THREAD MANAGEMENT

7-

- Ready: a ready thread may be scheduled for execution.
- Standby: a standby thread has been selected to run next on a certain processor. The thread waits in this state until the processor is made available.
- Running: the thread is dispatched by the kernel and is running.
- Waiting: a thread enters the wait state when it’s blocked on an event or for synchronization purposes.
- Transition: a thread enters this state after waiting if it is ready to run.
- Terminated: the thread finished execution or its parent was terminated.

GENERAL QUESTIONS

8- a) 5 jobs. If one job is submitted, 16 drives will be left idle. If 2 jobs are submitted, 12 drives will be left idle and so on.

The maximum number of tape drives that may be left idle as a result of this policy: suppose that 9 jobs are submitted, for the first 5 jobs, all the drives will be in use. But, for the remaining 4 jobs, 4 drives will be left idle.

b) Since each job can start executing with 3 drives for a long time before requiring the fourth one, an alternative policy would be to allow each job to work with the three available drives until the fourth one is needed. This can allow 6 jobs to run at the same time but 2 drives will remain idle until they are needed as the fourth drive.

9-

a) If the number of kernel threads allocated to the program is less than the number of processors, this means that the remaining processors are left idle. In this case, the program doesn't make full use of the system resources resulting in a degraded performance.
b) When the number of the kernel level threads matches the number of processors, then the system resources are fully utilized which results in a higher performance.
c) If the number of the kernel level threads exceeds the number of processors, then some of the threads will have to wait until one of the processors is released.

10-

a) This program accomplishes multithreading using pthread library.
b) This program will produce an unexpected result each time it runs. This is because both the child thread and the main thread are modifying the same variable 'myglobal' at the same time with no synchronization mechanism. A better way to write this program would have been to write the for loop after calling pthread_join(). This would have ensured that the main thread doesn't modify the global variable until the child thread is over.