Operating System Concepts - Threads

Benefits of multithreading	Benefits from this practice include increased responsiveness to the user, resource sharing within the process, economy, and scalability issues such as more efficient use of multiple cores.
Heavyweight process	A traditional process that has a single thread of control.
Many-to-Many Model	Multiplexing many user-level threads to a smaller or equal number of kernel threads. Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor as they become available.
Many-to-One Model	Mapping many user threads to one kernel thread. Thread management is done by the thread library in user space, so it is efficient; but the entire process will block if a thread makes a blocking system call. Multiple threads are unable to run in parallel on multiprocessors.
One-to-One Model	Mapping each user thread to a kernel thread. It provides more concurrency than the many-to-one model. The only drawback to this model is that creating a user thread requires creating the corresponding kernel thread.
Pthreads	The POSIX standard defining an API for thread creation and synchronization.
Signal	Used in UNIX systems to notify a process that a particular event has occurred.
Target thread	A thread that is to be canceled.
Thread Library	Provides the programmer with an API for creating and managing threads.
Thread Pool	Creating a number of threads that wait for work. When a thread completes its service it returns to wait for more work. If there are no available threads the server waits until one becomes available.
Two-level model	A variation on the many-to-many model that multiplexes many user-level threads to a smaller or equal number of kernel threads but also allows a user-level thread to be bound to a kernel thread.