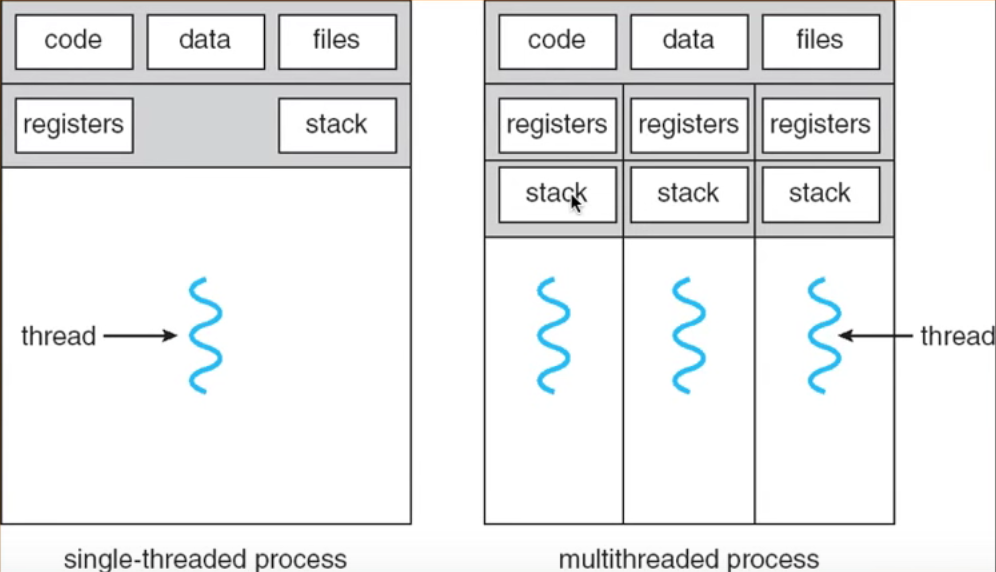
**Chapter 4 Threads**

Single and Multithreaded Processes



We can create multithread programs, basically split the work out among the program.

These threads share the theses stuff on the top

They can have performed their function depend on the other function

User Threads

* Thread management done by user-level threads library
* Three primary thread libraries:
* POSIX Pthreads
* Win32 threads
* Java threads

Kernel Threads

* Supported by the Kernel

Many to One Models

* Many user level thread mapped to single kernel thread

One to One Models

* Each user-level thread maps to kernel thread

Many to Many Models

* Allows many user level threads to be mapped to many kernel threads
* Allows the operating system to create a sufficient number of kernel threads

Semantics of fork () and exec ()

* Does fork () duplicate only the calling thread

Thread Cancellation

* Terminating a thread before it has finished
* Two general approaches:
* **Asynchronous cancellation** terminates the target thread immediately
* **Deferred cancellation** allows the target thread to periodically check if it should be cancelled

Signal Handling

* Signals are used in UNIX systems to notify a process that a particular event has occurred
* A signal handler is used to process signals
* Signal is generated by particular event
* Signal is delivered to a process
* Signal is handled
* Options:
* Deliver the signal to the thread to which the signal applies
* Deliver the signal to every thread in the process
* Deliver the signal to certain threads in the process
* Assign a specific thread to receive all signals for the process

Thread Pools

* Create a number of threads in pool where they await work
* Advantage:
* Usually slightly faster to service a quest with an existing thread that can create a new thread
* Allows the number of threads in the application(s) to be bound to the size of the pool

Thread Specific Data

* Allows each thread to have its own copy of data
* Useful when you do not have control over the thread creation process (i.e., when using a thread pool)

Scheduler Activations

* Both M:M and Two-level models require communication to maintain the appropriate number of kernel threads allocated to the application
* Scheduler activations provide **upcalls** – a communication mechanism from the kernel to the thread library
* This communication allows an application to maintain the correct number kernel threads

A race condition

* Because of the timing and which process starts first
* There is a chance that different executions may end up with different results