Technically, a thread is defined as an independent stream of instructions that can be scheduled to run as such by the operating system. But what does this mean?

To the software developer, the concept of a "procedure" that runs independently from its main program may best describe a thread.

To go one step further, imagine a main program (a.out) that contains a number of procedures. Then imagine all of these procedures being able to be scheduled to run simultaneously and/or independently by the operating system. That would describe a "multi-threaded" program.

POSIX threads, or Pthreads, is a standardized C language threads programming interface has been specified by the IEEE POSIX 1003.1c standard.
A process is created by the operating system and contains information about program resources and program execution state, including:

- Process ID, process group ID, user ID, and group ID
- Environment
- Working directory.
- Program instructions
- Registers
- Stack
- Heap
- File descriptors
- Signal actions
- Shared libraries
- Inter-process communication tools (such as message queues, pipes, semaphores, or shared memory).
What is a thread?

Threads use and exist within these process resources. They duplicate only the bare essential resources that enable them to exist as executable code, allowing them to be scheduled independently by the operating system.

This independent flow of control is accomplished because a thread maintains its own:

- Stack pointer
- Registers
- Scheduling properties (such as policy or priority)
- Set of pending and blocked signals
- Thread specific data.
What can I thread?

If routines can be interchanged, interleaved and/or overlapped in real time, they are candidates for threading.
Shared Memory

- All threads have access to the same global, shared memory
- Threads also have their own private data
- Programmers are responsible for synchronizing access (protecting) globally shared data.

- **Thread-safeness**: refers to an application's ability to execute multiple threads simultaneously without "clobbering" shared data or creating "race" conditions.
All Threads Share Memory
This is not thread-safe
What does Pthreads provide?

- **Thread management**: Routines that work directly on threads - creating, detaching, joining, etc. They also include functions to set/query thread attributes (joinable, scheduling etc.)
- ** Mutexes**: Routines that deal with synchronization, called a "mutex", which is an abbreviation for "mutual exclusion". Mutex functions provide for creating, destroying, locking and unlocking mutexes. These are supplemented by mutex attribute functions that set or modify attributes associated with mutexes.
- **Condition variables**: Routines that address communications between threads that share a mutex. Based upon programmer specified conditions. This group includes functions to create, destroy, wait and signal based upon specified variable values. Functions to set/query condition variable attributes are also included.
- **Synchronization**: Routines that manage read/write locks and barriers.
Let's look at an example

See pt_test.cpp
Let's look at a broken example

pt_bank.cpp contains race conditions

Can you spot them?
Exercise

Fix the bank example

Hint:

Use mutexes
Key Takeaways

• Every Process contains a thread
• A process can have multiple threads
• Multiple threads require independent work/procedures
• Communication between threads must be done carefully