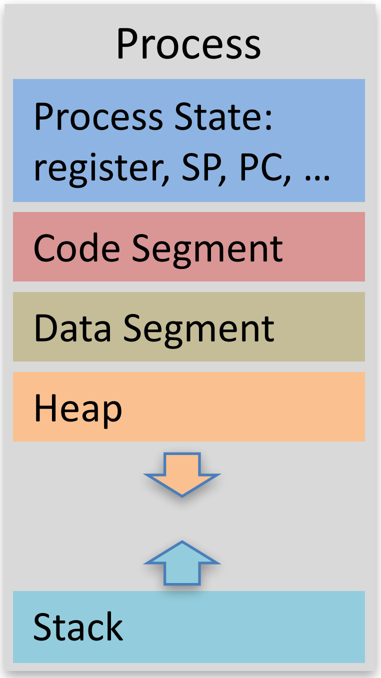
POSIX Threads: a first step toward parallel programming

George Bosilca
bosilca@icl.utk.edu

Process vs. Thread

- A process is a collection of virtual memory space, code, data, and system resources.
- A thread (lightweight process) is code that is to be serially executed within a process.
- A process can have several threads.



Threads executing the same block of code maintain separate stacks. Each thread in a process shares that process's global variables and resources.

Possible to create more efficient applications ?

Terminology

- Lightweight Process (LWP): or kernel thread
- X-to-Y model: the mapping between the LWP and the user threads (1:X – Unix & co., X:1 – User level threads, X:Y – Windows 7).
- Contention Scope: how threads compete for system resources
- Thread-safe a program that protects the shared data for its threads (mutual exclusion)
- Reentrant code: a program that can have more than one thread executing concurrently.
- Async-safe means that a function is reentrant while handling a signal (i.e. can be called from a signal handler).
- Concurrency vs. Parallelism - They are not the same! Parallelism implies simultaneous running of code (which is not possible, in the strict sense, on uniprocessor machines) while concurrency implies that many tasks can run in any order and possibly in parallel.

Thread vs. Process

- Threads share the address space of the process that created it; processes have their own address space.
- Threads have direct access to the data segment of its process; processes have their own copy of the data segment of the parent process.
- Threads can directly communicate with other threads of its process; processes must use interprocess communication to communicate with sibling processes.
- Threads have almost no overhead; processes have considerable overhead.
- New threads are easily created; new processes require duplication of the parent process.
- Threads can exercise considerable control over threads of the same process; processes can only exercise control over child processes.
- Changes to the main thread (cancellation, priority change, etc.) may affect the behavior of the other threads of the process; changes to the parent process does not affect child processes.

2.2.2 The Classical Thread Model in Modern Operating Systems 3e by Tanenbaum

Process vs. Thread

- Multithreaded applications must avoid two threading problems: deadlocks and races.
- A deadlock occurs when each thread is waiting for the other to do something.
- A race condition occurs when one thread finishes before another on which it depends, causing the former to use a bogus value because the latter has not yet supplied a valid one.

The key is synchronization

- Synchronization = gaining access to a shared resource.
- Synchronization REQUIRE cooperation.

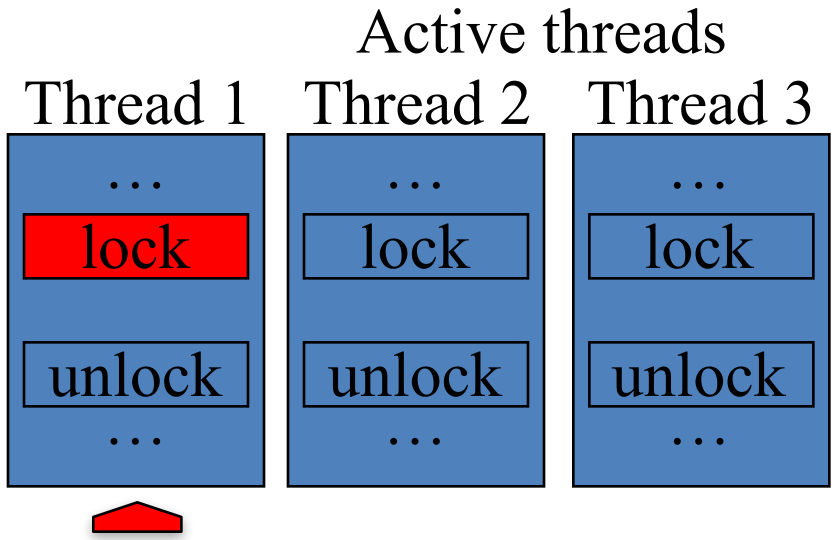
POSIX Thread

- What’s POSIX ?
  - Widely used UNIX specification
  - Most of the UNIX flavor operating systems

POSIX is the Portable Operating System Interface, the open operating interface standard accepted world-wide. It is produced by IEEE and recognized by ISO and ANSI.

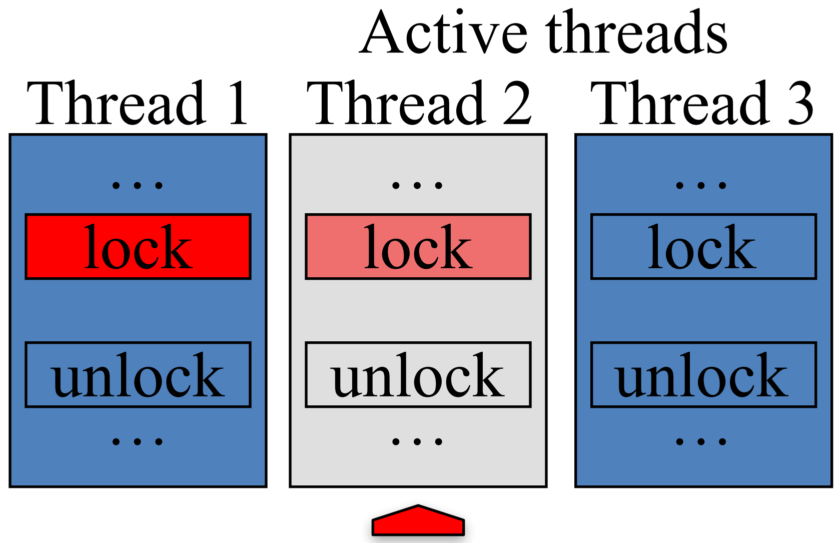
Mutual exclusion

- Simple lock primitive with 2 states: lock and unlock
- Only one thread can lock the mutex.
- Several politics: FIFO, random, recursive



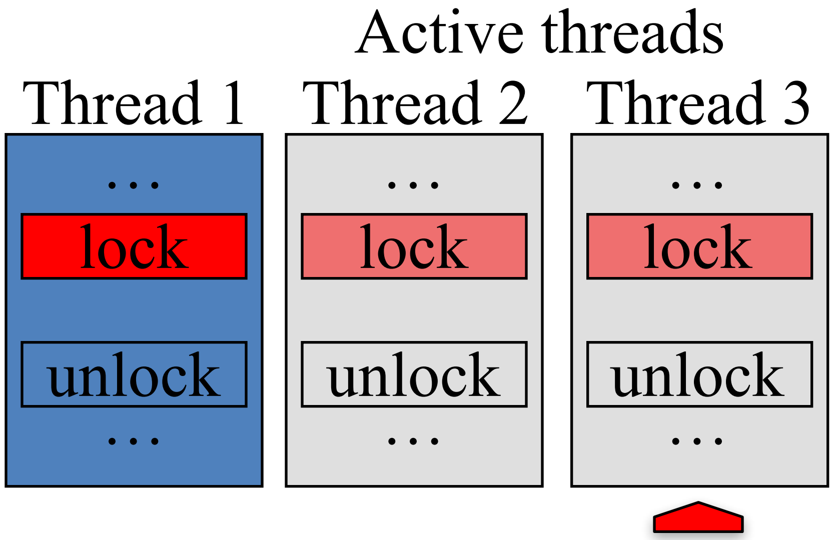
Mutual exclusion

- Simple lock primitive with 2 states: lock and unlock
- Only one thread can lock the mutex.
- Several politics: FIFO, random, recursive



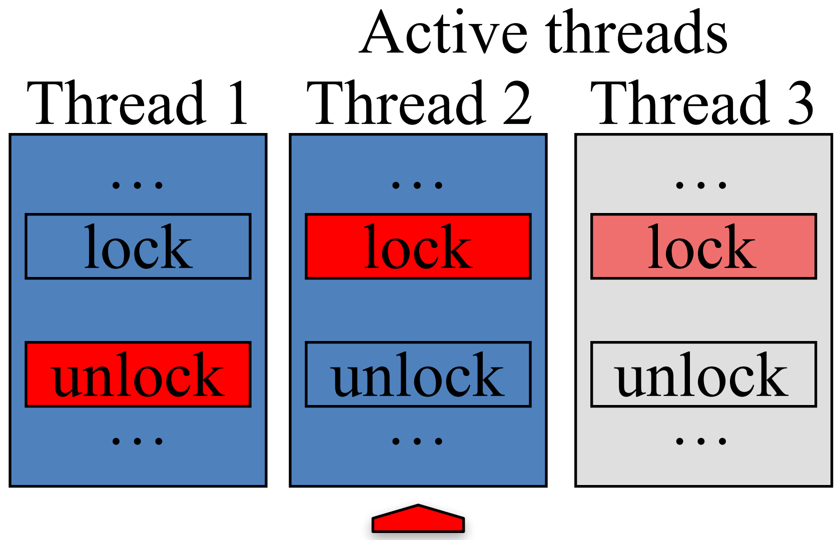
Mutual exclusion

- Simple lock primitive with 2 states: lock and unlock
- Only one thread can lock the mutex.
- Several politics: FIFO, random, recursive



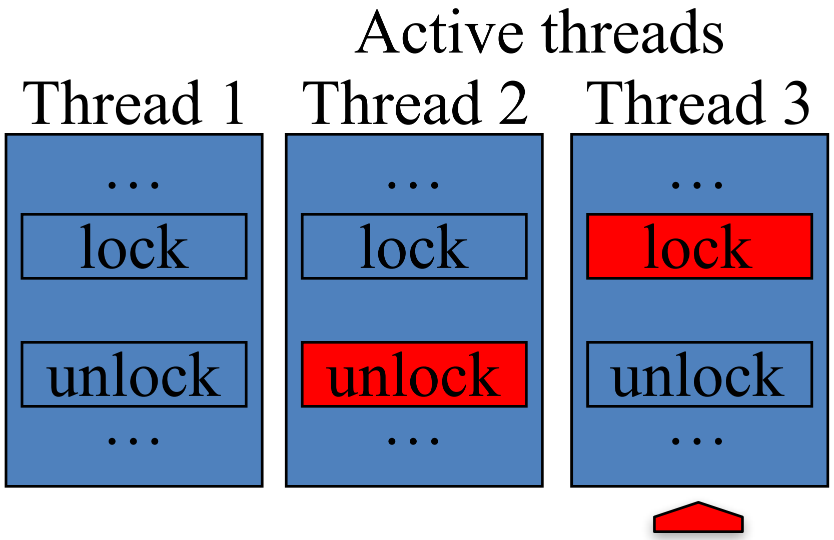
Mutual exclusion

- Simple lock primitive with 2 states: lock and unlock
- Only one thread can lock the mutex.
- Several politics: FIFO, random, recursive



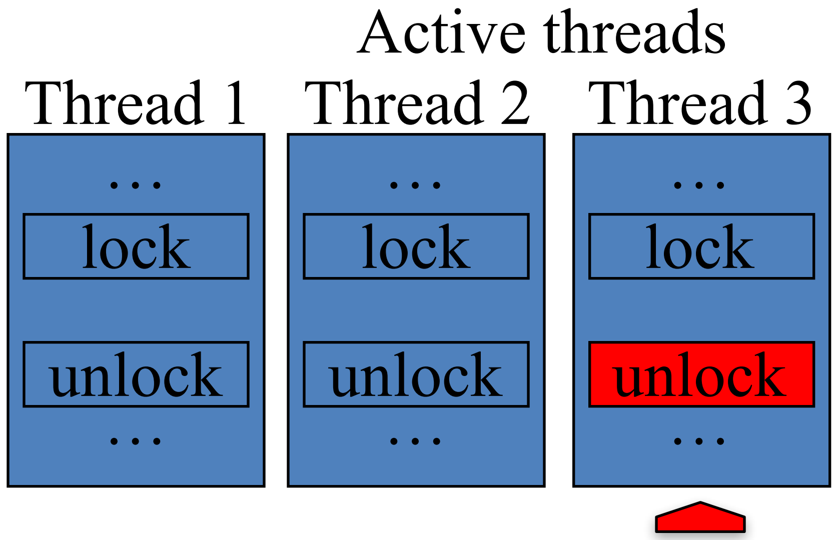
Mutual exclusion

- Simple lock primitive with 2 states: lock and unlock
- Only one thread can lock the mutex.
- Several politics: FIFO, random, recursive



Mutual exclusion

- Simple lock primitive with 2 states: lock and unlock
- Only one thread can lock the mutex.
- Several politics: FIFO, random, recursive



Mutual exclusion

- Spin vs. sleep ?
- What’ s the desired lock grain ?
  - Fine grain – spin mutex
  - Coarse grain – sleep mutex
- Spin mutex: use CPU cycles and increase the memory bandwidth, but when the mutex is unlock the thread continue his execution immediately.