

CSCI 3753

Operating Systems

Threads

Lecture Notes By

Shivakant Mishra

Computer Science, CU-Boulder

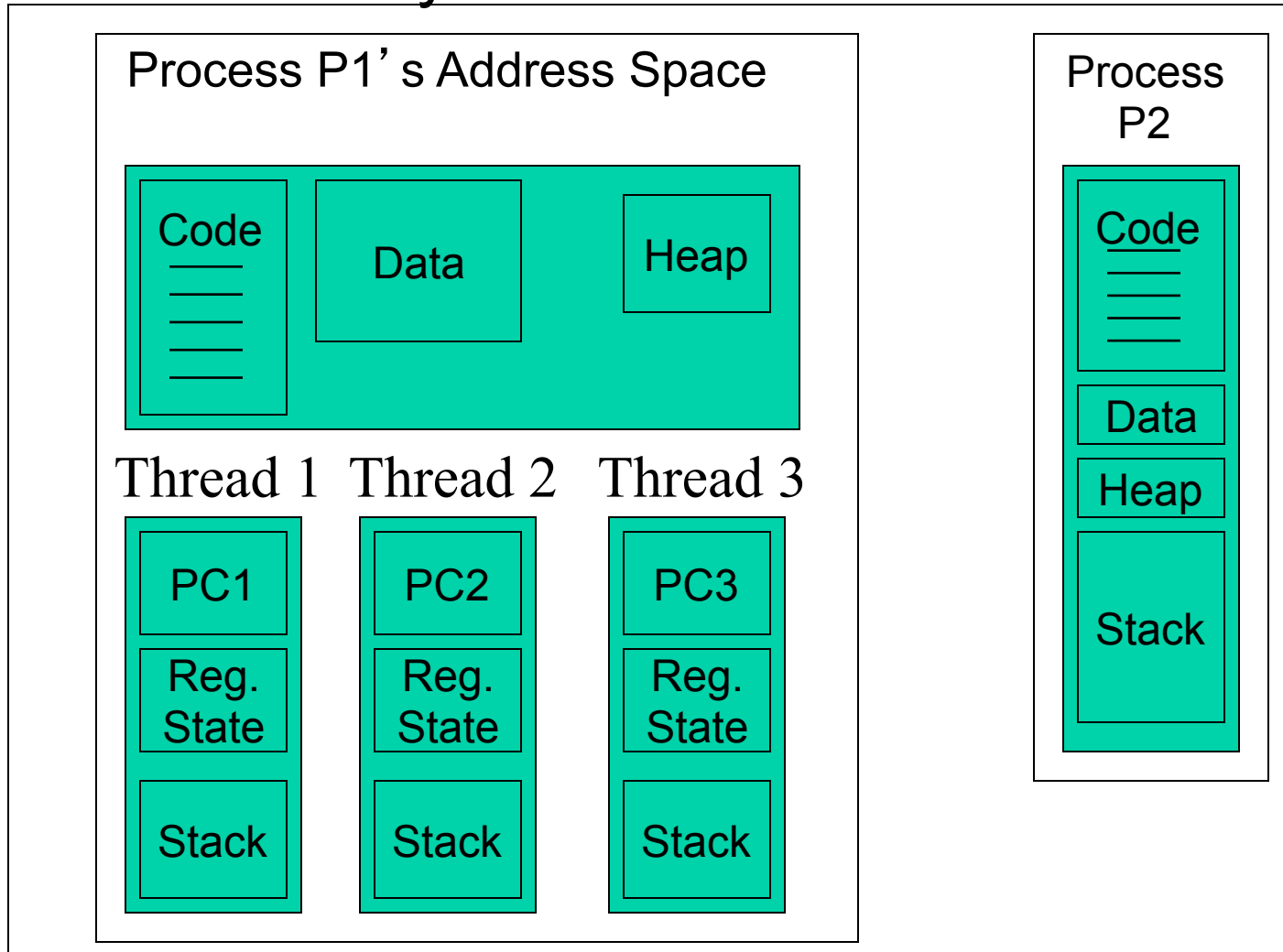
Last Update: 01/31/13

Threads

- A thread is a logical flow or unit of execution that runs within the context of a process
 - has its own program counter (PC), register state, and stack
 - shares the memory address space with other threads in the same process
 - share the same code and data and resources (e.g. open files)

Threads

Main Memory



- Process P1 is *multithreaded*
- Process P2 is *single threaded*
- The OS is *multiprogrammed*
- If there is preemptive timeslicing, the system is *multitasked*

Applications, Processes, and Threads

- Application = \sum_i Processes_i
 - e.g. a server could be split into multiple processes, each one dedicated to a specific task (UI, computation, communication, etc.)
- Process_j = \sum Threads_j
 - e.g. a Web server process could spawn a thread to handle each new http request for a Web page
- An application could thus consist of many processes and threads

Threads

- Why would you want multithreaded processes?
 - reduced context switch overhead
 - In Solaris, context switching between processes is 5x slower than switching between threads
 - Don't have to save/restore context, including base and limit registers and other MMU registers, also TLB cache doesn't have to be flushed
 - shared resources => less memory consumption => more threads can be supported, especially for a scalable system, e.g. Web server must handle thousands of connections
 - inter-thread communication is easier and faster than inter-process communication – threads share the same memory space, so just read/write from/to the same memory location!
 - thread also called a *lightweight* process

User-Space & Kernel Threads

- User-space threads are created without any kernel support
- User space threads are usually cooperatively multitasked, i.e. user threads within a process voluntarily give up the CPU to each other
- OS is unaware of user-space threads – only sees user-space processes
- User-space thread library
 - provides interface to create, delete threads in the same process
 - If one user space thread blocks, the entire process blocks in a many-to-one scenario (see text)
- User space thread also called a *fiber*

User-Space & Kernel Threads (2)

- *Kernel threads* are supported by the OS
- kernel sees threads and schedules at the granularity of threads
- Most modern OSs like Linux, Mac OS X, Win XP support kernel threads
- Mapping of user-level threads to kernel threads is usually one-to-one, e.g. Linux and Windows, but could be many-to-one, or many-to-many
- Win32 thread library is a kernel-level thread library

Pthreads

- *pthread*s is a POSIX threading API
- implementations of pthreads API differ underneath the API; could be user space threads; there is also pthreads support for kernel threads as well
- Details in recitation

User-Space & Kernel Threads (3)

- Java thread library is running in Java VM on top of host OS, so on Windows it's implemented on top of Win32 threading, while on Linux/Unix it's implemented on top of pthreads
- Possible scenarios:

