Chapter 4: Threads

Threads

- Objectives
  - To introduce the notion of a thread — a fundamental unit of CPU utilization, the basis of multithreaded computer systems
  - To briefly discuss the APIs for the Pthreads and Java thread libraries
  - To examine issues related to multi-threaded programming
- Topics
  - Multi-threading Models
  - Thread Libraries
  - Threading Issues
  - Operating System Examples
  - Linux Threads

Multi-core Programming

- All multi-core systems putting pressure on programmers, challenges include
  - Dividing activities
    - Define separate independent concurrent tasks
  - Balance
    - Identify tasks with equal work load
  - Data splitting
    - Divide data to be processed concurrently
  - Data dependency
    - Avoid serial dependencies of computations and synchronizations
  - Testing and debugging
    - Unknown parallel execution path and order

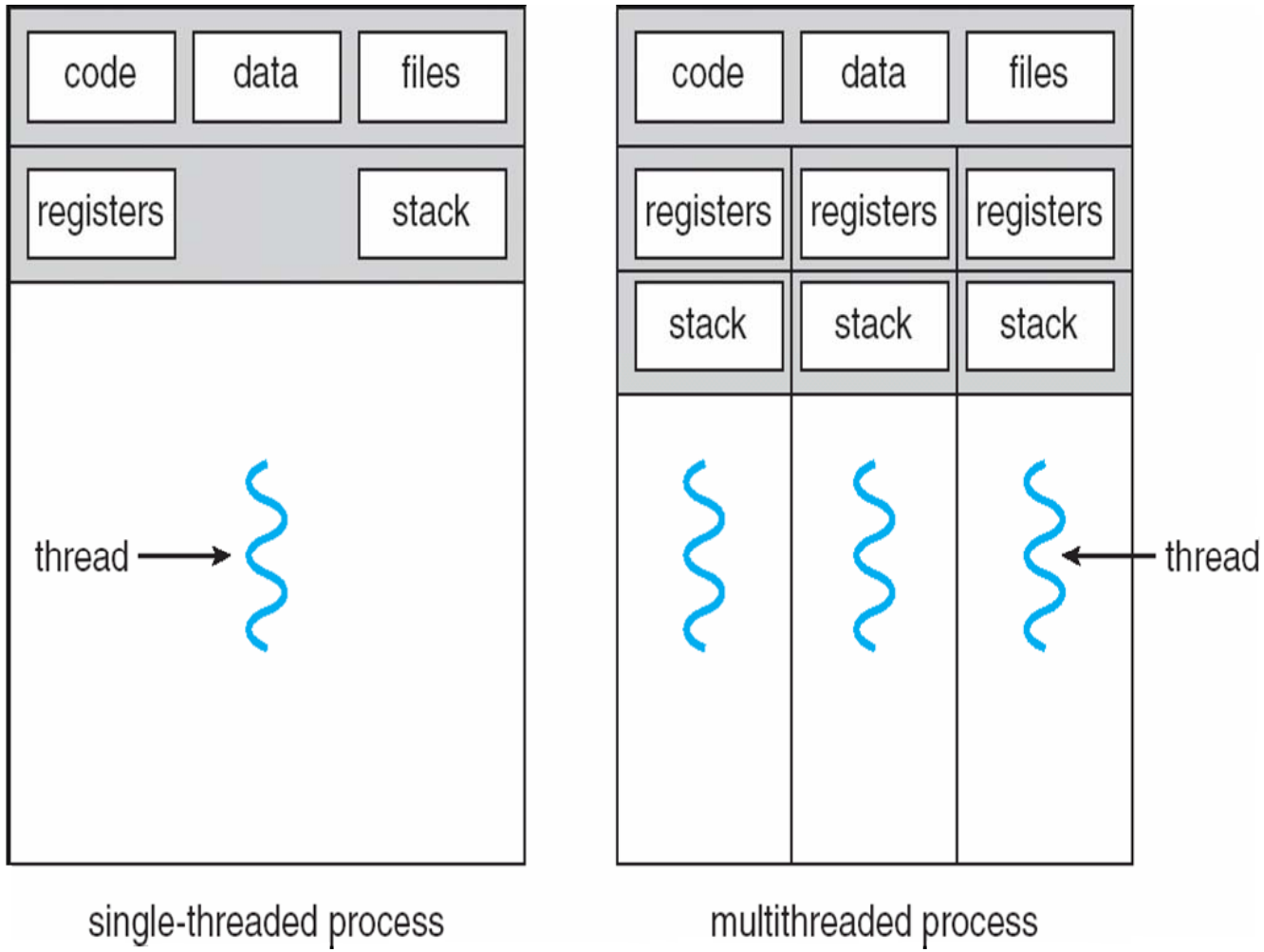
Threads

- Smallest unit of CPU utilization managed by the operating system scheduler
- Threads are lightweight processes
  - Sharing code, data, and files
  - But have separate registers and stacks
- Threads are contained inside a process
  - Multiple threads can exist within the same process and share resources, while different processes do not share their resources

Comparison: Threads and Processes

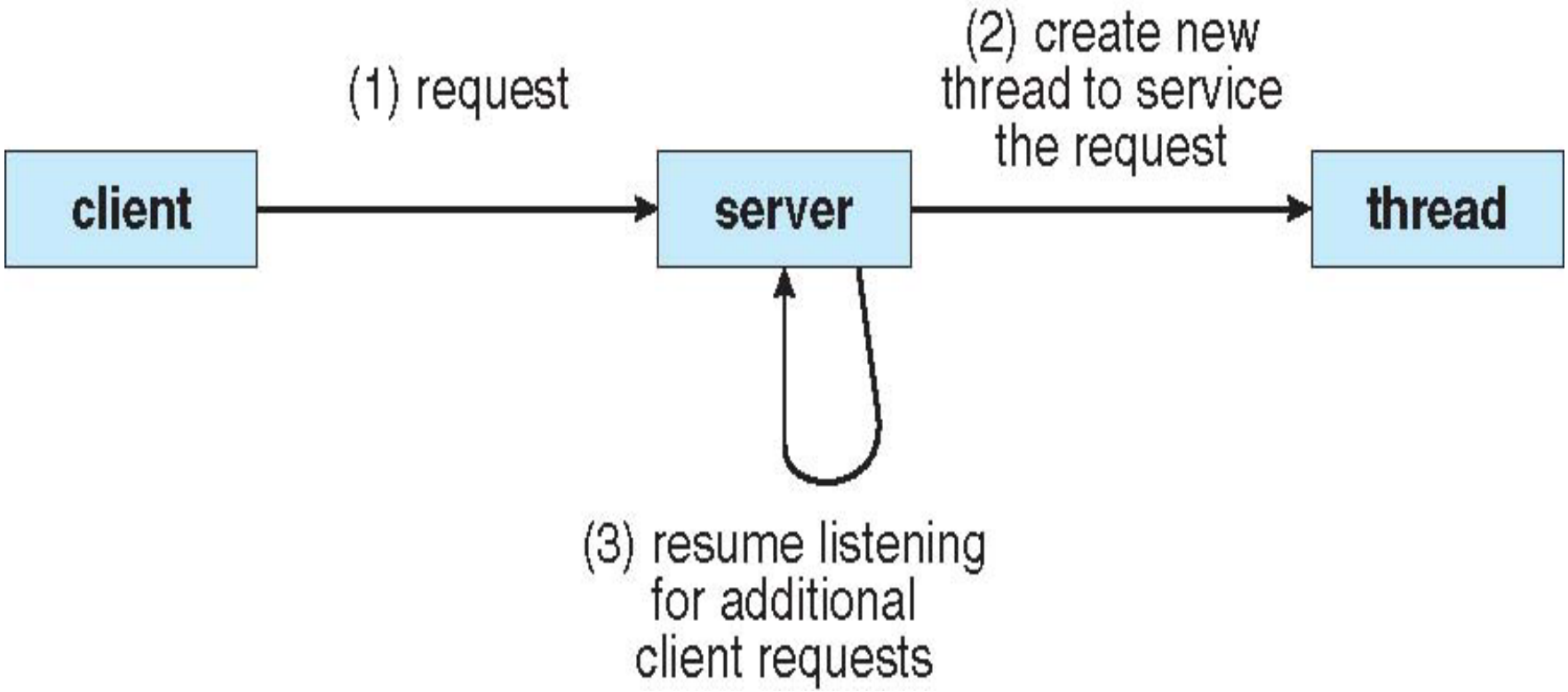
- Processes are typically independent, while threads exist as subsets of a process
- Processes carry considerably more state information than threads, whereas multiple threads within a process share process state as well as memory and other resources
- Processes have separate address spaces, whereas threads share their address space
- Processes interact only through system-provided inter-process communication mechanisms
- Context switching between threads in the same process is typically faster than context switching between processes

Single and Multi-threaded Processes



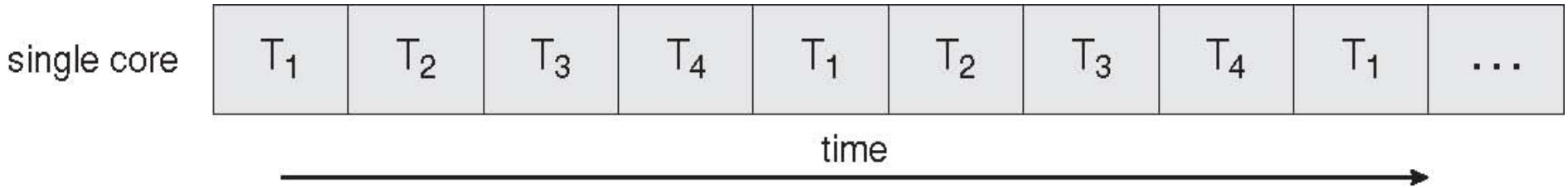
Benefits: Responsiveness – Resource Sharing – Economy – Scalability

Multi-threaded Server Architecture

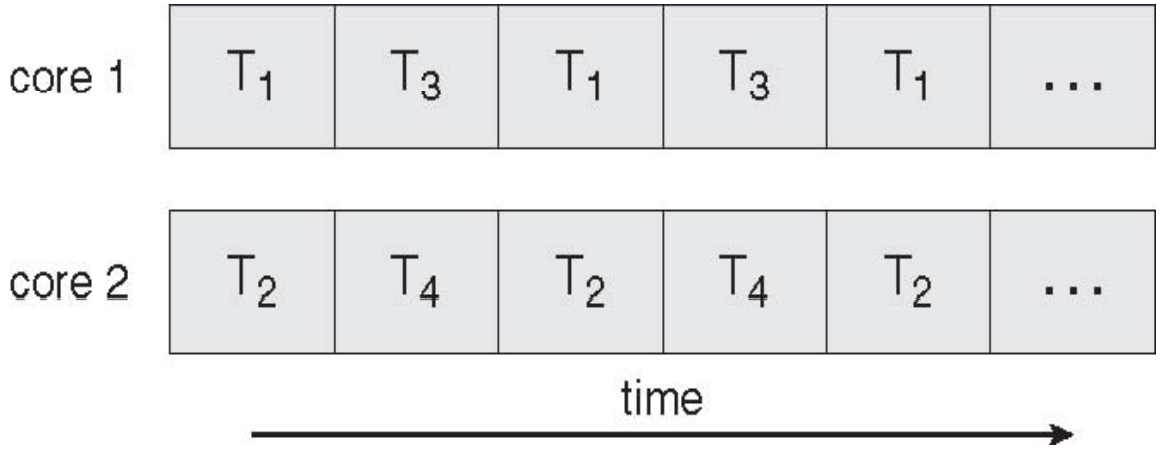


Threaded Single/Multi-core Systems

- Concurrent execution on single core



- Parallel execution on multi-core (here: 2 cores)



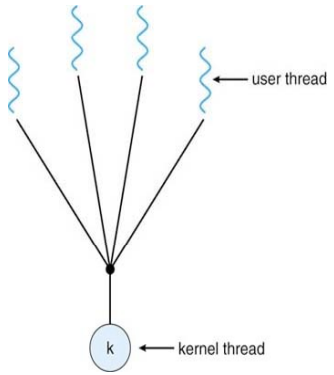
Challenges: Dividing Activities – Balance – Data Splitting – Data Dependency – Testing and Debugging

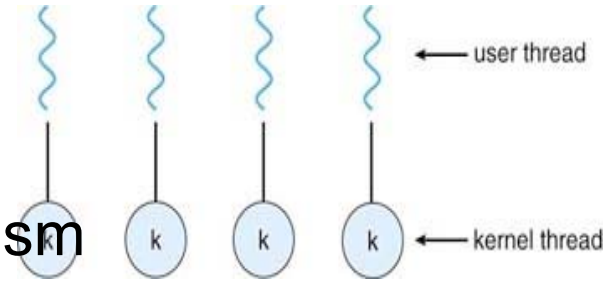
Kernel Threads

- At least one kernel thread exists within each process
  - Created and scheduled by the OS kernel
- One kernel thread assigned to each (logical) CPU core
  - Unit of independent process, thread scheduling
- Support for kernel threads
  - Linux
  - Mac OS X
  - Solaris
  - Tru64 UNIX
  - Since Windows XP/2000

User Threads

- User and application program thread management done above kernel by user-level threads library
- Ultimately, a relationship must exist between user threads and kernel threads
  - Many-to-one prohibits multithreading within user process
    - No concurrent execution as only one user thread can run on a CPU core
    - All user threads get blocked if underlying kernel thread is blocked
  - One-to-one and many-to-many imple-mentations enable multi-processor parallelism





Threading (1)

- Semantics of fork() and exec() system calls
  - Does fork() duplicate only the calling thread or all threads?
- Thread cancellation of target thread
  - Asynchronous cancellation terminates the target thread immediately
  - Deferred cancellation allows the target thread to periodically check, if it should be cancelled
- Thread pools
  - Usually slightly faster to service a request with an existing thread than creating a new thread
  - Allows the number of threads in the application(s) to be bound to the size of the pool

Threading (2)

- 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)

- Signal handling
  - … next slide

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
- Handling 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 Libraries

- Thread libraries provide programmers with an API for creating and managing threads
- Two primary ways of implementing
  - Library entirely in user space
  - Kernel-level library supported by the OS
- Three very common user-level thread libraries
  - POSIX Pthreads
  - Win32 threads
  - Java threads
- Operating thread systems:
  - Linux

Pthreads

- May be provided either as user-level or kernel-level
- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization
  - http://standards.ieee.org/findstds/interps/1003-1c-95_int/ (1996)
  - API specifies behavior of thread library
  - Implementation is up to the developer of the library

- Common in UNIX-like operating systems
  - Solaris, Linux, Mac OS X
  - Win32 threads quite similar to Pthreads

IEEE: Institute of Electrical and Electronics Engineers

Java Threads

- Java threads are managed by the JVM
  - Threads determine the fundamental program execution model in Java, combined with a rich set of features for creation of threads and their management
- Typically implemented using the threads model provided by underlying OS
- Java threads may be created by two alternatives
  - Create a new class derived from the Thread class and override `run()` method
  - Implementing the Runnable interface

Linux Threads

- Linux refers to them as *tasks* rather than threads
- Thread creation is done through `clone()` system call
  - Processes are created by `fork()` system call (Chapter 3)

- `clone()` allows a child task to share the address space of the parent task (process)

flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.