Threads and Concurrency

Pertemuan 7 dan 8

Kompetensi Khusus

- Mahasiswa mampu menentukan proses dan thread untuk diterapkan ke dalam manajemen thread yang akan dijalankan (C3, A2)(C1)

Materi

1. Overview
2. Multicore Programming
3. Multithreading Models
4. Thread Libraries
5. Implicit Threading
6. Threading Issues

1. Overview

Single and Multithreaded Processes

A traditional (or heavyweight) process has a single thread of control. If a process has multiple threads of control, it can perform more than one task at a time.

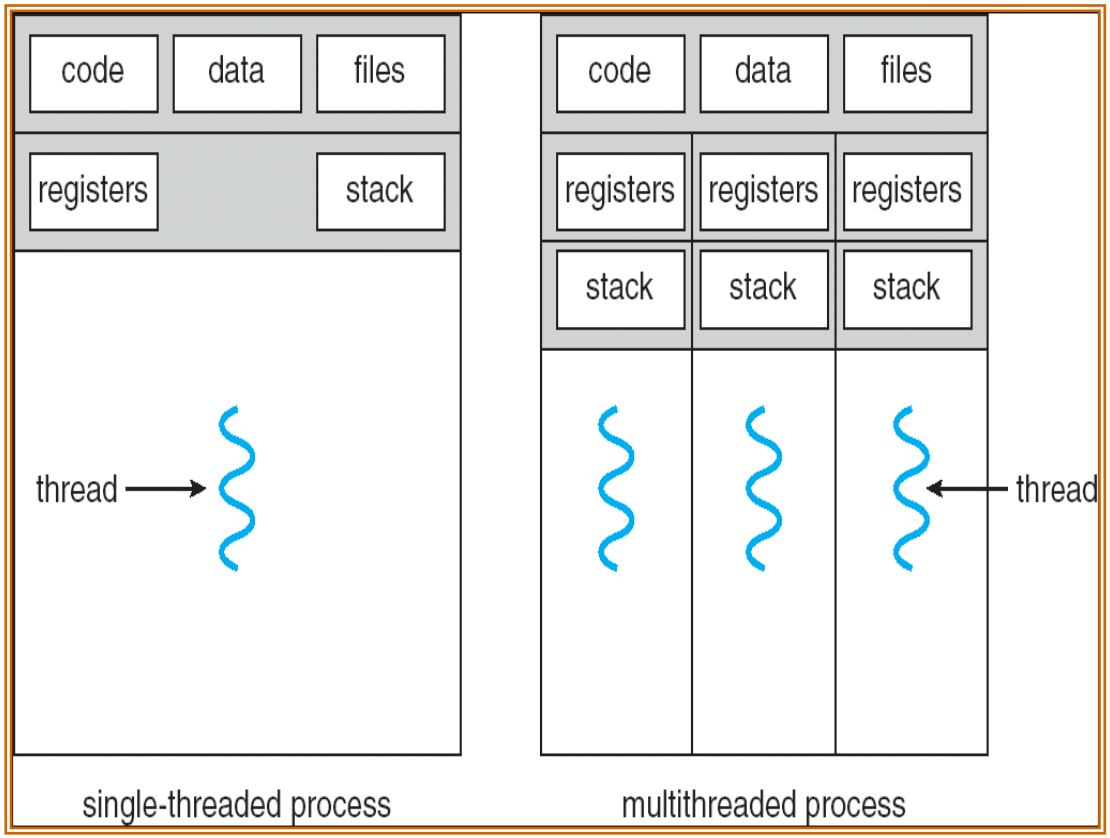


Fig. Single-threaded and multithreaded processes.

Benefits

- Responsiveness
- Resource Sharing
- Economy
- Utilization of MP Architectures

2. Multicore Programming

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

- Examples
  - Windows XP/2000
  - Solaris
  - Linux
  - Tru64 UNIX
  - Mac OS X

ubm
Universitas Bunda Mulia

3. Multithreading Models

3.1 Multithreading Models

- Many-to-One
- One-to-One
- Many-to-Many

3.1.1 Many-to-One

- Many user-level threads mapped to single kernel thread
- Examples:
  - Solaris Green Threads
  - GNU Portable Threads

3.1.1 Many-to-One

The many-to-one model maps many user-level threads to one kernel thread. Thread management is done by the thread library in user space, so it is efficient; but the entire process will block if a thread makes a blocking system call.

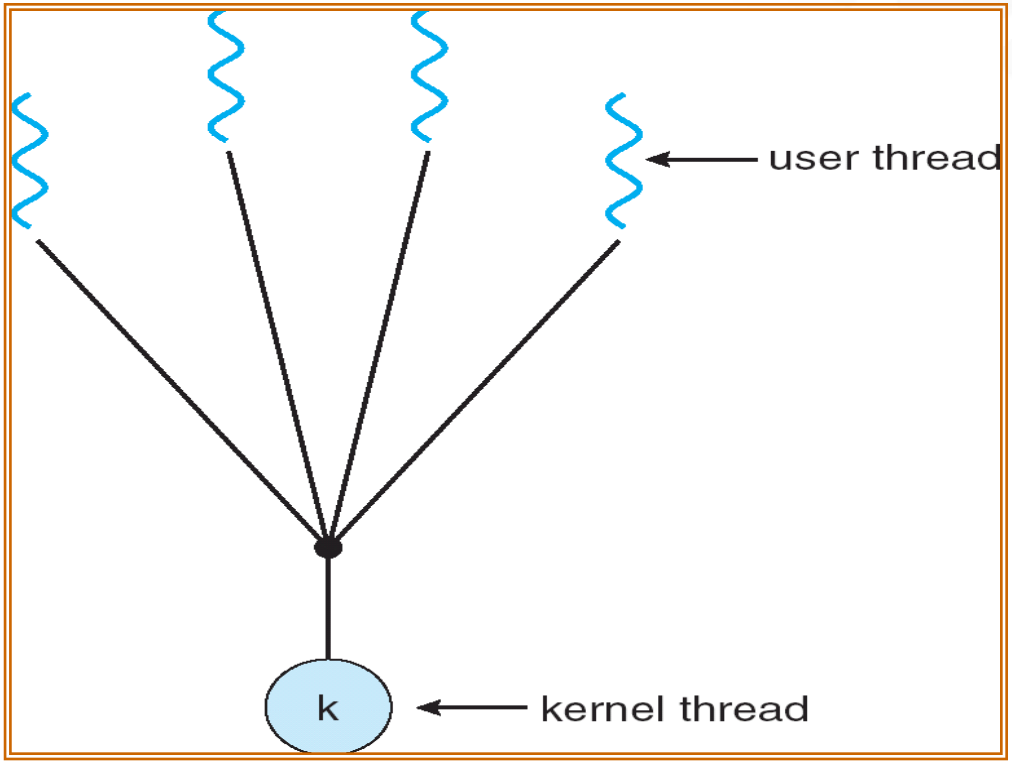


Fig. Many-to-one model

3.1.2 One-to-One

- Each user-level thread maps to kernel thread
- Examples
  - Windows NT/XP/2000
  - Linux
  - Solaris 9 and later

3.1.3 Many-to-Many Model

The many-to-many model multiplexes many user-level threads to a smaller or equal number of kernel threads.

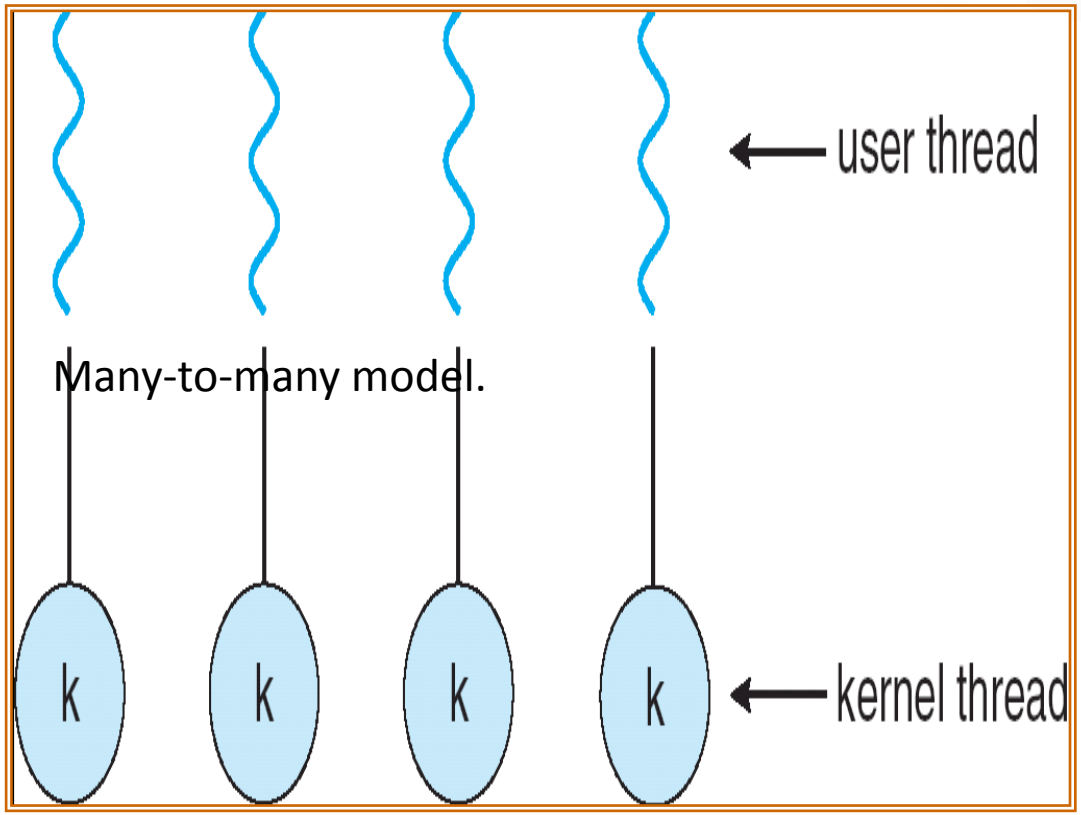


Fig. Many-to-manymodel

3.4 Two-level Model

- Similar to M:M, except that it allows a user thread to be bound to kernel thread
- Examples
  - IRIX
  - HP-UX
  - Tru64 UNIX
  - Solaris 8 and earlier

3.4 Two-level Model

One popular variation on the many-to-many model still multiplexes many user-level threads to a smaller or equal number of kernel threads but also allows a user-level thread to be bound to a kernel thread. This variation, sometimes referred to as the two-level model

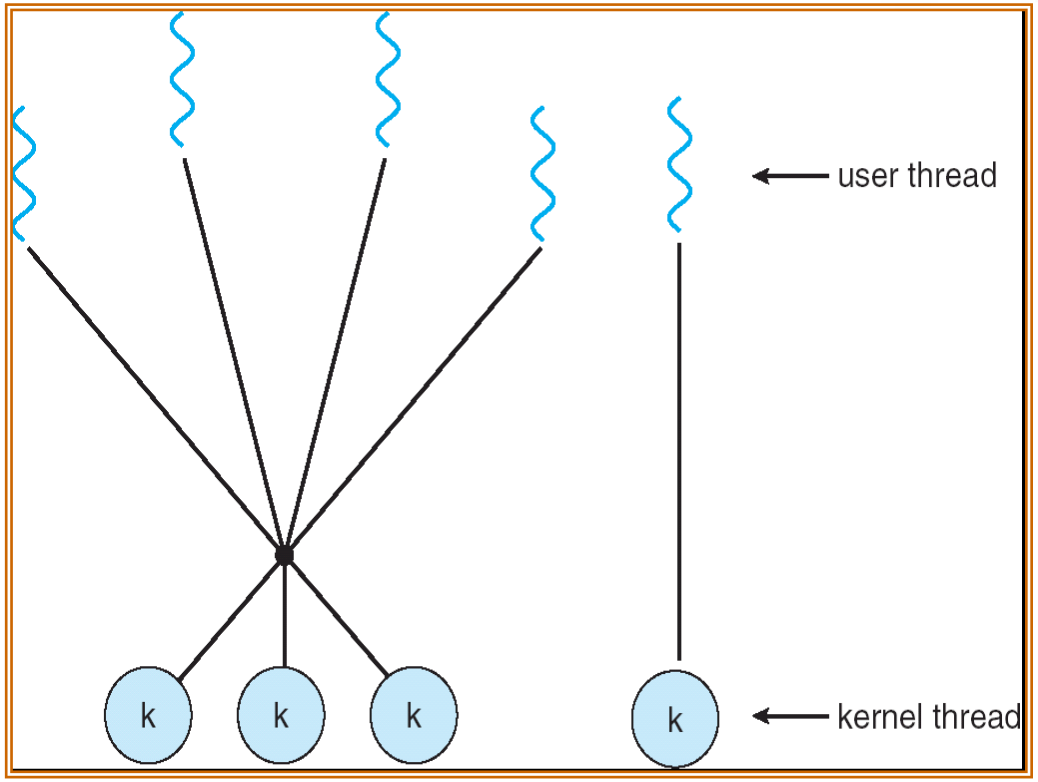


Fig. Two-level model

4. Thread Libraries

4.1 Threading Issues

- Semantics of fork() and exec() system calls
- Thread cancellation
- Signal handling
- Thread pools
- Thread specific data
- Scheduler activations

4.2 Semantics of fork() and exec()

- Does fork() duplicate only the calling thread or all threads?

4.3 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

ubm
Universitas Bunda Mulia

5. Implicit Threading

5.1 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 threa to receive all signals for the process

5.2 Thread Pools

- Create a number of threads in a pool where they await work
- Advantages:
  - Usually slightly faster to service a request with an existing thread than create a new thread
  - Allows the number of threads in the application(s) to be bound to the size of the pool

5.3 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)

5.4 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

ubm
Universitas Bunda Mulia

6. Threading Issues

6.1 Pthreads

- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization
- API specifies behavior of the thread library, implementation is up to development of the library
- Common in UNIX operating systems (Solaris, Linux, Mac OS X)

6.2 Windows XP Threads

- Implements the one-to-one mapping
- Each thread contains
  - A thread id
  - Register set
  - Separate user and kernel stacks
  - Private data storage area
- The register set, stacks, and private storage area are known as the context of the threads
- The primary data structures of a thread include:
  - ETHREAD (executive thread block)
  - KTHREAD (kernel thread block)
  - TEB (thread environment block)

6.3 Linux Threads

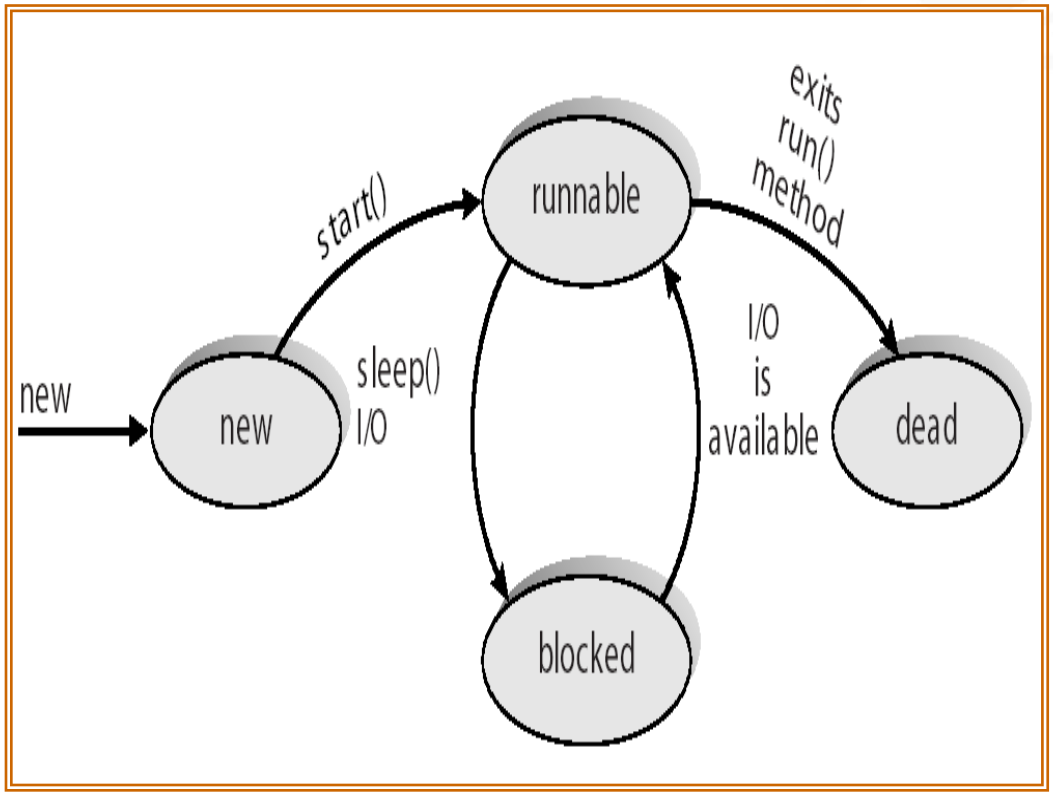
- Linux refers to them as tasks rather than threads
- Thread creation is done through **clone**() system call
- **clone**() allows a child task to share the address space of the parent task (process)

6.4 Java Threads

- Java threads are managed by the JVM
- Java threads may be created by:
  - Extending Thread class
  - Implementing the Runnable interface

6.4.1 Java Thread States

There are two techniques for creating threads in a Java program. One approach is to create a new class that is derived from the Thread class and to override its run 0 method. An alternative- and more commonly used_x0002_technique is to define a class that implements the Runnable interface.



Summary

- Benfit Threads are responsiveness, resource sharing, economy, utilization of MP architectures
- Multithreading Models consist of Many-to-One, One-to-One, Many-to-Many
- Advantages of Thread Pools are usually slightly faster to service a request with an existing thread than create a new thread and allows the number of threads in the application(s) to be bound to the size of the pool

ubm
Universitas Bunda Mulia

Thank You