Threads

- Overview
- Multithreading Models
- Threading Issues
- Pthreads
- Solaris 2 Threads
- Windows 2000 Threads
- Linux Threads
- Java Threads
Threads

- A thread (or lightweight process) is a basic unit of CPU utilization; it consists of:
  - program counter
  - register set
  - stack space
- A thread shares with its peer threads its:
  - code section
  - data section
  - operating-system resources
  - collectively know as a task.
- A traditional or heavyweight process is equal to a task with one thread

Threads (Cont.)

- In a multiple threaded task, while one server thread is blocked and waiting, a second thread in the same task can run.
  - Cooperation of multiple threads in same job confers higher throughput and improved performance.
  - Applications that require sharing a common buffer (i.e., producer-consumer) benefit from thread utilization.
- Threads provide a mechanism that allows sequential processes to make blocking system calls while also achieving parallelism.
- Kernel-supported threads (Mach and OS/2).
- User-level threads; supported above the kernel, via a set of library calls at the user level (Project Andrew from CMU).
- Hybrid approach implements both user-level and kernel-supported threads (Solaris 2).
5.5 Operating System Concepts

Single and Multithreaded Processes

Benefits

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

- Thread management done by user-level threads library
- Examples
  - POSIX Pthreads
  - Mach C-threads
  - Solaris threads

Kernel Threads

- Supported by the Kernel
- Examples
  - Windows 95/98/NT/2000
  - Solaris
  - Tru64 UNIX
  - BeOS
  - Linux
Multithreading Models

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

Many-to-One

- Many user-level threads mapped to single kernel thread.
- Used on systems that do not support kernel threads.
Many-to-One Model

Each user-level thread maps to kernel thread.

Examples
- Windows 95/98/NT/2000
- OS/2
### One-to-one Model

- User thread
- Kernel thread

### Many-to-Many Model

- Allows many user-level threads to be mapped to many kernel threads.
- Allows the operating system to create a sufficient number of kernel threads.
- Solaris 2
- Windows NT/2000 with the ThreadFiber package
5.15 Operating System Concepts

Many-to-Many Model

5.16 Operating System Concepts

Threading Issues

- Semantics of fork() and exec() system calls.
  - When a thread calls fork() should all threads be duplicated or only the thread that called it?
  - When exec() is called should all threads be replaced?

- Thread cancellation
  - Asynchronous cancellation
    - Terminates immediately
  - Deferred cancellation
    - Periodically checks and then terminates itself
    - Cancellation points
Threading Issues

- Signal handling
  - Deliver signal to the thread to which it applies
    - Synchronous Signals
  - Deliver signal to every thread in a process
  - Kill
  - Deliver signal to certain threads in a process
    - Non blocking thread(s)
  - Assign a specific thread to receive all signals
    - Solaris

- Thread pools
  - Common pool of threads to do the work
    - Control on the number of threads in a system

- Thread specific data

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 2 is a version of UNIX with support for threads at the kernel and user levels, symmetric multiprocessing, and real-time scheduling.

LWP – intermediate level between user-level threads and kernel-level threads.

Resource needs of thread types:
- Kernel thread: small data structure and a stack; thread switching does not require changing memory access information – relatively fast.
- LWP: PCB with register data, accounting and memory information; switching between LWPs is relatively slow.
- User-level thread: only new stack and program counter; no kernel involvement means fast switching. Kernel only sees the LWPs that support user-level threads.
Solaris Process

- process id
- memory map
- priority
- list of open files

Solaris process

Windows 2000 Threads

- Implements the one-to-one mapping.
- Each thread contains
  - a thread id
  - register set
  - separate user and kernel stacks
  - private data storage area
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).

Java Threads

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

```java
class Worker1 extends Thread {
    public void run() {
        System.out.println("I am a Worker Thread");
    }
}
```

Creating the Thread

```java
public class First {
    public static void main(String args[]) {
        Worker runner = new Worker1();
        runner.start();
        System.out.println("I am the main thread");
    }
}
```
The Runnable Interface

```java
public interface Runnable
{
    public abstract void run();
}
```

Implementing the Runnable Interface

```java
class Worker2 implements Runnable
{
    public void run()
    {
        System.out.println("I am a Worker Thread");
    }
}
```
Creating the Thread

```java
public class Second {
    public static void main(String args[]) {
        Runnable runner = new Worker2();
        Thread thrd = new Thread(runner);
        thrd.start();

        System.out.println("I am the main thread");
    }
}
```

Java Thread Management

- **suspend()** – suspends execution of the currently running thread.
- **sleep()** – puts the currently running thread to sleep for a specified amount of time.
- **resume()** – resumes execution of a suspended thread.
- **stop()** – stops execution of a thread.
Java Thread States

Operating System Concepts 5.31