

CS 5523 Lecture 16: Thread concepts

- Questions on laboratory 2 and midterm
- Thread state
- Strategies for threading servers
- Kernel-level vs user-level threads
- Java threads
- Examples

Why threads?

Upside:

- Cheaper to create than processes
- Cheaper to switch between than processes
- Easier to share data

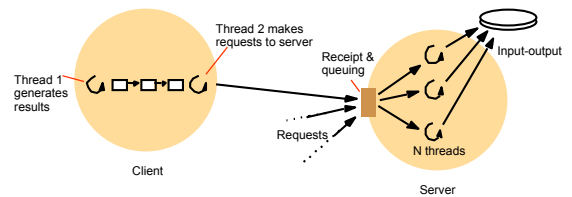
Downside:

- Less robust against programming errors
- Have to share resources allocated to the execution environment

What is a thread?

- Thread – an abstract data type representing flow of control within a process
- Multiple threads can be created within a single execution environment (a process) to achieve parallelism

Figure 6.5
Client and server with threads

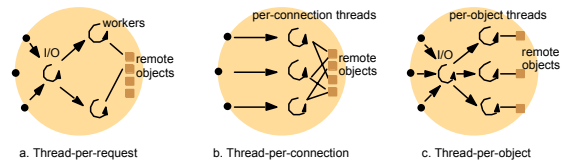


Instructor's Guide for Cookson, Dillman and Knuth. Distributed Systems: Concepts and Design. Edn. 3
© Addison-Wesley Publishers 2000

Thread state:

- Thread ID
- Machine state (registers, program counter, stack pointer)
- Priority
- Signal mask
- Error designation (e.g., errno)

Figure 6.6
Alternative server threading architectures (see also Figure 6.5)



Instructor's Guide for Cookson, Dillman and Knuth. Distributed Systems: Concepts and Design. Edn. 3
© Addison-Wesley Publishers 2000

Thread operations:

- **Thread management:**
 - ┆ creation
 - ┆ destruction
 - ┆ changing priority or other attributes
- **Synchronization**
 - ┆ join
 - ┆ condition variables
 - ┆ locks

Hybrid approaches:

- Application contains user-level scheduler to manage threads within a process
- Kernel allocates virtual processors for a process
- Kernel makes upcalls to notify application's thread scheduler of events:
 - ┆ Virtual processor allocated
 - ┆ Scheduler activation blocked
 - ┆ Scheduler activation unblocked
 - ┆ Scheduler application preempted

Kernel-level threads:

- Each thread is a schedulable entity
- Threads compete for process resources on system-wide basis
- Can take advantage of multiple processor
- Operating system must provide support
- Context switch involves the kernel and can be expensive

Examples of different approaches:

- User-level threads – pthreads
- Kernel-level threads – NT, Solaris Lightweight Processes
- Hybrid approaches – Solaris threads + LWP
FastThreads

Java threads are user-level, however, their mapping to underlying thread support is implementation dependent. (Threads are scheduled differently on different underlying OS's.)

User-level threads:

- Run on top of existing operating systems
- Encapsulated with processes and invisible to kernel
- Schedule by a runtime system that is part of process code
- Each library function and system call is enclosed by a jacket to let the run-time library i

Java threads:

- A thread is an object that represents a thread of execution
 - Each thread has a priority (higher numbers given preference)
 - A thread has a name (which can be set in constructor)
 - A thread can be either a user thread or a daemon thread
 - Program exits when only daemon threads remain
 - Two ways to make a thread:
 - ┆ Declare class to be a subclass of Thread
 - ┆ Declare a class to implement Runnable
- Both cases you need a `run` method and do a `start`

Java thread groups:

- *Every thread is a member of a thread group*
- *Thread groups allow you to manage groups of threads as a unit (e.g., you can suspend or interrupt all members of a group with the group `resume`, `stop` or `suspend` methods)*

- *Runtime system puts a thread in a group when it is constructed*

```
public Thread(ThreadGroup group, Runnable runnable)
public Thread(ThreadGroup group, String name)
public Thread(ThreadGroup group, Runnable runnable, String name)
```

Otherwise in current thread group

- *Attribute changes for a thread group do not apply to threads already in the group*

- **See** <http://java.sun.com/docs/books/tutorial/essential/threads/threadgroup.html>

Examples of threading code:

- *Look at examples from*

<http://developer.java.sun.com/developer/Books/javaprogramming/threads/chap13.pdf>

Reading:

- *Classic Thread Synchronization*

<http://developer.java.sun.com/developer/Books/performance2/chap3.pdf>

- *Java Monitors and Synchronization*

<http://developer.java.sun.com/developer/Books/performance2/chap4.pdf>

- *Read Chapter 6 of CDK*

- *Work on CDK questions*

6.4, 6.8, 6.9, 6.10, 6.14, 6.23, 6.24, 6.25