Comp 310
Computer Systems and Organization

Lecture #7
Threads
(Part 1 – Basic Architecture)

Prof. Joseph Vybihal

Announcements

- C Tutorials

T & TH 10:30-3:30
Trottier 3rd floor
Email: Web CT

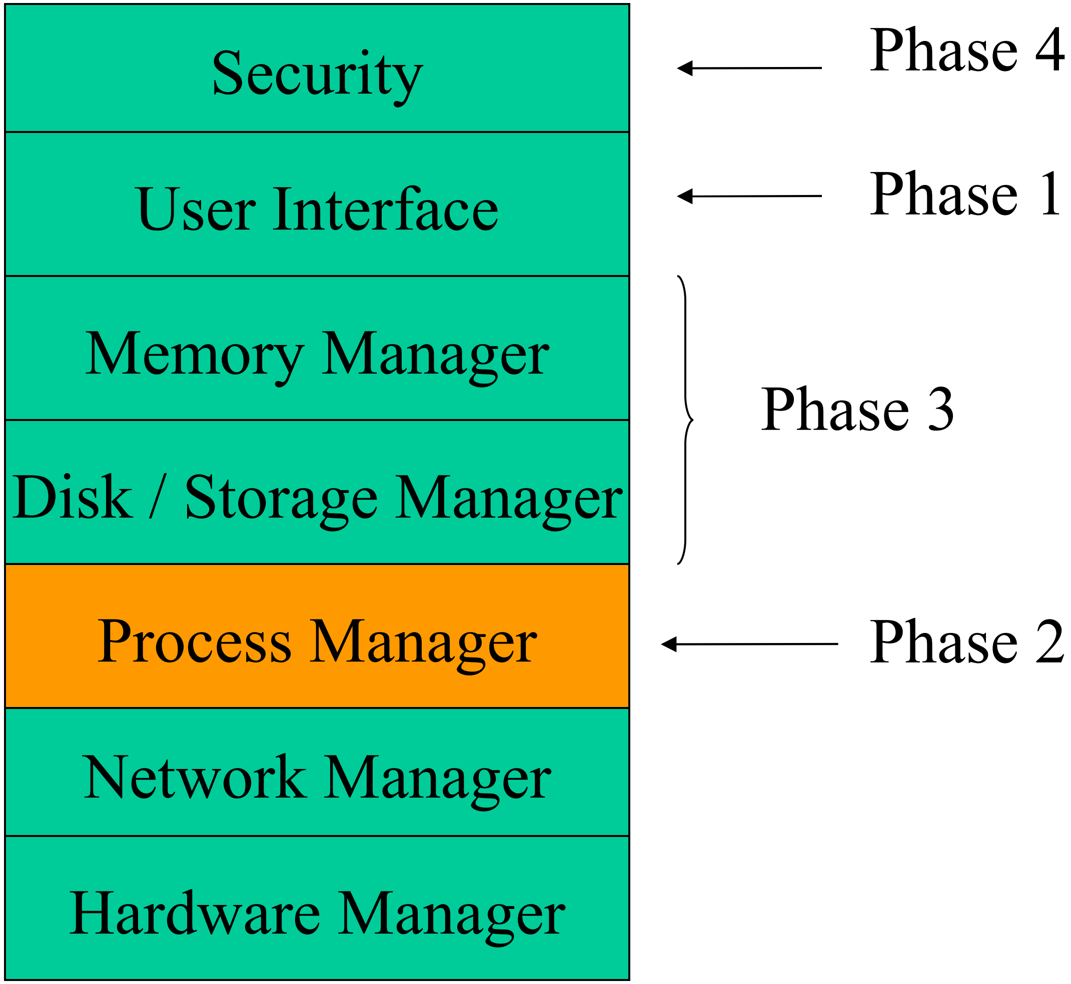
C Tutorial #2: TBA

Web TA
By Appointment
Email: Web CT

Unix & C Tutorial #1

Basic OS Architecture

(Course Table of Contents)



Security

User Interface

Memory Manager

Disk / Storage Manager

Process Manager

Network Manager

Hardware Manager

Phase 4

Phase 1

Phase 3

Phase 2

Part 1

Threads vs. Process

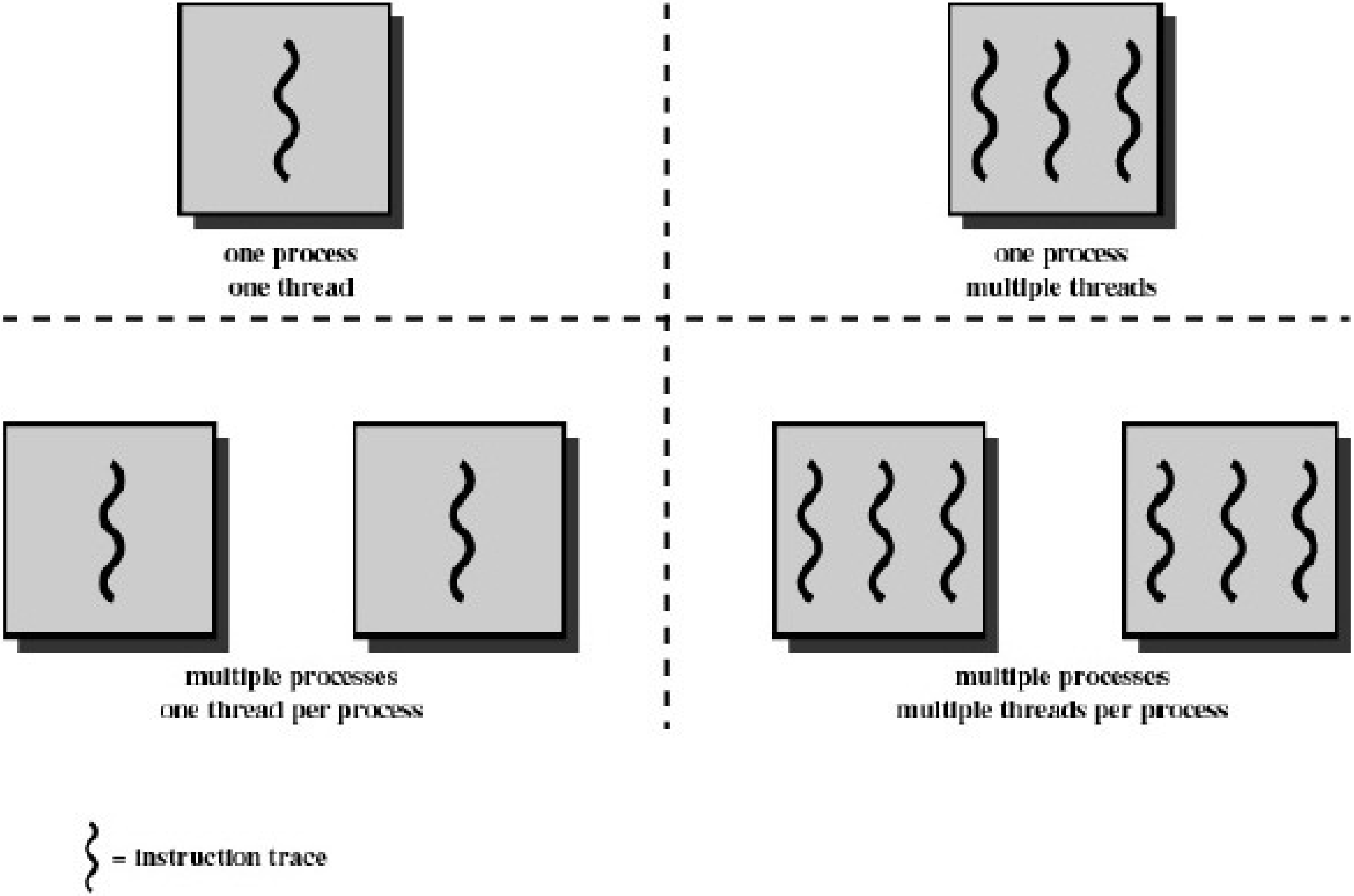
Purpose: in-depth view of OS run-time environment

Question

- What is a thread and how is it different from a process?

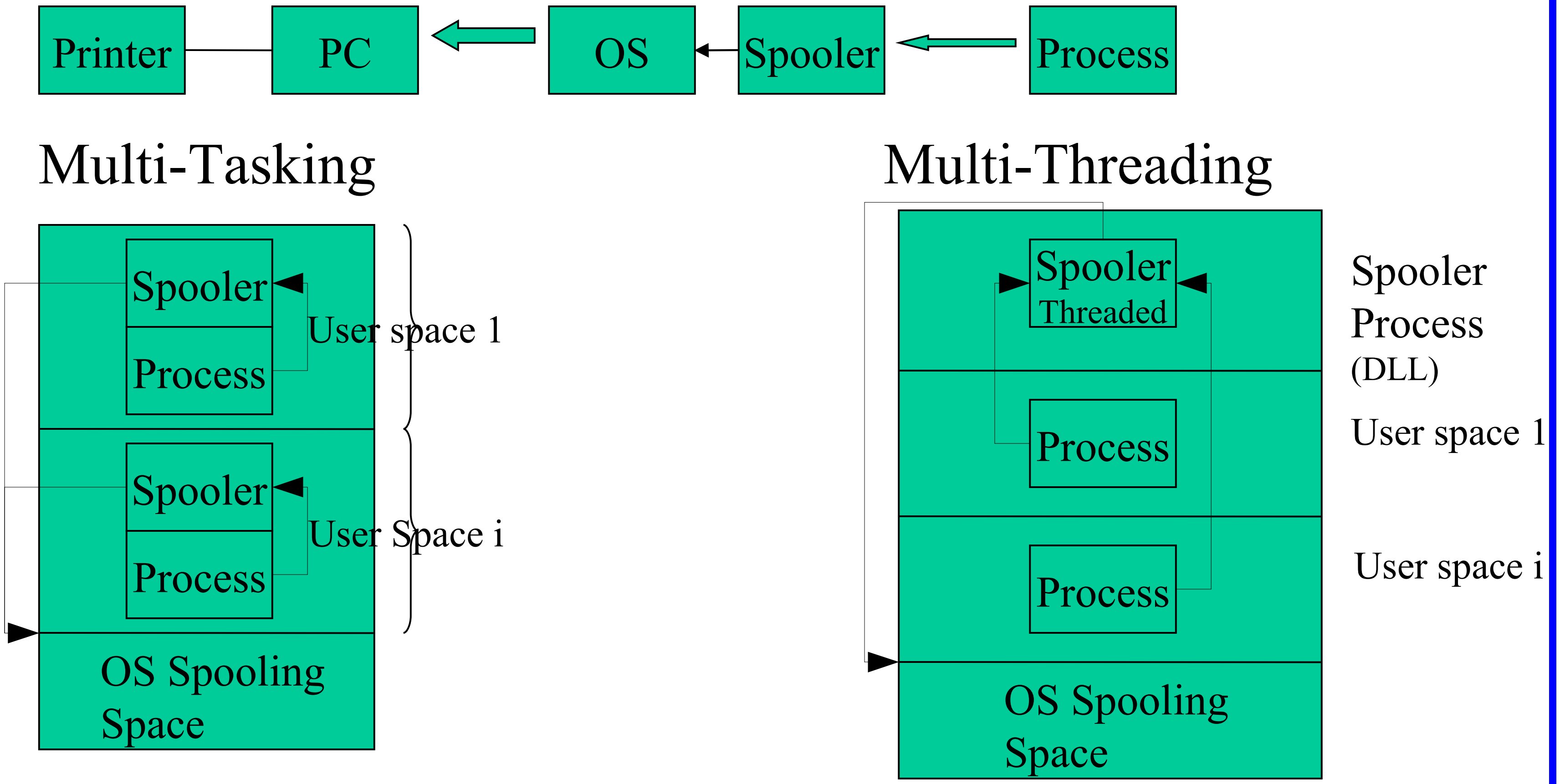
Task switching view...

Processes


one process
one thread
one process
multiple threads
multiple processes
one thread per process
multiple processes
multiple threads per process
= instruction trace

An Example

EX: Operating System Print Spooler



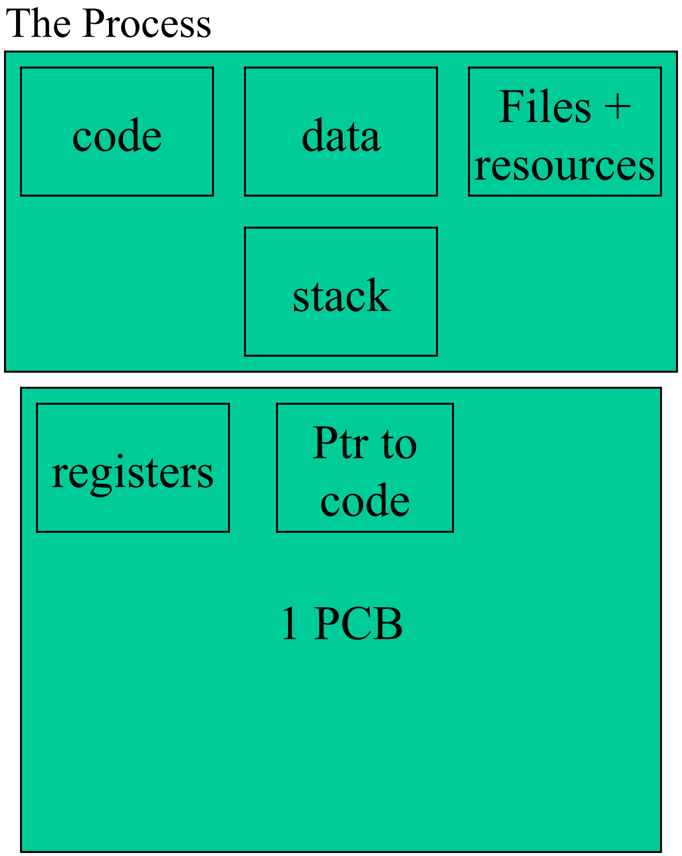
Question

- How would the print spooler code be different in multi-tasking vs. multi-threading?

Part 2

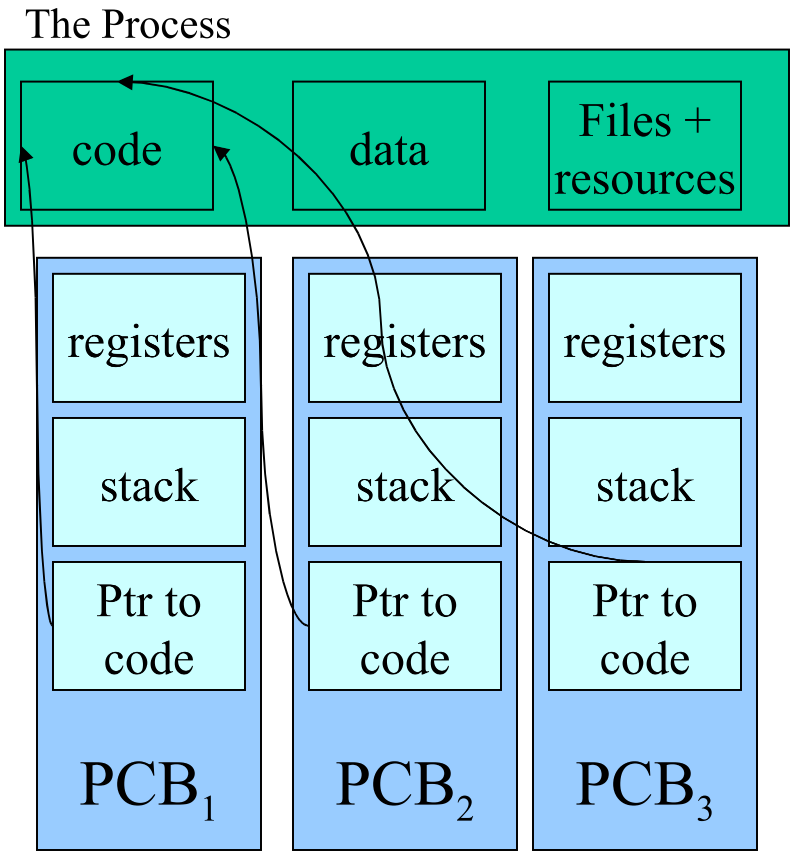
About Threads

What are they?



A Process
(no threads)

Each process has one PCB



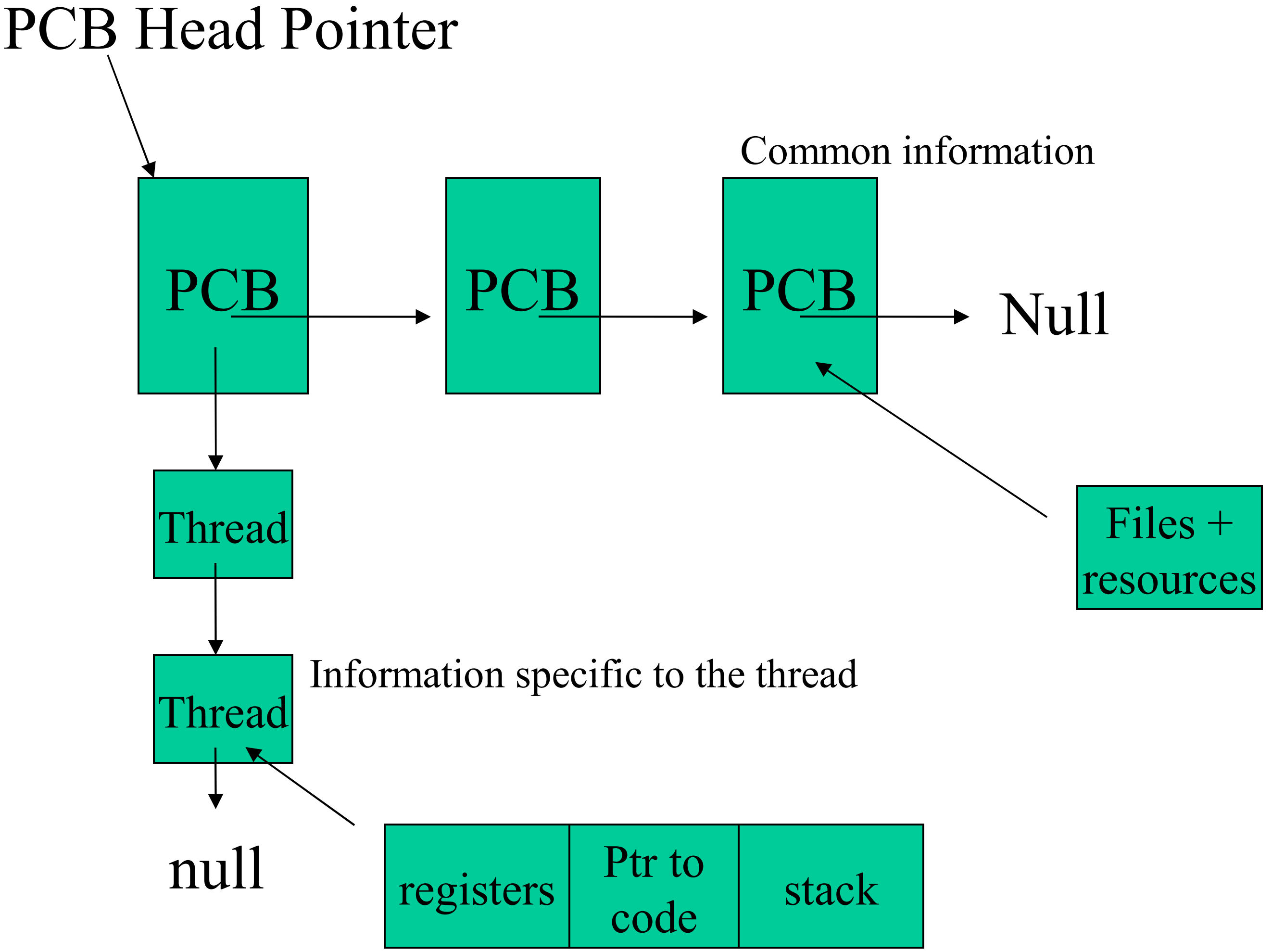
A Process
(with 1 or more threads)

By Definition

- A lightweight process (LWP)
- Contains:
  - Thread ID
  - Program counter
  - Register set
  - A run-time stack
- Shares:
  - Code
  - Data
  - OS Resources (files, interrupts, etc.)

Threads as a data structure

RAM:
-Code
-Static data



Example Usage

- Browser threads:
  - Display web page
  - Retrieve web page from network

- Word Processor threads:
  - Display graphical text and images
  - Read keyboard
  - Background spell check

Benefits

- Responsiveness
  - – Resource request blocked but can still execute
  - – Other users do not need to wait for you
- Resource Sharing
  - – DLL and Printer Queues, …
- Economy
  - – Process creation is more expensive than thread creation
- Multiprocessor Architectures

Question

- What would the complete memory look like with processes and OS PCB/Thread management?

In other words, how could we diagram it?
How would the OS execute everything?

Part 3

Threading Models

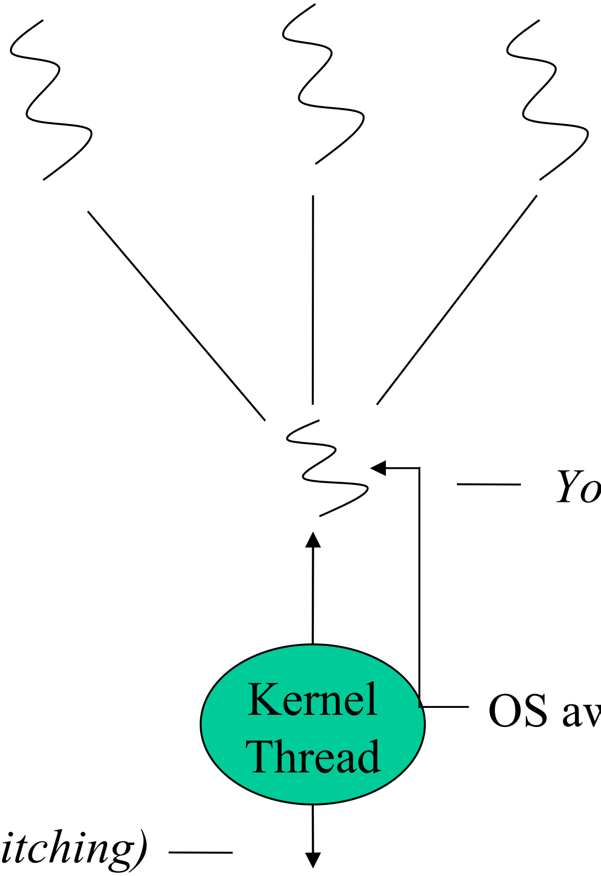
Two Thread Types

- User threads
  - Supported by the compiler, library, or your programming
  - Fast and easy to build
  - OS is not aware of them (quanta distributed across all)
  - Problem: if 1 thread blocked then entire process blocked.
  - Examples: Solaris 2
- Kernel threads
  - Supported directly by OS
  - Slower to build and uses a lot of OS resources
  - OS is aware of each thread, so no blocking problem
  - Benefit: Can take advantage of multi-CPU systems
  - Examples: Windows 2000, Solaris 2, Tru64 Unix, …

Threading Models

- The Many-to-One Model

If one blocks,
then kernel thread
sleeps & all become
blocked



User threads

Your code does task switching

Kernel
Thread

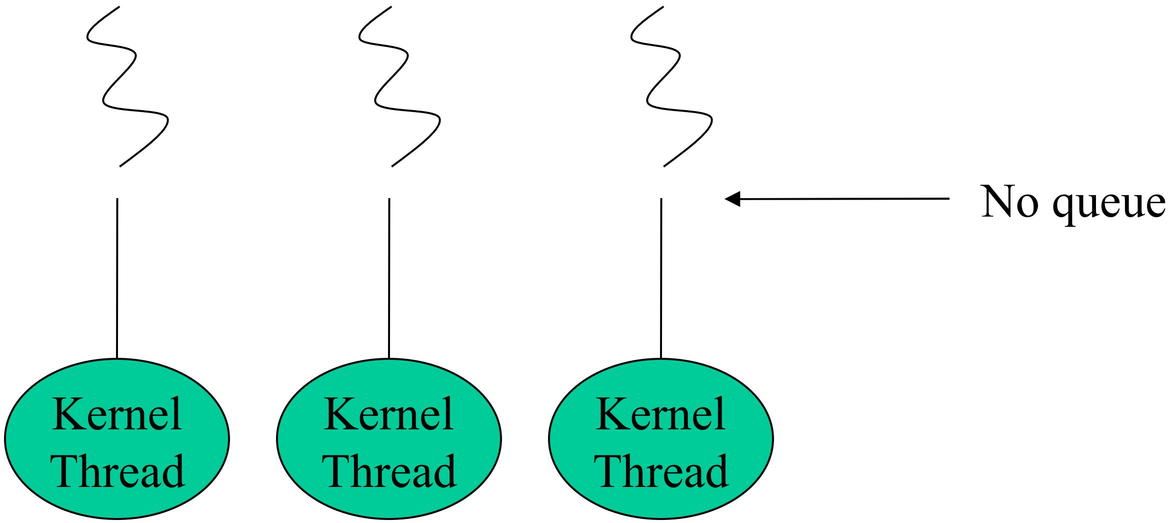
OS aware of only this thread

Kernel Queue (task switching)

Threading Models

- The One-to-One Model

(Multi-processor friendly)



(m threads * k bytes) + (n T-kernels * c bytes) = B bytes, s.t. B is large, therefore: n < LIMIT

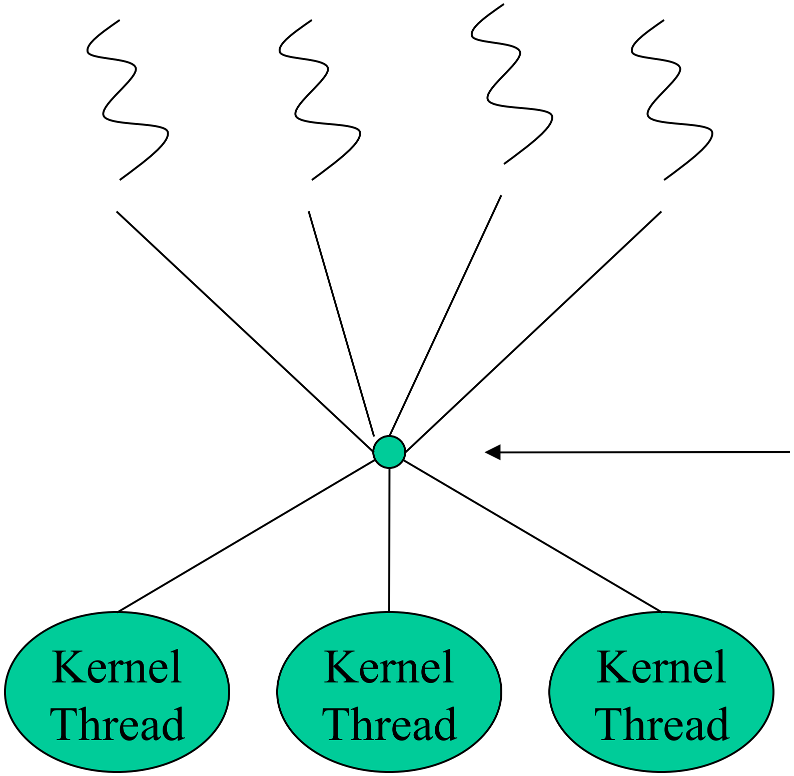
Question

- How might the OS be programmed for multiple processors in one-to-one?

How could we diagram it?
How would the OS manage it?

Threading Models

- The Many-to-Many Model

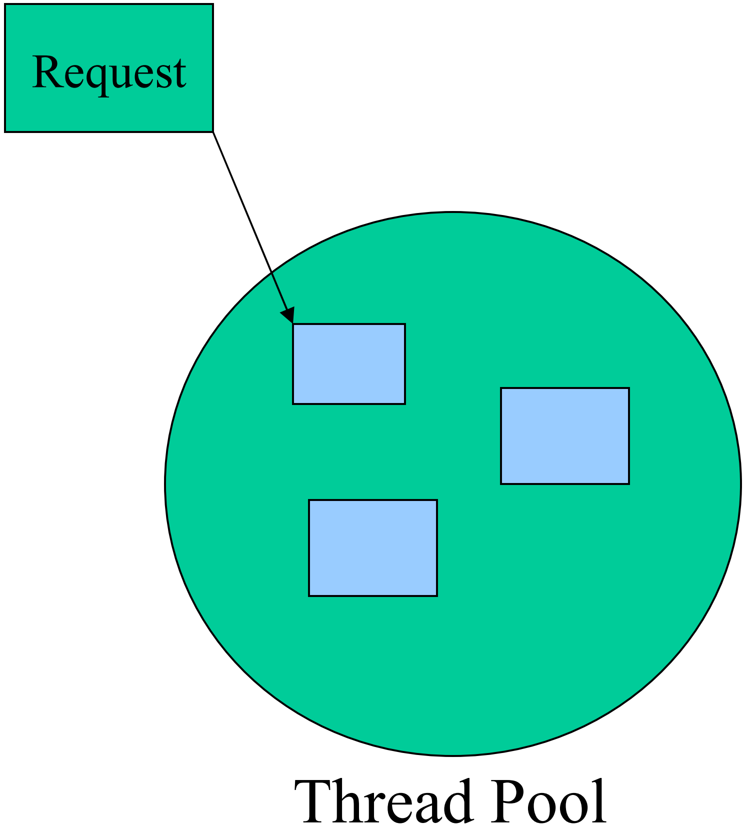


n user threads
m kernel threads
n >= m
m < limit
n > 0, no limit

Kernel Queue (task pool switching)

Multi-processor friendly

OS Resource Limits



- At OS boot a predefined number of threads are created.
- When a request is issued, it is assigned a sleeping thread from the pool, or gets queued.
- Benefits:
  - Time (no create/kill)
  - Limits (manage CPU)

Part 4

At Home

Things to try out

1. Write C programs to:
   - Fork
   - Exec
   - System

Try to overload your computer with multiple child processes and threads.

(do this gradually…the system staff don’t allow it here...)