Announcements/Reminders

- Exam 1 on Friday: same place/same time
- CS Women's group meeting: first Wednesday of the month at 12:30 in CS 150. In April: a speaker from IBM
- Lecture schedule will be adjusted a little: next week: monitors, networks, and interprocess communication.
- Lab 2: Due March 14 or 24th?

What Abstractions Does an Operating System Provide?

What Abstractions Does an Operating System Provide?

- Process (abstracts the processor)
- Virtual Memory (abstracts the memory hierarchy)
- File System (abstraction of physical disks and CDs)
- I/O Devices: printers, cameras, etc. (from their physical implementation)

What Protection Does an Operating System Provide?

What Protection Does an Operating System Provide?

- Memory space
- CPU utilization
- From other machines
- Resolve I/O contention
- Critical section support
- Kernel/User operation

Architectural Features Motivated by OS Services

Architectural Features Motivated by OS Services

OS Service	Hardware Support
Protection	Kernel/User mode
	Protected Instructions
	Base and Limit Registers
Interrupts	Interrupt Vectors
System calls	Trap instructions and trap vectors
I/O	Interrupts or Memory-Mapping
	Direct Memory Access
Scheduling, error recovery, billing	Timer
Synchronization	Atomic instructions
Virtual memory	Translation look-aside buffers

Properties of Interrupts and Traps

Properties of Interrupts and Traps

- Representation of events that have taken place inside of the processor or in an I/O device.
- Traps: internal processor events (floating point exception, system calls, page fault).
- Processor responds to an interrupt by saving state and passing control to the appropriate handler.
- On completion of the interrupt handler, state is restored and a process is returned to the CPU (but not necessarily the same process).

Process Data Structures

Process Data Structures

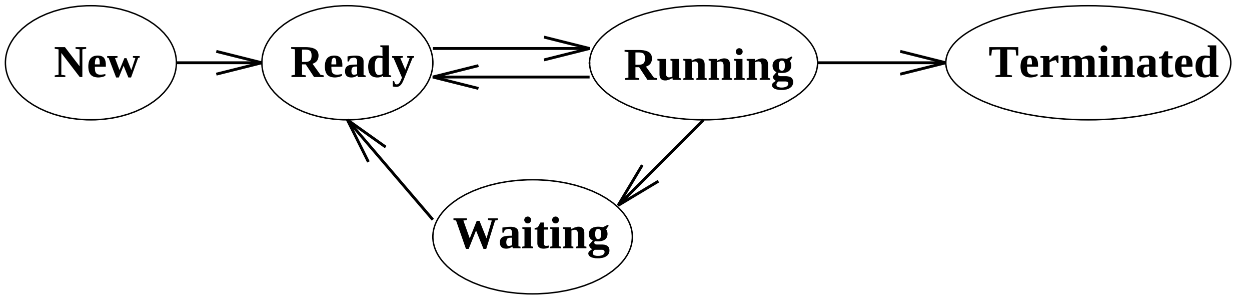
- Executable code (the program)
- Stack (and a stack pointer)
- Heap (and associated pointers).
- Program counter
- Static data
- Register values
- Process execution state
- Set of OS resources in use (e.g., open files and communication ports).

Process Execution State

- Execution state of a process indicates what it is doing

new:	the OS is setting up the process state
running:	executing instructions on the CPU
ready:	ready to run, but waiting for the CPU
waiting:	waiting for an event to happen (e.g., I/O completion)
terminated:	the OS is destroying this process

- As the program executes, it moves from state to state, as a result of the program actions (e.g., system calls), OS actions (scheduling), and external actions (interrupts).



High-Level Issues in OS Design

High-Level Issues in OS Design

- Monolithic vs layered structure
- Components placed in kernel vs user space

⇒ What are the costs in making these decisions?

Processes vs Threads

- Threads share memory and data segments, but have separate stacks and program counters.

$\Rightarrow$ Context switch overhead is lower for threads.

Kernel- vs User-Level Threads

User-level threads do not involve the kernel at all:

- No system calls to manage user threads

⇒ Even less overhead to switch between user threads.

- Can implement custom scheduler without touching the kernel

But:

- A single user thread can block the entire set of threads by making a system call.
- The kernel can't participate in the fair management of the CPU resources for a set of threads.

Scheduling

- How to measure performance?
- Why different scheduling algorithms?
- What different scheduling algorithms are there and what are their properties?
- Preemptive vs non-preemptive: can a process/thread be interrupted without explicitly giving up the CPU?

Process Synchronization

- Why synchronization?
- Hardware support for synchronization is really necessary, but inconvenient to use (by a user). Mechanisms include disabling interrupts and special instructions (e.g., Test&Set). What are their advantages and disadvantages?
- Language support for synchronization: Locks, Semaphores, and Monitors (Monitors will not be on the test).
- Make sure that you understand the semaphore examples that we have looked at in class.

Exam Distribution

- 25 pts: Hardware and OS Design
- 20 pts: Processes and Threads
- 29 pts: Scheduling
- 24 pts: Synchronization (including a short programming question)