

CS162, Spring 2002
Section 2
Steve Martin

Welcome to Section

- Personal introduction, undergrad experience, grad school, research interests, etc. (For new attendees this week.)
- My class web page: <http://www.cs/~emilong/classes/cs162>
- Office hours: Tuesday 10-11am, Friday 2-3pm, f11 Soda

More Project Info

- Mainly need to understand basics of language, classes, etc. (not all of the crazy APIs)
- You will not be using Java threads (directly) - so don't really need to understand them either
- Only need to look at a few files for the first project (threads/*.java and some of machine/*.java)
- First project design due on Wed 2/18/02
 - o Example design document online now (Click on 'Projects')
 - o Keep them short: 3 pages max
- GET STARTED NOW! Don't wait until last minute
- Expectations
 - o How you will solve each of the project sections
 - o How you will test each project section
- Please use CVS:
 - Will really help your project
 - CVS tutorial later in section

Address spaces for isolation and abstraction

- Give each program its own ADDRESS SPACE:
- **QUESTION:** What is an address space?
 - o **ANSWER:** Range of "virtual addresses" (usually in range of 0 to 2^{32}) which it can access
 - Program thinks that each virtual address corresponds to its own "private memory"
 - Important as this ISOLATES processes
- OS maps virtual memory addresses to physical memory address with the help of the MMU (in hardware)
 - o More on h/w support later
- **QUESTION:** What if we have less physical memory than virtual memory?
 - o **ANSWER:** Paging: OS swaps memory to and from disk

- Get into all of the details later in the course
- Regardless of paging, address spaces isolate the memory accesses that a program can make
 - Programs can only access their own memory, not that of other processes or the OS
 - So if program X writes to "*" 0xdeadabba" then program Y reads from "*" 0xdeadabba", X and Y really writing to different addresses
- **QUESTION:** What happens if program X reads/writes memory it doesn't have access to?
 - **ANSWER:** It will get trapped by memory control hardware and killed. "segmentation fault: core dumped"!
- A nice feature to support for inter-process communication is shared memory: Will be discussed later in the course

Difference between PROCESS and THREAD

- A "process" generally refers to a single program that you run, with its own code, address space, etc.
 - The main thing that the operating system deals with
- A "thread" is an entity corresponding to a single flow of control through that program
 - Within the program, the unit of scheduling
- One process can have many threads - those threads all share the same address space, but have their own stack, CPU registers, etc.

Multiprogramming

- One of the trickiest aspects of OS design to understand
- The illusion to support: Multiple programs on the same hardware which each think they have infinite resources and the whole machine to themselves
- The reality: Single machine with limited resources (one CPU, memory, network, disk, etc.)
- How does it all work?

How does Multiprogramming work?

- How many threads of execution can run on a CPU at once?
 - So, how many processes?
- **QUESTION:** So if I start running a program (i.e. Netscape), how do I prevent it from spinning on the CPU and just hanging the machine?
 - Note that Win 3.1 used to have this problem; programs had to explicitly yield to the OS

- **ANSWER:** Program is interrupted somehow and OS takes over to handle the interrupt.
 - o When can this happen?
 - Hardware timer fires off
 - Other hardware triggers an interrupt (disk, network, keyboard . . .)
 - Program calls a SYSTEM CALL
 - Program explicitly yields the CPU (via a system call)
 - o The OS can then decide to PREEMPT the program and run another on the processor.
 - Can decide to CONTEXT SWITCH to another program

What is a CONTEXT SWITCH?

- OS does the following:
 - o Save state of previous process in a data structure called the PCB.
 - What is saved?
 - Registers
 - I/O descriptors
 - Stack register
 - Program counter
 - Memory pointers (eg to page table)
 - o Restores machine registers to state of new program
 - o Sets up stack pointer, etc.
 - o Sets up other state (I/O descriptors, etc.)
 - o Sets up PAGE TABLES to map address space of new program
 - More on this later
 - o Flushes TLB
 - More on this later
 - o Calls special instruction (sometimes) to "return from system call": resume CPU from user program state at user's protection level
- Note that we can't do much of this without some help from hardware. What do we need?
 - o MMU: To provide virtual->physical address mapping
 - o Page tables constructed by OS
 - o TLB is a cache on the virtual->phys mappings
 - o Hardware timers: To allow preemption
 - o Protection levels: Disallow user code from doing certain things (i.e. mucking with page tables, etc.)

Issues with Threads and Concurrency (things to think about)

- How to pick next thread to run? (SCHEDULING.)
- Want to maximize fairness and still have low response time
 - o Much detail on this later in the course
- How to let threads interact safely? (SYNCHRONIZATION.)
 - o I.e. two threads each writing to the same shared variable
 - o How to avoid having one thread overwrite the other's value

- **QUESTION:** What is a RACE CONDITION?
- **ANSWER:** When some event between multiple threads of control is not strictly regulated—i.e. you can make no guarantees about what the state will be at any point in time.

Quick CVS Primer

- Rather than write the commands on the board, a walkthrough on the web under 'Projects'
- Basic model: Single REPOSITORY which stores the "master" version of the code, as well as all previous versions
 - o Users do NOT edit anything in the repository - rather, they "check out" a copy of the files, and edit that copy
- After making changes, you "commit" the changes back to the repository
- Other users only see those changes if they do an "update"
- Repository will be created for each project group - stay tuned

IMPORTANT CVS CAVEATS:

Note that CVS does not allow you to rename files - you need to "remove" the file and then "add" the file under the new name. In general it's best not to rename files, since this loses all of the editing history for that file.

Also note that CVS does not allow you to remove or rename directories once they have been added to the repository. **THIS IS VERY IMPORTANT!!** Do not add directories to the repository temporarily - you can never get rid of them. This is a real drawback to CVS, but it's something you just have to live with.