Welcome to CS162!

About your TA

Name: Angela Juang
Email: ajuang@berkeley.edu (Put [CS 162] in the subject line)
Website: http://inst.eecs.berkeley.edu/~cs162-ta
Sections: Tuesday 10-11am, 320 Soda
11-12am, 320 Soda

Office Hours: TBD

Announcements

- Make sure to log in with your cs162-xx account sometime this week
- Make sure you are in CS 162 on Piazza!
- You should be forming project groups – signups by this Friday on the CS162 website
    - Groups of 4-5 people
    - Everyone needs to be in the same section
    - You need to sign up for at least TWO possible section times – we'll try to make everyone happy but please be as flexible as possible

Processes and Threads

We're used to being able to run programs on our computers, but how do operating systems manage all those programs? A process is what the OS uses to represent an executing program. Multiple processes can be executing at a time, so the job of the OS is to provide scheduling (so that all processes get CPU time) and protection (so processes don't crash or modify each other).

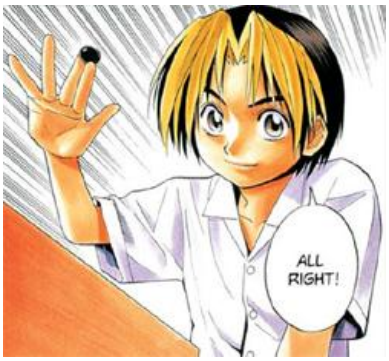
We can take care of protection between processes by using address translation. Address translation gives each process a distinct mapping to physical memory so that processes have their own address space and don't have access to another process' address space. (Processes can have shared address space too, though – this might be done for communication purposes.) The kernel also protects itself from corruption by user processes with dual-mode operation. With dual-mode operation, user processes run in user mode, while the kernel runs in kernel mode; important resources/commands are reserved for kernel mode. To access these resources, processes must use system calls to request a trap to kernel mode to service the request.

To deal with concurrency, each process is broken down into a collection of threads. A thread is a single stream of execution within the process; threads belonging to the same process can run concurrently with each other but do not need to be protected from each other. Therefore, memory and I/O state corresponding to the entire process are shared between threads and stored by the process control block (PCB), but individual stack pointers, registers, and other such information are kept private to the thread and stored in the corresponding thread control block (TCB). The OS can give CPU time to different threads/processes by performing a context switch. With each context switch, the current state of the running thread/process will be saved into the corresponding TCB/PCB and the new one will be loaded in.

CS 162 – Fall 2011 TA: Angela Juang
Discussion Notes Week #01

Coming up…

You will have 4 projects this semester that build on each other. Each project will have the following timeline:



- Design phase (~ 1 week)
- Design review (20 mins)
- Implementation phase (~ 1 week)
- Code due

During a design phase, you and your team will come up with a plan for how you're going to code the project, and write a design doc specifying all the details of the design you came up with. The projects are made to be somewhat open-ended when it comes to implementation so that you get a chance to make decisions on data structures, classes, methods, etc. yourself. We'll provide the basic mechanics of the game and some interfaces for you to start off with.

After the design review, your TA will go over your design with you in a design review and give you feedback on it, making sure you're on the right track so you can complete your code implementation during the next week. During the implementation phase, we will be running a partial autograder for you to test your code on. Please note that the test autograder is NOT a replacement for testing on your own – relying solely on the test autograder to catch your bugs will likely cause you to miss tests that will come up on the complete autograder used for assigning project grades.

There are no slip days for the projects – each day your code is late, you'll incur a 10% penalty on your project grade. Please get started on projects as early as possible!

To do the projects, you will need to know Java. The recommended IDE for this course is Eclipse, but you are free to use whatever editor you prefer. (I'm looking at you, vim enthusiasts…) We will also be using SVN for version control; if you prefer to use git, please let me know and we can get that set up.