Course Introduction

CS 537 – Spring 2013

Operating Systems

Michael Swift

Today's agenda

- Administrivia
  - course overview
    - course staff
    - general structure
- What is an operating system?
- History

Course overview

- Everything you need to know will be on the course web page:

  http://www.cs.wisc.edu/~cs537-2

  - Schedule
  - Readings
  - Homeworks
  - Writings
  - Projects

- But to tide you over for the next hour …
  - course staff
    - Mike Swift
    - TBA
  - general structure
    - Lectures do introduce material
    - Text book readings help further understanding for assignments
    - sections will focus on C programming, the projects, quizzes, writing, and homework
    - we really want to encourage *discussion*, both in class and in section

Workload

- This class has a significant amount of work
  - No midterm, Optional Final
  - 5-6 Programming projects (some individual, some group)
  - 7-8 Quizzes in section
  - Some homeworks (~ 6)
  - Dates are not flexible
- If you're going to drop this course
  - please do it soon!

Programming

- All programming is in C
  - All operating systems (almost) are written in C
  - Most high-performance code is written in C
- You will get an opportunity to learn about
  - revision control for group projects
  - makefiles to automate compilation of larger programs
  - debugging
- Example:

```
#include <stdio.h>
int l;int main(int o,char **O,
int I){char c,*D=O[1];if(o>0){
for(l=0;D[l              ];D[l
++]-=10){D   [l++]-=120;D[l]-=
110;while   (!main(0,O,l))D[l]
+=   20;   putchar((D[l]+1032)
/20   )   ;}putchar(10);}else{
c=o+     (D[I]+82)%10-(I>l/2)*
(D[I-l+I]+72)/10-9;D[I]+=I<0?0
:!(o=main(c/10,O,I-1))*((c+999
)%10-(D[I]+92)%10);}return o;}
```

Computers

- All programming projects will be graded by running them on a CSL workstation
  - It is fine to do the projects on your own machine
  - In general they can be done on MacOS or Windows (with CygWin) as well
  - It is your responsibility to make sure your code works on a CSL machine before turning it in.
- There are many computer labs on the 1st floor for your use (1350-mumble, 1351-king,1370-adelie)
- If you have not used Unix, please attend the CSL Linux orientation next week
  - Signs around the building say when/where

Grades

- Exams: 25%
  - Final is optional
- Programming: 50 %
- Quizzes: 35%
  - I will drop your lowest score
- Homeworks and class participation: 15%

Readings

- Textbook: Operating Systems: Three Easy Pieces
  - Readings will be assigned to cover material from lecture
  - You can do readings before or after lecture, based on your learning style
  - … But most helpful before lecture
  - … Very helpful before quizzes

Quizzes

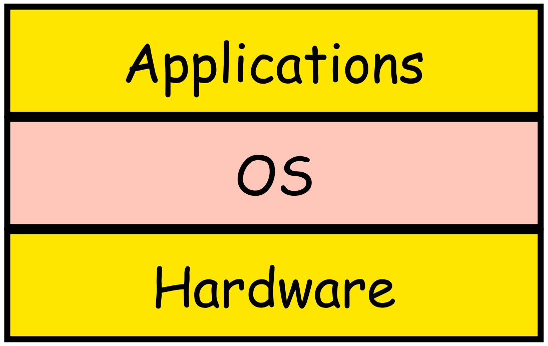
- Quizzes are held at the beginning of the discussion section
  - Don’t miss them and don’t be late
  - They take about 15 minutes
- The questions are similar in difficulty to the homework questions

Course Content

- In this class we will learn:
  - what are the major components of most OS's?
  - how are the components structured?
  - what are the most important (common?) interfaces?
  - what policies are typically used in an OS?
  - what algorithms are used to implement policies?
- Philosophy
  - you may not ever build an OS
  - but as a computer scientist or computer engineer you need to understand the foundations
  - most importantly, operating systems exemplify the sorts of engineering design tradeoffs that you'll need to make throughout your careers – compromises among and within cost, performance, functionality, complexity, schedule …

What is an Operating System?

- An operating system (OS) is:
  - a software layer to abstract away and manage details of hardware resources
  - a set of utilities to simplify application development



  - “all the code you didn’t write” in order to implement your application

The OS and hardware

- An OS mediates programs' access to hardware resources
  - Computation (CPU)
  - Volatile storage (memory) and persistent storage (disk, etc.)
  - Network communications (TCP/IP stacks, ethernet cards, etc.)
  - Input/output devices (keyboard, display, sound card, etc.)
- The OS abstracts hardware into logical resources and well-defined interfaces to those resources
  - processes (CPU, memory)
  - files (disk)
    - programs (sequences of instructions)
  - sockets (network)

Why bother with an OS?

- Application benefits
  - programming simplicity
    - see high-level abstractions (files) instead of low-level hardware details (device registers)
    - abstractions are reusable across many programs
  - portability (across machine configurations or architectures)
    - device independence: 3Com card or Intel card?
- User benefits
  - safety
    - program "sees" own virtual machine, thinks it owns computer
    - OS protects programs from each other
    - OS fairly multiplexes resources across programs
  - efficiency (cost and speed)
    - share one computer across many users
    - concurrent execution of multiple programs

What Functionality belongs in OS?

- No single right answer
  - Desired functionality depends on outside factors
  - OS must adapt to both user expectations and technology changes
    - Change abstractions provided to users
    - Change algorithms to implement those abstractions
    - Change low-level implementation to deal with hardware
- Current operating systems driven by evolution

Major Themes in OS

- Virtualization
  - Taking physical hardware and making a software version that is sharable, easier to use, more powerful
  - Examples:
    - CPU: we can run two programs at the same time
    - Memory: programs see a linear range of addresses but underlying DRAM is shared in 4kb chunks
    - Disk: we use files/folders, disk internally has blocks
- Concurrency
  - Maintaining correctness when many things happen at once
  - Examples:
    - Code on 2 CPUs try to increment the same variable
- Persistence
  - Keep data safe across system crashes/reboots