Operating Systems
CMPS 311

Professor Scott Frees

sfrees@ramapo.edu

Today's Topics

- Brief Overview/Introduction
- Course information and policies

Contact Information

- Office Location: G315
- Office Phone #: 201-684-7726
- Office Hours: TBD
- Email: sfrees@ramapo.edu
- Course Web Site: moodle.ramapo.edu
- My Website: http://pages .ramapo.edu/~sfrees

Course Information

- Text Book:
  - Operating System Concepts (9th Edition) by Silberschatz, Galvin, and Gagne. ISBN: 0471694665
    - Older Editions are OK, but you might be missing some of the later topics….

Grading Policy

- 60% Quizzes/Exams
  - 3 Exams during semester (20% each)
- 20% Final Exam
- 15% Homework Assignments
- 5% Weekly **short** quizzes

- Late Homework / Programs Policy
  - **A 10-point penalty will be applied *per day* late.**
  - **Assignments more than 5 days late will not be graded under any circumstances.**

What is an Operating System?

- The OS manages the interaction between hardware and applications
  - Nothing more than a program, made of executable code just like any other application.
  - Special privileges and responsibilities

High-Level Responsibilities of OS

- CPU Allocation

These are the main topics of this course.

- Memory and Storage Management

- Regulate and Provide Access to Peripherals

What to expect…

- Concepts + Programming
  - Some struggle with the concepts (reading before class will *really* help)
    - **Please ask questions whenever something is not clear!**
  - Programming will be challenging if you are not familiar with C or C++
    - **Come to office hours!**

- **This class works best when its interactive – ask questions!**

Topic #1
Services & System Calls

Detailed OS Responsibilities

- Process Scheduling - Multiprogramming
  - Ensure efficient use of CPU
- System calls and services
  - Insulate processes from hardware, other processes
- Memory & Storage Management
  - Hierarchy of memory, File system
- User management & Security
- Communication (Inter-process & internet)

3 Ways to Define an OS

1. Services the OS provides to the user
2. Interface OS provides to applications
3. Interface/Interconnections between OS components

Services Provided to User

- **User Interface**
  - Program Execution
  - I/O operations
  - File-system manipulation
  - Communication
  - Error Detection

3 Ways to Define an OS

1. Services the OS provides to the user
2. Interface OS provides to applications
3. Interface/Interconnections between OS components

OS's interface to applications

- OS exposes functionality to applications via **system calls**
- System are normally C or C++ subroutines or functions
  - From a programmer's perspective, they are nothing special…
  - Their implementation is **very** different however…

Dual Mode Execution

- Implemented using Trap/Interrupt and Mode-Bit
  - Must be supported by hardware
- Privileged Instructions cannot be executed by user code
  - Allows OS to make sure programs “play nice”.

Example: writing to a file

- Java:

```
FileOutputStream fos = new FileOutputStream("out.txt");

fos.write("Hello");
```

- C:

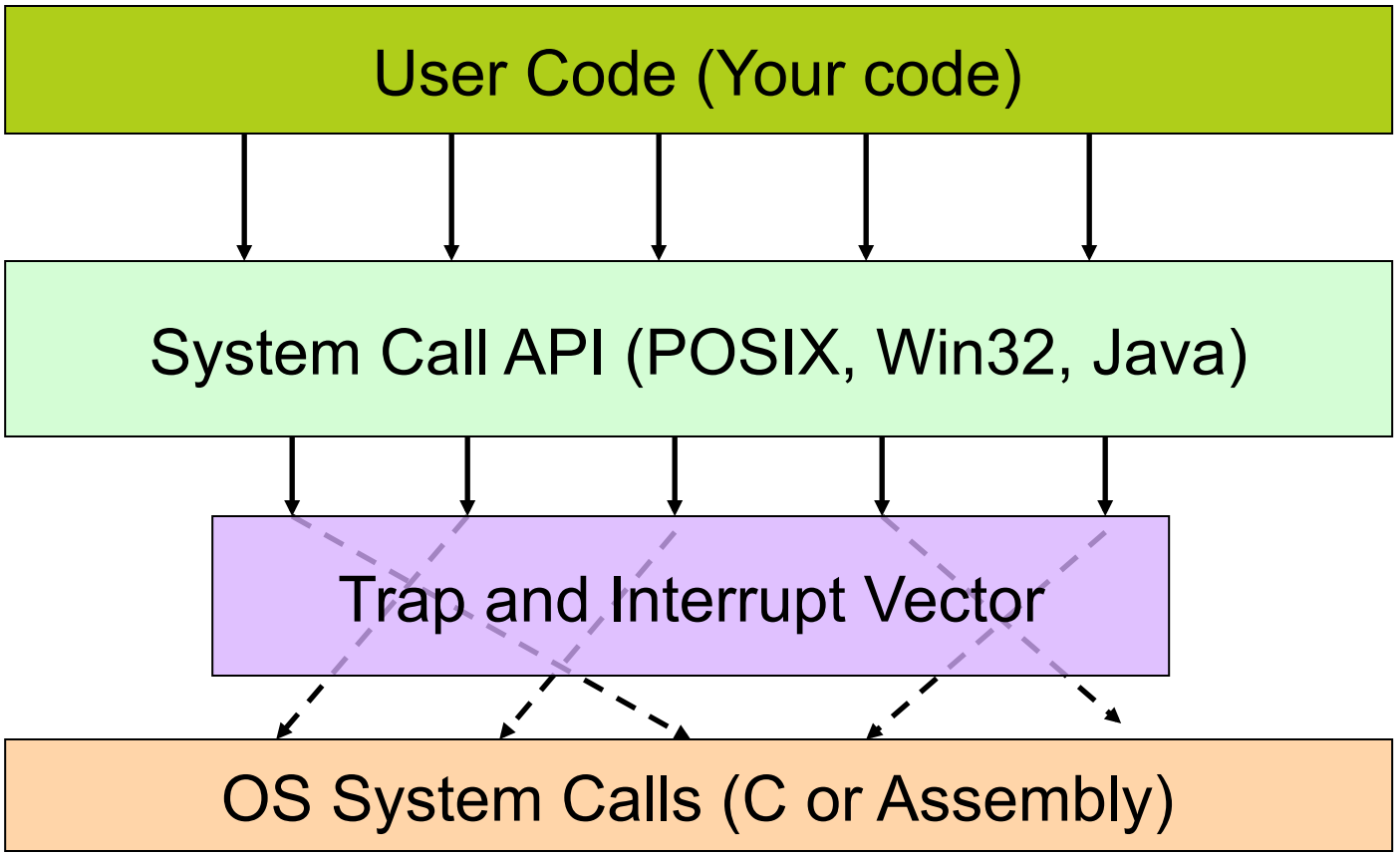
```
int fd = fopen("out.txt");

fprintf(fd, "Hello");
```

- These calls are eventually implemented using system calls
  - The system call depends on the operating system

Portability through an API

- All OS's provide different system calls
  - This is *partially* why MS word doesn't run on Linux and iPhoto doesn't run on Windows!
- POSIX: common API "wrapper"
  - Fully supported by most flavors of UNIX, Linux, BSD, Mac OS X
  - Partially supported by Windows

Programming Layers



Types of System Calls

- Process Control
- File Management
- Device/Peripheral Management
- Memory Management
- Communications
- ***System programs*** *use system calls to expose functionality to users*

3 Ways to Define an OS

1. *Services the OS provides to the user*
2. *Interface OS provides to applications*
3. Interface/Interconnections between OS components
   - There are many ways to design and build an OS

OS Components?

- We've already seen a little of the "scheduler".
- Other components:
  - Memory Manager
  - File System
  - User Management
  - Network Communication

How do you write an OS?

- Early operating systems written entirely in assembly code.
  - IBM System OS/360:
    - Millions of lines of assembly,
    - complex,
    - limited functionality (by today's standards),
    - Buggy
- Almost all modern operating systems written in C or C++
  - May contain portions of assembly code for performance

Designing an OS

- OS is still quite complex and difficult to write!
- Design impacts both functionality and quality
- Simple Design: Monolithic
  - MS-DOS
  - Original versions of UNIX
  - Why?

Varying Priorities

- Many implementation choices for:
  - Scheduler
  - Memory Management
  - Filesystems
  - Security/Communication
  - etc.

Modern Design Principles

- Layered
  - Advantages: Layers interchangeable, easy debugging
  - Disadvantage: Interdependencies
- Micro-Kernels:
  - Advantages: Flexible, Secure
  - Disadvantage: Takes Discipline, Inefficient
- Modules:
  - Advantages: Flexible, Secure, Efficient

Homework

- Homework 1 is posted on moodle
- PLEASE READ THE DOCUMENT POSTED ABOUT GETTING ACCESS TO POSIX MACHINES
  - Ask me right away when you are stuck!
- Read Chapters 1-3 on *Processes*