**Department of Physics,
Computer Science & Engineering**

CPSC 410 – Operating Systems I

Chapter 0: Introduction

Keith Perkins

Topics

- Admin
  - prereqs, text, eval, etc…
- What is an Operating System (OS)?
- Where are OS coming from?
  - History by functionality
- Closing remarks
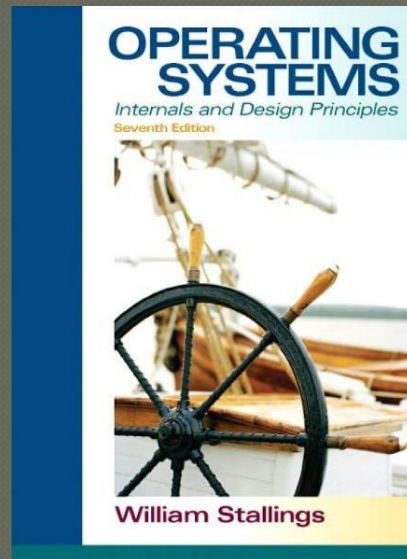 - OS & Software Engineering

Admin: Your Background

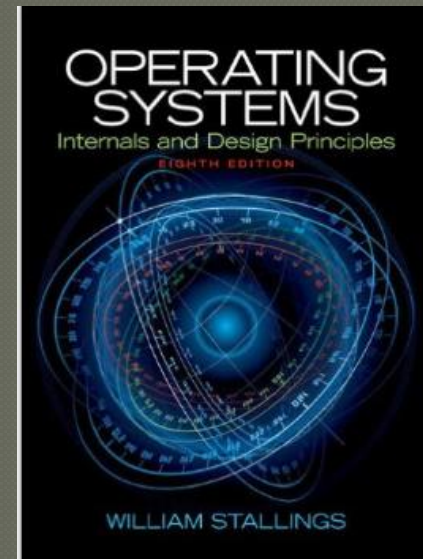- Some high level programming language
- CPEN 214, CPSC 327 (C++)

Admin: Text

- Operating Systems Internals and Design Principles, William Stallings, 7th or 8th Edition, Pearson, 2011. (do not need access code!)

or

Admin: Evaluation

- Multiple projects ← Probably 3 but may be as many as 5
- 1 midterm ← Lateish in the semester
- 1 final

Admin: What you get from this class

- Some C++ experience
- How an OS works (multitasking)
- Scheduling
- Memory Management
- I/O management and File Management ⬅ Wish
- Threads and concurrency ⬅ Most Useful

Admin: Language

- C++
  - A brief review
  - One C++ starter project
  - The rest will be OS specific
- Why C++

Admin: Compiling and debugging

- C and C++ both compile to an executable
- Can use many compilers (clang, gcc, G++ …)
- Show you how to use gcc and g++
- Show you IDEs for C++
  - And I hope to demonstrate the advantage of an IDE

Admin: Development Environment

- Could use vim, g++, gdb, valgrind, tmux for a command line only dev environment
- Or an Integrated Development Environment (IDE)
- Lots to choose from, Codeblocks, Netbeans,Ms Visual Studio, Eclipse CDT...Clion
- We will use Eclipse CDT

Admin - Git

- You will use version control professionally, best to learn it now
- See 'Git-The simple guide' on course website
- In class demo

Admin – Linux –Why?

- Most of the worlds OSs are based on unix like kernels (Mac, Android, Linux)
- Client side, games, laptops , desktops
  - Windows
- Handsets
  - -Android
- Servers
  - Linux
- Most high performance remotely hosted (cloud) machines are linux (AWS, paperspace, etc)
- See Linux - tutorials on course website
- In class demo

What is an OS?

- An Operating System is software that acts as:
  - 1) an Extended Machine
    - To provide consistent, simpler, high-level <u>interfaces</u> (<u>abstractions</u>) between user programs and hardware
  - 2) a Resource Manager
    - To provide an orderly and <u>controlled access</u> to hardware.

13

What is an OS?
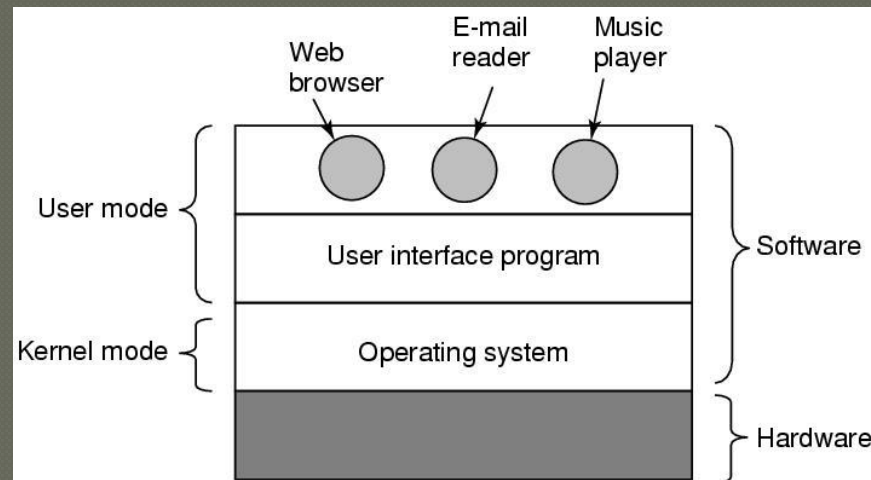
If OS didn't exist…
user programs would access hardware directly…
who wants to create programs dealing with <u>details</u> in
{ memory | disks | printers | network } ?

- An Operating System is:
  - 1) an Extended Machine
    - To provide consistent, simpler, high-level <u>interfaces</u> (<u>abstractions</u>) between user programs and hardware
  - 2) a Resource Manager
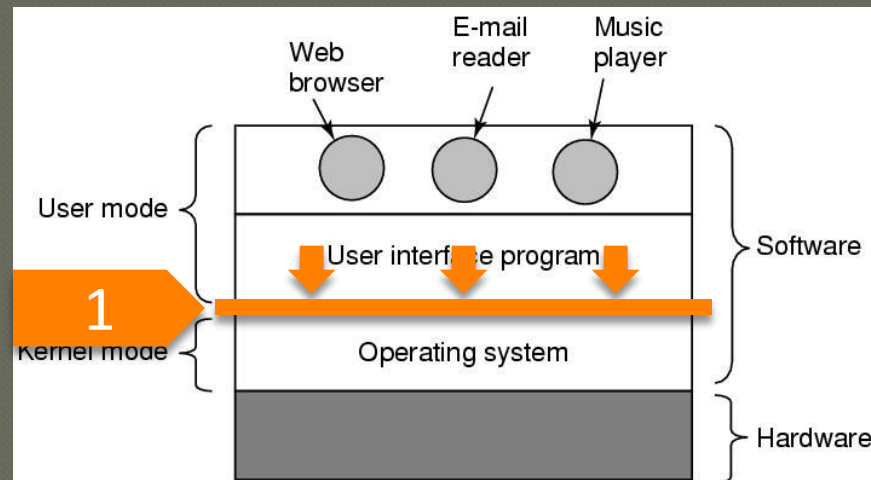   - To provide an orderly and <u>controlled access</u> to hardware.

simple abstractions

1

14

What is an OS?

- An Operating System is software that acts as:
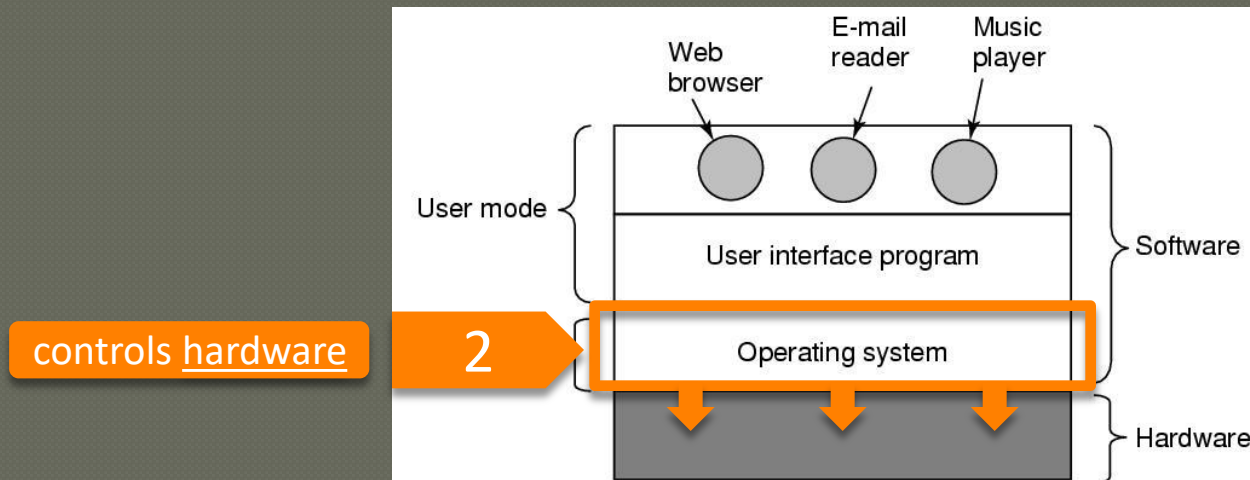
  - 1) an Extended M...

    - To provide consist...
      (abstractions) betw...

  imagine CPSC150

  If OS didn't exist…
  user programs would access hardware directly…
  what if 2+ programs concurrently access the same
  { memory | disks | printers | network } ?

  - 2) a Resource Manager

    - To provide an orderly and controlled access to hardware.

controls hardware → 2

Web browser   E-mail reader   Music player

User mode

User interface program

Software

Operating system

Hardware

- 1. Using I/O devices
  - Problem
    - Programs should not handle I/O devices directly
      - e.g., spinning disk, moving disk arm, find tracks to read/write
  - Solution
    - Build modules (device drivers) to:
      - manage the interaction with I/O devices.
      - provide a high-level & consistent abstraction across device types
        - e.g. the concept of a "file" accessed through read/write operations.
  - Result
    - Programs dealt with a well-defined & simple interface for a range of device types (e.g., disks/tapes, printers)

- 2. Idle CPU Time (between job runs)
  - Problem
    - Because of how <u>early computers</u> were used (programmers booked <u>time slots</u>), the <u>CPU was idle</u> most of the time <u>waiting for input</u> (from one programmer to the next) and <u>output</u> (e.g., printer, disk).
  - Solution
    - Queue incoming programs, so that they can be executed one immediately after another (batch processing), and queue outgoing results for processing (spooling).
  - Result
    - CPU is freed sooner to take next job

History of OS

- 3. Idle CPU Time (waiting for I/O)
  - Problem
    - Since the CPU is faster than I/O devices (e.g., printer, disk), the CPU must sit idle waiting for I/O operations to finish.
  - Solution
    - Allow another job to take over the CPU while an I/O operation in the current job finishes (multiprogramming).
    - Use memory buffers (to hold I/O data) and interrupts (to signal the CPU when an I/O operation is over).
  - Result
    - CPU performs cooperative multitasking (non-preemptive)

- 4. Concurrent interactive programs
  - Problem
    - Several users provide real-time input and expect real-time output from programs, but dedicated computer time is expensive.
  - Solution
    - Allow several programs to alternate execution (multiprogramming) using time interrupts (preemptively)
  - Result
    - Users access a computer through multiple terminals, while given the illusion that their programs execute concurrently (at the same time).

OS Lessons & Challenges

- Lessons for Software Engineering
  - OS are some of the larger software projects ever attempted.
    - Hundreds of programmers
    - Millions of lines of code
  - How to tackle such an effort while…
    - creating few bugs?
    - being adaptable to change?

Next Time

- C++ outline
- Compiling, linking, running, debugging
  - The hard way
  - The easy way
- If you would like to follow along then please follow the instructions outlined in 'Install Eclipse CDT on Linux' on the course website