Operating Systems: Introduction
CS400

Week 1: 16th Jan
Spring 2020
Oliver BONHAM-CARTER
Consider this…

Group Discussion

- Go online to find a small software that is an OS, or fulfills a role as one. Discuss as a group. You could even choose one from these slides.
  - What are the main differences between this OS and other popular OS’s? (i.e., MacOS, Windows, Linux)
  - Why was your OS developed?
  - What does the media report about the OS?
  - What sorts of functions does this OS provide?

Please write down your discussion points to help you introduce your OS to the class.
ClassDocs: All Class Materials

- We will be using GitHub to manage all class material. Clone your repository using the following:
  - **ssh**
    - git@github.com:Allegheny-Computer-Science-400-S2020/classDocs.git
  - **https**
    - https://github.com/Allegheny-Computer-Science-400-S2020/classDocs.git
Class Repositories

- **PULL** your classDocs before class.
  
  ```bash
  git pull
  ```

- **PUSH** assignment repos to submit homework
  
  ```bash
  git add -A
  git commit -m "My mesg"
  git push
  ```
Docker for Running Software

- We will be using Docker in lab and class
- Please be sure that your machine will work with the regular Docker, not Docker ToolBox.
- Verify: www.cs.allegheny.edu/canirundocker
Atom: Suggested for Programming

- We will be programming and Atom facilitates this task
- If you do not already have it, please download it from: https://atom.io/
Please Install Your Software

- We will be using Git and GitHub. Please setup your account by next class at https://github.com/ and also download a Git client software from https://git-scm.com/downloads

- We will also be using the Atom editor to write code. Please download and install your editor from https://atom.io/

- For most labs, we will be using Docker. Please download and install your Docker Desktop installation (note: not the Docker ToolBox) from https://www.docker.com/. Help: https://hub.docker.com/

- If necessary, please help each other to install this software. Or see the department’s Technical Leaders with questions.
To Consider Today ...

- Where is the OS?
- How does an OS control the machine (generally speaking)?
- What is abstraction? And how does an OS help to simplify the interactions between humans and computers?
Chapter 1, Tanenbaum

- Chapter 1 topics:
  - (1.1) What are operating systems (OSs)?
  - (1.3) Computer Hardware
  - (1.4) The Operating System Zoo
  - (1.5) Operating System Concepts
  - (1.6) System Calls
  - (1.7) Operating System Structure

- And more
First: What is an OS???

- *Management in computing*
First: What is an OS???

- We talked about this already, but briefly.
  - Let’s be brief again!

- Your computer is made out of stuff
  - One or more processors
  - Memory: RAM, ROM (perhaps other types)
  - Storage: Hard drives, SATAs, SSDs, Flash, USB-C types of storage (Apple), etc.
  - Printing: paper-based, 3D
  - Types of input and output devices
First: What is an OS???

- All these parts must work together!

Managing all these components requires a layer of software – the operating system
What is the software?

- Allowing software to work together with hardware on system
Where is the OS in the software?

- The OS is the lowest level of software before we transition to the hardware level of computer management.
The OS as a Resource Manager

- Allow multiple programs to run at the same time
- Multiplexing (sharing) resources
  - By *Time*: When a resource is time multiplexed, different programs or users take turns using it. Ex: One CPU must be shared across several different tasks according to *time-sharing*.
The OS as a Resource Manager

- Multiplexing (sharing) resources, continued
- By Space: Instead of taking turns, resources get a (small) part of a resource to use.
- Ex: Memory is used by several tasks simultaneously.
Modes of Use

- Two modes of operation in Computing
  - **Kernel mode** (supervisor mode)
  - **User mode**

- Access
  - The **kernel** mode has complete access to all parts of the machine (software, hardware, internal clocks, memory blocks, storage blocks, etc.) and can execute any instruction the machine is capable of executing.
  - The **user** mode is given *very slight* privileges which allow for basic application launching and some other *high-level* software activity.

Let us consider the “kernel” to be the “brain” behind the OS.
Permissions: Levels of Access

Kernel Space

• Regulates
  • Manage hardware resources,
  • Determine app printer privileges,
  • Maintain the memory pages,
  • Handle hardware restrictions,
  • Display graphics from graphics chips,
  • Allocate time to the CPU for multitasking
  • Access, reallocate app memory
  • Control the computer cooling system
  • Write/read to disk
  • Interface with drivers ...
Permissions: Levels of Access

User Space

- Regulates
  - Update software,
  - Use of email client,
  - Install and use a new browser,
  - Choose any video player,
  - Play a video game,
  - Use a word processor,
  - Online banking applications,
  - Run a spreadsheet,
  - Code software
  - Run a program ...
The Hard/Software Connection

- The OS as an Extended Machine
The Hard/Software Connection

- It is EXTREMELY difficult to work directly with hardware.
- Many types (kinds) of connections from software to make with hardware.
Abstraction: Dealing with the Complexity Using Simplicity

- Directly interacting with the hardware is complicated.
- Software is created to handle this interaction once.
- Other software interacts with this software to form shortcuts for using hardware resources.
Abstraction: Dealing with the Complexity Using Simplicity

- Software drivers for I/O devices
- Drivers enable basic (simple) programming rules to perform complex work.
- Ideally, drivers should work across many different same-type systems such as similar Linux distributions.
Ex1: Some Linux Drivers
Abstraction: Complexity of Working with Storage
Abstraction: Dealing with the Complexity Using Simplicity

- Ex2: saving files is very complicated
  - Determine how quickly disk is turning
  - Attribute the 1’s and 0’s in correct places of disk sectors
  - Check for reading / writing errors
  - Check for bad sectors
The OS Makes Connections

- An API (Application Programming Interface) is a set of defined functions and methods for interfacing with the underlying operating system or another program or service running on the computer.
- Connections by establishing a reference to a software library or importing functions from software such as to dynamic link libraries (dll), in case of Windows.

Programming simple methods

OS drivers

Complex methods for hardware
Abstraction: Making a Cup of Coffee

By the process of abstraction, a the coffee maker hides all but the relevant inputs to make coffee in order to reduce complexity and increase efficiency, and make a cup of joe.
Consider This

- An abstraction model in Python3 programming
- File: abstraction.py
- A method to handle the messy inputs which are necessary by another
- Take a moment to play with this simple, yet overly complicated, program.
Begin()
msg_str = "Hello World"

Get and set relevant parameters, handle memory, call functions, interact with machine code, etc

print(msg_str)
"Hello World"

The OS handles this kind of abstraction for each process to manage systems resources to work together.