Introduction to Operating Systems I

Benjamin Brewster

Tools versus Theory

- C++? Java? *nix? Apple? You’re CS majors, not *nix majors!
- After this class, you should be able to hold an intelligent conversation about any operating system by studying a model OS like UNIX

*NIX

- Why *nix?
  - Stable: good luck crashing it
  - Powerful: dense commands
  - Standard: used everywhere

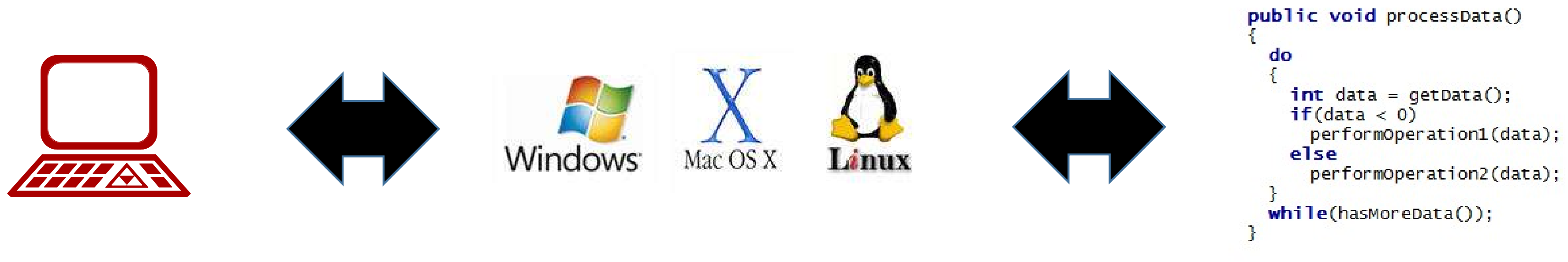
- Worldwide Device Shipments in 2015 (smartphones, tablets, laptops and PCs)
  - Android (Linux): 1.3 billion
  - Windows: 283 million
  - iOS (UNIX): 276 million
  - OSX (UNIX): 21 million
  - Others (Some Linux): 550 million

NIX is 82.3% of non-Other OSs shipped

Source: Gartner, 4/2016

What is an Operating System?

- A software program that sits between software applications and the computational hardware



```
public void processData()
{
  do
  {
    int data = getData();
    if(data < 0)
      performOperation1(data);
    else
      performOperation2(data);
  }
  while(hasMoreData());
}
```

Why are OSs Important?

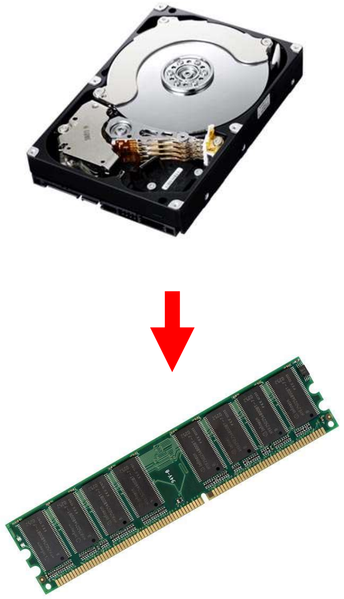
- Most applications interact with the OS in some fashion
- As a programmer, you will need to:
    - Use the capabilities of the OS to do most anything
    - Be aware of the policies and limitations of the OS

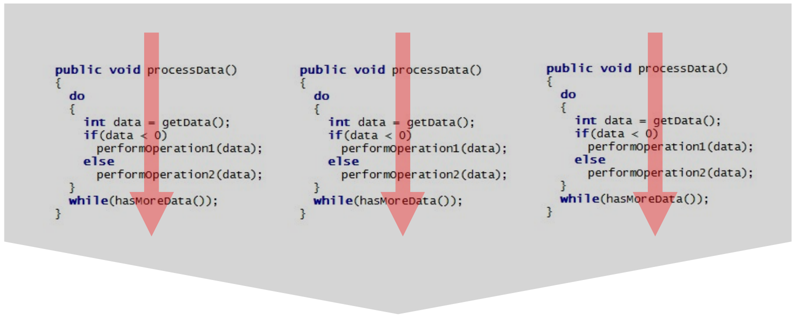
Goals of an Operating System

- Universal Goals
  - Provide convenient software interface to hardware resources
  - Maximize utilization of hardware
  - Solve contention
  - Provide services
- Common Goals
  - Provide security
    - Protect against other buggy applications/crashes
    - Control access to your data by others
  - Support software development
  - Provide standardized software libraries
    - Including a standardized user interface

Definitions

- Program
  - A stored algorithm or plan of execution
- Process
  - A program that has been loaded into memory and is executing
- Thread
  - A line of execution in a process





```
public void processData()
{
  do
  {
    int data = getData();
    if(data < 0)
      performOperation1(data);
    else
      performOperation2(data);
  }
  while(hasMoreData());
}
```

```
public void processData()
{
  do
  {
    int data = getData();
    if(data < 0)
      performOperation1(data);
    else
      performOperation2(data);
  }
  while(hasMoreData());
}
```

```
public void processData()
{
  do
  {
    int data = getData();
    if(data < 0)
      performOperation1(data);
    else
      performOperation2(data);
  }
  while(hasMoreData());
}
```

Standard OS Services

1. Process and thread management
    - Starting a new program (becomes a process & thread)
    - Ending a process/thread
    - Debugging programs/processes

Standard OS Services

1. Process and thread management
   - Starting a new program (becomes a process & thread)
   - Ending a process/thread
   - Debugging programs/processes

2. File and input/output management
   - Organizing bits into meaningful structures: Files
   - Providing interfaces for reading and writing to files
   - Communicating with external devices
   - Organizing files: Directories

Standard OS Services

3. Interprocess communication (IPC)
    - Signals, pipes, network sockets (TCP/IP)
    - Including between two different computers

Standard OS Services

3. Interprocess communication (IPC)
   - Signals, pipes, network sockets (TCP/IP)
   - Including between two different computers

4. Process coordination
   - Contention management leads to shared access

Interacting With the OS

- Users
  - via Graphical User Interface (GUI)
  - via Command Line Shell (|-|4><0|2$)
- Programs
  - via Functions
    - System calls
    - Application Programming Interface (API) - Functions
  - via Network communication
    - Message-based
    - Connection-based

Enjoy!

- If you’re not having fun, you’re (probably) doing it wrong.