Operating Systems: Purpose and Roles

What is an OS?

- No universally accepted definition
  - Is it everything that comes on a computer?
    - Used to be, then came Microsoft (US v. Microsoft, 1998)
    - Now this varies widely
- Program that is always running
  - Ha.

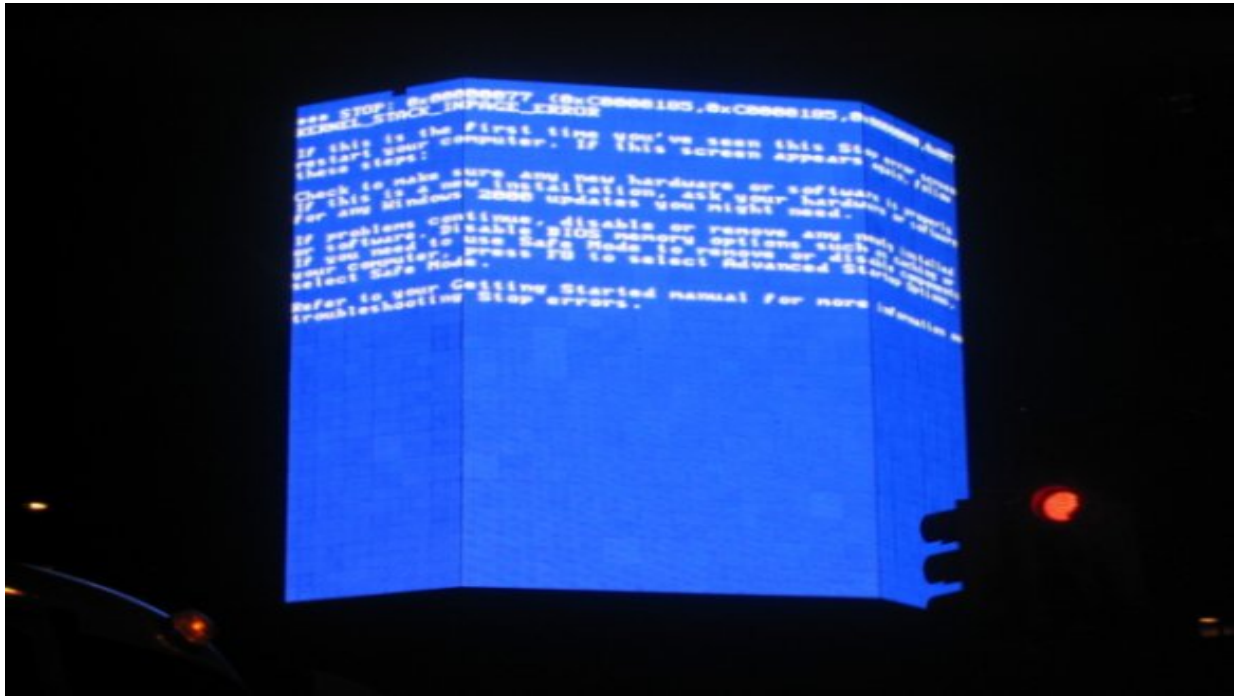
Operating System: A definition

Software that manages a computer's resources

- Makes it easier to write the applications you want to write
- Makes you want to use the applications you wrote by running them efficiently

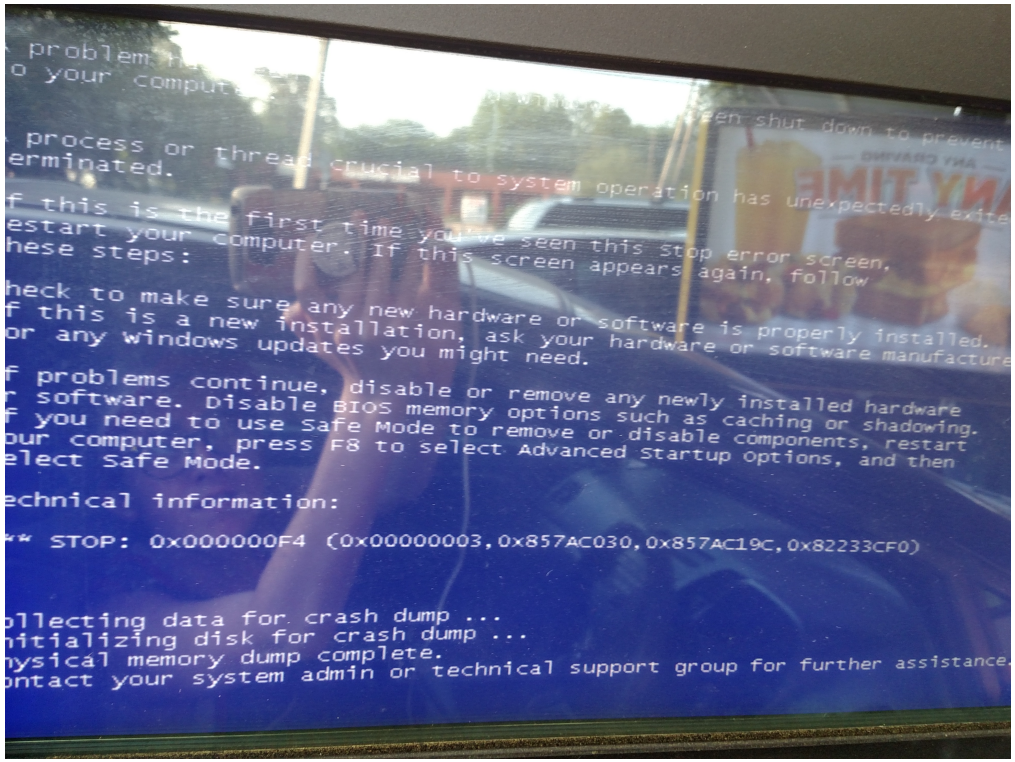
Why Study Operating Systems?

- To learn how computers work
- To learn how to manage complexity through appropriate abstractions
- To learn about system design
  - Performance vs. simplicity, HW vs. SW, etc
  - Design trade-offs made in the past do not necessarily apply now
  - Those made now will not necessarily apply in the future
- Operating Systems are everywhere!



Where’s the Operating System?

Where’s the Operating System?



Where's the Operating System?

Where’s the Operating System?

Operating Systems: More than One Hat

Operating Systems as Referee

- Manages shared resources
  - Coordinates multiple applications and users to achieve fairness and efficiency
- Isolation
  - Protects processes from one another
    - One application's bugs should not crash another (or the whole system!)
    - If it does crash, should fail gracefully
- Communication
  - Allow processes to work together

Operating Systems as Illusionist

Illusion of resources that are not really present

- Virtualization: processor, memory, screen space
- Entire computer!

Operating Systems as Glue

Provides standard interfaces to the hardware

- Simplifies application design and facilitates sharing
- Decouples hardware and application development
- Start, stop, and clean up after a program
- Examples: File system, virtual memory, networking

Evaluating an Operating System

Reliability

- OS does exactly what is designed to do
  - The ability of a computer-related hardware or software component to consistently perform according to its specifications.
- In theory, a reliable product is totally free of technical errors (yeah, right)
- Availability: percentage of time system is useful

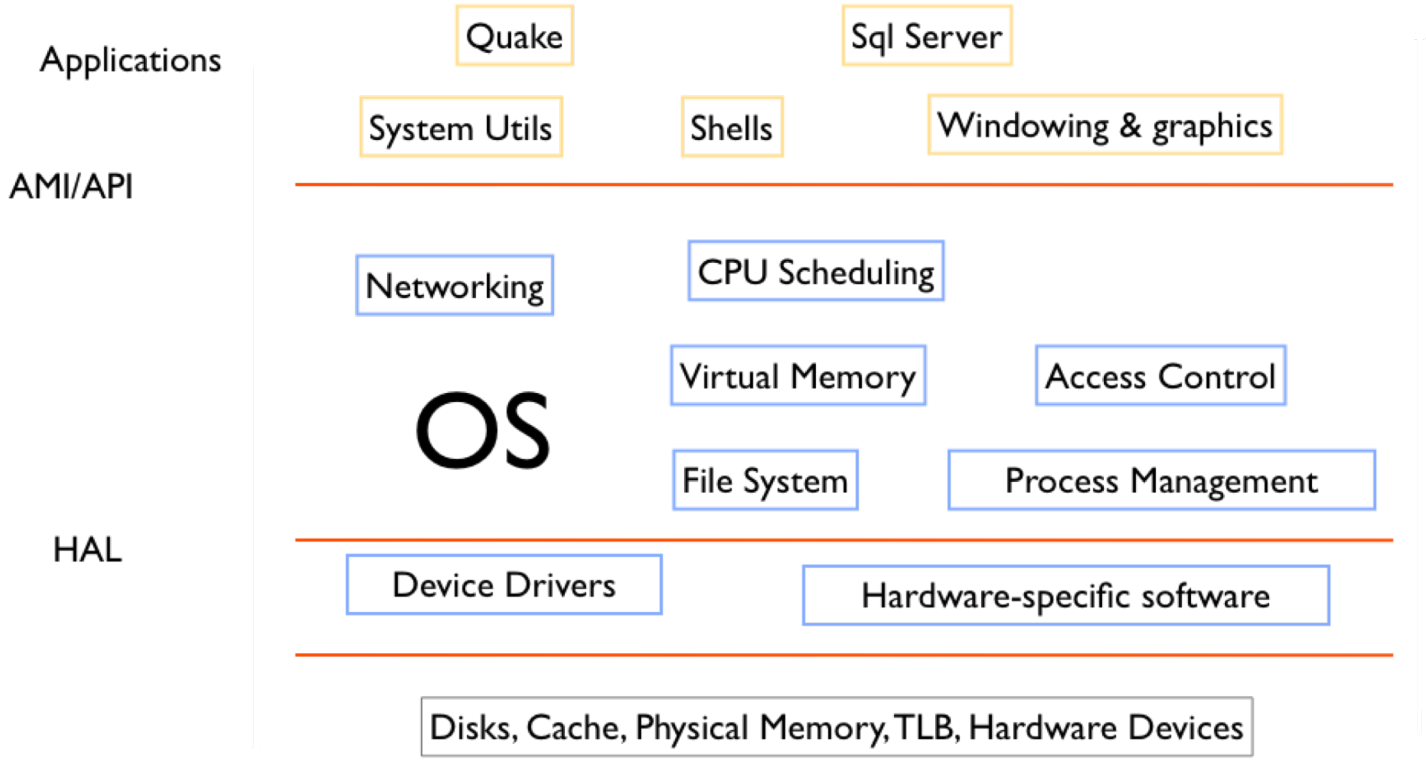
Security

- OS cannot be compromised by a malicious attacker
  - Includes privacy: data on the computer only accessible to authorized users
- Security policy
  - Defines what is permitted
- Enforcement mechanism
  - Ensures only permitted actions are allowed
- Strong fault isolation helps but is not enough
  - Security mechanisms should not prevent legitimate sharing!

Portability

- OS does not change as hardware changes
- OSs can live a really long time
  - Must support applications not yet written
- Three interfaces
  - Abstract Machine Interface (AMI)
    - Between OS and apps: API + memory access model + legally executable instructions
  - Application Programming Interface (API)
    - Function calls provided to apps
  - Hardware Abstraction Layer (HAL)
    - Abstracts hardware internally to the OS

Logical OS Structure



Performance

- Efficiency/Overhead
  - How much is lost by not running on bare hardware?
- Fairness
  - How are resources divided?
- Response time
  - How long does a task take to deliver a response to the user?
- Throughput
  - How many tasks complete per unit of time?
- Predictability
  - Are performance metrics consistent over time?