What is an Operating System?

January 20th, 2016
Prof. Anthony D. Joseph
http://cs162.eecs.Berkeley.edu

Greatest Artifact of Human Civilization...

Operating Systems at the heart of it all...

- Make the incredible advance in the underlying hardware available to a rapid evolving body of applications.
  - Processing, Communications, Storage, Interaction

- The key building blocks
  - Scheduling
  - Concurrency
  - Address spaces
  - Protection, Isolation, Security
  - Networking, distributed systems
  - Persistent storage, transactions, consistency, resilience
  - Interfaces to all devices
Example: What’s in a Search Query?

- Complex interaction of multiple components in multiple administrative domains
  - Systems, services, protocols, ...

Why take CS162?

- Some of you will actually design and build operating systems or components of them.
  - Perhaps more now than ever
- Many of you will create systems that utilize the core concepts in operating systems.
  - Whether you build software or hardware
  - The concepts and design patterns appear at many levels
- All of you will build applications, etc. that utilize operating systems
  - The better you understand their design and implementation, the better use you’ll make of them.

Goals for Today

- What is an Operating System?
  - And – what is it not?
- Examples of Operating Systems Design
- What Makes Operating Systems So Exciting?
- Oh, and “How does this class operate?”

Interactive is important!
Ask Questions!

What is an operating system?

- Special layer of software that provides application software access to hardware resources
  - Convenient abstraction of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication amongst logical entities
Operator …

Switchboard Operator

Computer Operators

OS Basics: “Virtual Machine” Boundary

Interfaces Provide Essential Boundaries

• Why do interfaces look the way that they do?
  – History, Functionality, Stupidity, Bugs, Management
  – CS152 ⇒ Machine interface
  – CS160 ⇒ Human interface
  – CS169 ⇒ Software engineering/management

• Should responsibilities be pushed across boundaries?
  – RISC architectures, Graphical Pipeline Architectures

OS Basics: Program => Process
What makes Operating Systems Exciting and Challenging?

Technology Trends: Moore’s Law

- Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.
- Microprocessors have become smaller, denser, and more powerful.

People-to-Computer Ratio Over Time

- Today: Multiple CPUs/person!
  - Approaching 100s?

New Challenge: Slowdown in Joy’s law of Performance

- VAX: 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present

⇒ Sea change in chip design: multiple “cores” or processors per chip
ManyCore Chips: The future is here
• Intel 80-core multicore chip (Feb 2007)
  – 80 simple cores
  – Two FP-engines / core
  – Mesh-like network
  – 100 million transistors
  – 65nm feature size
• Intel Single-Chip Cloud Computer (August 2010)
  – 24 “tiles” with two cores/tile
  – 24-router mesh network
  – 4 DDR3 memory controllers
  – Hardware support for message-passing

• “ManyCore” refers to many processors/chip
  – 64? 128? Hard to say exact boundary
• How to program these?
  – Use 2 CPUs for video/audio
  – Use 1 for word processor, 1 for browser
  – 76 for virus checking???
• Parallelism must be exploited at all levels

Another Challenge: Power Density
• Moore’s Law Extrapolation
  – Potential power density reaching amazing levels!
• Flip side: Battery life very important
  – Moore’s law can yield more functionality at equivalent (or less) total energy consumption

Storage Capacity
• Retail hard disk capacity in GB
Internet Scale: One Billion Hosts!

Internet Domain Survey Host Count

Source: Internet Systems Consortium (www.isc.org)
https://www.isc.org/solutions/survey

Internet Scale: Almost 3.4 Billion Users!

WORLD INTERNET USAGE AND POPULATION STATISTICS
NOVEMBER 30, 2015 - Update

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td>1,188,355,663</td>
<td>16.0 %</td>
<td>330,965,359</td>
<td>28.6 %</td>
<td>7,231.5 %</td>
<td>0.9 %</td>
</tr>
<tr>
<td>Asia</td>
<td>4,032,405,028</td>
<td>55.5 %</td>
<td>1,622,044,293</td>
<td>40.2 %</td>
<td>1,319.1 %</td>
<td>40.2 %</td>
</tr>
<tr>
<td>Europe</td>
<td>621,555,004</td>
<td>11.3 %</td>
<td>604,147,260</td>
<td>73.6 %</td>
<td>474.0 %</td>
<td>18.0 %</td>
</tr>
<tr>
<td>Middle East</td>
<td>236,137,236</td>
<td>3.3 %</td>
<td>125,172,132</td>
<td>52.2 %</td>
<td>3,849.8 %</td>
<td>0.7 %</td>
</tr>
<tr>
<td>North America</td>
<td>557,176,254</td>
<td>8.9 %</td>
<td>313,867,363</td>
<td>97.9 %</td>
<td>190.4 %</td>
<td>3.3 %</td>
</tr>
<tr>
<td>Latin America / Caribbean</td>
<td>517,049,710</td>
<td>8.5 %</td>
<td>344,824,159</td>
<td>65.0 %</td>
<td>1,906.4 %</td>
<td>10.2 %</td>
</tr>
<tr>
<td>Oceania / Australia</td>
<td>37,168,663</td>
<td>0.6 %</td>
<td>27,200,530</td>
<td>73.2 %</td>
<td>256.0 %</td>
<td>0.9 %</td>
</tr>
<tr>
<td>WORLD TOTAL</td>
<td>7,289,002,241</td>
<td>100.0 %</td>
<td>3,386,261,156</td>
<td>46.4 %</td>
<td>832.0 %</td>
<td>100.0 %</td>
</tr>
</tbody>
</table>

NOTES: (1) Internet Usage and World Population Statistics updated as of November 30, 2015. (2) Click on each world region name for detailed regional usage information. (3) Demographic (Population) numbers are based on data from the US Census Bureau, Eurostat and from local census agencies. (4) Internet usage information comes from data published by Nielsen Online, by the International Telecommunications Union, by GfK, by local ICT Regulators and other reliable sources. (5) For definitions, disclaimers, navigation help and methodology, please refer to the Site Surfing Guide. (6) Information in this site may be cited, providing the due credit and placing a link to www.internetworldstats.com. Copyright © 2001 - 2016, Minawati Marketing Group. All rights reserved worldwide.

(source: http://www.internetworldstats.com/stats.htm)

Not Only PCs connected to the Internet

• Smartphone shipments exceed PC shipments!

• 2011 shipments
  - 487M smartphones (73.7 million in Q3 2015)
  - 414M PC clients
    » 210M notebooks
    » 112M desktops
    » 63M tablets
  - 25M smart TVs (97.4 million in 2014)

• 4 billion phones in the world → smartphone over next decade

Societal Scale Information Systems
(Or the “Internet of Things”?)

• The world is a large distributed system
  – Microprocessors in everything
  – Vast infrastructure behind them

MEMS for Sensor Nets

Scalable, Reliable, Secure Services

Databases
Information Collection
Remote Storage
Online Games
Commerce
…
Who Am I?

- Anthony D. Joseph
  - 465 Soda Hall (AMP Lab)
  - Web: http://www.cs.berkeley.edu/~adj/
  - Office hours: Mondays and Tuesdays 10-11 in 465F Soda

- Taught BerkeleyX MOOC in July 2015
  - Big Data and Apache Spark
  - 76,000 students with >11% completion rate

- Research areas:
  - Cancer Genomics/Precision Medicine (ADAM/Apache Spark), Secure Machine Learning (SecML), DETER security testbed
  - Previous: Cloud computing (Apache Mesos), Peer-to-Peer networking (Tapestry), Mobile computing, Wireless/Cellular networking

Infrastructure, Textbook & Readings

- Infrastructure
  - Website: http://cs162.eecs.berkeley.edu
  - Piazza: https://piazza.com/berkeley/spring2015/cs162
  - Webcast: Cal Central - https://calcentral.berkeley.edu/academics/teaching-semester/spring-2016/class/compsci-162

  - Copies in Bechtel

- Online supplements
  - See course website
  - Includes Appendices, sample problems, etc.
  - Networking, Databases, Software Eng, Security
  - Some Research Papers!

Syllabus

- OS Concepts: How to Navigate as a Systems Programmer!
  - Process, I/O, Networks and Virtual Machines

- Concurrency
  - Threads, scheduling, locks, deadlock, scalability, fairness

- Address Space
  - Virtual memory, address translation, protection, sharing

- File Systems
  - I/O devices, file objects, storage, naming, caching, performance, paging, transactions, databases

- Distributed Systems
  - Protocols, N-Tiers, RPC, NFS, DHTs, Consistency, Scalability, multicast

- Reliability & Security
  - Fault tolerance, protection, security

- Cloud Infrastructure
Learning by Doing

• Individual Homeworks: Learn Systems Programming
  – 0. Tools, Autograding, recall C, executable
  – 1. Simple Shell
  – 2. Web server
  – 3. Memory allocation
  – 4. Two Phase Commit and Key-Value Store

• Three Group Projects (Pintos in C)
  – 1. Threads & Scheduling
  – 2. User-programs
  – 3. File Systems

Getting started

• Start homework 0 immediately
  – Gets cs162-xx@cory.eecs.berkeley.edu (and other inst m/c)
  – Github account
  – Registration survey
  – Vagrant virtualbox – VM environment for the course
    » Consistent, managed environment on your machine
  – Get familiar with all the cs162 tools
  – Submit to autograder via git

• Waitlist ???
  – This is an Early Drop Deadline course (January 29th)
  – If you are not serious about taking, please drop early

Group Project Simulates Industrial Environment

• Project teams have 4 members (try really hard to find 4 members – 3 members requires serious justification)
  – Must work in groups in “the real world”
  – Same section much preferred

• Communicate with colleagues (team members)
  – Communication problems are natural
  – What have you done?
  – What answers you need from others?
  – You must document your work!!!

• Communicate with supervisor (TAs)
  – What is the team’s plan?
  – What is each member’s responsibility?
  – Short progress reports are required
  – Design Documents: High-level description for a manager!

Grading

• 40% two midterms and comprehensive final exam
  – Midterms: 3/9 and 4/20, 6-7:30P
  – NEW: Exam Group 3, May 9 3-6P

• 40% projects

• 15% homework

• 5% participation

• Projects
  – Initial design, Design review, Design document, Code, Final design
  – Submission via git push triggers autograder
Personal Integrity

- UCB Academic Honor Code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."


CS 162 Collaboration Policy

- Explaining a concept to someone in another group
- Discussing algorithms/testing strategies with other groups
- Helping debug someone else's code (in another group)
- Searching online for generic algorithms (e.g., hash table)

- Sharing code or test cases with another group
- Copying OR reading another group's code or test cases
- Copying OR reading online code or test cases from prior years

We compare all project submissions against prior year submissions and online solutions and will take actions (described on the course overview page) against offenders

Typical Lecture Format

- 1-Minute Review
- 20-Minute Lecture
- 5-Minute Administrative Matters
- 25-Minute Lecture
- 5-Minute Break (water, stretch)
- 25-Minute Lecture
- Instructor will come to class early & stay after to answer questions

Lecture Goal

Interactive!!!
What is an Operating System?

- **Referee**
  - Manage sharing of resources, Protection, Isolation
    » Resource allocation, isolation, communication

- **Illusionist**
  - Provide clean, easy to use abstractions of physical resources
    » Infinite memory, dedicated machine
    » Higher level objects: files, users, messages
    » Masking limitations, virtualization

- **Glue**
  - Common services
    » Storage, Window system, Networking
    » Sharing, Authorization
    » Look and feel

Challenge: Complexity

- Applications consisting of...
  - ... a variety of software modules that ...
  - ... run on a variety of devices (machines) that
    » ... implement different hardware architectures
    » ... run competing applications
    » ... fail in unexpected ways
    » ... can be under a variety of attacks

- Not feasible to test software for all possible environments and combinations of components and devices
  - The question is not whether there are bugs but how serious are the bugs!

Increasing Software Complexity

How do we tame complexity?

- Every piece of computer hardware different
  - Different CPU
    » Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory, disk, …
  - Different types of devices
    » Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    » Cable, DSL, Wireless, Firewalls,…

- Questions:
  - Does the programmer need to write a single program that performs many independent activities?
  - Does every program have to be altered for every piece of hardware?
  - Does a faulty program crash everything?
  - Does every program have access to all hardware?
OS Tool: Virtual Machine Abstraction

Application

Operating System

Hardware

• Software Engineering Problem:
  – Turn hardware/software quirks ⇒ what programmers want/need
  – Optimize for convenience, utilization, security, reliability, etc…
• For Any OS area (e.g. file systems, virtual memory, networking, scheduling):
  – What’s the hardware interface? (physical reality)
  – What’s the application interface? (nicer abstraction)

Virtual Machines

• Software emulation of an abstract machine
  – Give programs illusion they own the machine
  – Make it look like hardware has features you want
• Two types of "Virtual Machine”s
  – Process VM: supports the execution of a single program; this functionality typically provided by OS
  – System VM: supports the execution of an entire OS and its applications (e.g., VMWare Fusion, Virtual box, Parallels Desktop, Xen)

Process VMs

• Programming simplicity
  – Each process thinks it has all memory/CPU time
  – Each process thinks it owns all devices
  – Different devices appear to have same high level interface
  – Device interfaces more powerful than raw hardware
    » Bitmapped display ⇒ windowing system
    » Ethernet card ⇒ reliable, ordered, networking (TCP/IP)
• Fault Isolation
  – Processes unable to directly impact other processes
  – Bugs cannot crash whole machine
• Protection and Portability
  – Java interface safe and stable across many platforms

System Virtual Machines: Layers of OSs

• Useful for OS development
  – When OS crashes, restricted to one VM
  – Can aid testing programs on other OSs
What is an Operating System, Really?

• Most Likely:
  – Memory Management
  – I/O Management
  – CPU Scheduling
  – Communications? (Does Email belong in OS?)
  – Multitasking/multiprogramming?

• What about?
  – File System?
  – Multimedia Support?
  – User Interface?
  – Internet Browser? 😐

• Is this only interesting to Academics??

Example: Protecting Processes from Each Other

• Problem: Run multiple applications in such a way that they are protected from one another

• Goal:
  – Keep User Programs from Crashing OS
  – Keep User Programs from Crashing each other
  – [Keep Parts of OS from crashing other parts?]

• (Some of the required) Mechanisms:
  – Address Translation
  – Dual Mode Operation

• Simple Policy:
  – Programs are not allowed to read/write memory of other Programs or of Operating System

Operating System Definition (Cont.)

• No universally accepted definition
• "Everything a vendor ships when you order an operating system" is good approximation
  – But varies wildly
• “The one program running at all times on the computer” is the kernel.
  – Everything else is either a system program (ships with the operating system) or an application program

Address Translation

• Address Space
  – A group of memory addresses usable by something
  – Each program (process) and kernel has potentially different address spaces.

• Address Translation:
  – Translate from Virtual Addresses (emitted by CPU) into Physical Addresses (of memory)
  – Mapping often performed in Hardware by Memory Management Unit (MMU)
Example of Address Translation

<table>
<thead>
<tr>
<th>Translation Map 1</th>
<th>Translation Map 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Address Space</td>
<td></td>
</tr>
<tr>
<td>Prog 1 Virtual Address Space 1</td>
<td></td>
</tr>
<tr>
<td>Prog 2 Virtual Address Space 2</td>
<td></td>
</tr>
<tr>
<td>Code 1</td>
<td></td>
</tr>
<tr>
<td>Data 1</td>
<td></td>
</tr>
<tr>
<td>Stack 1</td>
<td></td>
</tr>
<tr>
<td>Code 2</td>
<td></td>
</tr>
<tr>
<td>Data 2</td>
<td></td>
</tr>
<tr>
<td>Stack 2</td>
<td></td>
</tr>
<tr>
<td>OS code</td>
<td></td>
</tr>
<tr>
<td>OS data</td>
<td></td>
</tr>
<tr>
<td>OS heap &amp; Stacks</td>
<td></td>
</tr>
</tbody>
</table>

Address Translation Details

- For now, assume translation happens with table (called a Page Table):
  - Virtual Address
    - V page no
    - offset
  - Page Table
    - index
    - into page table
    - table located in physical memory
    - P page no
    - offset
    - Physical Address

- Translation helps protection:
  - Control translations, control access
  - Should Users be able to change Page Table???

Dual Mode Operation

- Hardware provides at least two modes:
  - “Kernel” mode (or “supervisor” or “protected”)
  - “User” mode: Normal programs executed

- Some instructions/ops prohibited in user mode:
  - Example: cannot modify page tables in user mode
    - Attempt to modify ⇒ Exception generated

- Transitions from user mode to kernel mode:
  - System Calls, Interrupts, Other exceptions

UNIX System Structure

<table>
<thead>
<tr>
<th>User Mode</th>
<th>Kernel Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>signals terminal handling</td>
</tr>
<tr>
<td>Standard Libs</td>
<td>character I/O system terminal drivers</td>
</tr>
<tr>
<td></td>
<td>file system swapping block I/O system disk and tape drivers</td>
</tr>
<tr>
<td>system-call interface to the kernel</td>
<td>CPU scheduling demand paging virtual memory</td>
</tr>
<tr>
<td>Kernel</td>
<td>Hardware</td>
</tr>
<tr>
<td></td>
<td>terminal controllers terminals</td>
</tr>
<tr>
<td></td>
<td>device controllers disks and tapes</td>
</tr>
<tr>
<td></td>
<td>memory controllers physical memory</td>
</tr>
</tbody>
</table>
“In conclusion…”

- Operating systems provide a virtual machine abstraction to handle diverse hardware
- Operating systems coordinate resources and protect users from each other
- Operating systems simplify application development by providing standard services
- Operating systems can provide an array of fault containment, fault tolerance, and fault recovery

- CS162 combines things from many other areas of computer science –
  - Languages, data structures, hardware, and algorithms