CS162
Operating Systems and Systems Programming
Lecture 1

What is an Operating System?

August 26th, 2020
Prof. Ion Stoica
http://cs162.eecs.Berkeley.edu
Instructor: Ion Stoica

- Web: [http://www.cs.berkeley.edu/~istoica/](http://www.cs.berkeley.edu/~istoica/)
  - 465 Soda Hall (RISE Lab)

- Research areas:
  - ML systems (Ray, Clipper, …)
  - Big Data systems (Apache Spark, Succinct, …)
  - Previous: Cluster computing (Apache Mesos, Alluxio), Peer-to-Peer networking (Chord), Networking QoS

- Industry experience: co-founded
  - Conviva (2006 - ): Video distribution and analytics
  - Databricks (2013 - ): Company behind Apache Spark
CS162 TAs: Sections TBA

Edward Zeng (Head TA)  Sean Kim (Head TA)  Alina Dan  Guanhua Wang  Lianmin Zheng

Marcus Plutowski  Nancy Huang  Paras Jain  Zhuohan Li  Zongheng Wang
Enrollment

• This is an Early Drop Deadline course (September 3rd)
  – If you are not serious about taking, please drop early
  – Department will continue to admit students as other students drop
  – Really hard to drop afterwards!
    » Don’t forget to keep up with work if you are still on the waitlist!

• On the waitlist/Concurrent enrollment?
  – All decisions concerning appeals for enrolling in the course are made by CS departmental staff; the course staff have no say in the matter. So please do not e-mail us about waitlist or concurrent enrollment.
  – If people drop, others will be moved off the waitlist
Still mostly remote

- All lectures will be remote
- All exams will be remote
- 13 discussion sections will be remote; 4 will be in person
  - 4 discussion sections on Thursdays
  - 13 discussion sections on Fridays (4 in person)

- Please make sure you have a working CAMERA
  - Zoom proctoring of exams
  - Required sections, design reviews, interactions with your group (screen shots!)
Goals for Today

• What is an Operating System?
  – And – what is it not?
• What makes Operating Systems so exciting?
• “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
What Does an OS do?

• Provide abstractions to apps
  – File systems
  – Processes, threads
  – VM, containers
  – Naming system
  – …

• Manage resources:
  – Memory, CPU, storage, …

• Achieves the above by implementing specific algos and techniques:
  – Scheduling
  – Concurrency
  – Transactions
  – Security
  – …
Hardware/Software Interface

What you learned in CS 61C – Machine Structures (and C)

The OS *abstracts* these hardware details from the application.
What is an Operating System?

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

Process: Execution environment with restricted rights provided by OS

Compiled Program

System Libs

Hardware

ISA

Processor

PgTbl & TLB

Memory

OS Mem

Storage

I/O Ctrlr

Threads

Address Spaces

Files

Sockets

Operating System

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

Compiled Program

System Libs

Process: Execution environment with restricted rights provided by OS

Compiled Program

System Libs
Compiled Program’s View of the World

Operating System

- Application’s “machine” is the process abstraction provided by the OS
- Each running program runs in its own process
- Processes provide nicer interfaces than raw hardware
System Programmer's View of the World

Process: Execution environment with restricted rights provided by OS

- Application's "machine" is the process abstraction provided by the OS
- Each running program runs in its own process
- Processes provide nicer interfaces than raw hardware

Program

```
#include <stdlib.h>
int main(void) {
    printf("Hello!\n");
}
```
What’s in a Process?

A process consists of:

• Address Space
• One or more threads of control executing in that address space
• Additional system state associated with it
  – Open files
  – Open sockets (network connections)
  – …
For Example...
Operating System’s View of the World

- Compiled Program 1
  - System Libs
- Process 1
  - Threads
  - Address Spaces
  - Files
  - Sockets
- Compiled Program 2
  - System Libs
- Process 2
  - Threads
  - Address Spaces
  - Files
  - Sockets

- Operating System
- Hardware
  - Processor
  - Memory
    - OS Mem
  - Storage
  - Networks
  - I/O Ctrlr

- Compiler
- ISA

8/26/21 CS162 © UCB Fall 2021 Lec 1.17
• OS translates from hardware interface to application interface
• OS provides each running program with its own process
What is an Operating System?

- **Referee**
  - Manage protection, isolation, and sharing of resources
    - Resource allocation and communication

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

Compiler

Hardware

Processor

Memory

Operating System

Compiled Program 1

System Libs

Process 1

Threads

Address Spaces

Files

Sockets

Compiled Program 2

System Libs

Process 2

Threads

Address Spaces

Files

Sockets

Compiled Program 1

System Libs

Process 1

Threads

Address Spaces

Files

Sockets

Compiled Program 2

System Libs

Process 2

Threads

Address Spaces

Files

Sockets

Compiled Program 1

System Libs

Process 1

Threads

Address Spaces

Files

Sockets

Compiled Program 2

System Libs

Process 2

Threads

Address Spaces

Files

Sockets
OS Basics: Switching Processes

Compiled Program 1

System Libs

Process 1

Thread
Address Spaces
Files
Sockets

Compiled Program 2

System Libs

Process 2

Thread
Address Spaces
Files
Sockets

Operating System

Compiler

Processor

Memory

OS Mem

I/O Ctrlr

Storage

Networks

Hardware
OS Basics: Switching Processes

Compiled Program 1

System Libs

Process 1

Threads Address Spaces Files Sockets

Compiled Program 2

System Libs

Process 2

Threads Address Spaces Files Sockets

Operating System

Compiler

Hardware

Processor

Memory

Storage

Networks

I/O Ctrlr

ISA
OS Basics: Switching Processes

- Compiled Program 1
  - System Libs
- Process 1
  - Threads
  - Address Spaces
  - Files
  - Sockets
- Process 2
  - Threads
  - Address Spaces
  - Files
  - Sockets
- Compiled Program 2
  - System Libs

- Operating System
- Hardware
  - Processor
  - Memory
  - Storage
- Compiler
- ISA
- Networks
- I/O Ctrlr
OS Basics: Switching Processes

Compiled Program 1
- System Libs

Process 1
- Threads
- Address Spaces
- Files
- Sockets

Compiled Program 2
- System Libs

Process 2
- Threads
- Address Spaces
- Files
- Sockets

Operating System

Compiled Program 1
- System Libs

Process 1
- Threads
- Address Spaces
- Files
- Sockets

Compiled Program 2
- System Libs

Process 2
- Threads
- Address Spaces
- Files
- Sockets

Operating System
OS Basics: Protection

- Process 1
  - Compiled Program 1
  - System Libs
  - Threads
  - Address Spaces
  - Files
  - Sockets

- Process 2
  - Compiled Program 2
  - System Libs
  - Threads
  - Address Spaces
  - Files
  - Sockets

- Hardware
  - Processor
  - Memory
  - Storage

- Compiler

- Operating System

- Networks

- ISA
OS Basics: Protection

Compiled Program 1

System Libs

Process 1

Threads
Address Spaces
Files
Sockets

Operating System

Compiled Program 2

System Libs

Process 2

Threads
Address Spaces
Files
Sockets

Segmentation fault (core dumped)
OS Basics: Protection

- OS *isolates* processes from each other
- OS *isolates* itself from other processes
- … even though they are actually running on the same hardware!
What is an Operating System?

• Referee
  – Manage protection, isolation, and sharing of resources
    » Resource allocation and 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
OS Basics: I/O

- OS provides common services in the form of I/O
OS Basics: Look and Feel

Process: Execution environment with restricted rights provided by OS

- Compiled Program
- System Libs
- Operating System
- Hardware
- ISA
- Processor
- Threads
- Address Spaces
- Files
- Sockets
- Windows
- Storage
- Networks
- Displays
- I/O Ctrlr
- Compiler
- PgTbl & TLB
- Memory
- OS Mem
- System Libs

8/26/21
CS162 © UCB Fall 2021
Lec 1.30
OS Basics: Background Management

Process: Execution environment with restricted rights provided by OS

- Threads
- Address Spaces
- Files
- Sockets
- Windows

Operating System

Compiled Program
- System Libs

Hardware
- Compiler
- ISA
- Processor
- Memory
- Storage
- PgTbl & TLB
- OS Mem
- I/O Ctrlr

Networks
- Network Manager

Displays
- Power Manager

Battery
What is an Operating System?

• Referee
  – Manage protection, isolation, and sharing of resources
    » Resource allocation and 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
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.
What makes Operating Systems Exciting and Challenging?
Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 2 years

- Microprocessors have become smaller, denser, and more powerful

Corollary: Performance double roughly every 1.5 years (18 months)

- Faster growth because transistors are closer to each other so electrical signals travel faster
The end of the Moore’s Law

From 2x every 18 months to 1.05x every 18 months!
AI Compute demands: 2012 - 2019

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

(https://openai.com/blog/ai-and-compute/)
AI Compute demands: 2012 – 2020

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

35x every 18 months

(https://openai.com/blog/ai-and-compute/)
Growing gap between demand and supply

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

35x every 18 months

Moore’s Law (2x every 18 months)

CPU

(https://openai.com/blog/ai-and-compute/)

[8/26/21]
Specialized processors to the rescue?

TPU v3

Nvidia A100

AWS's Inferentia

AMD Radeon

Cerebras

Intel Alchemist
Specialized hardware note enough

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

- Moore's Law (2x every 18 months)
- TPU* (35x every 18 months)
- GPU*
- CPU

* assume 0.33 utilization

(https://openai.com/blog/ai-and-compute/)
Memory demands growing as fast

China's gigantic multi-modal AI is no one-trick pony

Sporting 1.75 trillion parameters, Wu Dao 2.0 is roughly ten times the size of Open AI's GPT-3.

A. Tarantola
06.02.21
Memory demands growing as fast

- GPT-2 (8.3B)
- BERT
- GPT-1 (1.5B)
- Turing Proj. (17B)
- GPT-3 (175B)
- Wu Dao 2.0 (1.75T)

Number of parameters

- 40x every 18 months
- 340x every 18 months

January 2016 to January 2021
Huge Gap between memory demands and single-chip memory

Number of parameters vs. time:
- ResNet-50
- Transformer
- GPT-1
- BERT
- GPT-3 (175B)
- Turing Proj. (17B)
- Wu Dao 2.0 (1.75T)

- 1.7x every 18 months
- 34x every 18 months

1,000+ GPUs just to store parameters!
Storage Capacity is Still Growing!
Storage Capacity

Largest SSD 3.5-inch drive: 100 TB @ $40K ($400/TB)

Largest HDD 3.5-inch drive: 18 TB @ $600 ($33/TB)

Cheaper, but
• Slower (10x-100x)
• Consumes more power
• Less reliable

Samsung SSD 2.5-inch drive: 4 TB @ $380K ($95/TB)
SSD/Flash will Dominate

Figure 9 - Exabyte Storage Shipments Split by SSD, Other Flash, HDDs, and Tape

Figure 4 - SSD/HDD Pricing Ratio 2013 - 2030
Network Capacity

People-to-Computer Ratio Over Time

- Today: multiple CPUs/person!
  - Approaching 100s?
And Range of Timescales

Jeff Dean: “Numbers Everyone Should Know”

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache reference</td>
<td>0.5</td>
</tr>
<tr>
<td>Branch mispredict</td>
<td>5</td>
</tr>
<tr>
<td>L2 cache reference</td>
<td>7</td>
</tr>
<tr>
<td>Mutex lock/unlock</td>
<td>25</td>
</tr>
<tr>
<td>Main memory reference</td>
<td>100</td>
</tr>
<tr>
<td>Compress 1K bytes with Zippy</td>
<td>3,000</td>
</tr>
<tr>
<td>Send 2K bytes over 1 Gbps network</td>
<td>20,000</td>
</tr>
<tr>
<td>Read 1 MB sequentially from memory</td>
<td>250,000</td>
</tr>
<tr>
<td>Round trip within same datacenter</td>
<td>500,000</td>
</tr>
<tr>
<td>Disk seek</td>
<td>10,000,000</td>
</tr>
<tr>
<td>Read 1 MB sequentially from disk</td>
<td>20,000,000</td>
</tr>
<tr>
<td>Send packet CA-&gt;Netherlands-&gt;CA</td>
<td>150,000,000</td>
</tr>
</tbody>
</table>
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!
Not only specialized processors…

- Multi-core processors
  - Intel Xeon Phi: 64 cores
  - AMD Epyc: 64 cores; planned 128 cores (?)
A Modern Processor: Intel Sandy Bridge

- Package: LGA 1155
  - 1155 pins
  - 95W design envelope
- Cache:
  - L1: 32K Inst, 32K Data (3 clock access)
  - L2: 256K (8 clock access)
  - Shared L3: 3MB – 20MB
- Transistor count:
  - 504 Million (2 cores, 3MB L3)
  - 2.27 Billion (8 cores, 20MB L3)
- Note that ring bus is on high metal layers – above the Shared L3 Cache
HW Functionality comes with great complexity!

Intel Sandy Bridge I/O Configuration

- Proc
- Caches
- Memory
- Busses
- Controllers
- I/O Devices: Disks, Displays, Keyboards, Networks
Not only specialized processors...

- Multi-core processors
  - Intel Xeon Phi: 64 cores
  - AMD Epyc: 64 cores; planned 128 cores (?)

- System On a Chip (SOC): accelerating trend
Increasing Software Complexity

From MIT's 6.033 course
Everything going distributed

• Need to scale
  – Machine learning workloads
  – Big Data analytics
  – Scientific computing
  – …

• Everything is connected!
Greatest Artifact of Human Civilization…
Running Systems at Internet Scale

Worldwide Internet Users

Worldwide Internet Users (1965-2025)

- 1969
- 1974
- 1990

- ARPANet
- RFC 675 TCP/IP
- HTTP
- WWW
Not Only PCs connected to the Internet

• Smartphone shipments exceed PC shipments!

• 2011 shipments:
  – 487M smartphones
  – 414M PC clients
    » 210M notebooks
    » 112M desktops
    » 63M tablets
  – 25M smart TVs

• 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
Example: What's in a Search Query?

- Complex interaction of multiple components in multiple administrative domains
  - Systems, services, protocols, …
How do we tame complexity?

• Every piece of computer hardware different
  – Different CPU
    » Pentium, ARM, ColdFire, ARM
  – Different specialized hardware
    » Nvidia GPUs, TPUs, AWS Inferentia, …
  – Different amounts of memory, disk, …
  – Different types of devices
    » Mice, Keyboards, Sensors, Cameras, Fingerprint readers, Face recognition
  – Different networking environment
    » Fiber, 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

Virtual Machine Interface

Operating System

Physical Machine Interface

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
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 (2 weeks) - preliminary
  – 0. Tools & Environment, Autograding, recall C, executable
  – 1. Lists in C
  – 2. BYOS – build your own shell
  – 3. Scheduling
  – 4. HTTP Server
  – 5. Memory allocation (Malloc)
  – 6. Memory management

• Three (and ½) Group Projects
  – 0. Getting Started (Individual, before you have a group)
  – 1. User-programs (exec & syscall)
  – 2. Threads & Scheduling
  – 3. File Systems
Preparing Yourself for this Class

• The projects will require you to be very comfortable with programming and debugging C
  – Pointers (including function pointers, void*)
  – Memory Management (malloc, free, stack vs heap)
  – Debugging with GDB
• You will be working on a larger, more sophisticated code base than anything you've likely seen in 61C!
• "Resources" page on course website
  – Ebooks on "git" and "C"
• C programming reference:
  – https://cs162.eecs.berkeley.edu/ladder/
• First two sections are also dedicated to programming and debugging review:
  – Attend ANY sections in first two weeks
Group Projects

• Project teams have 4 members!
  – never 5, 3 requires serious justification
  – Must work in groups in “the real world”
  – Same section (at least same TA)

• Communication and cooperation will be essential
  – Regular meetings WITH CAMERA TURNED ON!
    » Extra credit for screen shots of all of you together in zoom with camera enabled
  – Design Documents
  – Slack/Messenger/whatever doesn’t replace face-to-face!

• Everyone should do work and have clear responsibilities
  – You will evaluate your teammates at the end of each project
  – Dividing up by Task is the worst approach. Work as a team.

• 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

- 36% three midterms (12% each)
- These will be ZOOM-Proctored. Camera REQUIRED.
- 36% projects
- 18% homework
- 10% participation (Sections, Lecture, …)
- No final exam
- Projects
  - Initial design document, Design review, Code, Final design
  - Submission via `git push` triggers autograder
Infrastructure, Textbook & Readings

• Infrastructure
  – Website:  http://cs162.eecs.berkeley.edu
  – Piazza:  https://piazza.com/berkeley/fall2021/cs162
  – Lecture Recordings: Tentatively as links off main class page (next day)

  – Suggested readings posted along with lectures
  – Try to keep up with material in book as well as lectures

• Supplementary Material
  – Operating Systems: Three Easy Pieces, by Remzi and Andrea Arpaci-Dusseau, available for free online
  – Linux Kernel Development, 3rd edition, by Robert Love

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

Interactive!!!
“In conclusion…”

• Operating systems provide a virtual machine abstraction to handle diverse hardware
  – Operating systems simplify application development by providing standard services

• Operating systems coordinate resources and protect users from each other
  – 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