What is an Operating System?

Professor Natacha Crooks
https://cs162.org/

Slides based on prior slide decks from David Culler, Ion Stoica, John Kubitawociz, Alison Norman and Lorenzo Alvisi
Goals for Today

• Why should you care?

• Why is it hard?

• What is an Operating System?

• Administratrivia & Course Policy
Goal 1: Why should you care?
The OS is everywhere

Every device, from your smartwatch, your smart light bulb, to your mobile phone and laptop runs an operating system

Every program you will ever write will run on an operating system

Its performance and execution behaviour will depend on the operating system
Goal 2: Why is designing an OS hard?
What do these have in common?
Across many devices

Have an operating system

Communicate over the Internet

Interface across huge diversity of devices
Bell’s Law

One new device class every 10 years
### Across many timescales

<table>
<thead>
<tr>
<th>Description</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache reference</td>
<td>0.5 ns</td>
</tr>
<tr>
<td>Branch mispredict</td>
<td>5 ns</td>
</tr>
<tr>
<td>L2 cache reference</td>
<td>7 ns</td>
</tr>
<tr>
<td>Mutex lock/unlock</td>
<td>25 ns</td>
</tr>
<tr>
<td>Main memory reference</td>
<td>100 ns</td>
</tr>
<tr>
<td>Compress 1K bytes with Zippy</td>
<td>3,000 ns</td>
</tr>
<tr>
<td>Send 2K bytes over 1 Gbps network</td>
<td>20,000 ns</td>
</tr>
<tr>
<td>Read 1 MB sequentially from memory</td>
<td>250,000 ns</td>
</tr>
<tr>
<td>Round trip within same datacenter</td>
<td>500,000 ns</td>
</tr>
<tr>
<td>Disk seek</td>
<td>10,000,000 ns</td>
</tr>
<tr>
<td>Read 1 MB sequentially from disk</td>
<td>20,000,000 ns</td>
</tr>
<tr>
<td>Send packet CA-&gt;Netherlands-&gt;CA</td>
<td>150,000,000 ns</td>
</tr>
</tbody>
</table>

**Jeff Dean’s Numbers Everyone Should Know**
With increased complexity
Why so much complexity?

- Hardware is becoming smarter!
- Better reliability and security
- Better performance (more efficient code, more parallel code)
- Better energy efficiency
Goal 3: What is an Operating System?
Operating System

Manages multiple tasks and users

A set of interconnected components with an expected behaviour observed at the interface with its environment
An operating system is the layer of software that interfaces between (diverse) hardware resources and the (many) applications running on the machine.
An operating system implements a virtual machine for the application whose interface is more convenient than the raw hardware interface (convenient = security, reliability, portability)
Three main hats

**Referee**
Manage protection, isolation, and sharing of resources

**Illusionist**
Provide clean, easy-to-use abstractions of physical resources

**Glue**
Provides a set of common services
OS as a referee

Allow multiple (untrusted) applications to run concurrently
OS as a referee

**Fault Isolation**
- Isolate programs from each other
- Isolate OS from other programs

**Resource Sharing**
- How to choose which task to run next?
- How to split physical resources?

**Communication**
- How can OS support communication to share results?

**Process**
- Dual Mode Execution

**Scheduling**
- Pipes/Sockets
What does this program do? (CS61C)

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>
#include "common.h"

int main(int argc, char *argv[]){
    char *str = argv[1];
    while (1) {
        printf("%s\n", str);
    }
    return 0;
}
```

crooks@laptop> gcc -o cpu cpu.c -Wall
crooks@laptop> ./cpu "A"
A
A
A
A
...
crooks@laptop> ./cpu A & ./cpu B & ./cpu C

A  B  C
A  B  C
A  B  C
A  B  C
...

crooks@laptop> ./cpu ; ./cpu B
Segmentation Fault
B
...
```
Refereeing is hard!

Mac V8 (1997)

OS cannot force program to give up control!

crooks@very-old-laptop>
./cpu A & ./cpu B & ./cpu C

A
A
A
A
A
A
...

File Edit View Special Help
About This Computer…
Apple System Profiler
Appletalk Audio Player
Automated Tasks
Calculator
Chooser
Connect To...
Control Panels
Find File
Jigsaw Puzzle
Key Caps
Note Pad
Recent Applications
Recent Documents
Recents
Scrapbook
Simple Sound
Sticker
Trash
Three main hats

**Referee**  
Manage protection, isolation, and sharing of resources

**Illusionist**  
Provide clean, easy-to-use abstractions of physical resources
## OS as Illusionist

Mask the restrictions inherent in computer hardware through virtualization

<table>
<thead>
<tr>
<th>All alone</th>
<th>All powerful</th>
<th>All expressive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide abstraction that application has exclusive use of resources</td>
<td>Provide abstraction that hardware resources are infinite</td>
<td>Provide abstraction of hardware capabilities that are not physically present</td>
</tr>
</tbody>
</table>
What does this program do? (CS61C)

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]){
    int *p = malloc(sizeof(int));
    printf("(%d) p: %p\n", getpid(), p);
    *p = 0;
    while (1) {
        *p = *p + 1;
        printf("(%d) p: %d\n", getpid(), *p);
    }
    return 0;
}
```

crooks@laptop> gcc -o memory memory.c -Wall

crooks@laptop> ./memory

(120) p: 0x200000

(120) p: 1

(120) p: 2

(120) p: 3

(120) p: 4

crooks@laptop> ./memory & ./memory

(120) p: 0x200000

(254) p: 0x200000

a) (120) p: 1
   (254) p: 2
   (120) p: 3
   (254) p: 4
   (120) p: 5
   (254) p: 6
   ...

b) (120) p: 1
   (254) p: 1
   (120) p: 2
   (254) p: 2
   (120) p: 3
   (254) p: 3
   ...
```
Three main hats

**Referee**
Manage protection, isolation, and sharing of resources

**Illusionist**
Provide clean, easy-to-use abstractions of physical resources

**Glue**
Provides a set of common services
OS as Glue

Provide set of common, standard services to applications to simplify and regularize their design

**Make sharing easier**
Simpler if all assume same basic primitives

**Minimize reuse**
Avoid re-implementing functionality from scratch.
Evolve components independently

File System, User Interface, Network, etc.
Web browsers?

Are web browsers part of the OS?

*United States of America v. Microsoft Corporation*
Putting it all together

Referee + illusionist + Glue

=> Easy to use virtual machine
Evaluation Criteria: Performance

OS must implement the abstraction efficiently, with low overhead, and equitably

**Overhead**: added resource cost of implementing an abstraction

**Fairness**: How “well” are resources distributed across applications

**Response time**: how long does it take for a task to complete

**Throughput**: Rate at which group of tasks can be completed

**Predictability**: Are performance metrics constant over time?
Evaluation Criteria: Reliability

System does what it is supposed to do

OS failures catastrophic!

Availability: mean time to failure + mean time to repair
Evaluation Criteria: Security

Minimize vulnerability to attack

**Integrity:** Computer’s operation cannot be compromised by a malicious attacker

**Privacy:** Data stored on computer accessible to authorized users

**Enforcement Policy**
How the OS ensures only permitted actions are allowed

**Security Policy**
What is permitted
A portable abstraction does not change as the hardware changes

Can’t rewrite application (or OS!) every time

Must plan for hardware that does not exist yet!
Three “Prongs” for the Class

Understanding OS principles

System Programming

Map Concepts to Real Code
Topic Breakdown

Virtualizing the CPU

Virtualizing Memory

Persistence

Distributed Systems

Process Abstraction and API

Threads and Concurrency

Scheduling

Virtual Memory

Paging

IO devices

File Systems

Challenges with distribution

Data Processing & Storage
Administrivia
Natacha Crooks

Assistant Professor in EECS
- 723 Soda Hall (Sky Lab), http://nacrooks.github.io

Member of the RDI center and Sky Lab

Research areas:
- Large Scale Distributed Systems And Databases
- Decentralized Systems

Outside Activities
- Consultant for Blockchain companies and Data Orchestration Companies.
CS162 TAs: Sections TBA

David Zhang (Head TA)

Rohan Kumar (Head TA)

Vasanth Kumar

Daniel Zhu

James Holcombe

Levy Deng

Alvin Xu

Wilson Nguyen
Enrollment

Class has 368 limit
  – No guaranteed expansion

This is an Early Drop Deadline course (September 2nd)
  – 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?
  – If people drop, we can move others off waitlist
  – Concurrent enrollment is after the waitlist
Infrastructure, Textbook & Readings

Infrastructure

– Website:  http://cs162.org

Textbook: Operating Systems: Principles and Practice (2\textsuperscript{nd} Edition)
Anderson and Dahlin

– 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, 3\textsuperscript{rd} edition, by Robert Love
Class Expectations

Lectures
– Come! Cannot guarantee content will be identical to previous years.
– Electronic devices used only for note taking.

Sections
– Attendance mandatory
– Meet your fellow students, they are your future colleagues!

Office Hours
– Come and ask for help early. No stupid question.
– We like teaching and want to meet you!
Communication Policy

Communicate with course staff through EdStem rather than Email.

Check Website for Piazza/EdStem etiquette. Posts will be deleted if they do not follow guidelines.
Individual Homeworks (2 weeks) - preliminary
0. Tools & Environment, Autograding, recall C, executable
1. Lists in C
2. BYOS – build your own shell
3. Memory & Management
4. Sockets & Threads in HTTP server
5. Distributed Systems

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
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)

- 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!
Getting started

Start homework 0 and Project 0 right away

– 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

Sections on Friday – attend any section you want

– We’ll assign permanent sections after forming project groups
– Section attendance will be mandatory after we form groups
Preparing Yourself for this Class

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!

Review Session on C/C++:
- TBA

"Resources" page on course website
- Ebooks on “git” and “C”

C programming reference (still in beta):
- https://cs162.org/ladder/
- First two sections are also dedicated to programming and debugging review. Attend ANY sections in first two weeks
Grading (Tentative breakdown)

36% three midterms (12% each)
- Tuesday, 27/9, 7-9 Wheeler 150
- Thursday, 27/10, In Class
- Thursday, 1/12, In Class

36% projects

18% homework

10% participation (Sections, Lecture, Clicker response ...)

Projects
- Initial design document, Design review, Code, Final design, Peer Review
- 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."

https://asuc.org/honor-code-landing/

All academic violations will be reported
CS 162 Collaboration Policy

Explaining a concept to someone in another group
Discussing algorithms/testing strategies with other groups
Discussing debugging approaches with other groups
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
✅ Helping someone in another group to debug their code

We compare all project submissions against prior year submissions and online solutions

Don’t put a friend in a bad position by asking for help that they shouldn’t give!
Course Environment

Implicit Bias
A form of bias that occurs automatically and unintentionally, that nevertheless affects judgments, decisions, and behaviors.

Every student has the right to learn and work in a safe environment

Ask yourself: I think/said X of/to student Y. Why?

Please reach out to course staff or Title IX office if there is an issue
New this year!

(Optional)

Lunch on the Glade
September 15th, October 16th, Nov 18th

Bring your lunch and meet Course Staff, Fellow Students for an informal gathering.

Get to know your future colleagues!
Summary: Goals for Today

• Why should you care?

• Why is it hard?

• What is an Operating System?
Summary: Goals for Today

- Why should you care?  
  The OS is everywhere

- Why is it hard?  
  Deal with many different devices, many different time scales. Safety-critical

- What is an Operating System?  
  Provides abstraction of a simple, infinite virtual machine  
  Three roles: illusionist, referee and glue  
  A good OS cares about performance, reliability, security and portability