Introduction

CS 4410

Operating Systems

Summer 2019

Edward Tremel

Cornell CIS
COMPUTING AND INFORMATION SCIENCE

[R. Agarwal, L. Alvisi, A. Bracy, M. George, E. Sirer, R. Van Renesse]

Course Logistics

Happy Independence Day!

- University is officially closed today
- Tomorrow is also a “break” day
- Summer session classes must meet anyway (on both days)

Who am I?

- PhD student in Computer Science
- About to graduate
- Previously: Brown class of 2013
- Research: distributed systems, datacenter networking, data privacy
- Advised by Prof. Ken Birman

Class Setup

- **Every day**, 11:30-12:45, in Gates G01

- Policies:
  - Sit near the front – this classroom is too big
  - No **cell phones** or **laptops** out during class
  - Studies show that classrooms without laptops are far more effective

- Please ask questions!
  - Small class, time for everyone to participate

Important Information

Website: `http://www.cs.cornell.edu/courses/cs4410/`

- Contains schedule, syllabus, links
- Lecture slides will be posted here

CMS: `https://cmsx.cs.cornell.edu`

- Assignments and due dates
- Submission and grades

Piazza: `https://piazza.com/cornell/summer2019/cs4410`

- Announcements by the instructor
- Ask and answer questions

Getting Help

Office Hours

- MWF 1-2 pm, T/Th 2-3 pm
- Gates 445

Piazza

- For help with assignments, concepts
- Private posts for communicating with just the instructor

Please no emails to personal email accounts

Assignments and Grades

Homework (5)

- Due each Monday before class (except Jul 8)
- Mix of written and programming problems

Quizzes (5)

- In-class quizzes, one each Wednesday

Grade Weights

- Homework: 45%
- Quizzes: 25%
- Final: 25%
- Class Participation: 5%

Academic Integrity

Closed-book exams, no calculators/phones

All submitted work must be your own

- OK to discuss concepts together
- Don’t share or copy solutions
- Don’t look up solutions to similar problems
- Don’t copy course materials

Introduction to Operating Systems

Cornell University Founded A.D. 1865

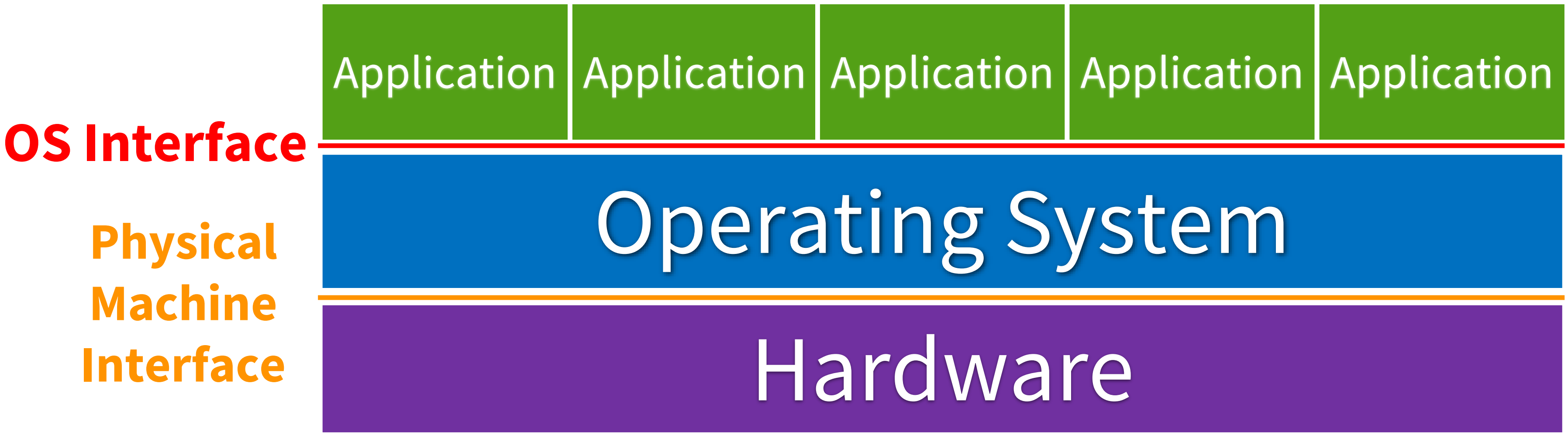
Cornell CIS
COMPUTING AND INFORMATION SCIENCE

Meet the OS

- 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

What is an OS?

An Operating System implements a virtual machine whose interface is **more convenient*** than the raw hardware interface



* easier to use, simpler to code, more reliable, more secure…

"All the code you did not write"

OS Wears Many Hats

Referee

- Manages shared resources: CPU, memory, disks, networks, displays, cameras, *etc.*

Illusionist

- Look! Infinite memory! Your own private processor!

Glue

- Offers set of common services (*e.g.*, UI routines)
- Separates apps from I/O devices

OS as Referee

Resource allocation

- Multiple concurrent tasks, how does OS decide who gets how much?

Isolation

- A faulty app should not disrupt other apps or OS
- OS must export less than full power of underlying hardware

Communication/Coordination

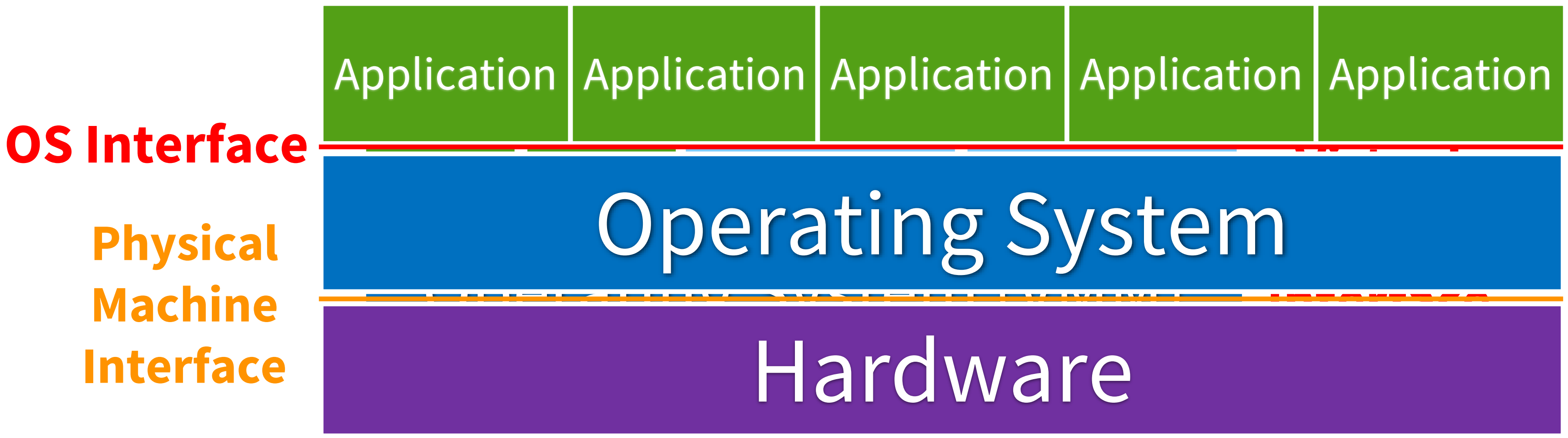
- Apps need to coordinate and share state

OS as Illusionist (1)

Illusion of resources not physically present

Virtualization:

- processor, memory, screen space, disk, network
- the entire computer:
  - fooling the illusionist itself!
  - ease of debugging, portability, isolation



OS as Illusionist (2)

Illusion of resources not physically present

- Atomic operations
  - HW guarantees atomicity at word level
    - what happens during concurrent updates to complex data structures?
    - what if computer crashes during a block write?
  - At the hardware level, packets are lost…
- Reliable communication channels

OS as Glue

Offers standard services to simplify app design and facilitate sharing

- send/receive of byte streams
- read/write files
- pass messages
- share memory
- UI

Decouples HW and app development

A Short History of Operating Systems

History of Operating Systems

Phase 1: Hardware expensive, humans cheap

User at console: single-user systems

Batching systems

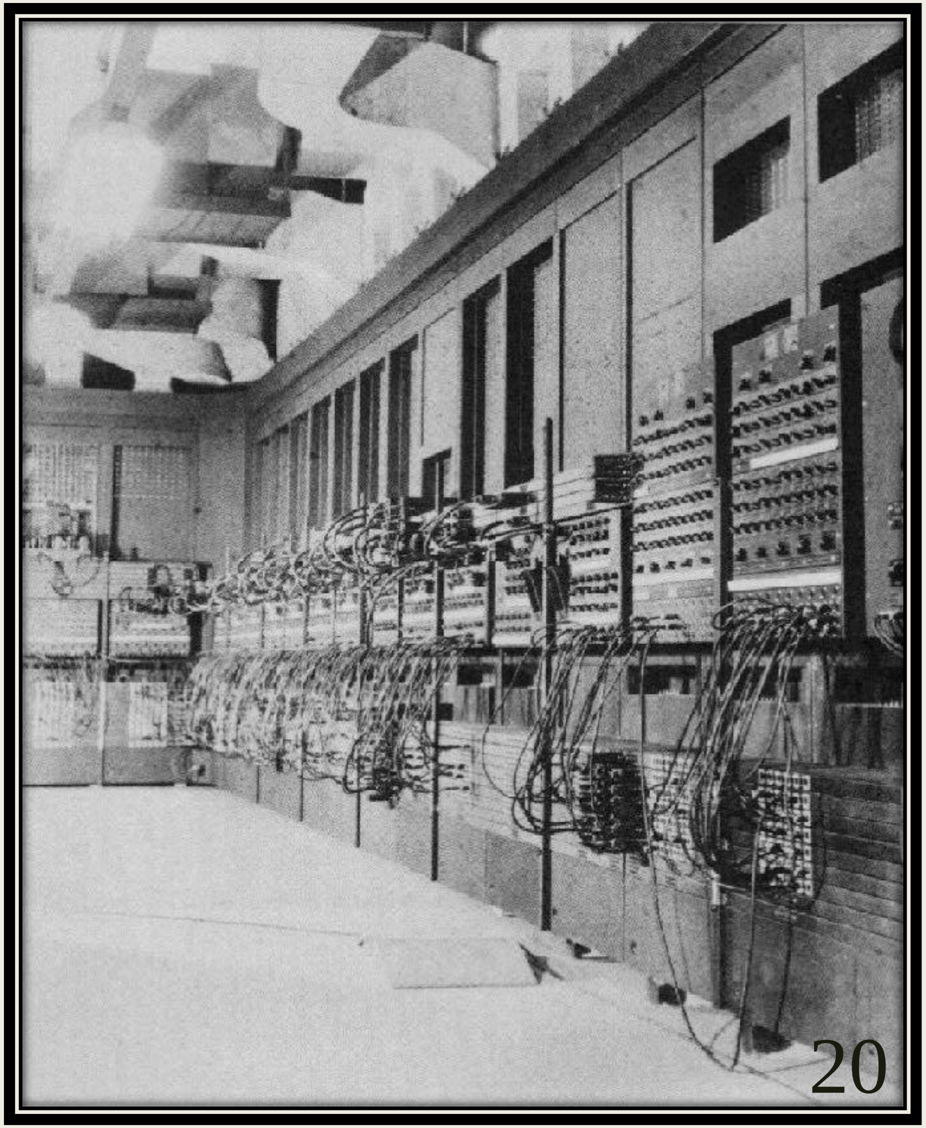
Multi-programming systems

Hand programmed machines (1945-1955)

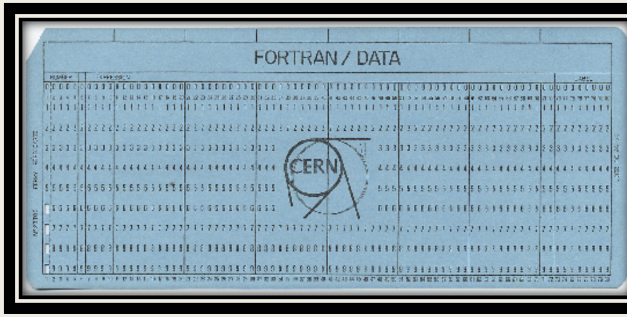
Single user systems

OS =
loader + libraries

Problem:
low **utilization** of expensive components



Batch Processing (1955-1965)



OS = loader +
sequencer +
output processor

Input

Card Reader

Tape

Compute

User Data

User Program

“System Software”

Operating System

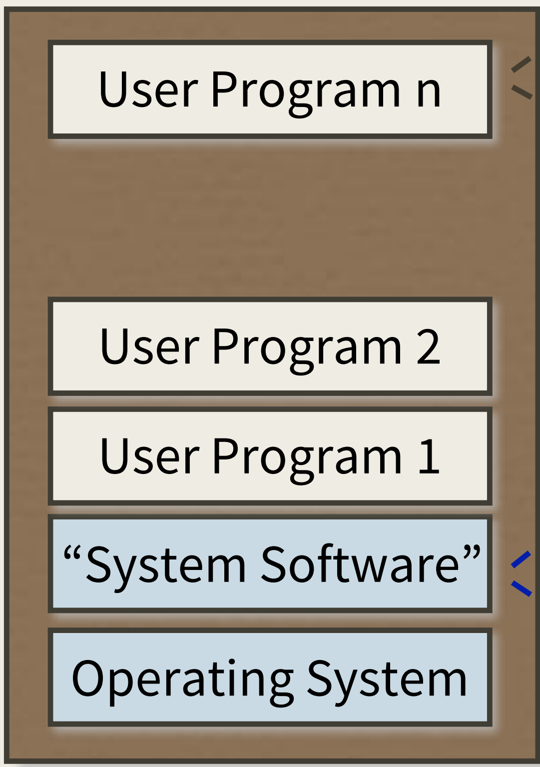
Tape

Printer

Output

Multiprogramming (1965-1980)

- Keep several jobs in memory
- Multiplex CPU between jobs.



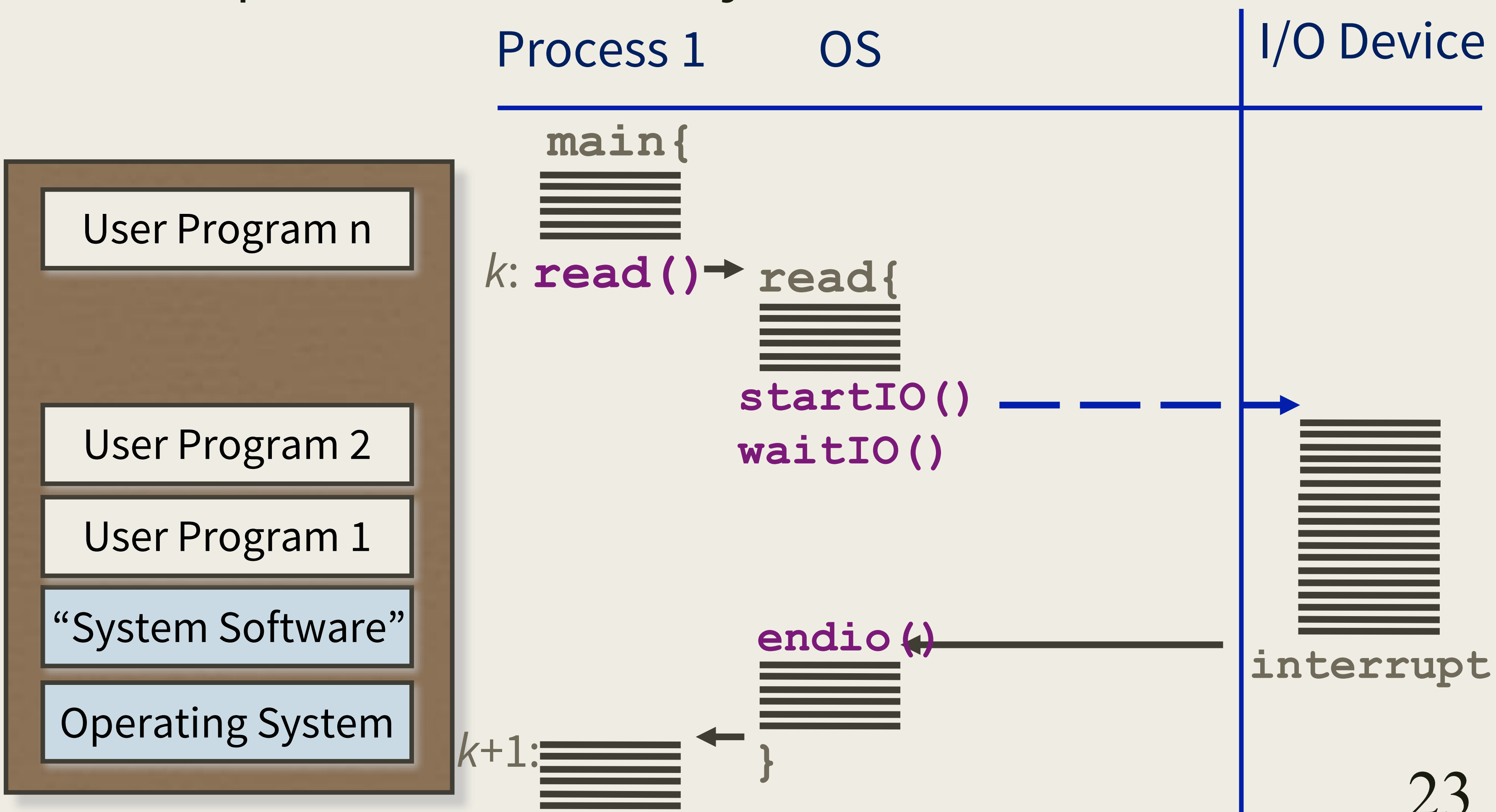

```
program P
begin
   ...
   Read(var)
   ...
end P
```

```
system call Read(var)
begin
   StartIO(input device)
   WaitIO(interrupt)
   EndIO(input device)
   ...
end Read
```

Multiprogramming (1965-1980)

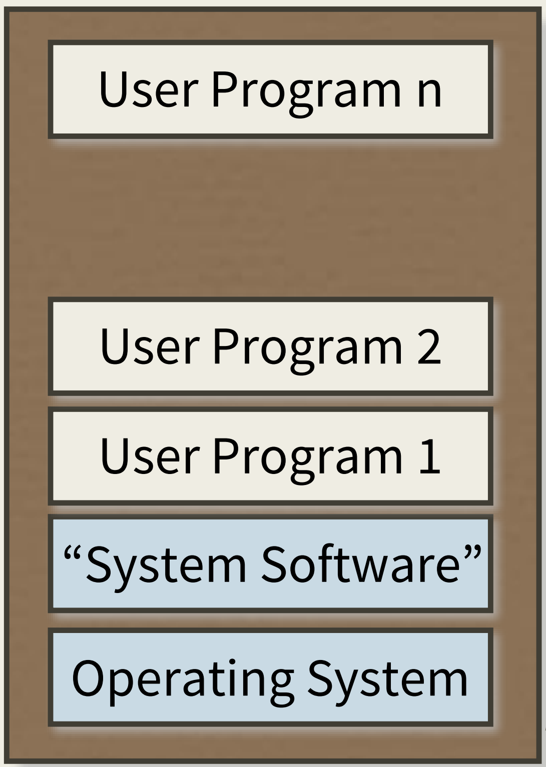
Keep several jobs in memory

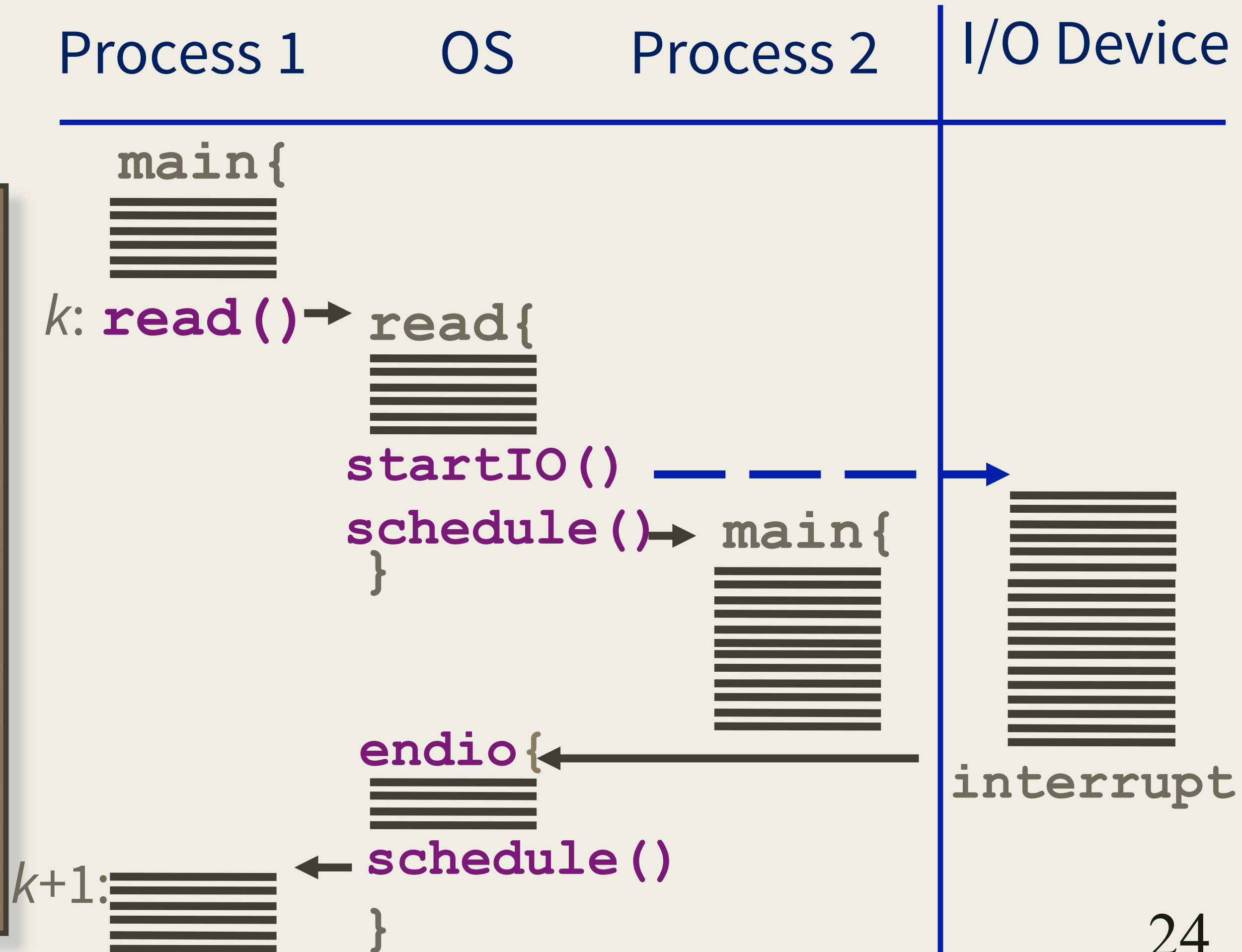
Multiplex CPU between jobs.



MULTIPROGRAMMING (1965-1980)

- Keep several jobs in memory
- Multiplex CPU between jobs.





History of Operating Systems

Phase 1: Hardware expensive, humans cheap

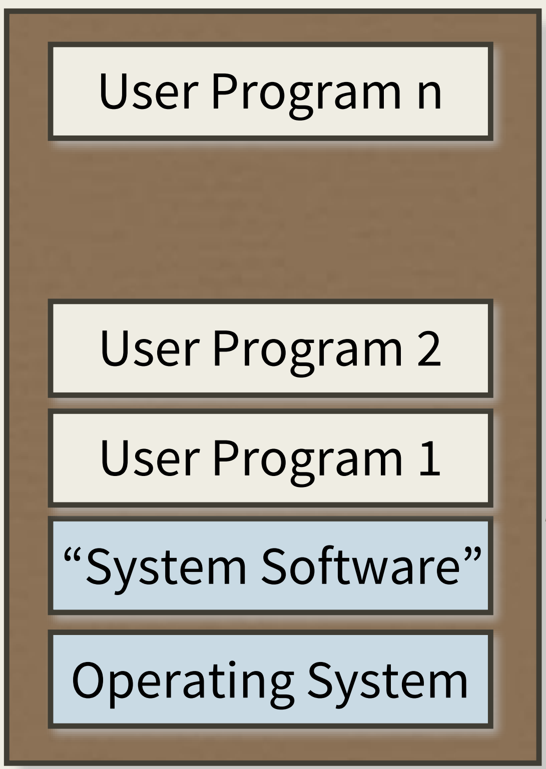
- *User at console: single-user systems*
- *Batching systems*
- *Multi-programming systems*

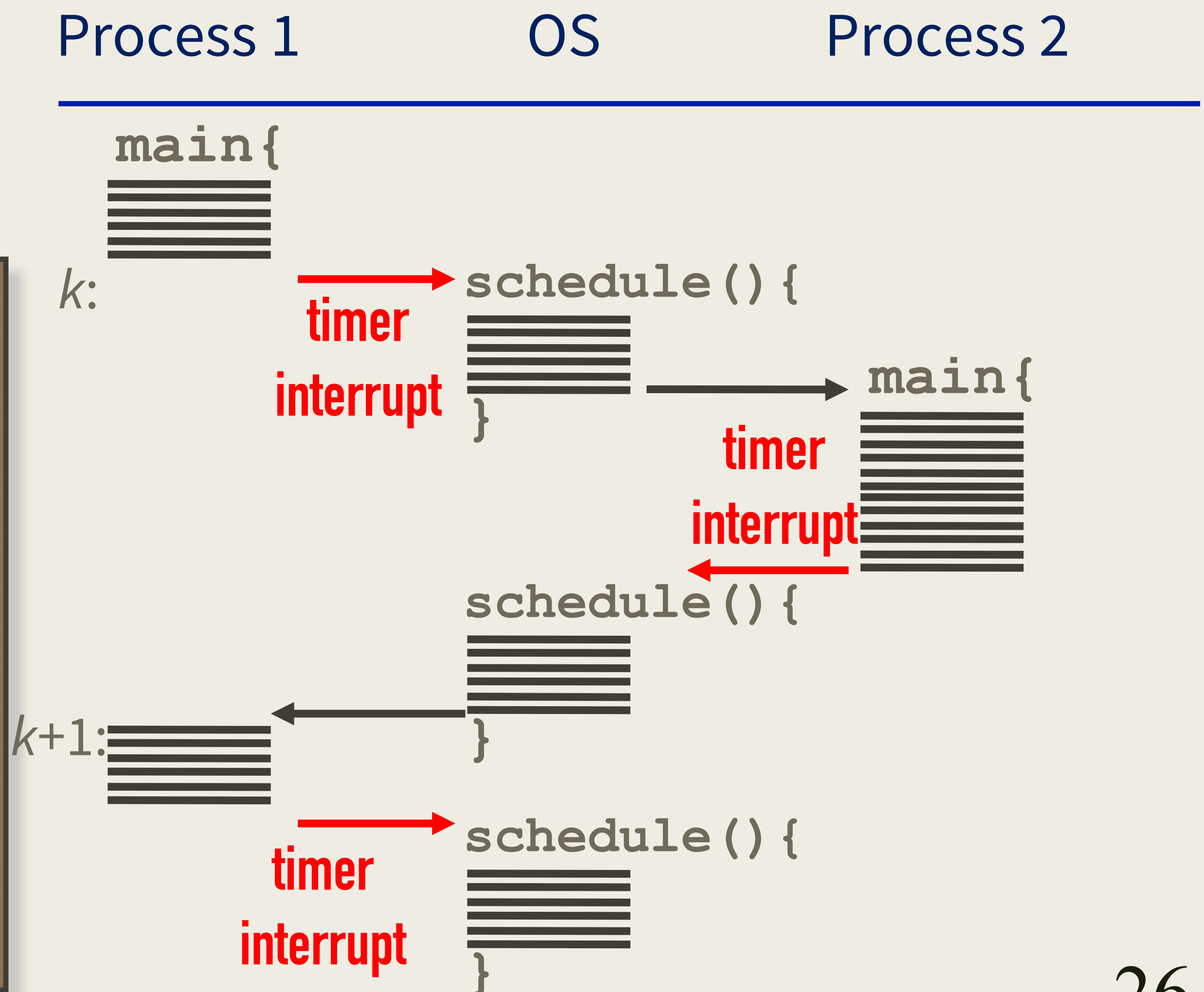
Phase 2: Hardware cheap, humans expensive

- *Timesharing: Users use cheap terminals and share CPU*

Timesharing (1970-)

- Timer interrupt used to multiplex CPU between jobs





History of Operating Systems

Phase 1: Hardware expensive, humans cheap

- *User at console: single-user systems*
- *Batching systems*
- *Multi-programming systems*

Phase 2: Hardware cheap, humans expensive

- *Timesharing: Users use cheap terminals and share CPU*

Phase 3: H/W **very** cheap, humans **very** expensive

- *Personal computing: One system per user*
- *Distributed computing: many systems per user*
- *Ubiquitous computing: LOTS of systems per user*

Operating Systems for PCs

Personal computing systems

- Single user
- Utilization no longer a concern
- Emphasis on user interface and API

Evolution

- Initially: OS as a simple service provider (libraries)
- Now: Multi-application with support for coordination

THE END

Why Study Operating Systems?

To Learn:

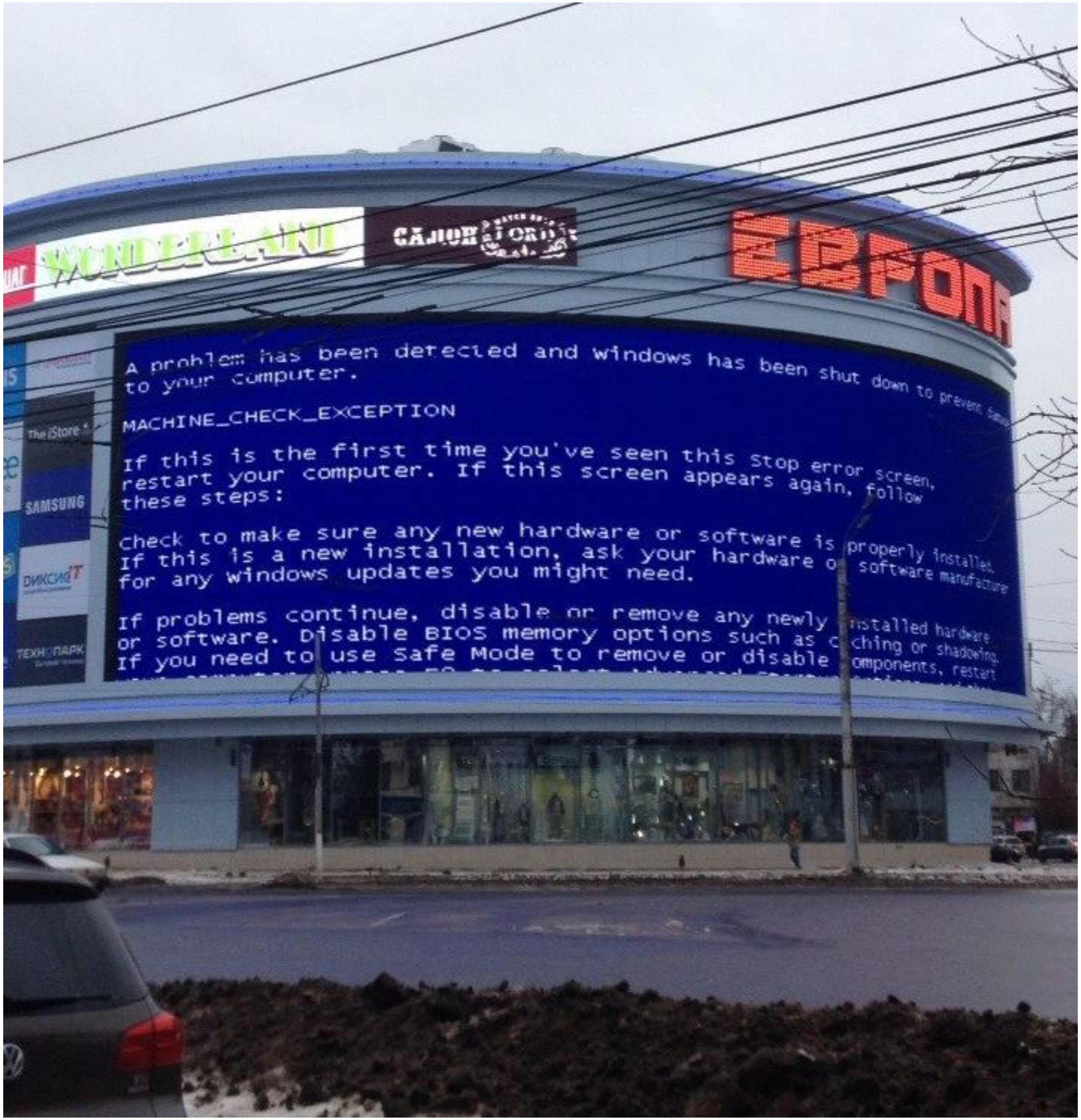
- **How to manage complexity** through appropriate abstractions
  - infinite CPU, infinite memory, files, locks, *etc.*
- **About design**
  - performance vs. robustness, functionality vs. simplicity, HW vs. SW, etc.
- **How computers work**

Because OSs are everywhere!

Where’s the OS?
Las Vegas

Where’s the OS?
New York


FUTUREPARK
A problem has been detected and windows has been shut down to prevent damage to your computer.
UNMOUNTABLE_BOOT_VOLUME
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen
these steps:
Check to make sure any new hardware
If this is a new installation, ask y
for any windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup options, and then
select Safe Mode.
Technical information:
*** STOP: 0x000000ED (0x82F5D030,0xC0000006,0x00000000,0x00000000)


WONDERLAND
ЕВРОПА
The iStore
SAMSUNG
ТЕХНОПАРК
A problem has been detected and windows has been shut down to prevent
to your computer.
MACHINE_CHECK_EXCEPTION
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart


memory opt
Mode to remo
to select Ac
00000004, 0x
h dump ...
ash dump ...

System Building is Hard

- The world is increasingly dependent on computer systems
  - Connected, networked, interlinked

- There is huge demand for people who deeply understand and can build robust systems (most people don’t and can’t)

- OS is a great example of a complex system that must be robust

Issues in OS Design

- **Structure:** how is the OS organized?
- **Concurrency:** how are parallel activities created and controlled?
- **Sharing:** how are resources shared?
- **Naming:** how are resources named by users?
- **Protection:** how are distrusting parties protected from each other?
- **Security:** how to authenticate, authorize, and ensure privacy?
- **Performance:** how to make it fast?

More Issues in OS Design

- **Reliability:** how do we deal with failures??
- **Portability:** how to write once, run anywhere?
- **Extensibility:** how do we add new features?
- **Communication:** how do we exchange information?
- **Scale:** what happens as demands increase?
- **Persistence:** how do we make information outlast the processes that created it?
- **Accounting:** who pays the bill and how do we control resource usage?

What’s this course about?

Ostensibly, operating systems

- architecting complex software
- identifying needs and priorities
- separating concerns
- implementing artifacts with desired properties

In reality, software design principles

- OSes happen to illustrate organizational principles and design patterns

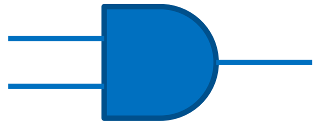
Topics (OS components)

- Devices and Architecture
- Processes and Threads
- Scheduling and Synchronization
  - Writing correct multithreaded programs
- Memory management
- Filesystems and storage
- Networking
- Security

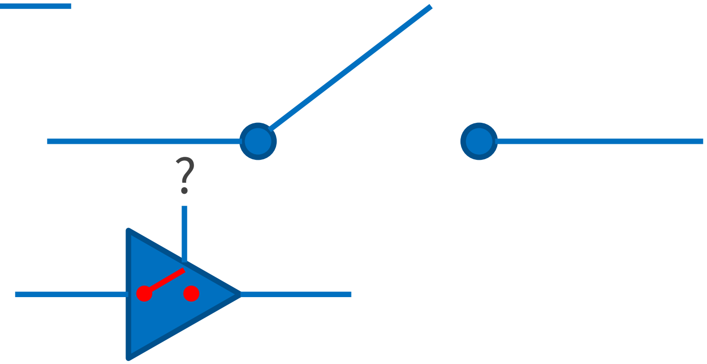
Activity: Keyboard Design

Keyboard Components

- Logic gates



- Switches for keys



- Tri-state buffers
- Encoders, multiplexers, latches…

Simple “Soviet-Era” keyboard

- Only 1 key pressed at a time
- CPU just needs to know which key