OPERATING SYSTEMS OVERVIEW

Azza Abouzied

ABOUT US

CLASS LOGISTICS

INFORMATION SOURCES

All the materials, assignments, readings, etc for the course can be found here: `azzadev.github.io/osbook` The passcode is os2022

Textbooks

DINOS: Operating System Concepts 9th Edition by Abraham Silberschatz, Peter B. Galvin, Greg Gagne.

Optional

LOVE: Linux Kernel Development 3rd Edition by Robert Love.

...optional books and readings will be shared online.

EXPECTATIONS

Deliverables

- 4-5 Lab projects: Bootloader, Kernel with no process preemption, Kernel with process preemption, Interprocess Communication, Virtual Memory, (Optional) File System
- Lots of work: 2-3 weeks per lab. Start on the day that the lab was assigned
- Group work: 2-3 students per group. Start to like your peers
- 100 hours total late days: No exceptions!
- Each lab has a graded *design review* a week after the assignment.
- Tutorials will be held online and Miro will schedule those as needed for each lab.
- Bonus Problem Sets

EXPECTATIONS

Readings

- Keep up with all the readings.
- You will interact better in class.
- Midterms can include material in assigned reading that was not covered in lecture.
- Your first readings for this week are **The Unix Time Sharing System** & **The Night Watch**.

Paper Cuts

- Read, understand, defend, fight, entertain
- Research papers
- Two groups debate each other, predetermined stance!

PLAGIARISM

DO NOT PLAGIARIZE

Do not share your code base with or without solutions online. Princeton has gladly shared their labs with us and other universities and we should honor that.

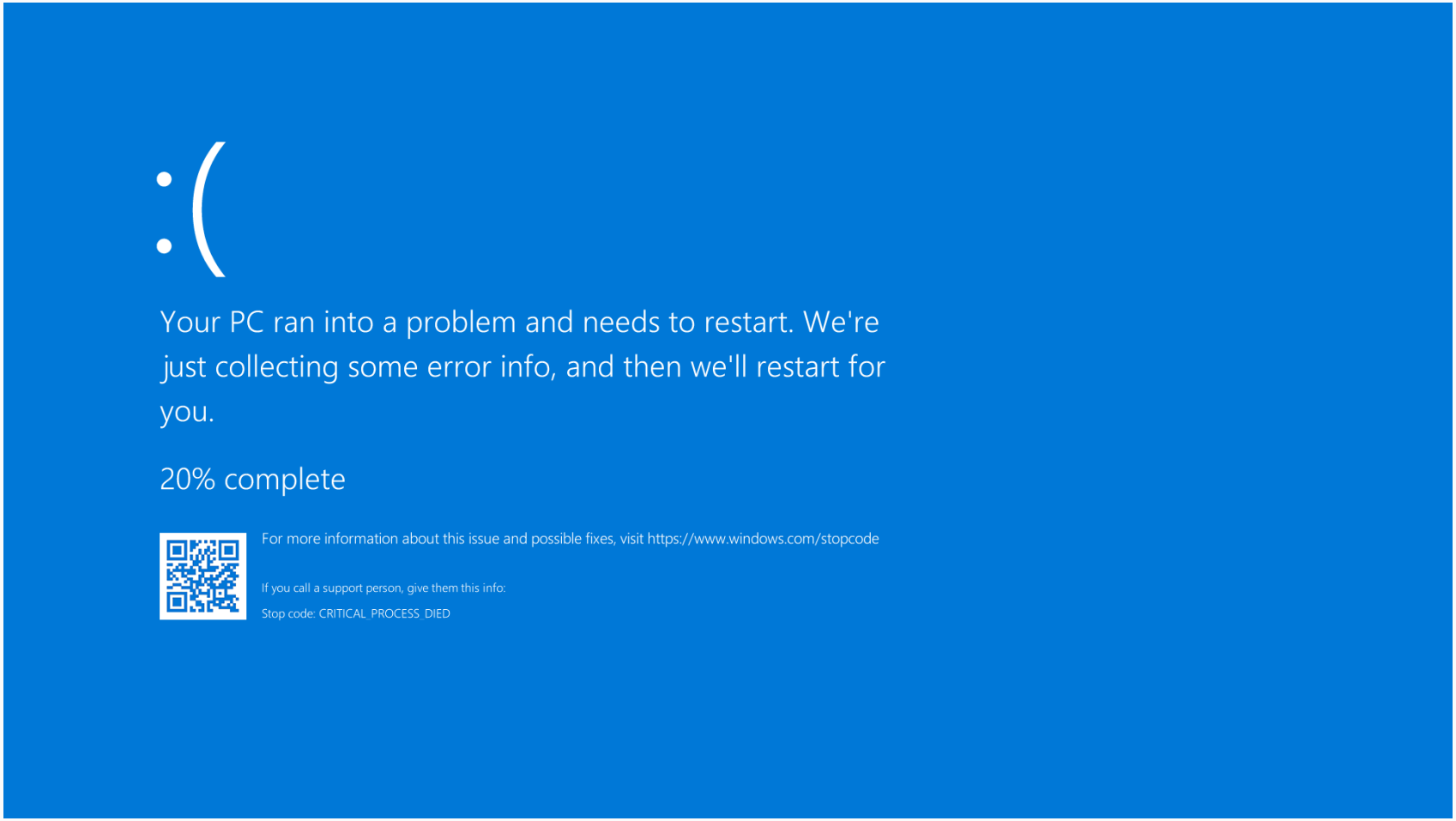
Feeling Overwhelmed

- Get help: talk to me or Miro, and do not wait until the last minute.
- Talk to your peers

WHAT IS AN OS?

THE END-USER PERSEPCTIVE

THE END-USER PERSEPCTIVE



Goal 1: Multiplex Resources

Goal 2: Abstract = Hide + Pretend

I/O Abstraction - *All hardware is the same*

- printer, screen, mouse: stream of bytes
- Disk, flash drives, remote network storage: bag of bytes
- *Unix: all devices are files!*
- Device drivers, standardized interface to all hardware provided by the OS

File system abstraction

- Load a file by a name
- Hide details of exact location, disk blocks, error correcting codes, pointer structures, disk vs. memory

WHAT IS THE PRETENSE?

Process Abstraction:

Actual resource: 1 CPU;

Pretense: Infinite CPU

Virtual Memory Abstraction:

Actual resource: 2GB of RAM;

Pretense: You have it all to your user program.

THE SYSTEM DEVELOPER PERSPECTIVE

WHAT MAKES BUILDING A KERNEL DIFFERENT?

Writing Linux kernel code is *different* from writing user-space code

1. No access to the C library or standard C headers
2. Coded in GNU C (instead of strict ANSI C) — has special features/conventions
    - Inline functions: trades-off memory footprint for speed
    - Inline assembly - kernel code is a mixture of C and assembly
    - Branch annotation - which if conditions are likely or unlikely to occur
3. No memory protection
4. Floating-point operations should be avoided
5. Small per-process fixed-size stack
6. Synchronization and Concurrency are hard
7. Has to port to different hardware architectures: endian neutral, 64-bit/32-bit support, do not assume word or page sizes, etc.

WHY STUDY IT?

1. It is required :)
2. Suffering
3. Modularity: any large system has to be divided into parts:
   - Layered
   - Micro-kernel
   - Exokernel
   - Monolithic
   - Virtualization
4. Security
   - Who do you trust? users, programmers, the kernel developers?
   - Separate address spaces, files, permissions, ...
5. Performance
6. The systems mindset: Tradeoffs
7. Short answer: things you learn here are relevant everywhere: you will be a cut above the rest!

HISTORY

UNIX

- 1969: Dennis Ritchie and Ken Thompson
- Grew out of painful experience with multics
- 1973: Rewritten in C!
- 1977: Picked up Berkeley leading to BSDs on different hardware platforms. Final BSD released in 1994 (4.4)
- Portability + Elegance = Influence

WHAT ABOUT UNIX?

- Simple: handful of system calls - designed with purpose.
- Elegant: everything is a file `open()`, `read()`, `write()`, `lseek()`, `close()`, `...`
- Written in C: portability
- Fast process creation time: `fork()`, `exec()`
- Clean IPC primitives ... library of simple programs that do one thing and do it well: a beautiful shell.

LINUX

- 1991: Linus Trovalds on the 80386 μ-processor develops Linux
- Tired of Prince of Persia on MS-DOS
- Posted it to the internet!
- Linux: free, and open source.

THE OS AS THE PROCESS | HARDWARE INTERFACE

FROM C CODE TO AN EXECUTABLE

- The Compilation Process
- The memory layout of a process
- Loading and executing
- What about shared libraries?

Questions?