Slide 1

Administrivia

- (I really do apologize about Monday. Not a good start to the semester, but things happen. I'm trying! "it's complicated" maybe.)
- Reading quiz 1 coming soon. *Note* that although all(?) of you have had a class with me before, I'm doing a few things differently this semester, in particular in how I ask you to turn things in . . .

Slide 2

Turning Work In

- In the past I've asked students to turn in work via e-mail. In principle that works well; in practice not. Also students say they can't remember.
- So this semester I'm going to ask you to use a setup similar to how I communicate grade information to you:

  One Google Drive folder per student, shared only between me and student. "Grades" folder for me to communicate with you; "TurnIn" folder for you to turn work in. Subfolders for each assignment.
- Folders should be set up and shared, though I didn't have Google notify you. Folder names of the form:

  CSCI1120-lastname,_initial
- Do use them; I set things up this way so it fits well with the rest of my course infrastructure.

Slide 3

Course Infrastructure

- Many instructors use some sort of "learning management system" such as TLEARN or Google Classroom.
- I don't — I have my own system, going back to before these were so popular, and partially automated using a collection of programs and scripts. Switching would be possible but painful.

Slide 4

The Pandemic and Trinity

- News from outside continues to be scary. But inside the Trinity bubble, we're safe? Maybe. Latest "Covid by the numbers" e-mail from the adminministration sounds good.
- Only three cases among faculty/staff! But one of those cases is — one of our visiting faculty! Somehow this makes it more real. Hers is a cautionary tale: She visited some family, all of them fully vaccinated except an 11-year-old. The 11-year-old is back in school in person, but had recently tested negative. They felt safe, took off their masks, and . . . all got sick. (The negative test result was misleading.) No one got *very* sick, but.

Slide 5

The Pandemic and Trinity, Continued

- Take-away message (my view!): Trinity insisting on masks in the classroom is more than hygiene theater.
- In fact in general caution is probably still a good idea. You may not be worried for yourself, but consider others, some of whom may be more vulnerable, some of whom are legitimately worried about exposing people they live with, etc.
- My two cents' worth!

Slide 6

Introduction

- The textbook's first two chapters are — I think — such a good, and readable, introduction to the what we will talk about that I doubt I can improve on them much.
- Yes, the authors are opinionated, and yes, it looks like they're a bit Linux-centric, but then so am I, and maybe it makes for livelier reading.
- A few things worth noting . . .

Slide 7

Textbook Versus(?) Topics from Syllabus

- Role and purpose of operating systems — introduction.
- History of operating systems — introduction (in a bit more depth shortly).
- Processes and process management, including a discussion of concurrency and related issues — virtualization (of CPU), concurrency.
- Memory management — virtualization (of memory).
- Input/output and device management — persistence.
- File systems — persistence.
- Operating system security — everywhere?
- Other topics as time permits (very likely some programming with Linux system calls) — everywhere?

Slide 8

Another Definition of Role and Purpose of O/S

- (From the textbook I had been using.)
- Top-down view: Provide a "virtual machine" easier to use than the real one. Key abstractions are processes, files.
- Bottom-up view: Manage physical resources on behalf of multiple applications, possibly multiple users.

Slide 9

Why Review History?

- To understand roots/development of current operating systems.
- As a way of getting many perspectives on "what do we want an O/S to do, and how do we make it do that?"
- Because history is intrinsically interesting? Try to imagine what using some of those early machines might have been like.
- (To allow the author and/or instructor to relive the days of his/her youth? Not so much with the authors of this year's book!)

Slide 10

The Early Days (1940s)

- Programming done by making physical connections on a plugboard (!).
- Better than no computer at all, but tedious and inefficient!
- Example: the ENIAC (picture via "links" page).

Slide 11

The Early Days (1940s – 1950s)

- Key improvements: stored-program concept, punch cards.
- Programming done by encoding machine language into cards.
- Program included code to start up computer, read rest of program into memory, do all input and output, etc. — no operating system.
- One program at a time, machine operated by programmer.
- Better, but still tedious and inefficient!

Slide 12

The Early Days (1950s)

- Key improvements: assemblers and compilers, libraries of commonly-used code, specialists to run machine (operators).
- Programming done in assembly language (or early high-level language), punched into cards.
- Separate steps to translate to machine language, execute.
- One program at a time, but machine operated by specialist.
- Less tedious, less inefficient.
- Still cumbersome for programmers, CPU idle between steps.

Slide 13

Batch Systems (1950s)

- Key improvement: "batch" idea — automate transitions between steps (translate program, execute, translate next program, etc.).
- How to make this work? separate input by "control cards", write primitive operating system to interpret them, manage transitions.
- Less inefficient, but I/O devices slow, so CPU idle a lot — still one program at a time.
- Still cumbersome for programmers — punch program into cards, give to operator, wait for output.

Slide 14

Control Cards — Example

Sketch of control cards for IBM mainframe O/S to compile and run:

```
//jobname   JOB  acctno,name, ....
//stepname EXEC PGM=compiler_name,PARM=(options)
//STEPLIB   DD   DSNAME=path_for_compiler
//SYSUT1    DD   UNIT=SYSDA,SPACE=(parameters)
//SYSPRINT DD    SYSOUT=A
//SYSLIN    DD   DSNAME=object_code,UNIT=SYSDA,
//               DISP=(MOD,PASS),SPACE=(parameters)
//SYSIN     DD   *
source code
/*
//stepname EXEC PGM=load-and-go
....
```

Slide 15

Multiprogramming Systems (1960s – ?)

- Key improvement: "multiprogramming" — more than one program in memory, so when one has to wait another can run.
- How to make this work? requires much more complex operating system — must share memory and I/O devices among programs, switch between them, etc.
- Efficient use of hardware.
- Still cumbersome for programmers — no real changes here.
- Example: IBM mainframe and peripherals (pictures on "links" page).

Slide 16

Timesharing Systems (1960s – ?)

- Key improvements: "interactive" users (using text terminals), utility programs to support them (shells, text editors, etc.).
- How to make this work? like multiprogramming, but now programs sharing memory are interactive users wanting fast response.
- Efficient use of hardware.
- Much less cumbersome for program development!
- Example: IBM terminal (picture on "links" page).

Slide 17

Personal Computers (1980s – ?)

- Similar evolution of operating systems — initially very simple, gradually becoming more complex/capable.
- Features from mainframes adopted as hardware permitted.
- A key difference — emphasis on user convenience rather than efficient use of hardware.

Slide 18

Evolution of Operating Systems, Recap

- Increasing hardware capability.
- Increasing O/S functionality and complexity — from simple program loader to complex multitasking system.
- Parallels between evolution of mainframe O/S and PC O/S. (Similar evolution may be happening with operating systems for “smart phones”?)

Slide 19

Minute Essay

- How are you liking the textbook so far?