Slide 1

Administrivia

- Things to note in the syllabus:
  - My office hours and e-mail address.
  - Course Web page (especially schedule).
  - Requirements and grading.
  - Policies on late work and academic integrity.

Slide 2

What Is An Operating System?

- Definition by example:
  - Windows, Linux, Unix, BeOs, OS X (Mac), . . .
  - VMS, MVS, VM/370, . . .
- Definition(s) from operating systems textbooks:
  - Something that provides “virtual machine” for application programs and users (“top down”).
  - Something that manages computer’s resources (“bottom up”).
- Definition justifying making you study them:
  - Important part of computer system.

CSCI 4320 August 28, 2003

Slide 3

What The Hardware Can Do

- CPU: fetch machine instruction from memory, execute; repeat.
- Disk: read data from / write data to location on disk.
- And so forth — very primitive.

Slide 4

What The Software Must Do

- Programs students usually write in PAD I/II:
  - Define and manipulate data structures.
  - Do arithmetic/logical calculations.
  - Read stdin / write stdout.
  - Call GUI/graphics library routines.
- The magic cloud:
  - Read from keyboard, write to screen.
  - Manage what’s on screen — windows, taskbar, etc.
  - Run multiple applications “at the same time”.
  - Manage disk contents — files, directories/folders.
  - Share the machine with other users.

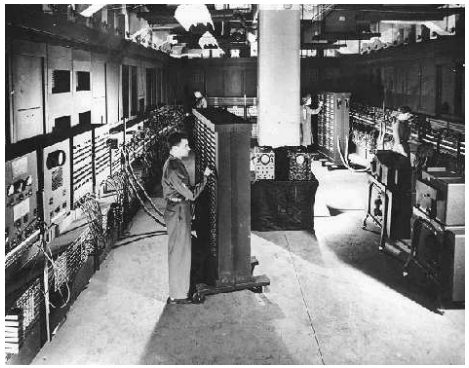
Slide 5

The Early Days (1940s)

- Programming done by making physical connections on a plugboard (!).
- Better than no computer at all, but tedious and inefficient!

Slide 6

ENIAC



Slide 7

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 8

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.

Batch Systems (1950s)

Slide 9

- 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.

Multiprogramming Systems (1960s – ?)

Slide 10

- 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.

IBM 360

Slide 11

Timesharing Systems (1960s – ?)

Slide 12

- 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!

Slide 13

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 14

Operating System Functions

- Implement useful abstractions:
  - Processes.
  - Filesystems.
- Manage resources for multiple users/applications:
  - CPU.
  - Memory.
  - I/O devices.

All this takes a lot of code! millions of lines, for current operating systems.

CSCI 4320 August 28, 2003

Slide 15

Minute Essay

- What are your goals for this course?
- Would you like to do a little programming in this course? a lot? none?
- What operating systems have you used?