Administrivia

- Any problems with lab machines? if so, ask me and/or send mail to CSadmin mailing list.
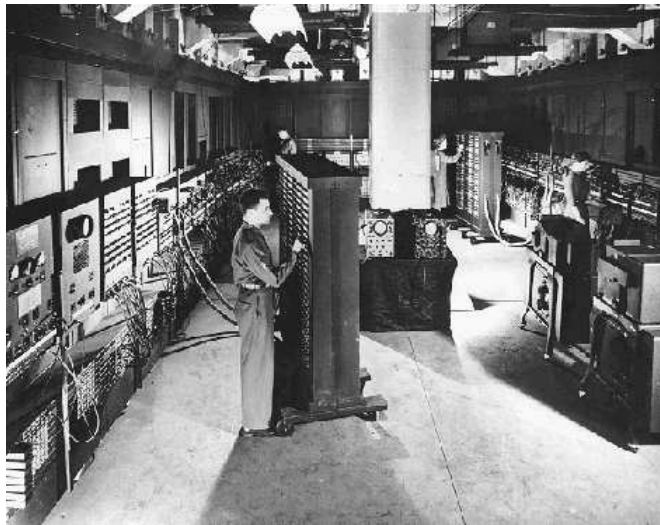
Slide 1

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?"

- (To allow the instructor to relive the days of his/her youth??)

Slide 2

Slide 3

The Early Days (1940s)

- Programming done by making physical connections on a plugboard (!).

- Better than no computer at all, but tedious and inefficient!

Slide 4

ENIAC

The Early Days (1940s – 1950s)

Slide 5

- 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!

The Early Days (1950s)

Slide 6

- 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)

- 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.

Slide 7

- 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 – ?)

- 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.

Slide 8

- Efficient use of hardware.

- Still cumbersome for programmers — no real changes here.

Slide 9

IBM 360

Slide 10

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!

Slide 11

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 12

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.

Slide 13

Minute Essay

- None — sign in.