Slide 1

Administrivia

- Homework 1 on Web (linked from “Lecture topics and assignments” page). Due next Monday.

Slide 2

Shell

- History — early batch systems had to interpret “control cards”; modern equivalent is to interpret “commands” (usually interactive).
- Not technically part of o/s, but important and related.
- Typical shell functionality:
  - Invocation of programs (optionally in background).
  - Input/output redirection.
  - Program-to-program connections (pipes).
  - “Wildcard” capability.
  - Scripting capability.
- Examples — MS-DOS `command.com`; Unix `sh`, `bash`, `csh`, `tcsh`, `ksh`, `zsh`, . . .

Slide 3

Recap — Operating System Functionality

- Two goals:
  - Bridge gap between what hardware will do (very primitive) and "virtual machine" useful for application-level programs.
  - Manage physical resources on behalf of multiple applications / users.
- Major functions:
  - Process management.
  - Memory management.
  - I/O subsystem.
  - File systems.
  - Security.
  - Shell.

Slide 4

Operating System Structures

- Clearly o/s could involve a whole lot of code (e.g., second edition of textbook says 29M for Windows 2000). How to structure?
- Choices include:
  - Monolithic systems.
  - Layered systems.
  - Microkernels.
  - Client-server model.
  - Virtual machines.
  - Exokernels.

Slide 5

Monolithic Systems

- Tanenbaum's description in the previous edition of the textbook — "The Big Mess". (Not completely unstructured, but close.)
- Examples include MS-DOS, early Unix.
- Advantages? "works, sort of" — often justification is historical.
- Disadvantages? "big mess". (Not everyone agrees, though.)

Slide 6

Layered Systems

- Idea — use layers of abstraction, just as one structures application programs.
- Examples include THE, MULTICS, OS/2, Windows NT (more so in early releases).
- Advantages? nice separation of concerns, modularity.
- Disadvantages? tricky to plan layers, performance can be slow.

Slide 7

Virtual Machines

- Idea — o/s provides a simulation of the actual physical machine, this "virtual machine" then runs another o/s – or several of them.
- Examples include VM/370, Windows support for old MS-DOS programs, VMware, Mac-on-Linux, Java Virtual Machine.
- Advantages? separates multiprogramming from other concerns, emulation aspect can be useful, useful in o/s development.
- Disadvantages? another layer, so can be slower.

Slide 8

VM/370

- Idea — provide multiple "virtual machines", each running its own o/s, which could be:
  - "Real" o/s such as MVS (another mainframe o/s) — in turn running many processes.
  - Not-quite-real o/s CMS — interactive single-user system rather like MS-DOS, runs under VM/370 only (not on real hardware).
- Allows sharing of physical resources among multiple "client" o/s's:
  - CPU sharing — similar to multitasking.
  - I/O device sharing — share physical devices, or allow exclusive use.

Slide 9

VM/370, Continued

- How does this work? briefly:
  - Client o/s's run native code, request o/s services in the usual way (interrupt or system call).
  - Interrupt handler is part of VM/370 — so it processes I/O requests/interrupts, errors, etc.
  - Client o/s system code runs in simulated supervisor mode (really user mode).
- Successors to VM/370 (VM/ESA, z/VM) currently being used to run many copies of Linux on a mainframe (!).

Slide 10

Minute Essay

- There is an old joke that says that any programming problem can be solved by adding a layer of abstraction, while any performance problem can be solved by removing a layer of abstraction.

  How (if at all) does this apply to operating systems and how they are structured?