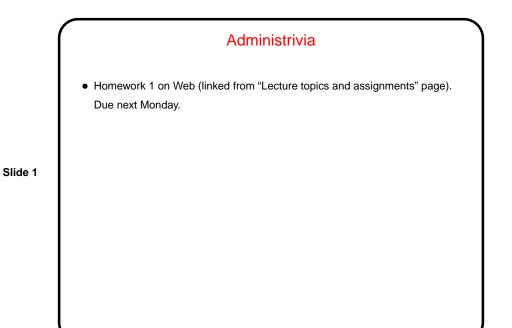
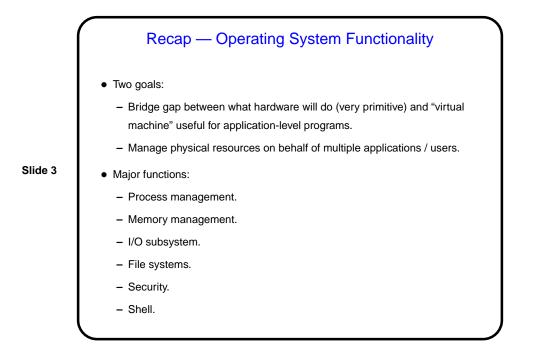
Slide 2



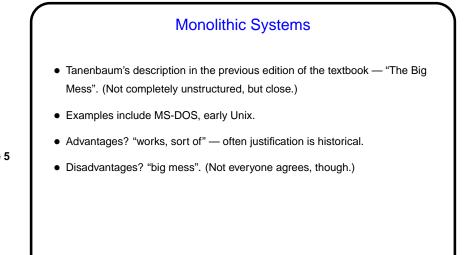
Shell
History — early batch systems had to interpret "control cards"; modern equivalent is to interpret "commands" (usually interactive).
Not technically part of o/s, but important and related.
Typical shell functionality:

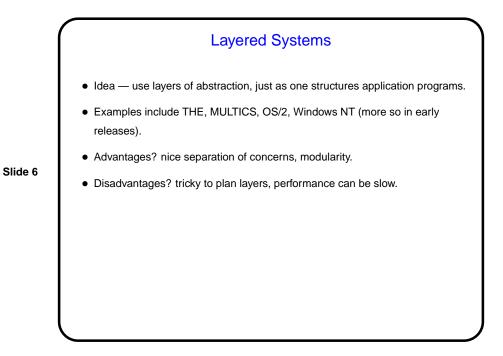
Invocation of programs (optionally in background).
Input/output redirection.
Program-to-program connections (pipes).
"Wildcard" capability.
Scripting capability.

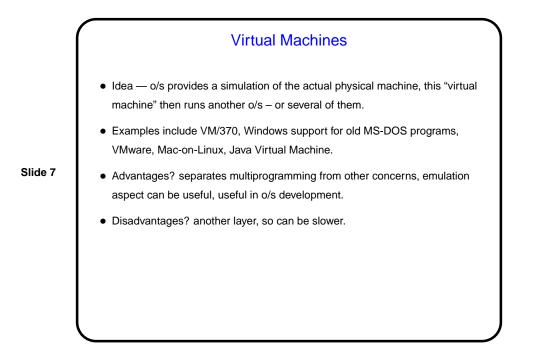
Examples — MS-DOS command . com; Unix sh, bash, csh, tcsh, ksh, zsh, ...

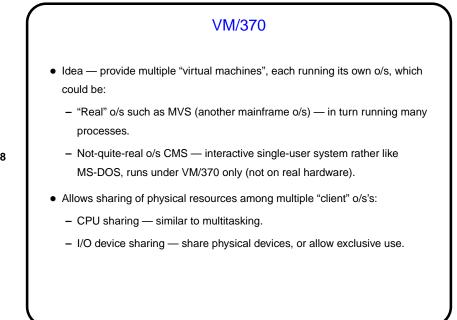


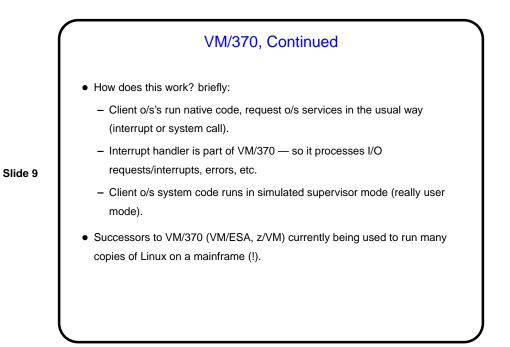
Operating System Structures • Clearly o/s could involve a whole lot of code (e.g., second edition of textbook says 29M for Windows 2000). How to structure? • Choices include: • Monolithic systems. • Layered systems. • Layered systems. • Client-server model. • Virtual machines. • Exokernels.











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?