What is an Operating System?

- Makes using the computer convenient
  - does a lot of the dirty work for you
  - hides details about the system behind a clean interface
- Makes using the computer efficient
  - expertly manages and allocates resources
- These goals are often contradictory

Views of the OS

- The OS is a context
  - An environment for user applications to run in
  - Provides the services that applications need
  - All programs on the system use this context
- The OS is a controller
  - Controls the I/O devices and user programs
  - Prevents and handles errors
Views of the OS (continued)

• The OS is a resource allocator
  » A system has many resources: CPU time, memory, disk space, access to I/O devices
  » The OS allocates these resources
  » Policies are generally configurable
    • allocate evenly among all uses, or
    • give more to those who pay more, or
    • prefer to give it to uses with high priority, or ...

What makes up the OS?

• “Just the kernel”
  » the program that starts running at boot time, manages all user programs, and runs until shutdown
• or “All the code you didn’t write”
  » all system libraries, compilers, assemblers
  » all the software shipped with the machine

OS issues for the user

• how are resources shared among users?
• what level of performance is available?
• how are failures prevented and dealt with?
• how are resources named and assigned?
• how is the flow of information restricted?
• how do we control and charge for resource usage?

OS issues for the sysadmin

• how are programs protected from others?
• how are new features added?
• what happens as resource needs increase?
• are new versions always compatible with old?
• can the components of the system be geographically separated?
OS issues for the programmer

• how can the data for a program persist?
  » from one execution to the next
  » from one generation to the next
• how is information exchanged?
  » between systems, applications, users, ...
• how are parallel activities controlled?
• how is the OS organized?

In Olden Times...

• The first operating systems were known as *batch systems*
  » OS was loaded once into a portion of memory
  » Programs stored on punch cards or paper tape
  » One by one, programs were loaded and run
  » Each program came with *control cards* telling the OS what to do

Multiprogramming

» Increase utilization of the processor

• Enabling technology
  » decrease in memory prices
• Keep multiple jobs loaded in memory
• While one program waits for I/O, run another one for a while

Timesharing

» Allow multiple users/programs to share a single system concurrently

• Based on time-slicing (1960s)
  » divide the CPU equally among the users
• For the first time, users could view, edit, and debug programs “on-line”
• Multics was first large timesharing system
Minicomputers

- Enable “small scale” applications
- Low cost hardware could run sophisticated applications (1970s)
  » didn’t need all the overhead of large mainframe system installations
  » small businesses, science and engineering
  » still focussed on efficient multi-user services
- One computer per department
  » See the VAX/780 in the Allen Center Atrium

Microcomputers

- Enable “small scale” applications
- Low cost hardware could run sophisticated applications (1980s)
  » didn’t need all the overhead of minicomputer systems
  » very small businesses, scientists and engineers
  » very focussed on the individual user

Networked Workstations

- Enable enterprise and web applications
- Individual workstation is only part of the system
- Connectivity and security very important
- Rebirth of sophisticated operating systems for the end user

Windows XP / Mac OSX / Linux are “real” operating systems

Real-Time Operating Systems

- Specialized operations: subway systems, flight control, factories, nuclear power plants, ...
- RTOS must guarantee response to physical events in a fixed time interval
  » Problem is to schedule all activities in order to meet all of the critical requirements
  » Solution is over-capacity and careful design
Tightly-coupled Systems

- Support parallel applications wishing to get speedup of computationally complex tasks
- Needs basic primitives for dividing one task into multiple parallel activities
- Supports efficient communication between those activities
- Supports synchronization of activities to coordinate sharing of information

Loosely-coupled Systems

- Sharing of distributed resources, hardware, and software to improve utilization and performance
  - speedup through parallelism
  - improved reliability
- Supports communication between parts of a job or different jobs
- Incorporate commodity processors
- Potentially huge clusters (Google, others)