Operating Systems
CS451

Brian Bershad
http://www.cs.washington.edu/homes/bershad
http://www.cs.washington.edu/homes/bershad/cs451

Today

- Biz
  - » grading
  - » office hours
  - » agenda
- History

Agenda

- Week 1: Intro issues
  - » readings: Chapters 1 & 2
- Week 2: Processes and process management
- Week 3: Threads and synchronization
- Week 4: More synchronization
- Week 5: Virtual memory
- Week 6: Filing and Secondary Storage
- Week 7: Networks
- Week 8: Distributed File Systems
- Week 9: Slop

Getting in Touch

- My office hours
  - » M @ 1.30pm
  - » F @ 10.30am
  - » 318 Sieg
- Gretta's hours
  - » TBA

Grading

- 100% total
  - » 40% homeworks and quizzes
    - – quizzes due (and postmarked) by Fri 11.59pm
      - cse451-hw@cs.washington.edu
  - » 60% exams
    - – 40% final
    - – 20% midterm

What is an Operating System?

- There are lots of definitions
  - » "all the code that you didn't write"
  - » the code that manages physical (hardware) resources
- provides users with "logical" well-behaved environment
- O.S. defines a set of logical resources (objects) and a set of well-defined operations on those objects (i.e., an interface to the use of those objects)
- provides *mechanisms* and *policies* for the control of objects/resources
- controls how different users and programs interact
- The *platform* that every developer can expect to be there
  - » critical market/customer/ISV control point

What resources need to be managed?

- the CPU (computation)
- primary memory
- secondary memory devices (disks, tapes, etc)
- networks
- input devices (keyboard, mouse)
- various I/O devices (printers, display, cameras, speakers)

Major issues in Operating Systems

- structure -- how is an operating system organized?
- sharing -- how are resources shared among users
- naming -- how are resources named (by users or programs)
- protection -- how is one user/program protected from another
- security -- how to restrict the flow of information
- performance -- why is it so slow?
- reliability and fault tolerance -- when something goes wrong
- extensibility -- how do we add new features?
- communication -- how and with whom can we communicate (exchange information)

Major issues in OS (2)

- concurrency -- how are parallel activities created and controlled?
- scale and growth -- what happens as demands or resources increase?
- persistence -- how to make data last longer than programs
- compatibility -- can we ever do anything new?
- distribution -- accessing the world of information
- accounting -- who pays the bills, and how do we control resource usage?

A brief history of operating systems

- "in the beginning", the OS was just code to which you linked your program, loaded the whole thing into memory, and ran your program; basically, just a run-time library
  - » each computer vendor/system had its own OS
  - » OS/360 in 1964 changed this
- simple batch systems were first real operating systems:
  - OS was stored in part of primary memory
  - it loaded a single job (from card reader) into memory
  - ran that job (printed its output, etc.)
  - loaded the next job...
  - *control cards* in the input file told the OS what to do
- *Spooling* and *buffering* allowed jobs to be read ahead of time onto tape/disk or into memory.

Multiprogramming

- *Multiprogramming* systems provided increased utilization
  - brought on by decrease in RAM prices
  - keeps multiple runnable jobs loaded in memory
  - overlaps I/O processing of a job with computes of another
  - benefits from I/O devices that can operate asynchronously
  - requires the use of interrupts and DMA
  - tries to optimize *throughput*

Timesharing

- *Timesharing* supported *interactive* use of
  - each user feels as if he/she has the entire machine (at least late at night!)
  - timesharing tries to optimize *response time*
  - based on time-slicing -- dividing CPU equally among the users
  - permitted active viewing, editing, debugging, participation of users in the execution process
- MIT Multics system (mid-late 1960s) was first large timesharing system

Real-Time Operating Systems

- real-time os are used for specialized applications: subway systems, flight control, factories, nuclear power plants...
- basic idea: os must guarantee response to physical events in a fixed time interval
- problem is to schedule all activities in order to meet all of their critical requirements
- basic approach: over-capacitize

Distributed Operating Systems

- distributed systems facilitate use of geographically distributed resources
- supports communication between parts of a job or different jobs
- sharing of distributed resources, hardware and software to improve utilization
  - » also can speedup through parallelism
  - » improve reliability

Parallel Operating 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

CSE 451

- In this class we will learn:
  - what are the parts of an O.S.
  - how is the o.s. and each sub-part structured
  - what are the important interfaces
  - what are the important policies
  - what algorithms are typically used
- We will do this through reading, lectures, and a project.
- You will need to keep up with all three of these.

For this week

- Read P&S Chapters 1,2
- Homework: All questions