Introduction to Operating Systems and Practicum in Operating Systems

COS 414/415
Spring 2001
Prof. Emin Gün Sirer

Administrative

- Instructors:
  - Prof. Emin Gün Sirer, eg@cs, 4119A Upson, MF 2:30-3:30
- Communication
  - cs414@cs.cornell.edu
- TAs:
  - Ben Atkin, batkin@cs
  - Indranil Gupta, gupta@cs
  - Ranjita Jain, rjain@cs
  - Shrinivas Samat, samat@cs
- All official information is on the web site

Grading

- Course prerequisite: Mastery of the material in CS 314, computer architecture
- 414: Intro to Operating Systems
  - Reading Assignments (~20%)
  - Midterm (~30%)
  - Final (~50%)
- 415: Practicum in Operating Systems
  - Three projects (100%)
- This is a rough guide
Course Outline

• History, architectural support, structure
• Concurrency, processes, threads
• Synchronization, monitors, semaphores
• Memory Management, virtual memory
• Storage Management, I/O, filesystems
• Networking, distributed systems
• Security

What is an Operating System?

• Definition: An Operating System (OS) provides a virtual machine on top of the hardware that is more convenient than the raw hardware interface
  • “All of the code you did not write”
  • Simpler
  • More reliable
  • More secure
  • More portable
  • More efficient
  • …

What Resources Need to Be Managed?

• The CPU(s)
• Memory
• Storage devices (disks, tapes, etc)
• Networks
• Input devices (keyboard, mouse, cameras, etc.)
• Output devices (printers, displays, speakers, etc.)

What do Operating Systems Do?

• Manage physical and virtual resources
• Provide users with a well-behaved environment
• Define a set of logical resources (objects) and a set of well-defined operations on those resources (i.e. an interface to those objects)
• Provide mechanisms and policies for the control of resources
• Control how different users and programs interact
What’s in an OS?

<table>
<thead>
<tr>
<th>Machine Dependent Services</th>
<th>Physical Machine Inf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupts, Cache, Physical Memory, TLB, Hardware Devices</td>
<td></td>
</tr>
</tbody>
</table>

Generic I/O

Machine Independent Services

OS Interface

Logical OS Structure

Applications

• Quarters
• SQL Server

System Utils

Shell

Windows & graphics

Networking

Virtual Memory

Access Control

Genie ID

File System

Process Management

Device Drivers

Memory Management

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 authenticate, control access, and secure privacy?
• Performance -- why is it so slow?
• Reliability and fault tolerance -- how do we deal with failures?
• Extensibility -- how do we add new features?
• Communication -- how can we 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 outlast the processes that created them
• 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

• Initially, the OS was just a run-time library
  • You linked your application with the OS, loaded the whole program into memory, and ran it
  • How do you get it into the computer? Through the control panel?
• Simple batch systems
  • Permanently resident OS in primary memory
  • It loaded a single job from card reader, ran it, and loaded the next job...
  • Control cards in the input file told the OS what to do
  • Spooling allowed jobs to be read ahead of time onto tape/disk or into memory

Compute I/O

Compute I/O
Multiprogrammed Batch Systems

- Multiprogramming systems provided increased utilization
  - Keeps multiple runnable jobs loaded in memory
  - Overlaps I/O processing of a job with computation of another
  - Benefits from I/O devices that can operate asynchronously
  - Requires the use of interrupts and DMA
  - Optimizes for throughput at the cost of response time

Timesharing

- Timesharing supported interactive computer use
  - Each user connects to a central machine through a cheap terminal, feels as if she has the entire machine
  - Based on time-slicing -- dividing CPU equally among the users
  - Permitted active viewing, editing, debugging, participation of users in the execution process
  - Security mechanisms required to isolate users from each other
  - Requires memory protection hardware for isolation
  - Optimizes for response time at the cost of throughput

Distributed Operating Systems

- Distributed systems facilitate use of geographically distributed resources
  - Machines connected by wires, no shared memory or clock
  - Supports communication between parts of a job or different jobs
  - Interprocess communication
  - Sharing of distributed resources, hardware and software
  - Resource utilization and access
  - Permits some parallelism, but speedup is not the issue

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
  - It’s common now to use networks of high-performance PCs/workstations as a parallel computer
Real-time Operating Systems

- Goal: To cope with rigid time constraints
- Hard real-time
  - OS guarantees that applications will meet their deadlines
  - Examples: TCAS, health monitors, factory control, etc.
- Soft real-time
  - OS provides prioritization, on a best-effort basis
  - No deadline guarantees, but bounded delays
  - Examples: most electronic appliances
- Real-time means “predictable”
  - NOT fast

Personal Computing

- Computers are cheap, so give everyone a dedicated computer
- Initially, the OS became a library again due to hardware constraints
- Multiprogramming, memory protection, and other advances were added back
  - For entirely different reasons

Ubiquitous Computing

- The decreased cost of processing makes it possible to embed computers everywhere. Each “embedded” application needs its own control software:
  - PDAs, cell phones, intelligent appliances, etc.
- In the near future, you will have 100s of these devices
  - If not already
- Poses lots of problems for current systems
  - Structure, naming, scaling, security, etc.
- We will tackle some of them in this class

Lessons from History

- The point is not that batch systems were ridiculous
  - They were exactly right for the tradeoffs at the time
- The tradeoffs change

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>30</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>RAM</td>
<td>100</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>Disk</td>
<td>100</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>Net Bandwidth</td>
<td>100kb, 1000kb, 10gb</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Need to understand the fundamentals
  - So you can design better systems for tomorrow’s tradeoffs
COS 414/415

• In this class we will learn:
  • What the parts of an OS are
  • How the OS and each sub-part is structured
  • What the important mechanisms are
  • What the important policies are
  • What algorithms are typically used

• We will do this through reading, lectures, and a project
  • Project will involve some aspect of ubiquitous computing using Palmax @migo 600's equipped with Aeronet cards
  • Reading: Chapters 1 & 2
  • Reading Assignment: Due Friday, described on the web

• You will need to keep up with all three of these