CS 355
Operating Systems

Introduction

Administrative Things

• Lab
  – Tuesdays after class, but some Thursdays too
  – Location TBA (either 231 or 232)
  – No lab today
• Course website
• CS account
  – “powerpuff” is our gateway
  – Do NOT program on “powerpuff” directly

Background

• Working knowledge of C/C++
• Familiarity with Unix
• Basic understanding of computer organization and architecture
• A “systems” mentality

Grading

• Weekly or bi-weekly projects 60%
• 2 in-class exams 30%
• Written psets and discussions 10%

Systems Programming

• You WILL be stuck a lot
• Learn to read the manual pages
• Memory errors and segfaults are not tolerated
• Use a debugger!
• Outside references
  – Stackoverflow is rarely right
  – Using code you don’t understand NEVER helps
  – If you use any, you MUST attribute

What is an OS?

• You probably think you know right now.
• You probably won’t think you know 6 weeks down the line.
• You may or may not know at the end of this class.
What does an OS do for you?

- Want to print “Hello World!” on a printer.

An Operating System Provides an Interface btw Hardware and User Programs

<table>
<thead>
<tr>
<th>Banking system</th>
<th>Active reservation</th>
<th>Web browser</th>
<th>Compilers</th>
<th>Editors</th>
<th>Command interpreter</th>
<th>Operating system</th>
<th>Machine language</th>
<th>Microarchitecture</th>
<th>Physical devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application programs</td>
<td>System programs</td>
<td>Hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Operating System as an Extended Machine

- Beautiful Interface
- Ugly Interface

An OS

- Get printer manual
  - Find out how to send messages to it
- Write program
  - Put string “Hello World!” in memory buffer
  - Do what printer requires to send buffer to it
  - Go into waiting loop
- Get hold of a Computer

NO OS

- Translate your program into machine code
- Figure out a way to load the program into memory
- Start the program (somehow)
- Turn off computer

Resource Abstraction and Sharing

- It is an extended machine providing abstraction of the hardware
  - Hides the messy details which must be performed
  - Presents user with a virtual machine, easier to use

- It is a resource manager
  - Time on CPU is shared among multiple users/programs
    - Who gets to run when?
  - Space in memory and on disks is shared among multiple users/programs

Pentium Architecture
Abstractions in OS

Hardware
- Disks
- Memory
- Processors
- Network
- Monitor
- Keyboard
- Mouse

OS abstraction
- Files
- Programs
- Threads / Processes
- Communication
- Windows and GUI
- Input
- Locator

Sharing of Memory

Issues
- Allocation schemes
- Protection from each other
- Protecting OS code
- Translating logical addresses to physical
- Swapping programs
- What if physical memory is small: Virtual memory

Timesharing

At any point, only one program can run on CPU
- Context switch: changing the program that has control of CPU
- When to switch (goal: to optimize the CPU usage)
  - A program terminates
  - A program has run "long enough"
  - A program executes a system call or waits for I/O
  - An external interrupt arrives (e.g. mouse click)
- OS must do all the book-keeping necessary for context switch, with minimum number of instructions

Challenges in OS

- Performance is critical
  - how to reduce the memory and time overhead due to OS
- Synchronization and deadlocks due to shared resources
- Scheduling of multiple programs
  - Fairness, response time, real-time applications

Challenges in OS

- Memory management
  - Virtual memory, paging, segmentation
- Security and protection
  - Authorization, authentication and viruses
- Interrupt management and error handling
- Marketability and backwards compatibility

Supervisor(Kernel)/User Mode

- Kernel/User space
- When a CPU is in kernel mode, it can do anything and address any part of memory
  - OS kernel runs in supervisor mode
  - Supervisors can switch to user mode at will
- User mode only has access to its own address space and can’t talk directly to devices
  - User processes run in user mode
  - Must execute a “trap” to switch to kernel mode
Trap

- A user program can switch to kernel mode by executing a trap
  - it then starts running in protected kernel space
- Traps are used to implement system calls to provide services to user processes:
  - communication with a device
  - communication with another process
- Traps also happen when devices issue "interrupts"
  - clock ticks, disk errors, message arrives on network, etc

Inside a CPU

- State of a running program
  - Registers
  - Program counter (PC)
  - Stack pointer
  - Program status word (PSW)
- Key distinction in PSW: user mode vs kernel (OS) mode

Different types of Memory

<table>
<thead>
<tr>
<th>Typical access time</th>
<th>Typical capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 nsec</td>
<td>&lt;1 KB</td>
</tr>
<tr>
<td>2 nsec</td>
<td>4 MB</td>
</tr>
<tr>
<td>10 nsec</td>
<td>1-8 GB</td>
</tr>
<tr>
<td>10 nsec</td>
<td>1-4 TB</td>
</tr>
</tbody>
</table>

- Use of disks unavoidable (permanence and size)
- Access time is significantly slower for disks

How an OS runs processes

- OS keeps track of which processes are assigned to which part of memory
- To run a new process, read it from disk, assign it to some memory and copy its code there
  - Switch to user mode and start running at the first address of the program
- The OS keeps record of every process
  - State of running process — called the context
  - Keep enough to restart process where it left off

Eventually, a trap happens...

- Either because the running process issued it or a device did (clock tick)
- OS records the state of current process’ context
- Service the trap
  - i.e. send something to the printer
- Pick a user process to start again
  - Maybe the last one, maybe not
  - The picking is the interesting part — known as scheduling

Brief History of Operating Systems

- First generation 1945 - 1955
  - vacuum tubes, plug boards
- Second generation 1955 - 1965
  - transistors, batch systems
- Third generation 1965 – 1980
  - ICs and multiprogramming
- Fourth generation 1980 – present
  - personal computers
1940’s - First Computers

- Computer dedicated to one user/programmer at a time.
  - Program loaded manually by programmer, using console switches.
  - Debugging using console lights.
- Expensive machine idle most of time, because people are slow.
- Each program must include code to operate peripherals - error prone, device dependencies.
- Libraries of subroutines to drive peripherals are example of typical OS service.

1950’s – Batch Processing

- User/programmer submits a deck of cards that describes a job to be executed.
- Jobs submitted by various users are sequenced automatically by a resident monitor.
- Tape drives available for batching of input and spooling of output.
- Computer system is kept busier.
- No longer interactive; longer turnaround time.
- CPU is still idle for I/O-bound jobs.

Typical Batch System

Early batch system
  - bring cards to 1401
  - read cards to tape
  - put tape on 7094 which does computing
  - put tape on 1401 which prints output

Structure of a Typical Batch Job

Multiprogramming System with 3 Jobs in Memory

1960’s – Multiprogramming/Timesharing

- The advent of the I/O processor made simultaneous I/O and CPU processing possible.
- CPU is multiplexed (shared) among a number of jobs.
- Interactiveness is restored.
- Hardware and OS requirements become significantly more complex.
  - Most OS issues arise from trying to support timesharing – CPU scheduling, deadlock, protection, memory management, etc
- CTSS (Compatible Time Sharing System), Multics
1970's - Minicomputers & Microprocessors

- Personal computers, rather than a single mainframe.
- Early minicomputers and microprocessors were small, so there was some regression to earlier OS ideas.
  - e.g. DOS on PC is still essentially a batch system similar to those used in 1960, with some modern OS ideas thrown in (e.g., hierarchical file system).
- This trend changing rapidly because of powerful new microprocessors.
- The user interface (GUI) became more important.
- UNIX, DOS

1980's - Networking

- Powerful workstations (e.g., PDP, VAX, Sun work stations, etc.)
- Local area networks (e.g., Ethernet, Token ring) and long-distance network (Arpanet)
- Networks organized with clients and servers
- Decentralization of computing requires more communication (e.g., resource sharing)
- OS issues – network communication protocols, data encryption, security, reliability, consistency of distributed data
- Real-Time Systems – timing constraints, deadlines, QoS (quality of service)

1990's and Beyond

- Parallel Computing (tera-flops)
- Powerful PCs, Multimedia computers
- High-speed, long-distance communication links to send large amounts of data, including graphical, audio and video
- World Wide Web
- Electronic notebooks and PDAs using wireless communication technologies
- Embedded computers: medical devices, cars, smartphones
- OS issues – Large heterogeneous systems, mobile computing, utilization of power, security, etc.