Operating Systems

Chapter 4

Functions of Operating Systems

• Oversee operation of computer
• Store and retrieve files
• Schedule programs for execution
• Coordinate the execution of programs
• Provide an interface to the user to access machine functions
Evolution of Shared Computing

- Manual jobs
- Batch processing
- Interactive processing
  - Requires real-time processing
- Time-sharing/Multitasking
  - Implemented by Multiprogramming
- Multiprocessor machines

Types of Software

- Application software
  - Performs specific tasks for users
- System software
  - Provides infrastructure for application software
  - Consists of operating system and utility software
Software classification

Operating System Components

- **Shell**: Communicates with users
  - Text based
  - Graphical user interface (GUI)
- **Kernel**: Performs basic required functions
  - File manager
    - Where files are on the disk, clusters
  - Device drivers
    - Interface with physical devices
  - Memory manager
  - Scheduler and dispatcher
Memory Manager

• Allocates space in main memory
• May create the illusion that the machine has more memory than it actually does (virtual memory) by playing a “shell game” in which blocks of data (pages) are shifted back and forth between main memory and mass storage

Getting it Started (Bootstrapping)

• **Bootstrap**: Program in ROM (example of firmware)
  – Run by the CPU when power is turned on
  – Transfers operating system from mass storage to main memory
  – Executes jump to operating system
The booting process

BIOS – Basic I/O System – software utilities for fundamental I/O activities stored on the ROM along with the bootstrap program

Processes

- **Process**: The activity of executing a program
- **Process State**: Current status of the activity
  - Program counter
  - General purpose registers
  - Related portion of main memory
Process Administration

- **Scheduler**: Adds new processes to the process table and removes completed processes from the process table.
- **Dispatcher**: Controls the allocation of time slices to the processes in the process table.
  - The end of a time slice is signaled by an interrupt.

Time-sharing between process A and process B

See Windows Task Manager for an example
Race Condition

• When two processes want to use a common shared resource a “race condition” may result and cause undesirable results

• Example: Two processes writing to the same location in memory (one to subtract 10, one to add 20)

<table>
<thead>
<tr>
<th>Process 1</th>
<th>Process 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load value from memory to register</td>
<td>Load value from memory to register</td>
</tr>
<tr>
<td>Add 20</td>
<td>Subtract 10</td>
</tr>
<tr>
<td>Store register back to memory</td>
<td>Store register back to memory</td>
</tr>
</tbody>
</table>

Attempt to fix: use register 0 as “in use” flag

<table>
<thead>
<tr>
<th>Process 1</th>
<th>Process 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>If register 0 is 0</td>
<td>If register 0 is 0</td>
</tr>
<tr>
<td>Set register 0 to 1</td>
<td>Set register 0 to 1</td>
</tr>
<tr>
<td>Load value from memory to register</td>
<td>Load value from memory to register</td>
</tr>
<tr>
<td>Add 20</td>
<td>Subtract 10</td>
</tr>
<tr>
<td>Store register back to memory</td>
<td>Store register back to memory</td>
</tr>
<tr>
<td>Set register 0 to 0</td>
<td>Set register 0 to 0</td>
</tr>
<tr>
<td>Else</td>
<td>Else</td>
</tr>
<tr>
<td>wait until register 0 is 0</td>
<td>wait until register 0 is 0</td>
</tr>
</tbody>
</table>

Will this fix the problem?
Handling Competition for Resources

- **Semaphore**: A “control flag”
- **Critical Region**: A group of instructions that should be executed by only one process at a time
- **Mutual exclusion**: Requirement for proper implementation of a critical region

Solution: Semaphore

In this example we use an uninterruptible Test and Set Instruction

Process 1

If register 0 is 0 set register 0 to 1
Load value from memory to register
Add 20
Store register back to memory
Set register 0 to 0
Else
wait until register 0 is 0

Process 2

If register 0 is 0 set register 0 to 1
Load value from memory to register
Subtract 10
Store register back to memory
Set register 0 to 0
Else
wait until register 0 is 0
Deadlock

• Processes block each other from continuing
• Conditions required for deadlock
  1. Competition for non-shareable resources
  2. Resources requested on a partial basis
  3. An allocated resource cannot be forcibly retrieved

A deadlock resulting from competition for non-shareable railroad intersections
Security

• Attacks from outside
  – Problems
    • System errors
    • Insecure passwords
    • Sniffing software
  – Counter measures
    • Auditing software
    • Firewalls, scanners

Security (continued)

• Attacks from within
  – Problem: System errors
  – Counter measures: patches, virtual machine

  – Problem: Unruly processes
  – Counter measures: Control process activities via privileged modes and privileged instructions