CS341 – Operating Systems

Instructor: Christine Bassem
cbassem (at) wellesley (dot) edu
Office: Science Center E122

What’s an operating system?

- A program that controls the execution of application programs
- An interface between applications and hardware

Main objectives of an OS:
- convenience
- efficiency
- ability to evolve

Evolution of Operating Systems

- A major OS will evolve over time for a number of reasons:
  - hardware upgrades
  - new types of hardware
  - new services
  - Fixes

Figure 2.1 Computer Hardware and Software Structure
Operating System Services

• Program development
• Program execution
• Access I/O devices
• Controlled access to files
• System access
• Error detection and response
• Accounting

Key Interfaces

• Instruction set architecture (ISA)
• Application binary interface (ABI)
• Application programming interface (API)

The Role of an OS

• A computer is a set of resources for the movement, storage, and processing of data

• The OS is responsible for managing these resources

Operating System as Software

• Functions in the same way as ordinary computer software

• Program, or suite of programs, executed by the processor

• Frequently relinquishes control and must depend on the processor to allow it to regain control
In CS341, you will learn about the foundational concepts of operating systems.

By the end of this course, you will be able to:
- Summarize the key functions of an operating system
- Understand the difference between process and thread, and their management within an OS
- Summarize and discuss the basic concepts of concurrency and deadlocks
- Discuss the main concepts in memory management and file systems
- And many more...

All while discussing the current implementations in Windows and Linux

What is this course about?

Administrivia

- Lectures:
  - Tuesdays and Fridays at 11:10am

- Office hours:
  - Tuesday 2:45pm - 4:30pm
  - Wednesday 11am - 2pm
  - Or by appointment

Textbook

Assignments

- Schedule
  - Posted on Friday
  - Usually due two weeks later

- All assignments will be submitted to the course GitHub page

- Programming problems
  - In most assignments, you can use the language of your choice, but I recommend using Java

- Non-programming problems
  - Write your solutions in latex, try sharelatex.com
    - You will be provided with a LaTeX template every assignment
  - Bring hard copy of the solution to class on Friday

More ...

- You’ll have three exams during the semester,
  - 1st exam on Feb 23
  - 2nd exam on Mar 20
  - 3rd exam on Apr 24

- All exams are in-class and open book
- There will also be a final exam.

Grading scheme

- Course grades will be distributed as follows:
  - Programming assignments: 25%
  - Written assignments: 15%
  - In-class exams: 30%
  - Final exam: 20%
  - Participation: 10%

Collaboration in CS341

- Starting from the second week, you will start working in groups of (4)

- With your group, you will,
  - Work on class exercises
  - Discuss ideas to solve assignment problems

- In assignments,
  - Work in pairs on programming problems
    - Partner must be from the same group
  - Work individually on non-programming problems
Brief history of Operating Systems

Evolution of Operating Systems

- Stages include:

Serial Processing

Earliest Computers
- No operating system
- Computers ran from a console with display lights, toggle switches, some form of input device, and a printer
- Users have access to the computer in "series"

Problems
- Scheduling
  - Most installations used a hardcopy sign-up sheet to reserve computer time
- Setup time
  - A considerable amount of time was spent just on setting up the program to run

Simple Batch Systems

- Early computers were very expensive
  - Important to maximize processor utilization
- Monitor
  - User no longer has direct access to processor
  - Job is submitted to computer operator who batches them together and places them on an input device
  - Program branches back to the monitor when finished
Within a batch system

Monitor Point of View
- Monitor controls the sequence of events
- Resident Monitor is software always in memory
- Monitor reads in job and gives control to it
- Job returns control to monitor

Processor Point of View
- Processor executes instructions from the memory
- "Control is passed to a job" means processor is fetching and executing instructions in a user program
- "Control is returned to the monitor" means that the processor is fetching and executing instructions from the monitor program

Figure 3.3 Memory Layout for a Resident Monitor

Desirable Hardware Features

Memory Protection for Monitor
- While the user program is executing, it must not alter the memory area containing the monitor

Timer
- Prevents a job from monopolizing the system

Privileged Instructions
- Can only be executed by the monitor

Interrupts
- Gives OS more flexibility in controlling user programs

Modes of Operation

User Mode
- User program executes in user mode
- Certain areas of memory are protected from user access
- Certain instructions may not be executed

Kernel Mode
- Monitor executes in kernel mode
- Privileged instructions may be executed
- Protected areas of memory may be accessed

Simple Batch System Overhead

- Processor time alternates between execution of user programs and execution of the monitor

- Sacrifices:
  - Some main memory is now given over to the monitor
  - Some processor time is consumed by the monitor

- Despite overhead, the simple batch system improves utilization of the computer
Uni-programmed Batch Systems

- Processor is often idle

<table>
<thead>
<tr>
<th>Action</th>
<th>Time (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read one record from file</td>
<td>15</td>
</tr>
<tr>
<td>Execute 100 instructions</td>
<td>1</td>
</tr>
<tr>
<td>Write one record to file</td>
<td>22</td>
</tr>
<tr>
<td>TOTAL</td>
<td>38</td>
</tr>
</tbody>
</table>

Percent CPU Utilization = \( \frac{38}{38} = 0.032 = 3.2\% \)

Figure 2.4 System Utilization Example

Multi-programming

- Also known as multitasking
- Memory is expanded to hold three, four, or more programs and switch among all of them

<table>
<thead>
<tr>
<th>Program</th>
<th>Run</th>
<th>Wait</th>
<th>Run</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combined</td>
<td>Run</td>
<td>Wait</td>
<td>Run</td>
<td>Wait</td>
</tr>
</tbody>
</table>

Multi-programmed Batch Systems

- There must be enough memory to hold the OS (resident monitor) and one user program
- When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O