CS 31: Intro to Systems Course Recap

Martin Gagne
Swarthmore College
April 27, 2017
Buffer Overflow

Two varieties:

• **Buffer Overread**
  – use buffer overflow to read more memory than should be available

• **Buffer Overwrite**
  – use buffer overflow to write to places you shouldn’t normally be allowed (where?)
Course Survey

• >66 % have responded

• Thank you to those who have responded

• Extra quiz/participation grade will be dropped if we reach 90% (need 11 more to answer)
Final Exam

• Friday, May 5, 7:00 PM. SCI 199

• Similar format to the midterm

• You get ~100% more time
• Exam is ~33% longer

• between \( \frac{1}{2} \) and \( \frac{2}{3} \) post-midterm material
Is any of these days completely impossible for you?
Review Session

A. Monday
B. Tuesday
C. Wednesday
D. Thursday (morning / early afternoon)
E. Don’t care and/or don’t plan to attend

Which one of these days would you prefer?
Is any of these times completely impossible for you?
Which one of these times would you prefer?

A. early morning
B. late morning
C. early afternoon
D. late afternoon
E. evening
Course Recap

• This course was a vertical slice of computer
  – From lowest level: simple logic
  – To high level: large, complex programs run on OS

• Big goal: make complex machine easier to use
  – Hide details with the right abstractions
  – Improve performance when possible
Lowest Level

• Storing and representing data
  – 2’s complement integers, floating point, etc.
  – Arithmetic using bits

• Logic gates: simple hardware
Hardware Abstraction: Circuits

• Combining gates to build specific circuits
  – arithmetic (adders, ALUs)
  – storage (latches, registers)
  – control (decoder, multiplexer)
CPU

• Combine circuits to create a CPU
  – Periodic clock: fetch, decode, execute instructions
Instruction Set Architecture

- ISA defines CPU / software interaction
  - Machine properties (# registers, address modes)
  - Method for controlling hardware (assembly lang)

\[ x = y \gg 3 \mid x \times 8 \]

```
movl -8(%ebp), %eax  # R[%eax] ← x
imull $8, %eax       # R[%eax] ← x*8
movl -12(%ebp), %edx # R[%edx] ← y
rshl $3, %edx        # R[%edx] ← y >> 3
orl  %eax, %edx      # R[%edx] ← y >> 3 \mid x*8
movl %edx, -8(%ebp)  # M[R[%ebp-8]] ← result
```
Conventions

• Agreed upon system for using ISA
  – e.g., manipulating the stack, register meaning

Callee’s frame.

Callee’s local variables.

Caller’s Frame Pointer

Return Address

First Argument to Callee

…

Final Argument to Callee

Caller’s local variables.

…

Older stack frames.

…

Caller’s frame.
Storage and Memory

• Allocating memory (stack vs. heap)

• Pointers
  – malloc() / free()
  – address of (&)
  – dereferencing
  – arrays, 2D arrays
The Memory Hierarchy

- Registers: 1 cycle to access
- Cache(s) (SRAM): ~10’s of cycles to access
- Main memory (DRAM): ~100 cycles to access
- Flash SSD / Local network
- Local secondary storage (disk): ~100 M cycles to access (slower than local disk to access)
- Remote secondary storage (tapes, Web servers / Internet)

- Smaller
- Faster
- Costlier per byte

- Larger
- Slower
- Cheaper per byte

CPU instructions can directly access

On Chip Storage
Caching

• Improve performance by keeping a small memory for frequently-used data
  – Many parameters inform address division (tag, idx)
    • direct map vs. associative
    • block size

• Exploit major idea: **Locality**
  – temporal / spatial
Operating System

• Software supports: making programs easy/fast

• Three major abstractions:
  1. Process
  2. Thread
  3. Virtual memory
Processes

• Program in execution
  – fork() / exit() to create / terminate

• Represents all of the resources of a task
  – virtual address space (process memory) with associated page table
  – open files
  – process ID, other accounting info

• One or more threads of execution
Virtual Memory

• Allow processes to behave as if they have the entire memory of the machine
• Translate from virtual (fantasy) address to physical
Virtual Memory

• Each virtual page is allocated only when necessary
• The page table keeps track of whether a virtual page is held in memory (which frame), on disk, or if it has never been referenced before
• One such page table is needed for each process
Virtual Memory

- Use disk to store data that hasn’t been used lately
  - (Another instance of exploiting locality)
Mechanism & Policy

• Mechanism: the method to do something

• Policy: rules for governing the mechanism(s)

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context switching</td>
<td>CPU scheduling</td>
</tr>
<tr>
<td>Cache eviction</td>
<td>Cache replacement policy</td>
</tr>
<tr>
<td>VM paging to disk</td>
<td>Page replacement policy</td>
</tr>
</tbody>
</table>

• “Best” policy usually varies by workload!
Concurrency & Parallelism

• Single CPU core performance has plateaued
  – Hardware giving us more CPU cores instead

• Programmer’s responsibility to use them!

• Big opportunity for performance benefits!
Threads

- Execution context within a process
- Independently scheduled
Multi-threading in Practice (pthreads)

• Not always intuitive to reason about...

• Potential problems
  – race conditions
  – deadlock
  – priority inversion, etc.

• Requires careful synchronization
  – mutex, barrier, semaphore, cond_variable
Systems Courses

• OS (probably next spring… maybe…)
• Networks (this fall)
• Compilers (right now)
• Parallel & Distributed (probably next spring)
• Databases (taught last semester)
• Architecture (Eng)
Final Exam

• Friday, May 5, 7:00 PM.  SCI 199

• Similar format to the midterm

• You get ~100% more time
• Exam is ~33% longer

• between ½ and ⅔ post-midterm material
Questions?

• Thank you for a great semester!