CSE 220: Systems Programming

Final Review

Ethan Blanton

Department of Computer Science and Engineering
University at Buffalo

The Compiler and Toolchain

- The “C compiler” is actually a chain of tools
  - We invoke the compiler driver
  - The preprocessor transforms the source code
  - The compiler turns C into assembly language
  - The assembler turns assembly language into machine code in object files
  - The linker links object files into an executable

Compiler Optimization

- Optimizing compilers must be correct first, then efficient
- Write compiler-friendly code
  - Avoid optimization-blockers like function calls and references to non-local memory
- Compiler optimization is based on static information
- Tune innermost loops first
- Common code optimizations:
  - Code motion
  - Reduction in strength
  - Reuse of subexpressions
  - Loop unrolling for superscalar processors
- Superscalar processors can work on more than one instruction per clock cycle

Dynamic Memory Allocation (1)

- Dynamic memory allocators provide memory management tools to the programmer at run time
- This memory is the heap
- Heap allocators optimize for performance:
  - Throughput
  - Overhead
- Overhead comes in the form of internal and external fragmentation
- Placement policies can trade off throughput for overhead
- Explicit free lists provide higher throughput by using data structures to track available heap blocks
  - Particularly when the heap is very large or almost full
- More complex data structures (segregated or sorted free lists, *e.g.*) can make certain operations faster

Caching

- The speed gap between CPU, memory and mass storage continues to widen
- Well-written programs exhibit a property called locality
- Memory hierarchies based on caching close the gap by exploiting locality
- Each level of the hierarchy is much smaller and much more expensive per byte than the one below it
- Cache misses occur when cache is too small, locality is violated, or implementation has limitations
- Flash memory progress outpacing all other memory and storage technologies (DRAM, SRAM, magnetic disk)

Virtual Memory

- Virtual memory:
  - uses a memory management unit
  - allows the CPU to operate in a virtual address space that may be different from the physical address space
  - the MMU translates virtual addresses to physical addresses
- Paging is a common model for virtual memory.
- Paged systems break both address spaces into pages.
- Pages can be mapped individually between virtual and physical addresses.
- Page tables allow the MMU to translate addresses.
- Page faults bring mapped but unallocated pages into memory.

Processes, Threads, and Concurrency

- Logical control flows are execution steps through programs.
- Concurrency is multiple logical control flows at one time.
- Multiprocessing versus Multitasking
- Processes versus Threads

Races and Synchronization

- A race is a situation where program correctness depends on the order of operations in concurrent flows.
- Data races are races involving modification of data.
- Synchronization is the deliberate ordering of events in a program.
- A critical section is a region of code that must be accessed by at most one concurrent flow at a time.
- Progress graphs visualize concurrent flows.
- Synchronization primitives:
  - Atomic operations
  - Mutexes
  - Semaphores
  - Condition variables
- Deadlock is a program error caused by synchronization.

POSIX Threads and Synchronization

- The POSIX threads (pthreads) API provides a thread abstraction on Unix
- POSIX provides many synchronization primitives:
  - Mutexes
  - Semaphores
  - Condition variables
  - Thread joining
- CS:APP covers semaphores in detail

System I/O

- POSIX files are a sequence of bytes
- Devices on POSIX systems are represented as special files
- Unix I/O exposes files and devices via a simple interface:
  - `open()`, `close`
  - `read()`, `write()`
  - `lseek()`
- Unix I/O is very fast, but rather primitive
  - Small reads are just as expensive as large reads
  - No concept of records/lines/*etc.*
- Open files are represented by file descriptors which represent file table entries in the kernel
- Standard I/O provides buffered access to Unix I/O
- POSIX makes no distinction between file types, but standard I/O provides certain facilities for text files

Exceptions

- Exceptions are a transfer of control to the OS kernel in response to some event
- Interrupts, or asynchronous exceptions, come from external to the processor
- Synchronous exceptions are caused by processor operations
- System calls are traps (a form of synchronous exception) into the kernel
- Page faults are faults that cause the kernel to either fix up a memory access or notify the process of error
- Signals are pure software exceptions

Final Exam

The final exam is Wednesday, December 11 at 3:30 PM

Locations:

- Section A: Davis 101
- Section B: Knox 109

Bring your student ID and a writing implement

Closed book, closed notes, closed neighbor

Expect a format similar to the midterm, but longer

License

To retrieve a copy of this material, or related materials, see https://www.cse.buffalo.edu/~eblanton/.