Main learning outcomes for CST334

or: what someone who's taken operating systems should know

Glenn Bruns
CSUMB

Map of this course

process management

memory management

concurrency

file management

language processing

bash

C, system calls

awk

sed

C, pthreads

Guiding questions

- What are the virtual resources or services offered to users? Are they easy to use?
- How to ensure fair sharing of resources between users?
- How to protect users from each other, and protect the system from users?
- What workloads do we use to measure performance?
- What metrics do we use to measure performance?
- How efficiently are resources managed?

Process Management

- Simulate the mechanics of context switching
  - including the use of interrupts, traps, the interrupt table, the trap table
  - don't forget user/kernel modes of the CPU
- Be able to compute avg. turnaround time and avg. response time for scheduling algorithms
  - first-come first-served, shortest job first, shortest time to completion first, round robin, MLFQ
- Write code to that uses the Linux process API
  - like we did with MSH

Memory Management

- Define the regions of process memory and their purpose (stack, heap, etc.)
- For memory management schemes:
  - base-and-bounds, segmentation, paging, multi-level paging
  - define structure of a virtual address
  - be able to translate virtual to physical address
- Explain and implement caching
  - including cache replacement policies
  - LRU, random, optimal, FIFO
  - compute average memory access time
- Simulate memory allocation with a free list
- Write code that uses the Linux memory API

Concurrency

- Explain threads and how they relate to processes
- Define critical section, mutual exclusion, critical section, execution path, deadlock, etc.
- Simulate operation of synchronization primitives: locks, condition variables, semaphores
- Write pthreads code in Anderson/Dahlin style
- Write concurrent programs using semaphores in "Little Book of Semaphores" style

Syntax and parsing

- Given a description of a language, write a BNF grammar for it
- Given a BNF grammar, say whether a string can be derived from it
- Given a BNF grammar, transform the grammar in the form needed for predictive parsing
- Given a BNF grammar in the right form, write a predictive parser that will accept only string in the language of the grammar

File management

- Simulate the interaction of the CPU with a device
- Given disk drive specs, compute:
  - avg. rotational delay
  - access time
  - time to perform a set of random or sequential accesses
- Define file system structure
  - very simple file system; inode, superblock, inode table, indirect pointer, etc.
- Simulate file system operations
  - read file given path, list directory contents, etc.

Command line

- read and write common bash commands
- read and write bash commands to set file and directory permissions
- write bash scripts
  - command-line arguments, exit status, control flow
- write basic awk code at command line or in scripts
- write basic regular expressions
- write grep, sed, and find commands
- write simple Make files