CSE 333 – SECTION 10

Concurrency Wrap-Up & Final Review

Concurrency Wrapup: fork() example

- Primary use patterns of fork():
  - Server handling connections (lecture example)
  - Shell executing programs (this example)

Demo: hpsh.c

The Rest of C++

- Templates – template definitions vs instantiation
- STL, containers and iterators.
- vector, list, map
  - Smart pointers, using with STL.
  - unique_ptr (cannot be copied, but can move ownership to another)
  - shared_ptr (reference counting)
  - weak_ptr (using to break cycles and why this is needed)
- Subclasses, inheritance, virtual functions, dynamic vs static dispatch (function calls), vtables, constructors and destructors in subclasses
- Pure virtual functions and abstract classes (what they are)
- Using class hierarchies with STL and smart pointers, assignment slicing, value vs pointer semantics
- C++ casts

Network Programming

- Basic network layers: physical, data link, IP, TCP, application
- Packets, and packet encapsulation across layers
- IP addresses, address families (IPv4, IPv6), DNS, ports
- Stream sockets, file descriptors, read, write
- Client steps:
  - address resolution, create socket, connect socket to server, read/write (including retries), close
- Server steps:
  - determine address and port, create socket, bind socket to address/port, listen (and how the OS queues pending connections), accept connection, read/write, close
- Very basic HTTP and HTML

Concurrency

- Multiple processes and fork() (mostly CSE351 review), shared file descriptors and close() in forked processes
- Threads - concurrent execution inside a single process; know a few of the pthread basics (i.e., what it means to create a thread and start execution in a function)
- Use of concurrency to improve throughput and resource utilization

The Pre-Midterm Topics

- Don’t forget the first half of the class, the final is cumulative!
- Brief summary:
- C programming, pointers and memory management
- POSIX libraries
  - read, write, open, close, etc.
- The beginning of C++
  - classes/modularity, references, operator overloading, and other differences from C