CSE 333 – SECTION 10
Final Review
Administrivia

- Final exam March 15\textsuperscript{th} (Wednesday)
- 80 mins(subject to change)

- Hw6 in-person grading will be held on Monday & Tuesday
Overview

• C Programming, tools, and workflow
  • Compilation, linking, runtime
• Memory, types, conversions
• System interfaces and services (file I/O, etc.)
• C++ :
  • “better C” + classes + STL + smart pointers + ...
• Networking basics: TCP/IP, sockets, ...
The C/C++ Ecosystem

- System layers: C/C++, libraries, operating system
- Building programs
  - `cpp`: `#include`, `#ifndef`, and all that
  - compiler: `source → .o`
  - loader (ld): `.o + libraries → executable`
- GCC: The C multitool
- Make and related tools to automate the process
What’s a process?

A process is basically a program in execution.

Address space

Thread(s) of execution

Environment (arguments, open files, ...)

When a program is loaded into the memory and it becomes a process, it can be divided into four sections — stack, heap, text and data.
• How to clone processes?
• Fork() - What is actually inherited?
  • - new pid
  • - file descriptors are the same
  • - Context is the same
  • - VM is copied
• FIFO, pipe, etc
• Threads: concurrent execution inside a single process; know a few of the pthread basics (how to create a thread and start execution in a function)
• Processes vs. threads
  • Threads are used for small tasks, whereas processes are used for more ‘heavyweight’ tasks – basically the execution of applications.
  • Another typical difference is that threads (of the same process) run in a shared memory space, while processes run in separate memory spaces.
Processes vs threads on one slide

Stack pointer:
- OS kernel [protected]
  - stack
  - shared libraries
  - heap (malloc/free)
  - read/write segment
    .data, .bss
  - read-only segment
    .text, .rodata

Program Counter:
- parent
- fork()
- child

Stack pointer:
- OS kernel [protected]
  - stack
  - shared libraries
  - heap (malloc/free)
  - read/write segment
    .data, .bss
  - read-only segment
    .text, .rodata

Program Counter:
- PC
- PCparent
- PCchild

Stack:
- SP
- SPparent
- SPchild

Heap:
- heap (malloc/free)
C language

• Structure of C programs
  • Header files and implementations; declaration vs definition
  • Internal vs external linkage (extern/static)
  • Standard types and operators (scalars including things like uint64_t, structs, arrays, typedef, etc. E.g. structs – how to define and use, meaning of p->x ( = (*p).x )
  • Functions: defining, using, execution model
  • Standard libraries and data structures (strings, streams, ...)
    • C standard library, system calls, and how they are connected
  • Handling errors in a language without exception handling
    • return codes, errno, and friends
Memory

- Object scope and lifetime (static, automatic, dynamic)
- Pointers and associated operators ( &, *, ->, [ ] )
- Dynamic memory allocation (malloc/free; new/delete)
  - Who is responsible for dynamic memory & what happens if not done right (dangling pointers, memory leaks, ...)
- Tools: debuggers (gdb), monitors (valgrind), ...
- Most important tool: logics & thinking(!)
File I/O

- fread, fwrite, fopen, fclose, fflush, etc.
- read, write, open, close, etc.
- buffer
C++ (and C++11)

• A “better C”
  - Type-safe streams and memory management (new, delete, delete[], etc.
• References and const
• Classes (and objects)
  - Constructors, copy constructor, move constructor, destructor, assignment
• Subclasses and inheritance
  - Dynamic vs static dispatch, virtual functions, vtables
• Pure virtual functions and abstract classes
• C++ casts - static_cast, dynamic_cast, const_cast, reinterpret_cast
C++ (and C++11)

- Templates: parameterized classes and functions
  - How C++ implements templates
  - *How the idea is similar to Java generics and what’s different
- 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 (used to break cycles)
Network Programming

- Basic network layers: physical, data link, IP, TCP, application
  - Particularly IP and TCP
  - What they do, how they are related, how they differ
- 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
Recources

- Review lecture/section slides, assignments, exercises
- Look at topic list on the web
- Try the Practice Exercises