Introduction

CS3100

Kenneth Sundberg

Operating Systems and Concurrency

- Concurrent Programing
- Operating Systems
  - API
  - Memory Managment
  - Scheduling
  - File Systems

Textbook

- The text for the course is "Operating System Concepts" by Silberschatz, Galvin, and Gagne
- Any version of the book
- Slide references from the ninth edition

Grade Break Down

- Programming Assignments - 60%
- Exams - 30%
- Quizzes - 10%

Late Policy

- 10 % per day late
- No work accepted after 3 days
- No exceptions - even if life happens

ADA

- This course is ADA compliant as required
- Discuss with the DRC for arrangments

Shell

- We will interact primarily through a shell (command line interface)
- Linux is case sensitive

Help

- man < *section* > command
- Displays the manual page for a command
- The optional section number can disambiguate
- 1 - General commands
- 2 - System calls
- 3 - Library functions
- 7 - System specific miscellanea

Navigation

- pwd - print working directory
- ls - list current directory contents
- cd - change directory
- mkdir - make directory

File manipulation

- rm - remove
- mv - move file (also for renaming)
- rmdir - remove directory
- cp - copy file
- ps - process list
- kill - kill a running process

Editors/Viewers

- cat *file* - display file contents
- more - display file one page at a time
- vim - text editor

GCC

- We will use GNU Compiler Collection version 5.4
- g++ *flags sourcefiles*
- O{s,0,1,*2*,3} Optimization level
- g{1,2,*3*} Debug symbol level
- -Wall -Wextra -Werror Turn on most warnings and treat them as errors
- -o Specify output file name
- -c Compile only, no link step

CMake

- Cross-Platform Make
- Generates Unix-Makefiles
- Generates Visual Studio Solutions
- Many others also

Overview

- Many programming tasks rely on a set of common building blocks
- Rewriting these blocks is a poor idea
- Language proficiency includes knowledge of standard libraries

Containers

- Abstraction of a ‘collection’
- Many types, but two main ones
  - vector
  - map

vector

- Default container
  - Consider other options as optimizations
- Perfer push_back
- reserve() space if able

map

- Default associative container
- Maps a key type to a value type
- Provides $O(logn)$ insertion and deletion
  - Implemented as a balanced binary tree

others

- Associative containers
  - set
  - unordered_map
- Non-associative containers
  - list
  - deque
  - priority_queue
  - array

Iterators

- Iterators are an important *internal* abstraction
- Iterators are pointers
  - Many implementations of iterators are just typedefed pointers
  - Similar cautions apply
- Use standard algorithims

Overview

- Algorithims represent actions on containers
- Often can be customized through a functor parameter

Functors

- Any object that can be called like a function
- OO version of a function pointer
- Examples:
  - An object with operator()
  - A bind expression
  - A lambda expression
  - A function pointer

Lambda Expressions

- A λ function is an anonymous function
- They should be very short (2 functions calls and a conjunction)
- Lambdas have an ineffable type
- Naming things is an important part of the programming process - don't use λ functions just to avoid this step

Lambda Syntax

- Capture List
- Parameter List
- Return Type
- Qualifier List
- Body

A Few Useful Standard Algorithims

- sort
- for_each
- remove_if
- find_if
- count_if
- copy_if
- rotate
- min_element
- max_element
- transform
- accumulate

Defining Algorithims

- Template functions
- A template parameter is treated like a function
- Any callable object (functor) can be passed to this parameter

shared_ptr

- Reference counted smart pointer
- Thread safe
  - Thread safty of the object pointed to is still your responsibility
- Last reference calls destructor

unique_ptr

- Smart pointer
- Movable but not copyable
- Zero overhead compared to raw pointer

new and delete are radioactive

- Never call new or delete directly
- use make_shared or make_unique instead

Raw pointers

- Raw pointers are still fine for non-owning pointers

optional

- Available in C++17
- Available in BOOST
- Represents a variable that may not be there

RAII

- Resource Aquisition Is Initialization
- Possibly the most important resource pattern

The Rule of Five or None

- Consequence of RAII
- Resource wrappers need:
  - Destructors
  - Copy Constructors
  - Copy Assignment Operators
  - Move Constructors
  - Move Assignment Operators
- Non resource wrappers:
  - Should use the compiler provided default implementations

AAA Style

- Almost Always Auto
- Declare variables using the auto keyword
  - Prevents casts
  - Prevents uninitalized variables
  - Supports refactoring
- There are some exceptions

Prefer Algorithims to Loops

- Algorithims encode higher level semantics
- They are already debugged
- They are very efficient
  - Can use implementation details

Textbook sections covered:

- Section 01-05 (frame 2)
- Section 01-06 (frame 2)
- Section 01-07 (frame 2)