C++ Multithreading - Hybrid

Fall 2021

CSC 362

instructor: Ed Keenan
email: ekeenan2@cdm.depaul.edu

office hours: Remote: Thurs 3-4:30 pm Synchronous Zoom Meeting or by appointment
office: CDM 830
phone: (312)362-6747
website: piazza.com/depaul/fall2021/csc461361 (Preferred communication)
lecture: In-Person: CDM 224 Thursday 5:45-9:00pm

- Week 1, 6, Final

REMOTE: Thursday 5:45-9:00pm Synchronous Zoom Meeting

- Week 2,3,4,5,7,8,9,10

Desired to Learn (D2L): d2l.depaul.edu (Grades, Viewing lectures, Announcements)
Version Control: perforce: ***perforce.dpu.depaul.edu:1666***
All Communication: Piazza for all communication, reply within 24 hrs during school week

Description

Software architecture of applied C++ concurrency and multithreading fundamentals. Basics threading concepts: process model, threads, stacks, fibers, mutexes, semaphores, atomics and events. Leveraging advanced C++ language features relating to the memory model and the threading support in large multithreaded architectures. Architecting lock-based and lock-free concurrent data structures in applications. Designing a threaded management system to control the access and reuse of threads in applications. Designing multithread architecture for real-time performance.

Prerequisites

- GAM 471 - Optimized C++ or CSC 461 Optimized C++
  - Implies CSC 403 & CSC 407

Reference Material

- **C++ Concurrency in Action: Practical Multithreading**, 2012 by Anthony Williams,
  - ISBN: 978-1933988771
- **Windows System Programming** (4th Edition), 2010, Johnson Hart,
  - ISBN: 978-0321657749
- ***The C++ Programming Language:*** Stroustrup
  - ***4th Edition 2013 (new edition) or 3rd Edition (either are acceptable)***
  - Stroustrup Addison-Wesley Longman/Pearson, 2014. ISBN: 978-0321563842

Learning Objectives

- Students will be able to demonstrate basic thread knowledge (threads, mutexes, semaphores, atomics, events)
- Students will be able to optimally design applications to minimize stalls due to thread waits and dependencies.
- Students will be able to design a thread management system to control the access and lifecycles of threads in an application.
- Students will design multithreaded applications to improve its run-time performance and resource usage.

Grading

- 15% - Basics assignments
  - 6 basics threading assignments
    - (2.5%) each basics
- 70% - Programming Assignments
  - 30 % - Simplified Threading program – Jetsons
    - Program using basic multithreading primitives
      - Several fixed threads, Mutexes, Callbacks, Critical Sections
    - (10%) File <-> Coordinator thread
    - (10%) Waveout <-> Playback thread
    - (10%) Complete systems with greater 23 threads
  - 40 % - Conversion problem - Maze
    - Puzzle program or something similar
      - Converting a single thread system into multiple threaded solution
    - (10%) Bottom up Solver
    - (10%) Two Thread Solver
    - (10%) Beat 4 – 10K x 10K mazes
    - (10%) Uber Maze 30K x 30K – Extra Credit (under grad not required)
- 15 % - Final Exam In-Class
  - Finals week Thursday

Software

- ***Microsoft Visual Studio 2019 Enterprise Edition (not Community)***
  - MSDNAA Depaul – Visual Studio 2019 Enterprise
    - C++ and C# install (future classes)
  - Any other variants are not used in this class
  - Students are responsible keeping their development tools working
- ***Perforce Server***
  - Download and configuration instructions will be provided in class
  - ***Perforce – Helix Visual Client (p4v)***
    - https://www.perforce.com/downloads/helix-visual-client-p4v
    - Server address: ***perforce.dpu.depaul.edu:1666***

Readings

Textbooks are used for references and learning new topics. It is suggested that you research and investigate material and ancillary topics covered in the class through these books as needed. High performance programming requires both breadth and depth knowledge in the C++ language, therefore everyone's needs and research will vary based on their own experience and evolving mastery of the material.

Grading Scale:

93-100: A	87-89: B+	77-79: C+	67-69: D+	0-59: F
90-92: A-	83-86: B	73-76: C	60-66: D	
	80-82: B-	70-72: C-		

Topics:

Understanding the multi-threading primitives / basic concepts

- Managing Threads
- Sharing Data between Threads
- Synchronizing concurrent operations
- C++ Memory Model
- Lock-based current data structures
- Lock Free concurrent data structures

Design Software

- Dividing work between threads
- Thread Management
- Debugging
- Testing
- C++ 11 Language Features

Primitives

- Threads
- Fibers
- Atomics
- Mutexes
- Semaphores
- Events

Basics Threading Programs

- There will be at least 6 weekly proficiency problems, validating your threading fundamental knowledge and advanced C++
- The programs are easy to implement, but is a motivator for those who need encouragement to learn more thoroughly the threading material.
- Topics range from:
    - Threads, Mutexes, Futures, Promise

- If you already know the material, the assignments are a way to validate that knowledge. Small assignments to complement material in the class, the lowest of the 6 Basics is drop

  Basics 1 – C++ 11

  - Big six operators
  - Default constructor, copy constructor
  - Assignment operator, destructor
  - Move operator, assignment move operator

  Basics 2 – Creating threads

  - Creating threads and passing arguments
  - Creating threads using Function pointer, Functor, lambdas, methods, bind, etc.

  Basics 3 – Locks

  - Mutex, Move
  - Locks, guards

  Basics 4 – Conditional Variables / Synchronization

  - Conditional variables
  - Signals between threads

  Basics 5 – Thread termination

  - Cleanly control external shutdown of threads
  - Spin lock, variable waiting
  - Trigger with no blocking

  Basics 6 – Parallelization

  - Rework data structures to promote parallelization
  - Changing the process of operations to separate for multiple threads

Programming Assignments

- PA1: Simplified Threading program - Jetsons
  - Program using basic multithreading primitives
    - Several fixed threads, Mutexes, Callbacks, Critical Sections
  - Broken down into 3 parts
    - File <-> Coordinator thread
    - Waveout <-> Playback thread
    - Complete systems with greater 23 threads
  - Awesome fun assignment
- PA2: Conversion problem - Maze
  - Converting a single thread system into multiple threaded solution

    - Broken down into 4 parts
        - Bottom up Solver
        - Two Thread Solver
        - Beat 4 – 10K x 10K mazes
        - Uber Maze 30K x 30K

Tentative Topics schedule:

- Week 1
    - Introduction to Concurrency
    - C++ 11 extensions
- Week 2
    - Launching a thread
    - Waiting for a thread to complete
    - Passing arguments to a thread
    - Transferring ownership of a thread

- Week 3
    - Sharing Data between thread
    - Race conditions
    - Protecting shared data with mutexes
    - Deadlocks
    - Flexible locking (C++ specifics)
    - Transferring mutex ownership
    - Recursive locking
- Week 4
    - Synchronization concurrent operations
    - Waiting for an event or other condition
    - Waveout Demo
- Week 5
    - Waiting for one-off events with futures
    - Return from background tasks, promise - future
    - Future packing as parameters
- Week 6
    - Async calls
    - Clocks, Timepoints
    - Waiting with a time limit
    - Better Thread Termination
- Week 7
    - C++ memory model
    - Reordering of instructions
    - Rules of Elision
- Week 8
    - Atomic Types in C++
    - Synchronizing operations and enforcing ordering using Atomics
    - Fences
    - Maze Demo
- Week 9
    - Lock-based concurrent data structures
    - A thread-safe stack using locks
    - A thread-safe queue using locks and condition variables

    - A thread-safe queue using fine-grained locks and condition variable
    - Writing a thread-safe list using locks
    - Race conditions
- Week 10
    - Performance tuning
    - ABA, Diners, etc problems

Final Exam

Final exam covering the concepts and the material of the class

- In-Person exam
- Multithreading concepts
- Diagramming communication between threads
- Applied problems

Perforce Submissions

- Everyone is expected to submit several submissions to perforce a week.
    - Minimum of ***five*** significant (real) submissions on ***three*** separate days.
    - To promote incremental development and prevent last day rush.
    - Grade deduction will occur if not followed
- The biggest reason students get into trouble with software design:
    - Not starting the project early
    - Not working on the material frequently enough
    - Taking too large of a bite(byte) of the design
- Both are minimized with this *Perforce RULE*
- Even my simplest programs take 10-20 submissions.
    - For these project assignments, my average is 40-400 submissions, so five will be no problem.
- Detailed perforce changelist comments are expected

Piazza Discussion forum

- Previous classes have highly participated in Piazza and did quite well
- Statistics show: students who participate more and help other students do better!
    - The correlation is ridiculous!
        - Poor understanding / poor participation.
        - Great understanding / Great participation
    - As you master the material, help others learn!
        - Want to be a Master programmer so master it!
- Everyone is ***expected*** and encouraged to participate on the Piazza discussion forum. All class-related discussion here this term.
- Everyone is expected to keep up with the material on Piazza and are responsible for its content. Critical class updates and directions will be presented there.
    - Not participating or reading the material on Piazza is ***NOT*** an ***Excuse***.

- All correspondence that is not personal in nature should be vectored through Piazza
    - Sensitive material, use Piazza private note, not email.
- The quicker you begin asking questions on Piazza (rather than via emails), the quicker you will benefit from the collective knowledge of your classmates and instructors. I encourage you to ask questions when you are struggling to understand a concept.
- Keep the forum professional and positive, help each other out.
    - Karma really pays off here.
    - Help each other whenever you can.
        - There will be a time when you will need help from the class (trust me).

NOTE: Do ***NOT*** post until you have watched the entire lecture ***FIRST*** (in class or online)

- This will prevent frustration on all sides (members asking or answering questions)

Collaborating together on programming assignments

- You are encouraged to work together
    - Use the Piazza forums heavy
    - Even share your material with others in the common directory
        - Obviously not the answers
- Everyone is 100% responsible for the work they do.
    - If you get help with a section of code,
    - Please refactor the code the ***snot out of it***
        - Comment and understand that material
        - Transform the code to ***make it yours***
    - Be able to answer ***any*** question regarding the code you commit
- System for Detecting Software Plagiarism
    - We will be using MOSS - Measure of Software Similarity (Stanford University)
        - Indicates possible code infringements (plagiarism)
        - MOSS - will detect the similarity independent of naming convention, indentation style or formatting, it compares abstract syntax tree of your code.
    - I will pursue any plagiarism/integrity violations aggressively, arguing for full expulsion from the university for the offenders.
        - Don't put me or you in this scenario
        - Any integrity violations will result in a failing grade for the course
- If you gain significant support / help from another student or website
    - Fully disclose the support / help you had in a Readme.txt file submitted with your assignments.
        - Disclosing the help, is ***not permission*** for copying the code.
        - Only there to clarify and acknowledge help you were given from a fellow student.
- Modifying any Unit Test or Project setting to alter the outcome results is also an ***Academic Integrity Violation***
- If you are stuck and find yourself even tempted to plagiarize
    - Ask for help!!!!

        - Use on Piazza -> Visit during offices hours, make an appointment
        - ***Don't ever compromise your integrity!***
- Material was uniquely created for this Class.
    - By the process of tuition, you "paid" for the contents and material of this class.
        - Do not share this ***copyrighted*** material in any form
        - It is design for your personal use, while enrolled in the Class.
    - Do ***NOT*** post any content or revealing material to any external website or forum outside of this class.
        - The Class Piazza forum is provided for this service, ask questions there, not on the internet (i.e. StackOverflow and other software forums)
- After you leave this class
    - You are expressly ***FORBIDDEN*** to provide or share the content with others.
    - Academic Integrity Violations can still be applied to students who provide material support to other students even after completion of the class.
- Just follow the golden rule:
    - ***"I have neither given, nor received, nor have I tolerated others' use of unauthorized aid."***

Miscellaneous

- ***Late Policies***
    - Due dates and times are verified by the submission record on the Perforce Server
        - No extensions are allowed
    - All assignments need to compile without warnings
        - Failure to compile “as-is” results in a 0 for the grade
- ***Memory Leaking***
    - For assignments that have memory tracking enabled
        - If an assignment is determined that its leaking memory
            - A deduction of 20% is applied to the grade of that assignment
    - Leaking status is provided during development
- ***Crashing***
    - Assignments are expected to work for a set duration
      (long enough to demo all the features)
        - A grade of 0 is given to any project that throws an exception, ends unexpectedly, crashes or hangs (not proceeding forward).
        - Crash – program locking up or quitting unexpectedly
- ***Integrity Violation***
    - Any form of integrity violation will receive an “F” letter grade for the course, no exceptions
    - All material submitted is from this current offering of class, any material from the outside is considered a violation

Tentative Class Schedule

Week	Lecture	Assignment	Due
1	Overview C++ 11 Review Concurrency	Chapter 1 Basics1 – Big 6 operators	
2	Thread Creation Join, Detatch Passing parameters/scope	Chapter 2 Basics2 – Thread Creation	**Basics1**
3	DeadLocks Mutex, Guards Unique, Move	Chapter 3 Basics3 – Locks	**Basics2**
4	Synchronization Condition Variables Launch/Container	Chapter 4 Basics4 - Conditional Variables **PA1_A File-->Coordinator**	**Basics3**
5	Future/Promise Promise WaveOut Demo	Chapter 4 Basics5 -Thread termination **PA1_B Waveout-->Playback**	**Basics4** **PA1 - part A**
6	Better Termination Async, Queues	Chapter 4 Basics6 - Parallelization **PA1_C Complete**	**Basics5** **PA1 - part B**
7	Memory Model Reordering of instructions Maze Demo	Chapter 5 **PA2_A - MAZE_BottomUp**	**Basics6** **PA1 - Jetsons**
8	Atomics Maze Demo	Chapter 5 **PA2_B -TWO Thread Solution**	**PA2 - part A**
9	Lock Based Structures Maze challenges	**PA2_C - Beat Fixed 10K Maze**	**PA2 - part B**
10	Unique Problems: Diners, ABA,etc.	**PA4_D - UBER Maze**	**PA2 - 10K maze**
Finals	Unique Problems: Diners, ABA,etc.	**Final Exam (in-class)**	**PA2 - UBER maze**

University Dates (Drop, Withdrawal, Audit, Exam)

- https://academics.depaul.edu/calendar/Pages/default.aspx

Course Policies

Changes to Syllabus

This syllabus is subject to change as necessary during the quarter. If a change occurs, it will be thoroughly addressed during class, posted under Announcements in D2L and sent via email.

COVID-19 Health and Safety Precautions

Keeping our DePaul community safe is of utmost importance in the pandemic. Students, faculty and staff are expected to (1) wear a mask as required at all times while indoors on campus; (2) refrain from eating and drinking in classrooms; (3) keep current with their COVID-19 vaccinations or exemptions; (4) stay home if sick; (5) participate in any required COVID-19 testing; (6) complete the online Health and Safety Guidelines for Returning to Campus training; and (7) abide by the City of Chicago Emergency Travel Advisory. By doing these things, we are Taking Care of DePaul, Together. The recommendations may change as local, state, and federal guidelines evolve. Students who do not abide by the mask requirement may be subject to the student conduct process and will be referred to the Dean of Students Office. Students who have a medical reason for not complying with any requirements should register with DePaul's Center for Student with Disabilities (CSD).

Respect for Diversity and Inclusion at DePaul University as aligned with our Vincentian Values

At DePaul, our mission calls us to explore "what must be done" in order to respect the inherent dignity and identity of each human person. We value diversity because it is part of our history, our traditions and our future. We see diversity as an asset and a strength that adds to the richness of classroom learning. In my course, I strive to include diverse authors, perspectives and teaching pedagogies. I also encourage open dialogue and spaces for students to express their unique identities and perspectives. I am open to having difficult conversations and I will strive to create an inclusive classroom that values all perspectives. If at any time, the classroom experience does not live up to this expectation, please feel free to contact me via email or during office hours.

Online Course Evaluations

Evaluations are a way for students to provide valuable feedback regarding their instructor and the course. Detailed feedback will enable the instructor to continuously tailor teaching methods and course content to meet the learning goals of the course and the academic needs of the students. They are a requirement of the course and are key to continue to provide you with the highest quality of teaching. The evaluations are anonymous; the instructor and administration do not track who entered what responses. A program is used to check if the student completed the evaluations, but the evaluation is completely separate from the student's identity. Since 100% participation is our goal, students are sent periodic reminders over three weeks. Students do not receive reminders once they complete the evaluation. Please see https://resources.depaul.edu/teaching-commons/teaching/Pages/online-teaching-evaluations.aspx for additional information.

Academic Integrity and Plagiarism

This course will be subject to the university's academic integrity policy. All students are expected to abide by the University's Academic Integrity Policy which prohibits cheating and other misconduct in student coursework. Publicly sharing or posting online any prior or current materials from this course (including exam questions or answers), is considered to be providing unauthorized assistance prohibited by the policy. Both students who share/post and students who access or use such materials are considered to be cheating under the Policy and will be subject to sanctions for violations of Academic Integrity.

More information can be found at https://resources.depaul.edu/teaching-commons/teaching/academic-integrity/Pages/default.aspx.

Posting work on online sites, such as Hero
All students are expected to abide by the University's Academic Integrity Policy which prohibits cheating and other misconduct in student coursework. Publicly sharing or posting online any prior or current materials from this course (including exam questions or answers), is considered to be providing unauthorized assistance prohibited by the policy. Both students who share/post and students who access or use such materials are considered to be cheating under the Policy and will be subject to sanctions for violations of Academic Integrity.

Academic Policies

All students are required to manage their class schedules each term in accordance with the deadlines for enrolling and withdrawing as indicated in the University Academic Calendar. Information on enrollment, withdrawal, grading and incompletes can be found at:
http://www.cdm.depaul.edu/Current%20Students/Pages/PoliciesandProcedures.aspx

Incomplete Grades

An incomplete grade is a special, temporary grade that may be assigned by an instructor when unforeseeable circumstances prevent a student from completing course requirements by the end of the term and when otherwise the student had a record of satisfactory progress in the course. All incomplete requests must be approved by the instructor of the course and a CDM Associate Dean. Only exceptions cases will receive such approval. Information about the Incomplete Grades policy can be found at
http://www.cdm.depaul.edu/Current%20Students/Pages/Grading-Policies.aspx

Preferred Name & Gender Pronouns

Professional courtesy and sensitivity are especially important with respect to individuals and topics dealing with differences of race, culture, religion, politics, sexual orientation, gender, gender variance, and nationalities. I will gladly honor your request to address you by an alternate name or gender pronoun. Please advise me of this preference early in the quarter so that I may make appropriate changes to my records. Please also note that students may choose to identify within the University community with a preferred first name that differs from their legal name and may also update their gender. The preferred first name will appear in University related systems and documents except where the use of the legal name is necessitated or required by University business or legal need. For more

information and instructions on how to do so, please see the Student Preferred Name and Gender Policy at http://policies.depaul.edu/policy/policy.aspx?pid=332

Students with Disabilities

Students seeking disability-related accommodations are required to register with DePaul's Center for Students with Disabilities (CSD) enabling them to access accommodations and support services to assist with their success. There are two office locations:

- Loop Campus (312) 362-8002
- Lincoln Park Campus (773) 325-1677
- Email: csd@depaul.edu

Students who register with the Center for Students with Disabilities are also invited to contact Dr. Gregory Moorhead, Director of the Center, privately to discuss how he may assist in facilitating the accommodations to be used in a course. This is best done early in the term. The conversation will remain confidential to the extent possible.

Please see https://offices.depaul.edu/student-affairs/about/departments/Pages/csd.aspx for Services and Contact Information.

Faculty Resources Available from the Dean of Students Office

The online classroom https://offices.depaul.edu/student-affairs/resources/faculty-staff/faculty-questions/Documents/Faculty_Resources_Online_Classroom.pdf

Syllabus Resources Available from Teaching Commons

https://resources.depaul.edu/teaching-commons/teaching-guides/course-design/Pages/syllabuses.aspx#samples