CS 3331: Concurrent Computing

Instructor: Jean Mayo

Office: Rekhi 304

Office Hours: Tuesday 2:00-3:00, Thursday 2:00-3:00 Other times by appointment.

The office hour on Tuesday will be in-person and through Zoom. The Thursday office hour is by Zoom only. Students who come to office hours in person will take priority over Zoom attendees when both are present.

Zoom: Use **ID: 418 643 7423**. A Zoom meeting will be ongoing throughout the office hours. Just join the meeting. You will be put in a waiting room. As soon as I am free, I will let you in. Send me an email if you think we have lost synchronization.

In-person: You may stop by my office in Rekhi 304.

Email: jmayo@mtu.edu

Lecture: Lecture is in-person. There is no Zoom option. Lectures will be recorded and posted after class

Prerequisites:

CS 1142 or (CS1141 and CS1040) and CS2311 and CS2321. If the scheduling works, I strongly recommend that you complete CS 3411 before taking this course.

Text:

There is no required textbook. The lecture notes and UNIX manual pages disseminate the course material.

Description:

This course introduces students to the issues that arise when multiple processes access the same data. This shared access can significantly complicate application design. Arguably, this is one of the most important courses you will take because virtually all applications now require concurrent computing and the design and implementation of these complex applications can make or break functionality and usability.

The overarching goals of this course are for students to:

1. Know the fundamental concepts of concurrent computing
2. Understand the importance of race conditions and the impact on synchronization and hence mutual exclusion
3. Be able to use modern synchronization primitives in your programs
4. Learn to find race conditions and avoid deadlock

Topics Covered:

We will cover the following topics in the order below.

1. Basic systems concepts important to concurrent computing
2. Multi-process and multithreaded programming basics
3. Race conditions, critical sections, and synchronization
4. Pure software and hardware solutions for mutual exclusion
5. Synchronization primitives (i.e., semaphores, mutex locks, monitors, basic message passing), deadlocks and livelocks
6. Security issues and consequences of concurrent executions (e.g., CPU modes, memory protections, race conditions)
7. Survey of some languages (e.g., Java) and libraries (e.g., Pthreads)

Attendance:

Students are expected to attend lecture.

Exams will be in person except by prior arrangement.

You are not expected to (and should not) come to class or an exam when you are sick or have COVID-19 symptoms. If you are sick for a week or less, **you are expected to use your slip days**. If you are sick for more than a week, you will be asked to provide documentation and must contact me to make arrangements for completing missed work. If you go into isolation, contact me if the isolation impacts your ability to complete required class activities.

Evaluation:

The course grade will be comprised of the following components.

40 % Tests - There will be three equally weighted exams. The last exam will be given during the final exam period.

50 % Projects – Five projects will be assigned over the course of the semester. The projects will be equally weighted.

10% Participation – Attendance in general is not required. However, there will be several class periods over the semester at which attendance is required. Some of these class sessions will additionally require completion of problems prior to the class. Completion of all the problems and attendance will be graded at 100% for that class period; anything else will be graded as 0%. The sessions will be equally weighted. Required attendance at a class will be announced by the class period preceding the required lecture.

Homework will be assigned but will not be collected or graded. It is imperative that you do the homework. I do not collect and grade homework because this material requires a lot of practice to master. This policy gives you a chance to learn the material without being graded. If you do not do the homework, you will almost certainly do poorly on the exams.

The final letter grade will be determined according to the following table.

Numeric Course Score	Letter Grade
>= 93	A
[88 – 93)	AB
[83 – 88)	B
[78 – 83)	BC
[73 – 78)	C
[68 – 73)	CD
[60 – 68)	D
<60	F

The cutoff between letter grades may be lowered, but it will not be raised.

You must get a 60% average across both the exam and programming portions to get a passing grade in the course.

Assignment Submission Requirements:

Programming projects will be submitted via Canvas. Details about each submission (file names, submission format, etc.) will be given in the project specification. Please pay careful attention to file naming requirements. Small file name differences can break the grading scripts and significantly increase the grading time unnecessarily.

It is expected that you will check each submission to ensure it compiles and performs as you expect at the time of submission. Projects will be graded **`colossus.it.mtu.edu`**, **`guardian.it.mtu.edu`**, or a CS lab machine. When your program does not perform as you expected, you will be asked to demonstrate its performance on one of these machines. The fact that it ran on your own computer will **not be taken into consideration**.

Assignment Scheduling and Late Submissions:

Over the course of the semester, you may have **five project** slip days. Each person is given an automatic extension of five calendar days which you can use on any **programming project** during the semester, as long as the total amount of lateness does not add up to more than five days. For instance, you can hand in one project five days late, or two projects three and two days late respectively. When you hand in project hard copy, **you must identify at the top of the assignment**: (i) how many days late is the submission (**even if it is zero days late**) and (ii) how much of the total slip time you have left (even if you have zero days left). Slip days are used in increments of days, not hours. For example, if an assignment is submitted one hour after the due date and time specified on the project handout, the assignment is one day late and a slip day has been used. Each calendar day that passes beyond the due date counts as a slip day. Once the five slip days have expired, projects can be submitted at a penalty of 20% per day. **No slip days can be used beyond 5pm on Friday of the last week of classes, and no assignments will be accepted after 5pm on Friday of the last week of classes**. Slip days are intended to account for **routine illness, job interviews, system outages, etc**.

The number of slip days is based on the electronic timestamp associated with the code submission. Due dates and times are given in Eastern Standard Time (EST). Slip days will be calculated using submission times in EST.

Extensions to assignment due dates will only be granted in exceptional circumstances.

Note that project grading will not start until either all students have submitted a project or five days has passed since the due date. This is so that all the projects can be graded at once. Notice that this can significantly increase the turnaround time for project grades.

Collaboration Policy:

Unless stated otherwise, programming projects are to be performed individually. You may neither show anyone your work/code nor look at the work/code of anyone else. Unless explicitly stated otherwise, this policy extends to any external resource, including code found on the web or individuals who are not enrolled in the course.

For certain programming projects, you may engage in empty hands discussions **with anyone who is taking the course this semester**. If empty hands discussions are allowed for a particular programming project, this will be noted in the project specification. The model for an empty-hands discussion is a conversation between two or more people who are carrying nothing and are conversing within an entirely empty room. No participant in an empty-hands discussion may look at a book, handwritten notes, code printout, or computer. If you are unsure about the acceptability of a collaboration activity, you are expected to check with me before engaging in the activity.

ChatGPT or other code generating tools may not be used. Any violation of the academic integrity policy (https://www.mtu.edu/deanofstudents/policies-resources/integrity/) will result in a grade of F in the course.

Notes:

- Any concern about the grading of an assignment **must be raised within 3 days** after return of the assignment.
- No course requirement supersedes the University, State or CDC recommendations for your protection from COVID-19.
- This course will use C and C++. We will discuss C++ to some degree but barely enough for our purpose. Thus, you should get a good reference of C/C++.
- Always start doing your programming assignments EARLY.

- Concurrent programming requires careful planning, Concurrent applications can be much more complex than sequential applications. This means that your first solution is likely to miss some important case. When you receive a programming assignment, do not sit in front of a workstation and start typing your program immediately. Its common that your initial plan is wrong and then you try to tweak your existing code a few times, and then you realize you must sit down and think it through much more carefully. Then much of the code you typed in already is useless. It's a significant waste of time**. The best way to write a concurrent program is to think carefully with a pencil and a piece of paper before touching a computer keyboard**.
- **Since the behavior of a concurrent program is dynamic, which means it acts differently from time to time, you should try to reason about the correctness of your program. In general, just running for one or two test cases is not sufficient**. Our grader will read your program to find possible errors. When writing sequential programs, the same bug appears at the same place every time you run the same program. However, bugs of a concurrent program (e.g., race conditions and deadlocks) may not appear every time. As a result, you may have an incorrect program even though you may have had hundreds correct test runs.

University Policies:

Student work products (exams, essays, projects, etc.) may be used for purposes of university, program, or course assessment. All work used for assessment purposes will not include any individual student identification.

Michigan Tech has standard policies on academic misconduct and complies with all federal and state laws and regulations regarding discrimination, including the Americans with Disabilities Act of 1990. For more information about reasonable accommodation for or equal access to education or services at Michigan Tech, please call the Dean of Students Office, at (906) 487- 2212 or go to the University Policies for Course Syllabi web page.