CS444: An Introduction to Operating Systems
SPRING 2026v3

Dr. J. Holly DeBlois
Office: McCormack, 3rd floor, room M-3-201-32
Office hours: Tues/Wed/Thurs 2:30-3:30pm

Lectures and Class	Tuesday & Thursday, 4:00-5:15, W01-0005 section 02 (course 3891) Tuesday & Thursday, 5:30-6:45, W01-0005 section 01 (course 3745)
Instructor Email	jane.deblois@umb.edu
Instructor Website	Lecture and assignments are at: https://www.cs.umb.edu/~hdeblois/cs444/s26 Grades are on canvas https://www.umb.edu/canvas/
Portal:	Register for cs444 at https://portal.cs.umb.edu/ to create your course directory
Piazza:	Join Piazza: https://piazza.com/umb/spring2026/cs444

Course Description: In CS444, we present the basic aspects of operating systems, a layer of software between hardware and user software, and we code projects in C within the OS context.

Textbooks: The main reference in this course is *Modern Operating Systems, 5th edition.*, (Pearson, 2023) by Andrew S. Tanenbaum and Herbert Bos. The C code reference is: *The C Programming Language, 2nd edition* (Prentice Hall, 1978) by Brian W. Kernighan and Dennis M. Ritchie, which presents coding in C in 8 chapters and has an appendix on grammar of C. For emacs editor: https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf.

Attendance: Mandatory.
Note: If you miss more than 5 classes, your final grade will be lowered. If you miss 6 or 7, lowered by 1 letter grade step. If you miss 8 or more, lowered by 2 letter grade steps. See chart of scores and grade steps on last page.

Topics: We shall cover the following topics:
- **Operating System** definitions and features
- **Processes** and scheduling algorithms, threads, the critical region problem and semaphores
- **Memory management**, including cache memory, virtual memory, paging and segmentation
- **File systems**: implementation, management and optimization
- **Input/Output control**: buffering, spooling, disk management and disk access scheduling
- **Deadlocks**: detection, recovery, avoidance and prevention
- **Virtualization** and cloud computing
- **Multiple processor systems**
- **Security** threats and defenses
- **Linux/Android/Windows**: differences and similarities
- **C Code Grammar**: concise and lasting

I**n-Class Exercises:** You must bring a computer to class and have a setup in place to access your course directory on the CS Linux servers. We hand-write C code. We use emacs to create .c files on the server. We compile and run on the server. Both kinds of exercises are graded. Since you are allowed 5 missed classes, your 5 lowest scores of work for particular classes will be dropped.

Note: No courses required by the CS major, minor or certificate may be taken pass/fail.

Prerequisites: CS310 Algorithms and CS341 Computer Architecture. You need to describe and implement algorithms in C. You need to draw timelines and architecture down to driver and hardware level. You need skills from CS240 Programming in C and from probability distributions in CS220 Applied Discrete Math. We offer optional ungraded hw0/0.5 to assure that you can work on the servers and access your course directory without changing privacy settings (do not use a linux move command to put files into it, use a copy command (cp -r -p <file>).

Evaluation: Your grade in the course is determined by: four C code programming projects (40%), four homework assignments (30%), two tests (20%), class exercises (explained above) (10%) and whether you miss more than 5 classes (explained above). Submit your programming and homework by uploading files of the specified type to the server and placing them in your course directory as directed. You lose points if permissions are incorrect on your course dir. You lose points if scripts cannot find your files.

Programming projects: Design carefully and explain. Upload, edit, compile and run on the server each time you work since C libraries on the server differ from those on your machine. Follow the rules in the **Academic integrity section** below – give sources for and limit the amount of non-original C code you copy, copy and modify or generate. Provide a minimum of three incremental submissions of each project on the server starting at least two days before the assignment is due. Name the files as directed. This shows your development process. We test your code using Measure of Software Similarity (MOSS) provided by Stanford University. **Software similarity to other students' code greater than 45% earns 0 points.** We may call you in to explain your code.

Homework: Upload your homework to the server. Drawings in the homework are handwritten, photographed, and included in your typeset document converted to PDF. Some are .txt files.

Tests: The two tests are oral exams, answering the instructor's questions in a 30- or 15-minute sessions. For test1, students form teams of two. For test2, each student is individually tested.

Collaboration policy: You are encouraged to ask questions in class and may consult with your fellow students about assignments. When writing C code, do not stay "stuck" – ask for help via piazza. Be careful not to help another student by giving code – keep your laptop closed while conversing and don't share videos that picture code – too high a MOSS similarity means 0 for both.

Late Penalties: In class exercises, homework, C code projects and tests are scheduled for precise times. **Late hwk or code submission loses 1% of the score per hour.** Late test arrival means you will be given less of the test. Late class arrival means you may miss hand-written or server work.

Academic integrity and Student Conduct will be strongly enforced. Your code and homework must be your own product, may include some code copied or modified from outside sources or generated by chatGPT or other large language models, but must be sufficiently original to pass the MOSS criteria given in Programming projects above. See https://www.umb.edu/academics/provost/academic-integrity. Since UMB does not have any guidance for putting sources in code yet, we will use the MIT guidance. See https://integrity.mit.edu/handbook/writing-code/ In addition, since neither has guidance for citing chatGPT or other large language model generated code, **document any code you generated and used by marking START, citing the prompt you entered and further down marking END.** We recommend you take steps to not allow chatGPT or any large language model to store your results. If you have a series of prompts, list them in your readMe.txt and put only the last one in the code. Student Conduct: You must be honest in all your conduct. The University presupposes that work for academic credit is the student's own and complies with University policies above and here**: https://www.umb.edu/camp-life/dean-of-students/student-conduct-process/.**

Accommodation: Section 504 of the Rehabilitation Act of 1973 offers guidelines and support for curriculum modifications and adaptations for students with documented disabilities. Contact the Ross Center at 617-287-7430 and please dcuss your accommodations with the instructor.

Syllabus and Schedule Subject to Change: The instructor reserves the right to change the syllabus when necessary and will let you know. Here is the tentative Schedule (29 classes over 15 weeks):

week	class/date	chapter	topic	assignment
1	#1-Tue 1/27	Syllabus/ 1	Introduction	In class C code and grammar, hand-written and on server (c/g-h/s) part1, login, emacs
	#2-Thu 1/29	1	OS, Huffman code	hw1 posted, in class C c/g-h/s part2, emacs
2	Thurs 2/5add/drop ends			
	#3-Tue 2/3	2	processes, threads	In class C c/g-h/s part3, passwordless login
	#4-Thurs 2/5	2	processes, threads	In class C c/g-h/s part4, hw1 due, malloc
3	#5-Tue 2/10	3	memory, Hamming code	proj1 posted, in class C c/g-h/s part5
	#6-Thu 2/12	3	memory	In class C c/g-h/s part6
4	Mon 2/16 Holiday			
	#7-Tue 2/17	4	file systems	In class C code readMe.txt design part1
	#8-Thu 2/19	4	file systems	In class C code readMe.txt design part2
5	#9-Tue 2/24	5	input/output	hw2 posted, in class C code1, proj1 due
	#10-Thu 2/26	5	input/output	proj2 posted, in class C code2, test signup
	Fri 2/27 N/A grades due			
6	#11-Tue 3/3	12	designing an OS/	In class C code3, hw2 due
	#12-Thu 3/5	1-5,12	review(Ch 1-5,12, C code)	
7	Mon-Wed 3/9-3/11	testing	test 1 (Ch 1-5, 12, C code)	no class Tues
	#14-Thu 3/12	6	deadlock	In class C code4, proj2 due
-	3/15-3/22 Spring Break			
8	#15-Tue 3/24	6	deadlock	hw3 posted, in class C code5
	#16-Thu 3/26	7	virtualization	In class C code6
9	#17-Tue 3/31	7	virtualization	proj3 posted, in class C code7, hw3 due
	#18-Thu 4/2	8	multi-processor systems	In class C code8
10	#19-Tue 4/7	8	queuing theory, posix threads	In class C code9
	#20-Thu 4/9	9	security	In class C code10
11	#21-Tue 4/14	9	security	hw4 posted, in class C code11, proj3 due
	#22-Thu 4/16	10	Unix, Linux, Android,	proj4 posted, in class C code12, test signup
12	Mon 4/20 Holiday			
	#23-Tue 4/21	10	Unix, Linux, Android	In class C code read1 handwritten,hw4 due
	#24-Thu 4/23	10	Unix, Linux, Android	In class C code read2 handwritten
	Fri-4/23		Pass/Fail/Withdraw deadline	
13	#25-Tue 4/28	11	Windows	In class C code read3 handwritten
	#26-Thu 4/30	11	Windows	In class C code read4 handwritten, proj4 due
14	#27-Tue 5/5	6-11	review(Ch 6-11, C code)	review
	Wed-Fri 5/6-5/8	testing	test 2 (Ch 6-11, C code)	no class Thurs
15	#29-Tue 5/12		special topic	last class

Other References. *Operating Systems Design and Implementation, 3rd edition* (Pearson, 2006) by Andrew S.Tanenbaum and Albert S. Woodhull, which includes 392 pages of C code for the MINIX OS. C code of the OS of the server can be found in the file tree.

Revisions: Section numbers corrected. Posted 1/22/2026. Piazza link/add drop day corrected 3feb2026

Table of score to grade conversions (default grading scheme "UMB letter" in canvas):

93 <= S = A
90 <= S < 93 = A-
87 <= S < 90 = B+
83 <= S < 87 = B
80 <= S < 83 = B-
77 <= S < 80 = C+
73 <= S < 77 = C
70 <= S < 73 = C-
67 <= S < 70 = D+
63 <= S < 67 = D
60 <= S < 63 = D-
S < 60 = F