CS444: An Introduction to Operating Systems

FALL 2026 22 Sep 2026 Rev2

Dr. J. Holly DeBlois

Office: McCormack, 3rd floor, room M-3-201-32, phone 617 287-6490

Office hours: 12:30-1:30pm Tues, Wed and Thurs

Lectures and Class	Tues & Thur, 2:00-3:15, Wheatley-Peters W01-0005 section 01 (course 1409) Tues & Thur, 5:30-6:45, Wheatley-Peters W01-0004 section 02 (course 3737)
Midterm oral tests	Sign-up required: test1: 30 minutes, test2: 20 minutes. **See Tests, p4.**
Instructor Email	jane.deblois@umb.edu
Instructor Website	Syll., lectures and assignments: https://www.cs.umb.edu/~hdeblois/cs444/f26
Grades:	Grades are on canvas https://www.umb.edu/canvas linked to Gradescope
Graders:	Ernesto BrauMonteverde and Prentice Tang (OH: noon-1pm, Mon, Fri)
Portal:	**Register for cs444 course directory**: https://portal.cs.umb.edu/
Piazza:	Join and post questions: https://piazza.com/umb/fall2026/cs444

Course Description: Description of current operating systems, with focus on one or two in particular. Topics include: defining the operating system as distinct from the hardware on one side and software systems on the other; process concepts; memory management; CPU scheduling; device management; file systems; network support. We code in C. Most OS are written in C. We study C code grammar. We use Linux and the GNU Emacs editor on the CS servers.

Prerequisites: CS310 Algorithms and CS341 Computer Architecture. Use skills from CS240 Programming in C and from probability distributions in CS220 Applied Discrete Math. **We offer optional hw0 to review using Linux on the CS servers.**

Attendance: Mandatory. **If you miss >5 classes, your final grade will be lower.** If you miss 6 or 7 classes, you lose 1 letter grade step (e.g., B to B-). If you miss >7 classes, you lose 2 letter grade steps (e.g., B to C+). Because you are allowed to miss 5 classes, requests to be excused from class are rarely needed. If you need to be excused more, you must submit documentation. See https://www.umb.edu/registrar/policies/attendance/ . Being late to class or leaving class for more than a few minutes will lower your participation points. **See Participation, p5.**

Course Objectives/Outcomes: We shall cover the following topics:

- **Operating System** definitions, user interface including command line interface (CLI) or graphical user interface (GUI)), hardware interface, speeds, sizes and features
- **Processes** and scheduling algorithms, threads, critical regions in code and semaphores
- **Memory management**, binary, ASCII, and hex codes, drawings of process space, cache memory, virtual memory, paging and segmentation
- **File systems**: implementation, sizes, permissions, management and optimization
- **Input/Output control**: buffering, spooling, disk management and disk access scheduling
- **Deadlocks**: detection, recovery, avoidance and prevention
- **Virtualization** and cloud computing, VMware case study, Popek paper
- **Multiple processor systems:** multiprocessor, multicomputer and distributed systems
- **Security** threats and defenses
- **Linux/Android/Windows**: case studies, differences and similarities
- **C Code:** grammar, sample code, MINIX3 code, code for algorithms to implement

- **CS Linux Server Skills**: access, secure copy, passwordless login, emacs and CLI
- **Diagrams**: hardware, software layers, timelines, architectures and mechanisms
- **Portable Operating System Interface (POSIX) standards** and licensing

By the end of this course, the student will know basic OS topics and trade-offs involved in operating system design. The student will be proficient in C code and its system interfaces with an OS. The interfaces are American National Standards Institute (ANSI) standard libraries including Input/Output (I/O) and printf. The student will know how to use the Linux OS on the CS servers and the editor GNU Emacs. The student will be familiar with POSIX standards.

Textbooks:
(1) Purchase or rent: *Modern Operating Systems, 5th ed.* (Pearson, 2023) by Andrew S. Tanenbaum and Herbert Bos (T&B), with ch1-12.
(2) Access online C code textbook: *The C Programming Language, 2nd edition* (Prentice Hall, 1978) by Brian W. Kernighan and Dennis M. Ritchie (K&R), with C in ch1-6, std libr. in ch7-8, ApA, B https://archive.org/details/the-ansi-c-programming-language-by-brian-w.-kernighan-dennis-m.-ritchie.org .
(**3**) Read weekly assignments in library on reserve (or purchase or rent): *Operating Systems Design and Implementation, 3rd edition* (Pearson, 2006) by Tanenbaum and Albert S. Woodhull (T&W), with ch1M-5M and C code of the MINIX3 OS, Ap B, pp 639-1030. See www.minix3.org .
(**4**) Access GNU Emacs editor, Richard Stalllman, 1984, https://www.gnu.org/software/emacs/ and see also: https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf .

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

Bring Suitable Computer and Pencil to Class. You must bring a computer to class each day that is capable of accessing your CS course directory and doing in class assignments there, which will be script-collected and graded. For handwritten work, bring a pencil or pen. A page of paper with the assignment will be provided. Fill in your name as it is in wiser and your student id. Hand it in. The instructor or grader will scan all pages into gradescope which will post to canvas.

Avoid Lost Points on Submitting In-Class (IC) Exercises. There are two types of IC assignments. For submitting On-Server work, you will get zero points if you do not create the files in your course directory during class. For Handwritten work, use the given assignment page. Fill it in carefully. You will get zero points if you do not hand in the page during class.

Be careful when Uploading or Rearranging Files in your Course Directory. C code projects, Homework and some In-Class assignments require uploading files to your course directory. Always use the linux copy commands (“cp -p <file>”) or (“scp -p <file>”) and not the Linux move command (“mv”), which can disrupt permissions. Use “ls -la” or “ls -l” to check all file information. **Allow extra time ahead of when due.** You get zero points if permissions are incorrect on your course directory or subdirectory so graders cannot run your code or copy your files. You also get zero points if scripts cannot find your subdirectories and files (Linux is case-sensitive so “HW1” is NOT the same file as “hw1” and “Proj1” is NOT “proj1”).

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 too much, either by giving code or giving the specifics of your code design or by giving answers to your homework questions – keep your laptop closed while conversing and don’t share videos that picture code, proofs, calculations or diagrams.

Grading: Your score is a weighted average of assignments in 11 groups. They are defined here, shown on the Schedule, p5-8 below and listed in canvas. In each group, your score is a fraction, the

sum of points earned divided by the total points on all assignments in the group. See the table for conversion of score to grade, p9 below. The first two groups , "**C**" and "**H**", have multiple parts to submit. Part weights are set by the number of points designated in each assignment. **Each of the first 10 groups is described further below.** Here are the groups and weights:

- **C** code programming projects, 4 of them (**32%**) each including 4 parts:
  - subdirectory in course dir named proj<n> ("proj1" etc.),
  - <reqd-starter>.c, C code you must include (in-class handwritten, collected, scanned),
  - readMe (handwritten page, added to as you work, in-class collected, scanned),
  - project C files, compiled on server, with executables that run and >=3 versions,
- **H**omework assignments, 3 of them (**20%**) each including 3 parts:
  - subdirectory in course dir named hw<n> ("hw1" etc.),
  - hw<n>.txt (in-class handwritten page and on-server edited file),
  - hw<n>.pdf (uploaded, including photos of handwritten drawings),
- **T**ests, oral, 2 of them (**20%**),
- **S**erver access and CLI skills (in-class on-server) (**3%**),
- **E**diting using Emacs, compile and run (in-class on-server) (**3%**),
- **D**iagrams of OS abstractions (in-class handwritten, collected, scanned) (**3%),**
- **G**rammar of C (in-class handwritten, collected, scanned) (**5%**),
- **U**se case (in-class photo of partial C project handwritten readMe, uploaded) (**1%**),
- **M**INIX3 code T&W, identify OS features (in-class handwritten, collected, scanned) (**1%**),
- **P**articipation: whether you participate during class (**2%**), and
- **A**ttendance: whether you miss >5 classes, **described above on p1** (**10%**).

Dropped Scores. Because 5 missed classes are allowed, 5 of your lowest in-class exercise scores will be dropped **at mid-semester**, one each from assignment groups **S, E, D, G and U**.

C code projects: With <n> as the project number, the Schedule shows when each of the 4 C code projects will be posted on the class website (not canvas). Choosing your Use Case is required. **See Use cases, p5.**

Each group "C" assignment has 4 parts:

1) Create a subdirectory , "proj<n>", in your course directory. The first one is named proj1 exactly.

2) In class, given a formatted page <required-starter>.c, you handwrite the C code that you are required to include in your code and submit the handwritten page. Put the same code in your code and compile and run.

3) Given a second formatted page, you start handwritng your readMe. Write your name and student id. Select your Use case. Then start designing your code. Keep the readMe page and handwrite more on it as you go along. Record each date/time, what you worked on, what was difficult for you in that session and cite your sources specifically including AI sources if any.

4) Create your C code incrementally and submit it on the server. **See Avoid Lost Points by Doing Incrementals below.** You are required to use gcc or a Makefile to compile your code on the server. Run your executable each time you work. See **Avoid Lost Points by Using C Code Libraries on the Server and Keeping Enough Disk Space** below.

When we grade your code, we run it, copy it, recompile it, check that the executables match what was submitted and check it for similarity to other students' code using Measure of Software Similarity provided by Stanford University. **Software similarity to other students' code or sample generated code >45% earns 0 points.** We do a practice MOSS run on <required-starter>.c. We may call you in to explain your code. If your code works, but we cannot identify your own work in C vs. your generated code meaning you did not demonstrate your understanding of the code you submit, your grade on the C code files will be at most 50% of the points. Your final submission of **code is due before the start of your class section.**

Avoid Lost Points by Doing Incrementals. Your submitted project C files must include code and executables with at least two prior versions of each. To save an earlier version of your <file>.c, copy it to a name with a date indicator ('cp -p <file>.c <file_23jul>.c'). Then edit <file>.c. Incrementals' timestamps must match comments about starting the next version in your handwritten readMe. You get zero points if you forget to make, mention and submit incrementals.

Avoid Lost Points by Using C Code Libraries on the Server and Keeping Enough Disk Space. The C code libraries on the CS server are NOT the same as the ones on your laptop or home computer, so you MUST compile, run gcc and test your executables on the server. Using VSCode is not enough. You need to be proficient "on" the servers. VSCode can cause you to exceed your given disk space and be literally unable to submit. Type 'du -h' to check your usage. Delete unneeded files to fix or email operator@cs.umb.edu for help and copy jane.deblois@umb.edu.

Homeworks: With <n> as the homework number, the Schedule shows when each of the 3 Homeworks will be posted on the class website (not on canvas). Each group "H" assignment has 3 parts:
1) Create a subdirectory , "hw<n>", in your course directory. The first one is named hw1 exactly.
2) In class, given a formatted page called hw<n>.txt, you handwrite and use Emacs to type on the server the required information. It will be collected in class for scanning and script-collected from your course directory on the server. Be aware of **Avoid Lost Points by … Keeping Enough Disk Space** above.
3) Upload your homework to the server. Drawings in the homework are handwritten, photographed, and included in your typeset document converted to PDF by an editor. Be sure to login to the server and use "ls -la" to check that your filenames are exact.

Tests: The Schedule shows the day to sign-up using Doodle and the three weekdays when each test is given. The two tests are oral exams, answering the instructor's questions in a 30- or 20-minute session. For test1, students take the test in pairs. To take test1 with a particular student, sign up at the same time. Doodle cannot do 20 minute time blocks, so test2 has 3 seats per hour listed and we go in order of sign-up. The two students in test1 are given the same score, so it does not matter who gives the correct answer. Ross Center students may request to be tested alone, but they must ask ahead of the sign-up. For test2, each student is individually tested. Tests are randomized.

Server access and server CLI skills. The Schedule shows when each of the #S<n> assignments will be given. These are in-class on-server early work to be sure you know how to use client-server setup and move around on the servers.

Editing using Emacs, use with redirection commands, list hex with hexdump or compile with gcc or Makefile and run. The Schedule shows when each of the #E<n> assignments will be given. These are in-class on-server tasks using Emacs. The .emacs file gets created and checked. Be aware of **Avoid Lost Points by … Keeping Enough Disk Space above**, since unable to save would mean losing points on in-class work.

Diagrams of OS abstractions include timelines, mechanisms, hardware, software layers or networked systems. The Schedule shows when each of the #D<n> assignments will be given (in-class handwritten, collected, scanned).

Grammar of C exercises will cover the grammar rules in Appendices A&B of K&R in the approximate order of the way examples are presented in T&B and T&W. The Schedule shows when each of the #G<n> assignments will be given (in-class handwritten, collected, scanned.)

Use case for sources is your choice. The Schedule shows when each of the 4 #U<n> assignments are due. AI research is dual-sphered, part with only looking at prompt engineering, part with looking at 'full-stack' including neural networks. Photo your readMe including your choice of use case for assignment <n> and upload (in-class photo of early C proj handwritten readMe, uploaded). Note: National Institute of Standards and Technology (NIST) AI Standards Zero Drafts Project is requesting input including use cases. **Use Cases** identified (so far) include:
Case 1-none) You can do the project without using any AI, but you must say that Case1 is what you chose and you must cite the traditional sources you used, if any, and the code you copied, if any, or say you didn't use any.
Case 2-ahead) You can use AI before you start coding get your design together and then write the code yourself without using any generated code from AI. Also, say whether you used any traditional sources before you started coding, and whether you copied any of that code, which case you must cite it.
Case 3-during) You can use AI when your code doesn't work, and then correct your code by inserting sections that are generated code marked by START and END markers with a prompt accompanying the START marker. Cite traditional sources and copied code if you use any.
Case 4-entire) You can use AI to generate the entire assignment but then you have to supply handwritten tests and handwritten comments in a separate document to show you understand your code. You will likely be called in to demo your tests and explain the code design and details.

MINIX3 code, read selected pages of code. The Schedule shows when each of the #M<n> assignments will be given. We'll follow the order of T&B chapters (which differs from the order in T&W). Relate selected source code to the OS features described in the chapter (in-class handwritten, collected, scanned).

Participation: The instructor may deduct participation points if you are late, leave early or leave the classroom for too long or too often. We use 1 pt per 10 mins, so you earn 8-9 points per session minus the deduction. The Schedule shows 25 sessions total, since there is no class on the Tuesday of the test periods. It is participation to be sure the instructor and your classmates know your name, so please say it ahead of asking or answering question or requesting or giving help.

Late Work/Missed Work. In-class exercises, homework, C code projects and tests are scheduled for precise times. Late test arrival means you will be given proportionately less of the test and thus a lower grade. Late class arrival means you may lose participation points or miss hand-written work and be graded zero. UMB-mandated extensions will be given, but you must write the instructor before the work is due. If you are ill, write the instructor and we will work something out. If you are ill enough to miss a test, your make-up can only earn 50% of the full grade unless you provide a doctor's note. Traveling or delayed flights are not valid reasons for extensions. Joining the class late earns you 48 hours extension on hwk1 and an opportunity to make-up in-class assignments you missed. **You will see a zero posted for work that is not submitted when due.** Ross Center students must request an extension on an assignment before the work is due; when graded, their score will be posted in place of the zero. If a student is excused from doing an assignment on-time, their score will be posted in place of the zero when the work is graded. **Late penalty is 1% per hour down to half the score.** The instructor reserves the right to correct a grade up or down if it has been posted in error.

Academic integrity 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 Large Language Models (LLMs), but must pass the MOSS criteria given above and you must understand the code you submit. See https://www.umb.edu/academics/provost/academic-integrity. We use the MIT guidance on putting sources into code. See https://integrity.mit.edu/handbook/writing-code/ In addition, for citing code generated by a Large Language Model (LLM), **document the code you**

generated and used by marking the start as START, citing the prompt you entered and further down marking the end as END. Do not forget to include the prompt(s) and single START and END markers for code that is entirely generated or you will receive zero points.

AI Policy. This course DOES allow the use of AI on C Code projects, so include START and END markers stated above. This course DOES NOT allow the use of AI in generating answers for homework, which is mainly handwritten and photographed. When using AI, we recommend not allowing any LLM to store results for retraining. For C Code projects, you are required to know the name(s) of the LLM(s) you use and list them in your handwritten readMe. Note: Google search bar is "staffed" by Gemini. UMB provides student access to Microsoft 365 Copilot. Student access to Claude Code is being explored, but not yet provided by UMB. Free-tier Claude does not include Claude Code. ChatGPT is also available.

Required Statement from Provost: AI is allowed with attribution: Use of AI tools, including ChatGPT, is permitted in this course on certain assignments [either detail assignments or types of assignments here, or clarify how they will know which ones! **See above**.] To adhere to our scholarly values and to the Student Code of Conduct, students must cite any AI-generated material that informed their work; citations should include not only in-text citations and listing in the references, but also the full text of cited ChatGPT (or other Large Language Model (LLM) generator) as an appendix to the assignment. Using an AI tool to generate content without proper attribution qualifies as academic dishonesty. Students are also responsible for making sure that any AI generated text does not contain false or erroneous information. If students are unsure about whether or not a source is appropriate to use in the assignment, they should contact the instructor.

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/dean-of-students/student-conduct . You may not post the work that you submit for this class publicly either during or after the semester is concluded.** It may be useful in applying for jobs, but you can supply it privately if asked.

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 by email, ross.center@umb.edu, by phone, 617-287-7430, or in person, Campus Center, UL, Room 211. The Ross Center will write your instructor regarding your accommodations, not your specific disability. Please discuss accommodations with the instructor. To get extensions, you must write the instructor each time before the work is due.

Student Assistance for Health, Well-being and Success:

- Are you in emotional distress? Call 617.287.5690 to speak with a licensed clinician 24/7 who can offer support, crisis recommendations, and assistance with finding resources.
- Have a campus question or issue? Use Here4U in the UMass Boston app or via www.umb.edu/here4U.
- Want advice in navigating a university or life situation? Contact the Dean of Students Office at www.umb.edu/deanofstudents.
- Want to connect with housing and food insecurity support, student life groups and events, or recreation activities? Visit Campus Life - UMass Boston.
- Want to access resources specifically for immigrant-origin, DACA, TPS, and undocumented students? Visit Immigrant Student Programs - UMass Boston.

- Want to make the most of your academic experience? Visit Student Equity, Access & Success.
- Looking for additional identity-based community support? Find more resources at Inclusive Identity - UMass Boston.

Unable to attend class on a specific date or participate in an exam or class requirement due to a religious observance? Fill out the excused absence form (requires 2-weeks' notice) to request religious accommodation at Right To Excused Absence Because Religious Belief - UMass Boston

Syllabus and Schedule Subject to Change: The instructor reserves the right to change the syllabus when necessary and will let you know. The Schedule covers 14 weeks, 1 holiday, 2 tests (no class on Tues), so 25 class sessions including two review sessions. In-class work is abbreviated IC. Not all #S, #E, #D, #G, or #M assignments are listed yet. Here is the Schedule:

WEEK	Class#/Date/ Post&Due	Reading	Lecture Topic	Class Activities **Class lectures are recorded.** **C code proj2 not shown YET**
1	#1-Tue Sep 8 **hw1 posted**	444Website, Syllabus, ch1, p1-81	ch1: Introduction, client-server	**Class website. Syllabus**: view and read **Lecture 1**: ch1 T&B slides 1-22, 34-41 hw1: view and read, demo 'mkdir hw1' InClass (IC) On-server: **S1** login (register at portal first), 'ls -la', pwd, cd cs444, scp/cp -p IC Handwritten: **D1** draw client-server Hpath
	#2-Thu Sep 10 **H1-1 dir due by class**	Huffman, Emacs, ASCII chart	Huffman slides, C code	**Lecture 2:** Huffman slides, grammar hw1: view and read IC On-server: **S2** CLI shell, ps -eF, ps -U, top IC On-server: **E1** Emacs data, hexdump -C IC Handwritten: **D2** draw simple Huff tree IC Handwritten/On-server: **H1-2 huff**
--	----	----	----	----
2	#3-Tue Sep 15 hw1 due Add/drop Ends	ch12, ch1M, K&R, ApA	ch1: Introduction ch12: Design OS	**Lecture 3:** ch12, grammar, ch1M proj1: view and read **I**C On-server: **S3** ls -la hw1, shell, IC On-server: **E2** Emacs <file>.c, comment grammar, gcc, run IC Handwritten: **C1-2** <reqd-start-proj1>.c
	#4-Thurs Sep 17 Sat Sep 19 H1-3 due	ch1, ch2 K&R pointers, ApB	ch1: Introduction ch2: Processes, threads	**Lecture 4:** ch1 sl.23-33, ch2 IC On-server: **S4** show processes IC On-server: **E3** emacs f.c print pointers IC Handwritten/On-server:**G1** endian.c gram.
--	----	----	----	----
3	#5-Tue Sep 22 **C1-proj1 posted** **C1-1 dir due in class**	ch2, ch2M includes, **main.c**	ch2: Scheduling	**Lecture 5**: ch2, ch2M **read proj1 description** IC Handwritten: **D3** timeline diagram **IC Handwritten: C1-2a required code** **IC On-server: C1-2b compile and run**

	#6-Thu Sep 24	ch3	ch1: Introduction ch3: Memory	**Lecture 6:** ch3, ch3M IC On-server: **S5** passwordless login IC Handwritten/On-server: **G2** IC Handwritten: **D4** draw memory I**C Handwritten: C1-3 readMe started**
--	----	----	----	----
4	#7-Tue Sep 29	ch3, ch4M	ch1: pointers ch3: Memory	**Lecture 7**: ch3, read hw2 IC On-server: **M2** MINIX3 IC On-server: **S6** output formats IC On-server: **E5** Emacs file IC Handwritten: **D5** draw memory 2
	#8-Thu Oct 1 **Sat Oct 3:** **C1-4 proj1 due**	ch1, ch4	ch1: Introduction ch4: file systems	**Lecture 8**: ch1, ch4 IC On-server: **E6** Emacs file
--	----	--	----	----
5	#9-Tue Oct 6 **hw2 posted**	ch4, ch5M	ch1: Introduction ch4: File systems	**Lecture 9**: ch4, ch5M, read hw2 IC On-server: **M3** MINIX3 IC On-server: **E7** Emacs look MINIX3
	#10-Thu Oct 8 Fri Oct 9 - N/A grade due	ch5	ch5: I/O	**Lecture 10**: ch5, read proj2 IC On-server: **E8** Emacs look at shell, test IC Handwritten/On-Server: **G3** IC Handwritten: **D6** lost print job
- -	----	----	----	----
6	Mon Oct 12 - Holiday #11-Tue Oct 13 hw2 due	ch5, ch3M	ch5, ch3M	**Lecture 11**: ch5 I/O, raid, nibbles IC On-server: **M4** MINIX3 IC On-server: **U2** photo C2 readMe, upload IC Handwritten/On-server: **G4** IC Handwritten: **D7**
	#12-Thu Oct 15 review posted	Popek	h7.11 VMware	**Lecture 12**: ch1, ch7.11, read review IC Handwritten/On-server: **G5** IC Handwritten: **D8** IBM virt, early pc virt
--	----	----	----	----
7	#13-Tue Oct 20 proj2 due, proj3 posted **C2-4 due by class** **C3-1 dir due by class**	ch1, Hamming	ch1: Introduction Hamming	**Lecture 13**: Hamming, raid, nib., read proj3 IC On-server: **E9** IC Handwritten: **C2-3** collect readMe page, IC Handwritten: **C3-2** <reqd-start-proj3>.c IC Handwritten: **D9** Hamming
	test1 signup #14 Thu Oct 22	ch1-5,12,C	Review session	**Lecture 14: Review**, sample problems
--	----	----	----	----
8	Mon-Wed Oct 26-28 no class Tues	ch1-5,12, C testing	test **1** (Ch1-5, 12, C code)	**Test1: Mon-Wed, 8am-8pm, students come two at a time, about every 30 minutes**
	#15-Thu Oct 29	ch6,ch3M	Deadlock	L**ecture 15**: ch6, read proj3

				IC Handwritten/On-server: **G6** code fr hw2old IC Handwritten: **D10**
--	----	----	----	----
9	#16-Tue Nov 3	ch7	ch1,ch7, Virtualization and the Cloud	**Lecture 16**: ch7 to ch7.10 IC On-server: **M5** MINIX3 IC On-server: **U3** photo C3 readMe, upload IC Handwritten:/On-server: **G7** IC Handwritten: **D11** Layers of virtualized
	#17-Thu Nov 5	ch8	ch8: multi-processor systems	**Lecture 17**: ch8 IC On-server: **G8 access another server** IC Handwritten: **D12 world internet**
--	----	----	----	----
10	#18-Tue Nov 10 hw3 posted, proj3 due **C3-4 due by class** **Add to canvas: 5 dropped assignments** Wed Nov11 -Holiday	queueing, threads	queueing, posix threads	**Lecture 18**: queuing slides, pthreads slides Video: wiring servers featuring Tom Mullaly IC Handwritten: **C3-3** collect readMe page **Field trip**, last 25 mins, CS server room
	#19-Thu Nov 12 **H3-1 dir due by class**	ch9	ch9: Security	**Lecture 19**: ch9 IC On-server: **M6** MINIX3 IC Handwritten/On-server: **H3-2** hw3.txt
--	----	----	----	----
11	#20-Tue Nov 17 hw3 due, proj4 posted **H3-3 due midnight** **C4-1 dir due by class** #21-Thu Nov 19 Pass/Fail/Withdraw deadline	ch1, ch10 ch10.8	ch10: Unix, Linux, Android ch1: Introduction ch10.8: Android	**Lecture 20**: ch10, read proj4 Video: Patrick Brady, Android design IC On-server: **G9** IC Handwritten: **C4-2** <reqd-start-proj4>.c **Lecture 21**: ch1, ch10.8, read proj4 IC On-server: **G10 access Android code** IC Handwritten: **D13 Android layers**
--	----	----	----	----
12	#22-Tue Nov 24 Holiday Thu Nov26-29 review posted	ch11	ch1,ch11: Windows	**Lecture 22:** ch1, ch11 Windows, ch5.6 IC On-server: **U4** photo C4 readMe, upload IC Handwritten/On-server: **G11** IC Handwritten: **D14 Windows layers**
--	----	----	----	----
13	#23-Tue Dec 1 proj4 due **C4-4 due by class** **test2 signup**	ch11	ch11	**Lecture 23**: ch11 Windows IC Handwritten/On-server: **G12** IC Handwritten: **C4-3** collect readMe page
	#24 Thu Dec 3	ch 6-11, C	review session	**Lecture 24**: **Review**, sample problems
--	----	----	----	----

14	Mon-Wed Dec9 no class Tues #25-Thu Dec 10	ch6-11, C testing to be posted	test2 Ch6-11, C code special topic	**Test2: Mon-Wed, 8am – 8pm, individual, about every 20 minutes** **Lecture 25**: special topic, last class
	14 weeks, 1 holiday, 27 class meetings	2 off for testing	25 lessons, 2 are review sessions	25 lectures

Revisions:
Rev 2 adds grader OH, added compile and run to C code proj and updated schedule, shows C1-proj1, but does not show C2-proj2 YET, but it will soon.
Rev 1 increases the Attendance weight in grading to match how mandatory Att. is described on p1

Grading: 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