Syllabus for SE 14-317: Operating Systems
Department of Software and Information Systems
Achi Racov School of Engineering
Kinneret College on the Sea of Galilee

Instructor: Michael J. May

Semester 1 of 5787

1 Course Details

The course meets **8:30am–11:30am** on Sunday at **Room 803**. The laboratory session for the course is **8:00am–10:00am** and **10:00am-12:00pm** on Wednesday. The course has **3** hours of lecture and **2** hours of recitation. The metargel for the course is Avi Zion.

2 Course Prerequisites

Course prerequisites: SE 14-221: Computer Organization and Programming.

3 Overview

This is an introductory course to modern operating systems. Topics include process synchronization and interprocess communication, processor scheduling, memory management, virtual memory, signal handling, device management, I/O, and file systems. The course recitations and homework will include a hands-on study of the Linux operating system design and kernel internals, including work with installed (host) and virtual (guest) Linux environments. The course will include experience with commercial virtualization tools and open source software.

The material for the course is primarily from Anderson and Dahlin [1] and Arpaci-Dusseau and Arpaci-Dusseau [2], but also from Silberschatz, *et al.* [8] and Love [6].

4 Course Goals

At the end of the course the student will be able to:

1. Write command line programs in C for standard Linux that use standard system calls (*e.g.,* `fork()` and `nice()`), low-level I/O (*e.g.,* `open()`, file descriptors), and stream-level I/O (*e.g.,* `fopen()`, `FILE*`).
2. Write a Makefile and use it to compile C programs using `make` and `gcc`.
3. Use the GNU debugger `gdb` to debug command line applications in Linux, including multi-process and multi-threaded ones.
4. Manage source code repositories using `git` and Github.

5. Explain the basics of OS protection mechanisms and structures (*i.e.* process, thread, memory address space, virtual memory).
6. Use the PThreads library to write C programs which use user level threads, semaphores, locks, and condition variables.
7. Explain the fundamentals of mutual exclusion and deadlock detection and prevention.
8. Explain the fundamentals of process and thread scheduling and how the mechanisms used in the O(1) Scheduler and the Linux Completely Fair Scheduler (CFS) work.
9. Explain the operation of memory management techniques: segments, pages, multi-level page table, inverted page table.
10. Explain the operation of file systems (FAT, NTFS, FFS) and fundamental data structured related to them (inode, MFT).

5 Lecture Schedule

The course lectures are structured in the following way. The relevant chapters of Anderson and Dahlin (AD) and Arpaci-Dusseau (OSTEP) are in the indicated columns. Material not covered well in the books may be supplemented from papers or other sources as shown in the O column.

#	Subject	AD	OSTEP	O
1	Introduction, OS Overview and History	1		
2	Virtual Machines, OS Definitions	1		[3]
	4 OS Fundamental Concepts	2.1-2.7, 3.1-3.3		
3	4 OS Fundamental Concepts	2.1-2.7, 3.1-3.3		
	Base and Bound Memory Protection	8.1-8.2		
4	Base and Bound to Segments	8.1-8.2		
	Introduction to Schedulers, PCB	7.1		
	SMT, Fork			
5	Signals and Process Control	4.1-4.3		
	I/O High and Low	3.1-3.3, 11.1-11.2		[6]2–3
	Drivers, Sockets			
6	Concurrency: Processes and Threads	4		[8] 2.7, 3.6
	Cooperating Threads			
7	Synchronization	4		[8] 6
	Mutual Exclusion, Lock Implementation	5		
	Semaphores, Condition Variables	5.4, 5.7		
8	Barrier Synchronization	5		
	Monitors, Readers/Writers	5.6, 6		
	Monitors and High Level languages			
9	Scheduling Intro	7.1		[6]9–10
	Basic Scheduling Algorithms	7		
	Advanced Scheduling	7		[7, 4]
	Scheduling Case Studies	7		
10	Starvation, Deadlock	2.7, 6.5		[5]
	Memory Management and Translation	8.1-8.2		[6]4
11	Segments, Segment Management		16	
	Pages, Page Tables		18, 20	
12	File Systems	13	41	
	Very Simple File System		40	
	FAT	13	41	
13	FFS, NTFS	13	41	
	Caching, TLB, Demand Paging	8.3, 9	19, 22	

Since this is an advanced course, students **are expected to come to class having read the material listed above in the lecture schedule**. Students who do not come prepared will find themselves at a significant disadvantage.

6 Assignments

There will be 4 assignments and 3 interactive labs during the course of the semester. All will involve a large amount independent learning.

Each assignment can be done alone or in groups of two (2) students. More details of the projects will be distributed during the course of the semester.

7 Recitations and Weekly Turn-In Work

Laboratory periods will include hands-on development on Linux. Every student is expected to have a personal computer available for work off campus, but the lab in room 6201 will be available for use when classes are not scheduled in it.

Recitations will include a short turn-in recitation task. The task is to be completed by the end of the recitation session. The task will automatically graded using Moodle or GitHub for quick feedback.

There will be a total of 12 turn-in recitation tasks. The final recitation session (#13) will be a review for the exam and will not include a turn-in task. The grades for the lowest 2 of the turn-in tasks will be ignored when calculating the grade.

Every student must submit the turn-in work (no groups).

8 Attendance

Students are responsible for all material presented in class, recitation, and laboratory sessions, all assigned readings, and all material provided for additional reading out of class. Students who miss a lecture or targil can look at the course syllabus and web page to see which material was missed.

Attendance is required for the course. Students who miss more than 2 lecture or recitation sessions may be expelled from the course.

Similarly, there is a weekly in class task for each recitation session. Attendance will not be taken directly, but there will not be an opportunity to make up missed recitation tasks.

8.1 Decorum

Students who attend lecture are expected to give their full attention to the material. Reading newspapers, talking on cellular phones, text messaging, or other distracting behavior will not be tolerated.

Students must arrive to lectures **on time.** After ten minutes, the door will be locked and no student will be allowed entry. The door will be opened at the next break in the lecture (approximately every 50 minutes). Students who need to leave during lecture for some urgent matter must leave quietly and may return at the next break.

As per college policy, the instructor reserves the right to expel from the classroom any student who is disturbing the lecture or others.

9 Submissions

9.1 How to Submit Assignments

Lab assignments must be submitted via Moodle.

Students must submit programming assignments via the private per-assignment GitHub repositories managed by the instructor. Materials sent via email or via any other method risk being ignored or ungraded without consideration of their merits. Materials posted to other locations (*i.e.* public repositories, student managed repositories, GitHub repositories not connected to the specific assignment) will not receive a grade.

For work submitted by a team of students (more than 1) via GitHub, every student on the team must make at least one significant code commit to the GitHub repository. If a student's name appears on a submission, but the student doesn't perform at least one significant code commit to the assignment repository, the student **will not** receive a grade for the assignment.

9.2 Assignment Late Submission Policy

Students are expected to be on time with their assignment submissions. Each assignment must be turned in by the date it is due.

Each student will be given 36 slip hours to use for late submissions. The slip hours can be used on a single assignment or divided up among several. Slip hours are rounded up per assignment (*i.e.* 70 minutes late = 2 slip hours, 3 minutes late = 1 slip hour). Once the slip hours are finished, a student's submissions will no longer be accepted.

36 hours after the due date of any assignment, no submissions will be accepted.

Students who are called up to Miluim duty will have their assignment deadlines extended in accordance with college policy.

9.3 How to Submit Recitation Turn-In Work

Recitation turn-in work will be due immediately following the conclusion of the week's recitation session. Submissions will primarily be via a private-per week GitHub repository. Some work will be submitted via Moodle. Each weekly turn-in work task will specify the proper submission method. Materials sent via email or via any other method risk being ignored without consideration of their merits. Materials posted to other locations (*i.e.* public repositories, student managed repositories, GitHub repositories not connected to the specific assignment) will not receive a grade.

9.4 Recitation Turn-In Work Late Submission Policy

Students are expected to be on time with their recitation turn-in work submissions. Late submissions will <u>not</u> be accepted.

10 Use of Generative Artificial Intelligence Tools

Generative Artificial Intelligence (GenAI) tools are widespread and often used to quickly generate code and learn about new subjects. We will use GenAI tools in class to help understand certain topics and to generate simulations that demonstrate core coures concepts.

Unfortunately, GenAI is often used by students to do their assignments for them, reducing the amount learned from labs and assignments. To encourage active learning and prevent the submission of AI generated responses or code that students did not write and do not understand, the following policies will be put into effect.

1. Responses that are completely or substantively AI generated in free-response questions or code submissions will not be given credit. If 2 or more elements of an assignment is denied credit, the entire assignment will not be graded.

2. Responses that are suspected of being partially AI generated will receive a grade only after an in person interview with the student submitting. If the student(s) interviewed does not show sufficient understanding or mastery of the response, the student will not be given credit. In case of submission by a group, one or more of the group may be interviewed. The responses from the student(s) interviewed will determine whether credit is given.

GenAI tools may be used to help with reviewing, studying, planning, and executing assignments, but submitted work must be performed by the student.

11 Cheating

Cheating of any sort will not be tolerated. Student collaboration is encouraged, but within limits as set forth in the college's rules on academic integrity. Any students caught cheating will be immediately referred to the department head and the Dean and may receive a failing grade for the course.

Cheating includes:

- Copying information, content, or verbatim text from other students, internet sites, books (other than the ones listed in the bibliography), other unaffiliated individuals to answer questions, solve problems, or aid in programming projects.
- Copying or submitting source code, documentation, or other programming aids **without attribution** from other students, **web sites**, online repositories, text books, open source programs, or other unaffiliated individuals.
- Project teams which submit work which is identical or substantially identical to work submitted by other project teams, whether current or from previous years.
- Other forms of academic misconduct as described on the site: `https://catalog.upenn.edu/pennbook/code-of-academic-integrity/` or as reasonably assessed by the instructor, program head, or dean.

If you have any questions about what constitutes cheating in the above rules, contact the instructor as early as possible.

12 Exams

There will be a final exam at the end of the course that covers all course material.

13 Grading

Final grades will be calculated by combining grades from assignments, weekly turn-in work, and exam. The grades are weighted as follows:

10%	Recitation turn-in work (ignore 2 lowest grades)
30%	Assignments
60%	Exam

The instructor will not address questions about specific individual grades during the lecture or review sessions. Students may contact the instructor *in person* during office hours or after the lecture/review sessions at the instructor's convenience.

14 Books

The following books are used for the class: Anderson and Dahlin [1], Arpaci-Dusseau and Arpaci-Dusseau [2], Silberschatz *et al.* [8], and Love [6]. The library has copies of the books listed, but students are encouraged to purchase the books as needed.

A bibliography of the books and articles used in the course of the semester is shown below.

15 Contact Information

Instructor:	Michael J. May
Email:	`mjmay@mx.kinneret.ac.il`
Web page:	`https://mjmay-kinneret.github.io`

References

[1] Thomas Anderson and Michael Dahlin. *Operating Systems: Principles and Practice*. Recursive Books, 2nd edition, 2014.

[2] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 1.10 edition, Nov 2023. `https://pages.cs.wisc.edu/~remzi/OSTEP/`.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[4] Justinien Bouron, Sebastien Chevalley, Baptiste Lepers, Willy Zwaenepoel, Redha Gouicem, Julia Lawall, Gilles Muller, and Julien Sopena. The battle of the schedulers: FreeBSD ULE vs. Linux CFS. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 85–96, Boston, MA, July 2018. USENIX Association.

[5] Horatiu Jula, Daniel Tralamazza, Cristian Zamfir, and George Candea. Deadlock immunity: Enabling systems to defend against deadlocks. In *Proceedings of the 8th Conference on Symposium on Operating Systems Design and Implementation*, OSDI'08, page 14, USA, 2008. USENIX Association.

[6] Robert Love. *Linux Kernel Development*. Addison-Wesley Professional, 3rd edition, 2010.

[7] Jean-Pierre Lozi, Baptiste Lepers, Justin Funston, Fabien Gaud, Vivien Quéma, and Alexandra Fedorova. The Linux scheduler: A decade of wasted cores. In *Proceedings of the Eleventh European Conference on Computer Systems*, EuroSys '16, New York, NY, USA, 2016. Association for Computing Machinery.

[8] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating Systems Concepts*. Wiley, 10th edition, April 2018.