Linux Kernel Internals: Data Structures for Open Computing

Global Higher Education Alliance for the 21st Century (GHEA21)
American University of Central Asia (AUCA)

1 Course Description

Dive deep into the heart of the Linux kernel by exploring its internal data structures: linked lists, dynamic arrays, hash tables, red-black trees, bitmaps, and more. Through hands-on projects, you'll directly modify kernel code, deliberately replacing well-designed structures with inferior alternatives to measure and understand the performance consequences. By breaking things methodically, you'll discover firsthand why kernel developers made the design decisions they did. Using modern kernel development tools, including performance profilers, tracers, and eBPF observability frameworks, you'll quantify the real-world impact of different design approaches. Rooted in Linux's open-source philosophy, this course equips you to develop technology that fosters collaboration, transparency, and innovation.

2 Course Information

Course Materials
`https://github.com/auca/com.274`

Course Code
COM-274

Course ID
6379

Prerequisite (GHEA21)
Background in structured or procedural programming (ideally C)
Familiarity with Linux environments (basic shell, Git usage)

Prerequisite (AUCA)
COM-119, Object-oriented Programming

Credits
6 (ECTS), 3 (US)

Time and Place

Lecture: Monday 17:00–18:15 (Bishkek Time, UTC+6), Room 410
Lab: Wednesday 17:00–18:15 (Bishkek Time, UTC+6), Room 410

Offline and online, synchronous live sessions casted on Zoom
All sessions recorded and made available for asynchronous access

Please join the scheduled lectures and labs on Mondays and Wednesdays at 17:00 Bishkek Time (UTC+6), which is 12:00 Central European Time (UTC+1) and 06:00 Eastern Time (UTC-5). Note that Bishkek does not observe daylight saving time. If your country transitions to or from daylight saving time mid-semester, the class time relative to your local time will shift by one hour. Please verify that this shift will not cause scheduling conflicts. We will not be adjusting the class times, as most of our students are in the Bishkek time zone.

3 Contact Information

Professor

Dmitrii Toksaitov
toksaitov_d@auca.kg

Office (GHEA21)

Remote office hours (via Zoom)

Office (AUCA)

Room 315

Office Hours

Available by appointment, either on-site or remotely, during work hours (Monday to Friday). Please contact your professor to schedule a meeting.

4 Topics Covered

- **Week 1–3: Introduction to the Linux Kernel (9 hours)**
  - Setting up a development environment with virtme-ng for rapid iteration
  - Basics of C and the Unix Command-Line Environment
  - Kernel architecture overview and core subsystems
  - Building, booting, and testing custom kernels
- **Week 4–5: Linked Lists in Process Management (6 hours)**
  - Kernel list macros, intrusive data structures, and usage patterns
  - Exploring process lifecycle and task structures
  - Performance measurement with perf and stress testing
- **Week 6–7: Bitmaps in File Systems (6 hours)**

  - Efficient bitmap operations for resource tracking
  - Block and inode allocation in ext4
  - Measuring fragmentation and allocation performance

- **Week 8–9: Dynamic Arrays in the VFS Layer (6 hours)**
  - Growth strategies and amortized complexity analysis
  - File descriptor tables and kernel memory allocation
  - Profiling memory behavior and cache effects

- **Week 10–11: Hash Tables and eBPF Observability (6 hours)**
  - Hash table design, collision handling, and the dentry cache
  - Building kernel observability tools with eBPF
  - Real-time performance monitoring and visualization

- **Week 12–14: Red-Black Trees in the Scheduler (9 hours)**
  - Self-balancing trees and the Completely Fair Scheduler
  - Measuring scheduling latency and fairness
  - Advanced profiling with perf sched and ftrace

- **Week 15–16: Open-Source Contribution Workflows (6 hours)**
  - Submitting patches, code reviews, and best practices via GitHub
  - Reflection on open computing, future directions

5 Learning Outcomes

By the end of this course, students will be able to:

1. **Analyze and Apply Kernel Data Structures**:
   - Understand and explain key data structures (arrays, linked lists, hash tables, red-black trees, B-trees, bitmaps) in the Linux kernel
   - Modify and integrate these structures into kernel subsystems
2. **Demonstrate Kernel-Level Programming Expertise**:
   - Build, configure, and debug a custom Linux kernel
   - Apply best practices for patch submission and version control
3. **Utilize Open-Source Collaboration Tools**:
   - Work with Git and GitHub for version control, reviews, CI/CD
   - Collaborate across diverse geographies and backgrounds, reflecting the open-source development model
4. **Foster Critical Thinking and Innovation**:

- Evaluate and compare different data structures for specific kernel tasks
- Propose kernel improvements or new features based on data structure optimizations

5. **Practice Effective Communication and Inclusivity**:
   - Present kernel modifications in written or oral forms
   - Engage respectfully in collaborative online discussions with global peers

6 Assignments and Exams

6.1 Projects, Moodle/e-Course Checkpoints

Students will be assessed through incremental, hands-on projects following a consistent methodology: replace a kernel data structure with a deliberately inferior alternative, rigorously benchmark both implementations, and analyze the performance degradation. Each project targets a different subsystem—process management, file systems, the VFS layer, caching, and scheduling—allowing students to explore diverse areas of the kernel.

Each project has to be submitted to GitHub. Students must maintain instructor-provided private GitHub repositories for their assignments. They must periodically commit and push the required number of project solutions, as directed by the course staff. Instructors or teaching assistants will review the work either during lab sessions (on-site) or after the submission deadline (off-site) and award points based on completed assignments.

6.2 Exams or Evaluations

We will schedule one or two formal evaluation checkpoints (Midterm and Final Exams). These evaluations might involve quizzes testing conceptual knowledge and coding challenges performed in a supervised environment.

7 Course Materials, Recordings and Screencasts

All course materials are available on GitHub at `https://github.com/auca/com.274`. Using GitHub will help students become familiar with the Git version control system and the widely used GitHub platform.

We aim to record each class and upload it to YouTube for accessibility; Links to YouTube recordings can be found in the course repository at `https://github.com/auca/com.274`.

Access lecture screencasts remotely via Zoom at either `http://com-274-zoom.auca.space`. When joining a Zoom session, students must identify themselves using their properly capitalized first and last names in the Latin alphabet. Install and configure the Zoom on your computer during the first week of the semester so you can properly share your camera, microphone, and screen.

8 Software

Students are recommended to install the following software on their machines.

- Git: `https://git-scm.com`

The compilers, assemblers, and debuggers will be available on the remote course server provided by AUCA. A *Vagrantfile* detailing the software used on the course server will be available in our public course GitHub repository, should you wish to recreate the environment on your personal computer.

9 Hardware

There are no specific hardware requirements, as you can use the course server where we guarantee you will be able to complete all the tasks of our course. However, we recommend a machine with more memory (rather than less), high-speed disk I/O, and a GPU compatible with current graphics and compute APIs.

10 Reading

1. Understanding the Linux kernel, Third Edition by Daniel P. Bovet and Marco Cesati (AUCA Library Call Number: QA76.76.O63 B683 2006, ISBN: 978-0596005658)

10.1 Supplemental Reading

1. Linux Kernel Development, 3rd Edition by Robert Love (ISBN: 978-0672329463)

11 Grading

The preliminary distribution of points is outlined below. Please note that the distribution may change throughout the course if tasks are canceled, merged, or made optional (for bonus points). This usually happens for reasons such as software issues, online service outages, or classes canceled due to events outside our control.

Remember that some LMS platforms, such as Moodle, may not calculate final scores and grades correctly until all tasks have been published in the system and properly weighted. Do NOT assume your grade is accurate until all tasks are in the system and your instructor has asked you to review your scores and grades.

One common mistake students make is assuming that attendance points are part of the grade and will be summed at the end—they will not. These points are used only to help us decide on X grades in the middle of the semester and will be removed from the totals in Moodle at the end of the semester.

11.1 Moodle/e-Course Checkpoints

Your instructor will periodically announce reviews of your work. For such checks, you can be awarded up to the specified number of points.

- Projects (60%)

11.2 Exams

In the middle and at the end of the course, you will need to pass the Midterm and Final examinations. These exams are significant and will greatly affect your grade.

- Midterm Exam (20%)
- Final Exam (20%)

11.3 Totals

- 100% is formed from the Moodle/e-Course Project Checkpoints (60%) and the two exams (40%).

11.4 Scale

- [94%–100]%: A
- [90%–94)%: A-
- [87%–90)%: B+
- [83%–87)%: B
- [80%–83)%: B-
- [77%–80)%: C+
- [73%–77)%: C
- [70%–73)%: C-
- [67%–70)%: D+
- [63%–67)%: D
- [60%–63)%: D-
- Less than 60%: F

Please note that requests for a higher grade due to points being marginally close will be ignored. For instance, 93.99 is an A-, NOT an A. Similarly, requests for extra assignments to boost points will also be disregarded.

12 Rules

First and foremost, in addition to all the rules listed in the syllabus document, students are required to follow the Code of Conduct of the American University of Central Asia and GHEA21.

12.1 Questions

A question raised by one student is often relevant to others as well. Therefore, students are encouraged to use the online discussion board of the LMS (Learning

Management System) they are enrolled in (e.g., AUCA e-Course System) to post questions publicly so that all students may benefit from the discussion.

Students must not post complete source code for any task on the LMS discussion board. Any such public post will result in a grade of zero for that assignment. In addition, students should refrain from submitting vague or overly general questions such as "Why doesn't my code work?" Instead, they are expected to carefully analyze and debug their code before seeking assistance.

12.2 Late Policy

Late submissions and late exams are not permitted. Exceptions may be granted at the professor's discretion only in cases of force majeure. If you become ill, experience serious personal difficulties, or encounter technical problems with your computer or internet connection, you must notify the instructors at least 24 hours before the deadline. Failure to do so will result in no extension being granted. Requests made on the day of the deadline will be treated as procrastination and will not be considered. No student will be granted more than one emergency extension throughout the course.

Beginning six hours before any assignment deadline, instructors will enter "silent mode." During this period, no assignment-related questions will be answered, and no requests for office hours will be considered. At all other times, instructors will make every effort to respond to questions and provide assistance, whether via Zoom or in person during office hours.

12.3 Exam and Task Submission Ceremonies

Students must follow all exam and task submission protocols. This means strictly adhering to all rules specified by instructors, whether in written or verbal form. Failure to do so will result in a loss of points.

Throughout your career, you will work with various supporting documents such as contracts and timesheets. Beyond programming, you will also need to use project management systems, application life cycle management tools, version control, and continuous integration and deployment pipelines. You will manage complex configurations, collaborate with colleagues by following specific methodologies, and share code through pull requests or other means. In all these processes, proper etiquette and attention to detail are essential.

In our courses, we aim to help you build these skills. We will deduct points for not following submission rules or protocols, and we may even refuse to accept your exam defense or submitted tasks. We will also award zero points for missing task deadlines or violating strict exam timing rules, such as arriving late or failing to submit exam documents on time.

12.4 Administrative Drop

Instructors reserve the right to drop a student from the course for non-attendance. If a student misses three or more classes without an acceptable excuse, the faculty may assign an X grade and remove the student from the course.

12.5 Incomplete Grade

Consistent with the late exam policy, a grade of *I* (Incomplete) will be granted only under highly exceptional circumstances. Students must initiate the request for an *I* grade well in advance of the final exam; requests made the day before the final exam or later will not be considered.

12.6 Academic Honesty

Plagiarism is the act of copying or appropriating someone else's words or ideas and presenting them as one's own. In the context of this course, plagiarism may occur in many forms, including but not limited to program code, comments, software documentation, design specifications, requirement documents, project reports, and technical analyses.

More specifically, examples of plagiarism in a Software Engineering course include:

- Submitting code written by others or generated by AI as your own
- Modifying existing code or designs (e.g., changing variable names or restructuring code) and claiming originality
- Incorporating code snippets, sentences, design patterns, or intellectual content from any source without citation
- Using algorithms, patterns, or architectural designs without proper acknowledgment
- Using graphics, datasets, audio, video, or other materials from external works without attribution
- Purchasing or otherwise acquiring code, software, or project content and presenting it as original work

Plagiarism is unethical and undermines the educational process. The consequences are as follows:

- First instance: A grade of zero for the plagiarized work, and a report filed with the Registrar's Office
- Second instance: A failing grade (F) for the entire course

Both parties, the student who plagiarizes and the student whose work was copied, will face equal consequences, as every student is responsible for safeguarding their assignments, code, and related materials to prevent unauthorized use. This policy reflects real-world expectations in software engineering, where failing to protect intellectual property can result in termination, reputational damage, compromise the safety of customers and users, and even lead to lawsuits in a professional environment.

All work in this course must be completed individually. Group work, collaboration with peers, or seeking assistance from other students is strictly prohibited.

At the introductory level, relying on classmates often leads to the spread of misunderstandings and increases the risk of academic dishonesty. Students may seek help only from instructors, teaching assistants (TAs), or WARC tutors, ensuring that guidance comes from qualified sources and that each student develops their own skills and understanding.

The use of artificial intelligence, including but not limited to generative AI tools, to complete any assignments, projects, or exams, either off-site or on-site, is strictly prohibited. If there is suspicion that a student has used AI assistance in their work, the student will be required to perform a similar task and answer relevant questions in a supervised setting with the course instructors. Failure to satisfactorily complete this task or to answer questions convincingly will lead to the conclusion that the work was AI-generated. In such cases, the Academic Honesty policies outlined previously will be enforced, including potential disciplinary actions. Note that for some tasks, the instructor may allow the use of AI tools, but you must obtain explicit written permission in the requirements to use them.

Finally, this course adheres to all global AUCA university policies on academic honesty. If university-wide regulations are stricter than those outlined here, the stricter rules will take precedence.

13 Access and Support Services

This course is committed to fostering an inclusive learning environment that supports the diverse needs of all students. If you have a disability or require specific accommodations to participate fully, please contact the instructors as early as possible. We will work with you to ensure that appropriate adjustments are made to support your learning experience.

For guidance on time management, presentation skills, writing, or study strategies beyond the scope of Software Engineering, we encourage you to connect with the AUCA Advising Office. The Advising Office offers practical workshops and peer advising to help you navigate academic challenges.

If you prefer working with peers, you may schedule an appointment with a student tutor through the AUCA Writing and Academic Resource Center (WARC). Tutoring is available both in person and online. Additional resources can be found on the WARC webpage. Please note, however, that team or group work with other students is not permitted for assignments in this course.

If you are feeling stressed, overwhelmed, or struggling with emotions, relationships, or other mental health concerns, we encourage you to seek support from the AUCA Counseling Service. The Counseling Service provides confidential and free professional assistance to students.