Advanced Operating Systems: Syllabus

Prof. Robert N. M. Watson

2023-2024

Instructors: Prof. Robert N. M. Watson

Prerequisites: Undergraduate operating-systems course; please see below for further details

Structure: Lent 2023 delivery of the module is intended to be as follows:

- 5x 1½-hour lectures (in person),
- 1x 1-hour lecture (prerecorded),
- 3x ½-hour lecturelets (prerecorded),
- 3x 2-hour labs (in person).

If circumstances do not permit in-person teaching, then the contingency plan is as follows:

- All lectures will be delivered in prerecorded format.
- Scheduled Q&A sessions will be offered via Zoom for each pair of lectures.
- Mini-supervisions will be offered for labs, with a minimum of one per student per assignment.

1 Aims

Systems research refers to the study of a broad range of behaviours arising from complex system design, including low-level operating systems, resource sharing and scheduling, interactions between hardware and software, network-protocol design and implementation, separation of mutually distrusting parties on a common platform, and control of distributed-system behaviours such as concurrency and data replication. This module will:

1. Teach systems-analysis methodology and practice through tracing and performance profiling experiments;
2. Expose students to real-world systems artefacts such as the OS scheduler, Inter-Process Communication (IPC), and network stack, and consider their hardware-software interactions with storage devices and CPUs;
3. **ACS/Part III only**: Develop scientific writing skills through a series of laboratory reports; and
4. **ACS/Part III only**: Assign a selection of original research papers to give insight into potential research topics and approaches.

The teaching style will blend lectures, readings, and hands-on labs that teach methodology, underlying design principles, common implementation approaches, and practical skills. Students will be taught about (and assessed via) a series of lab-report-style assignments based on in- and out-of-classroom practical work. ACS/Part III students will complete and submit written lab reports. Part II students will complete structured assignments based on similar experimental exercises. The systems studied are real, and all wires will be live.

2 Prerequisites

It is strongly recommended that students:

1. Have previously (and successfully) completed an undergraduate operating-system course with practical content – or have equivalent experience through project or open-source work.
2. Have reasonable comfort with the C and Python programming languages. C is the primary implementation language for systems that we will analyse, requiring reading fluency; userspace C programs will also be read (and may be extended) as part of lab exercises. Python will be used for data collection and processing, and provides numerous useful tools for analysis and presentation (e.g., `matplotlib`).

   Students without a Python background may wish to complete an online Python tutorial prior to term, as the language will be used in data collection, analysis, and presentation from the first lab.
3. Review an undergraduate OS textbook (such as Silberschatz, et al.) to ensure that basic OS concepts such as the *process model*, *Inter-Process Communication*, *filesystems*, and *virtual memory* are familiar.
4. Be comfortable with the UNIX command-line environment including compiler/debugging tools. Students without this background may wish to sit in on the undergraduate UNIX Tools lecture series in Michaelmas.

If a student lacks experience in one of these areas, it is likely they will be able to pick it up as the module proceeds. However, if they lack experience in multiple areas, they may wish to meet with the module instructor and/or appropriate student office to determine whether it makes sense to take the module.

3 Lectures, Labs, Lab Reports, and Lab Assignments

The various elements of the module (lectures, readings, labs, and lab reports) are intended to be complementary to one another, rather than repetitive, introducing different aspects of operating-system research, design, implementation, experimentation, and analysis.

Introduction to kernels and tracing/analysis The purpose of this submodule is to introduce students to the structure of a contemporary operating-system kernel through tracing and profiling.

All students will perform a short-answer lab exercise to learn about kernel tracing tools.

Lecture 1: Introduction: OSes, Systems Research, and tracing

The first lecture reintroduces the idea of an operating system, its role, contemporary operating-system structure, and current operating-system research areas and venues. We will also discuss how (and why) operating systems are taught, and the approach taken in this module.

Lecture 2: Kernels and Tracing

The second lecture continues our exploration of OS structure. We look at the goals, implementation, potential uses of DTrace as a means of kernel instrumentation and tracing, and the probe effect. We also consider the high-level structure of a kernel (is it just a complex C program?) and its execution model.

Lab 1: Getting Started with Kernel Tracing The first lab explores using DTrace to learn about basic kernel behaviours, building skills required for later assignments.

- For ACS/Part III students, this assignment contributes 20% of the overall mark.
- For Part II students, this assignment is optional but strongly recommended.

Deliverable: Tracing Results

The Process Model This submodule introduces students to concrete implications of the UNIX process model: processes and threads in both userspace and kernelspace, the hardware foundations for kernel and process isolation, system calls, and traps.

Lecture 3: The Process Model The second lecture looks at the evolution of the process model, from its 1960s origins to the 1990s deployment of ELF, dynamic linking, and multithreading in UNIX. We take an initial dive into virtual memory, as well as the hardware foundations for system calls and traps.

Lecture 4: The Process Model (2) We continue our discussion of system calls and traps, considering their semantics, the system-call table and surrounding software stack, and their (desirable) security properties. We also begin to explore the implied (and very real) cost of the process model itself, revisiting virtual memory through insights from the Mach project.

Lab 2: Kernel and microarchitectural implications of IPC In Lab 2, the ACS/Part III and Part II assignments differ:

- ACS/Part III students will write their first full lab report. This assignment contributes 30% of the overall mark.
- Part II students will complete their third lab assignment. This assignment contributes 40% of the overall mark.

This lab consists of two parts:

1. The first part uses DTrace to understand the dynamics of local Inter-Process Communication: kernel memory allocation, copying, locking, scheduling, and message-based IPC. Of particular concern will be building an understanding of basic IPC functionality, but also of how it interacts with buffering and the scheduler to affect IPC latency and throughput.
2. The second part introduces a new performance analysis mechanism, *hardware performance counters*, which allow direct monitoring of low-level architectural and micro-architectural details of performance. Using this tool, we will revisit existing benchmarks to explain the use of CPU time by the application and kernel.

Deliverable: Lab Report *or* Lab 2 - Inter-Process Communication Performance

Submodule 3: TCP/IP This submodule introduces a contemporary network stack, with a particular interest in the TCP protocol.

Lecture 4: The Network Stack (1) The third lecture introduces the history and role of networking in OS design, with a focus on the Berkeley Sockets API and TCP/IP stack. We explore the flow of memory both from the perspective of hardware (NIC, DMA, memory, caches, and processor) and software (applications, buffering, the protocol stack, memory allocator, and device driver). We consider the input, output, and forwarding paths, with an interest in dispatch models and their interaction with multiprocessor systems. Finally, we look at two recent pieces of network-stack research, Netmap and network-stack specialisation.

Lecture 5: The Network Stack (1) The final lecture explores TCP protocol and implementation behaviour in greater detail. We consider the objectives and evolution of TCP, especially with respect to congestion control, performance, and denial-of-service (DoS) resistance. We also explore the evolution of in-kernel data structures in network-stack scalability. Research topics include the development of congestion control and differing models for network-stack multiprocessing. Finally, we consider how changes in NIC, bus, and processor hardware have impacted (and continue to affect) the implementation of TCP.

Lab 3: TCP In Lab 2, the ACS/Part III and Part II assignments differ:

- ACS/Part III students will write their second full lab report. This assignment contributes 50% of the overall mark.
- Part II students will complete their third lab assignment. This assignment contributes 60% of the overall mark.

Deliverable: Lab Report *or* Lab 2 - TCP

4 Objectives

On completion of this module, students should:

- Have an understanding of high-level OS kernel structure.

- Gained insight into hardware-software interactions for compute and I/O.
- Have practical skills in system tracing and performance analysis.
- Have learned how to perform systems-style performance evaluations.
- **ACS/Part III only**: Have been exposed to several current topics in systems research.
- **ACS/Part III only**: have developed scientific writing skills.

5 Coursework

The nature of the coursework depends on which version of the module or unit is being taken. The two versions of the course share a common introductory exercise (Lab 1), worth 20% of the overall mark, which teaches basic skills with the tracing tooling that we will be using. The remaining exercises share content across course versions, but differ substantially in deliverables and expectations:

ACS/Part III: Advanced Operating Systems Students taking the masters-level module will write and submit two lab reports to be marked by the instructor. The first report is a 'practice run' intended to help students develop analysis techniques and writing styles, and contributes 30% to the final mark. The remaining report constitutes 50% of the final mark.

Part II: Advanced Operating Systems Students taking the undergraduate-level unit of assessment will complete two further homework assignments to be marked by the instructor. The first assignment is a 'practice run' intended to help students develop data collection and analysis skills, and contributes 30% to the final mark. The remaining assignment is constituting 50% of the final mark.

6 Practical work

Online lab exercises, accompanied by short lab lecturelets and Q&A sessions, will ask students to develop and use skills in tracing and performance analysis as applied to real-world systems artefacts. Results from these labs will be the primary input to lab reports and assignments.

Please see the handout, *Advanced Operating Systems: Lab Setup*, for details on the lab platform. Typical labs will involve using tracing and profiling to characterise specific behaviours (e.g., file I/O in terms of system calls and traps) to diagnose application-level behaviours (e.g., with respect to effective use of TCP sockets for bulk data transport). Students may find it useful to work in pairs within the lab, but must prepare lab assignments and reports independently.

Students may seek support for lab activities via the Advanced Operating Systems Slack, as well as in scheduled demonstration sessions.

7 Assessment

Please see the handout, *Advanced Operating Systems: Lab Reports and Assignments*, for a description of the ACS/Part III lab-report format and its assessment, as well as Part II lab-assignment expectations.

8 Recommended reading

Please see the handout, *Advanced Operating Systems: Readings*, for a list of module texts, research readings, and supplemental texts.