L41: Advanced Operating Systems - Syllabus

Dr Robert N.M. Watson

Michaelmas Term 2016

Code: L41

Instructor: Dr Robert N. M. Watson

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

Structure: Six 1-hour lectures; five 2-hour labs

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. Develop scientific writing skills through a series of laboratory reports; and
4. Assign a selection of original research papers to give insight into potential research topics and approaches.

The teaching style will blend lectures and hands-on labs that teach methodology, design principles, 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. 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.

3 Lectures, Labs, and Lab Reports

Submodule 1: 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. The first lab report is written, which will be receive feedback from the instructor, but not contribute to the final mark.

Lecture 1: Introduction: OSes, Systems Research, and L41 (1h)

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 (1h)

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 - I/O (2h)

The first lab uses an exploration of POSIX file I/O to motivate learning about DTrace, user-kernel interactions, and performance analysis. The first lab report will describe this lab.

Deliverable: Lab Report 1 - POSIX I/O Performance Analysis

Submodule 2: 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. The second lab report will be written.

Lecture 3: The Process Model - 1 (1h) The third 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 (1h) The fourth lecture continues our discussion of system calls and traps. We consider 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 Implications of IPC (2h) The second lab 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.

Lab 3: Micro-Architectural Implications of IPC (2h) The third lab 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 2 - Inter-Process Communication Performance

Submodule 3: TCP/IP This submodule introduces a contemporary network stack, with a particular interest in the TCP protocol. Labs will consider both the behaviour of TCP connections, exploring the TCP state machine, socket-buffer interactions with flow control, and congestion control. Students will use Dummynet to simulate network latency and explore how TCP slow start and congestion avoidance respond to network conditions. The third (and final) lab report will be written.

Lecture 5: The Network Stack - 1 (1h) The fifth 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 6: The Network Stack - 2 (1h) 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 4: The TCP State Machine (2h) The fourth lab asks students to explore the TCP state machine in practice: how it is triggered by both API and network-level events. An early measurement of the impacts of network latency on TCP is performed.

Lab 5: TCP Latency and Bandwidth (2h) The final lab continues our investigation of the effects of network latency on TCP performance, and especially its interactions with congestion-control slow start and steady state. We also explore how socket-buffer configuration affects flow control, and the combined end effects on available bandwidth.

Deliverable: Lab Report 3 - The TCP State Machine, Latency, and Bandwidth

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 been exposed to research ideas in system structure and behaviour
- Have learned how to write systems-style performance evaluations

5 Coursework

Students will write and submit three 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 will not contribute to the final mark. The remaining two reports are marked and assessed, each constituting 50% of the final mark. Class participants are advised that, in prior teaching of this module, the average final mark of students submitting the first lab report was significantly greater than those who did not.

6 Practical work

Five 2-hour in-classroom labs, supplemented by follow-up work outside of the class itself, 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.

Please see the handout, *L41: 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 reports independently.

The module lecturer will give a short introductory lecture at the start of each lab, and instructors will be on-hand throughout labs to provide assistance. Lab participation is not directly included in the final mark, but lab work is a key input to lab reports, which are assessed.

7 Assessment

Please see the handout, *L41: Lab Reports*, for a description of the lab-report format and its assessment.

8 Recommended reading

Please see the handout, *L41: Readings*, for a list of module texts, research readings, and supplemental texts.