Advanced Operating Systems
(263-3800-00L)

Timothy Roscoe & Andrew Baumann

Herbstsemester 2010

http://www.systems.ethz.ch/education/courses/hs10/aos/

Aims

- *Thorough understanding* of design and implementation of modern operating systems
- *Theoretical material* not covered in existing ETH courses or textbooks
  - Focus on alternatives, principles, tradeoffs, and debates
  - Context: influence to two concrete systems: L4 and Barrelfish
- *Practical experience* implementing parts of a microkernel-based OS
  - Focus on concrete example and real implementation

- Key issues:
  - Memory management
  - Scheduling & Inter-process communication
  - Protection and security
  - Networking & Device drivers

Assumptions / Prerequisite

- C Programming: prerequisite
  - Practical labs are all in C. You will be writing an OS.
- Computer Architecture: prerequisite
  - You will be writing a device driver, pager, etc.
- Command line development environment: prerequisite
  - gcc, emacs/vi, make, etc.
  - Debugging facilities are primitive!
- Betriebsysteme: prerequisite
  - What the various bits of an OS are
  - Course material assumes Betriebsysteme (or the new OSNet)
- Systembau: alternative/complementary
  - Our emphasis: microkernels (practical) and advanced/research OS topics (theoretical)

Reading

- Most of our material is not (or insufficiently) covered in textbooks
  - See the website for (optional) background reading
- Readings (mostly research papers) for each week will appear on the website
  - Will help with understanding the lectures
  - May help with project work
  - **Will** help with the exam.

Acknowledgements

- This is the 4th year of AOS at ETHZ.

- Project is well-tested
  - Based on UNSW COMP9242
  - Many thanks to the UNSW ERTOS group

Dates, times, and people

- Lecture:
  - Thursday 10:00 - 12:00 in HG F.26.5

- Consultations and marking:
  - Friday 10:00 - 12:00 in CAB H.57

- People:
  - Timothy Roscoe, Andrew Baumann, Simon Peter

Exam (35%)

- Date(s)
  - To be determined.
- Material
  - Lecture subjects *and*
  - Lab exercise material
- Format
  - Oral exam (schedule will be arranged)
  - 15 minutes per student

Project (65%)

- Implement a simple OS over a microkernel (L4)
  - Programming in C
  - Real hardware (NSLU2 ARM-based machines)
  - Hosted from Linux
  - We provide support tools, basic code framework
- Milestone-based
- Somewhat independent of course material
  - Concepts should be familiar

Project milestones:

1. Familiarisation
2. Memory manager
3. A pager
4. System call interface
5. Implement filesystem
6. Timer driver and benchmarking
7. Demand paging
8. Process management
9. ELF Loading
10. Documentation

Run shell commands!

Important advice

- You will be graded on the *design* of your code
  - Does it work?
  - Handle corner cases, errors, invalid inputs, etc?
  - An operating system runs for a long time. Do you leak memory? Etc.
  - Have you thought about issues not explicit in the milestones, but important to a real OS?
- Not all these criteria can be well-specified in advance
  - Use common-sense in system design
  - You will be graded on issues you can think of and deal with

Important advice

You will be graded on the *quality* of your report

- Doxygen and other tools document *lines of code*, **not** the *design of a system*!
- Don't submit generated documentation in place of a report.
- Describe the *choices* you made (and didn't make)
- Talk about the tradeoffs
- Mention the *difficulties* and *challenges*, and how you overcame them.
- Show you understand how to build a system.

Important advice

- This course is a lot of fun, but a lot of work
  - and we are aware of this!
- It is important not to fall behind
  - If your team is struggling, ask for help.
- If you're good, it's tempting to be clever and cool
  - Resist this temptation!
  - Get the required work done before freestyling.
- We are here to help you
  - Particularly Simon ☺

This Project will use L4

- L4 is a µkernel, we'll see more about this later
- Flexibility means services can be implemented in a modular way as processes
- Minimality means there is lots to do
- Size means it comfortably fits on a small machine (NSLU2)
- Next up: introduction to L4.