University of Glasgow

Advanced Topics in Systems Programming

Advanced Systems Programming (H)

Lecture 1

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow | This work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License.
To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Course Coordinator and Materials

- Lecturer and course coordinator
  - Dr Colin Perkins (colin.perkins@glasgow.ac.uk)
  - Office: S101b, Lilybank Gardens

- Course materials on Moodle and https://csperkins.org/teaching/



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Rationale

- Technology shift: desktop PC → laptop, tablet, smartphone, cloud; mobile, power-aware, concurrent, real-time, connected
- But still programmed in C, running some variant of Unix – technology stack that's becoming increasing limiting
- This course will explore new techniques for safer, more effective, systems programming
  - Programming in an unmanaged environment, where data layout and performance matter
  - Operating systems kernels, device drivers, low-level networking code

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Aims and Objectives

- The course aims to explore the features of modern programming languages and operating systems that can ease the challenges of systems programming, considering type systems and run-time support.
- It will review the research literature on systems programming and operating system interfaces, discuss the limitations of deployed systems, and consider how systems programming might evolve to address the challenges of supporting modern computing systems.
- Particular emphasis will be placed on system correctness and secure programming, to ensure the resulting systems are safe to use in an adversarial environment.

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Intended Learning Outcomes (1/2)

- By the end of the course, students should be able to:
  - To discuss the advantages and disadvantages of C as a systems programming language, and to compare and contrast this with a modern systems programming language, for example Rust; to discuss the role of the type system, static analysis, and verification tools in systems programming, and show awareness of how to model system properties using the type system to avoid errors;
  - To discuss the challenges of secure low-level programming and write secure code in a modern systems programming language to perform systems programming tasks such as parsing hostile network input; show awareness of security problems in programs written in C;
  - To discuss the advantages and disadvantages of integrating automatic memory management with the operating system/runtime, to understand the operation of popular garbage collection algorithms and alternative techniques for memory management, and know when it might be appropriate to apply such techniques and managed run-times to real-time systems and/or operating systems;
  - …

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Intended Learning Outcomes (2/2)

- …
- To understand the impact of heterogeneous multicore systems on operating systems, compare and evaluate different programming models for concurrent systems, their implementation, and their impact on operating systems; and
- To construct and/or analyse simple programming to demonstrate understanding of novel techniques for memory management and/or concurrent programming, to understand the trade-offs and implementation decisions.

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Pre-requisites

- The following courses are pre-requisites:
  - Systems Programming (H)
  - Operating Systems (H)
  - Networked Systems (H)
  - Functional Programming (H)
- You are expected to be familiar with the C programming language, and to understand the basics of operating systems and networking
- A conceptual understanding of functional programming is assumed, but Haskell programming is not required

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Course Structure

Week	10:00-11:00 – Lecture Discussion	11:00-12:00	13:00-14:00 – Lab Sessions
1	#1: Introduction		#1: Introducing Rust
2	#2: Systems Programming	Office Hour	
3	#3: Types and Systems Programming	Office Hour	#2: Types and Traits
4	#4: Type-based Modelling and Design	Office Hour	#3: Ownership, Pointers, and Memory
5	#5: Resource Ownership and Memory Management	Office Hour	
6	#6: Garbage Collection	Office Hour	#4: Concurrency
7	#7: Concurrency	Office Hour	#5: Coroutines and Asynchronous Code
8	#8: Coroutines and Asynchronous Programming	Office Hour	
9			
10	#9: Security Considerations	Office Hour	#10: Future Directions

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Course Structure: Lectures

Week	10:00-11:00 – Lecture Discussion	11:00-12:00	13:00-14:00 – Lab Sessions
1	#1: Introduction		#1: Introducing Rust
2	#2: Systems Programming	Office Hour	
3	#3: Types and Systems Programming	Office Hour	#2: Types and Traits
4	#4: Type-based Modelling and Design	Office	
5	#5: Resource Ownership and Memory Management	Office	
6	#6: Garbage Collection	Office	
7	#7: Concurrency	Office	
8	#8: Coroutines and Asynchronous Programming	Office Hour	
9			
10	#9: Security Considerations	Office Hour	#10: Future Directions

- Lecture recordings will be made available ahead of time – approximately two hours of material per week, split into several parts
- Each lecture is accompanied by discussion questions
- Timetabled session from 10:00-11:00 on Mondays via Zoom is for discussion – watch the lecture and think about the discussion questions **before** the timetabled slot

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Course Structure: Office Hours

Week	10:00-11:00 – Lecture Discussion	11:00-12:00	13:00-14:00 – Lab Sessions
1	#1: Introduction		#1: Introducing Rust
2	#2: Systems Programming	Office Hour	
3	#3: Types and Systems Programming	Office Hour	#2: Types and Traits
4	#4: Type-based Modelling and Design	Office Hour	#3: Ownership, Pointers, and Memory
5	#5: Resource Ownership and Memory Management	Office Hour	
6	#	Office Hour	#4: Concurrency
7	#7: Concurrency	Office Hour	#5: Coroutines and Asynchronous Code
8	#8: Coroutines and Asynchronous Programming	Office Hour	
9			
10	#9: Security Considerations	Office Hour	#10: Future Directions

- Office hour from 11:00-12:00 on Mondays via Zoom for one-to-one discussion relating to the course

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Course Structure: Labs

Week	10:00-11:00 – Lecture Discussion	11:00-12:00	13:00-14:00 – Lab Sessions
1	#1: Introduction		#1: Introducing Rust
2	#2: Systems Programming	Office Hour	
3			#2: Types and Traits
4			#3: Ownership, Pointers, and Memory
5			
6			#4: Concurrency
7			#5: Coroutines and Asynchronous Code
8			
9			
10	#9: Security Considerations	Office Hour	#10: Future Directions

- Labs will run via Microsoft Teams
- Materials will be made available in advance, and are to be completed in your own time
- Timetabled sessions from 13:00-14:00 on Mondays are for live support with the lab exercises – try to solve the exercise, and think of questions you might need to ask, **before** the timetabled support slot
- Lab handouts tested with Rust 1.48.0 on macOS 10.14.6, but expected to run on student Linux servers (stlinux01 to stlinux08.dcs.gla.ac.uk) provided by the School

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Assessment

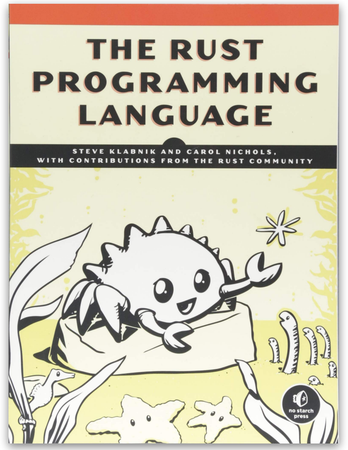
- This is a level H course, worth 10 credits
- Assessment is by examination (80%) and coursework (20%)
  - Sample exam paper, with worked answers, is available
  - Material from the lectures, labs, and cited papers is examinable
    - Aim is to test your understanding of the material, not to test your memory of all the details
    - Explain why – don't just recite what
- Assessed coursework:

Coursework	Date Set:	Date Due:	Weighting:	Topic:
Exercise 1	Lecture 5	Lecture 7	10%	Memory Management (essay)
Exercise 2	Lecture 7	Lecture 9	10%	Concurrent Programming in Rust (code)

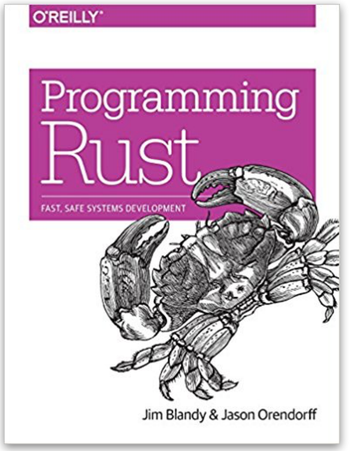
Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Recommended Reading (1/2)

- The Rust Programming language (https://rust-lang.org/) will be used to illustrate principles of ownership, memory management, and type-driven programming – **read one of these books**
- You are expected the learn the basics of programming in Rust



Steve Klabnik and Carol Nichols, "The Rust Programming Language", 2nd Edition, 2018, ISBN 978-1-59327-828-1 (Amazon, free online edition).

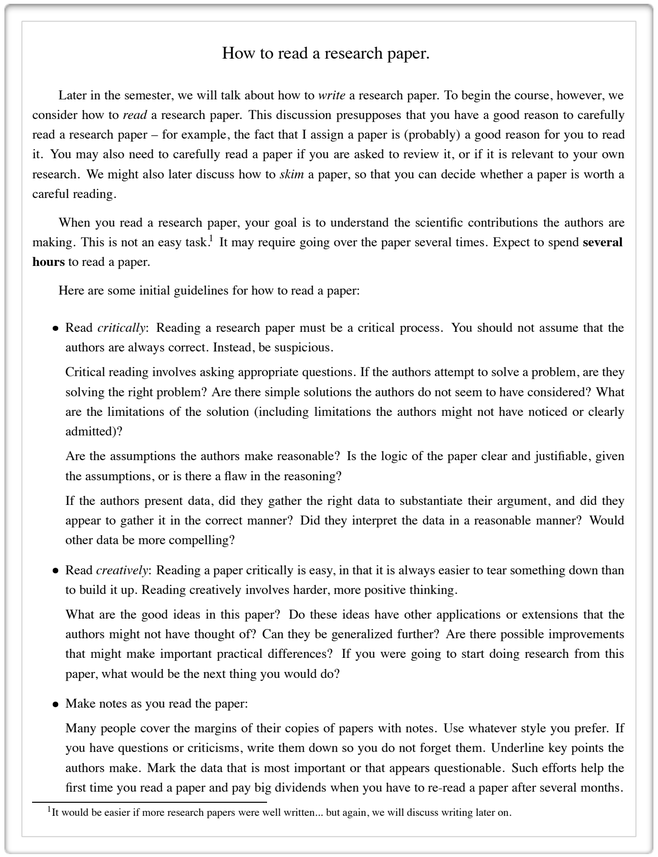


Jim Blandy and Jason Orendorff, "Programming Rust", O'Reilly, 2018, ISBN 978-1-491-92728-1 (Amazon).
Free access via University Library: https://tinyurl.com/y7gxx8dc

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Recommended Reading (2/2)

- Research papers will be cited to illustrate some concepts
  - Citations with URLs and/or DOIs (http://dx.doi.org/) provided
  - All papers are accessible at no cost from the campus network – **do not** pay to access research papers
- **You are expected to read these research papers**
  - Critical reading of a research paper requires practice
  - Read in a structured manner, not end-to-end, thinking about the material as you go; focus on the concepts, not the details
  - Realise that research papers are written to explore new ideas
    - Some will be good ideas, some less so
    - Some will be interesting but impractical
    - What's impractical today might be important tomorrow – changes in technology and/or society can change what's feasible/desirable
  - Think and judge for yourself!



How to read a research paper.

Later in the semester, we will talk about how to *write* a research paper. To begin the course, however, we consider how to *read* a research paper. This discussion presupposes that you have a good reason to carefully read a research paper – for example, the fact that I assign a paper is (probably) a good reason for you to read it. You may also need to carefully read a paper if you are asked to review it, or if it is relevant to your own research. We might also later discuss how to *skim* a paper, so that you can decide whether a paper is worth a careful reading.

When you read a research paper, your goal is to understand the scientific contributions the authors are making. This is not an easy task.[1] It may require going over the paper several times. Expect to spend **several hours** to read a paper.

Here are some initial guidelines for how to read a paper:

- Read *critically*: Reading a research paper must be a critical process. You should not assume that the authors are always correct. Instead, be suspicious.

  Critical reading involves asking appropriate questions. If the authors attempt to solve a problem, are they solving the right problem? Are there simple solutions the authors do not seem to have considered? What are the limitations of the solution (including limitations the authors might not have noticed or clearly admitted)?

  Are the assumptions the authors make reasonable? Is the logic of the paper clear and justifiable, given the assumptions, or is there a flaw in the reasoning?

  If the authors present data, did they gather the right data to substantiate their argument, and did they appear to gather it in the correct manner? Did they interpret the data in a reasonable manner? Would other data be more compelling?

- Read *creatively*: Reading a paper critically is easy, in that it is always easier to tear something down than to build it up. Reading creatively involves harder, more positive thinking.

  What are the good ideas in this paper? Do these ideas have other applications or extensions that the authors might not have thought of? Can they be generalized further? Are there possible improvements that might make important practical differences? If you were going to start doing research from this paper, what would be the next thing you would do?

- Make notes as you read the paper:

  Many people cover the margins of their copies of papers with notes. Use whatever style you prefer. If you have questions or criticisms, write them down so you do not forget them. Underline key points the authors make. Mark the data that is most important or that appears questionable. Such efforts help the first time you read a paper and pay big dividends when you have to re-read a paper after several months.

[1] It would be easier if more research papers were well written... but again, we will discuss writing later on.

http://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Advanced Topics in Systems Programming

- Lecture content starts in week 2 – watch lecture 2 and think about the discussion questions
- Labs start in week 1 – review the lab 1 handout and start on the exercise

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow