University of Glasgow

Advanced Topics in Systems Programming

Advanced Systems Programming (H/M)

Lecture 1

Course Team and Materials

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

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

Rationale

- Technology shift: desktop PC → laptop, tablet, smartphone, cloud; mobile, power-aware, concurrent, real-time, connected

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

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

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.

Intended Learning Outcomes (1/2)

- By the end of the course, students should be able:
  - 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;

Intended Learning Outcomes (1/2)

- By the end of the course, students should be able:
  - 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;

Intended Learning Outcomes (1/2)

- By the end of the course, students should be able:
  - 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;

Intended Learning Outcomes (2/2)

- By the end of the course, students should be able:
  - 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

Intended Learning Outcomes (2/2)

- By the end of the course, students should be able:
  - 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 programs to demonstrate understanding of novel techniques for memory management and/or concurrent programming, to understand the trade-offs and implementation decisions.

Pre-requisites

- 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
- This broadly corresponds to the material in the following courses:
  - Systems Programming (H)
  - Operating Systems (H)
  - Networked Systems (H)
  - Functional Programming (H)

Course Structure

Week	10:00-12:00 – Group Discussion	13:00-14:00 – Lab Sessions
1	#1: Introduction	#1: Systems Programming Languages
2	#2: Systems Programming	#2: Challenges in Systems Programming: Energy Efficiency
3	#3: Types and Systems Programming	#3: Introducing the Rust Programming Language
4	#4: Type-based Modelling and Design	#4: Types, Traits, and State Machines
5	#5: Resource Ownership and Memory Management	#5: Ownership, Pointers, and Memory
6	#6: Garbage Collection	
7	#7: Concurrency	#6: Closures and Concurrency
8	#8: Coroutines and Asynchronous Programming	#7: Coroutines and Asynchronous Code
9	#9: Security Considerations	
10	#10: Future Directions	

Course Structure: Lectures

Week	10:00-12:00 – Group Discussion	13:00-14:00 – Lab Sessions
1	#1: Introduction	#1: Systems Programming Languages
2	#2: Systems Programming	#2: Challenges in Systems Programming: Energy Efficiency
3	#3: Types and Systems Programming	#3: Introducing the Rust Programming Language
4	#4: Type-based Modelling and Design	
5	#5: Resource Ownership and Memory Management	
6	#6: Garbage Collection	
7	#7: Concurrency	
8	#8: Coroutines and Asynchronous Programming	#7: Coroutines and Asynchronous Code
9	#9: Security Considerations	
10	#10: Future Directions	

- Lecture recordings will be made available ahead of time, and each is accompanied by discussion questions
- Timetabled session on Monday mornings is for discussion: watch the lecture and think about the questions **before** the timetabled slot

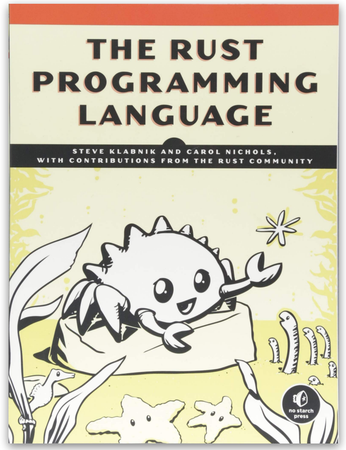
Course Structure: Labs

Week	10:00-12:00 – Group Discussion	13:00-14:00 – Lab Sessions
1	#1: Introduction	#1: Systems Programming Languages
2	#2: Systems Programming	#2: Challenges in Systems Programming: Energy Efficiency
		#3: Introducing the Rust Programming Language
		#4: Types, Traits, and State Machines
		#5: Ownership, Pointers, and Memory
		#6: Closures and Concurrency
8	#8: Coroutines and Asynchronous Programming	#7: Coroutines and Asynchronous Code
9	#9: Security Considerations	
10	#10: Future Directions	

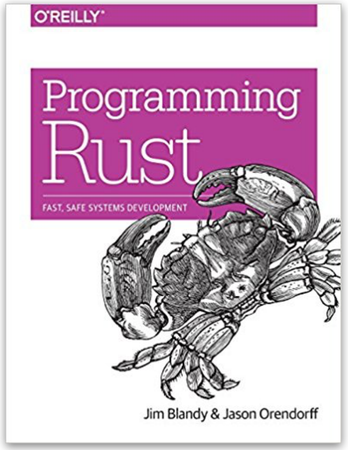
- Timetabled sessions on Monday afternoons are for discussion and support with the lab exercises – try to solve the exercise, and think of questions you might need to ask, **before** the timetabled support slot
- Lab exercises are tested on the student Linux servers (stlinux01 to stlinux08.dcs.gla.ac.uk) provided by the School but should run on any system with a recent version of the Rust programming language installed

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

Recommended Reading (2/2)

- Research papers and blog posts 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 or VPN – **do not pay to access research papers**
- **You are expected to read this material**
  - 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 rather than details
    - See http://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf
  - Realise that research papers are written to explore new ideas
    - Some will be good ideas, some less so; some will be interesting but infeasible; 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!

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 and blog posts** 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)

Advanced Topics in Systems Programming

- Discussion sessions start in week 2 – **watch the lecture 2 recording and think about the discussion questions**
- Labs start in week 1 – **review the lab 1 handout and start on the exercise**