University of Glasgow

School of Computing Science

60 YEARS OF COMPUTING AT GLASGOW

Introduction

Advanced Operating Systems (M)

Lecture 1

Rationale

- Radical changes to computing landscape;
  - Desktop PC becoming irrelevant
  - Heterogeneous, multicore, mobile, and real-time systems – smart phones, tablets – now ubiquitous
- Not reflected by corresponding change in operating system design practice
- This course will...
  - review research on systems programming techniques and operating systems design;
  - discuss the limitations of deployed systems; and
  - show how the operating system infrastructure might evolve to address the challenges of supporting modern computing systems.

Aims and Objectives

- To explore programming language and operating system facilities essential to implement real-time, reactive, and embedded systems
- To discuss limitations of widely-used operating systems, introduce new design approaches to address challenges of security, robustness, and concurrency
- To give an understanding of practical engineering issues in real-time and concurrent systems; and suggest appropriate implementation techniques

Intended Learning Outcomes (1)

- At the end of this course, you should be able to:
  - Clearly differentiate the different issues that arise in designing real-time systems;
  - Analyse a variety of real-time scheduling techniques, and prove the correctness of the resulting schedule; implement basic scheduling algorithms;
  - Understand how to apply real-time scheduling theory to the design and implementation of a real-world system using the POSIX real-time extensions, and be able to demonstrate how to manage resource access in such a system;
  - Describe how embedded systems are constructed, and discuss the limitations and advantages of C as a systems programming language, understand how managed code and advanced type systems might be used in the design and implementation of future operating systems;

    …

Intended Learning Outcomes (2)

…

- Discuss the advantages and disadvantages of integrating garbage collection with the operating system/runtime, 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 runtimes to real-time systems and/or operating systems;
- 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;
- 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.

Course Outline

- Key topics in systems programming and operating systems:
  - Systems programming
  - Hardware trends
  - Memory management
  - Implications of concurrency
  - High performance networking
  - Real-time systems
  - Virtualisation
- Focus is on ideas and advanced concepts
  - Research topics and new approaches

Systems Programming

- What do we mean by systems programming?
  - Systems programs interact with hardware
  - Systems programs have memory and data layout constraints
  - Systems programs strongly driven by bulk I/O performance
  - Systems programs maintain long-lived, concurrently accessed, state
  - This is writing operating system kernels, low-level services, embedded systems, etc.
- Why is systems programming challenging?
- How are systems programs implemented? How should they be implemented?

Hardware Trends

- How is hardware evolving? How does this affect systems programming?
  - Implications of Moore's law and Dennard scaling
  - Implications of concurrency
  - Implications of solid-state storage
  - Implications of high performance networking

Memory Management

- Implications of NUMA and the caching hierarchy
- Memory management
  - Layout of a processes address space
  - Manual memory management
  - Region based memory management; ownership and borrowing; Rust
  - Garbage collection algorithms

Implications of Concurrency

- Memory models, threads, and locks
- Why threads and shared state concurrency are not sufficient
- Alternative concurrency models
  - Transactions
  - Message passing: Akka, Erlang, etc.

High Performance Networking

- Interactions between high-performance networking and Moore's law
  - Why high-performance networking is becoming more challenging
- APIs and programming models
  - Message passing
  - netmap and StackMap
  - User-space protocol stacks

Real-time Systems

- Definition of a real-time system
  - Scheduling theory
  - Need for formalisation
- Scheduling periodic systems
  - Rate monotonic algorithm
  - Earliest deadline first algorithm
- Scheduling aperiodic and sporadic tasks
  - Sporadic servers

Virtualisation

- What is virtualisation?
  - Hypervisors
  - Cloud computing
- Full system virtualisation
  - Xen
- Containers
  - FreeBSD jails, Docker, etc.
  - Unikernels

Timetable (1)

Week	Lecture	Subject
1		
	Lecture 1	Introduction
	Lecture 2	Systems Programming
2	Tutorial 1	Systems Programming and Alternative Operating Systems
	Lecture 3	Hardware Trends
	Lecture 4	Hardware Trends: Implications for Systems Programming
3	Tutorial 2	Hardware Trends and their Implications
	Lecture 5	Memory Management
	Lecture 6	Region-based Memory Management
4	Tutorial 3	Region-based Memory Management
	Lecture 7	Garbage Collection (1)
	Lecture 8	Garbage Collection (2)
5	Tutorial 4	Garbage Collection
	Lecture 9	Implications of Concurrency for Systems Programming
	Lecture 10	Managing Concurrency using Transactions

Timetable (2)

Week	Lecture	Subject
6	Tutorial 5	Implications of Concurrency for Systems Programming
	Lecture 11	Message Passing (1)
	Lecture 12	Message Passing (2)
7	Tutorial 6	Message Passing
	Lecture 13	Network Programming
	Lecture 14	High Performance Networking
8	Tutorial 7	Networking
	Lecture 15	Real-time Systems: Concepts and Scheduling Periodic Tasks
	Lecture 16	Real-time Systems: Scheduling Aperiodic and Sporadic Tasks
9	Tutorial 8	Real-time Scheduling
	Lecture 17	Virtualisation: Hypervisors
	Lecture 18	Virtualisation: Jails and Containers
10	Tutorial 9	Virtualisation
	Lecture 19	Wrap-up

Timetable (3)

- Note: frequent room changes in second half of the semester

Tutorials

- Level M course → assessing critical thinking skills; ability to read research papers, extract key insights
- Tutorials intended to facilitate this:
  - To provide space to discuss the Further Reading highlighted at the end of the lectures in the previous week, to consolidate learning, and emphasise key points of the material
  - You are expected to have read the highlighted papers, and to come to the tutorial prepared to discuss the material
  - Write your own summaries of the papers: what are the key concepts and ideas? what isn't clear? what's unimportant detail?
  - Discuss material that isn't clear in the tutorials → you're not expected to understand everything in the papers

Assessment

- Level M course; 10 credits
- Coursework (20%)

Exercise	Weight	Topic	Set	Due
1	15%	Region-based Memory Management	Lecture 6	Lecture 12
2	5%	Real-time Systems	Lecture 16	Lecture 18

- Examination (80%)
  - Two hours duration; sample and past papers are available on Moodle or the course website
  - All material in the lectures, tutorials, 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

Pre- and co-requisites

- Required pre-requisites:
  - Computer Systems 2
  - Operating Systems (H)
  - Advanced Programming (H)
  - Functional Programming (H)
- Recommended co-requisites:
  - Computer Architecture (H)

Required Reading

- No set text book – research papers will be cited:
  - DOIs are provided; resolve via http://dx.doi.org/ – all papers accessible for free from on campus (some may be paywalled from elsewhere)
  - You are expected to read the papers; it will be beneficial to follow-up on some of the references and do further background reading
    - Critical reading of a research paper is difficult and requires practice; read in a structured manner, not end-to-end, thinking about the material as you go; read claims carefully; realise that not everything written in a research paper is necessarily correct or a good idea – think and judge for yourself!
    - Advice on paper reading: http://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf
    - S. Keshav, “How to Read a Paper”, ACM Computer Communication Review, 37(3), July 2007 DOI: 10.1145/1273445.1273458
- The contents of the highlighted “Further Reading” research papers is examinable – but exam questions covering that material will focus on concepts, rather than on details

Resources and Contact Details

- Lecture slides and other materials are on Moodle
  - Also https://csperkins.org/teaching/2016-2017/adv-os/
  - Printed lecture handouts will not be provided – learning is enhanced by taking your own notes during lectures and tutorials
- Course coordinator:
  - Dr Colin Perkins, Room S101b, Lilybank Gardens
  - Email: colin.perkins@glasgow.ac.uk
  - No assigned office hours – email to make appointment if needed