University of Glasgow

School of Computing Science

Wrap-up

Advanced Operating Systems (M)

Lecture 9

Colin Perkins | https://csperkins.org/ | Copyright © 2018 | 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.

Lecture Outline

- Review of Rationale, Aims and Objectives, and the Intended Learning Outcomes
- Review of Lectures
- Summary and Discussion
- Assessment and Examination

Colin Perkins | https://csperkins.org/ | Copyright © 2018

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:
  - reviewed research on systems programming techniques and operating systems design;
  - discussed the limitations of deployed systems; and
  - showed how the operating system infrastructure might evolve to address the challenges of supporting modern computing systems.

BY ND

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

CC BY ND

Intended Learning Outcomes (1)

- 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;
  - …

CC BY ND

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.

CC BY ND

Review of Material

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

BY ND

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?

CC BY ND

Hardware Trends

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

CC BY ND

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

BY ND

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.

CC BY ND

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

CC BY ND

Virtualisation

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

CC BY ND

Summary and Discussion

- Computing systems are changing – hardware and usage changes
- Operating systems need to change too:
  - Mobile, Real-time, Embedded, Concurrent, Reliable, Secure
  - Is Unix the right basis for the operating system of the future? Unclear –many interesting new systems, approaches, languages, runtimes
  - Systems programming should move away from C
    - Tremendous potential in languages such as Rust for low-level systems and software – security and robustness benefits due to memory and type safety
    - Higher-level approaches for networked services/applications: message passing, “your server as a function” – significant productivity and safety benefits
- Significant ongoing changes – make sure you look beyond traditional approaches

CC BY ND

Assessment

- Coursework (20%)
  - Exercises that have been completed
- 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
  - 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

BY ND