

University of Glasgow | School of Computing Science

Systems Programming

Advanced Operating Systems (M)
Tutorial 4

Tutorial Outline

- Review of exercise 2
- Review of lectured material
- Discussion

Review of Exercise 2

- Question 1: A system comprises a set of independent periodic tasks, scheduled in a preemptive manner on a single processor, and one aperiodic task. The aperiodic task could be scheduled using a background server, or it could be scheduled using some form of bandwidth-preserving server, for example a deferrable server. Discuss what are the advantages and disadvantages of the bandwidth-preserving server [2 marks].

Review of Exercise 2

- Question 2: A system of three independent preemptable periodic tasks, $T_1 = (10, 2)$, $T_2 = (5, 0.5)$, and $T_3 = (6, 1)$ are scheduled using the earliest deadline first algorithm on a single processor. A deferrable server with period $p_s = 7.5$ and budget $e_s = 2$ is also present, to handle execution of aperiodic tasks. Is this system guaranteed to be schedulable? Explain your working. [4 marks]

Review of Exercise 2

- Question 3: A system of independent preemptable periodic tasks is scheduled in priority-driven manner on a single processor using the rate monotonic algorithm. In addition to the periodic tasks, sporadic jobs may arrive at any time, and are executed in earliest deadline order within a sporadic server task. In your own words, and without using mathematics, outline how the acceptance test for the sporadic jobs works [6 marks].

Review of Exercise 2

- Question 4: The fixed priority sporadic server has a very complex set of budget consumption and replenishment rules. Discuss whether you believe the benefits of this server outweigh its complexity [3 marks].

Review of Lectures

- Programming real-time and embedded systems
  - Interacting with hardware; interrupt and timer latency; memory issues; power, size and performance constraints; system longevity; development and debugging
- Possible evolution of systems programming
  - Language and runtime support for low-level programming: interrupt handling; device access; etc.
  - Language and runtime support for automatic memory management, including real-time garbage collection
  - Language and runtime support for real-time systems: periodic threads; timed statements/timing annotations
  - Language and runtime support for concurrency: type systems to ensure correctness; message passing; transactional memory

Key Learning Outcomes

- Understand how real-time and embedded systems are constructed
- Discuss the limitations and advantages of C as a systems programming language
- Understand how modern languages with advanced type systems might be used in the design and implementation of future operating systems

Discussion

- J. Shapiro, “Programming language challenges in systems codes: why systems programmers still use C, and what to do about it”, Proceedings of the 3rd workshop on Programming Languages and Operating Systems, San Jose, CA, October 2006, DOI 10.1145/1215995.1216004


Programming Language Challenges in Systems Codes
Why Systems Programmers Still Use C, and What to Do About It

Jonathon Shapiro, Ph.D.
Systems Research Laboratory
Department of Computer Science
Johns Hopkins University
shap@cs.jhu.edu

Abstract

1. Introduction

- Discussion points:
  - What are “systems programs”? Systems programs operate in constrained memory; are strongly driven by bulk I/O performance; performance matters; data representation matters; and retain state
  - Fallacies: factors of 2 don’t matter; boxed representation can be optimised away; the optimiser can fix it; the legacy problem is insurmountable
  - Challenges: application constraint checking; idiomatic manual storage; representation; state

- Is this a reasonable characterisation of the issues?

Any Further Questions?