University of Glasgow | School of Computing Science

Systems Programming

Advanced Operating Systems (M)
Tutorial 4

Tutorial Outline

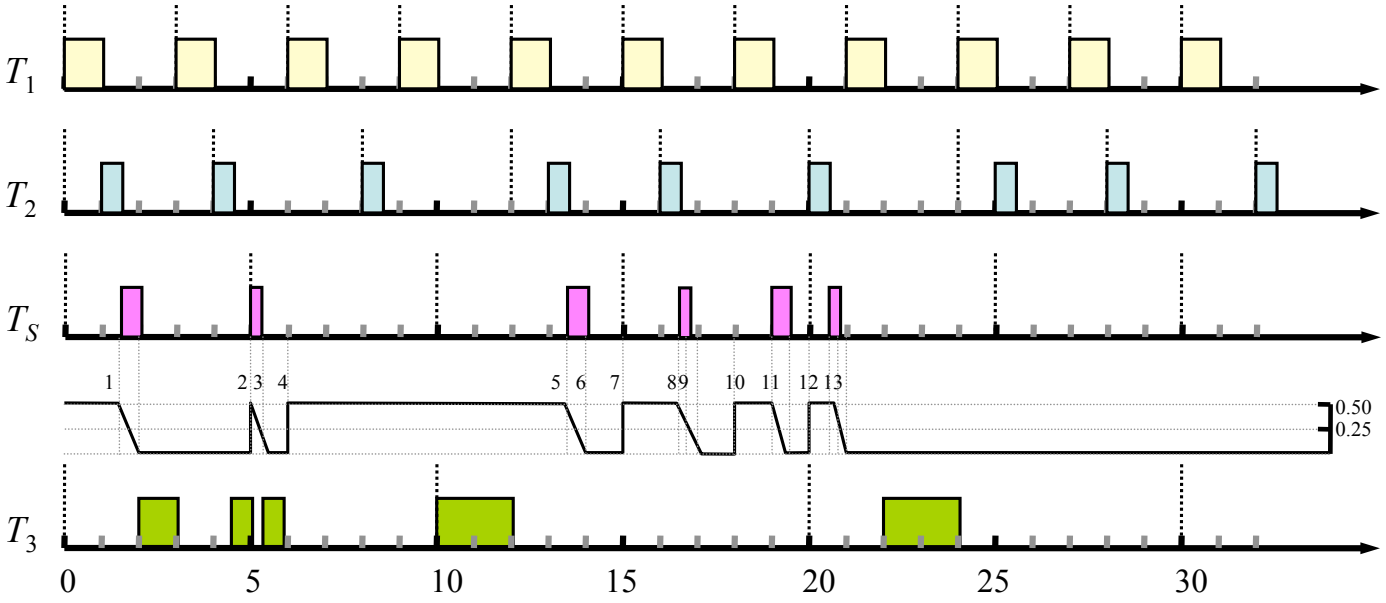
- Review of tutorial 3 sporadic server exercise
- Review of exercise 1
- Review of lectured material
- Discussion

Tutorial 3 Sporadic Server Exercise

- Formative exercise from Tutorial 3:
  - Consider a system of three periodic tasks: $T_1 = (3, 1)$, $T_2 = (4, 0.5)$, $T_3 = (10, 2)$. The system must support three aperiodic jobs:
    - A_1 which is released at time 0.5
    - A_2 which is released at time 12.25
    - A_3 which is released at time 17
  - The aperiodic jobs execute for 0.75 units of time. The system is scheduled using RM, with a simple sporadic server $T_s = (5, 0.5)$ supporting the aperiodic jobs.
  - Simulate the system for sufficient time to show how the aperiodic jobs are scheduled. What is the response time for each of the aperiodic jobs?

Tutorial 3 Sporadic Server: Worked Answer

1) C1; R2 ⇒ t_e = MAX(t_r, *BEGIN)* = 0; replenish at t_e+p_s = 5
2) Replenished due to previous R2; executes according to C1
   R2 ⇒ t_e = t_f = 5 since *END* < t_f; replenish at t_e+p_s=10
3) Job A_1 ends, but T_s continues according to C2
4) Replenished early due to R3(b)
5) C1; R2 ⇒ t_e = MAX(t_r, *BEGIN*) = 12; replenish at t_e+p_s=17
6) Budget exhausted (R3(a) does not apply, already replenished at step 4)
7) Replenished early due to R3(b)
8) C1; R2 ⇒ t_e = MAX(t_r, BEGIN) = 15; replenish at t_e+p_s=19
9) C2
10) Replenished early due to R3(b)
11) C1; R2 ⇒ t_e = MAX(t_r, BEGIN) = 18; replenish at t_e+p_s=23
12) Replenished early due to R3(b)
13) C1



Review of Exercise 1: Question 1

- Consider the following systems of independent preemptable periodic tasks that are scheduled on a single processor. Can these systems be scheduled using the Rate Monotonic algorithm or the Earliest Deadline First algorithm? Explain your answers
  - T_1 = (5,1), T_2 = (3,1), and T_3 = (15,3)
  - T_1 = (5,2), T_2 = (4,1), T_3 = (10,1), and T_4 = (20,3)

Review of Exercise 1: Question 2

- How does the schedulability test of Earliest Deadline First scheduling change if the relative deadline of a task differs from that task's period?

Review of Exercise 1: Question 3

- We considered several priority-driven scheduling algorithms. It was noted that these algorithms make locally optimal decisions about which job to run, but the resulting schedules are often not globally optimal.
- Explain the difference between locally and globally optimal, and discuss why priority-driven scheduling algorithms typically do not produce globally optimal schedules.

Review of Exercise 1: Question 4

- The periodic tasks T_1 = (3, 1), T_2 = (4, 2), and T_3 = (6, 1) are pre-emptively scheduled according to the rate monotonic algorithm on a single processor. Draw a graph of the time-demand function for each of the three tasks. Are these tasks schedulable? Justify your answer.

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

Jonathan Shapiro, Ph.D.

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?