University of Glasgow | School of Computing Science

Basics of Real-time Systems and Clock-driven Scheduling

Advanced Operating Systems (M)
Tutorial 1

Tutorial Outline

- Review of lectured material
- Worked examples
- Question and answer

Review of Lectured Material

- Lecture 1
  - Administration; introduction
- Lecture 2: introduction to real-time systems
  - Outline of terminology
  - Reference model
  - Hard and soft real-time systems
- Lecture 3: clock-driven scheduling
  - Concepts; static cyclic schedulers
  - Structured cyclic schedules: choosing the appropriate frame size
  - Slack stealing for aperiodic jobs; acceptance test for sporadic jobs
  - Practical considerations

Review of Lectured Material

- Key learning outcomes on real-time systems:
  - Understanding terminology; what is a real-time system?
  - Understanding importance of job scheduling; demonstration of timeliness
  - The ability to identify the jobs and tasks that form a system

Identification of Real-time Tasks

- Example: consider a hypothetical helicopter flight control system
- In each 1/180th second cycle:
  - Validate sensor data and select data source; on failure reconfigure system
  - Do the following 30-Hz avionics tasks, each once every 6 cycles:
    - Keyboard input and mode selection; data normalisation and coordinate transformation; tracking reference update
  - Do the following 30-Hz computations, each once every 6 cycles:
    - Control laws of the outer pitch-control loop; control laws of the outer roll-control loop; control laws of the outer yaw- and collective-control loop
  - Do each of the following 90-Hz computations once every 2 cycles, using outputs produced by the 30-Hz computations
    - Control laws of the inner pitch-control loop; control laws of the inner roll- and collective-control loop
  - Compute the control laws of the inner yaw-control loop, using outputs from the 90-Hz computations
  - Output commands to control surfaces
  - Carry out built-in-test

- What are the jobs and tasks in this example?

Clock-driven Scheduling

- Example – building a cyclic schedule:
  - Consider a system of independent preemptable periodic tasks, with no precedence or resource constraints, running on a single processor: $T_1 = (6, 2)$, $T_2 = (12, 3)$, and $T_3 = (4, 1)$
  - All jobs have phase equal to zero, and relative deadline equal to their period
  - Construct a cyclic schedule for the tasks, and show that the system meets all its deadlines

Clock-driven Scheduling

- When implementing a clock-driven scheduler, it's common to use a schedule based around a fixed frame size, rather than one with arbitrary job durations
- Why is a schedule based on a fixed frame size desirable?

Clock-driven Scheduling

- Example – frame sizes:
  - Consider a system of independent preemptable periodic tasks, with no precedence or resource constraints, running on a single processor: $T_1 = (6, 2)$, $T_2 = (12, 3)$, and $T_3 = (4, 1)$
  - All jobs have phase equal to zero, and relative deadline equal to their period
  - What would be an appropriate frame size for these tasks, if using a frame-based cyclic scheduler?

Question and Answer