Tutorial Outline

- Review of exercise 1
- Review of lectured material
- Discussion of papers
  - Why systems programmers still use C
  - Singularity

- Exercise 2 is due now
Review of Exercise 1 – Question 1

- Consider the following two 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.

  - \( T_1 = (6, 1) \), \( T_2 = (8, 1) \), and \( T_3 = (24, 5) \)
  - \( T_1 = (5, 2) \), \( T_2 = (4, 1) \), \( T_3 = (10, 1) \), and \( T_4 = (20, 2) \)
Review of Exercise 1 – Question 2

• A system contains three independent, preemptable, periodic tasks:
  • \( T_1 = (3, 1) \)
  • \( T_2 = (8, 2) \)
  • \( T_3 = (8, 3) \)

• Want to increase the period of \( T_3 \) so system can be scheduled using the RM algorithm

• What is minimum period necessary for the system to be guaranteed to be correctly scheduled?
• How does the maximum utilisation test for earliest deadline first scheduling change if the relative deadline of a task differs from that task's period?
We considered several priority-driven scheduling algorithms for real-time systems. These algorithms make *locally optimal* decisions about which job to run, based on the priorities of the runnable tasks when a scheduling decision is to be made, but the resulting schedules are often not globally optimal. Discuss why the resulting schedules are often not globally optimal.
Review of Exercise 1 – Question 5

- Periodic tasks $T_1 = (3, 1)$, $T_2 = (4, 2)$, and $T_3 = (6, 1)$ are scheduled in a pre-emptive manner using RM on a single processor. Draw a graph of the time-demand function for each of the three tasks. Can these tasks be scheduled? Justify your answer.
Review of Lectured Material

• Resource management protocols
  • Priority inheritance protocol – simple, but transitive blocking and potential deadlock
  • Priority ceiling protocol – reduced blocking and no transitive blocking, but requires a-priori knowledge of resource usage; must track system priority ceiling; avoidance blocking prevents deadlock
  • Stack-based priority ceiling protocol – further reduction in blocking if jobs never self-suspend; blocks jobs from starting until resources available
  • Maximum duration of blocking; operation in dynamic priority systems

• Real-time and embedded systems programming
  • Ensuring predictable timing
  • Device drivers – hardware interactions; options for improving robustness
  • System longevity; desire to improve robustness through alternate system implementation techniques
Key Learning Outcomes

- Understand operation of resource management protocols; trade off between different algorithms
- Understand differences between embedded and real-time systems and traditional desktop systems
  - Interactions with hardware
  - Desire for predictability rather than raw performance
  - Limitations of the traditional C-based programming model
Discussion of Papers


- Systems programming: constrained memory, I/O performance, data representation, state matters

- Fallacies: factors of 1.5–2 don’t matter; boxed representation can be optimised; the optimiser can fix it; legacy issues insurmountable

- Suggests: annotating code to check application constraints

- Suggests: manual but automatically checked storage management; explicit control over data representation

- The BitC project wasn’t a success, but are the ideas valid?
Discussion of Papers


• Use of strongly-typed languages to build an operating system; software isolated processes; message passing – is this a sound basis for the system?

• Type-safe message passing through channels; checked state machines for communication protocols (e.g., to control device driver state) – useful tool to help ensure correctness, or over-complex and stifling?

• Small unsafe microkernel, with type-safe system layered above – can the microkernel be written in a safe language?

• Threads and exchange heap; garbage collection – overheads?

• Is the idea of running everything in a virtual machine reasonable?