OBL3-OS

August 12, 2020

This is a mandatory assignment. Use resources from the course to answer the following questions. **Take care to follow the numbering structure of the assignment in your submission**. Some questions may require a little bit of web searching. Some questions require you to have access to a Linux machine, for example running natively or virtually on your own PC, or by connecting to `gremlin.stud.iie.ntnu.no` over SSH (Secure Shell). Working in groups is **permitted**, but submissions must be **individual**.

1 Synchronisation

1. The principle of process isolation in an operating system means that processes must not have access to the address spaces of other processes or the kernel. However, processes also need to communicate.
   (a) Give an example of such communication.
   (b) How does this communication work?
   (c) What problems can result from inter-process communication?
2. What is a critical region? Can a process be interrupted while in a critical region? Explain.
3. Explain the difference between busy waiting (polling) versus blocking (wait/signal) in the context of a process trying to get access to a critical section.
4. What is a race condition? Give a real-world example.
5. What is a spin-lock, and why and where is it used?
6. List the issues involved with thread synchronisation in multi-core architectures. Two lock algorithms are MCS and RCU (read-copy-update). Describe the problems they attempt to address. What hardware mechanism lies at the heart of each?

2 Deadlocks

1. What is the difference between resource starvation and a deadlock?
2. What are the four necessary conditions for a deadlock? Which of these are inherent properties of an operating system?
3. How does an operating system detect a deadlock state? What information does it have available to make this assessment?

3 Scheduling

1. Uniprocessor scheduling
   (a) When is first-in-first-out (FIFO) scheduling optimal in terms of average response time? Why?
   (b) Describe how Multilevel feedback queues (MFQ) combines first-in-first-out, shortest job first, and round robin scheduling in an attempt at a fair and efficient scheduler. What (if any) are its shortcomings?
2. Multi-core scheduling
   (a) Similar to thread synchronisation, a uniprocessor scheduler running on a multi-core system can be very inefficient. Explain why (there are three main reasons). Use MFQ as an example.
   (e) Explain the concept of work-stealing.