TENTAMEN / *EXAM*

TDDB68/TDDE47

Processprogrammering och operativsystem /
Concurrent programming and operating systems

2020-03-27 Part C

Instructions for part C (also see the instructions on the web page):

- Answer the questions below using a word processing tool of your choice and convert it to a PDF.
- Submit your solution in Lisam no later than 18.00.
- You are not allowed to copy text from other sources. Urkund will be used to check for plagiarism.
- Motivate answers clearly and elaborate your reasoning. You should demonstrate that you are able to *identify concurrency problems, design solutions to these problems and analyse trade-offs of design choices as* well as to *describe and analyse different requirements, functionality, and architectures for operating systems*.
- Keep to the subject and be precise in your statements. Elaborating on a topic does not mean adding random discussions with very long text. Except for the first question where the pseudocode might take up some space, it is recommended to keep the answer to *at most* one page per question.

1. **Synchronization** (3p)
   Large scale computation that makes use of massively parallelized work tasks contributed by volunteers has been used both in the search for extraterrestial intelligence (SETI@Home which is now sadly shutting down) as well as more recently in the search for a cure against Corona through Folding@home. In this assignment you should design a simple work distribution and collection mechanism with the following operations.

   - AddWorkTask(t) - This operation accepts a new work task t. The system has a maximum capacity of 1000 ongoing tasks so if the number of tasks in the system is at this limit, this operation should wait until there is free capacity.
   - FetchWorkToCompute() - This operation is invoked by a worker to receive a task to compute. Naturally, a task should be given to just one worker.
   - SubmitCompleteTask(t,result) - When a worker is done with a task, this operation is called to submit the result.
   - GetResult() - This operation should return the result of one finished task. Important: the order in which the tasks are returned by this task must be the same as the order they were originally added by the AddWorkTask operation. If the correct task is not yet finished, this operation should wait.

   Use the monitor mechanism to design your solution (in pseudocode). Motivate your design choices.

2. **Scheduling** (3p)
   Determining an appropriate scheduling policy for a given system can be difficult since there are many variables involved, both in terms of requirements, but also relating to differences in workload. For each of the system types, comment on how this particular domain impacts the choice of scheduling policies (algorithms) and why certain requirements might be more important that others in this particular case. For each case, suggest one scheduling policy that would at least be reasonable to use and argue why.

   - A smart phone
   - A web server serving hundreds of requests every second
   - A lawn mower robot

3. **Memory management** (3p)
   The so-called large page table problem has a number of different potential solutions, including the ones mentioned below.

   - Increasing the page size
   - Multi-level hierarchical page tables
   - Inverted page tables

   Comment on the trade-offs involved and compare these approaches with each other with respect to relevant metrics.