Operating Systems (5 op), Exam 14.8.2019

Write in each answer sheet course name, date, your name, signature and student id.

No calculators. Exam paper is 2-sided. For each question, it is sufficient to give a 1-2 page answer.
This is ordinary final (separate) exam and covers the whole course. Answer all questions 1-4.

1. [9 p] **Threads, concurrency**
    a. [3 p] Give a pseudocode example on a critical section problem.
       Give a scenario (A) relating to your example, where the result is correct.
       Give a scenario (B) relating to your example, where the result is incorrect. Explain, why it is incorrect.

    b. [2 p] Give a semaphore-solution to critical section problem. Explain, why scenario (B) is now not possible.

    c. [4 p] Explain the consumer-producer problem in the case with infinite buffer size. In what type of a situation could the solution be based on infinite buffer, even though in practice all buffers have finite size? Give a (pseudocode) solution to the infinite buffer producer-consumer problem, where synchronization is done with semaphores.

2. [9 p] **Deadlock**
    a. [3 p] Explain the Dining Philosophers Problem. Give a deadlock scenario for it. Explain why deadlock occurs in your scenario.

    b. [3 p] How does DDA (Deadlock Detection Algorithm) work in main principles?
       When and how is it used to solve the deadlock problem?

    c. [3 p] In operating systems kernel code deadlocks are often prevented by reserving critical sections always in some given order, e.g., in alphabetical order ”A B C D”.
       Why will the method prevent deadlock in all scenarios?
       In which case would order ”C B D A” be better than order ”A B C D”? Explain.

TURN

3. [9 p] **Virtual memory**
    a. [2 p] Page size is 4KB. How do you find the referenced data from main memory using 1-level virtual memory?
    As an example, assume that the code references byte address 0x11223344. Which main memory address is referenced?

    b. [3 p] What does "trashing mean"? What causes it? What is bad with it? How do you observe it in practice?
    When and how can you recover from it or prevent it? When can you not recover from it and what do you do then?

    c. [4 p] Clock and PFF (Page Fault Frequency) algorithms can both be used during the page fault processing. What problems do they solve, and how do the algorithms work in main principles?

4. [9 p] **Real Time Scheduling**
    a. [3 p] Why you usually can not use the same scheduling algorithms (e.g., FIFO or priority based systems) with real time systems than with ordinary systems? For exaple, why would FIFO be bad scheduling algorithm for real time systems?

    b. [3 p] However, Rate Monotonic Scheduling (RMS) is a priority based algorithm specifically intended for real time systems. When can you use it, and how does it work? How are the priorities defined?

    c. [3 p] What is the priority inversion problem sometimes occurring in real time systems? How can you solve it? (One solution method is sufficient)