Chalmers University of Technology and Gothenburg University

# Operating Systems
# EDA093, DIT 401
*Exam 2017-10-21*

*Date, Time, Place*: Saturday 2017/10/21, 14.00-18.00, Hörsalar på hörsalsvägen

*Course Responsible*: Vincenzo Gulisano (031 772 61 47), Marina Papatriantafilou (031 772 54 13)

*Auxiliary material*: You may have with you
- An English-Swedish, Swedish-English dictionary.
- No other books, notes, calculators, PDA's etc.

*Grade-scale ("Betygsgränser")*:
CTH: 3:a 30-39 p, 4:a 40-49 p, 5:a 50-60 p
GU: Godkänd 30-49p, Väl godkänd 50-60 p

*Exam review ("Granskningstid")*:
Will be announced after the exam.

*Instructions*

- Do not forget to write your personal number, if you are a GU or CTH student and at which program ("linje").
- Start answering each assignment on a new page; number the pages and use only one side of each sheet of paper.
- Write in a **clear manner** and **motivate** (explain, justify) your answers. If it is not clear what is written, your answer will be considered wrong. If it is not explained/justified, even a correct answer will get **significantly** lower (possibly zero) marking.
- If you make **any assumptions** in answering any item, do not forget to clearly state what you assume.
- The exam is organized in groups of questions. The credit for each group of questions is mentioned in the beginning of the respective group. Unless otherwise stated, all questions in a group have equal weight.
- Answer questions in English, if possible. If you have large difficulty with that and you think that your grade can be affected, feel free to write in Swedish.

**Good luck !!!!**

1. (12 p)

   (a) (4 p) In DMA transfers, who is in charge of orchestrating signals and provide addresses for the information being transferred? Justify your answer.

   **HINT: CPU orchestrates with DMA controller**

   (b) (4 p) Irqbalance is a tool to help balance the CPU load generated by interrupts across all systems CPUs. Assuming each CPU has a dedicated cache, why would this be beneficial?

   **HINT: higher chance the routing to serve the interrupt is already in cache**

   (c) (4 p) Describe the differences between port-mapped (direct) I/O and memory-mapped I/O. Mention the disadvantages each of the two incurs.

   **HINT: See slides for I/O lecture.**

2. (12 p)

   (a) (4 p) Provide two examples showing how logical and physical separation in the context of memory management is beneficial with respect to the transferring of information between fast and slow devices.

   **HINT: COW / solving external fragmentation / swapping of pages instead of processes / virtual memory / ....**

   (b) (4 p) What is the max page fault rate to have an effective access time lower than or equal to 0.6 ms if a memory access takes 0.3 ms, a page fault management takes 0.9 ms and the swapping out and in of a page take each 1.2 ms? What if dirty bits are also used?

   **HINT: Solve 0.6 ¿= 0.3 (1 - p) + p (0.9 + 1.2 + 1.2) Notice: 0.9 + 1.2 + 1.2 OR 0.9 + 1.2 OR 0.9 + 0.3 + 1.2 OR... as long as the formula is correct**

   (c) (4 p) Discuss why the second-chance page replacement algorithm is an approximation of the least recently used page replacement algorithm.

   **HINT: Not necessarily swapping out the least recently used, just swapping out one not used in the recent past.**

3. (12 p)

   (a) (8 p) Write a program in which the main method spawns three threads and uses them to compute the sum of all the values in a very large array. Note: the code does not need to compile. It should nonetheless contain the relative system calls we discussed in the course. Describe what each system call is for and discuss also what is the speed up of this program compared to one in which the sum is computed by the main thread.

   **HINT: To get 8 points, you needed to mention thread attribute initialization, thread creation and thread joining. Also, discuss why in this case it is basically OK to approximate the speed up as 3 (as long as your program is really allowing threads to run in parallel).**

   (b) (4 p) Compute the serial and parallel portion of an application that observes a speedup of 9.9 when using 10 rather than 1 core.

   **HINT: Solve correct equation based on Amdhal's law.**

4. (12 p)

(a) (4p) Which of the following scheduling algorithms could result in starvation? Explain why/under which circumstances. i. First-come, first-served ii. Shortest job first iii. Round robin iv. Priority

**HINT: sjf, strict priority**

(b) (8p) Consider a system where you know that the offered load consists of periodic real-time tasks and interactive processes. As a system designer you are able to decide on the scheduling policy to use. Discuss the design of two scheduling policy alternatives suitable for such a system. (i.e. discuss how you would think in order to decide on a policy to use, the advantages and problems of the alternatives you are considering, how these methods could be implemented, whether you could you make use of additional information )

**HINT:**
**one possibility: use EDF, with dedlines for interactive jobs = time of issue + maximum reasonable response time for the user (if available)**
**other: maintain 2 queues, one for RT, one for interactive, serve the RT first with EDF or RM, when empty move to interactive. If use EDF in the RT and the offered RT load (sum[(exec-time)/period] is less than 1, there will be time left for interactive ones, else, there may be starvation to interactive jobs.**
**alternative: insert the interactive jobs with fixed priorities in the same queue as the RT tasks. fixing those priorities would need knowledge of how critical it would be to miss an interactive process (i.e. to let it starve, or to eliminate it from the queue in case of congestion) and what are the tolerated response-time margings for these processes.**

5. (12 p) Consider the concurrent readers/writers problem: An object is shared among many threads, each belonging to one of two classes: *Readers*: read data, never modify it; *Writers*: read data and modify it. To provide proper synchronization for this problem, we need to (i) allow multiple readers to read concurrently, (ii) guarantee mutual exclusion for writers among themselves, (iii) guarantee mutual exclusion between readers and writers and (iv) guarantee progress and (v) preserve fairness.

   Consider the following pseudocode proposing a solution using semaphores. Does it satisfy the above requirements? If yes argue why; if not give examples how it may fail to do so.

```
1   semaphore sem_mutex, sem_w ; // init 1
2   int rc ; // init 0
3
4   Reader::
5      Wait(sem_mutex);
6      rc := rc + 1;
7      if rc = 1 then Wait(sem_w) fi;
8      Signal(sem_mutex);
9      [READ];
10     Wait(sem_mutex);
11     rc := rc - 1;
12     if rc = 0 then Signal(sem_w) fi;
13     Signal(sem_mutex)
14
15  Writer::
16     Wait(sem_w);
```

```
17      [WRITE];
18      Signal(sem_w);
```

**HINT: studied in the exercise session for classic synchronization problems, cf corresponding notes**