Chalmers University of Technology and Gothenburg University

# Operating Systems
# EDA093, DIT 401

*Exam 2022-10-22*

*Date, Time*: Saturday 2022/10/22, 08.30-12.30

*Course Responsible*:
Vincenzo Gulisano (031 772 61 47)

*Auxiliary material*: You may have with you
- An English-Swedish, Swedish-English dictionary.
- No other books, notes, calculators, 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) (8 p) Write the C (pseudo-)code of a program in which one process counts how many times a certain number is found within a given array. The following constraints must be enforced:

   - there are N threads counting the elements
   - none of these N threads is the one that initially runs the main method
   - the thread that runs the main method is the one that finally outputs to the user how many times a certain number is found within the given array
   - the N threads cannot run concurrently nor in parallel. They can only run in order, one after the other: first thread 1 runs, then thread 2 runs, ..., then thread N runs. It is possible for the threads to iterate in a circular fashion. That is, after thread N runs, then thread 1 can run again, then thread 2 can run again, and so on.

   [**HINT:** Possible solution: the main thread spawns N threads and there's an array of N+1 binary semaphores shared with them. The first is set to 1, the others to 0. The extra semaphore can be used for the "searching thread" to say to the main thread the search is over. Every thread waits on its semaphore and signals the next one. All threads can operate on a shared variable for the count. Other solution, start one thread after the other has completed.]

   (b) (4 p) Which ones of the following conditions *needs* to be broken to avoid a deadlock? Motivate your answer.

   - Mutual Exclusion
   - Hold and wait
   - No preemption
   - Circular wait

   [**HINT:** Your answer needs to clearly show that you understand that any of them is OK, and breaking 1 is enough. If you said "except mutual exclusion" I decided to assume that is OK, but technically any of them is OK, the question is about avoiding a deadlock...]

2. (12 p)

   (a) (4 p) Provide at least three distinct *and detailed* examples of the benefits of separating logical and physical memory addresses.

   [**HINT:** E.g. process memory not bounded to physical memory, reduced fragmentation, copy-on-write... the list can go on...]

   (b) (6 p) Assume you have a disk of 100Mb (total physical space) and that the block size is 256Kb. Show a setup (indicating which blocks are used by which file) in which:

   - 10 files of 9 Mb each are stored with contiguous allocation
   - external fragmentation does not allow storing a file of 2 Mb

   [**HINT:** If you leave some space before the first file, in between files, and after the last file, you can have 11 holes each of approx 10/11 Mb each, which means there's no single hole with size 2Mb. of course it is also enough to just have each file followed by 1 Mb, that's even simpler...]

   (c) (2 p) Can you experience internal and external fragmentation at the same time? Motivate your answer.

   [**HINT:** Yes, contiguous allocation **on disk** with too-large block size (blocks are used with the disk).]

3. (12 p)

   (a) (12 p) True or false? Motivate your answers in detail

   - User-level threads are not beneficial if one of the parallel threads requests I/O.
   - Prioritizing I/O-bound processes over CPU-bound processes can improve the average completion time of such I/O- and CPU-bound processes.
   - Schedulers might rely on caches' retained data to improve process execution performance.
   - How much schedulers can rely on caches' retained data depends on the number of available CPUs and caches.
   - Non-preemptive schedulers with priority could deadlock spinning-based synchronization mechanisms.
   - 2 threads belonging to 2 different processes cannot read concurrently from the same frame.

   [**HINT:** Notice: correct answer without explanation means 0 points. Correct answers followed by text that is not about what should be explained means 0 points. Tautological correct answers and variations of that (e.g., "True, because schedulers might rely on caches' retained data to improve process execution performance") means 0 points. That said...
   False: if the other threads depend on the data requested by the I/O thread they are as beneficial as kernel ones.
   True: example shown in class.
   True: that's why a scheduler tries to reallocate a thread to the same core it was running before.
   True: higher CPUs higher chances of them being connected to different caches and/or higher chances of assigning processes to different cores. I can accept False too if the reason is "it's a previous executions problem, not an hardware one".
   True: exemplified in class.
   False: think about CoW.]

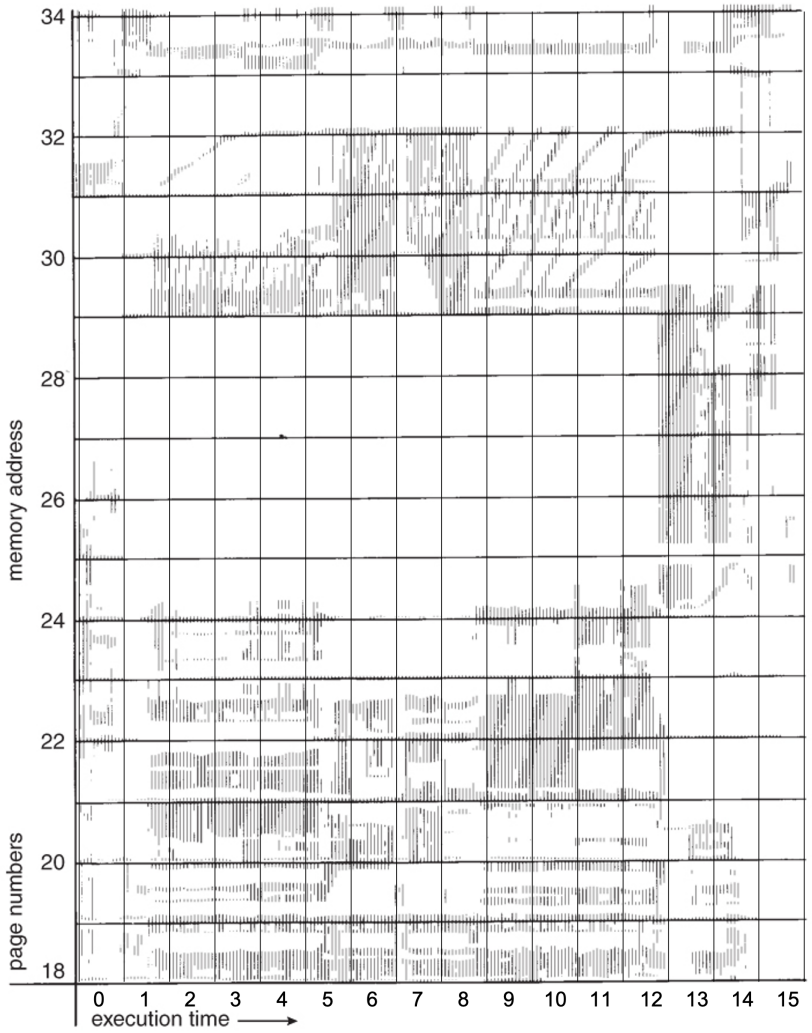
4. (12 p)

   (a) (6 p) Choose 3 random words in English, each word should have at least 5 letters. Write down your words and using the following conversion table, create your reference string. Does your reference string suffer from Belady's anomaly when using 4 rather than 3 frames? Motivate your answer. Please note: the chances of two people randomly picking the same 3 words in the same order are basically zero...

   [**HINT:**
   Notice: Belady's anomaly is for FIFO. If you try something else... 0 points.
   Also, 1 reference string, not 3 different reference strings... otherwise it is too easy! Why would I ask you to repeat the same exercise 3 times with 5 pages for 4 frames?
   Not counting page faults when the frames are free (only for replacements) means 0 points]

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 1 | 5 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 6 | 3 | 4 | 5 | 2 | 3 | 4 | 6 | 2 | 1 | 5 | 6 | 7 | 8 |

(b) (3 p) Consider the access to pages shown in the following picture.




Assuming the first locality starts at time 0 and that a locality changes when at least 2 pages that were not heavily requested in the current locality start being accessed, how many locality changes do you observe in the figure?

[**HINT:** The question is left a bit vague on purpose (heavily requested). I would say the localities start at 1 (pages 29 and 30), at 5 (30 and 31), at 9 (23 and 24 stars being requested again), and at 13 (25-28). I have accounted for the various interpretation of "heavily requested" each one had in my correction.]

(c) (3 p) What is the difference between a page table and a shadow page table?

[**HINT:** page table virtual memory, shadow page table related to the hypervisor.]

5. (12 p)

(a) (4 p) How does a one-way hash chain work? What is the benefit of using it?

[**HINT:** Check the relevant slides from the security lecture. In short P1 n-times hashing of s, P2 (n-1)-times hashing of s,... can check if p(n+1) is valid by hashing it can compare with p(n) but cannot go the other way. Can use n different **one-time** passwords.]

(b) (4 p) What is the advantage of having the set of sensitive instructions contained in the set of privileged instructions? Explain.

[**HINT:** Check the relevant slides from the security lecture. In short: allows to safely virtualize VMs/OSes because behavior of sensitive might be different depending on user/kernel mode otherwise, so OS might not trap to hypervisor.]

(c) (4 p) What are the benefits of DMA? Explain how it works in detail.

[**HINT:** Check the relevant slides from the I/O lecture. Reduces interrupts for memory transfer. The exact procedure is discussed in a slide during the course.]