Chalmers University of Technology and Gothenburg University

Operating Systems
EDA093, DIT 401

Exam 2023-10-21

Date, Time: Saturday 2023/10/21, 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. (36 p)

   (a) (4p) Given two file systems A and B using the same hard drive and containing the same files, A has internal fragmentation equal to 10% while B has internal fragmentation equal to 15%. Which of the two file systems is expected to read/write faster? Explain your answer.

   [**HINT:** B, because high fragmentation implies larger blocks, which implies faster reads/writes.]

   (b) (4p) Can you experience both internal as well as external fragmentation at the same time? Explain your answer.

   [**HINT:** Yes, for a file system with contiguous allocation and large (enough) blocks.]

   (c) (4p) Michael performs the login into a computer, enters the home directory of Jessica, and sees all the files there have "rwx" set on "other". Will it be possible for Michael to read all those files? Explain.

   [**HINT:** It depends on whether Michael is the owner of some of those files, belongs to the group associated with each file, or not.]

   (d) (4p) Make a specific/concrete example of an application in which several threads executing different portions of the code of such an application would perform better when executed through user-level rather than kernel-level threads. Reason in terms of some specific performance metric.

   [**HINT:** Many possible answers (e.g., streaming) but be clear on some performance metric that is not simply fairness among threads as the OS would do.]

   (e) (4p) Mario is sad because many of his processes are I/O bound and his CPU utilization is low. To increase his CPU utilization, Mario decides to stop using magnetic tapes for his file system and instead buys a very fast SSD. Is Mario going to experience higher CPU utilization because of this? (only Mario's processes are executing, the majority of his processes are I/O bound, no other changes are made to the hardware, and the rest of the hardware is modern, a performant computer you can buy in 2023).

   [**HINT:** Yes, I/O bursts will take less time.]

   (f) (4p) Is the CPU utilization a good indicator of whether the number of processes run in parallel/concurrently by an OS is adequate? Motivate your answer.

   [**HINT:** not necessarily, think about thrashing.]

   (g) (4p) List at least three mechanisms designed to reduce the time spent handling the page faults of a process instead of running the process itself.

   [**HINT:** Dirty bit, dedicated interrupt, proportional dynamic allocation.]

   (h) (4p) Is it possible to execute a program concurrently on a machine with a single CPU and a single core, using only kernel threads? If yes, how? Motivate your answer.

   [**HINT:** Yes, multiple processes, each with the kernel thread.]

   (i) (4p) Explain how one-time passwords work based on the one-way hash chain mechanism.

   [**HINT:** See relevant slides in the security lecture.]

2. (12p) Write a reference string that, using the LRU page replacement algorithm, and given 5 frames usable by the process invoking page requests according to your string (the frames are initially empty!) results in at least 20 page faults. Note that the following must hold:

(a) each page is accessed at least 2 times

(b) two pages A and B accessed one after another cannot be so that $B = A+1$

(c) after each multiple of 5 timeslot that is greater than 0 (that is, if timeslots start at 0 it means at 5, 10, 15, ...) it turns out that another process has stolen the first frame of this process, but this process is allowed to reclaim it

[**HINT:** There can be many answers, of course, but 1,1,3,3,5,5,7,7,9,9,...,21,21 will do: each page is accessed twice, no consecutive pages are so that B=A+1. Every time a new page is asked there's a page fault, so 20. No matter what the other process steals, page faults will not decrease, so still 20.]

3. (12p) Imagine you want to create an enhanced file system that uses compression and customizes what is read/written to blocks to minimize internal fragmentation. Your file system extends a file system that already exists and that already has mechanisms in place to handle the writing/reading of blocks. Describe what events your enhanced file system would need to intercept, what it would do on such events, and what extra supporting data structures it would rely on.

   [**HINT:** One event to intercept would be a block write. Then you would compress the block, find an existing block that, within it, has enough free/unused space to write the compressed block, and update a translation table/free blocks table as needed. The second event would be to the block read: read the translation table find the actual block/offset to read, read, decompress, and give back. DSs: translation table that maps blocks the user wants to actual blocks/offsets that contain the compressed version. Free blocks/portions of blocks/offsets.]