Chalmers University of Technology and Gothenburg University

Operating Systems
EDA092, DIT 400
Exam 2017-01-14

Date, Time, Place: Saturday 2017/01/14, 08.30-12.30, 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) Discuss (provide examples too) why a certain application might decide to spawn new processes (instead of threads) to run some tasks while another application might decide to spawn new threads (instead of processes) to run some tasks.

   (b) (4 p) *"Failure to join with a thread that is joinable [...] produces a "zombie thread". Avoid doing this, since each zombie thread consumes some system resources, and when enough zombie threads have accumulated, it will no longer be possible to create new threads (or processes)"* (from man `pthread_join`).

      i. What is a zombie process?

      ii. What system resources it consumes and why it could be no longer possible to create new threads or processes?

   (c) (4 p) If a certain process runs 3 times faster when using 10 cores instead of 1 core, can you compute how big its serial portion is? Motivate your answer and show how the result is computed.

   HINT:
   1.a. processes = modularity / safety. threads = perfomance / cost
   1.b.i child terminates but not parent waiting for it
   1.b.ii e.g. return value kept for missing parent. Because e.g. OS might allow only up to X processes consuming resources at the time.
   1.c Amdhal. $1/(X-(1-X)/10)=3$**. X = 13/33 = 40%**

2. (12 p)

   (a) (4 p) *"Protection strategies have a weakness in common: they rely on the integrity of the underlying operating system. Consequently, malware subverting the system's initialization steps remains undetected. A bootkit executes early during the boot process, long before the OS protection mechanisms kick in, allowing the bootkit to retain control throughout the infected system's boot phase"*[1].

      i. Make an example of a protection strategy that relies on the OS.

      ii. Describe how a bootkit could execute early during the boot process and elaborate on why this would be dangerous.

   (b) (4 p) Discuss why the utility of a tool such as defragmentation depends on the hardware being used and the OS implementation.

   (c) (4 p) Provide one scenario in which both internal and external fragmentation can be observed at the same time.

   HINT:
   2.a.i system calls;
   2.a.ii Being in the MBR. E.g. modifying other file systems.
   2.b contiguous allocation + rotating disk = useful, ssd + linked list allocation= not really
   3.c.i contiguous allocation (external) with too large block size (internal)

3. (12 p)

   (a) (3 p) After creating B (a copy of process A) by running `fork`:

[1] Sentence taken and cut from "Nice Boots! - A Large-Scale Analysis of Bootkits and New Ways to Stop Them, by Grill at al. DIMVA 2015".

i. why are A's and B's virtual addresses still valid despite the fact that they are the same?

ii. what mechanism would make possible for A's and B's virtual addresses to be equal also in terms of physical addresses?

iii. discuss when it would be safe (and when not) for A and B to access the same physical address.

(b) (6 p) Does the access sequence given by the reference string *dcbadcedcbae* suffer from Belady's anomaly when using 4 instead of 3 frames? (say what Belady's anomaly is and show how the answer is checked).

(c) (3 p) Specify whether the following sentences are true or false (motivating why).

i. If the valid/invalid bit is set to invalid, the referenced frame does not belong to the process.

ii. If the dirty bit is the to 1 (i.e., dirty) there is no need to swap out the page (if the latter is chosen as target).

iii. The functionality of the dirty bit could be achieved using the valid/invalid bit. For instance, by setting it to invalid whenever the referenced page is dirty.

HINT:
3.a.i Virtual start at 0 for all processes, translated to different physical addresses;
3.a.ii COW
3.a.iii OK as long as READ, concurrent write(s) and read(s) problematical.
3.b Belady's anomaly = more frames and nonetheless more page faults. YES
3.c.i false, can belong but is different frame
3.c.ii false, dirty means it has been modified, hence must be written
3.c.iii nope. if invalid the frame would be retaken from disk, nothing to do with swapping it out.

4. (12p)

(a) (2 p) Describe 3 of the goals of the process (or thread) scheduler in an operating system.

(b) (3 p) Connecting to these goals, explain possible actions/methods for the scheduler to achieve them.

(c) (3 p) Explain one trade-off that the scheduler needs to balance, wrt to at least 2 of the above goals. Explain how the trade-off can be balanced.

(d) (4 p) Explain the role of the data-structure that the operating system uses for the ready-queue, in order to facilitate priority scheduling. Provide an example and discuss about the benefit versus the overhead of having it.

HINT:
a. Maximize cpu-utilization, Min overhead, Max throughput, Max fairness (can be eg. minimize maximum turnaround time)
b. Examples: Utilization: keep CPU busy, avoid context switches, increase # processes (bonus point if connection to thrashing is discussed). For multiprocessors, do work-stealing.

Min overhead: avoid context switches, use efficient data structures for book-keeping. For multiprocessor systems avoid process migration
Throughput: go for eg shortest process first (best throughput)
Fairness: use dynamic priorities
c. Examples: Throughput-fairness trade-off: when using strict priorities the low-priority processes/threads may starve. Small time-slice in RR benefits fairness but decreases throughput. Dynamic time-slice; priorities with aging; multi-queue feedback scheduling.
In multicores, utilization-overhead trade-offs exist as well: to migrate or not to migrate; (bonus points if synchronization issues are mentioned and how they introduce other trade-offs related to processor affinity; gang scheduling)
d. Priority queue/heap of sorted tasks can facilitate the choice of the nect process/thread to dispatch. Maintaining a priority list is useful. Tree-data structures are also possible (eg Linux "completely-fair scheduler" does that). Instead of searching for the highest-priority every time, maintaining the heap gives the next task to execute in constant time, while the OS can run the maintenance of the tree in the background (eg in parallel on another core).

5. (12 p)

   (a) (2 p) Describe the critical section problem and the properties required from a solution to the problem.

   (b) (5 p) Describe Lamport's bakery algorithm for solving the critical section problem among N processes/threads. Write the pseudocode, explain the main idea and argue about its properties.

   (c) (5 p) Consider managing the procedure of job printing. Issuing a print request by a process P (in a set $\mathcal{P}$), implies that a printer job (e.g. a pointer to a file that is to be printed) is stored in a buffer, whose maximum capacity is K elements. A process C in control of interacting with the printer, needs to access those jobs and take care of printing them. Describe a solution to the problem of communication between the issuing and the printing processes, using semaphores. Argue about its properties.

HINT: Slide number refer to set of lectures on synchronization
a. slide 5;
b. slides 31-34 (extra bonus if correctly using methodology in slides 11, 12;
c. it is a producer-consumer problem with bounded buffer; solution outline slides 37-38.