Chalmers University of Technology and Gothenburg University

Operating Systems
EDA092, DIT 400

Exam 2015-08-18

Date, Time, Place: Tuesday 2015/08/18, 14:00−18:00, "Väg och vatten"-salar

Course Responsible: Vincenzo Gulisano, Marina Papatriantafilou

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) (4p) Discuss the benefits of multithreaded applications with respect to single-threaded ones.

   HINT: Responsiveness, Resource Sharing, Economy, Scalability. Please refer to slide 12 of lecture 3.

   (b) (4p) Describe the different multithreaded models for the relationship between user and kernel threads.

   HINT: Many-to-one, one-to-one, many-to-many. Please refer to slides 20-23 of lecture 3.

   (c) (4p) Given the code below, please answer the following questions (motivating why you answer true or false).

```c
#include <pthread.h>
#include <stdio.h>

int sum;
void *runner(void *param);

int main()
{
   pthread_t tid;
   pthread_attr_t attr;

   pthread_attr_init(&attr);
   pthread_create(&tid,&attr,runner,argv[1]);
   pthread_join(tid,NULL);

   ptrintf("sum = %d\n",sum);
}

void *runner(void *param) {
   int i, upper = atoi(param);
   sum = 0;

   for(i = 1; i <= upper; i++)
      sum += i;

   pthread_exit(0);
}
```

   i. The final value of sum can change at different executions.
   ii. The variable sum is not shared.
   iii. The main function can complete before the created thread completes.
   iv. `pthread_join` might find the thread has already completed when called in the `main` function.

   HINT: false (main waits for thread to finish and sum is modified only by the thread), false (it is shared by both main and thread), false (because of `pthread_join`), true (main thread might be slower because of several reasons.

2. (12 p)

   (a) (3p) Discuss why is Virtual Memory in place in existing operating systems and which are its benefits.

HINT: separate user logical memory from physical one. Benefits: allows for a process to be executed even if the latter is not entirely in main memory (e.g., allows for programs larger than memory)

(b) (3p) Given the logical and physical memory configurations presented in Figure 1 (taken from Operating System Concepts, Silberschatz, 2013), and supposing each time a page-fault occurs, the missing page is loaded in a free slot in the physical memory, how many page-faults will occur if all pages from A to H will be accessed by the process? And if all valid-invalid bits are set to "i"?.

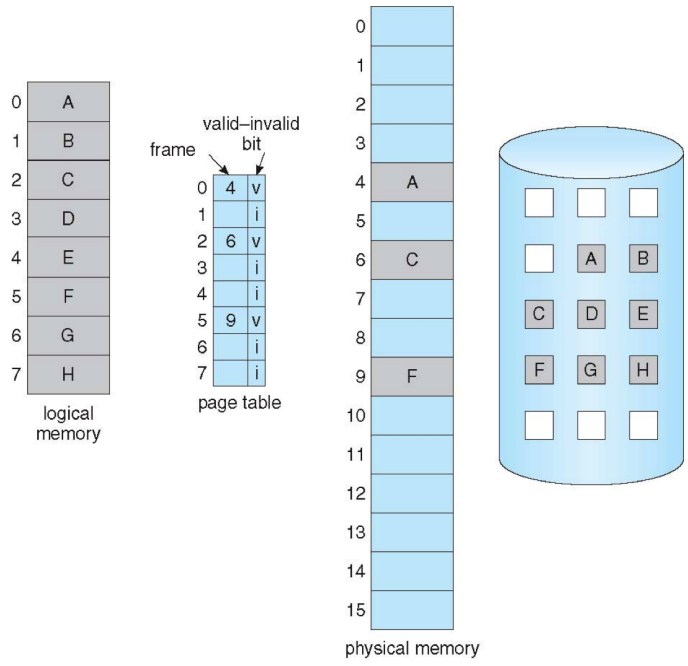

Figure 1: Sample logical and physical memory configurations

HINT: 1) each page not loaded results in a page-fault, so 5. 2) each page results in a page-fault, so 8

(c) (3p) If the memory access time for a page in memory is 0.1 milliseconds while it grows to 10 milliseconds in case of a page fault, which is the Effective Access Time (EAT) if the probability of observing a page fault is 5%?

HINT: EAT = 0.1*(1-0.05)+10*0.05 = 0.595 milliseconds

(d) (3p) What is the Belady's anomaly?

HINT: the possibility of observing more page faults for an increased number of frames for FIFO page replacement.

3. (12 p)

(a) (3p) Discuss the main benefit of a virtual file system.

HINT: To define a layer that separates different file systems (possibly with different interfaces) and provides an unified interface for the user

(b) (3p) Given the following list of files and read / write / execute permissions for owner, group and other:

- *r w x r - x r w x* *userB* *groupC* *file1*
- *r w x - - x - w x* *userA* *groupA* *file2*
- *r w x - - x - - -* *userC* *groupB* *file3*

answer the following questions (assuming userX belongs to groupX):

- Which files can be deleted by userA?
- Can an user belonging to groupB delete file1?
- Can *userB* execute *file3*?

HINT: file1 and file2; if he/she is userB yes, if he/she also belongs to groupC no, otherwise yes; yes

(c) (3p) What does the Process Control Block contain?
HINT: Please refer to page 10 of lecture 2

(d) (3p) Are the registers and stack of a process' PCB unique or thread-specific for a multithreaded process? Why?
HINT: thread-specific, because each thread can execute different parts of the code concurrently or in parallel.

4. (12 p)

(a) (2+4+4 (10) p)
(i) Regarding First-Come First-Serve scheduling, a main drawback is that convoy effects may appear. Explain what this means.
(ii) Regarding Shortest-Task-First scheduling, explain why it eliminates the convoy effect and give an argument why it minimizes the average waiting time among the tasks. What is the main drawback of the method?
(iii) Regarding Round-Robin scheduling, explain how it connects to the advantages and disadvatanges of the above methods. Explain also the trade-off in the choice of the length of the time-quantum.
HINT: (i) convoy: short-task waits for long ones; (ii) no longer short tasks wait for long ones; hence waiting time minimized; argument visualized on slide 15 in scheduling; (iii) it balances; short quantum better for fairness, but more expensive in context switching.

(b) (2p) Explain one of the main issues that makes scheduling in multiprocessor/multicore systems a more complex problem compared to single-processor scheduling.
HINT:shared versus per-processor; load balancing vs utilization; no-processor-affinity can result in large migration costs; processor affinity can be best for frequently synch-ing threads

5. (12 p)

(a) (6 p) Show a method that solves the mutual exclusion problem for arbitrary number of threads using the atomic TestAndSet instruction that is available in several processor architectures. Use pseudocode in the description and argue about the properties of the solution. (It is not necessary to describe a solution that guarantees fairness in this question, but if you can, of course it is ok).
HINT: pseudocode line 16 in synchronization slides; mutex is guaranteed because only one among competeing threads can suceed in TAS; not fair solution, can cause a thread to starve (fair solution on slide 18; not necessary to write it here for the answer to be correct)

(b) (6p) Design a solution to the mutual exclusion problem for arbitrary number of threads in a system where the *atomic instruction* called FetchAndAdd, whose functionality is described below, is available by the hardware. This solution needs to guarantee fairness. Use pseudocode in the description and argue about the properties of your solution.

```
int FAA(int *counter)
  *counter =*counter+1;
  return(*counter)
```

HINT: simulate lamport's bakery algo with FAA for ticket and main serving counter. threads enter CS in order of increasing tickets