TENTAMEN / *EXAM*

TDDB68

Processprogrammering och operativsystem /
Concurrent programming and operating systems

2017-06-05

Examiner: Mikael Asplund (0700895827)

Hjälpmedel / *Admitted material:*

– Engelsk ordbok / *Dictionary from English to your native language*;
– Miniräknare / *Pocket calculator*

General instructions

- This exam has 6 assignments and 8 pages, including this one.
  Read all assignments carefully and completely before you begin.
- Please **use a new sheet of paper for each assignment**, because they will be corrected by different persons.
  Sort the pages by assignments and number them consecutively.
- You may answer in either English or Swedish. English is preferred because not all correcting assistants understand Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- **Motivate clearly** all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 40 points. You may thus plan about 5 minutes per point.
- **How much to write?** No general policy, but as a rule of thumb: Questions for 0.5p can typically be answered properly in a single line. Correct and concise answers to questions for 1p usually require a few lines. Code and figures should be commented properly.
- Grading: U, 3, 4, 5. The preliminary threshold for passing is 20 points.

Good luck!

1. **Multiple choice quesions (10p)**

   Below are 10 multiple choice questions. Please answer them by removing the last page of the exam, fill the appropriate boxes, and hand it in together with the rest of your answers. Please note that there might be more than one correct option. Each question below can give 0 or 1 point(s), and to get 1 point you must have identified exactly the right set of choices.

   (a) Consider the program below, called *fork-test*.

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
  int pid1 = 0;
  pid1 = fork();
  int pid2 = 0;
  pid2 = fork();
  printf("PID1: %d, PID2: %d\n", pid1, pid2);
  return 0;
}
```

   Which of the lines below could have been produced by the *first* process that was started when running fork-test?

   A) PID1: 0, PID2: 0
   B) PID1: 0, PID2: 28252
   C) PID1: 28250, PID2: 0
   D) PID1: 28250, PID2: 28251

   (b) Which of of the following events will take place when switching from one process (old) to another (new).

   A) Changing the set of currently open files to match with the new process.
   B) Copying data from the process control block (PCB) of the new process into the interrupt vector table.
   C) Storing CPU register states into the PCB of the old process.
   D) Updating registers relating to memory management (e.g., base address pointer).

   (c) Assume a single-CPU system and the following set of processes with arrival times (in milliseconds), expected maximum execution time (ms), and priority (1 is highest, 5 is lowest priority). The time quantum is 2ms, and to break a tie between two processes arriving simultaneously to the queue, you can assume that a newly arrived process will be put first.

Process	Arrival time	Execution time	Priority (as applicable)
P_1	0	6	5
P_2	1	3	2
P_3	5	1	4
P_4	6	3	3
P_5	8	2	1

Which of the scheduling policies below could result in the following execuction trace?

$$P_1, P_1, P_2, P_2, P_1, P_1, P_2, P_3, P_4, P_4, P_1, P_1, P_5, P_5, P_4$$

A) FIFO
B) Round-robin
C) Shortest-job first with preemption
D) Priority scheduling with preemption

(d) Which of the following mechanisms will *prevent* deadlocks?

A) Only use safe resources
B) Always acquire resources according to a globally defined order
C) Ensure that $\forall j : R_j \geq \sum_i D(i,j)$, where R_j is the number of instances of resource j and $D(i,j)$ is the maximum demand of resource type j from process i.
D) Only request a resource when it is really needed.

(e) Which of these parameters will affect the effective access time (EAT) of demand paging?

A) Page fault rate (probability of page fault)
B) Segment table length
C) Time to swap page in from disk
D) Disk size

(f) Which of the terms below refer to methods used for allocation of disk space for a file?

A) Virtual allocation
B) Indexed allocation
C) Segment allocation
D) Contiguous allocation

(g) Which of the following concepts relate to multiprocessor scheduling?

A) Segmentation
B) Task migration
C) Hard vs. soft affinity
D) Co-scheduling

(h) Given a virtual memory system with 4 page frames, how many page faults occur with the *Least-Recently Used* replacement strategy when pages are accessed in the following order:

1, 2, 3, 4, 5, 1, 3, 4, 2, 3, 1, 5, 4.

A) 7
B) 9
C) 11
D) 13

(i) Virtualization has been an enabler for cloud-based services. Which of the following statements are true regarding this technique?

A) Full virtualization allows multiple operating system instances running on a single machine.

B) A computer with an Intel architecture and a full virtualization solution can only run a guest OS built for an Intel architecture.

C) Paravirtualization requires support in the guest OS.

D) Virtualization removes the need for having priviledged instructions in the CPU.

(j) Wihch of the following mechanisms reduce the impact of buffer overflow attacks?

A) Marking certain parts of the data as non-executable (e.g., non-executable stack, non-executable pages)

B) Arrays with built-in bound checks

C) Making sure that the source and destination buffers are of appropriate length when copying memory

D) Validate user input before processing it

2. **Synchronization (8p)**

A *doubly ended queue (deque)* D is a data structure that is a combination of a stack and a queue. It provides the following operations:

isempty(D) – returns 1 if D contains no items, and 0 otherwise

pushFront(D, v) – insert item v at the front of D;

popFront(D) – removes and returns item v from the front of D if D is nonempty, and returns NULL otherwise;

pushBack(D, v) – insert item v at the back of D;

popBack(D) – removes and returns item v from the back of D if D is nonempty, and returns NULL otherwise;

Assume that each element in the deque is represented using the following struct.

```
struct elem {
  item_type v;
  struct elem *next;
  struct elem *prev;
}
```

Your task is to provide pseudocode for the popFront method. The code should be thread-safe, meaning that concurrent invocations to any of the deque operations can be performed safely. You should define the deque data structure and modify the elem struct as you see fit.

(a) Provide a solution that uses a *single* lock or semaphore. (2p)

(b) Provide a solution that allows operations to be performed on the back and front of the queue without one operation blocking the other. Naturally, this can only be done if the queue is long enough (as otherwise the front and back of the queue will interfere). (4p)

(c) Another alternative solution is to use *lock-free* algorithms. Explain this concept and provide at least one advantage of lock-free algorithms compared to traditional blocking algorithms. (2p)

3. **Deadlocks (6p)**

(a) Most current operating systems do not implement the Banker's algorithm for deadlock avoidance but instead shift this task to the application programmer. Name 2 limitations of the Banker's algorithm that are the main reason for this. (1p)

(b) How can the occurrence of a deadlock be *detected* when only one instance of each resource type exists in a system? (1p)

(c) Consider the following resource allocation problem in a system with 3 resources (R1-R3), and 4 processes (P1-P4). The table indicates the currently allocated resources and in parenthesis the maximum possible demand.

	R1	R2	R3
P1	0 (3)	1 (3)	0 (2)
P2	0 (2)	1 (1)	3 (3)
P3	1 (3)	0 (0)	0 (3)
P4	0 (0)	0 (2)	0 (0)

The currently available resources are: [3, 5, 0]. Use Banker's algorithm to determine if the request [2, 0, 0] from Process P3 should be granted. (4p)

4. **Processes and scheduling (6p)**

(a) Define the terms *process* and *thread*.
In particular, what are the main differences between processes and threads, and what do they have in common? Be thorough! (2p)

(b) We introduced several multithreading models (i.e., mapping between user-level and kernel-level threads). Describe one model that is appropriate for multiprocessor systems, and explain why. (2p)

(c) Draw the general life cycle diagram (finite state machine) for a process in a system with *preemptive scheduling*, as introduced in the lecture. For each state (node) explain shortly what it represents. For each possible state transition (arrow) annotate which event(s) trigger(s) the transition. (2p)

5. **Memory Management (6p)**

(a) Which kind of fragmentation (external or internal) can occur in contiguous memory allocation? Explain your answer. (1p)

(b) Given a single-level paged memory system with a translation lookaside buffer (TLB). What is the effective memory access time if the physical memory access time is 90ns, a TLB lookup takes 10ns and the average TLB hit rate is 90%? Explain your calculation. (2p)

(c) Paging for large virtual address spaces and small page sizes leads to the large-page-table problem. Explain *one* technique how this problem can be solved. Be thorough! (3p)

6. **Security (4p)**

   (a) A good security design allows separating the *mechanism* from the *policy*. Explain why, and provide an example (from the IT domain, not the house in the slides). (2p)

   (b) Describe the difference between a vulnerability and an attack. Explain using a realistic example. (2p)

Intentionally empty page.

Multiple choice form for answering question 1. Please put X:s in the appropriate cells:

	A	**B**	**C**	**D**
1 a)				
1 b)				
1 c)				
1 d)				
1 e)				
1 f)				
1 g)				
1 h)				
1 i)				
1 j)				