TENTAMEN / *EXAM*

TDDB68

Processprogrammering och operativsystem /
Concurrent programming and operating systems

2017-03-18, 14:00–18:00

Jour: Mikael Asplund (0700895827); visiting ca. 15:30

Hjälpmedel / *Admitted material:*

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

General instructions

- This exam has 6 assignments and 7 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. How many times will it outout the line “Hello”?

   ```
   #include <stdio.h>
   #include <unistd.h>
   int main()
   {
     fork();
     printf("Hello\n");
     fork();
     printf("Hello\n");

     return 0;
   }
   ```

   A) 2 times
   B) 3 times
   C) 4 times
   D) 6 times

   (b) Which of the following alternatives are potential mechanisms for inter-process communication (IPC):

   A) Shared memory
   B) Interrupt
   C) Mailbox
   D) Dispatcher

(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 the following memory management tasks can be performed by the MMU:

A) Memory protection

B) Page table lookup

C) Page replacement

D) TLB lookup

(f) Which of the following items are found in a POSIX inode for a regular file?

A) File size

B) Pointer to the directory that the file is located in

C) Pointers to blocks on disk where the file is stored

D) The file position indicator (as set by an fseek command)

(g) Which of the following are security attributes?

A) Integrity

B) Protection

C) Authentication

D) Availability

(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) Which of the following virtualization configurations are possible when using *paravirtualization*:

A) Host running Ubuntu Linux on a x86-64 machine, Guest running Windows 10 for x86-64
B) Host running Ubuntu Linux on a x86-64 machine, Guest running Linux for ARM
C) Host running Windows 10 on a x86-64 machine, Guest running Ubuntu Linux for x86-64
D) Host running Ubuntu Linux on an IA-32 machine, Guest running Ubuntu Linux for x86-64

(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 (6p)**

   (a) Consider system that interacts with a medical device over a communication bus. The communication between the device and the control system is made through a special interface which has the following API:

   - `void initialize()` - This function initialises the device, should be done once during system startup.
   - `void sendCommand(c)` - Sends the command `c` to the device.
   - `bool checkStatus()` - Checks wether the most recent command was successful or not. Will return `OK` or `NOK`. The status will typically not be available until 100ms after the command was sent, but will always be available within 500ms. This function will block until the status is available.
   - `int readValue(v)` - Read value of variable `v`.

   Note that this API is not thread safe. Since only one function at time can access the bus, if multiple function calls (including `readValue`) are made concurrently the behaviour is undefined. During the time between a command is sent and the status is read it is allowed to invoke the `readValue(v)` function, but not the `sendCommand(c)` function.

   Write pseudocode for the two functions below (using either semaphores or a monitor):

   - `bool sendCommandTS(c)` - Sends the command `c` to the device, returns the status of the command `OK` or `NOK`.
   - `int readValueTS(v)` - Read value of variable `v`.

   As opposed to the original API, these functions should be thread safe (hence the TS in the names). That is, it should be possible to invoke them from concurrent processes/threads, without race conditions. Note that since the `sendCommandTS(c)` function can take a long time to return, your implementation must allow concurrent calls to the `readValueTS(v)` function (hint: you can make use of the `sleep` system call).

   (4p)

   (b) Explain (and illustrate with an example scenario) how mutual exclusion synchronization can interfere with priority based scheduling.

   (2p)

3. **Deadlocks (6p)**

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

   (b) What is the difference between *deadlock* and *starvation*? (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 (0)	0 (2)	0 (2)
P2	0 (3)	1 (1)	2 (3)
P3	0 (1)	3 (5)	0 (1)
P4	0 (1)	4 (4)	1 (1)

   The currently available resources are: [3, 0, 3]. Use Banker's algorithm to determine if the request [1, 0, 0] from Process P4 should be granted. (4p)

4. **CPU Scheduling (6p)**

   (a) Describe the concept of a time quantum. Explain the trade-offs involved when determining an appropate value for this parameter. Give one example of a system where a small time quantum is preferrable, and one example when a relatively larger time quantum is better. (3p)

   (b) What is a Multilevel-Feedback-Queue scheduler? Describe how it works and the reasons for using such a scheduler in a general-purpose operating system. (2p)

   (c) Describe the concept of turnaround time in the context of CPU scheduling. (1p)

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

   (a) Define the term *external fragmentation* (of memory), and give an example of a memory management technique (technical term, no details) that completely avoids external fragmentation. (2p)

   (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) What is *thrashing* in a virtual memory system? How does it occur? And what can be done about it? (2p)

6. **File systems (6p)**

   (a) Describe the file allocation method *indexed allocation* and briefly discuss its strengths and weaknesses. (2p)

   (b) How does the file system implementation keep track of unused disk space? Sketch one possible technique for free-space management. (2p)

   (c) Sometimes it may become necessary to run a file system consistency check. What does "inconsistency" of a file system mean, and what could possibly have caused it? (2p)

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)				