TENTAMEN / *EXAM*

TDDB68

Processprogrammering och operativsystem /
Concurrent programming and operating systems

2017-03-17, 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 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) Which of the following operations will result in a system call for a normal operating system?

   A) Reading the value of a variable stored on the stack
   B) Creating a new file
   C) Allocating memory on the heap
   D) Performing a jump (JMP) instruction

   (b) Which of the following events will *always* trigger an interrupt (hardware or software).

   A) A user program requesting to read data from a file.
   B) A user program requesting to read data from main memory.
   C) A key being pressed on the keyboard.
   D) A byte being transferred from disk to main memory.

   (c) Which of the following properties should be ensured according to the *critial section problem* as defined in the course book?

   A) Mutual exclusion
   B) Lock-freedom
   C) Progress
   D) Availability

   (d) Which of the following statements are true about deadlocks:

   A) If there is only a single instance of every resource, a cycle in the resource allocation graph means that there is a deadlock.
   B) All four Coffman conditions must me met for there to be a deadlock.
   C) Banker's algorithm is used to detect and remove deadlocks.
   D) Banker's algorithm guarantees freedom from starvation.

   (e) Which of the following statements are true?

   A) Different threads belonging to the same process share memory with each other.
   B) Each thread must have its own stack.
   C) Threads are guaranteed to be deadlock-free.
   D) Threads can only be implemented by the operating system.

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

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

(g) Which of the following statements are true about the TLB?

A) It is an abbreviation for "Transitive Loop Buffer"
B) It is used to reduce the average memory access time
C) It is used to reduce the worst-case memory access time
D) It is implemented in the kernel

(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 can be said to constitute vulnerabilities

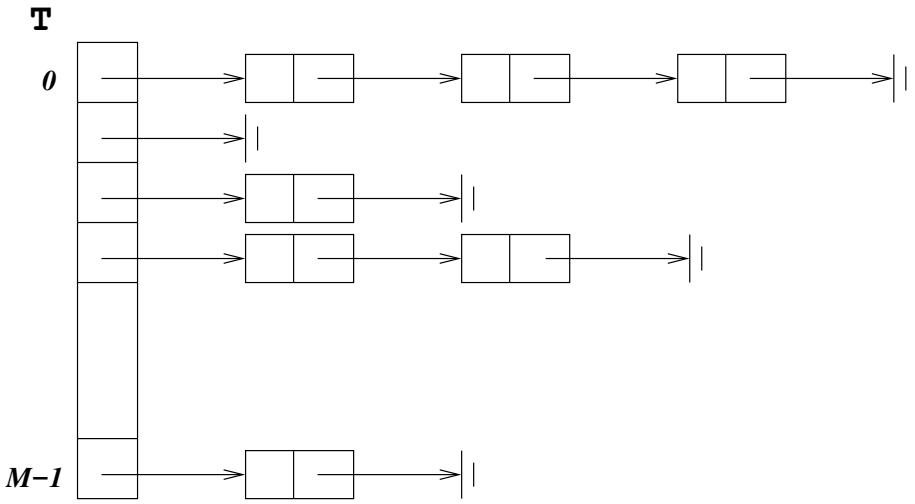
A) Lack of input validation in a program
B) An email with a virus attached
C) The Android operating system
D) Clear-text storage of passwords

(j) 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

2. **Synchronization (6p)**

Consider an ordinary *hash table* T of size M designed for a single-threaded environment. A hash function h that maps data items to the range $\{0, ..., M-1\}$ is given. Data items u, v hashed to the same hash value $h(u) = h(v) = i$ are chained in a singly-linked list, such that list $T[i]$ contains all items with hash value i (see the picture). Each list item contains a data element and a `next` pointer to the next item in the list (NULL for the last element in a list). Each $T[i]$ points to the first item in its list (or $T[i]$ is NULL if the ith list is empty), $i = 0, ..., M-1$.



A sequential implementation is given as follows. List elements are represented by the following data type:

```
struct htelem {
   struct dataitem *item;
   struct htelem *next;
};
```

The operation `insert`(T, u) inserts a data item u into hash table T at the beginning of list $T[h(u)]$:

```
void insert ( struct htelem *T[], struct dataitem *u )
{
  int i = h(u);
  struct htelem *e = (struct htelem *) malloc(sizeof(struct htelem));
  e->item = u;
  e->next = T[i];
  T[i] = e;
}
```

The boolean operation `lookup`(T, u) checks whether item u is currently stored in T or not:

```
int lookup ( struct htelem *T[], struct dataitem *u )
{
  int i = h(u);
  struct htelem *p;
  for (p = T[i]; p!=NULL; p = p->next )
     if (equal( u, p->item ))
        return 1;
  return 0;
}
```

where the boolean function `equal`(u, v) tests for equality of two data items u and v.

(a) Give an example scenario of a *race condition* with two threads that concurrently insert items into an unprotected shared hash table on a multi-tasking single-processor system with preemptive scheduling. I.e., give two different interleavings of the execution of two threads that lead to incorrect behavior, possibly even to a corrupted data structure or a run-time error. (1p)

Your task will now be to make the hash table implementation thread-safe to allow a hash table stored in shared memory to be accessed properly by multiple threads:

(b) Identify the critical section(s) in your pseudocode and suggest how to protect them properly against race conditions by using a single mutex lock. Show the revised (pseudo) code. (2p)

(c) What is the disadvantage of a single-lock solution if there are many threads accessing T frequently, and why? (1p)

(d) Describe a multi-lock approach that solves that problem. (1p)

(e) Threads executing an `insert` operation modify the status of T, while threads executing a `lookup` do not change it. Assume that `lookup` operations occur much more often than `insert` operations. Suggest a lock-based protection technique that exploits this fact to improve the degree of concurrency compared to your previous solution. (No details, no code. Just give the technical term and the basic idea how it works.) (1p)

3. **Deadlocks (6p)**

(a) Explain the difference between deadlock prevention and deadlock avoidance. Give an example of each type of mechanism. (2p)

(b) 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 (1)	0 (0)
P2	2 (3)	1 (3)	2 (3)
P3	1 (1)	0 (3)	0 (2)
P4	0 (1)	4 (5)	0 (0)

The currently available resources are: [0, 3, 2]. Use Banker's algorithm to determine if the request [0, 2, 0] from Process P2 should be granted. (4p)

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

(a) Write a simple Unix-style program (pseudocode using appropriate system calls) that *spawns exactly two (2) child processes*, each of which shall write `''Hello World''` to the standard output, and that writes `''Goodbye''` after the two child processes have terminated. Explain your code. (2p)

(b) Two main methods for inter-process communication in a computer are *shared memory* and *message passing*. Which of the two methods is likely to have less overhead if two processes communicate frequently with each other, and why? (2p)

(c) 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)

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

   (a) How does a simple segmented memory management system work, and what hardware support should be provided by the memory management unit (MMU)? (2p)

   (b) How can segmentation and paging be combined? (1p)

   (c) LRU is a popular strategy for page replacement in virtual memory. However it is just a heuristic technique. What is the (theoretical) *optimal* page replacement strategy, why is it not applicable in practice, and how does it differ from LRU? (3p)

6. **Security (6p)**

   (a) What is a buffer-overflow attack? Describe the vulnerability, the factors that contribute to it, and give a scenario of how it can be exploited by an attacker to hijack the control of a running server program. (3p)

   (b) Describe and explain the three main security attributes. (3p)

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)				