Tentamen i TDDD82 Säkra mobila system (Systemprogramvara)

2017-06-09, kl. 14-18

- Inga hjälpmedel är tillåtna.
- Kom ihåg att svaren på samtliga uppgifter måste MOTIVERAS, och att motiveringarna skall vara uppställda på ett sådant sätt att det går att följa hur Du tänkt. OMOTIVERADE SVAR GER 0 POÄNG OM INGET ANNAT SÄGS.
- Jour: Mikael Asplund (nåbar på tel. 0700-895827).
- Maxpoäng är 30 poäng. För betyg 3 krävs minst 15 poäng, för betyg 4 krävs 20 poäng och för betyg 5 krävs 25 poäng.

Lycka till!!!

1. **Electronic voting server**

   An electronic voting system uses as central data structure a global array

   ```
   unsigned int votes[N];    // N = number of parties
   ```

   for counting the number of votes for each of the N parties, initialized to zeroes.

   The voting server program, originally single-threaded, processes one vote request received from an external client at at time, by calling the function

   ```
   void cast_vote ( unsigned int i )
   {
     votes[i] = votes[i] + 1; // count vote for party i
   }
   ```

   In order to scale up to many millions of voters, the voting program should be multithreaded (with the `votes` array residing in shared memory of the voting server process) and run on a multicore server running a modern multithreaded operating system with preemptive scheduling, so that multiple calls to `cast_vote` can execute concurrently. It is your task to make the program thread-safe and preserve its correct behavior.

   (a) Using a simple contrived scenario (Hint: use $N = 2$ parties and few votes), show with a concrete example (interleaving of shared memory accesses) that, without proper synchronization, race conditions are possible that could even cause that the wrong party wins the election.

   1 point

   (b) Identify the critical section(s) and protect the program against race conditions with a *single* mutex lock. Show the (pseudo)code.

   2 points

   (c) Now extend your implementation of (b) by adding the shared global variable

   ```
   int current_leader = 0;  // init. to 0 at program startup
   ```

   that shall, at any time, contain the *index* of the currently leading party. Routine `cast_vote` needs to be extended to check if the just incremented vote counter exceeds `votes[current_leader]` and, if so, update `current_leader` accordingly. Show the resulting `cast_vote` pseudocode and explain how this change affects the critical section compared to (b).

   2 points

(d) Now add a *separate* thread that is responsible only for continuously monitoring `current_leader` and, in case of a change in the leader, updating a graphical display of the *currently leading party*. A naive solution using busy polling of `current_leader` could use a thread function like this one:

```
void *leader_monitoring ( void * arg )
{
 int last_observed_leader = 0;
 while (1) { // endless loop for continuous monitoring:
    if (current_leader != last_observed_leader) { // change:
       last_observed_leader = current_leader;
       update_display( current_leader );
    }
 }
 return NULL;
}
```

Suggest a more suitable solution (using a special synchronization construct, *mention its name and explain what it does*) such that the leader-monitoring thread does not waste CPU time by busy polling of `current_leader`, and such that no display updates are done that any new vote doesn't make necessary. (Assume that it is not critical if the update of the leader visualization is delayed a bit, or if a short-time change in the leader and back is not displayed.) *Show and explain* the resulting modified pseudocode of `cast_vote` and of `leader_monitoring`.

4 points

2. Construct a resource allocation graph with four processes and four resource instances such that

(i) the graph has a cycle and the processes on the cycle are deadlocked;

(ii) the graph has a cycle and the processes on the cycle are *not* deadlocked.

3 points

3. 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	2 (2)	0 (2)	0 (0)
P2	0 (6)	0 (9)	0 (1)
P3	0 (2)	1 (7)	0 (0)
P4	0 (2)	0 (1)	0 (0)

The currently available resources are: [4, 8, 2]. Use Banker's algorithm to determine if the request [4, 5, 1] from Process P2 should be granted.

4 points

4. Consider a server that stores and distributes the latest messages delivered by satellite communication from a nuclear reactor disaster area. The clients that are outside the area are thereby connected to the clients in the area via the server. Decide which of the following properties is a functional property and which is an extra-functional property that relates to quality of service. Motivate your answer!

   (a) An actor outside the area can ask for the messages received in the past 30 minutes to be dispatched at one go by the server.

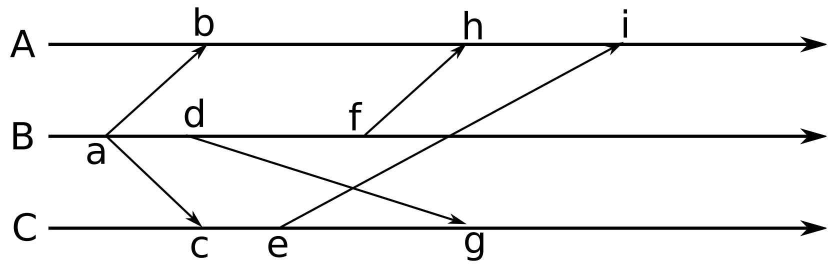
   (b) The messages can be sorted in length and the actor outside the area may ask for the shorter messages to be delivered by the server if there are bandwidth limitations.

   (c) The system should provide a throughput of at least 2kb/s between any two clients connected with the lowest bandwidth.

   (d) To avoid a single point of failure the server should be replicated with a cold-passive failover mechanism.

4 points

5. Consider the three timelines for nodes A, B and C in the figure below. The arrows between the lines indicate messages as they are sent and received by the respective nodes. Use Lamport's locgical clock algorithm to set timestamps for the events a-i.



What can you conclude from these timestamps regarding the following statements:

- $d \rightarrow e$
- $a \rightarrow f$
- $f \rightarrow h$

where $\rightarrow$ indicates the *happened-before* relation?

5 points

6. Use the terminology from IFIP Working Group 10.4 to analyse the fault-error-failure chain in the example below (The Guardian, June 2, 2017). Classify the fault as permanent/transient/intermittent.

> An investigation into the power outage that caused chaos for British Airways over the bank holiday weekend is likely to focus on human error rather than any equipment failure.
>
> A contractor doing maintenance work at a BA data centre is said to have inadvertently switched off the power supply, knocking out the airline's computer systems, according to a report in the Times.
>
> Quoting a BA source, the newspaper said the power supply unit that prompted the IT failure was working perfectly but was accidentally shut down by a worker. Business Today: sign up for a morning shot of financial news Read more
>
> BA had to cancel all flights from London's Heathrow and Gatwick airports last Saturday, leaving 75,000 passengers stranded. It blamed a power surge that knocked out its computer system, disrupting flight operations, call centres and its website.

5 points