Tentamen i TDDD82 Säkra mobila system (Systemprogramvara)

2017-03-03, kl. 08-12, Sal TER3

- 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. Consider system that interacts with a medical device. 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. That is, if multiple function calls 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 `sendCommand(c)` function can take a long time to return, your implementation should allow concurrent calls to the `readValue(v)` function.

   6 points

2. Describe the four Coffman conditions. How do these conditions relate to deadlock prevention?

   5 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	1 (1)	0 (1)	2 (4)
P2	0 (0)	0 (8)	0 (4)
P3	0 (1)	5 (5)	1 (1)
P4	0 (0)	3 (8)	1 (2)

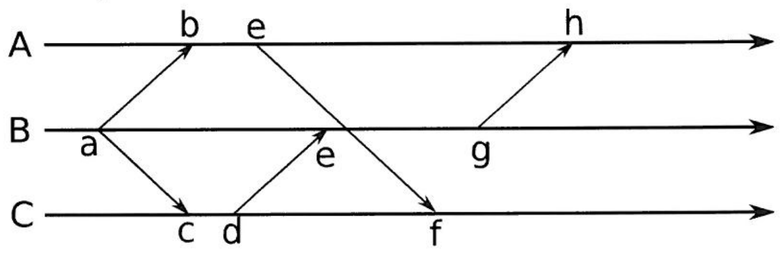
The currently available resources are: [1, 0, 5]. Use Banker's algorithm to determine if the request [0, 0, 4] from Process P2 should be granted.

4 points

4. Explain the concept of horizontal distribution. Use an example to clarify. Also explain how this concept relates to fault tolerance and replication.

5 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-h.



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

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

where $\to$ 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 (Computer Business Review February 1st 2017). Classify the fault as permanent/transient/intermittent.

   > GitLab has currently been taken offline after suffering a major backup restoration failure following an incident of accidental data deletion.
   >
   > The source-code hub released a series of tweets following the incident, one of which confirms the failure: "We accidentally deleted production data and might have to restore from back-up." This included a link to a Google Doc file with live notes.
   >
   > The data loss took place when a system administrator accidentally deleted a directory on the wrong server during a database replication process. A folder containing 300GB of live production data was completely wiped.
   >
   > GitLab said: "This incident affected the database (including issues and merge requests) but not the git repos (repositories and wikis)."
   >
   > It was identified that out of the 5 backup techniques deployed, none had either not been working reliably or set up in the first place. The last potentially useful backup was taken six hours before the issue occurred.
   >
   > However, this is not seen to be of any help as snapshots are normally taken every 24 hours and the data loss occurred six hours after the previous snapshot which results to six hours of data loss.

5 points