Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-08-29
Sal (1)	T1(4)
Tid	14-18
Utb. kod	TDDD82
Modul	TEN2
Utb. kodnamn/benämning **Modulnamn/benämning**	Projekttermin inklusive kandidatprojekt: Säkra, mobila system Systemprogramvara: Skriftlig tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	6
Jour/Kursansvarig Ange vem som besöker salen	Mikael Asplund
Telefon under skrivtiden	0700 895 827
Besöker salen ca klockan	15-16
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Annelie Almquist 013-28 29 34
Tillåtna hjälpmedel	Inga
Övrigt	
Antal exemplar i påsen	

Tentamen i TDDD82 Säkra mobila system (Systemprogramvara)

2019-08-29

- 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.
- Ansvarig: 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. **Synchronization** (7p)

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).

2. **Processes** (3p)

(a) How does an operating system prevent a process from monopolizing a processor, and what hardware support is required for that? (1p)

(b) 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 (2p)

state transition (arrow) annotate which event(s) trigger(s) the transition.

3. **Deadlocks** (5p)

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

The currently available resources are: [1, 1, 1]. Use Banker's algorithm to determine if the request [0, 0, 1] from Process P1 should be granted.

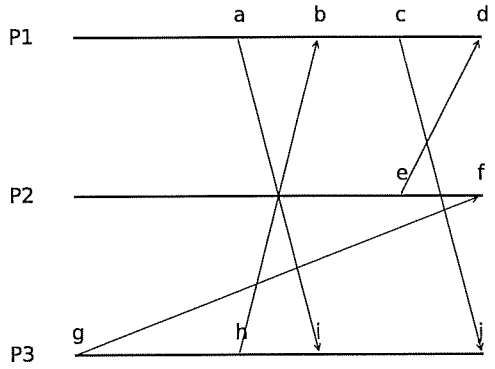
4. **Quality of Service** (6p)

(a) Describe the concept of generalised processor sharing (GPS) and how an approximation of this concept can be realised in a networking context. Also explain why a perfect realisation of GPS is not possible for a network router. (4p)

(b) Explain the Shortest Job First scheduling algorithm. (2p)

5. **Distributed systems** (5p)

Consider the three timelines for nodes P1, P2 and P3 in the figure below. The arrows between the lines indicate messages as they are sent and received by the respective nodes.



(a) Use Lamport's logical clock algorithm to set timestamps for the events a-j. (2p)

(b) Let $C(e)$ denote the Lamport clock for event e. For each bullet point below, either identify a pair of events (from events a-j) that match the requirements, or explain why they do not exist. (3p)

- $e_1 \rightarrow e_2$ and $C(e_1) < C(e_2)$
- $e_1 \rightarrow e_2$ and $C(e_1) > C(e_2)$
- $e_1 \nrightarrow e_2$ and $C(e_1) < C(e_2)$

6. **Dependability** (4p)

Explain the concepts of *availability* and *reliability*. Also explain how each of these attributes can be measured over time.