TTK4145 - Exercise 1

Buadu, Saagfors

January 12, 2017

Problem 1

- **What if the software controlling one of the elvators suddeny crashes?** What if the software controlling one of the elevators suddenly crashes? If the software controlling one of the elevators crashes, there must be a back-up process that can take over the primary actions. When the back-up in a sense becomes a primary, there must be created a back-up for the back-up.

- **What if it doesn't crash, but hangs?** If the software doesn't crash, but hangs... If we can confirm that the software actually hangs and hasn't crashed we could restart the process. However if this is time consuming one might want to transfer the elevators orders to the backup in the mean time.

- **What if a message between machines is lost?** One way to ensure that messages lost between machines aren't lost is to have two step messaging system. Elevator 1 sends a message Elevator 2 receives the message. Elevator 2 sends a conformation to Elevator 1 that the message is received. If elevator 1 doesn't receive the confirmation within reasonable time, the message is sent again. We must also implement something to ensure that the same order isn't received twice by a machine.. It might only be the confirmation that was lost, not the original message.

- **What if the network cable is suddenly disconnected? Then re-connected?** If a network cable is disconnected messages sent from one machine to another will never reach its destination. Hence, the message will be resent several times. When the cable is re-connected the same order will have been sent many times. Therefore some sorts of limitation to how many times the same message can be sent must be implemented. What should be done then is to be determined.. Either send the message to another machine or machine deals with the problem itself.

- **What if a user of the system is being a troll?** We would implement some sort of state machine. In a given state only some actions are possible to make until you have exited the state. If the button is already pressed, pressing it again will not change the state of the system. An elevator going up with a pit-stop, should start traveling in the opposite direction if that is the order given when the elevator is at the pit-stop.

- **What if the elevator car never arrives at its destination?** If waiting for an elevator: If an order is not completed withing reasonable time, the order will be transferred to the back-up. If someone is in the elevator: ...

Problem 2

A repository in GitHub has been created.

Problem 3

- What is concurrency? What is parallelism? What's the difference? Concurrency is form of multitasking. Two or more processes can be running at the same time on one or more machines. The atomic actions making up each of the processes might not run at the exact same time, but the whole of the processes are.

  Parallelism is when two or more threads are executed simultaneously. Meaning the atomic actions making up each of the threads actually run at the same time.

  The main difference between concurrency and parallelism in that for concurrency only one atomic action is done at a time, whilst in parallelism there are several atomic actions are run at the same time.

- Why have machines become increasingly multicore in the past decade?

  Since many of the processes we run consists of several atomic actions rather than one large on, it is easier to split them up into several smaller processes that can run in parallel.

  Concurrent execution is useful when dealing with heavy input/output programs. Time used when waiting for input or output in order to complete a task can be used for another task.

- Does concurrency make the programmers life easier? It makes the programmers life harder. More components to be aware of and logistics to consider. However, it is worth it in term of the performance of the system.

- What are the differences between processes, threads, green threads, and coroutines? Threads are a subset of a process. A processes are usually independent from other processes. Whilst threads within a process share memory and other resources.

  A green thread is similar to a normal thread but handled by a runtime library or a virtual machine. Thus threading can be achieved on platforms such as operating systems not supporting threading.

Coroutines are similar to threads but have more own resources like stack and instruction pointer but shares global variables with other coroutines. The can never be run in parallel as threads. Coroutines has several exit and entry point meaning it is possible to stop a coroutine and continue at a later point.

- Which one of these do pthread_create(), threading.Thread() , go create

  pthread_create() creates a thread.

  threading.Thread() creates a thread.

  go creates co-routine.

- How does pythons Global Interpreter Lock (GIL) influence the way a python Thread behaves GIL prevents several Python threads to be executing Python bytecode at the same time. This is done because pythons memory management is not thread safe.

- With this in mind: What is the workaround for the GIL (Hint: it's another module)?TT Python Multiprocessing module is an alternative when wanting to work around the GIL.

- What does func GOMAXPROCS(n int) int change? the function sets the maximum number of CPUs that can be executing simultaneously and return the previous setting.