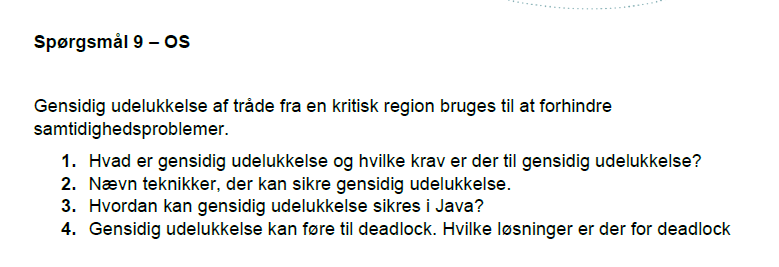
Spørgsmål 9



Står for mutual exclusion, hvilket er en metode hvor du sørger for at en ressource skal køres færdig af en tråd før en anden tråd kan tage fat i den.

Eksempelvis hvis man ikke har gensidig udelukkelse så kan en tråd overskrive hvad en anden tråd har skrevet mens den anden tråd er i gang. Vi lavede et forsøg omkring dette med et banksystem for 2 tråde tilføjede 100 kroner til en konto, og det betød så at der ofte blev overskrivet og at personen der tilføjede penge faktisk mistede nogle af dem i slutningen.

Du kan forhindre gensidig udelukkelse på 3 måder:

* Peterson, virker ganske godt, men den virker kun hvis der er 2 tråde der arbejder.
* Java’s Monitor. Du kan ved en metode tilføje synchonise, hvilket gør at hvis en metode er i brug kan den ikke tages i brug af andre før den er udført.
* Semaphore, hvilket er en lås der tæller op og ved bestemte kritiske sektioner, og kan derved sørge for at der ingen sharing a ressourcer sker.

For at der kan opstå en deadlock er der 4 ting der skal til. Hvis bare en af dem ikke er der kan der opstå en deadlock.

* Mutual Exclusion, eller gensidig udelukkelse.
* Hold and wait, en process kan holde på en resource indtil den kan tilgå den næste.
* No preemption, en process kan ikke fratages sin resource.
* Circular wait, dette er deadlocking hvor ingen resource kan komme videre, da de alle holder på en resourcer og venter på de andre slipper deres.

Der er måder at prevente deadlocks.

* Min favorite er Strudse metoden, hvilket bare er at ignorer den indtil det er stort nok problem og så dræbe processen så systemet kan komme videre. Den er effektiv i et system hvor der muligvis kan forekomme deadlocks, men det vil være et helvede at prøve at prevente dem på andre måder.
* Deadlock avoidance, hvilket er hvor man bevidst koder så der ikke kan opstå deadlocks, hvilket betyder man fjerner en af de metoder der kan resulterer i deadlocks, dette er ikke en god metode da det kan give problemer med system, eller man kan ikke drage nytte af flere trådet kodning.
* Bankers Algoritme, hvilket er en metode hvor du kende lige præcis hvor mange ressourcer der er og hvor mange processer der er, du kan derfor undgå deadlocks ved at kører processerne i bestemte rækkefølger for at kunne undgå deadlocks. Problemet med denne metode er du ikke altid kender hvor mange ressourcer der er eller mange processer du har.