5.1 Mutual exclusion: software approaches

Wednesday, April 8, 2020 10:22 PM

De sentrale temaene i OS-design har alle bekymringer i håndteringen av prosesser og tråder:

- **Multiprogrammering:** Håndteringen av flere prosesser i et uniprosessorsystem
- **Multiprosessering:** Håndteringen av flere prosesser i et multiprosessorsystem
- **Distribuert prosessering:** Håndteringen av flere prosesser som executer på flere, distribuerte computersystemer.

Fundamentalt for alle disse er konseptet om *samtidighet (concurrency)*. Samtidighet omhandler alt fra kommunikasjon mellom prosesser, deling av ressurser, synkronisering av aktiviteter på flere prosesser og tildeling av prosessortid til prosessene. Disse delikate problemene oppstår i alle tre ovennevnte systemer, og i tre forskjellige kontekster:

1. **Flere applikasjoner:** Multiprogrammering ble oppfunnet til å tillate prosesseringstid å være dynamisk delt mellom flere aktive applikasjoner.
2. **Strukturerte applikasjoner:** Som en utvidelse av prinsippet om modulært design og strukturert programmering kan noen applikasjoner effektivt programmeres som et sett med *samtidige prosesser.*
3. **OS-struktur:** Den samme struktureringen drar fordeler for systemprogrammer og vi har sett at operativsystemer selv blir implementert som et sett med prosesser eller tråder.

Samtidighet i prosesser kan implementeres ved hjelp av software tilnærminger for å execute på enkel-prosessorer eller en multiprosessor-maskin med delt hovedminne. Disse tilnærmingene antar gjensidig utelukkelse (*mutual exclusion*) ved minneaksessnivået. Det vil si at samtidige aksesser til samme lokasjon i minnet blir serialisert. Utenom dette er det ikke noe støtte i hardware, OS-et eller programmeringsspråk.

Dekkers algoritme (Ikke kommet på eksamener)

Er en algoritme for gjensidig utelukkelse for to prosesser som tar utgangspunkt i Dijkstras grådige tilnærming. Denne algoritmen handler om at prosesser må følge en prosedyre i execution, og må vente til de får tillatelse – dette kalles *busy waiting*. Denne algoritmen løser mutex-problemet men har et komplekst program som er vanskelig å følge og korrektheten er vanskelig å bevise.

Petersons algoritme (Ikke kommet på eksamener)

Denne algoritmen har en enkel og elegant løsning. Den løser også problemet ved å se på gjensidig utelukkelse for to prosesser. Uten å dykke dypt inn i koden til denne algoritmen også, så ligger hovedforskjellen at Dekker bruker en if-setning for å sjekke om den andre prosessen er i en kritisk fase, mens Petersons sjekker dette med en betingelse for while-løkken, som kun ser på den boolske verdien av flagget hos neste prosess i det globale variabel-arrayet.