SIF8041 OPERATIVSYSTEMER OG DATABASER

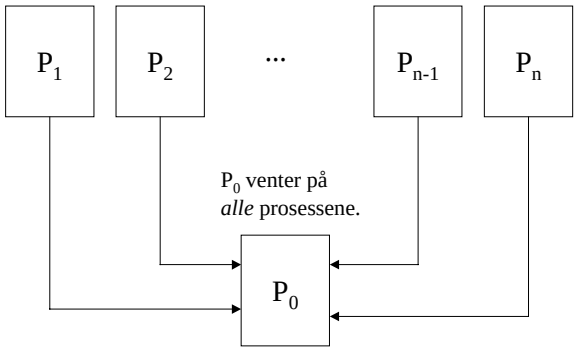
våren 2001

ØVING 2 : Synkronisering og vranglås

INNLEVERINGSFRIST : Onsdag 21.02.00
GRUPPESTØRRELSE : 2 -3. Lever svarene på papir (ikke håndskrevet) i boks 2 eta. E-blokka.

1. JOIN venting

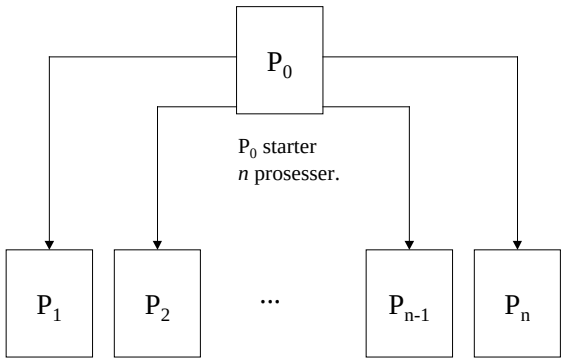
Gitt at en prosess P_0 venter på at et kjent antall (n) prosesser, P_i, skal avslutte:



Hvordan skal dette synkroniseres, dvs. hvordan skal P_0 aktiviseres når den siste av P_1 , P_2 , . . . ,P_n terminerer? Bruk semafor.

2. Felles start (hendelses-aktivisering)

Gitt at en prosess P_0 skal aktivisere (``trigge'') et ukjent antall (n) ventende prosesser P_i :



Hvordan skal dette synkroniseres? Hint: *Kun en semafor er nødvendig*. La P_0 starte P_1 som starter P_2 osv. som til slutt aktiviserer P_0 igjen.

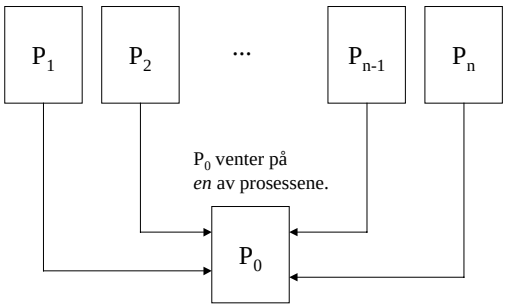
Eksempel på anvendelse:

```
var v: shared T;            T er en type.
region v do
     begin
          await B;          B er et generelt
          ...               logisk uttrykk.
     end;
```

Ved utgang må alle prosesser som venter på await B, vekkes opp for å utføre B testen om igjen. Disse avbrutte prosessene bør få prioritet framfor nye innkommende prosesser som venter på region v do.

3. Multippel venting (Gjentatt venting)

Dette likner på join-venting, men P_0 venter nå på at minst en av n prosesser, P_i , sender klarsignal:



Etter at en prosess (egentlig et signal fra en prosess) er behandlet, vil P_0 vente på en nytt signal fra en av prosessene.
Hvordan skal synkroniseringen gjøres ?

Hint: Innfør en ekstra semafor, samt en variabel som viser hvilken prosess som starter P_0.

4. Kritiske regioner

Synkroniseringsprimitivene LOCK og UNLOCK som benyttes ved inngang og utgang av kritisk region kan implementeres på en av to måter (nivåer) på en uniprosessor:

	Lavnivå	Høynivå
LOCK(s)	DisableInterrupts	P(s)
UNLOCK(s)	EnableInterrupts	V(s)

Vi har så følgende prosedyrer P1 og P2 hvor både P1 og P2 kan kalles direkte av en prosess, og P1 kan/vil kalle P2.

(se neste side)

procedure P1(...)	procedure P2(...)

lock(s);
...;
P2(...);
...;
unlock(s);
	lock(s);
	...;
	...;
	...;
	unlock(s);

a)
Beskriv hva som vil skje inni P2 ved nøstede LOCK/UNLOCK-kall, med henholdsvis lav og høynivåimplementasjon av disse. Kommenter forskjellen.

b)
For å unngå problemer som antydet over, kan vi lage ny versjoner av P og V primitivene, kalt PN og VN. Disse skal ikke passivisere den aktuelle prosess hvis vi har nøstede kritiske regioner på samme semafor (og ressurs). Virkemåten er altså som følger:

```
PN( ... );                      /* kaller P( s ) */
     PN( ... );                 /* kaller ikke P( s ) */
     .
     .
     .
      VN( ... );                /* kaller ikke V( s ) */
     VN( ... );                 /* kaller V( s ) */
```

Programmer PN og VN ved bruk av P og V. Anta systemfunksjoner som synes naturlig.
Hint: Det må innføres ett eller flere nye felter i den nye semafortypen.

5. Binær og generell semafor

En *binær* semafor er en semafor hvis verdi kun kan være 0 eller 1. Vis hvordan generelle semaforer (som kan ta andre verdier) kan implementeres med binære semaforer. Den generelle semaforen vil være en heltallsvariabel.

6. Oppgave 6.5 i læreboka

7. Oppgave 6.14 i læreboka

Implementer et kort program i Java. Merk: Det holder å levere koden.

8. Oppgave 6.17 i læreboka

9. Spisende filosofer

Det er 4 nødvendige forutsetninger for vranglås (deadlock) i ressursallokeringsproblemene vist i *Chinese Dining Philosophers* :

- 'Mutual exclusive' bruk av ressurser
- Ingen 'pre-emtion' av ressurser
- Holde en ressurs mens en venter på en annen
- Sykler (cycles) av venting

Er dette utsagnet SANT eller FEIL: Hvis vi tar vekk en (vilkårlig) av disse 4 forutsetningene for vranglås (f.eks. vi godtar 'pre-emtion'), er vi da garantert å få en korrekt løsning på Dining Philosophers Problemet? Forklar kort svaret.

10. Ressurstildeling

Du skal designe et system for ressurstildeling, der første prioritet er maksimal ressursutnyttelse. (M.a.o., det er ikke akseptabelt å vente med å tildele en ledig ressurs bare for å "være sikker".) Dessuten vet du at mønsteret for ressursbruk er slik at det i praksis bare sjelden ville bli vranglås.

Hvilken av strategiene *forhindring*, *unngåelse* eller *deteksjon + bryting* vil du velge, og hvorfor?

11. Oppgave 7.13 a) b) og c) i læreboka

Forklar kort svaret på oppg. 7.13 c).