Tentamen A2: Bedrijfssystemen, 22-01-2003

Neem na het tentamen een keer de tijd om de enquetes serieus in te vullen, want dat is voor ons erg belangrijk om volgend jaar dingen te verbeteren aan het vak! Lever de enquete in bij de surveillant of gooi ze in de komende 2 weken een keer in de inleverbak op de 5e verdieping.
Succes!!

Vraag 1 (2 punten)

Een concurrent proces bestaat uit twee threads, T_1 and T_2, die elk het volgende programma uitvoeren:

```
char x[1024];
read(f,x,1024); // lees 1024 bytes uit file f
compute(x);     // voer een of andere berekening uit
read(f,x,1024); // lees 1024 bytes uit file f
compute(x);     // voer een of andere berekening uit
thread_done();  // beeindig deze thread
```

Het duurt altijd precies 30 tijdseenheden voordat een aanroep van `read` zijn resultaat oplevert, zelfs als er meerdere aanroepen tegelijkertijd worden uitgevoerd. De rekentijd die `read` en `thread_done` gebruiken is verwaarloosbaar klein. De functie `compute()` neemt 10 tijdseenheden rekentijd, en bevat geen operaties die een thread kunnen blokkeren.

Het OS gebruikt round robin scheduling met een quantum van 20 tijdseenheden. Het OS kent enkel de procestoestanden *Ready*, *Running* en *Blocked*. Er zijn geen andere processen op de machine en thread T_1 zal als eerste de CPU krijgen als het proces start.

a) Neem aan dat we kernel level threads hebben.

   - Geef op een tijdbalk zoals hieronder met T_1 en T_2 aan welke thread tijdens welke tijdseenheid de CPU heeft.

1	2	3	4	5	6	7	8	9	10	11	...
											...

   - Zeg voor elke thread hoeveel tijdseenheden deze in de toestand *Ready* verblijft, en hoeveel in de toestand *Blocked*, gedurende de levensloop van het hele proces.

b) Beantwoord dezelfde vragen als hierboven, maar nu onder de aanname dat we user-level threads hebben.

Vraag 2 (3 punten)

Het NOS journaal berekent prognoses voor de verkiezingsuitslag gebaseerd op invoer van drie observaties. Op de computer bij de NOS draaien 4 processen, één voor elk van de drie observaties en één voor de berekening van de uiteindelijke prognose. Er is een stuk gedeeld geheugen voor de communicatie tussen de processen.

De drie invoerprocessen `invoer1`, `invoer2` en `invoer3` voeren de volgende (pseudo)code uit:

```
void main() {
     Shared_Buffer *buffer;
     Result *r;
     r = make_observation();
     put_result(buffer, r);
}
```

Het proces `bereken` dat de uiteindelijke prognose berekent voert de volgende (pseudo)code uit:

```
void main() {
     Shared_Buffer *buffer;
     Results *r1, *r2, *r3, *prognose;
     get_result(buffer, r1, r2, r3);
     prognose = calculate(r1, r2, r3);
     zet_op_teletekst(prognose);
}
```

Alle vier de processen starten op hetzelfde tijdstip. Voor alle duidelijkheid: alleen de `Shared_Buffer buffer` is gedeeld geheugen.

ICT-expert Maurice de Hond komt erachter dat deze code niet goed werkt, omdat er geen enkele vorm van mutual exclusion tussen de processen is gegarandeerd, en het `bereken` proces kan beginnen vóór alle drie de invoerprocessen klaar zijn.

Introduceer semaforen en voeg semafooroperaties `wait(...)` en `signal(...)` aan de code toe om de vereiste synchronisatie te realiseren. Vermeld duidelijk wat de initiële waarden van de semaforen moeten zijn.

ZOZ

Vraag 5 (1.5 punt)

Een OS gebruikt round robin (RR) scheduling, waarbij processen verschillende quanta kunnen hebben. Initiëel krijgt elk proces een vast quantum q. Als een proces zijn quantum helemaal opmaakt zonder te blokkeren wordt zijn quantum verdubbeld. Als een proces blokkeert wordt zijn quantum weer teruggezet op q.

a) Wat zou een motivatie kunnen zijn om deze variant van RR te gebruiken, ipv. gewone RR waarbij alle processen altijd hetzelfde quantum hebben?

b) Kan er starvation optreden?

We passen het scheduling algoritme aan met prioriteiten: processen met een kleiner quantum krijgen een hogere prioriteit. Dwz. de scheduler kijkt wat het kleinste quantum is dat in de ready-wachtrij voorkomt en kiest het eerste process in de wachtrij dat dit quantum heeft.

c) Wat zou een motivatie kunnen zijn voor deze aanpassing?

d) Kan er starvation optreden?

Motiveer je antwoorden!