Hertentamen PC: Processen & Concurrency
19-08-2005

Geef bij al je antwoorden voldoende motivatie: voor alleen getallen of ja/nee antwoorden krijg je geen punten. Succes!!

Vraag 1 (1 punt)

Wat is het verschil tussen een *suspended* en een *blocked* proces? Heeft het zin om zowel een suspended toestand als een blocked toestand te hebben? Geef vier mogelijke redenen om een proces in suspended toestand te brengen.

Vraag 2 (1.5 punt)

Wanneer een proces overgaat van de "Ready" toestand naar de "Running" toestand, dient ieder register in de CPU ofwel opnieuw geïnitialiseerd worden, ofwel de waarde krijgen die het had toen het proces voor het laatst onderbroken werd. Leg uit waarom het PC register typisch het laatste register is dat geladen wordt. Beschrijf in pseudo assembly-code hoe je een nieuw proces op de CPU kunt laden bij een processor met rekenkundige/logische registers R0-R3, program status word PSW, program counter PC, en instructie register IR. In de code kun je gebruik maken van load, store, jump en increment instructies.

Vraag 3 (1.5 punt)

Beschouw de pseudo code in Figuur 1.

a. Stel dat Producer gedurende tijd t_p blokkeert telkens bij het lezen in de file, en dat Consumer tijd t_c nodig heeft om de inhoud van een buffer te verwerken (Neem aan dat Consumer hierbij niet blokkeert). Neem verder aan dat de resterende delen van de code voor Producer en Consumer een verwaarloosbare hoeveelheid CPU tijd vergen. Hoeveel tijd kost het voordat beide threads klaar zijn wanneer ze volledig op het gebruikersniveau ("user-level") zijn geïmplementeerd? Je antwoord dient te bestaan uit een enkele expressie met daarin t_c en/of t_p.

```
Variables Shared by Both Threads
--------------------------------
data buffer[10]; /* array of buffers to hold produced data */
semaphore sem = 0; /* a counting semaphore to synchronize
                                     access to the buffers */

Produced Thread                      Consumer Thread
---------------                      ---------------
FOR i FROM 1 TO 10 DO                FOR i FROM 1 TO 10 DO
  read from file into buffer[i]        P(sem)
  V(sem)                               process contents of buffer[i]
ENDFOR                               ENDFOR
```

Figure 1: Producer/Consumer systeem

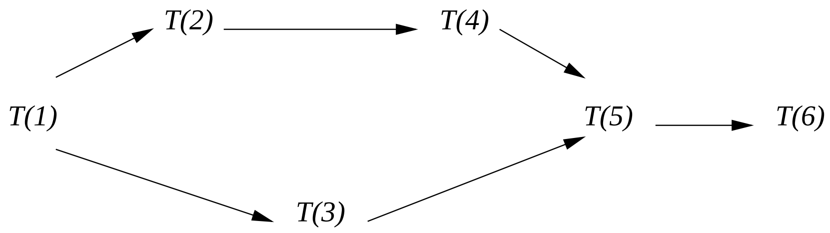
b. Herhaal onderdeel (a) aannemend dat het bedrijfssysteem multi-processing ondersteunt en beide threads direct implementeert ("kernel-level").

Vraag 4 (2 punten)

Laat zien hoe we met behulp van binaire semaforen de activiteiten van 6 threads kunnen coordineren. Ieder thread heeft de volgende structuur.

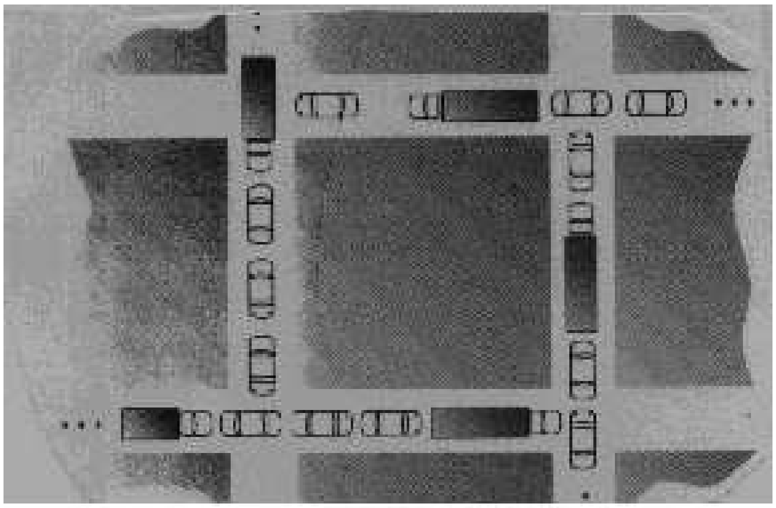
```
Thread(i)   {
Code Section
}
```

Plaats binaire semaforen rondom de Code Section van iedere thread om daarmee de onderstaande volgorde-beperkingen te implementeren. Een pijl van $T(i)$ naar $T(j)$ betekent dat de Code Section van $T(i)$ helemaal klaar moet zijn voordat begonnen wordt met de Code Section van $T(j)$.



Vraag 5 (1 punt)

Beschouw de verkeersdeadlock die hieronder staat afgebeeld.



1. Laat zien dat in dit geval voldaan is aan de vier noodzakelijke voorwaarden voor deadlock.
2. Geef een eenvoudige regel waarmee deadlock in dit systeem voorkomen kan worden.

Vraag 6 (2 punten)

Vijf jobs A t/m E arriveren bij een processor op tijdstippen 0, 10, 10, 80 en 85. De geschatte rekentijden zijn respectievelijk 75, 40, 25, 20 en 45 tijdseenheden. Neem aan dat gebruik wordt gemaakt van RR scheduling met een tijdquantum van 15 tijdseenheden.

a Illustreer de executie van de vijf jobs in een diagram.

b Wat is de turnaround time voor job C?

c Wat is de gemiddelde genormaliseerde turnaround time voor alle jobs?

Vraag 7 (1 punt)

Een real-time systeem heeft 4 periodieke taken met periodes van 50, 100, 200 en 250 msec. Neem aan dat de vier taken respectievelijk 35, 20, 10 en x msec CPU tijd nodig hebben. Wat is de grootste waarde van x waarvoor het systeem nog schedulable is?