Tentamen A2a: Bedrijfssystemen, 25-11-2001

Succes!!

Vraag 1 (2 punten)

Een proces P bestaat uit drie threads. Thread $T1$ rekent 10 msec, vraagt dan om invoer van de disk waar-ie 100 msec op moet wachten, en rekent vervolgens nog 10 msec. Thread $T2$ rekent 20 msec, vraagt dan om invoer van de CD-rom waar-ie 400 msec op moet wachten, en rekent vervolgens nog 10 msec. Thread $T3$ rekent 400 msec en hoeft daarbij nooit te wachten op invoer.

Hoe snel is het hele proces P op zijn snelst klaar als het draait op

(a) een operating system met kernel-level threads, en een computer met 1 processor?

(b) een operating system met user-level threads, en een computer met 1 processor?

(c) een operating system met kernel-level threads, en een computer met 12 processoren?

(d) een operating system met user-level threads, en een computer met 12 processoren?

Leg bij je antwoorden kort maar duidelijk uit hoe je eraan komt.

Vraag 2 (1 punt)

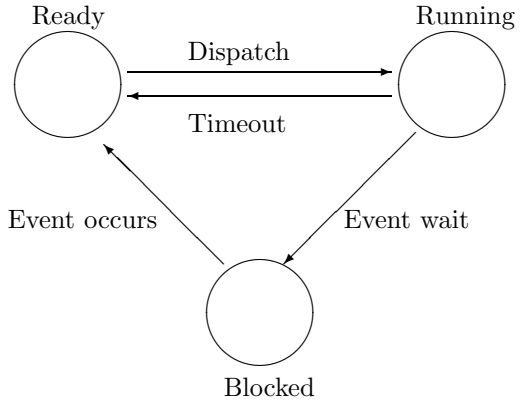
We hebben een computer met daarop aangesloten een printer, een CD-lezer, en een DVD-speler. Processen P_1, P_2 en P_3 hebben tijdens hun uitvoering soms exclusief gebruik van twee van deze drie stukken randapparatuur nodig. Ze claimen randapparatuur voor exclusief gebruik met `acquire` en geven deze weer vrij met `release`, in de volgende volgorde:

P_1 :	P_2 :	P_3 :
`acquire printer`	`acquire DVD`	`acquire printer`
`acquire CD`	`acquire CD`	`acquire DVD`
⋮	⋮	⋮
`release printer`	`release CD`	`release printer`
`release CD`	`release DVD`	`release DVD`

Laat zien dan er deadlock kan optreden, of geef een overtuigende argumentie waarom dit niet kan.

Vraag 3 (2 punten)

Het plaatje hieronder toont het toestandsdiagram voor processen van een of ander operating system. Het operating system draait enkel op single processor machines.



(a) De hardware van de computer heeft een 'test en set' instructie, die we gebruiken om mutual exclusion van kritieke secties in twee processen P en Q te garanderen. Stel dat P in zijn kritieke sectie is. In welke van de drie toestanden kan proces P in dat geval zijn ? En in welke van de drie toestanden kan proces Q in dat geval zijn ? Motiveer je antwoorden!

We willen dit operating systeem uitbreiden met twee operaties: *laat_slapen* en *maak_wakker*. Met de eerste operatie kan een proces de executie van een *ander* proces (tijdelijk) onderbreken. Dit andere proces zal dan niet meer draaien – dwz. geen CPU tijd meer krijgen – tot het weer wordt "wakker gemaakt"door een of ander proces.

(b) Voeg toestanden en/of toestandsovergangen toe aan het bovenstaande diagram om deze operaties mogelijk te maken. Zeg voor nieuwe toestanden kort waarvoor ze dienen, en voor nieuwe toestandsovergangen waardoor deze veroorzaakt kunnen worden.

(c) Als het operating system ook op multi-processor machines moet kunnen draaien, maakt dit wat betreft dit nieuwe toestandsdiagram enig verschil? Motiveer je antwoord.

ZOZ voor vraag 4 en 5!

Vraag 4 (2 punten)

We hebben drie processen, A, B en C. Proces A produceert invoer voor proces B, en proces B produceert hieruit invoer voor proces C.

De processen wisselen data uit via de harde schijf. Ze produceren en/of consumeren data in blokken van steeds dezelfde grootte, en deze blokken worden gebufferd op een harde schijf:

- Proces A produceert af en toe één blok uitvoer en schrijft dit naar de disk.
- Proces B haalt telkens één blok uitvoer van A van de disk, rekent dan een tijdje, en schrijft daarna één blok uitvoer naar de disk. (Proces B is dus nooit tegelijk een blok aan het lezen en een ander blok aan het schrijven.)
- Proces C haalt telkens één blok uitvoer van B van de disk.

Als er geen invoer voor een proces is, of geen ruimte op de harde schijf is voor z'n uitvoer, dan wacht dat proces totdat er wel invoer cq. ruimte op de harde schijf beschikbaar is.

De harde schijf biedt ruimte voor maximaal max blokken data. Hiervan kan een willekeurig aantal in gebruik zijn om uitvoer van A zolang op slaan, en een willekeurig aantal om uitvoer van B zolang op te slaan. Natuurlijk geldt wel altijd

$$out_A + out_B \leq max$$

waar out_A het aantal blokken nog onverwerkte uitvoer van A is en out_B het aantal blokken nog onverwerkte uitvoer van B.

(a) Wat is het voordeel van een grotere waarde van max, dwz. van meer bufferruimte ?

(b) Laat zien dat er deadlock op kan treden.

(c) Bedenk een oplossing waarbij deadlock wordt voorkomen, maar die nog wel zoveel mogelijk vrijheid biedt over hoeveel blokken van de disk gebruikt worden voor uitvoer van A en hoeveel voor uitvoer van B. Leg uit waarom je oplossing werkt.

Vraag 5 (3 punten)

Een proces bestaat uit 6 threads: A_1, A_2, A_3, A_4, B, C.

De code van de threads A_i is $\{\mathtt{s}_1; \mathtt{cs}_1; \mathtt{s}_2\}$

De code van B is $\{\mathtt{s}_4; \mathtt{cs}_2; \mathtt{s}_5\}$

De code van C $\{\mathtt{s}_5\}$

Introduceer één of meer semaforen en breidt de code van de threads uit met semafoor-operaties zodanig dat:

- er maximaal 2 van de threads A_1 t/m A_4 in het code-fragment $\mathtt{cs}_1$ kunnen zitten.
- thread B pas met $\mathtt{cs}_2$ kan beginnen zodra er 3 van de threads A_1 t/m A_4 helemaal klaar zijn.
- thread C pas kan starten zodra alle andere threads helemaal klaar zijn.

Vergeet niet te vermelden wat de initiële waarde van elke semafoor moet zijn.

(Mocht het je niet lukken om een oplossing te vinden die alle bovenstaande condities garandeert, maak dan in elk geval een oplossing die zoveel mogelijk van de condities garandeert.)