TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica
Sectie Parallelle en Gedistribueerde Systemen

TU Delft

TI-2720 Operating System Concepten

6 november 2012, 14.00 - 17.00 uur.

docent: H.J. Sips

Dit is een tentamen met **9** open vragen

Opgave	Punten
1	12
2	12
3	9
4	12
5	12
6	9
7	9
8	12
9	12
Kado	1
Totaal	100

Het gebruik van boeken, dictaten of aantekeningen is **niet** toegestaan

Schrijf **duidelijk** en formuleer **kort en bondig**

Opgave 1 (12 punten)

De kernel van een Operating System kan worden aangeroepen door middel van een interrupt of door middel van een trap.

a. Leg uit wanneer er een interrupt optreedt. Geef een voorbeeld.

b. Leg uit wanneer er een trap optreedt. Geef een voorbeeld

c. Interrupts en traps worden op dezelfde wijze afgehandeld. Geef kort weer wat er gebeurt bij het optreden van een interrupt of een trap

d. Wanneer we de termen synchroon en asynchroon willen gebruiken in verband met interrupts en traps, welke kunnen we dan synchroon noemen en welk asynchroon? Geef een korte toelichting.

Opgave 2 (12 punten)

Gegeven een multiprogrammeringssysteem met één processor waarin een aantal processen. In het systeem vinden per seconde veel context switches plaats.

a. Wat wordt er verstaan onder een proces?

b. Wat wordt er verstaan onder multiprogramming en wat onder multiprocessing?

c. Wat wordt er verstaan onder een context switch? Leg uit wat er bij een context switch gebeurt.

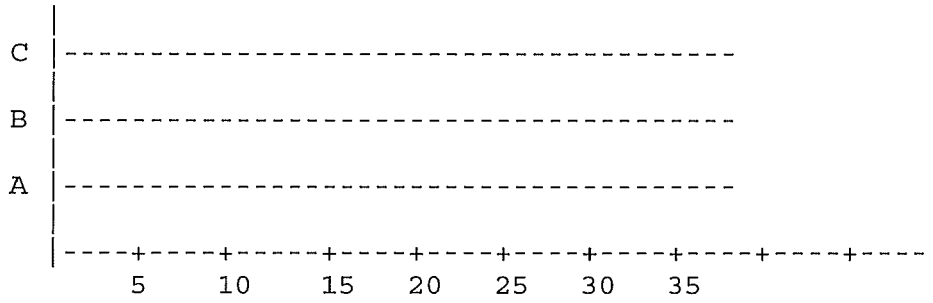
d. Kan een proces ontdekken dat hij de processor een tijdje kwijt is geweest?
Zo ja, geef aan hoe. Zo nee, waarom niet?

Opgave 3 (9 punten)

De short-term scheduler houdt zich bezig met het toewijzen van de CPU aan processen in de Ready queue. Eén van de mogelijke scheduling strategieën is preemptive Shortest Job First (SJF).
Onderstaande tabel geeft de aankomsttijden (in de Ready queue) van de processen A, B en C, en de benodigde CPU tijden. De processen doen tijdens de executie geen I/O.

	Aankomsttijd (seconden)	CPUtijd (seconden)
A	0	10
B	2	5
C	3	2

a. Neem het hieronder gegeven schema over op uw papier en geef voor ieder proces op zijn horizontale lijn aan wanneer deze Ready is (aangeven met R) en wanneer deze de CPU heeft (aangeven met C).



b. Bepaal de gemiddelde responstijd.

c. Als het eenvoudig implementeerbaar zou zijn, zou preemptive SJF dan de beste scheduling policy voor een Time-sharing systeem zijn?
Motiveer uw antwoord.

Opgave 4 (12 punten)

Onderstaande (pseudo)code geeft een oplossing voor het Producer-Consumer probleem met bounded buffer. Bij deze oplossing is gebruik gemaakt van semaforen. In de buffer is plaats voor maximaal *N* elementen.

Producer

```
[1] do {
[2]    produce an item in nextp
[3]    Wait(empty);
[4]    Wait(mutex);
[5]    add nextp to buffer
[6]    Signal(mutex);
[7]    Signal(full);
[8] } while(1);
```

Consumer

```
[11] do {
[12]    Wait(full);
[13]    Wait(mutex);
[14]    move an item to nextc
[15]    Signal(mutex);
[16]    Signal(empty);
[17]    consume the item in nextc
[18] } while(1);
```

Beantwoord hierover nu de volgende vragen.

a. Wat moeten de initiële waarden van elk van de semaforen zijn? Motiveer je antwoord.

b. Leg uit hoe de primitieven Wait() en Signal() efficiënt geïmplementeerd kunnen worden, d.w.z. zonder dat er gebruik gemaakt hoeft te worden van *busy waiting*.

c. Beargumenteer dat de oplossing ook goed is als regels [6] en [7] omgewisseld worden. Waarom verdient de gegeven (originele) oplossing dan toch de voorkeur?

d. Werkt de oplossing met semaforen, zoals gegeven in bovenstaande code, ook in een multiprocessor systeem? Licht je antwoord kort toe.

Opgave 5 (12 punten)

Een van de begrippen die een rol spelen bij het ontwijken (avoidance) van deadlocks is het begrip veilige volgorde.

a. Leg uit wat er verstaan wordt onder een veilige volgorde.

b. Stel we hebben een systeem waarin 8 buffers. Onderstaande tabel geeft aan hoe de buffers op tijdstip t=0 zijn verdeeld over de processen A, B en C, en geeft de maximale behoefte van de processen (hun *claim*)

proces	bezit	claim
A	1	4
B	2	3
C	3	7

Laat zien dat dit een veilige toestand is.

c. C vraagt nu zijn vierde buffer. Mag deze worden toegewezen als we deadlocks willen ontwijken? Licht uw antwoord kort toe.

d. Als een toewijzing niet zou mogen plaatsvinden en het zou toch gebeuren, gaat er dan gegarandeerd een deadlock optreden? Motiveer uw antwoord.

Opgave 6 (9 punten)

In een systeem waarin meerdere processen aanwezig zijn, kan ten behoeve van het binnenhalen van een pagina voor proces P1 een pagina van proces P2 uit het geheugen worden verwijderd. Welk proces 'slachtoffer' wordt, wordt bepaald door de frame allocation strategie. Een belangrijke eis aan een dergelijke strategie is, dat deze trashing moet voorkomen. Eén van deze strategieën is de Page Fault Frequency strategie.

a. Wat wordt in dit verband verstaan onder trashing?

b. Leg uit hoe de Page Fault Frequency strategie werkt.

c. In deze strategie wordt voor het tijdsbegrip gebruik gemaakt van virtuele tijd. Leg uit wat er wordt verstaan onder virtuele tijd, en waarom dit hier beter is dan de echte tijd.

Opgave 7 (9 punten)

Het toewijzen van schijfruimte aan bestanden kan worden gedaan met behulp van *contiguous allocation* of van *linked allocation*.

a. Beschrijf kort wat wordt onder beide allocatie-methoden wordt verstaan.

b. Leg kort uit wat er wordt verstaan onder externe fragmentatie en wat onder interne fragmentatie.

c. Welke vorm(en) van fragmentatie kunnen optreden bij contiguous allocation en welke bij linked allocation? Licht je antwoord toe.

Opgave 8 (12 punten)

Bij het beveiligen van een computer systeem speelt het Operating System een belangrijke taak.

a. Vermeld kort wat er wordt verstaan onder het begrip *protectiedomein*, en hoe er tussen domeinen geswitcht kan worden.

b. Leg uit wat het begrip *authenticatie* inhoudt, waarom het gebruik van passwords af te raden is, en wat een beter alternatief is.

c. Veel beveiligingsmethodieken zijn gebaseerd op *one-way functions*. Vertel kort wat dat zijn, en geef een concreet voorbeeld van de toepassing ervan (t.b.v. beveiliging).

d. Leg kort uit hoe een buffer-overflow attack in zijn werk gaat, en wat hier tegen te doen valt.

Opgave 9 (12 punten)

Gegeven zijn twee gedistribueerde sites S1 en S2, ieder met hun eigen resources. Processen kunnen op elk van deze sites resources aanvragen. Elke site houdt voor zijn eigen resources een Local Wait-For graph bij.

a. Hoe kunnen we deadlock detecteren middels het gebruik van een centrale coördinator? Geef een voorbeeld.

b. Geeft aan hoe deze benadering tot de detectie van False Cycles kan leiden. Hoe kunnen we dit oplossen?

c. Geef een oplossing door middel van een volledig gedistribueerd algoritme, waarbij er geen centrale coördinator is.

Einde tentamenopgave