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

TU Delft

IN1805-I Operating System Concepten

1 juli 2009, 09.00 - 12.00 uur.

docent: K.G. Langendoen

Dit is een tentamen met **8** open vragen

Opgave	Punten
1	12
2	12
3	12
4	14
5	12
6	12
7	14
8	12
Totaal	100

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

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

De tentamenstof omvat de hoofdstukken 1-12, 14 en 15 van het boek
A. Silberschatz et al, "Operating System Concepts", *Eighth Edition*

Opgave 1 (12 punten)

a. Wat wordt er verstaan onder de kernel van een Operating System?

b. Beschrijf kort het interface tussen applicaties en kernel.

c. Beschrijf kort het interface tussen hardware en kernel.

d. Hoe wordt in de meeste systemen bereikt dat een applicatie niet zelf rechtstreeks de hardware kan benaderen, maar dit moet doen via de kernel?

Opgave 2 (12 punten)

a Wat wordt er in het algemeen verstaan onder een virtuele machine?

b. In een 'klassieke' virtuele machine biedt de virtuele machine hetzelfde interface als de kale hardware. (De virtuele machine implementatie, ook wel aangeduid als Virtualiserende kernel, zorgt hiervoor.) Voor welke twee doeleinden wordt dit type virtuele machines gebruikt?

c. Illustreer en beschrijf kort hoe een de system call van een gebruikers applicatie afgehandeld wordt door een OS dat op een virtualiserende kernel draait. Vermeld duidelijk hoe de overgangen tussen de verschillende lagen verlopen.

d. Kan men een virtualiserende kernel testen op een virtuele machine? Motiveer uw antwoord kort.

Opgave 3 (12 punten)

Stel we hebben een toepassing met een groot aantal taken. Deze taken maken gebruik van gemeenschappelijke variabelen (shared variables). In principe kan deze toepassing op twee manieren worden geïmplementeerd. Men kan gebruik maken van threads of men kan kindprocessen gebruiken.

a. Wat is in deze situatie het belangrijkste voordeel van het gebruik van threads. Geef een korte toelichting.

b. Wat is in deze situatie het belangrijkste nadeel van threads. Licht uw antwoord kort toe.

c. Threads kunnen worden ondersteund door de kernel of op user niveau (d.m.v. libraries)
Als de hierboven beschreven toepassing veel gebruik maakt van I/O en thread support is op user niveau (libraries), welk probleem kan dan ontstaan? Motiveer uw antwoord.

Opgave 4 (14 punten)

Een belangrijke complicatie bij het beheren van resources is de mogelijkheid dat processen in een deadlock raken.

a. Noem de 4 noodzakelijke voorwaarden voor het optreden van een deadlock.

b. Deadlocks kunnen vermeden worden (prevention) door 1 van bovenstaande voorwaarden te elimineren. Geef voor 2 voorwaarden (naar keuze) een eliminatie strategie aan.

c. Deadlocks kunnen ook vermeden worden (avoidance). Waarom wordt in het algemeen avoidance geprefereerd boven prevention?

d. Beschrijf kort (!) wat het begrip *veilige volgorde* in houdt, en hoe het *Bankiers algoritme* hier gebruik van maakt om deadlocks te vermijden.

Opgave 5 (12 punten)

Gegeven een systeem met één processor. Hierin loopt een aantal processen dat gebruik maakt van hetzelfde programma. Het programma bevat een kritieke sectie. De programmeur wil deze sectie beschermen met behulp van een shared variabele BEZET die initieel de waarde 0 heeft.
Voor het begin van de kritieke sectie zijn in het programma de volgende statements opgenomen.

```
while BEZET=1 do no-op;
BEZET=1;
```

Na de kritieke sectie is de volgende opdracht geplaatst:

```
BEZET=0;
```

Beantwoord hierover de volgende vragen.

a. Is er op deze wijze gegarandeerd dat er nooit 2 processen tegelijk in de kritieke sectie zijn? Motiveer uw antwoord.

b. Hoe kan wederzijdse uitsluiting worden gerealiseerd door middel van een semafoor? Geef hierbij aan wat de initiële waarde van de semafoor moet zijn, en welke operaties op de semafoor moeten worden uitgevoerd bij het betreden en verlaten van de kritieke sectie.

c. Bij de eerst gegeven oplossing (met `BEZET`) is er sprake van busy waiting. Is er bij de oplossing met behulp van de semafoor ook sprake van busy waiting? Licht uw antwoord kort toe.

d. Werkt de oplossing met de semafoor ook in een systeem met meerdere processoren? Licht uw antwoord kort toe.

Opgave 6 (12 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 thrashing moet voorkomen. Eén van deze strategieën is de working-set strategie.

a. Wat wordt in dit verband verstaan onder thrashing?

b. Leg uit wat de working-set strategie inhoudt. Hierbij moet ondermeer duidelijk worden gemaakt, wat er onder een working set wordt verstaan.

c. Op grond van welk principe kunnen we verwachten dat de working-set policy effectief zal zijn. Geef ook aan onder welke omstandigheden, deze niet goed zal werken.

Opgave 7 (14 punten)

De implementatie van virtueel geheugen is gebaseerd op het concept paging, en maakt gebruik van speciale hardware: de MMU.

a. Leg beknopt uit waar de TLB (Translation Look-aside Buffer), onderdeel v/d MMU, voor dient en hoe hij werkt.

b. Leg uit wat er gebeurt bij de afhandeling van een page fault die optreedt als een proces een stuk virtueel geheugen adresseert dat zich niet in het main memory bevindt.

c. Stel dat we werken met een 32-bits machine voorzien van 2 GB fysiek geheugen, en 4 KB pages. Hoe groot is dan de (single-level) page tabel van een proces?

d. Om het geheugengebruik van page tabellen terug te dringen wordt er vaak gebruik gemaakt van multi-level page tabellen. Leg uit hoe, en waarom, er dan ruimte bespaard kan worden. Geef een concreet (reken) voorbeeld.

e. Een andere mogelijkheid is het gebruik van *inverted* page tabellen. Leg uit hoe deze werken, en bereken wat de grootte is bij de machine configuratie van onderdeel c. Zijn er ook nadelen aan deze methode?

Opgave 8 (12 punten)

Voor elke geopende file wordt er administratie bijgehouden zowel in een *per-process* Open File Table (OFT) als in een *system-wide* OFT.

a. Leg uit waarom er twee OFT's gebruikt worden.

b. Welke data (attributen) worden in de per-processs OFT bijgehouden, en welke in de system-wide OFT?

c. Stel dat twee processen een file gebruiken om onderling te communiceren als in een producer-consumer setting. Leg uit hoe de proces sycnchronisatie dan te werk gaat, ihb. hoe wordt voorkomen dat de consumer data leest die nog niet geschreven is door de producer?

d. Als een process een file sluit, kan dan de entry in de per-process OFT direct opgeruimd worden? En de entry in de system-wide OFT? Zo nee, wat moet er dan gebeuren?

Einde tentamenopgaven