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

TU Delft

IN1805-I Operating System Concepten

9 april 2010, 09.00 - 12.00 uur.

docent: D.H.J. Epema

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

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

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

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

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)

Een belangrijke taak van een Operating System is het beheren van de CPU.

a. Leg de relatie uit tussen processen en *threads*, in het bijzonder met betrekking tot CPU-scheduling.

b. Leg het verschil uit tussen *preemptive* en *non-preemptive* scheduling, en benoem de voor- en nadelen van beiden.

c. Leg het *mechanisme* uit waarmee *preemptive* scheduling geïmplementeerd kan worden. Licht kort toe wat er gebeurt bij een *process* switch.

d. Noem drie *scheduling policies* die gebruikt kunnen worden om te bepalen wat het volgende proces is dat de CPU toegewezen krijgt, en vertel kort hoe ze werken.

Opgave 3 (14 punten)

Onderstaande (pseudo)code geeft een oplossing voor het Producer-Consumer probleem met een *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 4 (14 punten)

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

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

b. Deadlocks kunnen vermeden worden (*prevention*) door één van deze voorwaarden te elimineren. Geef voor twee voorwaarden (naar keuze) een eliminatiestrategie 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)

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 strategie effectief zal zijn? Geef ook aan onder welke omstandigheden deze niet goed zal werken.

Opgave 6 (12 punten)

De implementatie van virtueel geheugen is gebaseerd op het concept paginering, en maakt gebruik van speciale hardware, de *Memory Management Unit* (MMU).

a. Leg beknopt uit waar de TLB (Translation Look-aside Buffer), onderdeel van de 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 hoofdgeheugen bevindt.

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

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

Opgave 7 (12 punten)

In een boomvormig directory-systeem kunnen extra verwijzingen opgenomen worden in de vorm van duplicaat entries (*hard links*) en symbolische links (*soft links*).

a. Waarom is deze uitbreiding nuttig, een boom is toch veel overzichtelijker?

b. Leg uit wat het verschil is tussen een hard link en een soft link.

c Als een delete opdracht wordt gegeven voor een bestand waarnaar een extra verwijzing bestaat, welke problemen ontstaan er dan wanneer het bestand onmiddellijk wordt verwijderd (maak hierbij onderscheid tussen de beide typen verwijzingen)?

d Hoe is in UNIX het probleem van het verwijderen van bestanden waarnaar hard links verwijzen opgelost?

Opgave 8 (12 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.

Einde tentamenopgaven