Tentamen Operating Systems

Vrijdag 31 januari 2014, 14:00-17:00

- Lees eerst een opgave volledig door, alvorens deze te maken.
- Schrijf netjes en zorgvuldig.
- Dit tentamen is 'Open Boek', d.w.z. dat het boek "*Operating Systems, Design and Implementation*" van Tanenbaum & Woodhull gebruikt mag worden als naslagwerk. Het is niet toegestaan ander materiaal, zoals college-aantekeningen en powerpoint-slides, te raadplegen.
- Het tentamen bestaat uit 7 opgaven. Je krijgt 10 punten cadeau. De opgaven 1, 2, 3 en 4 zijn 15 punten waard. De overige opgaven zijn 10 punten waard.
- Je hebt 3 uur de tijd, gebruik deze nuttig. Ook als je snel klaar bent, gebruik dan de resterende tijd om jouw antwoorden nog eens te controleren.
- Succes!

Opgave 1: Algemene OS vragen
(a) Wat is het fundamentele verschil tussen een proces en een thread?
(b) Implementeer (in pseudo-code) algemene semaforen (zogenaamde *counting-semaphores*) met behulp van binaire semaforen. Geef implementaties van de operaties `Psem` en `Vsem`, waarbij je gebruik mag maken van de binaire semafoor operaties `P` en `V`.
(c) Geef aan waarom DMA (Direct Memory Access) en puur virtual memory niet goed samen gaan.
(d) Wat is een UNIX zombie proces?
(e) Wat is het effect, op een standaard UNIX systeem, van het statement `while(fork());` ?

Opgave 2: Unix system programming
(a) Schrijf een codefragment dat 10 processen als volgt creëert; het oorspronkelijke proces (proces 0) creëert proces 1, proces 1 creëert proces 2, etc. Ieder proces drukt, nadat zijn kind-proces is getermineerd, zijn rangnummer in de keten af. De uitvoer van het programma dient dus te zijn:

```
9 8 7 6 5 4 3 2 1 0
```

(b) *Leg uit wat de onderstaande code bewerkstelligt:*

```
 1 #include <stdio.h>
 2 #include <unistd.h>
 3 #include <fcntl.h>
 4
 5 int main(int argc, char*argv[]) {
 6   int fd = open("file.txt", O_CREAT|O_RDWR);
 7   close(1);
 8   dup(fd);
 9   printf("Hello world\n");
10   close(fd);
11   return 0;
12 }
```

Opgave 3: Scheduling

(a) Het Unix besturingsysteem kent drie toestanden waarin een proces zich kan bevinden, namelijk de toestanden *Running*, *Ready* en *Blocked*. Geef van ieder van de onderstaande toestandsovergangen aan of deze mogelijk of onmogelijk is. Als de overgang mogelijk is, geef dan een voorbeeld van een situatie waarin deze overgang plaats zou kunnen vinden. Als de overgang niet mogelijk is, leg dan uit waarom.

1. Van *Running* naar *Ready*
2. Van *Running* naar *Blocked*
3. Van *Ready* naar *Blocked*
4. Van *Ready* naar *Running*
5. Van *Blocked* naar *Running*
6. Van *Blocked* naar *Ready*

(b) Een user-level thread library maakt het mogelijk om binnen een UNIX-proces meerdere threads te maken zonder dat de kernel dit kan zien. M.a.w. voor de scheduler bestaat er slechts één proces terwijl binnen dit proces meerdere threads kunnen bestaan. Leg uit hoe dit problemen kan leveren met een standaard round robin scheduler voor een proces dat bestaat uit een thread dat CPU-bound is en een thread dat I/ O-bound is.

(c) Gegeven zijn 3 processen die zich aanmelden aan het besturingsysteem op verschillende tijdstippen. We gaan uit van tijdstippen in gehele seconden. Van ieder proces is het tijdstip van aanmelden (aankomst) en de duur van de executie bekend, en weergegeven in de volgende tabel.

proces	aankomst	executietijd
A	0	9
B	1	6
C	3	4

Bepaal de volgorde van executie van de processen en de gemiddelde wachttijd (waiting time) voor ieder van de volgende systemen.

- Systeem met non-pre-emptive *First Come First Served* (FCFS) scheduling
- Systeem met non-pre-emptive *Shortest Job First* (SJF) scheduling
- Systeem met pre-emptive *Round Robin* (RR) scheduling met eem time quantum van 2 seconden. Als een proces zich de eerste keer aan het operating system (OS) meldt, dan plaatst het OS dit proces achter aan de ready-queue (FIFO-queue).

Opgave 4: Page frame replacement

Ga in deze opgave uit van een virtual memory systeem met een pagetable van slechts 3 pages per proces. We beschouwen een proces dat memory pages benadert volgens de volgende "reference sequence".

ref=[1, 2, 4, 3, 1, 5, 6, 2, 6, 5, 1, 2, 3, 6, 5, 4].

Deze sequence geeft weer dat het proces gedurende executie eerst page 1 gebruikt, vervolgens page 2, daarna page 4, etc.

(a) Bepaal het minimaal aantal pagefaults, m.a.w. het aantal pagefaults dat een optimaal page replacement algoritme voor dit proces genereert. Geef na ieder van de bovenstaande 15 page-referenties aan welke pages in het fysieke geheugen aanwezig zijn.

(b) Ga uit van een FIFO page replacement algoritme. Geef voor ieder van de bovenstaande page-referenties aan welke 3 pages in het fysieke geheugen aanwezig zijn. Wat is het aantal pagefaults?

(c) Hoeveel pagefaults levert het LRU-algoritme voor de bovenstaande reference sequence?

Opgave 5: File systems

(a) We beschouwen een file system met slechts 16 disk blokken. Een file system checker bepaalt van ieder blok hoe vaak het in gebruik is en tevens hoe vaak het in de free-list voorkomt. De checker heeft de volgende counters bepaald:

Block Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
In use	1	1	0	1	0	1	2	1	1	0	0	1	1	1	0	0
Free	0	0	0	0	2	0	0	0	0	1	1	0	1	0	1	1

Zijn er fouten in dit file system? Zo ja, welke en hoe kunnen ze gerepareerd worden?

We gaan in de rest van deze opgave uit van een UNIX-like file system. In dit file system bevat iedere inode precies 10 entries, ieder ter grootte van 4 bytes. De block-size is 4096 bytes.

(b) Neem aan dat alle entries van een inode pointers zijn die direct wijzen naar data blokken. Wat is de maximale file size van dit file system? Leg uit.

(c) Neem nu aan dat 7 van de 10 entries direct wijzen naar data blokken. Van de overige drie entries wijst er één naar een *single indirect block*, één naar een *double indirect block* en één naar een *triple indirect block*. De indirect blokken hebben ook een grootte van 4096 bytes en worden volledig gevuld met pointers naar data blokken. Wat is nu de maximale file size van dit file system?

Opgave 6: deadlock avoidance

(a) Een bekend deadlock-avoidance algoritme is het Banker's algoritme. Hierin speelt het begrip 'veilige toekenning van resource aanvragen' een belangrijke rol. Hoe wordt zo'n veilige toekennning bepaald?

(b) Gegeven de volgende situatie. We hebben 8 resources van het type R (unsharable en nonpreemptive) en 3 processen A, B en C. De situatie met betrekking tot het aantal toegewezen resources van het type R en de claims (d.w.z. het maximum aantal benodigd) is op tijstip T als volgt.

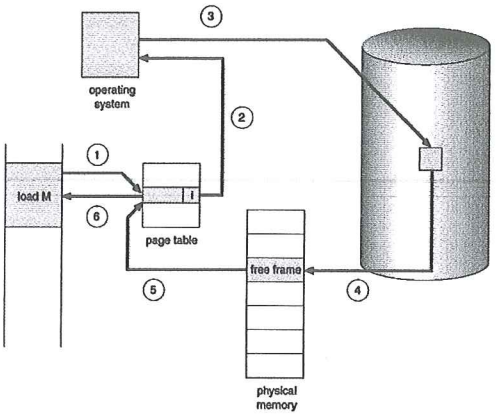
Proces	toegewezen	claim
A	3	5
B	1	4
C	2	6

Leg uit waarom een aanvraag van één eenheid door proces A wel gehonoreerd kan worden, terwijl dezelfde aanvraag komende van proces B niet gehonoreerd kan worden.

(c) Waarom wordt deze vorm van deadlock avoidance (zoals in onderdeel (b)) in de praktijk zelden toegepast?

Opgave 7: Virtueel geheugen

(a) Leg uit hoe virtueel geheugen werkt aan de hand van de 6 gemarkeerde punten uit de volgende figuur.



(b) Waarom is de grootte van een pageframe altijd een macht van twee?

(c) Beschouw een computer met 32 bits geheugenaddressering waarop een besturingsysteem draait met virtueel geheugen. De memory pages van dit systeem zijn 4KB groot. Geef commentaar op de volgende pagetable groottes:

- een pagetable van $2^{19} = 524288$ entries.
- een pagetable van $2^{20} = 1048576$ entries.
- een pagetable van $2^{21} = 2097152$ entries.

(d) Een gebruikelijke richtlijn voor het kiezen van de grootte van een swappartitie op een disk is twee maal de omvang van het fysiek aanwezige geheugen. Heeft het installeren van 6 GB swapspace op een 32 bits machine met 3 GB geheugen zin? Licht je antwoord toe.