

Prüfungsklausur Betriebssysteme

am 6. März 2006

Alle Aussagen sind so ausführlich wie nötig, aber so knapp wie möglich zu begründen!

1. Zur Verwaltung eines 4GByte großen virtuellen Adreßraums werde eine zweistufige Adreßumsetzung verwendet. Dabei werden die virtuellen Adressen aufgeteilt in ein 8-Bit-Feld der ersten Stufe und ein 10-Bit-Feld der zweiten Stufe, der Rest wird als Offset benutzt.
 - a) Wie groß ist eine Seite (in KByte), wie viele Seiten enthält der virtuelle Adreßraum?
 - b) Worin liegt der entscheidende Vorteil einer zweistufigen gegenüber einer einstufigen Adreßumsetzung? Hat diese Vorgehensweise auch Nachteile?
 - c) Was spricht für eine Aufteilung der Adresse auf die beiden Stufen in der angegebenen Weise, in welcher Situation ist eine umgekehrte Aufteilung (10 Bit für 1. Stufe, 8 Bit für 2. Stufe) günstiger?

7 Punkte

2. Die Adreßraumgröße eines virtuellen Speichers betrage 4GByte, die Seitengröße 4KByte; der Adreßraum werde durch eine einstufige Seitentabelle beschrieben. Ein Eintrag hat folgende Struktur, wobei COW für copy on write, W für schreibbar und P für present steht:

page frame number	undefined	COW	W	P
-------------------	-----------	-----	---	---

Gegeben sei folgender Ausschnitt einer Seitentabelle (leere Felder bedeuten nicht gesetztes Bit):

0x00000	0x1AD7			
0x00001	0x0A3D			P
0x00002	0x1237		W	P
	...			
0x002B3	0x2648			P
0x002B4	0x14D0	COW		P

Prüfen Sie den Zugriff auf die in nachstehender Tabelle gegebenen Adressen. Beschreiben Sie die Reaktion des Betriebssystems, falls der Zugriff nicht gestattet ist! Geben Sie andernfalls die zugehörige physische Adresse an!

8 Punkte

Zugriff auf	Art
0x00001AC1	schreibend
0x002B3ECA	lesend
0x00000620	lesend
0x00002E41	schreibend
0x002B4A57	schreibend

3. a) Welche beiden grundlegenden Bedingungen muß jeder Mechanismus zum Schutz kritischer Abschnitte erfüllen (Begriffe und Erklärungen!)?
- b) In WIKIPEDIA findet man die folgende Information:
 Der Dekker-Algorithmus ist ... eine vollständige Lösung des Problems des wechselseitigen Ausschlusses in der dezentralen Steuerung von Prozessen. Er vermeidet gegenseitiges Blok-

kieren und gewährleistet, dass stets genau ein Prozess in einen kritischen Abschnitt gelangen kann. Der Algorithmus kann mit folgendem Pseudo-C-Code schematisch beschrieben werden:

```
// globale Variablendeklaration
boolean flag0 = false, flag1 = false;
int turn = 0;

// Prozess #0
1 // ...
2 P:  flag0 = true;
3     while (flag1) {
4         if (turn == 1) {
5             flag0 = false;
6             goto P;
7         }
8     }
9     //<kritischer Abschnitt>
10    flag0 = false;
11    turn = 1;
12 // ...

// Prozess #1
1 // ...
2 P:  flag1 = true;
3     while (flag0) {
4         if (turn == 0) {
5             flag1 = false;
6             goto P;
7         }
8     }
9     //<kritischer Abschnitt>
10    flag1 = false;
11    turn = 0;
12 // ...
```

(Ende des Zitats.) Die jeweils links stehenden Zeilennummern gehören nicht zum Algorithmus; sie sollen lediglich zur Referenz für die folgende Aufgabenstellung dienen.

Untersuchen Sie, ob der Algorithmus in der angegebenen Form tatsächlich die erforderlichen Eigenschaften hat! Wenn ja, zeigen Sie dies! Wenn nein, so begründen Sie dies durch die Angabe eines entsprechenden Ablaufbeispiels mit kurzer Erläuterung! 10 Punkte

P.S. Obiger Code wurde 1 Tag nach der Klausur geändert, nicht allerdings der – zu den Änderungen nicht mehr passende! – ursprüngliche Korrektheits-„Beweis“.

4. Ein Betriebssystem habe eine einzige Betriebsmittelart zu verwalten, die in mehreren identischen Exemplaren (BME) vorliegt; dabei kann ein einzelnes Exemplar nur exklusiv genutzt werden. Anforderung, Zuteilung und Freigabe der BME geschehen nach folgenden Regeln:
- Ein Prozeß muß bei seinem Start angeben, wie viele BME er im Laufe seiner Existenz maximal gleichzeitig benötigt.
 - Ein Prozeß kann mehrere BME gleichzeitig anfordern; er kann jederzeit beliebig viele seiner ihm zugeteilten BME freigeben.
 - Überschreitet ein Prozeß bei einer Anforderung mit der Summe der angeforderten und bereits zugeteilten BME den Maximalwert, so wird er abgebrochen.
 - Sind alle angeforderten BME verfügbar, so werden sie dem Prozeß zugeteilt. Andernfalls erhält der Prozeß momentan kein einziges BME und wird blockiert.
 - Spätestens am Ende gibt ein Prozeß alle BME frei; beim Abbruch werden sie ihm entzogen.
- a) Welchen Vorteil hat diese Vorgehensweise im Vergleich zu einer bekannten ähnlichen Strategie? Worin liegt ihr entscheidender Nachteil?
- b) Das System verfüge über 12 BME. Vier Prozesse A, ..., D wollen maximal 4, 5, 5 bzw. 2 BME gleichzeitig nutzen. Gegenwärtig sind den Prozessen 2, 4, 2 bzw. 1 BME zugeteilt. Betrachten Sie in dieser Situation den Fall, daß zwei BME von Prozeß A angefordert werden bzw. alternativ von Prozeß B bzw. von Prozeß C. Entscheiden Sie in jedem der drei Fälle, ob die BME dem jeweiligen Prozeß zugeteilt werden, und beschreiben Sie die Konsequenzen für das Prozeßsystem! 9 Punkte

5. Wenn die Einplanbarkeit einer Taskmenge T mittels RMS durch Aufstellen eines Ablaufplans untersucht werden muß, so können leicht relativ lange (und damit unhandliche) Ablaufpläne entstehen. Daher schlägt jemand vor, die Task T_n mit der längsten Periode durch eine Task T_n' mit einer kürzeren Periode zu ersetzen – wodurch sich aber die Prioritätszuordnung nicht ändern soll – und die Ausführungszeit von T_n so anzupassen, daß T_n' zu derselben Auslastung wie T_n führt; die entstehende Taskmenge sei T' . Die Behauptung lautet dann: „Mittels RMS ist die Taskmenge T genau dann einplanbar, wenn es T' ist.“

a) Untersuchen Sie die Gültigkeit dieser Behauptung! Benutzen Sie dazu als Beispiel die Taskmenge

$$T = \{T_1 = (5; 7), T_2 = (6; 24)\};$$

dabei bezeichnet der erste Parameter die Ausführungszeit und der zweite Parameter die Periodendauer; die Tasks mögen die üblichen Voraussetzungen für RMS erfüllen.

b) Warum ist eine Periodendauer von 15 für T_1 (alle anderen Werte unverändert) *von vornherein* nicht zur Untersuchung des Problems geeignet? 7 Punkte

Hinweis. Ist zur Untersuchung der Einplanbarkeit einer Taskmenge mittels RMS ein Ablaufplan erforderlich, so genügt es, das Intervall $[0, t]$ zu betrachten, sofern alle Tasks zum Zeitpunkt 0 gestartet werden (t : Periodendauer der Task mit der niedrigsten Priorität).

6. Geben Sie die Ausgabe an, die bei der Abarbeitung des nachfolgend aufgeführten Programms entsteht! Nehmen Sie dabei an, daß 1 die Prozeßnummer (PID) des zugehörigen Prozesses ist und weitere Prozeßnummern bei `fork` anschließend fortlaufend vergeben und nicht wiederverwendet werden.

Hinweis. Die `printf`-Anweisung in dem Programm bewirkt eine zeilenweise Ausgabe der Form

```
PID 17 -> i = 31
```

falls der ausdrückende Prozeß 17 als PID hat und 31 der aktuelle Wert von `i` ist. Der Parameter 0 bei `wait(0)` bedeutet, daß der `exit`-Status ignoriert wird. 7 Punkte

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

unsigned i = 3;

int main (void)
{
    while (i > 0)
    {
        i = i-1;
        printf ("PID %u -> i = %u\n", getpid(), i);
        if (fork() > 0)
            wait (0);
    }

    return 0;
}
```