Scheinklausur

22. Februar 2000
Bearbeitungszeit: 90 Minuten

Hinweise

Sie dürfen erst mit der Bearbeitung der Aufgaben beginnen, nachdem die Klausuraufsicht die Bearbeitungszeit gestartet hat. Nutzen Sie daher die ersten Minuten, um die Aufgaben komplett und sorgfältig zu lesen. Wenn Sie eine Aufgabenstellung nicht verstehen, fragen Sie bitte sofort bei der Klausuraufsicht nach.

Als Hilfsmittel sind nur Schreibgeräte erlaubt.

Die Klausur ist so gestellt, daß Sie nicht alle Aufgaben in der zur Verfügung stehenden Zeit lösen können. Konzentrieren Sie sich daher zunächst auf die Aufgaben, die Ihnen „einfach" erscheinen. Antworten Sie kurz und prägnant.

Die Gesamtpunktzahl aller Aufgaben beträgt 85 Punkte. Wenn Sie mindestens 35 Punkte erreichen, haben Sie die Klausur bestanden.

Viel Erfolg!!!

Aufgabe 1 (5+5 Punkte)

In einem modernen Betriebssystem bezeichnet man ein gerade ausgeführtes Programm als Prozeß. Ein solcher Prozeß wird durch einen Prozeßkontrollblock (process control block, PCB) verwaltet.

a) Erklären Sie kurz die Bedeutung der folgenden Bestandteile des PCB:

   (i) Prozeßzustand (process state)
   (ii) Programmzähler (program counter)
   (iii) Registerzustand (register state)
   (iv) Schedulinginformationen (scheduling information)

b) Zeichnen Sie ein Prozeßzustandsdiagramm mit den Übergängen zwischen den *fünf* Zuständen für *einen* Prozeß. Beachten Sie dabei die folgenden Regeln und notieren Sie neben jedem Übergang die angewandte Regel:

   (i) Nach der Initialisierung ist ein Prozeß ausführbereit.
   (ii) Ein Prozeß kann sich selbst beenden.
   (iii) Wenn ein Prozeß an einem Deadlock beteiligt ist, kann das Betriebssystem entscheiden, diesen Prozeß abzubrechen.
   (iv) Der Scheduler wählt einen ausführbereiten Prozeß aus und läßt ihn ablaufen.
   (v) Wenn ein ausgeführter Prozeß auf externe Ressourcen wartet, wird er vom Scheduler unterbrochen.
   (vi) Ein Interrupt kann einen Prozeß preemptiv unterbrechen und einen Prozeßwechsel auslösen.
   (vii) Nachdem ein wartender Prozeß seine Ressource erhalten hat, ist er wieder ausführbereit.

   Erklären Sie kurz die einzelnen Zustände.

Aufgabe 2 (10+6 Punkte)

In der Vorlesung haben Sie den sogenannten Bakery-Algorithmus kennengelernt. Das folgende Programm implementiert den Algorithmus mit zwei Funktionen `lock` und `unlock`. Dabei wird angenommen, daß die Variablen vom Typ `int` beliebig große Zahlen speichern können.

```
  1  #define MAX_PROCESSES   5
  2  #define TRUE            1
  3  #define FALSE           0
  4
  5  int number[ MAX_PROCESSES ];
  6  int choosing[ MAX_PROCESSES ];
  7
  8  int get_process_id();    /* liefert die Prozeßnummer des laufenden Prozesses */
  9  int max( int a, int b ); /* bestimmt das Maximum von a und b */
 10
 11  void lock()
 12  {
 13     int pid = get_process_id();
 14     int current_max = 0;
 15     int i;
 16
 17     /* Nummer ziehen */
▶ 18     choosing[ pid ] = TRUE;
 19     for( i = 0; i < MAX_PROCESSES; i++ )
 20        current_max = max( current_max, number[ i ] );
 21     number[ pid ] = current_max + 1;
▶ 22     choosing[ pid ] = FALSE;
 23
 24     /* mit anderen Prozessen synchronisieren */
 25     for ( i = 0; i < MAX_PROCESSES; i++ ) {
▶ 26        while( choosing[i] );
 27        while( number[i] != 0
 28               && ( ( number[i] < number[ pid ] )
 29                    || ( number[i] == number[ pid ] && i < pid ) ) );
 30     }
 31     /* ab hier ist der kritische Bereich gesperrt */
 32  }
 33
 34  void unlock()
 35  {
 36     int pid = get_process_id();
 37     number[ pid ] = 0;
 38     /* ab hier ist der kritische Bereich wieder frei für andere Prozessoren */
 39  }
```

a) Ein naiver Student hat den Algorithmus etwas vereinfacht und die Zeilen 18, 22 und 26 entfernt. Funktioniert der Algorithmus dann immer noch? Begründen Sie Ihre Antwort. Können Sie ein Gegenbeispiel angeben?

b) Beweisen Sie die folgende Aussage:

Wenn der Prozeß P_i im kritischen Bereich ist, gilt:

$$\bigwedge_{k \neq i} (\texttt{number}[k] = 0) \ \vee \ (\texttt{number}[i] < \texttt{number}[k]) \ \vee \ \big((\texttt{number}[i] = \texttt{number}[k]) \ \wedge \ (i < k)\big)$$

Aufgabe 3 (4+6+4 Punkte)

In der Vorlesung wurde das Reader/Writer-Problem vorgestellt.

a) Erklären Sie, was man unter dem Reader/Writer-Problem versteht. Was sind die Besonderheiten bei den unterschiedlichen Ausprägungen dieses Problems?

b) Geben Sie einen Algorithmus für das allgemeine Reader/Writer-Problem in C-Notation an. Verwenden Sie dabei Semaphoren mit den Funktionen `init`, `wait` und `signal`. Welche Vor- und Nachteile hat Ihre Lösung?

c) Nennen Sie vier Beispiele für Stellen, an denen das Reader/Writer-Problem in einem Betriebssystem auftritt.

Aufgabe 4 (3+4+4+2+2 Punkte)

Der Intel i386 und seine Nachfolger (allgemein als x86 bezeichnet) unterstützen im sogenannten Protected Mode sowohl Segmenting als auch Paging. Zur Segmentierung bietet diese Prozessorfamilie die Segmentregister CS, DS, ES, FS, GS und SS. Jedes Segmentregister ist dabei 16 Bit groß. 3 Bit hiervon sind für diese Aufgabe nicht von Bedeutung. Mit den verbleibenden 13 Bit wird eines von 16384 Segmenten ausgewählt. Ein Segment kann dabei bis zu 4 GB groß sein.

Außerdem besitzt diese Prozessorfamilie eine Pagingeinheit, die über eine zweistufige Page-Tabelle logische Adressen auf physikalische Adressen abbildet. Die erste Page-Tabelle hat dabei Platz für 1024 Verweise auf die zweite Page-Tabelle. Die max. 1024 Page-Tabellen der zweiten Ebene bieten jeweils Platz für 1024 Pages. Eine Page ist 4096 Byte groß.

a) Ältere Betriebssysteme (z.B. OS/2 1.x, Windows 3.x) haben Segmente verwendet, während modernere Systeme (Windows NT, Linux) ein sogenanntes flaches Speichermodell verwenden. Wieso haben die Entwickler diesen Schritt wohl gewählt?

b) Zeichnen Sie graphisch, welche Schritte in der CPU vorgenommen werden, um die logische Adresse 1887 auf die physikalische Adresse 5983 abzubilden (Paging).

c) Zeichnen Sie graphisch, welche Schritte in der CPU vorgenommen werden, um die logische Adresse 1:1887[1] auf die physikalische Adresse 5983 abzubilden (Segmenting).

d) Hochleistungsrechner (z.B. Systeme von Cray) haben bis vor kurzem sowohl auf Segmenting als auch auf Paging verzichtet. Welche Gründe kann es dafür geben?

e) Die x86-Prozessoren besitzen einen TLB. Was ist ein TLB und wofür wird er verwendet?

[1] man könnte auch (1, 1887) schreiben

Aufgabe 5 (5+12+4 Punkte)

Betrachtet wird ein Betriebssystem mit fünf Prioritätsklassen:

Priorität	*Name*	*Verfahren*	*preemptiv?*	*Zeitquantum*
höchste	CLASS_REALTIME	Round Robin	nein	–
·	CLASS_HIGH	Round Robin	ja	1
·	CLASS_NORMAL	Round Robin	ja	2
·	CLASS_LOW	Round Robin	ja	4
niedrigste	CLASS_BATCH	Shortest Job First	ja	–

a) Erklären Sie in einfachen Worten, was man bei diesem Betriebssystem als Programmierer beachten muß. Überlegen Sie sich, welche Prioritätsklasse für welche Art von Aufgaben geeignet ist und welche Restriktionen es ggf. gibt.

b) Gegeben seien die in der folgenden Tabelle dargestellten Prozesse. Stellen Sie tabellarisch die fünf Warteschlangen für die ersten 20 Zeiteinheiten dar. Notieren Sie für jeden Prozeß wie lang er noch benötigt (z.B. A(5) bedeutet: Prozeß A benötigt noch 5 Zeiteinheiten). Kreisen Sie jeweils den gerade laufenden Prozeß ein.

Prozeß	*Ankunftszeit*[a]	*Priorität*	*Laufzeit*
A	0	CLASS_BATCH	100
B	3	CLASS_HIGH	4
C	4	CLASS_HIGH	5
D	5	CLASS_REALTIME	2
E	6	CLASS_REALTIME	1
F	7	CLASS_NORMAL	10
G	8	CLASS_BATCH	96
H	9	CLASS_LOW	2

Anmerkung:
[a] Ein Prozeß wird jeweils erst zur folgenden Zeiteinheit beachtet. Prozeß A wird also in Zeiteinheit 1 das erste Mal ausgeführt.

c) Berechnen Sie die Wartezeit für alle Prozesse. Als Wartezeit sei hier die Zeit bezeichnet, die ein Prozeß von seiner Ankunftszeit bis zu seinem Abschluß benötigt. Sie ergibt sich also als die Summe aus der Laufzeit und der Zeit, die der Prozeß wartend in einer Warteschlange verbringt.

Aufgabe 6 (2+2+1+2+2 Punkte)

Ein Student macht folgende Aussagen:

(i) „Ein Deadlock kann nur auftreten, wenn mindestens zwei verschiedene Betriebsmittel vorliegen."
(ii) „Ein Deadlock tritt immer auf, wenn zwei Prozesse das gleiche Betriebsmittel benötigen."
(iii) „Deadlocks treten auf Computern mit Windows NT nie auf."
(iv) „Deadlocks treten immer dann auf, wenn ein Prozeß zwei Betriebsmittel gleichzeitig benutzt."
(v) „Ein Deadlock tritt auf, wenn zwei Prozesse in umgekehrter Reihenfolge auf die gleichen zwei Betriebsmittel zugreifen wollen und dabei beide jeweils für einen kurzen Moment beide Betriebsmittel benötigen."

Bewerten Sie die Aussagen des Studenten. Wenn eine Aussage falsch ist, geben Sie ein Gegenbeispiel an und erklären Sie dieses. Wenn eine Aussage korrekt ist, begründen Sie Ihre Meinung.

Anmerkung: Ein Betriebsmittel sei in dieser Aufgabe eine Ressource, die genau einmal vorhanden ist und die zu jedem Zeitpunkt nur exklusiv von maximal einem Prozeß belegt werden kann.