AI-3 WS 06	**Klausur Betriebssysteme (BS) 24.01.2007**	Hbn
Seite: 1 / 8	**Name, Vorname:**	**Matr.-Nr.:**

Bewertungsschema:
Zuordnung Aufgaben – Punkte:

A1	A2	A3	A4	A5	A6	A7	Summe
6	4	4	8	7	5	8	42

Aufgabe A1

6 Punkte

a) Was ist ein Thread? Geben Sie einige Eigenschaften an.
b) Welche Vorteile bietet die Verwendung von Threads für die Programmierung paralleler Abläufe im Vergleich zu mehreren Prozessen?
c) Was ist der Unterschied zwischen „Kernel Level Threads“ und „User Level Threads“?

Aufgabe A2

4 Punkte

Analysieren Sie das folgende C-Programm:

```
main () {
  int i=0;
  if (fork() == 0)
        while (1) {
              sleep (2);
              i=i+1;
        }
  sleep (2);
  printf("i = %d",i);
 }
```

Ist die Bildschirmausgabe des Programms eindeutig? Wenn nein: Warum nicht?
Wenn ja: Wie lautet sie?

Aufgabe A3 4 Punkte

Durch welche Mechanismen verhindern moderne Betriebssysteme im Rahmen des Prozess-Scheduling ein „Verhungern“ von Prozessen?

Aufgabe A4 8 Punkte

Wir betrachten ein Erzeuger-Verbraucher-System mit beschränktem Puffer. Zur Synchronisation werden Semaphore eingesetzt.

```
Binäres Semaphor       S = 1    /* Synchronisation des Pufferzugriffs */
Allgemeines Semaphor B = 0    /* Anzahl belegter Pufferplätze  */
Allgemeines Semaphor F = N    /* Anzahl freier Pufferplätze  */
```

Erzeuger-Code:

```
<Erzeugen eines Datenpakets>;
P(F);
P(S);
<Datenpaket im Puffer speichern>
V(S);
V(B);
```

Verbraucher-Code:

```
P(B);
P(S);
<Datenpaket aus Puffer nehmen>
V(S);
V(F);
<Verbrauchen des Datenpakets>;
```

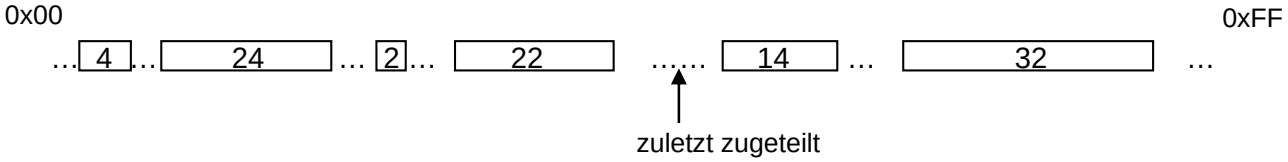
Wir nehmen nun an, dass nach wie vor beliebig viele Verbraucher, aber nur genau zwei Erzeuger (Erzeuger1 und Erzeuger2) existieren. Die beiden Erzeuger sollen sich über geeignete Semaphore so synchronisieren, dass sie jeweils abwechselnd genau zwei Datenpakete im Puffer ablegen (also Erzeuger1 zwei Datenpakete, danach Erzeuger2 zwei Datenpakete, danach wieder Erzeuger1 zwei Datenpakete, usw.), falls diese Pakete im Puffer Platz finden.

Definieren Sie geeignete zusätzliche Semaphore (mit Initialisierung) und geben Sie den entsprechend modifizierten Erzeuger-Code für Erzeuger1 und Erzeuger 2 an!

7 Punkte

Aufgabe A5

In einem System zur Hauptspeicherverwaltung mit dynamischer Partitionierung wird jedem Prozess eine Partition unterschiedlicher Größe zugewiesen. Dabei sehe eine aktuelle Speicherbelegung folgendermaßen aus, wobei hier nur die freien Speicherbereiche (in Mbyte) angegeben sind:



a) Platzieren Sie einen 20 Mbyte großen Prozess unter Verwendung folgender Algorithmen (Ergebnis bitte in der Graphik markieren):

(1) First-Fit

(2) Next-Fit

(3) Best-Fit

b) Welches Verfahren liefert in der Regel das beste und welches das schlechteste Ergebnis (hinsichtlich Speicherausnutzung und Ausführungsgeschwindigkeit)? Begründen Sie ihre Antwort kurz!

Aufgabe A6 5 Punkte

Betrachten Sie einen seitenorientierten virtuellen Speicher mit einer Seitengröße von 1024 Bytes. Gegeben ist der folgende Ausschnitt der Seitentabelle eines Prozesses:

Nummer der virt. Seite (VPN)	0	1	2	3	4
Seitenrahmennummer	1	0	0	0	2
„Valid“-Flag	true	false	true	false	true

Gegeben sind die folgenden virtuellen Adressen: 100, 1000, 2000, 3000

a) Geben Sie für diese Adressen die zugehörigen realen Hauptspeicheradressen an (falls sich die Seite im Hauptspeicher befindet).
b) Welche der angegebenen virtuellen Adressen löst bei einem Zugriff einen Seitenfehler („Page fault“) aus?

8 Punkte

Aufgabe A7

Wir betrachten ein kleines I-Node-basiertes Dateisystem (UNIX-ähnlich), das insgesamt 16 Blöcke für Benutzerdateien zur Verfügung stellt und das sich in folgendem Zustand befindet:

I-Node-Liste (Auszug):

I-Nodenr.	Plattenblockverweise
0	7, 11, 10, 0
1	13, 5
2	4, 10, 9
3	14, 15, 5

Freibereichsliste (hier: Liste mit Blocknummern freier Blöcke):
(2, 3, 8, 12)

Durch einen Systemabsturz ist das Dateisystem inkonsistent geworden. Sie wissen lediglich, dass alle I-Nodes korrekt durch einen Verzeichniseintrag referenziert werden.

a) Führen Sie einen Algorithmus aus, mit dem widersprüchliche Block-Informationen innerhalb der I-Node-Liste sowie zwischen I-Node-Liste und Freibereichsliste ermittelt werden können.
b) Geben Sie alle gefundenen widersprüchlichen Informationen an und erklären Sie jeweils, wie Sie diesen Zustand bereinigen würden!

Lösungen:

A1:

Lösung: 6 P.

a) *Ein leichtgewichtiger Unterprozess, der sich innerhalb eines Prozesses denselben Adressraum mit anderen Threads teilt, jedoch einen eigenen Ausführungsfaden besitzt (Befehlszähler, Stack + Registersatz).*

b) *Vorteile:*
- *Threads können schneller erzeugt werden als Prozesse*
- *Schnellere Umschaltung zwischen Threads als zwischen Prozessen (Kontext-wechsel)*
- *Gemeinsamer Zugriff auf dieselben Daten (gemeinsamer Adressraum!)*
- *Parallele und unabhängige Abläufe innerhalb desselben Programmcodes sind möglich.*

c) *Kernel Level Threads werden durch das Betriebssystem geschedult, User Level Threads durch den Benutzerprozess.*

A2:

Ja, da Eltern- und Kindprozesse vollständig getrennte Adressräume besitzen (die Variable i des Elternprozesses wird nicht beeinflusst) und der Kindprozess den „printf"-Befehl nicht ausführt, da er in der „Endlosschleife" festhängt!
Ausgabe: „i = 0"

A3:
- *Preemptives Multitasking: Zwangsweises Entziehen der CPU mittels Interrupt („Round Robin" - Zeitscheibenverfahren)*
- *Periodische Neuberechnung von Prioritäten mit Erhöhung der Priorität in Abhängigkeit vom Alter und der bisherigen Wartezeit eines Prozesses*

A4:

```
Binäres Semaphor    Turn1 = 1    /* Erzeuger1 hat Zugriff */
Binäres Semaphor    Turn2 = 0    /* Erzeuger2 hat Zugriff */
```

Erzeuger1-Code:

```
<Erzeugen zweier Datenpakets>;
P(Turn1);
P(F); P(F);
P(S);
<2 Datenpakete im Puffer speichern>
V(S);
V(B); V(B);
V(Turn2);
```

Erzeuger2-Code:

```
<Erzeugen zweier Datenpakets>;
P(Turn2);
P(F); P(F);
P(S);
<2 Datenpakete im Puffer speichern>
V(S);
V(B); V(B);
V(Turn1);
```

A5:

a)

(1) First-Fit

(2) Next-Fit

(3) Best-Fit

b)

Bestes Verfahren: First-Fit!

- *Schnellstes Verfahren wegen geringster Suchanforderungen (u.a. keine Verwaltung eines Zeigers auf die letzte eingefügte Partition nötig)*
- *Geringer „Verschnitt" (hinten bleibt meist noch Platz für große Prozesse)*

Schlechtestes Verfahren: Best-Fit!

- *Aufwendige Suche mit Überprüfung <u>aller</u> freien Bereiche*
- *Weil immer kleine Speicherreste bleiben (→ neue kleine Partitionen), die nicht genutzt werden können, muss das Betriebssystem am häufigsten umsortieren!*

A6:

a) Berechnung der virtuellen Seitennummer VPN (Index für den Seitentabellenzugriff):

VPN =floor (virtuelle Adresse / 1024)

Berechnung des Offsets (Bytenummer innerhalb der virtuellen Seite):

Offset = virtuelle Adresse mod 1024

Berechnung der realen Hauptspeicheradresse:

*Reale Adresse = Seitenrahmennummer * 1024 + Offset*

Virt. Adresse	*100*	*1000*	*2000*	*3000*
VPN	*0*	*0*	*1*	*2*
Offset	*100*	*1000*		*952*
Reale Adresse	*1124*	*2024*	*-*	*952*

b) virtuelle Adresse 2000 → virtuelle Seitennr. 1 → valid-Flag ist false!

A7:

Aufbau einer Tabelle durch Inspektion aller I-Nodes und der Freibereichsliste (ein Eintrag bedeutet die Anzahl an Verweisen auf diesen Block in der jeweiligen Liste):

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*Blocknummer*
1	*0*	*0*	*0*	*1*	*2*	*0*	*1*	*0*	*1*	*2*	*1*	*0*	*1*	*1*	*1*	*Block in I-Nodeliste referenziert*
0	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*Block in Freibereichsliste referenziert*

- *Block 1: Vermisster Block → In Freibereichsliste eintragen*
- *Block 5: Doppelter Datenblock → Block in zusätzlichen freien Block kopieren und einem der beiden I-Nodes zuweisen*
- *Block 6: Vermisster Block → In Freibereichsliste eintragen*
- *Block 10: Doppelter Datenblock → Block in zusätzlichen freien Block kopieren und einem der beiden I-Nodes zuweisen*