LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 1
19.10.2004

Informatik III

Abgabetermin: 25.10.2004, 12:00 Uhr s.t.

Achtung: Die Anmeldung zu den Übungsgruppen ist noch bis Donnerstag, 21.10., 20:00 Uhr freigeschaltet. Bitte beachten Sie dazu die Hinweise auf dem Merkblatt. Geben Sie bei der Abgabe Ihrer Hausaufgaben immer die Nummer Ihrer Übungsgruppe, Ihren Namen und Ihre Matrikelnummer an. Ihre Lösung wird sonst nicht gewertet. Falls nicht anders angegeben, fertigen Sie Ihre Hausaufgaben bitte handschriftlich an. Abgaben in Teams sind nicht zulässig. Zur Abgabe stehen Ihnen Briefkästen an der Bibliothek in der Theresienstraße und in der Garderobe in der Oettingenstraße zur Verfügung. Bitte halten Sie den Abgabetermin (siehe oben) ein.

Lesen: —

Aufgabe 1: (H) Grundlagen moderner Betriebssysteme (20 Pkt.)

a. Welche Charakteristika, die bei der Arbeit mit einem Desktop-PC heute selbstverständlich sind, werden erst durch das Betriebssystem ermöglicht?

b. Leiten Sie daraus die Haupt-Aufgaben eines modernen Betriebssystems ab.

c. Erklären Sie kurz die folgenden Begriffe aus der Betriebssystem-Welt:

   (i) Multi-Tasking
   (ii) Scheduling und Dispatching
   (iii) Speicherverwaltung
   (iv) Virtueller Speicher
   (v) Dateisystem
   (vi) Shell

d. Vergleichen Sie grob die Betriebssysteme MS DOS, Windows XP und Linux.

Aufgabe 2: (T) Betriebssystem-Schichten (8 Pkt.)

In dieser Aufgabe sollen Sie sich klar machen, wie das Betriebssystem mit den restlichen Komponenten eines Rechners verbunden ist. Finden Sie für die folgenden Begriffe eine sinnvolle Kategorisierung, und stellen Sie die sich ergebenen Rechner-Schichten grafisch dar:

Web-Browser, CPU, Dateisystem, Office-Programme, Ein- und Ausgaberoutinen, Festplatte, Hauptspeicher, Gerätemanagement, Benutzer, Scheduler, Shell, Speicherverwaltung, Unix-Compiler, Drucker, Windows-Systemsteuerung.

Aufgabe 3: (T) Betriebssystemtypen (8 Pkt.)

a. Definieren Sie die essenziellen Eigenschaften der folgenden Typen von Betriebssystemen: *Batch, Interaktiv, Time Sharing, Echtzeit, Verteilt.*

b. Was ist der Hauptvorteil von Multiprogramming?

c. Was sind die wesentlichen Vor- bzw. Nachteile eines Multiprozessor-Systems?

Aufgabe 4: (P) Einfache UNIX-Kommandos (9 Pkt.)

In dieser Aufgabe sollen Sie sich mit dem Betriebssystem UNIX vertraut machen. Zu einem Befehl `cmd` erhalten Sie durch Eingabe von `man cmd` oder `info cmd` eine Erläuterung.

a. Beschreiben Sie kurz Sinn und Zweck der folgenden Kommandos inklusive der angegebenen Flags. Flags sind durch vorangestelltes „–“ gekennzeichnet. Sie können beim `man`-Aufruf weggelassen werden.

   (i) `ls -l`

   (ii) `rm -i datei`

   (iii) `mkdir name`

   (iv) `less datei`

   (v) `wc datei`

b. Finden Sie eine Kategorisierung für die folgenden Befehle mit zwei Kategorien:

   `ls`, `kill`, `rm`, `mv`, `mkdir`, `killall`, `cp`, `fork`

Aufgabe 5: (P) Grundlagen der Shell-Programmierung (20 Pkt.)

Alle Unix-Shells stellen Konstrukte zur Verfügung, um so genannte Shell-Skripte zu programmieren. In dieser Aufgabe werden einige kleine Skripte für die gebräuchlichste Linux-Shell `bash` erstellt.

So funktioniert es:

– Starten Sie die Konsole (Befehlsfenster/Terminal), zum Beispiel indem Sie den Befehl `xterm` ausführen.

– Um herauszufinden, mit welcher Shell Sie arbeiten, geben Sie `echo $SHELL$` ein.

– Wenn Sie ein beliebiges Skript geschrieben haben, muss es ausführbar gemacht werden. Das funktioniert mit dem Befehl `chmod +x filename` (wobei `filename` der Dateiname des Skripts ist).

– Durch Eingabe von `./filename` wird das Skript gestartet.

Einige Linux-Konsolenbefehle:

Befehl	Beschreibung
`echo "mein text"`	gibt "mein text" auf der Konsole aus
`ls`	listet alle Dateien im aktuellen Verzeichnis auf
`cp quelle ziel`	kopieren
`mv alt neu`	umbennenen bzw. verschieben
`rm datei`	löschen
`grep 'text' datei`	sucht in einer Datei nach der Zeichenkette "text"
`cat datei`	gibt den Inhalt einer Datei auf dem Bildschirm aus
`head datei`	gibt einige Zeilen vom Dateianfang auf dem Bildschirm aus
`tail datei`	gibt einige Zeilen vom Dateiende auf dem Bildschirm aus
`file datei`	gibt aus, von welchem Dateityp eine Datei ist
`sort datei`	sortiert die Zeilen einer Datei alphabetisch
`wc -l datei`	zählt die Zeilen in einer Datei
`wc -w datei`	zählt die Wörter in einer Datei
`wc -m datei`	zählt die Anzahl der Buchstaben in einer Datei
`read var`	fordert den Benutzer zu einer Eingabe auf

a. Schreiben Sie ein Skript, das den Text "Hello World" zunächst in einer Variablen ablegt und dann auf der Konsole ausgibt.

b. Gegeben ist folgendes Shell-Skript:

```
#!/bin/sh
x=Raum
echo "$xschiff"
```

Welches Problem besteht? Wie sieht die Lösung aus?

c. Schreiben Sie (mit Hilfe von Pipes) ein Shell-Skript, das ausgibt, in wie vielen Zeilen der Datei `file.txt` das Wort "Text" vorkommt.

d. Schreiben Sie ein Shell-Skript `smart`, das erkennt, ob eine beliebige Datei eine ZIP-komprimierte Datei (Dateityp `.zip`) oder eine Text-Datei (Dateityp `.txt`) ist und diese Datei automatisch entweder korrekt entpackt oder (im Falle einer Text-Datei) im XEmacs-Editor öffnet.

e. Schreiben Sie ein Shell-Skript `select`, das den Benutzer nach dem Aufruf fragt, ob seine Datei im XEmacs- oder im KWrite-Editor geöffnet werden soll.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 14
26.01.2005

Informatik III

Abgabetermin: keine Abgabe erforderlich

Achtung:

- Anmeldung zur Klausur: 31.01. bis 04.02.2005
- Veröffentlichung der Teilnehmerlisten: 07.02.2005
- Letzte Nachmelde-Möglichkeit: 08.02.2005 (10:00 bis 17:00 Uhr)
- Klausur: 12.02.2005 (9:00 Uhr s.t. bis 12:00 Uhr)
- Klausureinsicht: 18.02.2005 (10:00 bis 11:00 Uhr)

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 3 bis 6 (wiederholen)

Aufgabe 60: (H) Dateisystem-Recovery (15 Pkt.)

Stellen Sie sich folgende Situation vor: Die Datei `myfile.html` (Dateigröße: 1,5 KB, I-Node-Nummer: 522, Datenblöcke: 20.392 und 20.393) liegt im Verzeichnis `myfolder` (Datenblock: 12.440). Der Referenzzähler im I-Node mit der Nummer 522 steht auf 1, das heißt es gibt keine weiteren Hardlinks auf `myfile.html`. Nun sollen nacheinander die folgenden beiden Aktionen durchgeführt werden:

I.) Ein Hardlink auf die Datei `myfile.html` soll im Verzeichnis `myhome` (Datenblock: 16.500) angelegt werden.

II.) Anschließend soll die Datei `myfile.html` aus dem Ordner `myfolder` entfernt werden.

Nehmen Sie für diese Aufgabe vereinfachend an, dass das **Anlegen eines Hardlinks** im Dateisystem allgemein in zwei Schritten erfolgt:

1. Zuerst wird der Datenblock des Verzeichnisses, in dem der Hardlink angelegt wird, neu geschrieben.

2. Danach wird der I-Node der Datei, auf die der Hardlink erstellt wurde, aktualisiert.

a. Der verwendete Datenträger sei mit dem Ext2-Dateisystem ohne Journaling formatiert. Die Blockgröße betrage 1 KB.

   (i) Wie sieht der I-Node 522 aus, **bevor** die Aktionen I und II zur Ausführung kommen? (Sie können sich bei der Darstellung des I-Nodes auf den Referenzzähler und die Verweise auf Daten- und Indirektionsblöcke beschränken.)

   (ii) Wie sieht der I-Node 522 **nach** der planmäßigen Ausführung von **Aktion I** (Hardlink anlegen) aus?

   (iii) Wie sieht der I-Node 522 **nach** der planmäßigen Ausführung von **Aktion II** (Datei aus `myfolder` entfernen) aus?

(iv) Während der Aktion I des Szenarios (Hardlink anlegen) falle die Stromversorgung aus, und zwar unmittelbar nachdem der Teilschritt 1 (Datenblock neu schreiben) beendet wurde. Der Rechner wird ohne Dateisystem-Scan neu gestartet. Was passiert, wenn die Datei `myfile.html` jetzt aus dem Ordner `myfolder` entfernt wird (mit Begründung)?

b. Nehmen Sie jetzt an, Ihr Datenträger wurde mit dem Ext3-Dateisystem formatiert. Als Journal-Art wurde `data=writeback` konfiguriert. Beschreiben Sie, wie das Dateisystem im Falle des Stromausfalls nach dem Neustart reagiert.

Aufgabe 61: (K) Deadlocks: Banker-Algorithmus (10 Pkt.)

Der **Banker's Algorithm** von Dijkstra soll in dieser Aufgabe wie folgt angwendet werden: Eine Bank stellt ihren Kunden einen Kreditrahmen (KR) zur Verfügung. Damit die Kreditwünsche der Kunden erfüllt werden können, stellt die Bank 9.000 Euro zur Kreditvergabe bereit. Folgende Tabelle zeigt die Kreditrahmen der Kunden und die einzelnen Kreditanfragen:

Kunde	KR (in TEUR)	Anfragen (in TEUR)
A	6	2, 1, 3
B	3	2, 1
C	8	3, 4, 1
D	4	3, 1

a. Finden Sie eine möglichst kurze Verfügungsreihenfolge, die alle Kreditwünsche befriedigt.

b. Übertragen Sie das Verfahren auf das Problem der Deadlocks in Betriebssystemen.

c. Genügt es, um einen Deadlock sicher vermeiden zu können, nur den nächsten Folgezustand zu berücksichtigen?

d. Diskutieren Sie die tatsächliche Anwendung des Banker-Algorithmus in einem Betriebssystem.

Aufgabe 62: (K) Multilevel Feedback Queuing (MLFQ) (12 Pkt.)

Gegeben seien die Prozesse $P_1, \ldots, P_5$ mit den folgenden Ankunfts– und Rechenzeiten:

Prozess	Ankunftszeitpunkt	Rechenzeit
P_1	0	3
P_2	0	7
P_3	3	4
P_4	7	5
P_5	8	2

Hierbei gilt: Trifft ein Prozess zum Zeitpunkt t ein, so wird er erst zum Zeitpunkt $(t + 1)$ berücksichtigt. Damit verbringt jeder Prozess mindestens eine Zeiteinheit im Zustand wartend. Wird ein Prozess zum Zeitpunkt t' unterbrochen, so reiht er sich auch zum Zeitpunkt t' wieder in die Warteschlange ein. Sind zwei Prozesse absolut identisch bezüglich ihrer relevanten Werte, so wird nach der Prozess–ID entschieden (eine niedrigere Prozess–ID wird bevorzugt).

a. Geben Sie für die Strategie **Multilevel Feedback Queuing (MLFQ)** in Form eines Gantt–Charts die Zuteilung der Rechenzeit an die Prozesse an. Gehen Sie von drei Prioritätsklassen mit folgenden Werten aus:

Priorität	Verfahren
0	Round-Robin mit Quantum 2
1	Round-Robin mit Quantum 4
2	First Come First Serve (FCFS)

b. Geben Sie für die Prozesse P_1 und P_4 jeweils die Verweilzeit und die Wartezeit an.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 13
19.01.2005

Informatik III

Abgabetermin: keine Abgabe erforderlich

Achtung:

- Anmeldung zur Klausur: 31.01. bis 04.02.2005
- Veröffentlichung der Teilnehmerlisten: 07.02.2005
- Letzte Nachmelde-Möglichkeit: 08.02.2005 (10:00 bis 17:00 Uhr)
- Klausur: 12.02.2005 (9:00 Uhr s.t. bis 12:00 Uhr)
- Klausureinsicht: 18.02.2005 (10:00 bis 11:00 Uhr)

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 3 bis 6 (wiederholen)

Aufgabe 55: (K) Multiple Choice: Dateisysteme & I-Nodes (5 Pkt.)

Sind die folgenden Aussagen zu **Dateisystemen** wahr oder falsch?

a. Die Anzahl der I-Nodes eines Dateisystems wird bei der Formatierung festgelegt und entspricht der maximalen Anzahl von Dateien.

b. Der Referenzzähler im I-Node wird erhöht, wenn ein neuer Hardlink auf die betreffende Datei angelegt wird.

c. Wird der I-Node-Referenzzähler auf 0 gesetzt, so ist sichergestellt, dass kein Soft- oder Hardlink mehr auf die betreffende Datei zeigt.

d. Für jeden Softlink muss ein eigener Indirektionsblock angelegt werden, der aus dem I-Node der Quelldatei referenziert wird.

e. Die maximale Größe einer Datei ist unter anderem abhängig von der Größe der Datenblöcke.

Aufgabe 56: (K) Wiederholung: I-Nodes (6 Pkt.)

Die zwei wichtigsten Parameter bei der Formatierung eines Volumes mit einem auf I-Nodes basierenden Dateisystem sind die Anzahl der zu erstellenden I-Nodes und die Größe der Daten- und Indirektionsblöcke. Wie wirken sich die folgenden Annahmen jeweils auf die Wahl der Parameter aus? Welche Probleme könnten sich ergeben?

a. Das Dateisystem soll besonders viele sehr kleine Dateien verwalten.

b. Das Dateisystem soll besonders große Dateien verwalten können.

Aufgabe 57: (T) Vergleich von Dateisystemen (15 Pkt.)

In dieser Aufgabe sollen die Dateisysteme FAT16, FAT32, NTFS, Ext2, NFS und ReiserFS bezüglich folgender Kriterien verglichen und bewertet werden:

- maximale Dateigröße
- maximale Dateinamenlänge
- maximale Partitionsgröße
- Gruppenrechte
- Links-Unterstützung
- Dateiattribute
- Komprimierung
- Verschlüsselung

Aufgabe 58: (T) Journaling-Dateisysteme (15 Pkt.)

Lange Jahre war Ext2 (Second Extended Filesystem) bei Linux-Nutzern das Dateisystem erster Wahl. Es implementiert die klassischen Konzepte traditioneller Unix-Dateisysteme, wie Blöcke und I-Nodes. Ext2 hat allerdings den großen Nachteil, kein **Journaling** zu unterstützen. In heutigen Rechnern sollte es daher nicht mehr eingesetzt werden und stattdessen durch seinen Nachfolger Ext3 oder das ReiserFS ersetzt werden.

a. Beschreiben Sie die grundlegende Architektur des Ext2-Dateisystems.

b. Welcher Hauptnachteil besteht, wenn ein Dateisystem auf Journaling verzichtet?

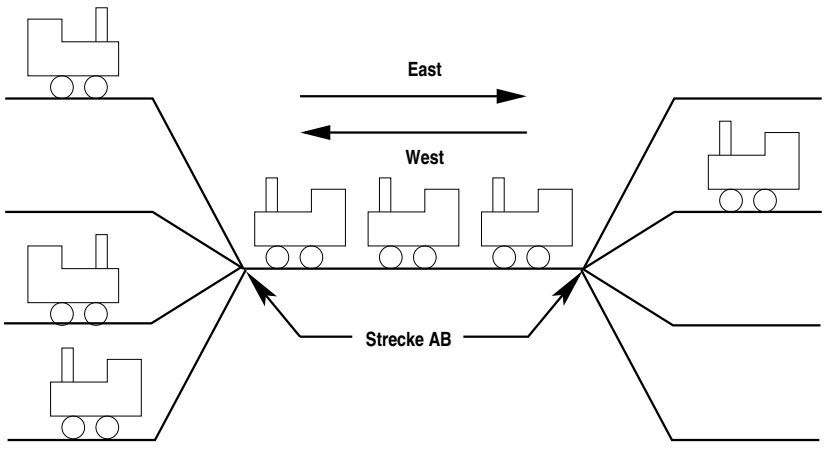
c. Erklären Sie die Arbeitsweise von Journaling-Dateisystemen (am Beispiel von Ext3). Gehen Sie dabei auch auf das Transaktionskonzept und die verschiedenen Arten des Loggings ein.

Aufgabe 59: (K) Java: Koordination von Threads (20 Pkt.)

In dieser Aufgabe sollen Züge koordiniert über einen **eingleisigen** Streckenabschnitt (AB) fahren können. Dieser Streckenabschnitt AB hat folgende Eigenschaften:

- Es kann gleichzeitig nur in genau eine Richtung gefahren werden (entweder West oder Ost).
- Es können sich maximal drei Züge gleichzeitig auf dem Streckenabschnitt befinden.
- Jeder Zug verlässt den Streckenabschnitt nach endlicher Zeit.

Folgende Abbildung zeigt eine mögliche korrekte „Belegung“ der Gleise:



Folgende Klassen sind gegeben:
Die Klasse `Main` erzeugt und startet die Züge (`Train`, Zeilen 11 und 12) sowie den Streckenabschnitt AB (`railAB`, Zeile 8).

```
public class Main
{
    public final static int MAX_THREADS= 10;
    public final static int MAX_TRAINS= 3;

    public static void main(String[] args)
    {
        RailAB ab= new RailAB(MAX_TRAINS);

        for(int i=0; i<MAX_THREADS; i++){
            Train train= new Train(ab, (i%2==1));
            train.start();
        }
    }
}
```

Die Klasse `Train` ist als Thread implementiert und ruft die Methoden `goEast()` (Zeile 14) bzw. `goWest()` (Zeile 18) auf und verlässt mittels `leaveAB()` (Zeile 20) wieder den Streckenabschnitt.

```
public class Train extends Thread
{
    private RailAB  railAB;
    private boolean west;

    public Train(RailAB ab, boolean w){
        railAB= ab;
        west= w;
    }

    public void run() {
        if ( west ) {
            System.out.println("Try to go West!");
            railAB.goWest();
        }
        else {
            System.out.println("Try to go East!");
            railAB.goEast();
        }
        railAB.leaveAB();
    }
}
```

Die Klasse `RailAB` soll die Koordination des Streckenabschnitts AB übernehmen. Die Lösung muss einerseits frei von Deadlocks sein, darf aber andererseits Züge nicht unnötig blockieren.

Implementieren Sie die komplette Klasse `RailAB`, bestehend aus:

a. dem Konstruktor (siehe Klasse `Main` Zeile 8),

b. der Methode `goWest()`, welche die Züge, die nach Westen fahren, koordiniert,

c. der Methode `goEast()`, welche die Züge, die nach Osten fahren, koordiniert, und

d. die Methode `leaveAB()`, welche von den Zügen aufgerufen wurd, die diesen Streckenabschnitt verlassen.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 12
12.01.2005

Informatik III

Abgabetermin: 24.01.2005, 12:00 Uhr s.t.

Achtung: Die Klausur zur Informatik 3 wird am Samstag, 12.02.2005, von 9:00 Uhr s.t. bis 12:00 Uhr stattfinden. Dazu ist für diejenigen Studierenden, die zur Klausur zugelassen sind, eine obligatorische Anmeldung zwischen dem 31.01.2005 (10:00 Uhr) und dem 04.02.2005 (12:00 Uhr) nötig. Die Veröffentlichung der Teilnehmerlisten erfolgt am 07.02.2005 (17:00 Uhr). Eine Nachmeldemöglichkeit besteht nur in Ausnahmefällen am 08.02.2005 zwischen 10:00 und 17:00 Uhr.

Zur Klausuranmeldung wird eine WWW-Seite freigeschaltet, die über die Info 3-Homepage zugänglich gemacht wird. Ist es einem Studierenden nicht möglich, diese WWW-Seite zu nutzen, so kann er sich auch persönlich im Anmeldezeitraum bei den Betreuern anmelden (bitte die Sprechstunden beachten). Anmeldungen nach dem letzten Anmeldetermin werden nicht akzeptiert. Nicht angemeldete Studenten können nicht an der Klausur teilnehmen.

Die endgültige Teilnehmerliste wird am Mittwoch, 09.02.2005 ab 17:00 Uhr sowohl im Internet als auch in den Schaukästen des Lehrstuhls veröffentlicht. Jeder Teilnehmer erhält eine verbindliche Platznummer in einem der reservierten Hörsäle. Beachten Sie, dass zur Klausur keinerlei Hilfsmittel erlaubt sind. Vergessen Sie auch nicht, Ihren Studentenausweis und Personalausweis mitzubringen.

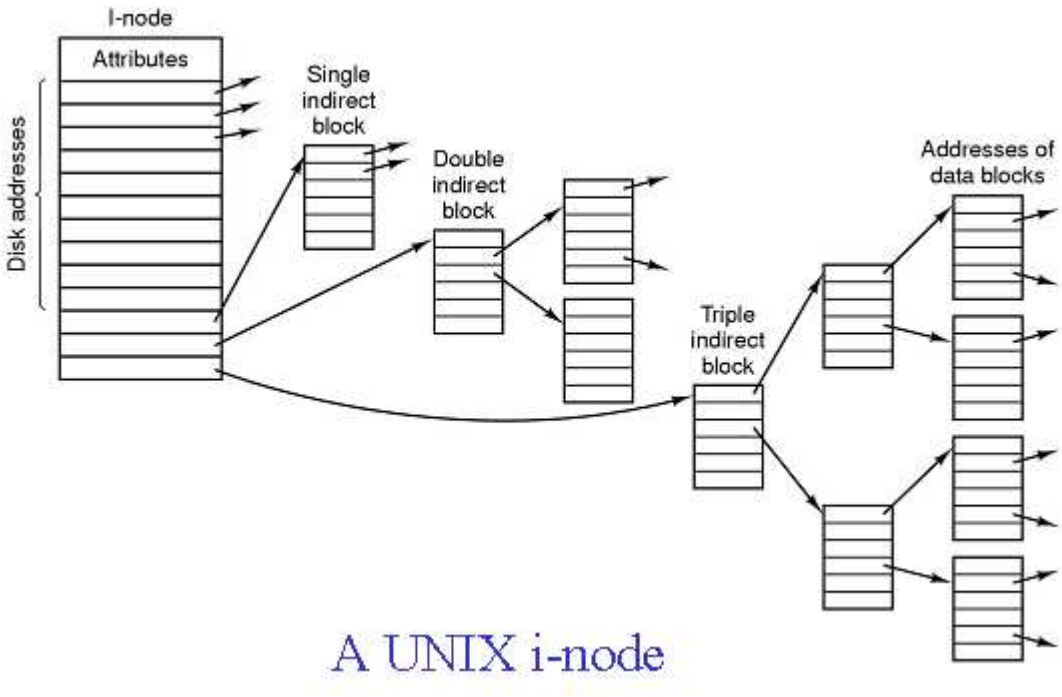
Die Klausureinsicht findet am Freitag, 18.02.2005 von 10:00 bis 11:00 Uhr im Hörsaal 1.15 in der Oettingenstraße 67 statt.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 6.1 bis 6.4

Aufgabe 52: (H) Dateiverwaltung mittels Inodes (14 Pkt.)

Die Verwaltung von Dateien und Verzeichnissen wird bei Unix mittels sogenannter I-Nodes vorgenommen. Dabei findet eine strikte Trennung der eigentlichen Datei von deren Verwaltungsinformationen in den I-Nodes statt. Die Anzahl der I-Nodes wird bei der Formatierung des Dateisystems festgelegt. Jeder I-Node besitzt eine eindeutige Nummer und hat eine feste Länge. Jede Datei wird durch genau einen I-Node repräsentiert, der verschiedene Attribute enthält: Eigentümer, Gruppe, Zugriffsrechte, Dateityp, Dateigröße, usw.

Die nächsten zehn Einträge sind direkte Adressen der Datenblöcke der Datei. Dann gibt es je einen Verweis auf den ersten Indirektionsblock, der wiederum nur aus Adressen von Datenblöcken besteht. Das Gleiche gilt für den zweiten und dritten Indirektionsblock mit jeweils einer Stufe mehr. Die Pfeile, welche ins Leere zeigen, zeigen auf einen konkreten Datenblock:



Ein Verzeichnis liegt ebenfalls als Datei vor (besitzt also einen I-Node), aber dessen Datenblöcke sind Tabellen, in deren Zeilen die I-Node-Nummern und zugehörigen Dateinamen der im Verzeichnis abgelegten Dateien gespeichert sind.

a. Es soll auf die Datei `/baum/zweig/ast` zugriffen werden. Wie viele I-Nodes und Datenblöcke müssen dazu mindestens gelesen werden?

b. Warum ist es sinnvoll, den Dateityp im I-Node festzuhalten?

c. Nennen Sie einen Vor- und einen Nachteil, dass Inodes feste Länge besitzen.

d. Ein Datenblock (sowie ein Indirektionsblock) sei 4 KByte groß. Wie groß ist die maximale Dateigröße, wenn 4 Bytes zur Adressierung eines Datenblocks benötigt werden?

e. Diskutieren Sie das Zugriffsverhalten für sehr kleine und sehr große Dateien.

Mit Hilfe des `ln`-Befehls lassen sich unter Unix für die selbe Datei verschiedene Namen vergeben. Dabei wird zwischen einem Hardlink (`ln quellDatei zielDatei`) und einem symbolischen Link (Softlink) (`ln -s quellDatei zielDatei`) unterschieden.

f. Machen Sie sich den Unterschied zwischen den beiden Varianten klar, indem Sie eine Datei anlegen. Erzeugen Sie einen symbolischen und einen Hardlink auf diese Datei. Mit Hilfe des Befehls `ls -li` wird in der ersten Spalte die I-Node-Nummer ausgegeben. Löschen Sie mit `rm` jeweils die Orginaldatei bzw. die Links und diskutieren Sie anhand der Unterschiede die beiden Verfahren.

g. Jeder I-Node enthält einen Linkzähler (Referenzzähler) als Eintrag (die 3. Spalte bei der Ausgabe von`ls -li`). Wofür wird dieser benötigt?

h. Was geschieht genau bei Anwendung des Umbennungsbefehls `mv`?

i. Legen Sie mit dem Befehl `mkdir` ein neues Verzeichnis an. Dabei werden in dem Verzeichnis automatisch die Verzeichnisse '.' und '..' angelegt. Entscheiden Sie, ob dabei jeweils ein symbolischer oder Hardlink angelegt wird.

Aufgabe 53: (T) Power Management: APM & ACPI (25 Pkt.)

In den frühen 90er-Jahren haben Intel und Microsoft den Standard APM (Advanced Power Management) entwickelt, in dem verschiedene Aktivitätsmodi für einen PC spezifiziert wurden. Das neuere ACPI (Advanced Configuration and Power Interface) wurde erstmals 1996 in der Version 1.0 veröffentlicht und hat APM bis heute praktisch verdrängt.

a. Nennen Sie zunächst einige Möglichkeiten zur Energieeinsparung bei Einzelkomponenten.

b. Welche Aktivitätsmodi werden in APM spezifiziert?

c. Das ATX-Format (Advanced Technology Extended) ist eine Norm für Gehäuse und Hauptplatinen (Mainboards) und unterscheidet im Gegensatz zum Vorgänger (AT) zwischen den Ausschaltzuständen Soft Off und Mechanical Off. Was ist der Unterschied?

d. Erläutern Sie die Möglichkeiten der Energieeinsparung unter Verwendung von ACPI, indem Sie auf die verschiedenen Zustandsmodelle eingehen.

e. Welcher Unterschied besteht zwischen APM und ACPI hinsichtlich der Steuerung?

Aufgabe 54: (T) Java Certification Exam (23 Pkt.)

Die folgenden Aufgaben sind Auszüge aus den Übungen zur Vorbereitung auf die Java Certification-Prüfung:

a. Was wird passieren, wenn Sie versuchen den folgenden Code zu kompilieren und auszuführen?

```
abstract class Base {
  abstract public void myfunc();
  public void another() {
    System.out.println("Another method");
  }
}

public class Abs extends Base {
  public static void main(String argv[]) {
    Abs a = new Abs();
    a.amethod();
  }
  public void myfunc() {
    System.out.println("My Func");
  }
  public void amethod() {
    myfunc();
  }
}
```

   (i) Der Code wird kompiliert und ausgeführt, wobei „My Func“ ausgegeben wird.

   (ii) Der Compiler bemängelt, dass die `Base`-Klasse keine abstrakte Methode enthält.

   (iii) Der Code wird kompiliert, aber es tritt ein Laufzeitfehler auf: Die `Base`-Klasse hat keine abstrakten Methoden.

   (iv) Der Compiler bemängelt, dass die Methode `myfunc()` in der `Base`-Klasse keinen Funktionskörper besitzt.

b. Sie wollen den Wert des letzten Elements eines Arrays mit dem folgenden Code herausfinden. Was passiert, wenn Sie ihn kompilieren und ausführen?

```
public class MyAr {
  public static void main(String argv[]) {
    int[] i = new int[5];
    System.out.println(i[5]);
  }
}
```

c. Was wird passieren, wenn Sie versuchen den folgenden Code zu kompilieren und auszuführen?

```
public class Bground extends Thread {
  public static void main(String argv[]) {
    Bground b = new Bground();
    b.run();
  }
  public void start(){
    for (int i = 0; i <10; i++){
      System.out.println("Value of i = " + i);
    }
  }
}
```

d. Was kann einen Thread dazu bringen, seine Ausführung zu unterbrechen?

(i) Das Programm beendet sich mit einem Aufruf von `System.exit(0);`.

(ii) Einem anderen Thread wird höhere Priorität gegeben.

(iii) Ein Aufruf der `stop`-Methode des Threads.

(iv) Ein Aufruf der `halt`-Methode des Threads.

e. Welche der folgenden Aussagen über Threads sind wahr?

(i) Man kann einen wechselseitigen, exklusiven Lock für Methoden in einer Klasse erhalten, die die `Thread`-Klasse erweitert oder das Interface `runnable` implementiert.

(ii) Man kann einen wechselseitigen, exklusiven Lock für jedes Objekt erhalten.

(iii) Ein Thread kann einen wechselseitigen, exklusiven Lock für eine `synchronized` Methode eines Objekts erhalten.

f. Welche der folgenden Aussagen beschreibt am besten die Funktionsweise des `synchronized`-Schlüsselwortes?

(i) Erlaubt zwei Prozessen parallel ausgeführt zu werden, aber miteinander zu kommunizieren.

(ii) Stellt sicher, dass nur ein Thread zur selben Zeit auf eine Methode oder ein Objekt zugreifen kann.

(iii) Stellt sicher, dass zwei oder mehr Prozesse zur selben Zeit starten und enden werden.

(iv) Stellt sicher, dass zwei oder mehr Threads zur selben Zeit starten und enden werden.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 11
22.12.2004

Informatik III

Abgabetermin: 17.01.2005, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 1 bis 4 (wiederholen)

Aufgabe 47: (H) Multiple Choice: Seitenersetzung (5 Pkt.)

Sind die folgenden Aussagen wahr oder falsch?

a. Solange die Anzahl N aller existierenden Seiten die Zahl der verfügbaren Seitenrahmen nicht übersteigt, können außer beim erstmaligen Einlagern einer Seite in den Arbeitsspeicher keine Seitenfehler auftreten.

b. Bei sehr starker (zeitlicher) Lokalität der Speicherzugriffe ist die LRU-Strategie zur Seitenersetzung problematisch im Hinblick auf die Seitenfehleranzahl, da sie besonders häufig referenzierte Seiten mit der Zeit „abwertet“.

c. Bei der LFU-Strategie zur Seitenersetzung wird für jede Seite im Arbeitsspeicher ein Zähler für die Häufigkeit ihrer Referenzierung mitgeführt. Im ungünstigsten Fall kann dies dazu führen, dass Seiten mit zunächst sehr hoher Zugriffszahl auch dann noch besonders lange den Arbeitsspeicher blockieren, wenn sie schon längst nicht mehr benötigt werden.

d. Alle Seitenrahmen des Arbeitsspeichers seien mit Seiten gefüllt. Nach der LIFO-Strategie wird bei jedem weiteren Seitenfehler immer nur im selben Seitenrahmen operiert.

e. Es gibt Fälle, in denen mit der LIFO-Strategie weniger Seitenfehler als mit der FIFO-Strategie erzielt werden.

Aufgabe 48: (H) Modellierung von Zählsemaphoren (10 Pkt.)

Gegeben ist das folgende korrekt synchronisierte Erzeuger-Verbraucher-Problem aus der Vorlesung (siehe Skript, Seite 80):

```
Erzeuger:
REPEAT
  <erzeuge Element>;
  wait(p);
  wait(s);
  <lege Element im Speicher ab>;
  signal(s);
  signal(b);
UNTIL FALSE;

```

```
Verbraucher:
REPEAT
  wait(b);
  wait(s);
  <entnimm Element aus Speicher>;
  signal(s);
  signal(p);
  <verbrauche Element>;
UNTIL FALSE;
```

Dabei ist s ein Binärsemaphor, p (Platz) und b (Bestand) sind Zählsemaphore.

a. Nehmen Sie an, dass der Zugriff auf ein Lager mit genau drei Plätzen synchronisiert wird. Wie müssen die Semaphore in diesem Fall initialisiert werden, wenn das Lager zu Beginn leer ist? Verwenden Sie die Darstellung `init(Semaphor,Wert)`.

b. Man kann zeigen, dass Zählsemaphore und Binärsemaphore die selbe Ausdruckskraft besitzen. Schreiben Sie das obige Programm so um, dass nur noch Binärsemaphore verwendet werden. Führen Sie dazu die allgemeinen Semaphore auf Binärsemaphore zurück und implementieren Sie die Semaphor-Operationen in einer von Ihnen gewählten Programmiersprache bzw. Pseudo-Code.

Aufgabe 49: (H) Engpassprobleme heutiger Rechner (10 Pkt.)

Gegeben ist folgende Aufgabe: Eine Applikation muss ein Datenvolumen von 100 MB verarbeiten. Die Daten liegen auf einer Platte, und zwar ausschließlich in direkt aufeinanderfolgenden Sektoren. Der Anwendung wurden vom Betriebssystem 20 MB Hauptspeicher zugeteilt, wovon die Hälfte für den Ausführungs-Stack und das Ablegen von Zwischenergebnissen benötigt wird. Die andere Hälfte steht als Puffer zur Verfügung.

Nehmen Sie für dieses Beispiel an, dass die Bearbeitung von 4 Bytes Daten durch die CPU im Schnitt 1 µs dauert. Die mittlere Suchzeit beträgt 4 ms, die Plattenumlaufdauer 10 ms, und die Transferrate liegt bei 64 MB/s.

a. Wie groß ist die mittlere Latenzzeit (allgemein und im Beispiel)?

   Zur Erinnerung: Unter der Latenzzeit versteht man die Zeitdauer, bis die Platte so rotiert ist, dass der Anfang der einzulesenen Seite unter dem Arm liegt.

b. Wie viel (reine) CPU-Zeit benötigt die Berechnung?

c. Wie viel Zeit wird für I/O-Operationen benötigt? Geben Sie hierfür auch eine Formel in Abhängigkeit von Suchzeit, Latenzzeit, Transferrate, Dateigröße und Puffergröße an.

Aufgabe 50: (T) Power Management (10 Pkt.)

Das Power Management (PM) übernimmt in modernen Rechnern weitreichende Aufgaben, die von reinen Energiesparfunktionen bis hin zur Steuerung betriebskritischer Einheiten (wie z.B. Lüfter) reichen. Für diese Steuer- und Regelaufgaben ist teils komplexe Software notwendig. Ausgehend von einem modularen Rechner, der aus Komponenten verschiedener Hersteller aufgebaut wird, sollen hier einige Implementierungskonzepte für PM-Software verglichen werden. Die Basiskonzepte:

– Jedes Hardware-Modul bringt eigene Software (Firmware) mit, die von einer eigenen Verarbeitungseinheit ausgeführt wird (z.B. Steckkarte mit Firmware und eigener eingebetteter CPU).

- Jedes Hardware-Modul bringt eigene Software (Firmware) mit, die von der Haupt-CPU des Rechners ausgeführt wird.
- Die PM-Funktionen werden nur durch die BS-Treiber des Hardware-Moduls implementiert.
- Die PM-Funktionen werden monolithisch im Betriebssystem selbst implementiert.

Bewerten und vergleichen Sie diese vier Implementierungsansätze hinsichtlich der folgenden Kriterien:

- Hardwarekosten
- Anpassungsaufwand bei wechselnder CPU-Plattform
- Anpassungsaufwand bei wechselnder Betriebssystemplattform (z.B. von Linux nach Windows)
- Zuverlässigkeit und Betriebsstabilität (z.B.: Funktioniert die Lüftersteuerung noch, wenn das Betriebssystem abstürzt?)
- Realisierbarkeit kombinierter PM-Funktionen zwischen mehreren Hardware-Modulen
- Update-Aufwand

Aufgabe 51: (P) Java: Ausgabe der Verzeichnisstruktur (10 Pkt.)

Schreiben Sie ein Java-Programm, das die Verzeichnisstruktur eines Dateisystems (Dateien und Verzeichnisse) ausgibt.

a. Implementieren Sie Ihr Programm so, dass es beim Starten ein Argument erwartet, in dem das Verzeichnis angegeben ist, ab dem die Ausgabe beginnen soll. Es sollen alle Unterverzeichnisse mit den darin enthaltenen Dateien ausgegeben werden. Nutzen Sie Methoden, die von der Klasse `java.io.File` zur Verfügung gestellt werden.

b. Muss bei der Programmierung berücksichtigt werden, welches Dateisystem verwendet wird?

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 10
15.12.2004

Informatik III

Abgabetermin: 10.01.2005, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 1 bis 4 (wiederholen)

Aufgabe 43: (H) Wiederholung (14 Pkt.)

Beantworten Sie die folgenden Fragen bitte in kurzen Stichpunkten.

a. Prozesskoordination:

   (i) Welches sind die gültigen Operationen auf Binärsemaphoren?

   (ii) Was muss zusätzlich bei Verwendung von Zählsemaphoren beachtet werden?

   (iii) Welches Ziel wird durch den Einsatz programmiersprachlicher Monitore verfolgt?

   (iv) Unterstützen Java-Monitore die Verwendung von Bedingungsvariablen? Falls nicht: Wie kann trotzdem verhindert werden, dass der Rumpf einer als `synchronized` deklarierten Methode zur Ausführung kommt, wenn bestimmte Voraussetzungen (noch) nicht erfüllt sind?

b. Speicherverwaltung:

   (i) Nennen Sie die Ihnen bekannten Konzepte zum Speicher-Management. Welche Arten von Fragmentierung können jeweils eintreten?

   (ii) Welche Segmentierungsstrategie sollte gewählt werden, wenn verhindert werden soll, dass ein Ende des Speicherbereichs mit der Zeit stark fragmentiert wird?

   (iii) Welche Segmentierungsstrategie ist mit hohem Suchaufwand verbunden?

   (iv) Welcher Seitenersetzungsalgorithmus ist bei hoher zeitlicher und räumlicher Lokalität der Speicherzugriffe besonders geeignet?

c. Deadlocks:

   (i) Erklären Sie den Zusammenhang zwischen Deadlocks und Multiprogramming.

   (ii) Lässt sich das Modell des Prozessfortschrittsdiagramms auch auf Multiprozessorsysteme anwenden? Was ändert sich?

   (iii) Nennen Sie ein Verfahren zur Deadlock-Vermeidung und ein Verfahren zur Deadlock-Behebung.

d. Prozesse, Threads und Scheduling:

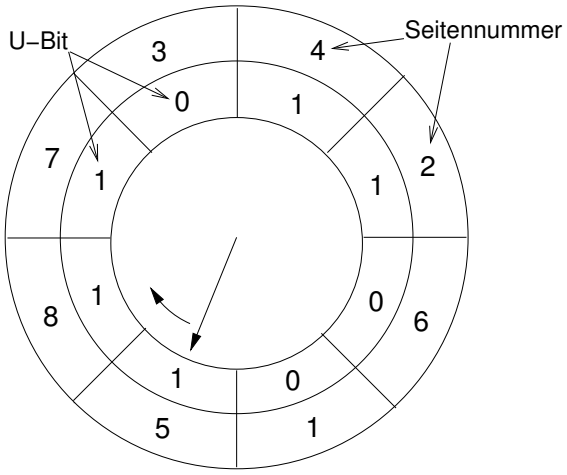
   (i) In welchem Fall ist ein Kontextwechsel zwischen Threads ohne vorherigen Moduswechsel möglich?

(ii) Eine preemptive Variante der Scheduling-Strategie SJF (auch manchmal SJN genannt) ist SRPT. In welchem Fall ist SRPT eine sinnvolle Alternative zu SJF?

(iii) Welche Informationen werden im PCB gespeichert?

Aufgabe 44: (H) Seitenersetzung: Second Chance (16 Pkt.)

Der Second-Chance-Algorithmus (eine Variante des Clock-Algorithmus) verwendet für die Auswahl der zu verdrängenden Seiten eine zyklische Datenstruktur wie die hier skizzierte:



Der einzige Unterschied zum Clock-Algorithmus besteht darin, dass der Zeiger immer auf die **zuletzt eingelagerte** Seite verweist. Bei einem Zugriff auf eine Seite wird das dazugehörige U-Bit (Use-Bit) von der Hardware auf 1 gesetzt.

a. Überlegen Sie sich, wie eine sinnvolle, die Seitenfehlerzahl unter Ausnutzung des Lokalitätsprinzips minimierende Paging-Strategie unter den gegebenen Rahmenbedingungen (zyklische Datenstruktur, genau ein Zeiger, U-Bits für jede Seite) aussehen müsste. Geben Sie also in natürlicher Sprache die Arbeitsweise des Second-Chance-Algorithmus an.

b. Eine Seite mit der Nummer 10 soll in den Hauptspeicher geladen werden. Welche Seite wird dafür aus dem Hauptspeicher verdrängt?

c. Skizzieren Sie die obige Datenstruktur nach dem Einlagern der neuen Seite.

d. Was passiert, wenn die U-Bits aller Seiten auf 1 gesetzt sind und ein Zugriff auf eine nicht im Hauptspeicher befindliche Seite erfolgt?

e. Der **Enhanced-Second-Chance-Algorithmus** verwendet zusätzlich zum Use-Bit (U-Bit) noch ein Modified-Bit (M-Bit), das angibt, ob eine im Hauptspeicher geladene Seite verändert wurde oder nicht. Jede im Hauptspeicher enthaltene Seite fällt damit in eine der folgenden Klassen:

- $U = 0, M = 0$
- $U = 1, M = 1$
- $U = 0, M = 1$
- $U = 1, M = 0$

(i) In welcher Reihenfolge sollten Seiten dieser unterschiedlichen Klassen für eine Verdrängung aus dem Hauptspeicher ausgewählt werden? Begründen Sie kurz Ihre Entscheidung.

(ii) Welche Verbesserung erhofft man sich durch diese Strategie?

f. Wie könnte der (einfache) Second-Chance-Algorithmus verbessert werden, sodass er Least Recently Used (LRU) besser approximiert?

Aufgabe 45: (T) Prozessfortschrittsdiagramm (15 Pkt.)

Hier wird das Konzept des Prozessfortschrittsdiagramm zur Modellierung und Erkennung von Deadlocks nochmals untersucht.

a. Gegeben seien zwei Prozesse A und B, A benötigt zu seiner Ausführung zehn Zeiteinheiten, B neun Zeiteinheiten. Ferner stehen sechs verschiedene Betriebsmittel (BM) zur Verfügung, die von den Prozessen während ihrer Ausführung benötigt werden. Die folgende Tabelle zeigt, in welchen Intervallen die beiden Prozesse die Betriebsmittel belegen:

Prozess	BM1	BM2	BM3	BM4	BM5	BM6
A	$1-2, 8-9$	$3-4, 5-8$	$3-5$	$4-6$	$2-4$	$7-8$
B	$1-3, 5-8$	$6-7$	$3-5$	$2-3$	$7-8$	$4-6$

Erstellen Sie das zugehörige Prozessfortschrittsdiagramm, und zeichen Sie alle *prinzipiell* verschiedenen Möglichkeiten ein, die Prozesse A und B terminieren lassen.

b. Zwei Prozesse A und B benötigen zu ihrer Ausführung je eine gewisse Zeitdauer t_A bzw. t_B und die drei Betriebsmittel X, Y und Z. t_{ij} bezeichne die Zeitdauer, die das Betriebsmittel i von Prozess j während der Ausführungszeit benötigt wird. Ein Experte trifft nun folgende Behauptungen:

- Ein Deadlock tritt auf keinen Fall auf, wenn
$$t_{XA} + t_{YA} + t_{ZA} < t_A \quad \text{und} \quad t_{XB} + t_{YB} + t_{ZB} < t_B.$$
- Ein Deadlock kann nicht auftreten, wenn
$$t_{XA} \cdot t_{XB} + t_{YA} \cdot t_{YB} + t_{ZA} \cdot t_{ZB} < t_A \cdot t_B.$$
- Ein Deadlock liegt automatisch vor, wenn
$$t_{XA} + t_{YA} + t_{ZA} > t_A \quad \text{und} \quad t_{XB} + t_{YB} + t_{ZB} > t_B.$$
- Ein Deadlock liegt automatisch vor, wenn
$$t_{XA} \cdot t_{XB} + t_{YA} \cdot t_{YB} + t_{ZA} \cdot t_{ZB} > t_A \cdot t_B.$$

Bewerten Sie diese Aussagen, indem Sie entweder einen Beweis oder ein Gegenbeispiel (in Form eines Prozessfortschrittsdiagramms) angeben.

Aufgabe 46: (T) Synchrone & asynchrone Kommunikation (10 Pkt.)

a. Zeigen Sie die Richtigkeit der folgenden zwei Behauptungen, und geben Sie jeweils ein Beispiel an.

(i) Synchrone, bidirektionale Nachrichten-Kommunikation kann durch asynchrone Kommunikation nachgebildet werden.

(ii) Asynchrone Nachrichten-Kommunikation kann durch synchrone Kommunikation nachgebildet werden.

b. Welches sind die Vor- und Nachteile synchroner und asynchroner Kommuniukation?

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 9
8.12.2004

Informatik III

Abgabetermin: 20.12.2004, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 4.1 bis 4.5

Aufgabe 39: (H) Seitenersetzung: Keller-Algorithmen (8 Pkt.)

a. Was versteht man unter einer Distanzkette? Welche Informationen enthält sie? Warum kann es vorteilhafter sein, die Distanzkette statt der Referenzkette zu betrachten, um ein Paging-System zu bewerten?

b. Beweisen Sie, dass der FIFO-Algorithmus zur Seitenersetzung kein Keller-Algorithmus ist.

Aufgabe 40: (H) Seitenersetzungs-Strategien (12 Pkt.)

a. Die Menge der Seiten sei gegeben durch $N = \{0, 1, 2, 3, 4\}$ und die Menge der Seitenrahmen, die für die Speicherung der Seiten im Arbeitsspeicher zur Verfügung steht, sei gegeben durch $Frame_3 = \{f_0, f_1, f_2\}$. Auf die fünf Seiten der Menge N werde in folgender Reihenfolge zugegriffen:

```
w = 4 1 2 3 1 4 3 1 2 3 4 0 3 2 0 4
```

Ein Seitenfehler liegt immer dann vor, wenn sich eine referenzierte Seite nicht im Arbeitsspeicher befindet. Dieser ist zu Beginn leer. Ermitteln Sie die Anzahl der Seitenfehler für die folgenden Paging-Strategien, indem Sie alle Veränderungen im Speicher tabellarisch dokumentieren.

(i) FIFO (First In, First Out)

(ii) LIFO (Last In, First Out)

(iii) LRU (Least Recently Used)

(iv) LFU (Least Frequently Used)

b. Die Menge der Seitenrahmen werde vergrößert zu $Frame_4 = \{f_0, f_1, f_2, f_3\}$. Arbeiten Sie analog Aufgabenteil a) und vergleichen Sie die Seitenfehlerzahlen.

Aufgabe 41: (K) Multiple Choice: Paging (5 Pkt.)

Sind die folgenden Aussagen wahr oder falsch?

a. Frames (Rahmen) des physischen Adressraums sollten die gleiche Größe besitzen wie die Seiten im logischen Adressraum.

b. Unter Demand Paging versteht man das Nachladen von Seiten aus dem Hintergrundspeicher in den Hauptspeicher bei Vorliegen eines Seitenfehlers.

c. Unter einem Seitenfehler versteht man das Vorliegen einer beschädigten Datenseite, zum Beispiel durch Fehler in der Bitübertragung.

d. Demand-Prepaging und Look-Ahead-Paging bezeichnen Paging-Strategien, bei denen gleichzeitig mehrere Seiten nachgeladen und verdrängt werden, um die Zahl der Zugriffe auf den Hintergrundspeicher gering zu halten.

e. Die Optimalstrategie zur Seitenersetzung ist praktisch nicht anwendbar, da zukünftige Seitenanforderungen nicht vorhersehbar sind.

Aufgabe 42: (T) Virtueller Speicher (15 Pkt.)

a. Was ist die Overlay-Technik? Warum wurde sie durch das Konzept des virtuellen Speichers ersetzt?

b. Gegeben sei folgendes Paging-System:

- Größe des physischen Speichers: 32 KB
- Größe des virtuellen Speichers: 64 KB
- Größe von Seiten und Seitenrahmen: 4 KB
- Wortgröße: 1 Byte

(i) Skizzieren Sie den virtuellen und den physischen Adressraum.

(ii) Aus wie vielen Bits bestehen die virtuellen und die physischen Adressen, und wie sind diese zusammengesetzt? Was ist ein Offset?

(iii) Stellen Sie die Übersetzung einer virtuellen Adresse in eine physische Adresse innerhalb der MMU (Memory Management Unit) graphisch dar. Übersetzen Sie die virtuelle Adresse 8196_{10}.

c. Aus wie vielen Seiten besteht der virtuelle Adressraum bei Verwendung von 32-Bit-Adressen (Seitengröße: 4 KB, byteweise Adressierung)? Ermitteln Sie die obere Schranke für die Größe des physischen Hauptspeichers. Kann es jemals sinnvoll sein, dass der Hauptspeicher größer ist als der virtuelle Speicher?

d. Bei großen Adressräumen werden auch die Seitentabellen sehr umfangreich. Zu lange Zugriffszeiten können so schnell zum Engpass werden. Außerdem geht zu viel Speicherplatz allein für die Seitentabellen verloren. Wie lässt sich dieses Problem lösen?

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 8
1.12.2004

Informatik III

Abgabetermin: 13.12.2004, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 4.1 bis 4.5

Aufgabe 34: (H) Java: Erzeuger/Verbraucher-Problem (6 Pkt.)

Gegeben ist eine Lösung des Erzeuger/Verbraucher-Problems in Java (siehe unten). Untersuchen Sie, ob durch folgende Änderungen immer noch eine korrekte Lösung vorliegt. Begründen Sie Ihre Antwort.

a. Wenn man die Zeilen 11 und 12 vertauscht, muss man keine Methoden mehr als `synchronized` deklarieren.

b. Wenn man die beiden `run()`-Methoden (Zeilen 22 und 37) als `synchronized` deklariert, kann man das `synchronized` bei `add()` (Zeile 61) und `subtract()` (Zeile 51) weglassen.

c. Es würde auch genügen, entweder nur `subtract()` (Zeile 48) oder nur `add()` (Zeile 58) als `synchronized` zu deklarieren.

d. Man kann das Inkrementieren bzw. Dekrementieren von `count` an den jeweiligen Anfang der Methoden setzen.

e. Man kann die `while`-Schleifen (Zeilen 52 und 62) durch `if`-Anweisungen ersetzen.

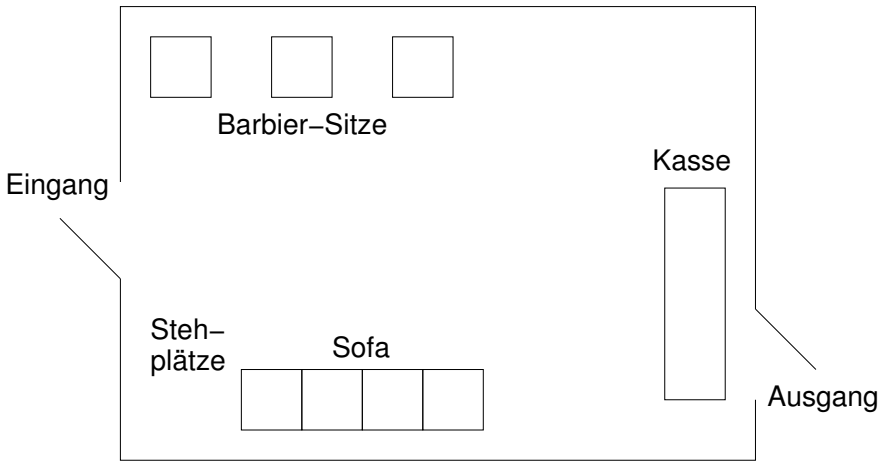
f. Man kann Zeile 57 mit 59 und 67 mit 69 vertauschen.

```
public class ProdCons {
2
    public static void main(String args[]) {
4       Counter counter = new Counter(5);
        Consumer consumer[]= new Consumer[10];
6       Producer producer[]= new Producer[10];

8       for ( i= 0; i < 10; i++ ){
                consumer[i] = new Consumer(counter);
10              producer[i]= new Producer(counter);
                consumer[i].start();
12              producer[i].start();
        }
14  }
}
16
```

```
class Consumer extends Thread {
    private Counter counter;

    public Consumer(Counter c) { counter = c;}

    public void run() {
        for(int i = 0; i < 20; ++i) {
            counter.subtract();
            // beliebiger anderer Code
            try {
                sleep(1);
            } catch(InterruptedException ie){};}
    }
}

class Producer extends Thread {
    private Counter counter;

    public Producer(Counter c) { counter = c;}

    public void run() {
        for(int i = 0; i < 20; ++i) {
            counter.add();
            // beliebiger anderer Code
        }
    }
}

class Counter {
    private int maxCount;
    private int count = 0;

    public Counter(int mct) {maxCount = mct;}

    public synchronized void subtract() {
        while (count < 1) {
            try {
                wait();
            } catch(InterruptedException ie){}
        }
        --count;
        System.out.println("- count is " + count);
        notifyAll();}

    public synchronized void add() {
        while (count >= maxCount) {
            try {
                wait();
            } catch(InterruptedException ie){}
        }
        ++count;
        System.out.println("+ count is " + count);
        notifyAll();}
}
```

Aufgabe 35: (H) Semaphoren: Schlafender Barbier (14 Pkt.)

Ein bekanntes Beispiel für die Nutzung von Semaphoren zur Synchronisation ist das **Problem des schlafenden Barbiers**. Unser Barbierladen habe drei Sitze, drei Barbiere und einen Warteraum, in dem Platz für vier Kunden auf einem Sofa und Raum zum Stehen vorhanden ist. Feuerschutzvorschriften beschränken die Gesamtanzahl von Kunden im Laden auf 20.



Verhaltensregeln:

- Ein Kunde darf den Laden nicht betreten, wenn dieser bereits mit der maximalen Zahl an Kunden gefüllt ist.
- Hat der Kunde einmal den Laden betreten, setzt er sich entweder auf das Sofa oder bleibt (falls das Sofa voll ist) im Warteraum stehen.
- Wenn ein Barbier frei ist, wird derjenige Kunde bedient, der am längsten *auf dem Sofa* sitzt.
- Wenn es stehende Kunden gibt, wird der frei gewordene Platz auf dem Sofa von dem am längsten stehenden Kunden eingenommen.
- Wenn ein Kunde bedient wurde, kann bei jedem Barbier die Rechnung bezahlt werden. Aber da es nur eine Kasse gibt, kann nur ein Kunde zu einer Zeit bezahlen.
- Die Barbiere verbringen ihre Zeit mit Haareschneiden, Kassieren und Schlafen (in ihrem Stuhl) – wartend auf einen Kunden.

Finden Sie eine Lösung für dieses Problem. Die Synchronisation soll über Zählsemaphoren (ohne Warteschlangen) erfolgen. Synchronisieren Sie alle möglichen Abläufe! Geben Sie eine Pseudocode-Implementierung für Kunden, Barbiere bzw. Kassierer an. Als Hilfestellung hier ein kleiner Ausschnitt aus einer möglichen Lösung:

Initialisierung:

```
init(max_capacity, 20);
init(sofa, 4);
init(barber_chair, 3); // Kunde wartet auf freien Stuhl
init(coord, 3);        // Warten auf Barbier (zum Schneiden oder Kassieren)
init(cust_ready, 0);   // Barbier wartet, bis Kunde auf dem Stuhl sitzt
...
```

Kunde:

```
PROCEDURE customer;

BEGIN
   wait(max_capacity);
   (enter shop);
   wait(sofa);
   (sit on sofa);
   ...
END;
```

Barbier:

```
PROCEDURE barber;

BEGIN
   REPEAT
      wait(cust_ready);
      ...
   UNTIL FALSE;
END;
```

Kassierer:

```
PROCEDURE cashier;

BEGIN
...
END;
```

Aufgabe 36: (T) Speicherverwaltung: Überblick (10 Pkt.)

Vergleichen Sie die folgenden grundlegenden Konzepte zur Speicherverwaltung. Geben Sie für jedes System an, wie Partitionierung, Freigabe und Belegung von Speicherbereichen umgesetzt werden, welche unterschiedlichen Strategien jeweils innerhalb eines Konzeptes zum Tragen kommen und welche Art der Fragmentierung auftreten kann.

a. Statische Speicherpartitionierung

b. Dynamische Speicherpartitionierung

c. Buddy-Systeme

Aufgabe 37: (K) Buddy-Systeme (10 Pkt.)

Ein mobiles Gerät verfüge über einen 1 MB großen Speicher, der nach dem Buddy-Verfahren verwaltet wird. Um diesen Speicher byteweise zu adressieren, benötigt man 20 Bits (2^{20} = 1.048.576 Bytes = 1 MB).

a. Nacheinander sollen die folgenden vier Programme in den Speicher geladen werden:

- P_1: 100 KB
- P_2: 220 KB
- P_3: 250 KB
- P_4: 60 KB

Zeichnen Sie den Buddy-Baum nach jeder Neubelegung. Tragen Sie auch die Zeiger auf die Freibereiche ein, und geben Sie für P_1 bis P_4 die Speicheradressen an.

Hinweis: Es wird immer das am weitesten links stehende Segment gesplittet und der am weitesten links stehende Buddy belegt.

b. Die Programme aus Teilaufgabe a benötigen insgesamt 630 KB Speicherplatz. Damit müssten noch $1024 - 630 = 394$ KB nutzbar sein. Warum ist das im Beispiel nicht der Fall? Welcher Effekt kommt hier zum Tragen? Wie viel nutzbarer Speicherplatz steht dem Benutzer noch zur Verfügung?

c. Gegeben ist eine weitere Anfrage:

   – P_5: 280 KB

   Kann P_5 noch zusätzlich in den Speicher geladen werden? Falls ja, zeichnen Sie den Buddy-Baum nach der Belege-Operation. Falls nein, begründen Sie Ihre Antwort.

d. Beurteilen Sie aufgrund der Ergebnisse aus den vorigen Teilaufgaben die Anwendung des Buddy-Verfahrens im Hinblick auf die Speicherausnutzung. Worin liegt der wesentliche Vorteil im Vergleich zu einem System mit fester Speicherpartitionierung?

Aufgabe 38: (T) Abstrakter Interpreter (24 Pkt.)

Wir betrachten ein Paging-System: Jeder Prozess erzeugt eine Folge von Speicherzugriffen. Da jeder Speicherzugriff einer bestimmten virtuellen Seite entspricht, kann man sich den Speicherzugriff eines Prozesses als eine Liste von Seitennummern vorstellen, die Referenzkette. Ein Paging-System wird durch die Referenzkette des in Ausführung befindlichen Prozesses, den Seitenersetzungsalgorithmus und die Anzahl m der physischen Seitenrahmen charakterisiert.

Ein abstrakter Interpreter könnte wie folgt funktionieren: Er enthält eine interne Tabelle M, die den Zustand des Speichers repräsentiert. M hat

– n Zeilen, wobei n die Anzahl unterschiedlicher virtueller Seiten ist,

– so viele Spalten, wie es Speicherzugriffe gibt (also Anzahl der Elemente in der Referenzkette) und

– zwei Teile, nämlich einen oberen Teil, bestehend aus den ersten m Zeilen der Tabelle, sowie einen unteren Teil, bestehend aus den restlichen $n - m$ Zeilen.

Wird der Prozess gestartet, so beginnt die Abarbeitung der Referenzkette. Für jede Seite wird geprüft, ob sie im Speicher (d.h. im oberen Teil von M) liegt. Falls nicht, ist ein Seitenfehler aufgetreten. Es wird wie folgt vorgegangen:

– Die angeforderte Seite wird in die erste Zeile der jeweiligen Spalte geschrieben.

– Befand sich an dieser Stelle bereits eine Seite, so wandert diese (und auch alle folgenden Seiten) um eine Position nach unten.

Die folgende Referenzkette sei gegeben:

```
w = 0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 1 3 4 1
```

a. Stellen Sie die Tabelle M auf, und ermitteln Sie dadurch die Anzahl der Seitenfehler, unter der Annahme, dass $m = 4$ physische Seitenrahmen zur Verfügung stehen.

b. Welche Ihnen bekannte Seitenersetzungs-Strategie wird durch diesen abstrakten Interpreter realisiert?

c. Spezifizieren Sie zunächst informell, dann auch formell die Klasse der Keller-Algorithmen. Liegt hier ein Keller-Algorithmus vor? Welchen Vorteil haben Keller-Algorithmen gegenüber Algorithmen ohne diese Eigenschaft?

d. Die sogenannte Distanzkette ist eine abstrakte Repräsentation der Referenzkette. In ihr wird jede Seite durch ihren Abstand vom oberen Ende des Kellers bei einem Zugriff repräsentiert. Der Abstand einer Seite, die nicht im Speicher liegt, sei unendlich. Geben Sie für obiges Beispiel die Distanzkette an.

e. Entwickeln Sie einen Algorithmus, der es auf der Grundlage einer Distanzkette ermöglicht, die Anzahl der Seitenfehler für verschiedene Speichergrößen vorherzusagen.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 7
24.11.2004

Informatik III

Abgabetermin: 6.12.2004, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 3.1 bis 3.6 und 2.3

Aufgabe 30: (H) Scheduler-Simulation: Implementierung (10 Pkt.)

Implementieren Sie in dieser Aufgabe einen einfachen Priority-Scheduler in Java. Die Java-Klassen `Simulation`, `Scheduler`, `PriorityProcess` und `P` sind bereits gegeben. Außerdem ist mit der Klasse `Process` eine Implementierung des Prozessdeskriptors als Lösungsvorschlag des letzten Übungsblattes gegeben (online ab 30.11.). Aus Platzgründen wurden die Klassen nicht auf dieses Übungsblatt gedruckt. Bitte laden Sie sich die Dateien von der Info-3-Homepage herunter.

a. Ergänzen Sie den fehlenden Code in der Klasse `PriorityScheduler`. Verwenden Sie die Methoden `Thread.suspend()` und `Thread.resume()`, falls erforderlich. Achten Sie darauf, alle relevanten Aktionen durch Verwendung von `PriorityProcess.printInfo()` zu dokumentieren. Kommentieren Sie Ihre Lösung sinnvoll.

b. Beschreiben Sie kurz Sinn und Ablauf der `main`-Methode der Klasse `Simulation`.

c. Starten Sie die Simulation. In der Programmausgabe werden auch die Ausführungszeiten der Prozesse angegeben, wenn diese fertig gerechnet haben. Führen Sie die Simulation drei mal durch und dokumentieren Sie die Zeiten in einer Tabelle. Wie erklären Sie sich eventuelle Differenzen?

```
public class PriorityScheduler extends Scheduler
{
  private PriorityProcess current;

  public PriorityScheduler() {
    super();    //Aufruf: Konstruktor der Oberklasse
    System.out.println("Scheduling-Strategy: Priority-Scheduling");
  }

  public void run() {   //Scheduler läuft als ständiger Thread
    while(true) {       //in einer Endlosschleife
      try {
        if(current == null) { //kein Prozess aktiv
          current = PriorityProcess.getFirst();
          //zu aktivierender Prozess = erster in Liste
          System.out.print("Scheduler decided: Start with first process in queue: ");
          current.printInfo();  //Prozess-Informationen ausgeben
          current.start();      //Prozess starten
```

```
        }
20      else {                  //bereits ein Prozess aktiv
          if(!current.isAlive()) {
22        //aktiver Prozess lebt nicht mehr, ist also fertig
            current.removeProcess();    //Prozess aus Liste entfernen
24          current = PriorityProcess.getFirst();
            //zu aktivierender Prozess = nächster in Liste
26
            if(current.isSuspended()) {
28          //zu aktivierender Prozess wurde zuvor suspendiert
              System.out.print("Scheduler decided: Restore process: ");
30            current.printInfo();      //Prozess-Informationen ausgeben
              current.unsetSuspended(); //Suspend-Flag entfernen (suspended = false)
32            current.resume();         //Prozess wiederherstellen
            }
34          else {
            //zu aktivierender Prozess wurde nicht suspendiert (ist also "neu")
36            System.out.print("Scheduler decided: Switch to next process in queue: ");
              current.printInfo();      //Prozess-Informationen ausgeben
38            current.start();          //Prozess starten
            }
40        }

42        if(current != PriorityProcess.getFirst()) {
          //aktiver Prozess entspricht nicht erstem in Queue
44          ////////////////////////////////
            //                            //
46          //                            //
            // Zu ergänzender Programmteil //
48          //                            //
            //                            //
50          ////////////////////////////////
          }
52        else {
          //aktiver Prozess entspricht auch erstem in Queue, also beibehalten
54          System.out.print("Scheduler decided: Keep process ");
            current.printInfo();        //Prozess-Informationen ausgeben
56        }
        }
58      Thread.sleep(20);
      }
60    catch (Exception e) {
        System.out.println("Nothing more to do.");
62      System.exit(0); //Simulation beenden (alle Prozesse terminiert)
      }
64  }
  }
66}
```

Aufgabe 31: (H) Deadlocks: Wiederholung (10 Pkt.)

Gegeben sei ein Computersystem mit vier relevanten Betriebsmitteln/Ressourcen, die zum betrachteten Zeitpunkt t_0 ungebunden sind. Alle Betriebsmittel sind exklusiv. Drei Prozesse konkurrieren um die Betriebsmittel und tätigen die in der folgenden Tabelle skizzierten Anforderungen. Ein Tabelleneintrag $A(R_n)$ in der Zeile P_i und der Spalte t bedeutet, dass der Prozess P_i zum Zeitpunkt $t_0 + t$ auf das Betriebsmittel R_n zugreifen möchte. Eine Anforderung wird sofort in eine feste Ressourcenbindung umgesetzt, wenn die angeforderte Ressource zum jeweiligen Zeitpunkt frei ist. Andernfalls muss der Prozess so lange warten, bis das Betriebsmittel wieder freigegeben wurde.

	1	2	3	4	5	6	7
P_1	$A(R_1)$				$A(R_4)$		
P_2		$A(R_2)$				$A(R_1)$	
P_3			$A(R_4)$	$A(R_3)$	$A(R_2)$		

a. Skizzieren Sie alle Ressourcenbindungen und -anforderungen zum Zeitpunkt $t_0 + 7$ in einem Resource-Allocation-Graph.

b. Entscheiden Sie anhand des Graphen, ob sich das System zu diesem Zeitpunkt in einem Deadlock befindet. Begründen Sie Ihre Antwort.

c. Deadlocks lassen sich durch eine lineare Ordnung auf den Ressourcen vermeiden (siehe letztes Übungsblatt). Es sei die Ordnung $F(R_n) = n$ gegeben. Ressourcenanforderungen können nur in aufsteigender Reihenfolge erfüllt werden. Welches Problem entsteht hierbei für das obige Beispiel, wenn Sie annehmen, dass die Reihenfolge der Ressourcenanforderungen für einen Prozess nicht verändert werden darf? Welche Schlussfolgerung lässt dies über das Verfahren der linearen Ordnung auf Ressourcen zur Deadlock-Vermeidung zu?

d. Gibt es überhaupt eine Ordnung $F(R_n)$, sodass dieses Problem speziell für das gegebene Beispiel nicht auftritt? Geben Sie F an oder beweisen Sie, dass ein solches F nicht existiert.

Aufgabe 32: (T) Race Conditions & Semaphoren (25 Pkt.)

Von einer **Race Condition** spricht man, wenn das Ergebnis nebenläufiger Prozesse von der Reihenfolge der Prozessaktivierung abhängt. Im folgenden seien die beiden Prozesse Ehemann und Ehefrau gegeben. Beide wollen zur gleichen Zeit an unterschiedlichen Geldautomaten Geld vom gemeinsamen Konto abbuchen.

Zeile	Ehemann	Ehefrau
	...	...
1	Lies Kontostand k;	Lies Kontostand k;
2	k' := k - 100;	k' := k - 400;
3	Schreibe Kontostand k';	Schreibe Kontostand k';
	...	...

a. Geben Sie eine zeitliche Abfolge der Prozesse Ehefrau und Ehemann an, sodass eine Race Condition eintritt und der Kontostand zum Ende beider Transaktionen inkonsistent wird.

b. Nennen Sie die vier Bedingungen, die erfüllt sein müssen, um Race Conditions zu verhindern.

c. Ein erster Lösungsvorschlag: Eine boolesche Variable wird von einem Prozess auf TRUE gesetzt, bevor er den kritischen Bereich betritt. Vorher wird geprüft, ob die Variable nicht bereits von einem anderen Prozess auf TRUE gesetzt wurde (in diesem Fall wird kein Eintritt gewährt).

Zeile	Ehemann	Ehefrau
	...	...
1	a:= FALSE;	b:= FALSE;
2	WHILE b DO {nothing};	WHILE a DO {nothing};
3	a:= TRUE;	b:= TRUE;
4	Lies Kontostand k;	Lies Kontostand k;
5	k' := k - 100;	k' := k - 400;
6	Schreibe Kontostand k';	Schreibe Kontostand k';
7	a:= FALSE;	b:= FALSE;
	...	...

Zeigen Sie, dass dieses Vorgehen nicht zum Ziel führt. Welche der vier Bedingungen wird verletzt?

d. Der nächste Lösungsvorschlag:

Zeile	Ehemann	Ehefrau
	...	...
1	a:= TRUE;	b:= TRUE;
2	y:= 0;	y:= 1;
3	WHILE (y=1) OR b DO {nothing};	WHILE (y=0) OR a DO {nothing};
4	Lies Kontostand k;	Lies Kontostand k;
5	k' := k - 100;	k' := k - 400;
6	Schreibe Kontostand k';	Schreibe Kontostand k';
7	a:= FALSE;	b:= FALSE;
	...	...

Bewerten Sie auch diesen Ansatz.

e. Lösen Sie das vorliegende Synchronisationsproblem mit Hilfe des Algorithmus von Peterson.

f. Lösen Sie das Problem unter Einsatz eines Binärsemaphors.

g. Worin besteht (in diesem Beispiel und allgemein) der Vorteil beim Einsatz von Semaphoren im Vergleich zu Petersons Lösung?

Aufgabe 33: (P) Java: Monitore (8 Pkt.)

Bei den amerikanischen Präsidentschaftswahlen werden in vielen Bundesstaaten Wahl-Automaten (statt Stimmzettel) eingesetzt. Leider kommt es damit immer wieder zu Problemen. Sie werden beauftragt, die Automaten zu verbessern. Betrachten Sie dazu die nachfolgende Implementierung (Download von der Info-3-Homepage möglich) für ein kleines Wahllokal mit zehn Wahl-Automaten, dargestellt als parallel ablaufende Threads. Im betrachteten Zeitraum wählen an jedem Automaten nacheinander 20 Personen. Die Entscheidung (für Demokraten oder Republikaner) wird hier zufällig generiert.

a. Machen Sie sich die Funktionsweise des Java-Programms klar, kommentieren Sie die wichtigsten Stellen im Code sinnvoll, kompilieren Sie das Programm und führen Sie es aus. Die Ausgabe sollte eigentlich ein gut lesbares Log-File liefern, aus dem später jede Stimmabgabe ablesbar und rekonstruierbar ist. Beobachten Sie, was stattdessen passiert. Welches Problem besteht?

b. Zur Synchronisation der Zugriffe unterschiedlicher Threads auf gemeinsam genutzte Methoden stellt Java das **Monitor-Konzept** bereit. Lösen Sie damit die auftretenden Probleme.

c. Es gibt zwei Modelle, wie die Ausführung in einem Monitor nach dem Aufruf von `signal` (in Java: `notify`) fortfährt (*A*: signalisierender Prozess, B: aufgeweckter Prozess):

   - *Signal-and-wait:* *A* muss nach seiner Signalisierung den Monitor sofort freigeben und warten, bis B den Monitor verlassen hat oder auf eine andere Bedingung wartet.
   - *Signal-and-continue:* B muss warten, bis *A* den Monitor verlassen hat oder auf eine andere Bedingung wartet.

   Welchem Konzept folgt Java?

```
import java.util.Random;

public class WahlenNaiv {

  public static void main(String args[]) {
    Waehlen w = new Waehlen();
    Automat automat[]= new Automat[10];

    for (int i= 0; i < 10; i++ ){
      automat[i] = new Automat(w);
      automat[i].start();
    }
  }
}

class Automat extends Thread {

  private Waehlen w;
  Random r;
  int rand;

  public Automat(Waehlen w) {
    this.w = w;
    r = new Random();
  }

  public void run() {
    for(int i = 0; i < 20; ++i) {
      rand = r.nextInt(2);
      if(rand==0) w.waehleDemokraten();
      else w.waehleRepublikaner();
    }
  }
}

class Waehlen {
  private int dem;
  private int rep;
  private String leading;
  private int leadCount;

  public Waehlen() {
    leading = new String("unentschieden");
    leadCount = 0;
  }

  public void waehleDemokraten() {
    dem++;
```

```
      System.out.println("1 Stimme für Demokraten");
50    if(leading.equals("Demokraten führen") || leading.equals("unentschieden")) {
        leadCount++;
52      leading="Demokraten führen";
      }
54    else { //Republikaner führen
        leadCount--;
56      if(leadCount==0) leading="unentschieden";
      }
58    System.out.println("Demokraten: " + dem + ", Republikaner: " + rep);
      System.out.println("Zwischenstand: " + leading);
60    System.out.println("-------------------------------------");
    }
62
    public void waehleRepublikaner() {
64    rep++;
      System.out.println("1 Stimme für Republikaner");
66    if(leading.equals("Republikaner führen") || leading.equals("unentschieden")) {
        leadCount++;
68      leading="Republikaner führen";
      }
70    else { //Demokraten führen
        leadCount--;
72      if(leadCount==0) leading="unentschieden";
      }
74    System.out.println("Demokraten: " + dem + ", Republikaner: " + rep);
      System.out.println("Zwischenstand: " + leading);
76    System.out.println("-------------------------------------");
    }
78 }
```

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 6
17.11.2004

Informatik III

Abgabetermin: 29.11.2004, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 3.1 bis 3.6

Aufgabe 25: (H) Scheduler-Simulation: Prozessdeskriptor (10 Pkt.)

Diese Aufgabe baut auf den Überlegungen aus Aufgabe 24 (letztes Übungsblatt) auf. (Ziel ist die Implementierung der Simulation eines einfachen Schedulers in Java.) Implementieren Sie eine Klasse `Process` für den *allgemeinen* (d.h. für alle gängigen Scheduling-Strategien grundlegenden) Prozessdeskriptor. Beachten Sie die folgenden Modellierungshinweise:

- Jeder Prozess besitzt eine Prozess-ID, die ihm bei seiner Erzeugung automatisch vom System zugeteilt wird.
- Im Prozessdeskriptor muss festgehalten werden, ob ein Prozess gerade aktiv ist oder suspendiert wurde. Die Klasse muss Methoden zur Verfügung stellen, um den jeweiligen Zustand abzufragen oder zu verändern.
- Eine Methode `setArriveTime()` soll dem jeweiligen Prozess bei ihrem Aufruf die aktuelle Systemzeit als Ankunftszeit zuweisen und in einer Variablen `arriveTime` speichern. Diese Methode hat keine Parameter und keinen Rückgabewert. Sie wird ausschließlich **von Unterklassen** von `Process` aufgerufen, also nicht schon im Konstruktor dieser Klasse.
- Analog zu `setArriveTime` soll die Methode `setDepartTime()` dem jeweiligen Prozess bei ihrem Aufruf die aktuelle Zeit als Endzeitpunkt zuweisen und in einer Variablen `departTime` speichern.
- Schließlich soll eine Methode zur Verfügung gestellt werden, die die Lebensdauer eines Prozesses als Differenz von `departTime` und `arriveTime` berechnet und zurück gibt.

Tipp: Das folgende Programmfragment legt ein Objekt `d` vom Typ `Date` an, das das aktuelle Datum und die genau Systemzeit enthält. In der Variablen `t` wird diese Zeit dann als Anzahl der Millisekunden seit dem 1. Januar 1970 (0.00 Uhr) gespeichert.

```
Date d = new Date();
long t = d.getTime();
```

Damit das funktioniert, muss zuvor die Klasse `java.util.Date` importiert werden.

Aufgabe 26: (H) Deadlock-Vermeidung (8 Pkt.)

Eine Methode, um Deadlocks zu vermeiden, ist es, eine der Bedingungen für das Entstehen von Deadlocks im Vorhinein auszuschließen.

a. Geben sie die vier Voraussetzungen für die Entstehung eines Deadlocks an.

b. Beschreiben Sie, wie durch eine Ordnung der Ressourcen bei geeigneter Reservierungsstrategie Deadlocks vermieden werden können.

   Tipp: Ordnung der Ressourcen bedeutet: Ob ein Prozess ein Betriebsmittel anfordern darf, hängt nicht nur davon ab, ob dieses gerade frei ist, sondern auch davon, welche Betriebsmittel der Prozess zuvor schon angefordert hat. Welche der vier Deadlock-Bedingungen könnte man mit diesem Ansatz ausschließen?

Aufgabe 27: (T) Deadlocks: Modellierung (10 Pkt.)

In dieser Aufgabe werden zwei Methoden zur Modellierung von Deadlocks in Computersystemen betrachtet: der *Resource-Allocation-Graph* und das *Prozessfortschrittsdiagramm*.

a. Erläutern Sie beide Verfahren jeweils anhand eines konkreten Beispiels.

b. Vergleichen Sie die beiden Modellierungsvarianten anhand sinnvoller Kriterien.

Aufgabe 28: (K) Multiple Choice: Deadlock-Erkennung (5 Pkt.)

Gegeben seien zwei Prozesse P und Q. Prozess P benötigt zu seiner Ausführung acht Zeiteinheiten, Prozess Q sechs Zeiteinheiten. Es steht das Betriebsmittel BM 1 zur Verfügung, das von den Prozessen während ihrer Ausführung benötigt wird. P benötigt BM 1 im Zeitraum 2 bis 4, Q benötigt BM 1 ebenfalls im Zeitraum 2 bis 4. Welche der folgenden Aussagen sind wahr?

a. Eine Deadlock-freie Ausführung von P und Q ist unmöglich.

b. Im zugehörigen Prozessfortschrittsdiagramm gibt es einen unsicheren Bereich.

c. Im zugehörigen Prozessfortschrittsdiagramm gibt es einen unmöglichen Bereich.

d. Ein Deadlock kann hier niemals eintreten, da die Deadlock-Bedingung „Circular Wait“ nicht erfüllt ist.

e. Ein Deadlock tritt frühestens nach zwei, spätestens nach vier Zeiteinheiten ein.

Aufgabe 29: (T) Deadlock-Detection & -Recovery (20 Pkt.)

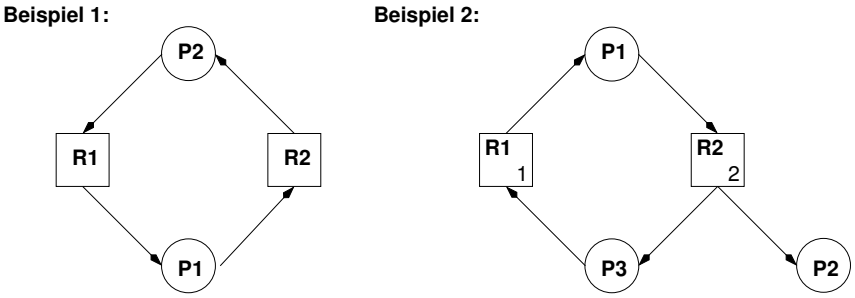
In dieser Aufgabe werden gewichtete Resource-Allocation-Graphen wie folgt verwendet:

- Ein Rechteck bezeichnet eine Ressource R_i, wobei die Anzahl der vorhandenen Instanzen dieser Ressource angegeben ist.
- Ein Kreis bezeichnet einen Prozess P_i.
- Eine Kante (P_i, R_j) mit dem Gewicht n modelliert die Anforderung von n Einheiten der Ressource R_j durch den Prozess P_i.
- Eine Kante (R_i, P_j) mit dem Gewicht n modelliert die Belegung von n Einheiten der Ressource R_i durch den Prozess P_j.

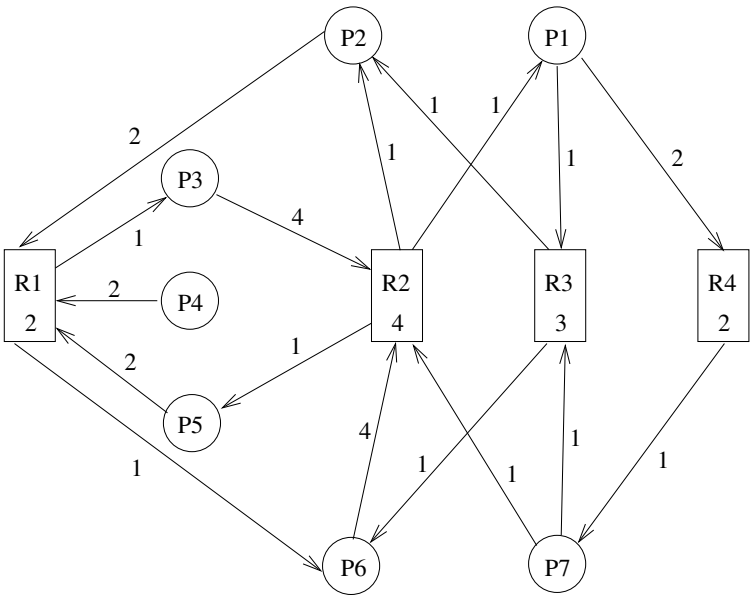
Gibt es in einem solchen Graphen keine Zyklen, so besteht auch kein Deadlock in dem zugrundeliegenden Prozesssystem.
Wenn es einen Zyklus im Resource-Allocation-Graph gibt, so unterscheidet man zwei Fälle:

- Besitzt jede Ressource höchstens eine Instanz, so liegt ein Deadlock vor (notwendige und hinreichende Bedingung).
- Besitzt jede Ressource mehr als eine Instanz, so kann ein Deadlock vorliegen, muss jedoch nicht. In diesem Fall ist der Zyklus eine notwendige, nicht jedoch hinreichende Bedingung.

In der folgenden Abbildung ist also der erste Graph verklemmt, der zweite jedoch nicht (wenn P_2 fertig ist, kann P_1 eine Instanz von R_2 belegen und damit auch fortfahren):



Sei nun der folgende markierte Resource-Allocation-Graph G für sieben Prozesse $P_1, \ldots, P_7$ und vier Betriebsmittel(-klassen) $R_1, \ldots, R_4$ gegeben:



a. Geben Sie für alle Prozesse P_i und alle Betriebsmittel R_j folgende Größen an:

   - $b(P_i, R_j)$: die von Prozess P_i belegten Einheiten des Betriebsmittels R_j
   - $f(P_i, R_j)$: die von Prozess P_i geforderten Einheiten des Betriebsmittels R_j
   - $\nu(R_j)$: die Anzahl der insgesamt vorhandenen Einheiten des Betriebsmittels R_j
   - $a(R_j)$: die Anzahl der noch zur Verfügung stehenden Einheiten des Betriebsmittels R_j

b. Offensichtlich lässt sich auf ein solches System die folgende Reduzierung anwenden: Gibt es einen Prozess P_i, sodass für alle j gilt $f(P_i, R_j) \leq a(R_j)$, so entferne man alle Kanten zu P_i aus dem Graphen und setze für alle j $a(R_j) := a(R_j) + f(P_i, R_j)$. Reduzieren Sie G so weit wie möglich und kommentieren Sie das Ergebnis.

c. Nennen Sie drei Methoden der Deadlock-Behebung (Deadlock-Recovery).

d. Bestimmen Sie, falls im Graphen eine Verklemmung vorliegt, eine minimale Prozessmenge $mp_{\min} \subseteq \{P_1, \ldots, P_7\}$, sodass nach Abbruch aller Prozesse in $mp_{\min}$ keine Verklemmung mehr vorliegt.

e. Zu welchem Ergebnis kommen Sie, wenn das Betriebsmittel R_1 aus zwei Einheiten mehr besteht?

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 5
10.11.2004

Informatik III

Abgabetermin: 22.11.2004, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 2.5

Aufgabe 21: (H) CPU-Scheduling: Wartezeiten (25 Pkt.)

Wenn auf einer CPU mehrere Prozesse (pseudo-)parallel ausgeführt werden, kommt es für einzelne Prozesse in der Regel zu Wartezeiten vor ihrer Ausführung und (preemptives Scheduling vorausgesetzt) zwischen zwei Ausführungseinheiten. In dieser Aufgabe sollen verschiedene Scheduling-Strategien im Hinblick auf die mittleren Warte- (und Verweil-)Zeiten der Prozesse verglichen werden.

Dazu sei ein Ein-Prozessor-System mit elf Aufträgen (Prozessen) $A_1, \ldots, A_{11}$, ihren Ankunftszeitpunkten $a = (0, 2, 3, 4, 6, 7, 12, 14, 16, 22, 26)$ und den Bedienzeiten $b = (5, 6, 2, 3, 2, 1, 2, 2, 2, 3, 1)$ gegeben (d.h. Auftrag A_i kommt zum Zeitpunkt a_i ins System und benötigt die CPU für b_i Zeiteinheiten). Vor dem Zeitpunkt $t = 0$ seien keine Aufträge vorhanden.

a. Stellen Sie das Prozess-Scheduling für die folgenden drei Strategien grafisch dar. Verwenden Sie dabei (abweichend von Aufgabe 20 auf dem letzten Übungsblatt) eine Darstellung, in der auch Warte- und Verweilzeiten erkennbar sind. Lesen Sie aus Ihren Diagrammen jeweils die Terminierungszeitpunkte sowie die Warte- und Verweilzeiten der einzelnen Prozesse ab. Berechnen Sie dann die mittlere Wartezeit und die mittlere Verweilzeit für jede der drei Strategien. Beachten Sie, dass Ausführungsunterbrechungen ausschließlich zu Ankunftszeitpunkten neuer Aufträge oder zu Terminierungszeitpunkten fertiger Aufträge möglich sind.

   (i) **SRPT (Shortest Remaining Processing Time)**: In jedem Zeitintervall wird einer der Aufträge mit kürzester Restbedienzeit ausgeführt.

   (ii) **LPT (Largest Processing Time)**: Es wird jeweils der Auftrag mit der längsten Gesamt(!)-Bedienzeit ausgeführt.

   (iii) **FCFS (First Come First Serve)**: Die Aufträge werden in der Reihenfolge ihrer Ankunft nicht-unterbrechend ausgeführt.

b. Der Vektor der Ankunftszeiten werde modifiziert zu $a' = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, d.h. alle Aufträge liegen schon von Beginn an vor. Die Aufträge werden nun sequenziell nach einer Zeitscheibenstrategie mit dem Quantum q (Round-Robin-Strategie) bedient, beginnend mit A_1. Ermitteln Sie wieder Terminierungszeitpunkt, Warte- und Verweilzeit für jeden Auftrag, sowie die mittlere Wartezeit und die mittlere Verweilzeit, und zwar

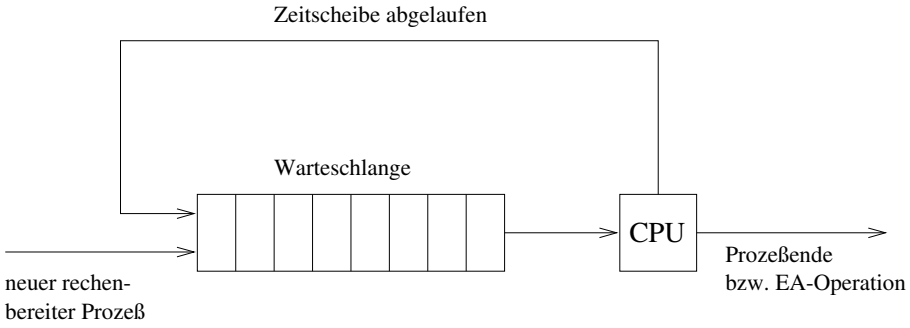
   (i) für das Zeitquantum $q_1 = \frac{3}{2}$ und

   (ii) für das Zeitquantum $q_2 = \frac{1}{2}$.

c. Warum lassen sich die Ergebnisse für die mittleren Warte-/Verweilzeiten aus Teilaufgabe a. nicht mit denen aus Teilaufgabe b. vergleichen?

Aufgabe 22: (T) Scheduling: Round-Robin (8 Pkt.)

Eine gebräuchliche Strategie zur Rechnerkernvergabe stellt das in der Vorlesung vorgestellte **Round-Robin-Verfahren** dar. Hierbei wird eine Warteschlange von Prozessen verwaltet, wobei jeweils dem in der Schlange ersten Prozess für eine feste Zeitdauer Q (die sog. *Zeitscheibe*) die CPU zugeteilt wird. Prozesse, deren Zeitscheibe abgelaufen ist, werden am Ende der Warteschlange wieder eingereiht.



Empirische Messungen haben ergeben, dass ein Prozess im Mittel für eine Zeitspanne T die CPU benutzt, bevor er das obige Warteschlangensystem aufgrund einer angestoßenen E/A-Operation bzw. bei Prozessende verlässt. Ein Prozesswechsel benötige eine Zeitdauer S, die als *Overhead* verloren geht.

a. Geben Sie für Round-Robin-Scheduling mit einer Zeitscheibe Q eine Formel für die CPU-Auslastung an.

b. Welche CPU-Auslastung ergibt sich in den folgenden Fällen?

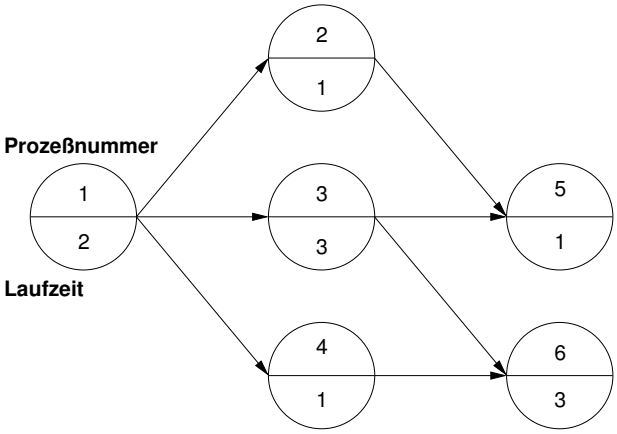
   - $Q > T$,
   - $Q = S \ll T$ und
   - $Q \approx 0$.

c. Leiten Sie daraus Kritieren für den Wert von Q ab. Dabei soll vorausgesetzt werden, dass stets rechenbereite Prozesse zur Verfügung stehen.

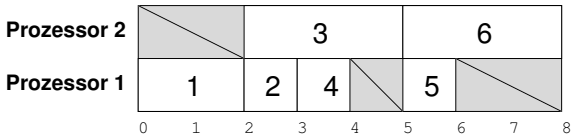
Aufgabe 23: (T) Grahams List Scheduling (16 Pkt.)

In der heutigen Zeit ist es interessant, parallele Programme auch auf mehreren Prozessoren gleichzeitig ausführen zu können. Grahams List-Scheduling ist ein für diesen Zweck entwickelter Scheduling-Algorithmus, der nach dem Greedy-Prinzip arbeitet: Ein freier Prozessor nimmt den ersten rechenbereiten Prozess aus der Prozesswarteschlange (geordnet nach Prozessnummern) und bearbeitet diesen. Sind mehrere Prozessoren frei, bedient sich der mit der kleineren Nummer zuerst.

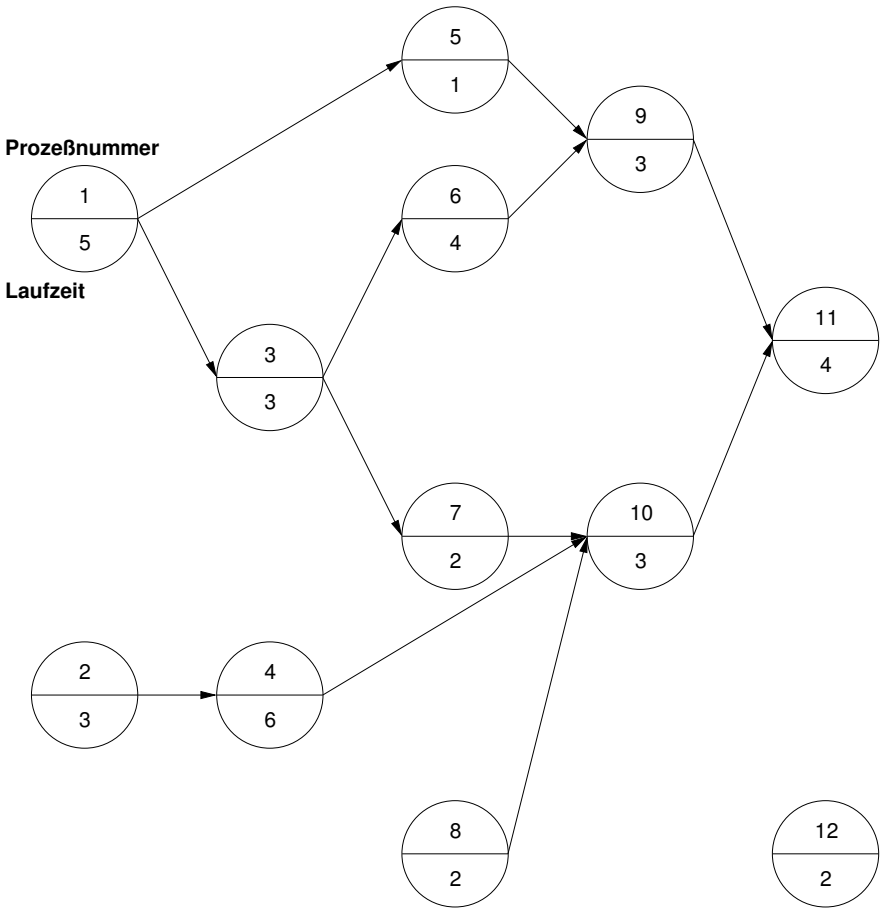
Die folgenden sechs Prozesse (gegeben durch Prozessnummer und Laufzeit) seien, wie im folgenden Diagramm dargestellt, voneinander abhängig:



Prozesse 2, 3 und 4 müssen also erst auf das Ergebnis von Prozess 1, Prozess 5 auf das Ergebnis von 2 und 3 etc. warten.
Bei zwei Prozessoren ergibt sich folgendes Gantt-Diagramm:



a. Geben Sie für folgenden Abhängigkeits-Graphen das zugehörige Gantt-Diagramm für drei Prozessoren an:



b. Bei Grahams List-Scheduling existieren drei Anomalien: Die Laufzeit kann steigen bei

(i) Erhöhung der Prozessoranzahl,

(ii) Entfernung von Kanten und

(iii) Reduzierung der Dauer eines Prozesses.

Geben Sie jeweils ein Beispiel an.

Aufgabe 24: (P) Scheduler-Simulation: Grundlagen (12 Pkt.)

Sie haben verschiedene Arten von Schedulern (preemptiv, nicht-preemptiv) sowie unterschiedliche Scheduling-Strategien (Prioritäten, FCFS, Round Robin usw.) in der Theorie kennen gelernt. Nun sollen Sie selbst einen Scheduler in der Programmiersprache Java implementieren. Es ist klar, dass es sich dabei nur um eine Simulation, nicht um einen realen und funktionsfähigen Scheduler handeln kann. Beantworten Sie die folgenden Fragen zunächst mit der Zielsetzung, einen preemptiven Priority-Scheduler zu simulieren.

a. Bei preemptivem Scheduling müssen laufende Prozesse nach bestimmten, festgelegten Zeitintervallen unterbrochen und die Kontrolle an den Scheduler abgegeben werden. Dieser entscheidet dann nach seiner Strategie, ob der gerade abgebrochene Prozess fortgesetzt wird oder ein anderer Prozess zur Ausführung kommt. Wie werden diese Unterbrechungen in realen Systemen realisiert, und wie können sie in der Programmiersprache Java simuliert werden? Welche Probleme können dabei auftreten?

b. In realen Systemen besitzt jeder Prozess einen Prozesskontrollblock, der alle Statusinformationen (Prozess-ID, Priorität usw.) für diesen Prozess enthält. Wie würden Sie diesen Umstand in Java realisieren?

c. Alle aktuellen Prozesse müssen in geeigneter Weise im Speicher gehalten, also verwaltet werden. Welche Datenstruktur würden Sie dazu einsetzen? Begründen Sie Ihre Antwort, und nennen Sie die wichtigsten Eigenschaften und Operationen auf dieser Datenstruktur.

d. Manche Prozess-Attribute werden bei allen Strategien benötigt (Beispiel: Prozess-ID), andere hingegen nicht (Beispiel: Prozess-Priorität). Wie lässt sich dies bei der Implementierung berücksichtigen? Wie können Sie Ihren Programmcode auf diese Weise für Erweiterungen um neue Scheduling-Strategien offen halten, ohne dass größere Code-Fragmente mehrfach in verschiedenen Klassen vorkommen?

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 4
3.11.2004

Informatik III

Abgabetermin: 15.11.2004, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 2.5

Aufgabe 16: (H) Wiederholung: Threads in Java (4 Pkt.)

Das folgende kurze Java-Programm können Sie von der Info 3-Homepage herunterladen:

```
public class Test extends Thread {
        String msg;

        Test(String m) {
                msg = m;
        }

        public void start() {
                for (int i = 0; i < 1000; i++) {
                        System.out.println(msg);
                }
        }

        public static void main(String[] args) {
                Test t1 = new Test("ping---->");
                Test t2 = new Test("<----pong");
                t1.start();
                t2.start();
        }
}
```

a. Welche Ausgabe liefert dieses Programm? Ist die Ausgabe deterministisch?

b. Damit die Instanzen `t1` und `t2` threaded ausgeführt werden, muss eine kleine Änderung an der Klasse `Test` vorgenommen werden. Skizzieren Sie diese Änderung und erklären Sie in wenigen Sätzen, was sie bewirkt.

Aufgabe 17: (H) Grundlagen von Threads (6 Pkt.)

a. Nennen Sie zwei Gründe, warum es nicht sinnvoll ist, zu viele Threads zu verwenden.

b. Nennen Sie zwei Gründe, warum Threads sinnvoll/wichtig sind.

c. Nennen Sie die wesentlichen Unterschiede zwischen User-Level und Kernel-Level Threads aus Sicht des Betriebssystems.

Aufgabe 18: (T) Thread-Verwaltung & Thread-Wechsel (6 Pkt.)

a. Welche Zustandsinformationen teilt ein Thread mit Threads desselben Prozesses, welche nicht?

b. Stimmt es, dass ein Kontextswitch zwischen zwei Threads desselben Prozesses weniger Aufwand verursacht als einer zwischen Threads in unterschiedlichen Prozessen?

Aufgabe 19: (K) Java-Threadmodell (6 Pkt.)

a. Zeichnen Sie ein Zustandsübergangsdiagramm für das Java-Threadmodell. Markieren Sie die Übergänge mit ihren jeweiligen Ursachen. Tragen Sie auch *deprecated*-Methoden der Thread-Klasse ein und markieren Sie diese.

b. Sind Java-Threads aus Sicht des Betriebssystems User- oder Kernel-Level-Threads?

Aufgabe 20: (T) CPU-Scheduling: Einführung (15 Pkt.)

Folgende Prozesse sollen betrachtet werden (die Zeiten seien in beliebigen Zeiteinheiten gegeben, die Prioritäten von 0 bis 2, wobei 0 die höchste Priorität bezeichne):

Prozess	Ankunftszeitpunkt	Laufzeit	Priorität
A	0	4	1
B	0	5	0
C	1	2	1
D	3	6	1
E	4	2	2
F	4	3	2
G	6	3	0
H	9	5	0
I	10	1	1

Der Ankunftszeitpunkt t bezeichnet auch den Zeipunkt, zu dem der Prozess in die Warteschlange eingereiht wird. Kommen zwei Prozesse zur gleichen Zeit, so wird die Ordnung auf den Prozessnamen herangezogen (Prozess A vor Prozess B, usw.). Wird ein Prozess vor seinem Terminieren zum Zeitpunkt t' unterbrochen, so reiht er sich hinten in die Warteschlange mit Ankunftszeit t' wieder ein.

Geben Sie für die Strategien FIFO (First In First Out), SJF (Shortest Job First), Round Robin mit Quantum $t = 3$, sowie für preemptives und nicht-preemptives Priority-Scheduling jeweils in Form eines Gantt-Charts für die ersten 20 (0 bis 19) Zeiteinheiten an, wann welchem Prozess Rechenzeit zugeteilt wird, und wann die Prozesse ggf. terminieren.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 3
27.10.2004

Informatik III

Abgabetermin: 8.11.2004, 12:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an.

Für Multiple Choice-Aufgaben beachten Sie bitte: Im Falle einer Bewertung erhalten Sie für jede korrekte Kennzeichnung 1 Punkt, für jede falsche Kennzeichnung wird 1 Punkt abgezogen. Für Fragen/Statements, die Sie nicht bearbeiten, erhalten Sie keine Punkte, es werden aber auch keine abgezogen.

Lesen: A. Tanenbaum, Moderne Betriebssysteme: Kapitel 2.1 bis 2.4

Aufgabe 11: (H) Zustand-Prozessmodelle (15 Pkt.)

a. Ausgehend vom 5-Zustands-Prozessmodell:

   (i) Geben Sie für jeden Zustandsübergang ein Beispiel an, das den jeweiligen Zustandswechsel auslösen könnte.

   (ii) Welche Zustände kann man einsparen, wenn das Betriebssystem in reinem Batch-Betrieb arbeitet?

   (iii) Wie ändert sich das Modell für ein Zwei-Prozessor-System?

b. Ausgehend vom 7-Zustands-Prozessmodell: Überlegen Sie sich verschiedene Gründe, aus denen ein Prozess in einen Suspendiert-Zustand übergehen kann.

Aufgabe 12: (H) Multiple Choice: Prozesse (5 Pkt.)

Sind die folgenden Aussagen wahr oder falsch?

a. Ein Prozess kann als Eigentümer von Ressourcen, ein Thread hingegen als Ausführungspfad verstanden werden.

b. Multiprogramming trägt dazu bei, das Problem der Wartezeiten durch langsame E/A-Vorgänge zu minimieren.

c. Dispatching bedeutet: Zuordnung des Rechnerkerns an einen Prozess/Thread.

d. Der Prozesskontrollblock (PCB) enthält alle Informationen, die das Betriebssystem über einen Prozess benötigt.

e. Um Prozesse in beliebiger Reihenfolge aktivieren und deaktivieren zu können, sollte deren Verwaltung kellerartig erfolgen.

Aufgabe 13: (K) Modellierung des Mehrprogrammbetriebs (10 Pkt.)

Mit dem *Mehrprogrammgrad* n bezeichnet man die Anzahl der Prozesse, die im Mehrprogrammbetrieb gleichzeitig im Hauptspeicher gehalten werden. Von der gesamten Bearbeitungszeit T eines Prozesses entfalle im Mittel eine Zeitspanne $p \cdot T$ auf das Warten auf das Ende von E/A-Operationen ($0 \leq p \leq 1$).

a. Geben Sie (ein Einprozessorsystem vorausgesetzt) eine Abschätzung für die CPU-Auslastung in Abhängigkeit von n und p an, unter der Annahme, dass alle n Prozesse unabhängig voneinander ihre E/A-Operationen durchführen.

b. Gegeben sei ein Rechner mit einem Hauptspeicher von 64 MByte, von denen das Betriebssystem 16 MByte belege. Ein Benutzerprozess habe im Mittel eine Größe von 8 MByte. Welche Durchsatzsteigerung ist mit der Abschätzung aus Teilaufgabe (a) zu erwarten, falls der Hauptspeicher verdoppelt wird und p den Wert $p = 0.85$ besitzt?

Aufgabe 14: (T) Betriebsmittel (16 Pkt.)

Eine der wesentlichen Aufgaben eines Betriebssystems ist die Verwaltung von Betriebsmitteln.

a. Finden Sie für die folgenden Klassen von Betriebsmitteln jeweils Beispiele aus dem Hardware- oder aus dem Software-Bereich:

   (i) einmalig benutzbar
   (ii) wiederholt benutzbar
   (iii) parallel benutzbar
   (iv) unterbrechbar
   (v) nicht unterbrechbar

b. Bei der Betriebsmittelverwaltung sind eine Reihe von strategischen Entscheidungen zu treffen. Geben Sie einige Problemkreise an, in denen solche Entscheidungen anfallen, und geben Sie Strategien bzw. Mechanismen zu deren Lösung an.

Aufgabe 15: (P) Java: Einsatz von Threads (20 Pkt.)

Das folgende Java-Programm können Sie von der Info 3-Homepage herunterladen. Es lässt einen virtuellen Ball innerhalb einer gegebenen Fläche „tanzen“:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Bounce
{  public static void main(String[] args)
   {  JFrame frame = new BounceFrame();
      frame.show();
   }
}

class BounceFrame extends JFrame
{  public BounceFrame()
   {  setSize(300, 200);
      setTitle("Bounce");
```

```
16
       addWindowListener(new WindowAdapter()
18        {  public void windowClosing(WindowEvent e)
             {  System.exit(0);
20           }
          } );
22
       Container contentPane = getContentPane();
24     canvas = new JPanel();
       contentPane.add(canvas, "Center");
26     JPanel p = new JPanel();
       addButton(p, "Start",
28        new ActionListener()
          {  public void actionPerformed(ActionEvent evt)
30           {  Ball b = new Ball(canvas);
               b.bounce();
32           }
          });
34
       addButton(p, "Close",
36        new ActionListener()
          {  public void actionPerformed(ActionEvent evt)
38           {  System.exit(0);
             }
40        });
       contentPane.add(p, "South");
42   }

44   public void addButton(Container c, String title,
       ActionListener a)
46   {  JButton b = new JButton(title);
       c.add(b);
48     b.addActionListener(a);
     }
50
     private JPanel canvas;
52 }

54 class Ball
   {  private JPanel box;
56   private static final int XSIZE = 10;
     private static final int YSIZE = 10;
58   private int x = 0;
     private int y = 0;
60   private int dx = 2;
     private int dy = 2;
62   // Der Ball beginnt bei (0,0) und bewegt sich um 2

64   public Ball(JPanel b) { box = b; }

66   public void draw()
     {  Graphics g = box.getGraphics();
68     g.fillOval(x, y, XSIZE, YSIZE);
       g.dispose();
70   }

72   public void move()
     {  Graphics g = box.getGraphics();
```

```
74        g.setXORMode(box.getBackground());
          g.fillOval(x, y, XSIZE, YSIZE);
76        x += dx;
          y += dy;
78        Dimension d = box.getSize();
          if (x < 0)
80        { x = 0; dx = -dx; }
          if (x + XSIZE >= d.width)
82        { x = d.width - XSIZE; dx = -dx; }
          if (y < 0)
84        { y = 0; dy = -dy; }
          if (y + YSIZE >= d.height)
86        { y = d.height - YSIZE; dy = -dy; }
          g.fillOval(x, y, XSIZE, YSIZE);
88        g.dispose();
          // Kein Aufruf der draw-Methode wg. getGraphics()
90     }

92     public void bounce()
       {  draw();
94        for (int i = 1; i <= 1000; i++)
          {  move();
96           try { Thread.sleep(10); }
             catch(InterruptedException e) {}
98        }
       }
100
   }
```

a. Kompilieren Sie das Programm, und führen Sie es aus. Beschreiben Sie kurz, welche Aufgaben die folgenden drei Methoden der Klasse `Ball` erfüllen:

   (i) `draw()`

   (ii) `move()`

   (iii) `bounce()`

b. Wie müssten Sie den Rumpf der Methode `bounce()` verändern, sodass sich der Ball langsamer bewegt?

c. Gibt es noch eine andere Möglichkeit, die Geschwindigkeit des Balls zu beeinflussen?

d. Wie müssten Sie den Rumpf der Methode `bounce()` verändern, sodass sich sich der Ball bei gleicher Geschwindigkeit länger im Feld bewegt?

e. Modifizieren Sie das Programm so, dass mehrere Bälle gleichzeitig tanzen können. Beim Drücken des `Start`-Buttons soll jeweils ein neuer Ball ins Feld geschickt werden.

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
PROF. DR. HANS JÜRGEN OHLBACH

Wintersemester 2004/2005
Übungsblatt 2
20.10.2004

Informatik III

Abgabetermin: 2.11.2004, 13:00 Uhr s.t.

Achtung: Bitte geben Sie bei der Abgabe Ihrer Hausaufgaben immer die **Nummer Ihrer Übungsgruppe**, Ihren Namen und Ihre Matrikelnummer an. Wegen des Feiertags am Montag (1.11.) verschiebt sich der Abgabetermin auf Dienstag, 2.11., 13:00 Uhr.

Lesen: —

Aufgabe 6: (H) Einfache Shell-Programmierung (8 Pkt.)

a. Schreiben Sie ein Shell-Skript, das alle Dateien des aktuellen Verzeichnisses ausgibt, deren Namen den Teilstring „info“ enthalten.

b. Die Variable `$*` ist eine spezielle Variable, die alle übergebenen Kommandozeilenargumente enthält. Schreiben Sie ein Shell-Skript `newname`, das allen übergebenen Dateien das Kürzel „new_“ voranstellt.
`./newname datei1 datei2 datei3` soll zum Beispiel die drei Dateien zu `new_datei1`, `new_datei2` und `new_datei3` umbenennen.

c. Wie müssen Sie das Skript aus der vorherigen Teilaufgabe verändern, damit bei seiner Ausführung allen Dateien im aktuellen Ordner das Kürzel „new_“ vorangestellt wird?

Aufgabe 7: (H) Nebenläufige Abläufe (6 Pkt.)

Betrachten Sie das untenstehende nebenläufige Programm mit zwei Prozessen, P und Q. A, B, C, D und E sind beliebige atomare Anweisungen. Die beiden Prozesse (realisiert als Prozeduren) seien parallel gestartet:

```
PROCEDURE P;
BEGIN
  A; B; C;
END;

PROCEDURE Q;
BEGIN
  D; E;
END;
```

Geben Sie alle möglichen Abläufe dieses Programmes (in Form der Reihenfolgen der atomaren Anweisungen) an.

Aufgabe 8: (K) Tamagottochi (8 Pkt.)

Das virtuelle Haustier „Tamagottochi“ (entwickelt an der RWTH Aachen) verfügt über verschiedene scheinbar „menschliche“ Züge: Es bekommt Hunger, muss aufs Klo und hat Langeweile. Sein Halter muss jeweils abhelfen, um es glücklich zu machen.

Ihre Aufgabe besteht nun darin, seine Verhaltensweise in einem Diagramm darzustellen. Erörtern Sie dazu, in welchen Zuständen sich unser kleiner Freund jeweils befinden kann, und durch welche Ereignisse diese gewechselt werden. Zu beachten ist dabei:

- Andauernder Hunger führt zum Tod.
- Andauernder Harndrang oder Langeweile veranlassen das Tamagottochi zur Flucht.
- Langeweile kann nur aufkommen, wenn das Tamagottochi weder Hunger hat noch auf die Toilette muss.
- Es ist möglich, dass sich das Tamagottochi gleichzeitig in zwei Teilzuständen befindet. Diese sind dann geeignet zu einem neuen eigenen Zustand zusammenzufassen.

a. Welche sieben Zustände kann das Tamagottochi unter obigen Bedingungen annehmen?

b. Entwerfen Sie ein Zustandsübergangsdiagramm.

Aufgabe 9: (P) Java: Einfacher Thread (8 Pkt.)

a. Betrachten Sie das folgende Java-Programm. Als Eingabeparameter verlangt es eine Integer-Zahl. Wie hängt diese Zahl mit der Ausgabe zusammen? Welche Ausgabe erwarten Sie für verschiedene Integer-Eingaben (zum Beispiel 1, 2, 100 oder 10000)?

b. Kompilieren Sie das Programm, und führen Sie es auf verschiedenen, Ihnen zugänglichen Systemen aus. Protokollieren und kommentieren Sie das Ergebnis.

```
public class SimplerThread extends Thread {

        String msg ;
        int cycles ;

        SimplerThread(String m, int c) {
                msg = m ;
                cycles = c ;
        }

        // overrides run() in Thread class to define object's
        // behavior.
        public void run() {
                for (int i = 0 ; i < cycles ; i++) {
                        System.out.println(msg) ;
                }
        }

        // command-line arguments are integers C.
        // builds and starts two threads of type SimplerThread.
        // continues for C cycles.

        public static void main(String[] args) {

                if (args.length < 1) {
                        System.out.println("Arguments are:") ;
                        System.out.println("  cycles") ;
                        System.exit(-1) ;
                }

```

```
                    int C = Integer.parseInt(args[0]) ;

                    SimplerThread t1 =
                            new SimplerThread("ping---->", C) ;
                    SimplerThread t2 =
                            new SimplerThread("<----pong", C) ;

                    t1.start() ;
                    t2.start() ;
            }
    }
```

Aufgabe 10: (P) Java: Wartezustand (Sleep) (4 Pkt.)

Ergänzen Sie den folgenden Java-Programmrahmen, sodass bei seiner Ausführung ein Fenster mit zwei Buttons angezeigt wird: Ein Button `Hello` soll den String „`Hello World`“ auf der Konsole ausgeben, ein weiterer Button `Sleep` soll das Java-Programm für einige Zeit (beispielsweise fünf Sekunden) mit Hilfe von

```
Thread.currentThread().sleep
```

in den Wartezustand versetzen. Beobachten Sie, was passiert, wenn Sie während der Wartezeit den `Hello`-Button drücken.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class Hang extends JFrame
{
    public Hang()
    {   JButton b1 = new JButton( "Sleep" );
        JButton b2 = new JButton( "Hello" );

        ////////////////////////////////
        //                            //
        // Zu ergänzender Programmteil //
        //                            //
        ////////////////////////////////

        getContentPane().setLayout( new FlowLayout() );
        getContentPane().add( b1 );
        getContentPane().add( b2 );
        pack();
        show();
    }

    public static void main( String[] args )
    {
        new Hang();
    }
}
```