Vorlesung Betriebssysteme I

UNIVERSITÄT SIEGEN

Fachbereich 12, Elektrotechnik und Informatik
Betriebssysteme / verteilte Systeme
Prof. Roland Wismüller

Beispielklausur

Name: **Mustermann**

Vorname: **Muster**

Matrikelnummer: **012345678**

Hinweise:

- Die Bearbeitungszeit beträgt **60 Minuten**, es sind **keinerlei Hilfsmittel** zugelassen.
- Tragen Sie auf jedes Blatt sofort **Name und Matrikelnummer** ein!
- Die Aufgaben 2a), 2c) und 6b) sind **auf der Angabe** zu lösen!
- Die **Angabe** ist am Ende mit **abzugeben**!
- Antworten Sie **kurz** und **präzise**, kennzeichnen Sie ungültige Lösungen **deutlich**!
- Zu den „Programmier"-Aufgaben:
  - es kommt nicht auf syntaktische Feinheiten an!
  - nicht in jede Leerzeile gehört zwangsläufig eine Anweisung!
- Die maximale Punktezahl beträgt **60 Punkte**. Zum Bestehen sind **30 Punkte ausreichend**.

Aufgabe 1: Prozesse und Threads (9 Punkte)

a) (*2 Punkte*) Geben Sie die **drei** wichtigsten Zustände eines Prozesses (oder Threads) an! In welchem der drei Zustände befindet sich ein **neu erzeugter** Prozeß zunächst?

b) (*3 Punkte*) Welche Zustandsübergänge

(i) können durch einen Interrupt (z.B. Uhr, E/A-Gerät) ausgelöst werden?

(ii) können beim **aufrufenden Prozeß/Thread** durch einen Systemaufruf ausgelöst werden?

(iii) werden durch den (präemptiven) Scheduler des Betriebssystems verursacht?

c) (*4 Punkte*) Geben Sie die einzelnen Schritte an, die das Betriebssystem ausführt, um einen **Prozeß**wechsel durchzuführen!

Ein **Thread**wechsel läuft prinzipiell genauso ab, beim Threadwechsel innerhalb **desselben** Prozesses fehlt aber im Vergleich zum Prozeßwechsel ein wesentlicher Schritt. Welcher?

Aufgabe 2: Synchronisation (14 Punkte)

a) (*6 Punkte*) Die folgende Realisierung eines binären Semaphors durch einen Monitor soll vervollständigt werden. Durch den Aufruf von `Lock()` wird das Semaphor gesperrt und durch `Unlock()` wieder freigegeben. Ein Thread kann somit für einen kritischen Bereich durch Einklammern in `Lock()` und `Unlock()` den wechselseitigen Ausschluß sicherstellen.

Falls das Semaphor bereits belegt ist (`locked = true`), soll `Lock()` den aufrufenden Thread so lange blockieren, bis ein anderer Thread `Unlock()` ausführt, d.h. `locked = false` gilt. Realsieren Sie die dazu notwendige Synchronisation über eine **Bedingungsvariable**!

```
monitor BinSema
    condition  unlock;
    boolean    locked;

    procedure  Lock()
    begin
        .....
        .....
        .....
        locked = true;
    end;
    ⋮
```

```
    ⋮

    procedure  Unlock()
    begin
        locked = false;
        .....
        .....
    end;

    locked = false;    // Initialisierung
end monitor;
```

b) (*2 Punkte*) Begründen Sie, warum zur Lösung von Aufgabe 2a) in der Monitor-Prozedur **`Lock()`** kein *Busy Waiting* verwendet werden kann! (Tip: was würde passieren, wenn in **`Lock()`** die Zeile

```
while (locked);    // warte in der Schleife bis locked==false
```

eingefügt würde?)

c) (*6 Punkte*) Gegeben ist der folgende Code für ein Erzeuger-/Verbraucher-Problem mit **unbeschränktem** Puffer. Die Prozedur **`insert_item()`** fügt ein Element in den Puffer ein, die Funktion **`remove_item()`** entfernt das erste Element und gibt seinen Inhalt als Ergebnis zurück. Ergänzen Sie den Code um die notwendige Synchronisation, um sicherzustellen, daß

- die Aufrufe von **`insert_item()`** und **`remove_item()`** unter wechselseitigem Ausschluß stehen,
- Thread 2 beim Versuch, ein Element aus einem **leeren Puffer zu entfernen**, so lange blockiert wird, bis der Puffer nicht mehr leer ist.

Hinweise: Verwenden Sie zur Lösung zwei **Semaphore** und, falls erforderlich, weitere Hilfsvariablen. Lösungen mit *Busy Waiting* werden **nicht** akzeptiert! Eine Synchronisation, um das Schreiben auf einen vollen Puffer zu verhindern, soll **nicht** realisiert werden.

Semaphore und globale Variablen

.....

.....

Thread 1

```
while (true)  {
    item = produce();  // Produziere item
    .....
    .....
    insert_item(item);  // Einfügen in Puffer
    .....
    .....
    .....
}
```

Thread 2

```
while (true)  {
    .....
    .....
    .....
    item = remove_item();  // Entfernen aus Puffer
    .....
    .....
    consume(item);  // Verarbeite item
}
```

Aufgabe 3: Verklemmungen (9 Punkte)

a) (*4 Punkte*) Gegeben sind die folgenden drei Prozesse, die in der angegebenen Reihenfolge Betriebsmittel belegen und wieder freigeben:

Prozeß 1	**Prozeß 2**	**Prozeß 3**

P(Drucker);
P(Festplatte);
// drucke von Plate
V(Festplatte);
P(CD_ROM); // (*)
// drucke von CD-ROM
V(CD_ROM);
V(Drucker);
	P(Festplatte);
	P(CD_ROM); // (*)
	// kopiere von CD auf Platte
	V(CD_ROM);
	V(Festplatte);
		P(CD_ROM);
		P(Drucker); // (*)
		// drucke von CD-ROM
		V(Drucker);
		V(CD_ROM);

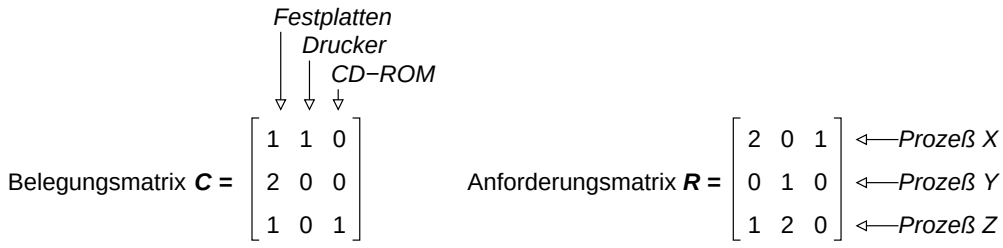
Die (ununterbrechbaren) Betriebsmittel sind jeweils nur einmal vorhanden.

Betrachten Sie den Zustand, bei dem sich die Prozesse an den mit (*) markierten Stellen befinden (die P-Operationen sind aufgerufen, aber noch nicht abgeschlossen).

(i) Zeichnen Sie für diesen Zustand einen Belegungs-Anforderungs-Graphen!

(ii) Liegt ein Deadlock vor? Falls ja, zwischen welchen Prozessen? Kurze Begründung!

b) (*5 Punkte*) In einem System stehen 5 Festplatten, 2 Drucker und 1 CD-Laufwerk als (ununterbrechbare) Betriebsmittel zur Verfügung. Betrachten Sie den folgenden Belegungszustand:



Die Anforderungsmatrix R beschreibt hier die **maximalen** zukünftigen Forderungen.

(i) Geben Sie den Ressourcenvektor E und den Ressourcenrestvektor A an.

(ii) Führen Sie den Bankiers-Algorithmus aus! Geben Sie an, ob Prozesse zu Ende laufen können und wenn ja, welche!

(iii) Ist der Zustand sicher? Was bedeutet dieses Ergebnis genau?

Aufgabe 4: *Scheduling* (11 Punkte)

Drei Prozesse treffen zu den in der Tabelle angegebenen Zeiten in der Bereitliste eines Schedulers ein. Von jedem Prozeß ist die Bedienzeit (= benötigte Rechenzeit) bekannt. Zusätzlich hat jeder Prozeß eine Priorität (0 stellt die höchste Priorität dar).

Prozeß	Ankunftszeit	Bedienzeit	Priorität
A	0	7	2
B	2	5	1
C	5	3	0

Die Prozesse benutzen nur die CPU und werden nie durch E/A oder sonstige Gründe blockiert. Der Scheduler entscheidet online, d.h. nur aufgrund der zum Scheduling-Zeitpunkt bereits vorliegenden Prozesse.

Betrachten Sie die folgenden Scheduler-Strategien:

a) *Shortest Job First* (SJF), nicht-präemptiv,

b) *Round Robin* (RR) mit einer Zeitscheibenlänge (Quantum) von 3,

c) *Prioritätsbasiertes Scheduling* (HPF), präemptiv.

Zeichnen Sie für jede der Strategien den zeitlichen Ablauf der Prozeßausführung als Gantt-Diagramm!

Aufgabe 5: Schutz (2 Punkte)

Definieren Sie den Begriff **Zugriffskontroll-Liste (ACL)**! Welche Informationen sind in einer solchen Liste gespeichert?

Aufgabe 6: Speicherverwaltung (15 Punkte)

a) (*9 Punkte*) Sowohl Segmentierung als auch Paging basieren auf einer Adreßübersetzung von virtuellen auf physische Adressen. Beschreiben Sie **exakt** die wesentlichen **Unterschiede** zwischen Segmentierung und Paging! Gehen Sie dabei insbesondere auf den **Aufbau des virtuellen und physischen Adreßraums**, den Unterschied zwischen **Segmenten** und **Seiten**, sowie die eigentliche **Adreßübersetzung** ein!

Achtung: Segmentierung wird in der aktuellen Vorlesung nicht mehr behandelt!

b) (*6 Punkte*) Einem Prozeß, dessen virtueller Adreßraum 5 Seiten umfaßt, wurden vom Betriebssystem **4 Kacheln** im Hautpspeicher zugeteilt. Zu einem bestimmten Zeitpunkt hat das Betriebssystem folgende Informationen (Seitentabelle + Hilfsinformation) über die Seiten des Prozesses:

	Seitentabelle				Hilfsinformation	
Seite	Present-Bit	Kachel	R-Bit	M-Bit	Ladezeit	Zeit des letzten Zugriffs
0	1	2	0	1	30	35
1	1	3	1	0	5	45
2	1	1	0	0	10	20
3	0	-	-	-	-	-
4	1	0	0	0	25	40

Der Prozeß führt nun

- zum **Zeitpunkt 50** einen **Schreib**zugriff auf **Seite 2** und danach
- zum **Zeitpunkt 55** einen **Lese**zugriff auf **Seite 3** durch.

Geben Sie unten an, wie Seitentabelle und Hilfsinformation nach dieser Folge von Zugriffen aussehen, wenn

(i) LRU (*Least Recently Used*)

(ii) *Second Chance*

als Seitenersetzungsalgorithmus verwendet wird!

(i) LRU

	Seitentabelle				Hilfsinformation	
Seite	Present-Bit	Kachel	R-Bit	M-Bit	Ladezeit	Letzter Zugriff
0						
1						
2						
3						
4						

(ii) *Second Chance*

	Seitentabelle				Hilfsinformation	
Seite	Present-Bit	Kachel	R-Bit	M-Bit	Ladezeit	Letzter Zugriff
0						
1						
2						
3						
4						