# Lösungsskizzen zur Abschlussklausur

# Betriebssysteme (BTS)

*7. Juli 2011*

**Name:** _____

**Vorname:** _____

**Matrikelnummer:** _____

**Studiengang:** _____

**Hinweise:**

- Tragen Sie zuerst auf allen Blättern (einschließlich des Deckblattes) Ihren *Namen*, Ihren *Vornamen* und Ihre *Matrikelnummer* ein. Lösungen ohne diese Angaben können nicht gewertet werden.
- Schreiben Sie die Lösungen jeder *Teil*aufgabe auf das jeweils vorbereitete Blatt. Sie können auch die leeren Blätter am Ende der Heftung nutzen. In diesem Fall ist ein Verweis notwendig. Eigenes Papier darf nicht verwendet werden.
- Legen Sie bitte Ihren *Lichtbildausweis* und Ihren *Studentenausweis* bereit.
- Als *Hilfsmittel* ist ein Taschenrechner zugelassen.
- Mit Bleistift oder Rotstift geschriebene Ergebnisse werden *nicht* gewertet.
- Die Bearbeitungszeit dieses Teils der Abschlussklausur beträgt *60 Minuten*.
- Stellen Sie sicher, dass Ihr Mobiltelefon ausgeschaltet ist. Klingelnde Mobiltelefone werden als Täuschungsversuch angesehen und der/die entsprechende Student/in wird von der weiteren Teilnahme an der Klausur **ausgeschlossen**!

## Bewertung:

| 1) | 2) | 3) | 4) | 5) | 6) | 7) | 8) | 9) | **Σ** | Note |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

*Lösungsskizzen zur Abschlussklausur*

# Betriebssysteme (BTS)

7.7.2011

M.Sc. Christian Baun

## Aufgabe 1 (6+2 Punkte)

a) Der Speicher eines Computersystems wird in die drei Kategorien **Primärspeicher**, **Sekundärspeicher** und **Tertiärspeicher** unterschieden. Beschreiben Sie die qualitativen Merkmale dieser Speichersorten und ihre Ausprägungen. Zeichnen Sie dazu ein Diagramm. Nennen Sie auch mindestens zwei Beispiele pro Kategorie.

b) Der **Tertiärspeicher** wird ebenfalls in zwei Kategorien unterschieden. Benennen Sie diese beiden Kategorien und beschreiben Sie diese.

## Aufgabe 2 (1+2 Punkte)

a) Es existieren zwei grundsätzliche Konzepte, um **Schreibzugriffe auf Cache** durchzuführen. Welche beiden Konzepte sind das?

b) Beschreiben Sie die beiden Konzepte. Gehen Sie auf die Unterschiede, Vor- und Nachteile ein.

## Aufgabe 3 (5 Punkte)

Zeichnen Sie das **5-Zustands-Prozessmodell** mit seinen Zuständen und allen Prozessübergängen.

## Aufgabe 4 (3 Punkte)

Nennen Sie die drei Arten von **Prozesskontextinformation**, die das Betriebssystem speichert und beschreiben Sie deren Inhalt.

## Aufgabe 5 (2+2 Punkte)

a) Was ist ein **Dispatcher** und was sind seine Aufgaben?

b) Was ist ein **Scheduler** und was sind seine Aufgaben?

## Aufgabe 6 (4+1+1+1 Punkte)

a) Beschreiben Sie den Ablauf der **verbindungsorientierten Kommunikation mit Sockets**. Gehen Sie dabei besonders auf die nötigen Operationen ein. Zeichnen Sie dazu ein Diagramm.

b) Bei welcher Art der Interprozesskommunikation müssen die beteiligten **Prozesse verwandt** sein?

c) Bei welcher Art der Interprozesskommunikation müssen die Entwickler sich selbst um die **Synchronisierung der Schreibzugriffe** kümmern?

d) Bei welchen **Pipes** kann die Interprozesskommunikation **bidirektional** erfolgen und bei welchen Pipes kann die Interprozesskommunikation nur **unidirektional** erfolgen?

## Aufgabe 7 (6+6+6 Punkte)

Auf einem Einprozessorrechner sollen sieben Prozesse verarbeitet werden.

| Prozess | CPU-Laufzeit (ms) | Ankunftszeit (ms) |
|---|---|---|
| A | 5 | 0 |
| B | 7 | 3 |
| C | 2 | 5 |
| D | 6 | 7 |
| E | 1 | 10 |
| F | 5 | 18 |
| G | 4 | 24 |

a) Skizzieren Sie die Ausführungsreihenfolge der Prozesse mit einem Gantt-Diagramm (Zeitleiste) für **Round Robin** (Zeitquantum $q = 1$ ms), **Longest Remaining Time First** (LRTF) und **Shortest Remaining Time First** (SRTF).

b) Berechnen Sie die **mittleren Laufzeiten** der Prozesse.

c) Berechnen Sie die **mittleren Wartezeiten** der Prozesse.

## Aufgabe 8 (2+2+1 Punkte)

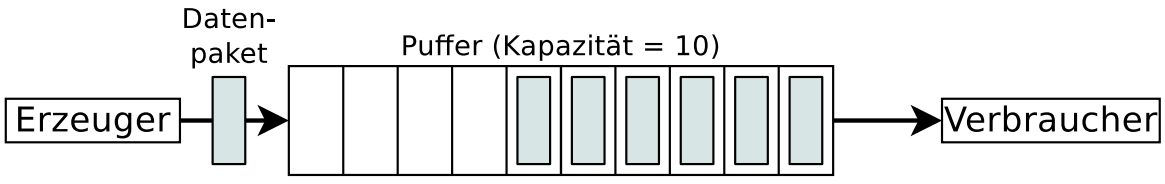
a) Was ist eine **Semaphore** und was ist ihr Einsatzzweck?

b) Welche beiden **Operationen** werden bei Semaphoren verwendet? Gesucht sind die Bezeichnungen und eine (kurze) Beschreibung der Funktionsweise.

c) Was ist der Unterschied zwischen **Semaphoren** und **Sperren**?

## Aufgabe 9 (7 Punkte)

- **Erzeuger/Verbraucher-Szenario**.
  - Ein Erzeuger soll Daten an einen Verbraucher schicken
  - Ein endlicher Zwischenspeicher (Puffer) soll die Wartezeiten des Verbrauchers minimieren
  - Daten werden vom Erzeuger in den Puffer gelegt und vom Verbraucher aus diesem entfernt
  - Gegenseitiger Ausschluss ist notwendig, um Inkonsistenzen zu vermeiden
  - Ist der Puffer voll, muss der Erzeuger blockieren
  - Ist der Puffer leer, muss der Verbraucher blockieren




- Aufgabe: **Synchronisieren Sie die beiden Prozesse**, indem Sie die notwendigen **Semaphoren** deklarieren, diese mit Startwerten versehen und Semaphor-Operationen einfügen.

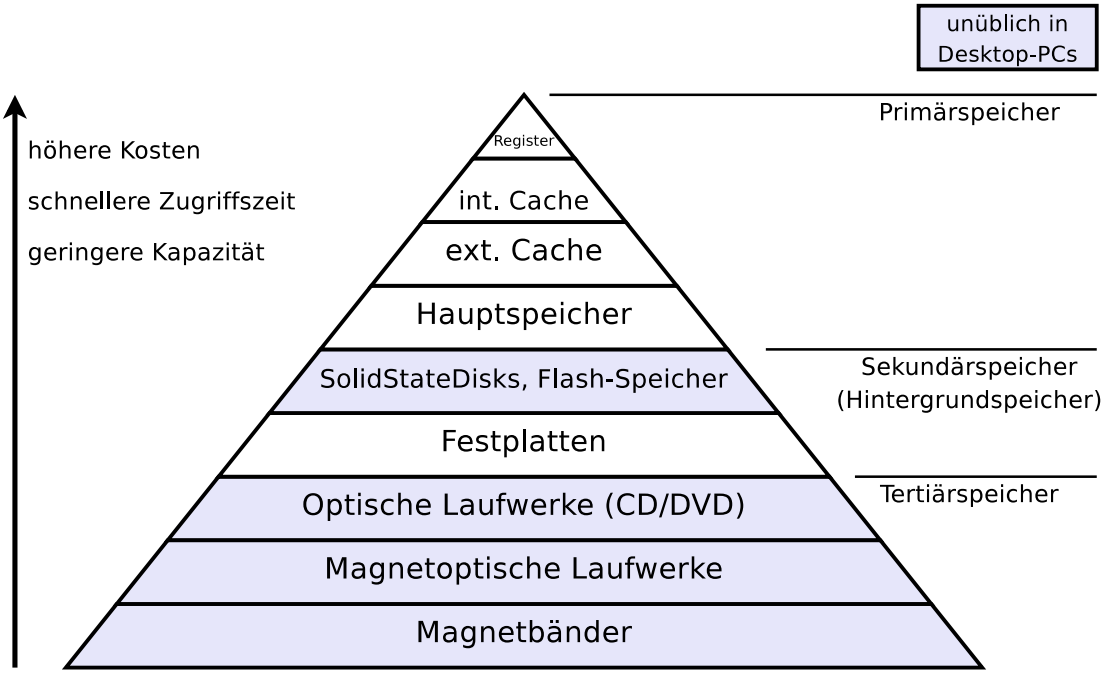
Name: Vorname: Matr.Nr.:

# Aufgabe 1)

Punkte: .....

a)

- Auf **Primärspeicher** kann der Hauptprozessor direkt zugreifen.
- Bei **Sekundärspeicher** handelt es sich um Hintergrundspeicher, der über einen Controller angesprochen wird.
- Der **Tertiärspeicher** ist nicht dauerhaft verfügbar, oder nur über ein Laufwerk mit dem Computer verbunden.




- Primärspeicher und Sekundärspeicher werden auch als **Onlinespeicher** bezeichnet, da sie eine feste Verbindung zum Computer und dadurch geringe Zugriffszeiten auf die Daten haben

b)

- **Nearlinespeicher**: Werden automatisch und ohne menschliches Zutun dem System bereitgestellt (z.B. Band-Library)
- **Offlinespeicher**: Medien werden in Schränken oder Lagerräumen aufbewahrt und müssen von Hand in das System integriert werden

Name: Vorname: Matr.Nr.:

# Aufgabe 2)

Punkte: .....

a) **Write-Back** und **Write-Through**

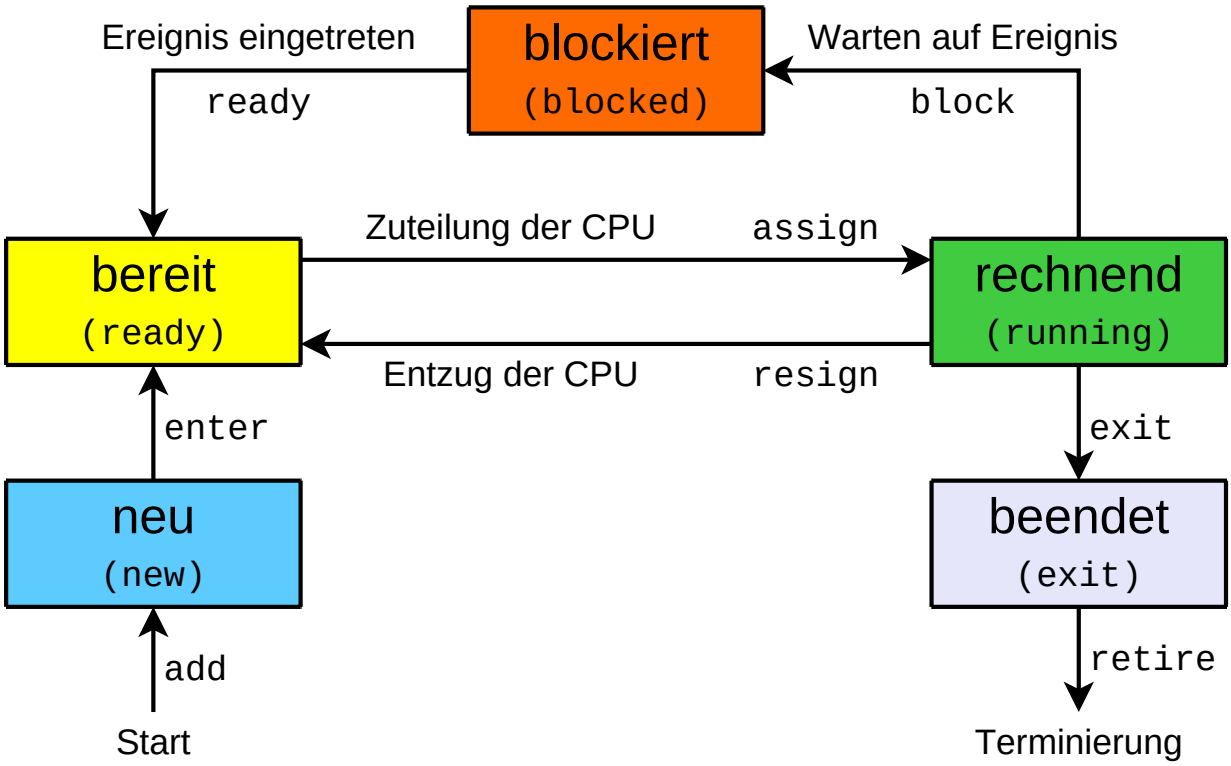
b)

- **Write-Back**: Schreibzugriffe werden nicht direkt an die nächst tiefere Speicherebene weitergegeben. Inkonsistenzen zwischen den Daten im Cache und auf dem zu cachenden Speicher entstehen. Die Daten werden erst zurückgeschrieben, wenn der betreffende Datenblock aus dem Cache verdrängt wird.
  - **Vorteil**: Höhere System-Geschwindigkeit
  - **Nachteil**: Daten gehen beim Systemausfall verloren
- **Write-Through**: Schreibzugriffe werden sofort an die nächst tiefere Speicherebene weitergegeben.
  - **Vorteil**: Datenkonsistenz ist gesichert
  - **Nachteil**: Geringere System-Geschwindigkeit

Name: Vorname: Matr.Nr.:

# Aufgabe 3)

Punkte: .....



Name: Vorname: Matr.Nr.:

# Aufgabe 4)

Punkte: .....

- Der **Benutzerkontext** sind die Daten des Prozesses im zugewiesenen Adressraum.
  - Jedem Prozess wird ein eigener Adressraum zugeordnet, in dem sich u.a. das ausführbare Programm, also die Maschinenbefehle, und Prozessdaten befinden.
    * Der Adressraum ist eine Liste von Speicherstellen, in denen der Prozess lesen und schreiben darf.
    * Der Adressraum ist eine Abstraktion des physischen Speichers und unabhängig von der verwendeten Speichertechnologie $\Longrightarrow$ Virtueller Speicher.
- Der **Hardwarekontext** umfasst die Inhalte der Register in der CPU zum Zeitpunkt der Prozess-Ausführung und die Seitentabelle.
  - Die Register, deren Inhalt bei einem Kontextwechsel gesichert werden muss, sind: Befehlszähler, Stack-Pointer, Basis- und Grenzregister, Akkumulator („General Purpose"-Register), Integer-Register, Floating-Point-Register.
  - Diese Informationen sind wichtig, wenn ein Prozess im Rahmen des Multitaskings bei einem Kontextwechsel durch einen anderen Prozess unterbrochen wird.
- Der **Systemkontext** sind die Informationen, die das Betriebssystem über einen Prozess speichert.
  - Der **Systemkontext** sind die Informationen, die das Betriebssystem über einen Prozess speichert
  - Beispiele sind: Eintrag in der Prozesstabelle, Prozessnummer (PID), Prozesszustand, Information über Eltern- oder Kindprozesse, Prioritäten, Zugriffsrechte auf Ressourcen, Quotas, Laufzeit, geöffnete Dateien, tugeordnete Geräte.

Die Informationen im Hardwarekontext und Systemkontext werden vom Betriebssystem im **Prozesskontrollblock** verwaltet.

Name: Vorname: Matr.Nr.:

# Aufgabe 5)

Punkte: .....

a)

- Der **Dispatcher** (Prozessumschalter) führt die Zustandsübergänge der Prozesse durch.
- Beim Prozesswechsel entzieht der Dispatcher dem derzeit aktiven, rechnenden Prozess die CPU und teilt sie dem Prozess zu, der in der Warteschlange an erster Stelle steht.
- Bei Übergängen zwischen den Zuständen `bereit` und `blockiert` werden vom Dispatcher die entsprechenden Prozesskontrollblöcke aus den Zustandslisten entfernt und entsprechend neu eingefügt.

b)

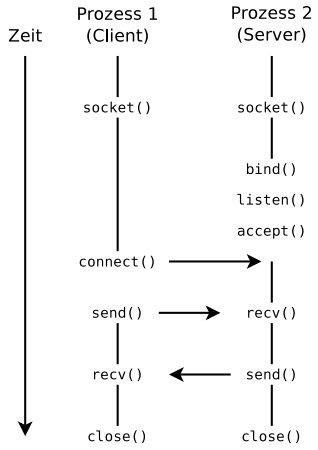
- Der Scheduler eines Betriebssystems legt Ausführungsreihenfolge der Prozesse im Zustand `bereit` fest.
- Die Entscheidung, welcher Prozess wann an der Reihe ist, ist vom Scheduling-Algorithmus abhängig.

Name: Vorname: Matr.Nr.:

# Aufgabe 6)

Punkte: .....

a)




- **Client**
  - Socket erstellen (`socket`)
  - Client mit Server-Socket verbinden (`connect`)
  - Daten senden (`send`) und empfangen (`recv`)
  - Socket schließen (`close`)
- **Server**
  - Socket erstellen (`socket`)
  - Socket an einen Port binden (`bind`)
  - Socket empfangsbereit machen (`listen`)
    - Richtete eine Warteschlange für Verbindungen mit Clients ein
  - Server akzeptiert Verbindungsanforderung (`accept`)
  - Daten senden (`send`) und empfangen (`recv`)
  - Socket schließen (`close`)

b) Bei anonymen Pipes müssen die beteiligten Prozesse verwandt sein.

c) Bei gemeinsamem Speicher müssen sich die Entwickler sich selbst um die Synchronisierung der Schreibzugriffe kümmern.

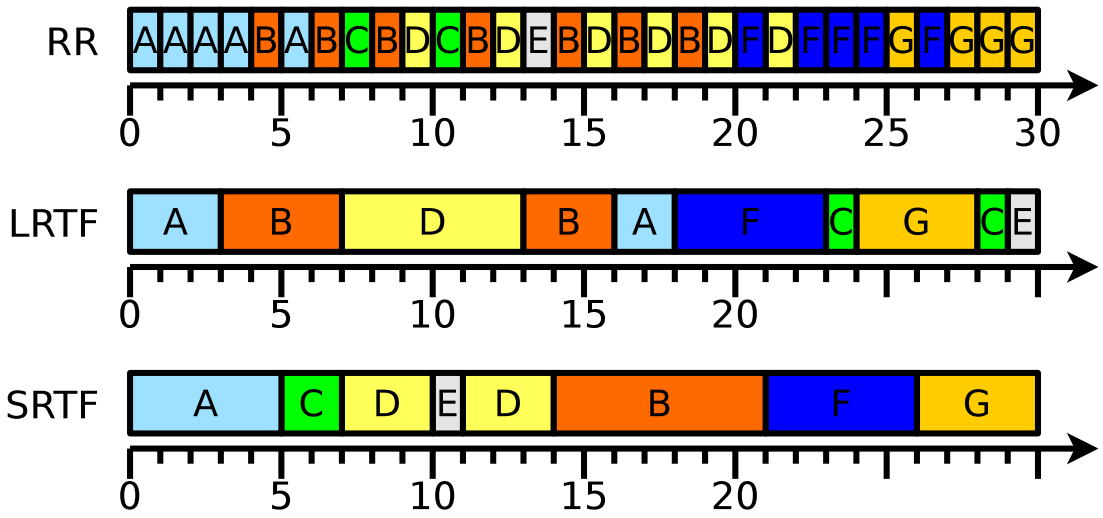
d) Anonyme Pipes kann man nur in eine Richtung verwenden ($\Longrightarrow$ unidirektional). Benannte Pipes ermöglichen Vollduplexbetrieb zwischen Prozessen ($\Longrightarrow$ bidirektional).

Name: Vorname: Matr.Nr.:

# Aufgabe 7)

Punkte: .....

a)



Name: Vorname: Matr.Nr.:

# Aufgabe 7)

Punkte: .....

b) Laufzeit (Turnaround Time) der Prozesse

| | **A** | **B** | **C** | **D** | **E** | **F** | **G** |
|---|---|---|---|---|---|---|---|
| Round Robin | 6 | 16 | 6 | 15 | 4 | 9 | 6 |
| Longest Remaining Time First | 18 | 13 | 24 | 6 | 20 | 5 | 4 |
| Shortest Remaining Time First | 5 | 18 | 2 | 7 | 1 | 8 | 6 |

| | | | |
|---|---|---|---|
| Round Robin | $\frac{6+16+6+15+4+9+6}{7}$ | = | $8,86$ ms |
| Longest Remaining Time First | $\frac{18+13+24+6+20+5+4}{7}$ | = | $12,86$ ms |
| Shortest Remaining Time First | $\frac{5+18+2+7+1+8+6}{7}$ | = | $6,71$ ms |

c) Wartezeit der Prozesse – Zeit in der `bereit`-Liste

| | **A** | **B** | **C** | **D** | **E** | **F** | **G** |
|---|---|---|---|---|---|---|---|
| Round Robin | 1 | 9 | 4 | 9 | 3 | 4 | 2 |
| Longest Remaining Time First | 13 | 6 | 22 | 0 | 19 | 0 | 0 |
| Shortest Remaining Time First | 0 | 11 | 0 | 1 | 0 | 3 | 2 |

| | | | |
|---|---|---|---|
| Round Robin | $\frac{1+9+4+9+3+4+2}{7}$ | = | $4,57$ ms |
| Longest Remaining Time First | $\frac{13+6+22+0+19+0+0}{7}$ | = | $8,57$ ms |
| Shortest Remaining Time First | $\frac{0+11+0+1+0+3+2}{7}$ | = | $2,43$ ms |

Name: Vorname: Matr.Nr.:

# Aufgabe 8)

Punkte: .....

a) Zur Sicherung (Sperrung) kritischer Abschnitte können außer den bekannten Sperren auch **Semaphoren** eingesetzt werden. Ein Semaphor ist eine Zählersperre **S**. Die Zählersperre ist eine ganzzahlige, nichtnegative Zählvariable.

b) Es existieren die beiden Operationen **P(S)** und **V(S)**.

- **V** kommt vom holländischen *verhogen* = erhöhen
  $\Longrightarrow$ Die Zählersperre wird um 1 erhöht.
- **P** kommt vom holländischen *proberen* = versuchen (zu verringern)
  $\Longrightarrow$ Es wird versucht, die Zählersperre um 1 zu verringern.

c) Im Gegensatz zu Semaphoren können Sperren immer nur einem Prozess das Betreten des kritischen Abschnitts erlauben

Name: Vorname: Matr.Nr.:

# Aufgabe 9)

Punkte: .....

```
#define N         8              // Plätze im Puffer
typedef int semaphore;           // Semaphore sind von Typ Integer
semaphore voll  = 0;             // zählt belegte Plätze im Puffer
semaphore leer  = N;             // zählt freie Plätze im Puffer
semaphore mutex = 1;             // steuert Zugriff auf kritische Bereiche

void erzeuger (void) {

  int daten;

  while (TRUE) {                 // Endlosschleife
    erzeugeDatenpaket(daten);    // erzeuge Datenpaket
    P(leer);                     // Zähler "leere Plätze" erniedrigen
    P(mutex);                    // in kritischen Bereich eintreten
    einfuegenDatenpaket(daten); // Datenpaket in Puffer schreiben
    V(mutex);                    // kritischen Bereich verlassen
    V(voll);                     // Zähler für volle Plätze erhöhen
  }
}

void verbraucher (void) {

  int daten;

  while (TRUE) {                  // Endlosschleife
    P(voll);                      // Zähler "volle Plätze" erniedrigen
    P(mutex);                     // in kritischen Bereich eintreten
    entferneDatenpaket(daten);    // Datenpaket aus Puffer holen
    V(mutex);                     // kritischen Bereich verlassen
    V(leer);                      // Zähler für leere Plätze erhöhen
    verbraucheDatenpaket(daten); // Datenpaket nutzen
  }
}
```