# BETRIEBSSYSTEME

- Übung 1 -

## Aufgabe 1:

Handelt es sich bei den folgenden Aufgaben um Stapelverarbeitung oder um interaktive Programme? Begründen Sie Ihre Antwort.

**Interaktiv:**

1. Software zum Ausfüllen einer Steuererklärung – die Steuererklärung muss vom Anwender ausgefüllt werden
2. ein Flugsimulator – dient zum Simulieren eines Fluges, der Anwender steuert den Flug
3. Textverarbeitung – der Inhalt einer Textdatei wird von einem Anwender mittels einer Software bearbeitet

**Stapelverarbeitung:**

1. Automatisches Generieren von Lohnzetteln für Arbeitnehmer basierend auf einer Datenbank – die Lohnzettel werden automatisch vom System erzeugt, menschliches Zutun ist nicht notwendig
2. Erstellen monatlicher Kontoauszüge – Kontoauszüge beinhalten Informationen über den Stand eines Kontos, die automatisiert erzeugt werden, um dem Kontoinhaber Veränderungen des Kontos in schriftlicher Form zu übermitteln
3. Berechnen der Zahl π auf 10.000 Dezimalstellen – ein Rechner (Computer, Taschenrechner etc.) errechnet automatisch ohne Zwischeneingabe eines Anwenders

Nennen Sie jeweils zwei weitere Beispiele für Stapelverarbeitung und für interaktive Programme.

**Interaktiv:** Videospiele, Bildbearbeitung
**Stapelverarbeitung:** Drucken, Internetspeedtest

## Aufgabe 2:

Nennen Sie die beiden Hauptaufgaben eines Betriebssystems.

1. Verwaltung der Ressourcen
2. Erweiterung der Hardware

## Aufgabe 3:

Erläutern Sie den Unterschied zwischen einem System- und einem Anwendungsprogramm.

**Systemprogramm:** Sie stellt eine Verbindung zur Hardware her, verwaltet ihre jeweiligen Ressourcen und umfasst eine Benutzeroberfläche für Anwender.
**Anwendungsprogramm**: Hauptsächlich externe Programme, die vom Anwender gesteuert werden und auf die Ressourcen der Hardware mittels Systemprogrammen zugreifen.

## Aufgabe 4:

Inwiefern unterscheiden sich monolithische Betriebssysteme von solchen mit einem Mikrokernel?

**Monolithische BS:** Integrieren sämtliche Systemfunktionen, auch nicht essentielle, in den Kernel.
**Mikrokernel BS:** Minimieren die Größe und Funktionalitäten des Kernels. Nur essentielle Features werden integriert, restliche Funktionen werden als Service in den Userspace ausgelagert.

## Aufgabe 5:

Was ist Multiprogramming und worin liegt seine Bedeutung?

Der physikalische Speicher wird partitioniert, jeder Prozess/Job erhält eine Partition. Muss ein Prozess auf I/O warten, wird dieser Prozess/Job unterbrochen, damit die CPU mit einem anderen Prozess/Job in der Zwischenzeit weiterarbeitet. Die CPU kann somit zu 100% ausgelastet werden.

- Übung 2 -

## Aufgabe 1:

Zählen Sie einige Unterschiede zwischen einem Betriebssystem für Server und einem Betriebssystem für Chipkarten auf. Welche Gemeinsamkeiten dürften solche Betriebssysteme haben?

**Server-BS:** hohe Rechenleistung, hohe Speicherkapazität und Ausfallsicherheit

**Chipkarten-BS:** extrem geringe Rechenleistung, geringe Speicherkapazität und Laufzeit

## Aufgabe 2:

Überlegen Sie: Welche Auswirkungen würde der Betrieb eines Chipkarten-Betriebssystems auf einem Server haben und welche Auswirkungen würde der Betrieb eines Betriebssystems für PCs auf einem Server haben?

**Chipkarten-BS auf Server:** bis auf die schnelle Berechnung von Verschlüsselungen könnte das Chipkarten-BS die Gesamtleistung der Hardware eines Servers nicht effizient nutzen, sie ist nicht auf die Client/Server-Architektur eines Server-BS ausgelegt.

**PC-BS auf Server:** PC-BS besitzen neben einer CPU mitunter auch eine Grafikkarte als Hardwarekomponente, die zum schnellen Rendern von visuellen Elementen eingesetzt wird. Server besitzen in den meisten Fällen eine hohe Menge CPUs und keine bis wenige Grafikkarten. Ein PC-BS wäre dazu gezwungen das Rendern mittels der CPUs eines Servers durchzuführen.

## Aufgabe 3:

Erklären Sie den Begriff `Spooling'. Welche Bedeutung hat Spooling heute?

**Spooling damals:** Parallel zur Aufführung eines Jobs/Prozess werden wartende Jobs von einem (langsameren) Medium eingelesen und auf ein schnelleres zwischengelagert, um wieder eingelesen und ausgeführt zu werden. Freier Speicher auf einer Festplatte kann somit zum Zwischenspeicher genutzt werden. Führt zu Perfomancegewinn.

**Spooling heute:** Bei Druckaufträgen werden die einzelnen Aufträge auf freien Speicher auf einem Medium gespeichert. Anschließend können sie nacheinander abgearbeitet werden.

## Aufgabe 4:

Grenzen Sie die Begriffe `Multiprogramming', `Spooling' und `Timesharing' voneinander ab. Was unterscheidet diese Konzepte?

**Multiprogramming:** Teilt physikalischen Speicher auf, jeder Prozess/Job erhält dadurch eine Partition. Dadurch arbeitet die CPU mit Prozessen/Jobs weiter, die nicht auf I/O warten. Die CPU kann somit voll ausgelastet werden.

**Timesharing (Verbesserung des Multiprogramming-Konzepts):** Mehrere Anwender können gleichzeitig mit einem System arbeiten, das System teilt den Prozessen der Anwender Rechenzeit zu.

**Spooling:** Parallel zur Aufführung eines Jobs/Prozess werden wartende Jobs von einem (langsameren) Medium eingelesen und auf ein schnelleres zwischengelagert, um wieder eingelesen und ausgeführt zu werden. Freier Speicher auf einer Festplatte kann somit zum Zwischenspeicher genutzt werden. Führt zu Perfomancegewinn.

- Übung 3 & 4 -

## Aufgabe 1:

Ein Betriebssystem läuft auf einem Server mit vier Prozessoren. Während des Betriebs wurde gemessen, dass die drei permanent laufenden Prozesse jeweils 10% ihrer Zeit auf Ein-/Ausgabe warten.
Wie hoch ist die zu erwartende CPU-Auslastung?

**Formel:** 1 – p^n-CPUs

**Zeit für I/O:** p = 10% = 0,1

**Auslastung von drei CPUs:** 1 – 0,1^3 = 1 – 0,001 = 0,999 = **99,9%**

Wie verändert sich die CPU-Auslastung, wenn die drei Prozesse jeweils 1% bzw. 20% ihrer Zeit auf Ein-/Ausgabe warten?

**Formel:** 1 – p^n-CPUs

**Zeit für I/O:** p1 = 1% = 0,01 p2 = 20% = 0,2

**Auslastung von drei CPUs bei 1%:** 1 – 0,01^3 = 1 – 0,000001 = 0,999999 = **99,99999%**

**Auslastung von drei CPUs bei 20%:** 1 – 0,2^3 = 1 – 0,008 = 0,992 = **99,2%**

## Aufgabe 2:

Auf einem Betriebssystem läuft eine parallele Software für astrophysikalische Berechnungen. Das Betriebssystem verwaltet dabei mehrere CPUs, die für das Berechnungsprogramm benutzt werden. Verwenden Sie für T1 = 10 Sekunden bzw. den Wert 1, da sich der Wert kürzen lässt. Berechnen Sie den theoretisch möglichen Speedup für die folgenden Fälle:

| Anzahl CPUs | 2 | 8 | 32 | 128 | 1024 |
|---|---|---|---|---|---|
| 75% Anteil serieller Code<br><br>f = 0,25 | 1,1428… | 1,28 | 1,3195… | 1,3298… | 1,3328… |
| 25% Anteil serieller Code<br><br>f = 0,75 | 1,6 | 2,9091… | 3,6571… | 3,9084… | 3,9883… |

Speedup = Sp = theoretisch mögliche Beschleunigung
p = Anzahl CPUs
T1 = Ausführungszeit auf einem System mit 1 CPU
Tp = Ausführungszeit auf einem System mit p-CPUs = Anteil serieller Code

**Formel:** ***Sp*** = T1 / Tp
***Tp*** = T1 * ((1 – f) + f / p)

Sp = T1 / T1 * ((1 – f) + f / p)

**Aufgaben 3 bis 9** sind Programmieraufgaben!

**Übung 5:**
**Aufgabe 1** ist Fortsetzung von Übung 3 & 4!
**Aufgabe 2** bezogen auf Interprozesskommunikation und die Implementierung von Textchat auf Basis von Unix-Domain-Sockets zu Programmieraufgaben von Übung 3 & 4!
**Aufgabe 3 bis 4** sind Programmieraufgaben!

## Aufgabe 1 & 2: Sleeping Barber-Problem (kommt nicht in Klausur dran)!

## Aufgabe 3:

Warum gibt es für Prozesse keinen direkten Übergang vom Zustand `blockiert‘ in den Zustand `rechnend‘?

Blockierte Prozesse müssen vom Scheduler zunächst Rechenzeit zugewiesen bekommen. Da bereits andere Prozesse berechnet werden oder darauf warten müssen ausgeführt zu werden, muss der blockierte Prozess die Reihenfolge der Prioritäten beachten und in den Rechenbereit-Zustand, um anschließend berechnet werden zu können.

## Aufgabe 4:

In zwei Tagen ist Prüfung! Sie bereiten sich eifrig auf die Prüfung vor und möchten schnell noch „den“ Tanenbaum und ein Linux-Grundlagenbuch ausleihen. Wie kann ein Deadlock entstehen?

**Deadlock-Bedingungen:**

1. Jede Ressource ist entweder verfügbar oder genau einem Prozess zugeordnet.
2. Prozesse, die Ressourcen schon reserviert haben, können noch weitere Ressourcen anfordern.
3. Ressourcen, die einem Prozess bewilligt wurden, können diesem nicht gewaltsam wieder entzogen werden. Der Prozess muss sie explizit freigeben.
4. Es gibt eine zyklische Liste von Prozessen, von denen jeder auf eine Ressource wartet, die dem nächsten Prozess gehört.

Studentin (Maja) hat sich das Buch „Tanenbaum“ ausgeliehen. Student (Mike) hat hingegen das Buch „Linux-Grundlagen“ ausgeliehen. Maja hat bereits eine Reservierung für „Linux-Grundlagen“ angefordert und Mike eine Reservierung für „Tanenbaum“.
**Schlussfolgerung:** Beide Büchern werden gleichzeitig ausgeliehen und reserviert, ein Deadlock entsteht.

## Aufgabe 5:

Oje, Sie fielen durch die Prüfung! (Hoffentlich nicht!) Welche Konsequenz ziehen Sie, um beim nächsten Versuch (auf legalem Wege) an die notwendigen Bücher zu gelangen?

Im Vorfeld sich Gedanken machen, wann man die Bücher ausleihen möchte.

**Aufgabe 6:** Programmieraufgabe zum Erstellen eines Deadlocks mittels Coffman-Deadlock-Bedingungen!

**Aufgabe 7:** Programmieraufgabe aus Aufgabe 6 an entsprechenden Code-Zeilen die Deadlock-Bedingungen erklären!

## Aufgabe 8:

Was ist ein Survey?

Ist eine beliebte und effiziente Methode zur mehr oder weniger systematischen Suche von Daten zu ganz unterschiedlichen Themen.

## Aufgabe 1:

Unter Linux beträgt die Frequenz für den Scheduler 1.000 Hz. Wie hoch ist somit die minimale Zeitscheibe für die Zuordnung von Rechenzeit (in ms)?

Die minimale Zeitscheibe ist 1 Millisekunde (ms).

## Aufgabe 2:

Erläutern Sie das Optimierungsproblem eines Schedulers, ohne auf einen spezifischen Algorithmus einzugehen.

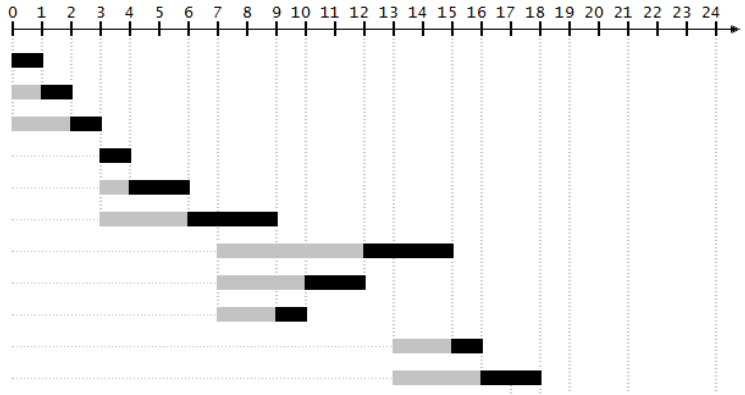
1. Der Scheduler verbraucht mit seinem Algorithmus ebenfalls Rechenzeit. Je komplexer der Algorithmus ist, desto weniger Rechenzeit steht anderen Anwendungen zur Verfügung.
2. Bei jedem Prozesswechsel wird der Scheduler aufgerufen. Je öfter wir Prozesse voneinander ablösen, umso öfter läuft der Scheduler, womit ein gewisser Overhead erzeugt wird.

## Aufgabe 3:

Zeichnen Sie ein Diagramm für den zeitlichen Ablauf der folgenden Prozesse im Falle des SJF-Algorithmus. Tragen Sie in das Diagramm auf einer Zeitachse **Wartezeit** und **Ausführungszeit** jedes Prozesses ein.
Die Wertepaare für die zu verwendenden Prozesse haben die folgende Form: Prozessnummer (Zeit der Verfügbarkeit, notwendige Rechenzeit) Prozesse:

**SJF-Algorithmus:** Shortest-Job-First bedeutet, die Prozesse mit der kürzeren Ausführzeit rechnen zuerst.

| | Wartezeit | Verg. Zeit |
|---|---|---|
| p1 (0, 1) | 0 Sek. | 0 Sek |
| p2 (0, 1) | 1 Sek. | 1 Sek. |
| p3 (0, 1) | 2 Sek. | 2 Sek. |
| p4 (3, 1) | 0 Sek. | 3 Sek. |
| p5 (3, 2) | 1 Sek. | 4 Sek. |
| p6 (3, 3) | 3 Sek. | 6 Sek. |
| p7 (7, 3) | 5 Sek. | 12 Sek. |
| p8 (7, 2) | 3 Sek. | 10 Sek. |
| p9 (7, 1) | 2 Sek. | 9 Sek. |
| p10 (13, 1) | 2 Sek. | 15 Sek. |
| p11 (13, 2) | 3 Sek. | 16 Sek. |
| | 22 Sek. | |




## Aufgabe 4:

Wie hoch ist die durchschnittliche Wartezeit der Prozesse aus Aufgabe 3?

Summe Wartezeit / Anzahl Prozesse = 22 / 11 = **2**

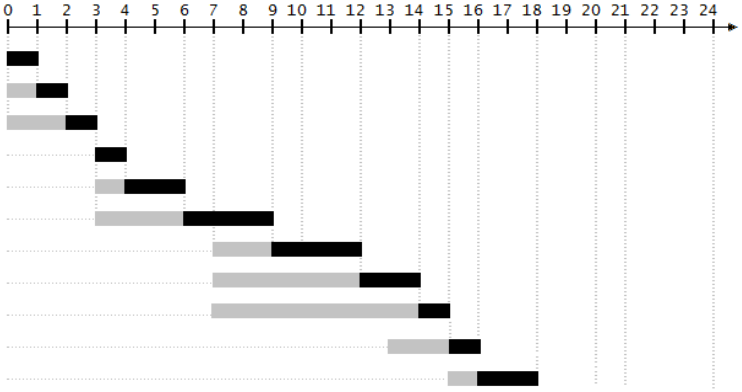
**Fortsetzung auf nächster Seite!**

## Aufgabe 5:

Wie hoch wäre die durchschnittliche Wartezeit der Prozesse aus Aufgabe 3 im Falle des FCFS-Algorithmus?

**FCFS-Algorithmus:** First-Come-First-Served bedeutet, Prozesse laufen in der Reihenfolge, in der sie Rechenzeit anfordern.

| | Wartezeit | Verg. Zeit |
|---|---|---|
| p1 (0, 1) | 0 Sek. | 0 Sek |
| p2 (0, 1) | 1 Sek. | 1 Sek. |
| p3 (0, 1) | 2 Sek. | 2 Sek. |
| p4 (3, 1) | 0 Sek. | 3 Sek. |
| p5 (3, 2) | 1 Sek. | 4 Sek. |
| p6 (3, 3) | 3 Sek. | 6 Sek. |
| p7 (7, 3) | 2 Sek. | 9 Sek. |
| p8 (7, 2) | 5 Sek. | 12 Sek. |
| p9 (7, 1) | 7 Sek. | 14 Sek. |
| p10 (13, 1) | 2 Sek. | 15 Sek. |
| p11 (13, 2) | 3 Sek. | 16 Sek. |
| | 26 Sek. | |




**blauer Bereich zeigt die Veränderung im Vergleich zum SJF-Algorithmus*

Summe Wartezeit / Anzahl Prozesse = 26 / 11 = **2,3636…**

## Aufgabe 6:

Zeigen Sie, das ein Echtzeitsystem mit den folgenden Bedingungen als ‚scheduable' gilt.

| **Ereignis** | **e_1** | **e_2** | **e_3** |
|---|---|---|---|
| **Periode (in ms)** | 100 | 200 | 500 |
| **Notwendige Rechenzeit (in ms)** | 50 | 30 | 100 |

Ein Echtzeitsystem gilt als ***„scheduable",*** wenn die folgende Bedingung erfüllt ist:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Das **Ereignis i** benötigt dabei ***Ci*-Sekunden Rechenzeit** und tritt mit einer Periode *Pi* auf.

(50 / 100) + (30 / 200) + (100 / 500) = 0,5 + 0,15 + 0,2 = **0,85 ≤ 1,0**

**Schlussfolgerung:** Unser Ergebnis 0,85 ist in diesem Fall kleiner/gleich dem Wert 1,0; somit ist unser Echtzeitsystem **„scheduable"**

## Aufgabe 7:

Gehen Sie von den Werten aus Ausgabe 6 aus. Ein neues Ereignis mit einer Periode von 1 Sekunde wird in das System einbezogen. Wie hoch darf der Bedarf an Rechenzeit dieses Ereignisses maximal sein, damit das System noch als ‚scheduable' gilt?

| **Ereignis** | **e_1** | **e_2** | **e_3** | **e_4** |
|---|---|---|---|---|
| **Periode (in ms)** | 100 | 200 | 500 | 1000 |
| **Notwendige Rechenzeit (in ms)** | 50 | 30 | 100 | GESUCHT |

**zusätzlich ins System hinzugefügt*

Unser Ergebnis aus Aufgabe 6 lautet **0,85**. Damit das System als weiterhin „scheduable" gilt, wird nachfolgend, die notwendige Rechenzeit für das Ereignis **e_4** ausgerechnet:

1,0 – 0,85 = 0,15 **= 150 (ms)** <- (notwendige Rechenzeit für e_4)

- Übung 8 -

## Aufgabe 1:

Ist das Konzept der Adressräume mit der Interprozesskommunikation (IPC) vereinbar? Begründen Sie Ihre Antworten.

Die Interprozesskommunikation ist ein Verfahren, das zum Informationsaustausch zwischen Prozessen eines Systems eingesetzt wird. Um IPC realisierbar zu machen, benötigt sie verschiedene Technik wie z. B. Pipes & FIFOS, Shared Memory, Message Queues, Unix-Domain-Sockets und Signale.

Prozesse können nicht ohne weiteres auf den Adressraum eines anderen Prozesses zugreifen bzw. auf die jeweiligen Daten. Ist der beschriebene Fall allerdings gewollt dann hilft die Interprozesskommunikation das Problem der Kommunikation zwischen den Prozessen zu lösen.

Daher ist das Konzept der Adressräume mit der Interprozesskommunikation **„vereinbar“**.

**Aufgabe 2:**

Ein Computer verfügt über einen 64-Bit-CPU und 64 GByte RAM. Wie groß ist der virtuelle und wie groß der physikalisch adressierbare Speicher?

Physikalisch adressierbarer Speicher bei 64-Bit-CPU: *64 GByte RAM* -> **64 GByte RAM**
Virtuell adressierbarer Speicher bei 64-Bit-CPU: **2^64 = 2147483648 GByte RAM**

Verständnisnotiz:
Nehmen wir an, ihr besitzt ein 32-Bit-Betriebssystem sowie 8 GByte RAM. In diesem Fall könnte euer Prozessor lediglich 4 (oder 4096) GByte RAM physikalischen Speicher adressieren.
Bei einem 64-Bit-Betriebssystem liegt diese Grenze bei 2^64 GByte RAM.

8 GByte RAM kosten durchschnittlich etwa 45 €. Wie viel würden 2147483648 GByte RAM kosten?

**Aufgabe 3:**

Folgende Pages wurden im System geladen. Spielen Sie das Szenario der Seitenersetzung jeweils mit den folgenden Algorithmen durch: NRU, FIFO, Second-Chance, Clock, LRU. Ersetzen Sie jeweils drei Seiten.

| Page Nummer und zugleich Reihenfolge | Referenziert (R), Modiziert (M) | Zeit seit letztem Zugriff (in ms) |
|---|---|---|
| 1 | 1, 0 | 10 ms |
| 2 | 0, 1 | 15 ms |
| 3 | 0, 0 | 20 ms |
| 4 | 1, 1 | 10 ms |
| 5 | 1, 1 | 5 ms |
| 6 | 1, 1 | 10 ms |

Reihenfolge, in der die Pages ersetzt werden:

| NRU | FIFO | Second-Chance | R, M | Clock | R, M | LRU |
|---|---|---|---|---|---|---|
| 3 | 1 | 2 | 0, 1 | 2 | 0, 1 | 3 |
| 2 | 2 | 3 | 0, 0 | 3 | 0, 0 | 2 |
| 1 | 3 | 1 | 0, 0 | 1 | 0, 0 | 1 |
| 4 | 4 | 4 | 0, 1 | 4 | 0, 1 | 4 |
| 5 | 5 | 5 | 0, 1 | 6 | 0, 1 | 6 |
| 6 | 6 | 6 | 0, 1 | 5 | 0, 1 | 5 |

- Übung 9 -

## Aufgabe 1:

Wie wird verfahren, wenn lange Dateinamen (bspw. von einer ext4-Partition) auf eine FAT-Partition mit der 8.3-Namenskonvention kopiert werden?

Eine FAT-Partition begrenzt Dateinamen auf 8 Zeichen (Buchstaben oder Ziffern), gefolgt von einem Punkt und der Namenserweiterung z. B. **.txt** (darf max. aus 3 Zeichen bestehen). Es wird nicht zwischen Groß- und Kleinbuchstaben unterscheiden, Dateinamen werden mit Großbuchstaben abgebildet. Sonderzeichen wie Doppelpunkt, Fragezeichen, Stern sind unzulässig.

**Beispiele:** TEXTFILE.TXT, EXERCISE.TXT, IAMAFILE.TXT

Möchte man eine Datei mit langem Namen z. B. textfile.txt auf einer FAT-Partition speichern, dann wird die Datei auf der Partition als LFN-Variante folgend gespeichert: TEXTFILE.TXT.

**1 Variante:** Wenn die Länge der ursprünglichen Datei kleiner/gleich 8 Zeichen oder Ziffern, dann wird die Datei in GROSSBUCHSTABEN umgewandelt. **Beispiele:** TextFile.txt -> TEXTFILE.TXT

**2 Variante:** Wenn die Länge der ursprünglichen Datei größer als 8 Zeichen oder Ziffern, dann wird die Datei in GROSSBUCHSTABEN umgewandelt, auf ihre ersten 6 Zeichen gekürzt, gefolgt von „~“ und einer fortlaufenden Nummer beginnend bei 1, einem Punkt und der Namenserweiterung.
**Beispiele:** TextFile1.Mine.txt -> TEXTFI~1.TXT

**3 Variante:** Im Falle, dass die ersten beiden Varianten nicht erfolgen, dann wird die Datei in GROSSBUCHSTABEN umgewandelt, auf ihre ersten 2 Zeichen gekürzt, gefolgt von einer zufälligen 4-stelligen Hexadezimalzahl sowie „~“ und einer fortlaufenden Nummer beginnend bei 1, einem Punkt und der Namenserweiterung.
**Beispiele:** TextFile.Mine.txt -> TE021F~1.TXT

*LFN ist aufgrund der erweiterten Variante VFAT möglich!*

## Aufgabe 2 bis 6: **Programmieraufgaben zum Erstellen eines Programm, das auf die Inode einer Datei und ihrem/ihrer Hardlinks zugreift und ausgibt sowie mittels strace Systemaufrufe überprüfen.**

- Übung 10 -

## Aufgabe 1:

Worin besteht der Unterschied zwischen Discretionary und Mandatory Access Control bei Betriebssystemen?

**DAC – Discretionary Access Control:** Benutzerbasierte und –bestimmbare Zugriffskontrolle; Zugriff auf Daten wird anhand einer Identität getroffen.

**Beispiel:** Snowden ist in einem engen Kreis aus Menschen, die über streng geheimes Wissen verfügen. Der Vater von Snowden gehört nicht zu diesen Menschen. Snowdens Identität erlaubt ihm dieser Gruppe anzugehören und das Wissen zu besitzen. Snowden entscheidet nun für sich, dass er seinen Vater einweiht. Snowden verfügt über die Identität seinen Vater in den engen Kreis hinzuzufügen. Daraus folgt, dass sein Vater nun ebenfalls eine Identität besitzt, die es ihm erlaubt dem engen Kreis anzugehören und andere Person in diesen engen Kreis hinzuzufügen.

**MAC - Mandatory Access Control:** Zugriffskontrolle auf Basis von Regeln und Attributen

**Beispiel:** Ein Bewerber auf einen Studienplatz an der HS Worms möchte zum ersten Mal auf Moodle zugreifen. Er fragt einen Administrator ob ihm der Zugang gewährt werden kann. Der Administrator ist für die Verwaltung der Computer zuständig sowie ihrer Instandhaltung. Allerdings besitzt er keine DAC-Identität, die es ihm erlaubt, dem Informatikstudent die **Zugrifsskontrolle** auf Moodle zu geben. Die Zugriffskontrolle kann **ausschließlich** von der Studentenverwaltung vergeben werden. Dafür legt sie die **Regeln** fest, dass der Student an der Hochschule Worms eingeschrieben sein muss und einer bestimmten Fakultät angehören muss. Werden die Regeln erfüllt, erhält der eingeschriebene Student seine Zugangsdaten, mit denen er sich identifizieren kann.

## Aufgabe 2:

In der Vorlesung wurde ein Beispiel für Alice, Bob und Database(x) eingeführt. Bekäme Alice lesenden bzw. schreibenden Zugriff auf Database(x), wenn folgendes gelten würde?

(L,C) für Database(x): (confidential, {research})

Alice: (secret, {accounting, research})
Bob: (confidential, {accountin, support})

Regelwerk:
L = Sicherheitslevel
s = Subjekt (Person)
o = Objekt (Information etc.)
Alice (L, C) = Compartement, z. B. Alice(secret (L), {accounting, research} (C))

1. Lesen erlaubt, wenn gilt: $L(o) \leq L(s)$

   **Erklärung:** Der Chef der NSA mit Sicherheitslevel = TOP SECRET darf das geheime Objekt (Information) mit Sicherheitslevel = SECRET lesen. Seine Angestellten mit Sicherheitslevel = CONFIDENTIAL allerdings nicht, denn es handelt sich um vertrauliche und machtvolle Informationen.

2. Schreiben erlaubt, wenn gilt: $L(o) \geq L(s)$

   **Erklärung:** Der Chef der NSA mit Sicherheitslevel = TOP SECRET darf nicht in das geheime Objekt mit Sicherheitslevel = SECRET schreiben. Seine Angestellten mit Sicherheitslevel = CONFIDENTIAL dürfen reinschreiben.
   Der Chef darf das nicht, da er über Wissen auf TOP-SECRET-Level verfügt, welches er theoretisch in das Objekt mit SECRET-Level reinschreiben könnte und jeder mit Sicherheitslevel = SECRET und entsprechender Abteilung könnte in diesem Objekt Information zu einem TOP-SECRET Objekt lesen. Seine Angestellten mit CONFIDENTIAL-Level dürfen allerdings in jedes Objekt ihrer Abteilung auf/über ihrem Sicherheitslevel Wissen zu Objekten mit CONFIDENTIAL-Level schreiben, denn jeder aus gleichen Abteilung mit höherem Sicherheitslevel verfügt schon über das Wissen.

## Fortsetzung auf nächster Seite!

Aufgabe:
Database(x) (**confidential**, {**research**}) ≤ Alice: (**secret**, {accounting, **research**}) : **Alice darf lesen!**

Database(x) (**confidential**, {research}) ≥ Alice: (**secret**, {accounting, research}) : **Alice darf nicht schreiben!**

**Lösung:** Alice erhält lesenden Zugriff, aber nicht schreibenden Zugriff auf Database(x).

## Aufgabe 3:

Sie lernten in der Vorlesung das Konzept der Multilevel Security (MLS) kennen. Welche typischen Sicherheitslevels (Geheimhaltungsstufen) gibt es in der BRD?

Die Geheimhaltungsstufen lauten in aufsteigender Reihenfolge: Verschlussache – NUR FÜR DEN DIENSTGEBRAUCH, Verschlusssache – VERTRAULICH, GEHEIM, STRENG GEHEIM

## Aufgabe 4:

Kann der in der Vorlesung vorgestellte verdeckte Kanal für Android-basierte Smartphones eher durch einen Menschen oder durch einen Computer detektiert werden?

NICHT SICHER!

## Aufgabe 5:

In der Vorlesung fiel kurz der Begriff `Sandbox‘. Recherchieren Sie genauer, was darunter zu verstehen ist.

Beim Sandboxing handelt es sich um eine Technik, um Software in einer speziellen Laufzeitumgebung auszuführen, die von übrigen Systemressourcen isoliert ist.

- Übung 11 -

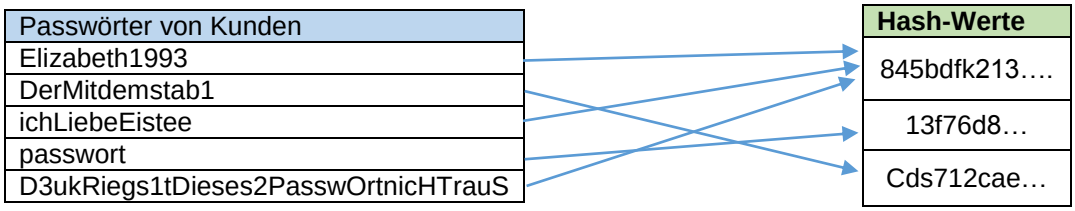
## Aufgabe 1:

Was ist eine Hashfunktion (grundlegend)? Wozu dient sie bei der Speicherung von Passwörtern? Welche Möglichkeiten gibt es, Passwörter sicher zu gestalten?

**Hash**

Eine Hashfunktion ist eine Funktion/Abbildung, die eine Beziehung zwischen einer Menge Eingabewerten (Passwörter) und einer kleineren Menge Hashwerten abbildet.

Sie dient dazu z. B. für Passwörter von Kunden eines Unternehmens eine Prüfsumme (Komprimat) zu erzeugen, um eine höhere Sicherheit zu gewährleisten.




**Notiz: Oft sind die von der Hashfunktion erzeugten Hashwerte Hexadezimal-Werte.*

**Info:**

Möchte der Kunde sich nun mit seinem Passwort einloggen, dann wird der erzeugte Hashwert für sein Passwort mit dem zuvor gespeicherten Hashwert aus der Datenbank verglichen.

**Info 2:**

Ein Cyberangriff auf die Datenbank des Unternehmens würde dem Angreifer es erschweren z. B. vom Hashwert 845bdfkj213…. einen Rückschluss auf die drei Passwörter zu ziehen.

**Problematik:**

Angenommen der Angreifer sammelte die Hashwerte aus der Datenbank der Firma Google. Um die Passwörter der Kunden aus den gesammelten Hashwerten herauszufinden, nutzt er nun ein Rainbow-Table womit er eine Reihe von geratenen Passwörter mit Hashwerten erzeugt. Danach vergleicht er jeden Hashwert aus seiner geratenen Listen mit den Hashwert aus der Datenbank der Firma Google und schon weiß er, welches Passwort zu welchem Hashwert gehört. Häufig nutzen Kunden die **gleichen Passwörter** auch für andere Dienste womit trotz unterschiedlicher Firmen und Datenbanken **die Hashwerte die gleichen** sind. Um dies zu „erschweren“ kommt **Salt** ins Spiel.

**Salt**

Mittels der Salt-Variante kann die Hashfunktion erweitert werden. Dem eingegeben Passwort wird ein zufälliger Salt-Wert hinzugefügt und erst dann ein Hashwert erzeugt. Beim erneuten Einloggen des Kunden wird nun sowohl der Hashwert als auch der Salt-Wert mit den gespeicherten Hash- sowie Salt-Werten der Datenbank überprüft. Die zufälligen Salt-Werte sind bei unterschiedlichen Datenbanken oder bei gleichen Passwörtern trotzdem unterschiedlich. Dadurch hat jedes Passwort einen individuellen Salt-Wert und kann nur sehr schwer erfasst werden, denn die genutzte Rainbow-Table des Angreifer würde durch die gespeicherten Salt-Werte enorm groß werden.

## Aufgabe 2:

Nennen Sie mindestens zwei Möglichkeiten, eine Authentifikation umzusetzen.

- Biometrisches Passwort (Gesichtserkennung, Fingerabdruck etc.)
- Zwei-Faktor-Authentifizierung
- Einmalige-Passwörter
- Benutzername und Passwort
- Chipkarte

## Aufgabe 3: **Filesystem Hierarchy Standard (kommt nicht in Klausur dran!)**

## Aufgabe 4:

Ermitteln Sie die Anzahl der Lines of Code (LOC) des Linux-Kernels und des FreeBSD-Kernels.

Der Linux-Kernel umfasst seit der Veröffentlichung der Version 4.20 vom 23. Dezember 2018 etwa 26 Millionen Zeilen Code verteilt auf 62.446 Dateien.

Der FreeBSD-Kernel umfasst seit der Veröffentlichung der Version 12.0 vom 11. Dezember 2018 etwa 24 Millionen Zeilen Code.

## Aufgabe 5: freiwillig!

- Übung 12 -

## Aufgabe 1:

Erläutern Sie das Konzept einer Inode.

Inodes werden in Dateisystem von Unix-Betriebssystemen eingesetzt. Jede Datei in einem Dateisystem verfügt über eine eigene Inode. Die Inode dient für die Datei als ein Speicherort ihrer Metadaten (Eigenschaften etc.) sowie als Speicherort für die Verweise auf die Dateisystemblöcke, die die eigentlichen Dateiinhalte beinhalten.

## Aufgabe 2:

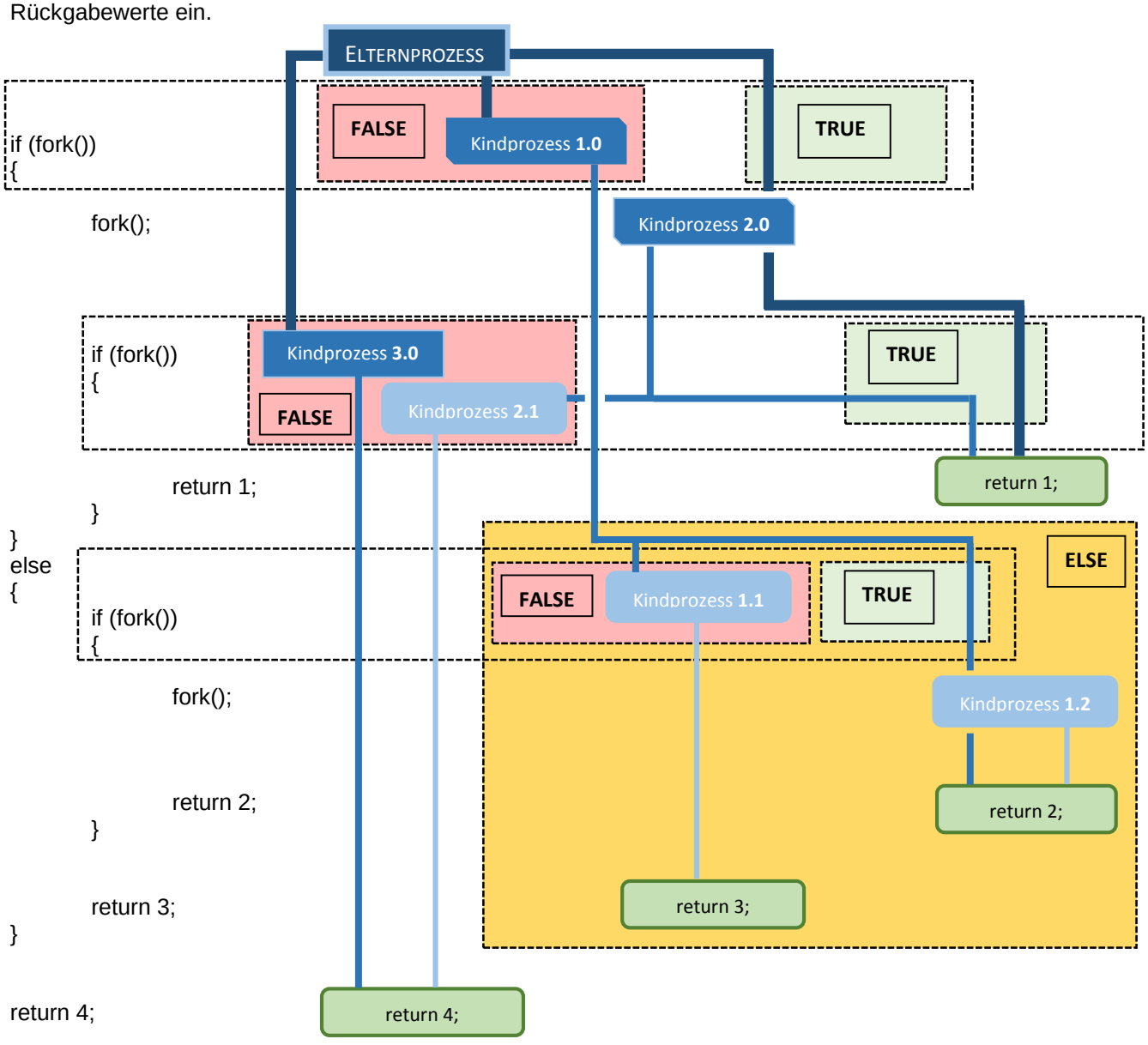
Erläutern Sie den Unterschied zwischen Timesharing und Multiprogramming.

**Multiprogramming:** Teilt physikalischen Speicher auf, jeder Prozess/Job erhält dadurch eine Partition. Dadurch arbeitet die CPU mit Prozessen/Jobs weiter, die nicht auf I/O warten. Die CPU kann somit voll ausgelastet werden.

**Timesharing (Verbesserung des Multiprogramming-Konzepts):** Mehrere Anwender können gleichzeitig mit einem System arbeiten, das System teilt den Prozessen der Anwender Rechenzeit zu.

## Aufgabe 3:

Zeichnen Sie den Prozessbaum zu folgendem Code und tragen Sie in Ihren Prozessbaum jeweils die Rückgabewerte ein.



## Aufgabe 4:
Erläutern Sie das Konzept der Address Space Layout Randomization (ASLR) und der GCC ProPolice.

## Aufgabe 5:
Erläutern Sie die Funktionsweise eines Buffer Overflows, einer Buffer Overflow Exploitation, eines Heap Overflows und eines Formatstring Overflows.

## Aufgabe 6:
Wie funktionieren Integer Promotions, Integer Overflows und Off-by-One-Errors?