

Operating Systems and Networks SoSe 25 Solutions

Igor Dimitrov

2024-12-18

Table of contents

Preface	3
1 Blatt 01	4
1.1 Aufgabe 1	4
1.2 Aufgabe 2	5
1.3 Aufgabe 3	5
1.4 Aufgabe 4	6
1.5 Aufgabe 5	6
1.6 Aufgabe 6	7
1.7 Aufgabe 7	7

Preface

1 Blatt 01

1.1 Aufgabe 1

Learning how to Learn:

- **Zwei Denkmodi aus „Learning How to Learn“**
 - **Fokussierter Modus:** Zielgerichtetes, konzentriertes Denken. Gut für bekannte Aufgaben und Übung.
 - **Diffuser Modus:** Entspanntes, offenes Denken. Hilft bei neuen Ideen und kreativen Verknüpfungen.
- **Aufgaben und passende Denkmodi**
 - a) Fokussierter Modus
Warum: Erfordert Konzentration und gezieltes Einprägen.
 - b) Zuerst diffuser, dann fokussierter Modus
Warum: Erst Überblick und Verständnis aufbauen, dann vertiefen.
 - c) Fokussierter Modus
Warum: Klare, schrittweise Übung – ideal für fokussiertes Denken.
 - d) Beide Modi
Warum: Fokussiert für Details & Übungen, diffus für Überblick & Vernetzung.

John Cleese:

- **Zwei Denkmodi:**
 1. **Offener Modus:** Locker, spielerisch, kreativ.
Beispiel: Ideen für eine Geschichte sammeln.
Warum: Offenheit fördert neue Einfälle.
 2. **Geschlossener Modus:** Zielgerichtet, angespannt, entscheidungsfreudig.
Beispiel: Bericht überarbeiten und fertigstellen.
Warum: Präzises Arbeiten und klare Entscheidungen nötig.
- **Vergleich mit „Learning How to Learn“**

- **Offen** \Leftrightarrow **Diffus**: Für Kreativität und Überblick.
- **Geschlossen** \Leftrightarrow **Fokussiert**: Für Detailarbeit und Umsetzung.
- **Alexander Fleming**:
 - **Modus**: Offen
 - **Warum**: Fleming entdeckte Penicillin zufällig, weil er offen und entspannt war – neugierig statt zielgerichtet. Im geschlossenen Modus hätte er die verschimmelte Petrischale wohl einfach weggeschmissen – zu fokussiert für zufällige Entdeckungen.
- **Alfred Hitchcock**:
 - **Modus**: Offen
 - **Wie**: Er erzählte lustige Anekdoten, um das Team zum Lachen zu bringen – so schuf er eine entspannte Atmosphäre, die kreatives Denken förderte.

1.2 Aufgabe 2

- i)
 - x64: 16 64 Bit GPRs¹ $\Rightarrow 16 \times 64 \text{ b} = 16 \times 8 \text{ B} = 2^7 \text{ B}$.
 - AVX2: 16 256 Bit GPRs² $\Rightarrow 16 \times 256 \text{ b} = 16 \times 32 \text{ B} = 2^9 \text{ B}$
 - ii)
 - x64: $\frac{2^7}{2^{30}} = \frac{1}{2^{23}}$
 - AVX2: $\frac{2^9}{2^{30}} = \frac{1}{2^{21}}$
- allgemein gilt: $10^3 \approx 2^{10}$, und $\frac{2^x}{2^y} = \frac{1}{2^{y-x}}$

1.3 Aufgabe 3

- Der Zugriff scheitert, weil der Arbeitsspeicher durch die **Memory Protection** (z.B. Paging mit Zugriffsrechten) vom Betriebssystem isoliert wird. Nur der Kernel darf die Speicherbereiche aller Prozesse sehen und verwalten.
- Ein Prozess kann trotzdem auf Ressourcen anderer Prozesse zugreifen über kontrollierte Schnittstellen wie IPC (Inter-Process Communication), Dateisysteme, Sockets oder Shared Memory, die vom Betriebssystem verwaltet und überwacht werden.
- Welche Risiken entstehen bei höchstem Privileg für alle Prozesse?
 - **Sicherheitslücken**: Jeder Prozess könnte beliebige Speicherbereiche lesen/schreiben.

¹<https://www.wikiwand.com/en/articles/X86-64>

²https://www.wikiwand.com/en/articles/Advanced_Vector_Extensions

- **Stabilitätsprobleme:** Fehlerhafte Prozesse könnten das System zum Absturz bringen.
- **Keine Isolation:** Malware hätte vollen Systemzugriff, keine Schutzmechanismen.

1.4 Aufgabe 4

Kernel-Code benötigt einen sicheren, kontrollierten Speicherbereich (seinen eigenen Stack), um zu vermeiden:

- Beschädigung durch Benutzerprozesse
- Abstürze oder Rechteauserweiterung (Privilege Escalation)

Daher hat jeder Prozess:

- Einen User-Mode-Stack (wird bei normaler Ausführung verwendet)
- Einen Kernel-Mode-Stack (wird bei System Calls und Interrupts verwendet)

1.5 Aufgabe 5

Entfernte Systemaufrufe

Systemaufruf	Grund für Entfernung
creat	Entspricht vollständig <code>open(path, O_CREAT O_WRONLY O_TRUNC, mode)</code> .
dup	Entspricht vollständig <code>fcntl(fd, F_DUPFD, 0)</code> .

Alle übrigen Systemaufrufe bieten **essenzielle Funktionen**, die nicht exakt durch andere ersetzt werden können.

Sie decken ab:

- Datei- und Verzeichnisoperationen (`open`, `read`, `write`, `unlink`, `mkdir`, etc.)
- Prozessmanagement (`fork`, `exec`, `wait`, `exit`, etc.)
- Metadatenverwaltung (`chmod`, `chown`, `utime`, etc.)
- Kommunikation und Steuerung (`pipe`, `kill`, `ioctl`, etc.)
- Zeit- und Systemabfragen (`time`, `times`, `stat`, etc.)

Ohne sie wären bestimmte Kernfunktionen unmöglich.

1.6 Aufgabe 6

script.sh auch im Zip:

```
cd $1
while :
do
    echo "5 biggest files in $1:"
    ls -S | head -5
    echo "5 last modified files starting with '$2' in $1:"
    ls -t | grep ^$2 | head -5
    sleep 5
done
```

1.7 Aufgabe 7

Vorteile:

- **Komplexitätsreduktion:** Abstraktionen verbergen technische Details und erleichtern das Entwickeln und Verstehen von Systemen.
- **Wiederverwendbarkeit:** Einmal geschaffene Abstraktionen (z.B. Dateisystem, Prozesse) können flexibel in verschiedenen Programmen genutzt werden.

Nachteile:

- **Leistungsaufwand:** Abstraktionsschichten können zusätzliche Rechenzeit und Speicherverbrauch verursachen.
- **Fehlerverdeckung:** Probleme in tieferen Schichten bleiben oft verborgen und erschweren Fehlersuche und Optimierung.