Universität Heidelberg
Sommersemester 2022

AG Parallele und Verteilte Systeme (PVS)
Artur Andrzejak

# 1. Übung zur Vorlesung „Betriebssysteme und Netzwerke“ (IBN)

Abgabedatum: 3.05.2022, 11:00 Uhr

---

## Aufgabe 1 (4 Punkte)

Als Begleitung zur Vorlesung werden wir öfter auf folgenden Online-Kurs der University of California, San Diego, zurückgreifen: *Learning How to Learn*[1]. Sie können sich dazu anmelden oder aber einfach Videos aus dem zugehörigen YouTube-Kanal[2] verwenden.

Schauen Sie das Video *Focused versus Diffuse Thinking*[3] an. In diesem Video werden zwei Modi des Denkens beschrieben, die beim Lernen relevant sind.

- Beschreiben Sie beide Modi in eigenen Worten. Geben Sie für diese Lernaufgaben an, welcher Modus geeigneter ist: (a) Vokabeln lernen, (b) ein völlig neues Konzept aneignen, (c) eine bestimmte Art von Rechenaufgabe trainieren und (d) auf die Klausur in IBN lernen. Begründen Sie Ihre Zuordnungen.
- Hören Sie sich die Anekdoten über die berühmten Denker Salvador Dalí und Thomas Edison im Video *Thomas Edison & Salvador Dali who used Focused and diffuse modes of thinking for problem solving* [4] an. Beschreiben Sie deren Technik, in den ersten Modus zu gelangen und dann in den anderen Modus überzugehen. Welcher ist welcher und welche Funktion übernehmen sie jeweils?

Im Jahr 1991 hielt der britische Künstler John Cleese einen Vortrag über Kreativität. Schauen Sie die Videoaufzeichnung[5] an, von 6:57 bis 10:27. Ein Transkript finden Sie bei dem Video selbst oder auf dieser Webseite[6] (allerdings dort ohne Zeitmarken).

- Cleese führt zwei Modi des Denkens ein. Beschreiben Sie beide Modi in eigenen Worten. Geben Sie für beide Modi je ein Beispiel für eine Aufgabe, die in diesem Modus besonders gut zu bewältigen ist. Welcher entspricht welchem Modus aus dem Video über Lernen und warum?
- Im Video werden Anekdoten von den Denkern Alexander Fleming und Alfred Hitchcock erzählt. In welchem Modus machte Fleming seine berühmte Entdeckung der Wirkung des Penicillin? Warum war dies in dem anderen Modus unwahrscheinlich? Hitchcock wendete eine Technik ein, um sein Team in einen der beiden Modi zu versetzen. Welcher Modus ist das und wie machte er dies?

[1] http://www.coursera.org/learn/learning-how-to-learn/
[2] https://www.youtube.com/channel/UCojOYrwehhFpaJfYG3DziHA
[3] https://www.youtube.com/watch?v=lJtUg-3DfUk
[4] https://www.youtube.com/watch?v=hbRc8uUKSHM
[5] https://www.youtube.com/watch?v=Pb5oIIPO62g
[6] http://genius.com/John-cleese-lecture-on-creativity-annotated

**Aufgabe 2** **(Bonus, 0.5 Punkte)**

Überlegen Sie, welche Vorteile die Verwendung von Abstraktionen[7] in den Betriebssystemen und Software allgemein haben kann. Welche Nachteile haben diese? Benennen Sie mindestens zwei Vorteile und zwei Nachteile.

**Aufgabe 3** **(3 Punkte)**

Stellen Sie die Formel für die erwartete Verzögerung $t_{ave}$ (Latenz) beim Zugriff auf den Speicher (d.h. RAM) auf, in einem System mit einem Cache (nur eine Cache-Stufe). Dabei soll $t_c$ die Latenz eines Cache-Zugriffs sein (gleiche Latenz, egal ob der angeforderte Speicherinhalt im Cache gefunden wurde, d.h. „cache hit", oder nicht, d.h. „cache miss"). Weiterhin sei $t_r$ die Latenz eines RAM-Zugriffs und $p_+$ die Wahrscheinlichkeit für ein „cache hit". Des Weiteren listen Sie mindestens zwei Aspekte auf, die in realen Systemen die Wahrscheinlichkeit $p_+$ beeinflussen könnten. Begründen Sie Ihre Lösungen. **Hinweis** zur Formel: schauen Sie sich die Definition eines Erwartungswertes an[8]. Überlegen Sie, was hier die Zufallsvariable ist.

**Aufgabe 4** **(Bonus, 1 Punkte)**

Als eine Erweiterung der vorangehender Aufgabe überlegen Sie, welchen Wert die Wahrscheinlichkeit $p_+$ mindestens haben muss, damit $t_{ave} \leq 5t_c$ sein soll, falls $t_r = 100t_c$ gilt. Geben Sie als Ergebnis einen exakten Wert an und zeigen Sie die Herleitung der Lösung.

**Aufgabe 5** **(1 Punkt)**

Schauen Sie sich die relevanten Teile des Artikels „Introduction to x64 Assembly"[9] an und finden Sie heraus, wie viele Bytes eine x64-CPU in den Datenregistern (d.h. GPR, MMX, XMM) speichern kann. Sie können zur Vereinfachung annehmen, dass jedes Register gleichwertig ist. Welchen Anteil des Hauptspeichers können diese Register abdecken, wenn ein Rechner 1 GiB RAM (1 gibibyte[10], d.h. $2^{30}$ Bytes) hat?

**Aufgabe 6** **(Bonus, 0.5 Punkte)**

Für einen Programmierer sieht ein Systemaufruf genauso aus wie der Aufruf einer Bibliotheksfunktion. Ist es für ihn / sie wichtig zu wissen, wann eine Bibliotheksfunktion einen Systemaufruf ausführt? Falls ja, unter welchen Umständen und warum?

[7] Ein ironischer Blick auf die Abstraktionen von Randall Munroe: https://xkcd.com/676/
[8] https://de.wikipedia.org/wiki/Erwartungswert
[9] https://www.intel.com/content/dam/develop/external/us/en/documents/introduction-to-x64-assembly-181178.pdf
[10] https://en.wikipedia.org/wiki/Gibibyte

## Aufgabe 7 (2 Punkte)

In Vorlesung 3 haben Sie den POSIX-Befehl `execve()` zum Erzeugen eines neuen Prozesses kennengelernt. Der POSIX-Standard sieht mehrere Varianten dieses Befehls vor. Listen Sie alle Varianten auf und geben Sie sie in ihrer vollen Signatur an. Für `execve()` sieht die Signatur z.B. wie folgt aus:

```
int execve(const char *filename, char *const argv[], char *const envp[]);
```

Beschreiben Sie, worin die Unterschiede der verschiedenen Funktionen bestehen. Welche Eigenschaften haben die verschiedenen Varianten und wann können die verschiedenen Eigenschaften nützlich sein?

## Aufgabe 8 (4 Punkte)

Erinnern Sie sich an den Befehl `fork()` und wie er funktioniert. Was bedeutet der Rückgabewert von `fork()`? Beantworten Sie folgende Fragen. Wenn Sie sich nicht sicher sind, oder Ihre Antwort einfach praktisch überprüfen wollen, fügen Sie eventuell Ausgaben oder `sleep(...)`-Befehle ein, kompilieren und starten Sie das Programm. In einer Bash-Shell können Sie sich mit dem Befehl `ps` über laufende Prozesse informieren.

**a.** Wie viele Prozesse (inklusive des ersten Elternprozesses) werden durch das folgende Programm erzeugt? Zeichnen Sie den Graph der entstehenden Prozesshierarchie, wobei dem Elternprozess das Label „1“ zugeordnet wird. Das erste Kind von Elternprozess „1“ sei dann „1.1“, das zweite „1.2“, usw.

```
int main(int argc, char* argv[]) {
  int i=0;
  if (fork()!=0) i++;
  if (i!=1) fork();
  fork();
  return 0;
}
```

**b.** Der Ausdruck `a % b` bezeichnet den Rest der Division der Integerzahl `a` durch die Integerzahl `b` (diese Operation wird *modulo* genannt). Wie viele erzeugte Prozesse (inklusive des ersten Elternprozesses) sind durch das folgende Programm beim Aufruf mit dem Kommandozeilenargument `10` maximal zu erwarten? Erklären Sie, warum das normalerweise so ist. Was passiert, wenn nebenbei noch andere Prozesse gestartet werden oder terminieren?

```
int main(int argc, char* argv[]) {
  while (fork() % atoi(argv[1]) != 0);
  return 0;
}
```

## Aufgabe 9 (2 Punkte)

Ein sog. *Hardcore-Informatiker* liest Bücher grundsätzlich nur über eine Shell (Kommandozeileninterpreter), und zwar in Form einer ASCII-Datei.

**a.** Benennen Sie und erläutern Sie kurz zwei Shell-Befehle unter Linux, die sich dafür eignen würden.

**b.** Unser Hardcore-Informatiker liest seit sechs Stunden den Roman „Die Abgründe des Kernels“ von Franz Kafka. Er möchte nun kurz unterbrechen, seine Emails überprüfen, und dann an gleicher Stelle weiterlesen. Es steht ihm aber nur ein Shell-Fenster zur Verfügung (auf dem er jetzt liest). Wie kann er das bewerkstelligen? Benennen Sie die nötigen Eingaben als auch Shell-Befehle und erläutern Sie kurz deren Funktion.

**c.** Nach einer Weile unterbricht er wieder das Lesen und startet das Kopieren einer sehr großen Datei (300 GB). Nach ca. einer Minute wird es ihm langweilig, er möchte weiterlesen, aber das Kopieren nicht unterbrechen (gleiche Bedingungen, d.h. nur ein Shell-Fenster). Wie kann er das erreichen? Geben Sie den Ablauf an. Mindestens welche Prozesse laufen dann auf seinem Rechner?

## Aufgabe 10 (Bonus, 2 Punkte)

Finden Sie mithilfe der angegebenen Quellen heraus sowie eigenen Recherchen, wie die Programmierung zu den drei angegebenen Zeiten praktisch funktioniert hat. Beschreiben Sie in Stichpunkten, wie jeweils die Arbeitsweise eines Programmierers aussah/aussieht, und was sich dabei verändert hat. Wie funktioniert/e jeweils das Schreiben des Codes, wie das Kompilieren, Testen und Debugging?

**Mitte 1950er:** Hochsprachen wie ALGOL[11] oder FORTRAN[12][13] waren bereits eingeführt. Allerdings gab es nur Stapelverarbeitungssysteme; Eingabe nur über Lochkarten und Magnetbändern. Die Videos geben Ihnen eine Vorstellung über die beiden Sprachen.

**ab 1961:** Die ersten Timesharing-Systeme werden eingesetzt. Im Jahr 1964 wird etwa am Dartmouth College das Dartmouth Timesharing System (DTSS) eingeführt[14]. Mit DTSS wurde ein interaktives Arbeiten mit einem Mainframe über ein Kommandozeileninterface (CLI) von einem Terminal aus möglich. Dieses CLI kann als erster Vorläufer moderner integrierter Entwicklungsumgebungen (IDEs) betrachtet werden. Heute wird auf den Webseiten des Dartmouth College ein Emulator[15] angeboten. Dieser Emulator wird nur für Windows und Mac angeboten. Über das URZ haben Sie allerdings Zugriff auf einen Windows-Terminalserver. Werfen Sie einen Blick in das Original-Handbuch aus dem Jahr 1964[16] (Kapitel 2.4 *Use of*

[11] https://www.youtube.com/watch?v=T-NTEc8Ag-I
[12] Teil 1: https://www.youtube.com/watch?v=zyHmxjyzC_g
[13] ... und Teil 2: https://www.youtube.com/watch?v=fRhLP9WKESg
[14] http://dtss.dartmouth.edu/timeline.php
[15] http://dtss.dartmouth.edu/index.php#download
[16] http://www.bitsavers.org/pdf/dartmouth/BASIC_Oct64.pdf

*the Time Sharing System*) und probieren Sie den Emulator aus, um eine Vorstellung für diese Zeit zu bekommen. Dort und in der Archivdatei des Emulators finden sie auch eine Übersicht über die Befehle für die CLI, sowie BASIC-Befehle und Beispiel-Programme.

**heute:** Beschreiben Sie stichpunktartig, wie Sie mit einer modernen IDEs[17] (wie Visual Studio Code, IntelliJ IDEA/Rider, Neovim, Emacs, Atom, ...) programmieren. Welche Schritte sind erleichtert worden? Welche neuen Funktionen sind hinzugekommen, die das Programmieren unterstützen? Was läuft alles automatisch, möglicherweise im Hintergrund, ab?

[17] https://pypl.github.io/IDE.html