Errata zur 3. Auflage von **Betriebssysteme kompakt**.
Erschienen 2022 bei Springer Vieweg. ISBN: 978-3-662-64717-2

Seite 26, 3. und 4. Zeile von Abschnitt 3.4

Ersetze „Mehrprogrammbetrieb“ durch „Mehrbenutzerbetrieb“.

Seite 29, 8. Zeile von unten

Ersetze „Mehrprogrammbetrieb“ durch „Mehrbenutzerbetrieb“.

Seite 61, Abschnitt 4.4.4, 1. Zeile

„Festplatten sind pro Bit. . . “

Im Buch fehlt das Wort „sind“.

Seite 86, Abbildung 5.2

In der letzten Spalte unterhalb von „Prozess A wird beendet und G gestartet“ muss der erste Prozess mit 18 MB Speicherbelegung `G` heißen und nicht `A`.

Seite 94, 2. und 3. Zeile von Abschnitt „Organisation und Adressierung des Speichers im Real Mode“

Ersetze

„Im Real Mode wird der verfügbare Speicher in gleich große Segmente unterteilt. Die Speicheradressen sind 16 Bits lang. Jedes Segment ist dementsprechend 64 Bytes ($= 2^{16} = 65.536$ Bits) groß.“

durch

„Im Real Mode wird der verfügbare Speicher in gleich große Segmente unterteilt. Jedes Segment ist 64 kB groß.“

Seite 100, 2. Zeile von unten

Ersetze „. . . der Grad. . . “ durch „. . . den Grad. . . “.

Seite 110, 4. Zeile von unten

„. . . Speicherschutz mehr bietet ist wegen des. . . “.

Im Buch fehlt das Wort „ist“.

Seite 117, Abschnitt 5.3.5, 2. Zeile

Ersetze „. . . bei die Auswahl. . . “ durch „. . . bei der Auswahl. . . “.

Seite 140, 11. und 12. Zeile des 3. Abschnitts

Streiche im Satz „Allerdings ist auch bei diesem Konzept nur die Konsistenz der Metadaten ~~ist~~ garantiert.“ das hier durchgestrichene Wort.

Seite 142, 5. Zeile von unten

Ersetze „Abb. 6.9“ durch „Abb. 6.11“.

Seite 158

Der Dateiname des Programmbeispiels Listing 7.1 stimmt im Buch nicht mit dem Schema der anderen Programmbeispiele überein. Das kann verwirrend sein.

Ersetze

```
$ gcc SysCallBeispiel.c -o SysCallBeispiel
$ ./SysCallBeispiel
```

durch

```
$ gcc Listing_7_1_Systemcall.c -o Listing_7_1_Systemcall
$ ./Listing_7_1_Systemcall
```

Die Programmbeispiele im Buch sind auch hier verfügbar:

`https://github.com/christianbaun/listings_of_my_text_books/tree/main/Betriebssysteme_kompakt_Auflage_3_Springer_Vieweg_2022`

Seite 162, Abschnitt 8.1

Ersetze:

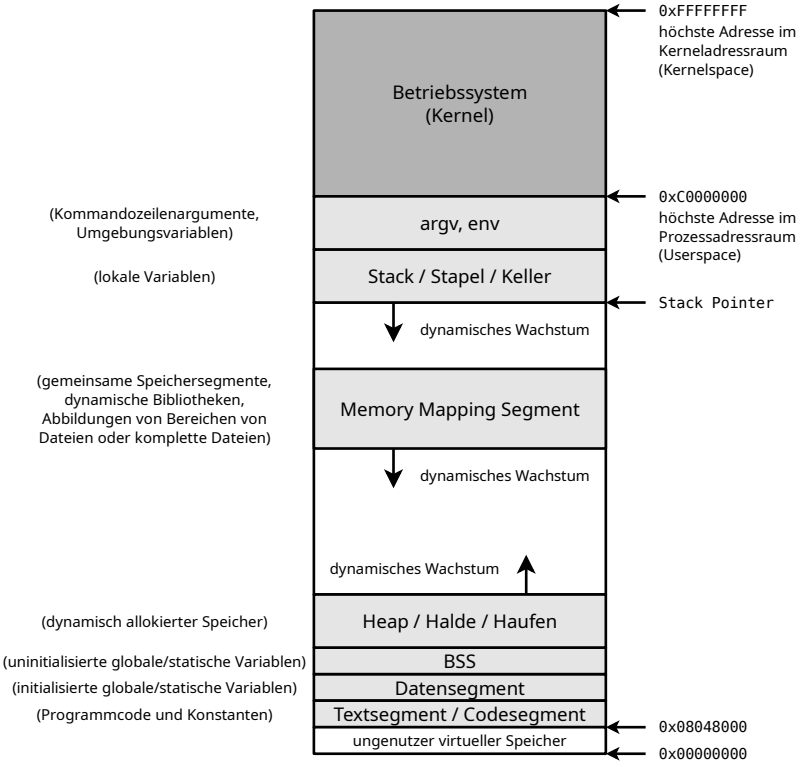
- Information über Eltern- oder Kindprozesse

Durch:

- Elternprozessnummer (PPID)

Seite 170-172, Abschnitt 8.3 (inkl. Abbildung 8.12)

Die Struktur eines Linux-Prozesses auf einem 32-Bit-Systemen im Speicher (wie in Abbildung 8.12) gezeigt ist im Buch nicht korrekt dargestellt.



- Das Textsegment enthält den ausführbaren Programmcode (Maschinencode) und ausschließlich lesbare Daten wie Konstanten.
- Das Datensegment enthält initialisierte Variablen, die entweder global sind oder lokal und zugleich statisch.
- Der Bereich BSS enthält diejenigen globalen Variablen und lokalen statischen Variablen, die beim Start des Prozesses nicht initialisiert werden.
- Der Heap wächst dynamisch. Hier kann ein Prozess dynamisch zur Laufzeit Speicher allokieren (mit `malloc`). Der Heap kann im Gegensatz zum Textsegment, Datensegment und BSS während der Laufzeit eines Programms wachsen.
- Kommandozeilenargumente (`argv`) des Programmaufrufs und die Umgebungsvariablen (`env`) liegen in einem Bereich, der am Ende des Userspace beginnt.
- Der Stack ermöglicht die Realisierung geschachtelter Funktionsaufrufe und arbeitet nach dem Prinzip LIFO. Mit jedem Funktionsaufruf wird eine Datenstruktur auf den Stack gelegt, die die Aufrufparameter, die Rücksprungadresse und einen Zeiger auf die aufrufende Funktion im Stack enthält. Die Funktionen legen auch ihre lokalen Variablen auf den Stack. Beim Rücksprung aus einer Funktion wird die Datenstruktur der Funktion aus dem Stack entfernt. Der Stack kann also während der Laufzeit eines Programms wachsen.

- Im Speicherbereich Memory Mapping, der sich im Adressraum zwischen Stack und Heap befindet, werden dynamische Bibliotheken (*Shared Libraries*) geladen. Auch gemeinsame Speicherbereiche sind hier abgebildet. Zusätzlich können komplette Dateien oder Bereiche von Dateien hier mit dem Systemaufruf `mmap` abgebildet werden.

Diese Anpassungen betreffen auch die entsprechenden Einträge im Glossar.

Eine ausführlichere und korrekte Darstellung der Struktur eines Linux-Prozesses auf einem 32-Bit-Systemen im Speicher enthält Auflage 4 des Buches.

Seite 177, Aufruf das Kommandos `taskset`

Ersetze

```
$ taskset --cpu-list 1 ./Listing_8.2_fork
```

durch

```
$ taskset --cpu-list 1 ./Listing_8_2_fork
```

Seite 182, vorletzte Zeile

Ersetze „... von der eine...“ durch „... von denen eine...“.

Seite 245, Listing 9.5, Zeile 8 im Quellcode

Die öffnende geschweifte Klammer am Ende der Zeile gehört hier nicht hin. Die Zeile soll sein:

```
int testpipe[2];
```

Seite 248, Listing 9.6, Zeile 11 im Quellcode

Eine ausführliche Erklärung zu `mkfifo` und den Zugriffsrechten wäre an der Stelle im Buch sinnvoll gewesen, da auf den ersten Blick die Zugriffsrechte der benannten Pipe nicht zum Quellcode passen.

In Listing 9.6 wird mit `mkfifo` eine benannte Pipe `testfifo` angelegt. Als Zugriffsrechte sind `0666` definiert. Die führende `0` kann hier ignoriert werden. Sie ist ein Platzhalter für das sogenannte Sticky-Bit, das Setgid Bit und das Setuid Bit. Diese erweiterten Dateirechte kommen eher selten zum Einsatz und spielen im Kontext von Listing 9.6 keine Rolle. Die Bedeutung der führenden Null bei der Oktalnotation mit vier Ziffern kann also hier ignoriert werden.

Die Zugriffsrechte der resultierende Pipe sind auf Seite 247 aber in der symbolischen Notation mit `rw-r--r--` angegeben, was in Oktalnotation `644` entspricht. Auf Ubuntu-basierten Systemen wird das Ergebnis hingegen in der symbolischen

Notation `rw-rw-r--` sein, was in Oktalnotation `664` entspricht. Auch ganz andere Ergebnisse sind je nach verwendetem Betriebssystem und vorgenommenen Einstellungen möglich.

Der Grund dafür ist, dass auf dem System die mit `umask` („Dateierzeugungsmaske“) gesetzten Zugriffsrechte entfernt („maskiert“) werden. Die Standardeinstellung von `umask` hängt vom verwendeten Betriebssystem ab und kann vom Systemadministrator verändert werden. Die `umask`-Standardwerte der Linux-Distributionen Debian sind Ubuntu sind z.B. `0022` bzw. `0002`.

Die aktuell eingestellte Dateierzeugungsmaske kann durch einen Aufruf des Kommandos `umask` ohne Parameter in der Kommandozeile ausgegeben werden:

```
$ umask
0022
```

Hat `umask` den Wert `0022` (auch hier kann die führende `0` ignoriert werden) sind die Zugriffsrechte der benannte Pipe aus Listing 9.6 `rw-r--r--`. Die Berechnung ist wie folgt:

Definierte Zugriffsrechte in `mkfifo` in Listing 9.6:	`rw-rw-rw-`	(666)
Abzug durch `umask` auf einem System mit Debian-Linux:	`----w--w-`	(022)
Ergebnis (Zugriffsrechte der benannten Pipe):	`rw-r--r--`	(644)

Auf einem System, bei dem `umask` den Wert `0002` hat, sind die Zugriffsrechte der benannten Pipe dementsprechend `rw-rw-r--`.

Weitere Informationen zum Thema `umask` und Zugriffsrechte sind u.a. hier zu finden:

- `https://wiki.ubuntuusers.de/umask/`
- `https://www.debian.org/doc/manuals/debian-reference/ch01.en.html#_control_of_permissions_for_newly_created_files_umask`

Seite 248, Listing 9.6, Zeile 20 im Quellcode

Die öffnende geschweifte Klammer am Ende der Zeile gehört hier nicht hin. Die Zeile soll sein:

```
pid_des_Kindes = fork();
```

Seite 260, Beispiel zu netstat

In der zweiten Zeile der Ausgabe des Kommandozeilenwerkzeugs `netstat` ist der Inhalt der letzten Spalte falsch. Die Ausgabe muss wie folgt sein:

```
$ netstat -tap | grep 50003
tcp  0 0  0.0.0.0:50003    0.0.0.0:*        LISTEN     119954/./Listing_9_
tcp  0 0  localhost:50003  localhost:59716  VERBUNDEN  119954/nc
tcp  0 0  localhost:59716  localhost:50003  VERBUNDEN  119982/./Listing_9_
```

Seite 265, Glossar, zweiter Eintrag

Ersetze „Adressbus“durch „AES“.

Seite 306, Glossar, Eintrag von Datensegment, 2. Zeile

Streiche „. . . und Konstanten“.