Errata zur 1. Auflage von **Operating Systems / Betriebssysteme – Bilingual Edition / Zweisprachige Ausgabe**.
Erschienen 2020 bei Springer Vieweg. ISBN: 978-3-658-29784-8

Seite 4, Tabelle 2.1, erste Zeile, mittlere Spalte

Ersetze $2^9 = 12$ durch $2^9 = 512$.

Seite 23, 4. und 5. Zeile von Abschnitt 3.4

Ersetze „Mehrprogrammbetrieb“ durch „Mehrbenutzerbetrieb“.

Seite 26, 3. und 4. Zeile von unten

Ersetze „Mehrprogrammbetrieb“ durch „Mehrbenutzerbetrieb“.

Seite 35, Abbildung 3.10

Ersetze „Inter Prozess Communication“ durch „Interprocess Communication“.

Seite 35, Bildunterschrift von Abbildung 3.10

Ersetze „or“ durch „of“.

Seite 55, rechte Spalte, Abschnitt 4.4.5, 1. Zeile

„Festplatten sind pro Bit. . . “

Im Buch fehlt das Wort „sind“.

Seite 58, Abschnitt 4.4.7

Ersetze am Anfang des zweiten Aufzählungspunkts „Die Zugriffsverzögerung“ durch „Die durchschnittliche Zugriffsverzögerung“.

Seite 58, Abschnitt 4.4.7

Ersetze die Formel durch:

$$\text{Average Rotational Latency Time [ms]} = \frac{1000\frac{[\text{ms}]}{[\text{sec}]} \times 60\frac{[\text{sec}]}{[\text{min}]} \times 0.5}{\frac{\text{revolutions}}{[\text{min}]}} = \frac{30,000\frac{[\text{ms}]}{[\text{min}]}}{\frac{\text{revolutions}}{[\text{min}]}}$$

Seite 85, 1. bis 4. Zeile von Abschnitt „Organization and Addressing of Memory in Real Mode“ (linke Spalte)

Ersetze

„Real mode splits the available memory into segments of equal size. The memory address length is 16 bits. Therefore, the size of each segment is 64 bytes ($= 2^{16} = 65,536$ bits).“

durch

„Real mode splits the available memory into segments of equal size. the size of each segment is 64 kB.“

Seite 85, 1. bis 4. Zeile von Abschnitt „Organisation und Adressierung des Speichers im Real Mode“ (rechte Spalte)

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 92, rechte Spalte, 6. Zeile von unten

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

Seite 101, rechte Spalte, Abschnitt 5.2.5, 6. Zeile von unten

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

Im Buch fehlt das Wort „ist“.

Seite 108, rechte Spalte, Abschnitt 5.3.5, 2. Zeile

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

Seite 126, Tabellenüberschrift von Tabelle 6.4

Ersetze

„Maximum File System Size of FAT32 for Clusters of different Size“.

durch

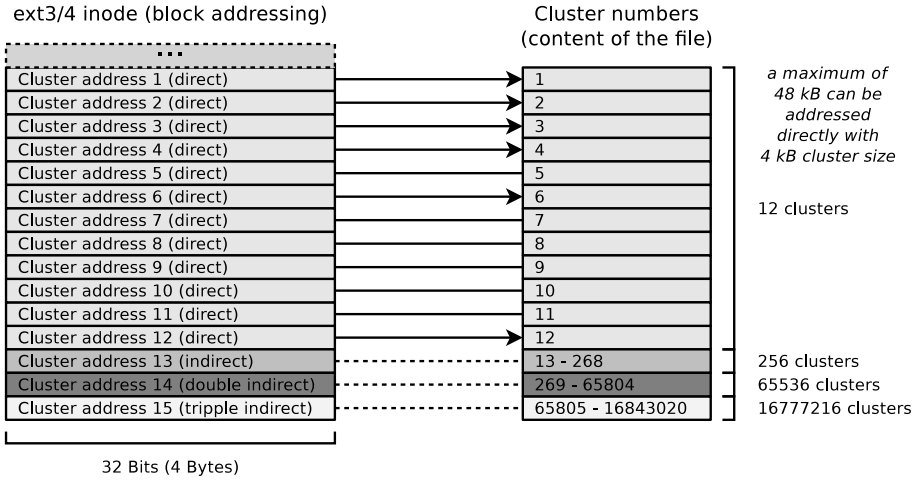
„Default Cluster Size of FAT32 for different Partition Sizes“.

Seite 128, rechte Spalte, 4. und 3. Zeile von unten

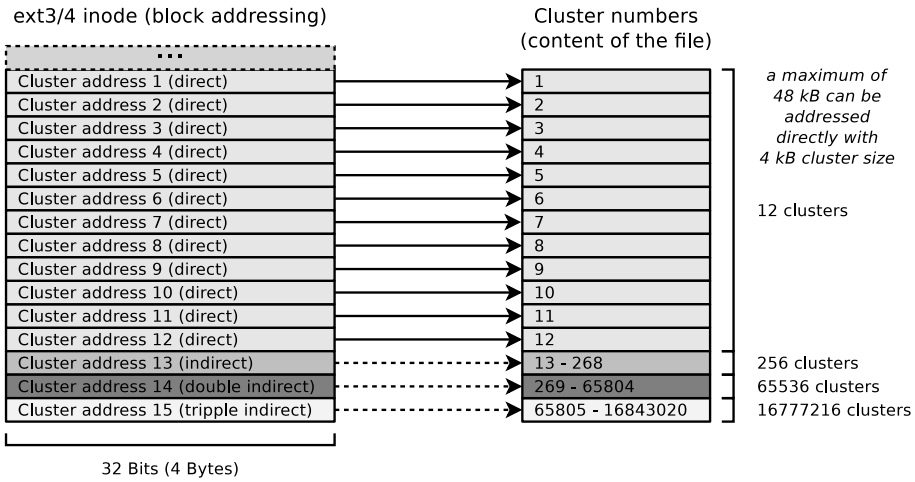
Streiche im Satz „Allerdings ist auch bei diesem Konzept nur die Konsistenz der Metadaten ~~ist~~ garantiert.“ das hier durchgestrichene Wort.

Seite 130, Abbildung 6.8

In der Abbildung im Buch fehlen einige Pfeilspitzen.



In der folgenden Abbildung sind die Pfeilspitzen komplett.



Seite 131, linke Spalte, 8. Zeile

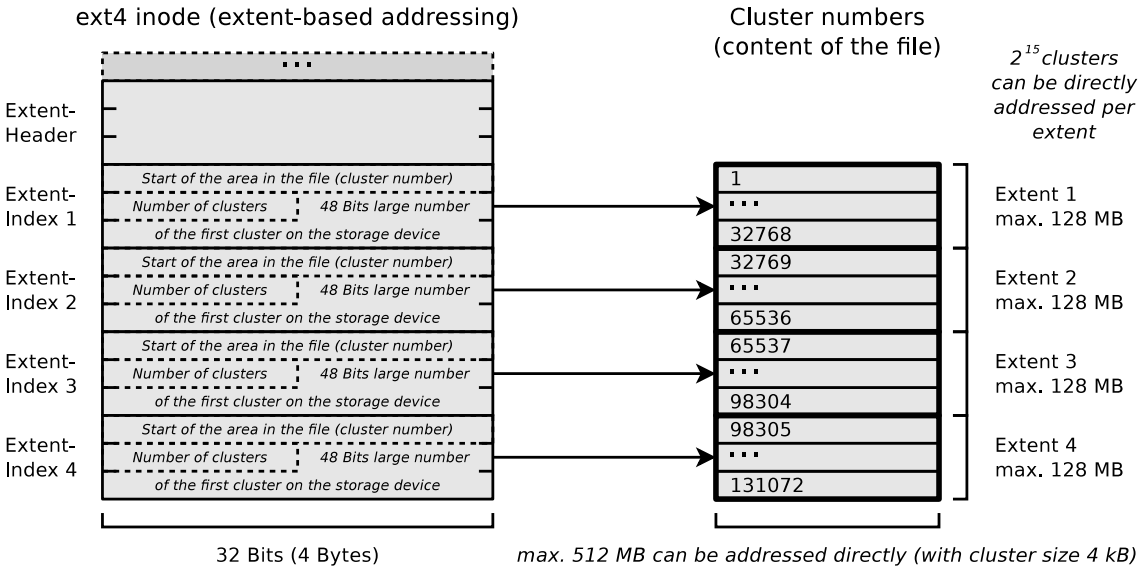
Ersetze „Figure 6.8“ durch „Figure 6.10“.

Seite 131, rechte Spalte, 10. Zeile

Ersetze „Abbildung 6.8“ durch „Abbildung 6.10“.

Seite 131, Abbildung 6.10

Die Beschriftung im Inode ist nicht korrekt und im Wort „addressing“ über dem Inode fehlt das „n“.



Seite 133, Tabellenüberschrift von Tabelle 6.5

Ersetze

„Maximum File System Size of NTFS for Clusters of different Size“.

durch

„Default Cluster Size of NTFS for different Partition Sizes“.

Seite 146, linke Spalte, Abschnitt 8.1

Ersetze:

- Information about parent or child processes

Durch:

- Parent Process ID (PPID)

Seite 146, rechte Spalte, Abschnitt 8.1

Ersetze:

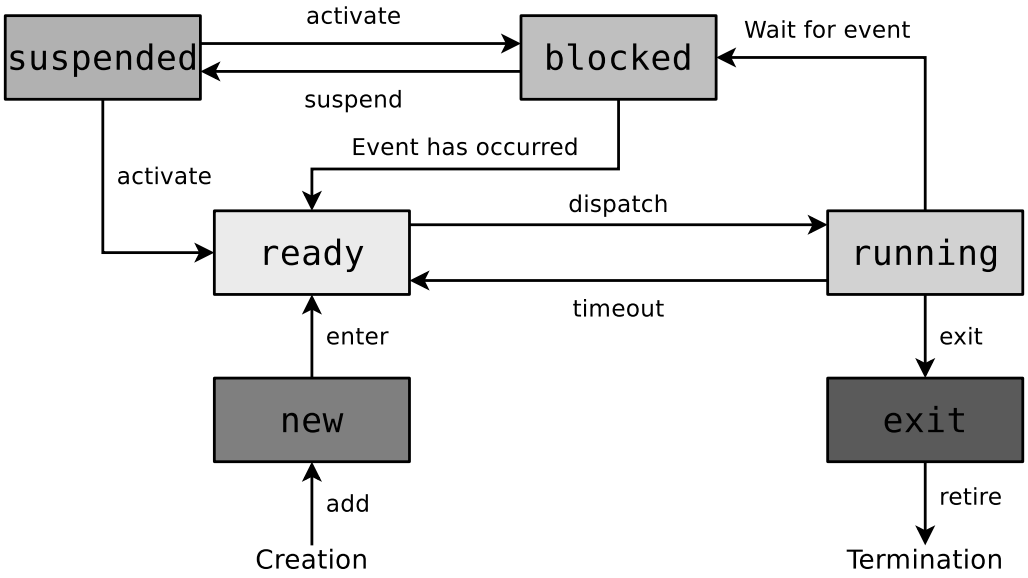
- Information über Eltern- oder Kindprozesse

Durch:

- Elternprozessnummer (PPID)

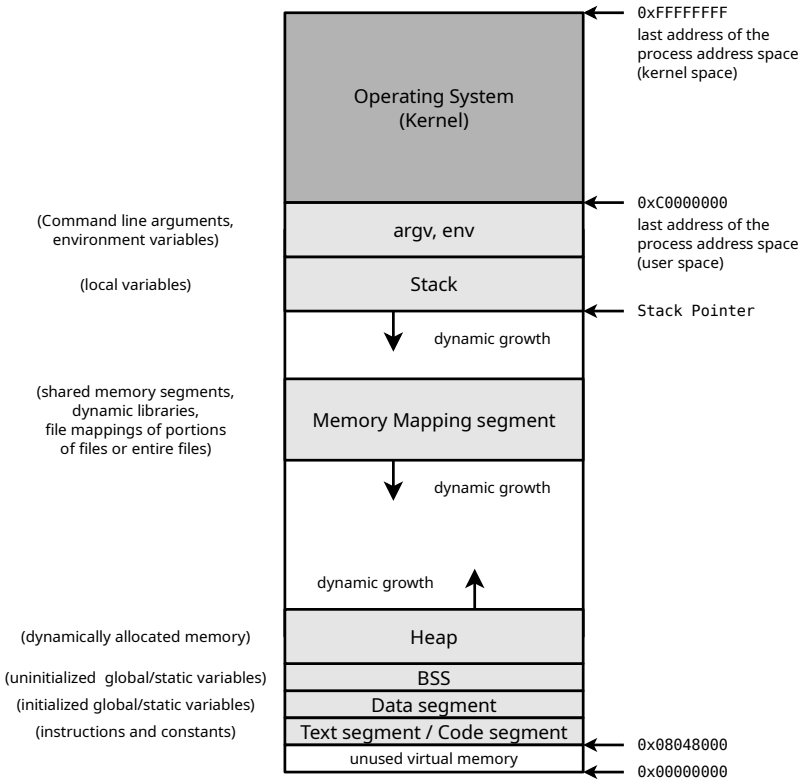
Seite 152, Abbildung 8.8

Im 6-Zustands-Prozessmodell in Abbildung 8.8 fehlt ein Prozessübergang `activate` von Prozesszustand `suspended` zu Prozesszustand `blocked`.



Seite 154-156, 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 3 des Buches.

Seite 164, rechte Spalte, vorletzte Zeile

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

Seite 186, Abbildung 9.8

Unterhalb der Operation `unlock(s)` ist „`process`“ zwei mal falsch geschrieben als „`prozess`“.

Seite 202, 2. Zeile des dritten Absatzes

Ersetze „Konversion“ durch „Konvertierung“.

Der Begriff „Konversion“ wird in vielen Bereichen (u.a. Religion, Stadtentwicklung und Konversion) verwendet, aber in der Informatik und ganz besonders im Kontext verschiedener Stellenwertsystem ist „Konvertierung“ der korrekte Fachbegriff.

Seite 202, vorletzter Absatz

Ersetze „Konversion“ durch „Konvertierung“.

Seite 209, Listing 9.4, 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 9.4 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 9.4 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 209 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` 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 211, letzter Absatz

Ersetze

„Das Kommando `lsof` gibt in einem Linux-Betriebssystem eine Liste aller existierenden und von mindestens einem Prozess verwendeten benannten Pipes aus.“

durch

„Das Kommando `lsof` gibt in einem Linux-Betriebssystem eine Liste aller aktuell offenen Dateien, also auch die existierenden benannten Pipes aus.“

Seite 234, letzte Zeile des ersten Absatzes (linke Spalte)

Ersetze „ring 1“ durch „ring 0“.

Seite 234, vorletzte Zeile des ersten Absatzes (rechte Spalte)

Ersetze „Ring 1“ durch „Ring 0“.

Seite 244, Glossar, linke Spalte, Eintrag von Data Segment 3. Zeile

Streiche „. . . and constants“.

Seite 244, Glossar, rechte Spalte, Eintrag von Datensegment, 2. und 3. Zeile

Streiche „. . . und Konstanten“.