Betriebssysteme Blatt 9

Baran Güner, bg160 Tobias Hangel, th151

23. Dezember 2022

Aufgabe 1

a)

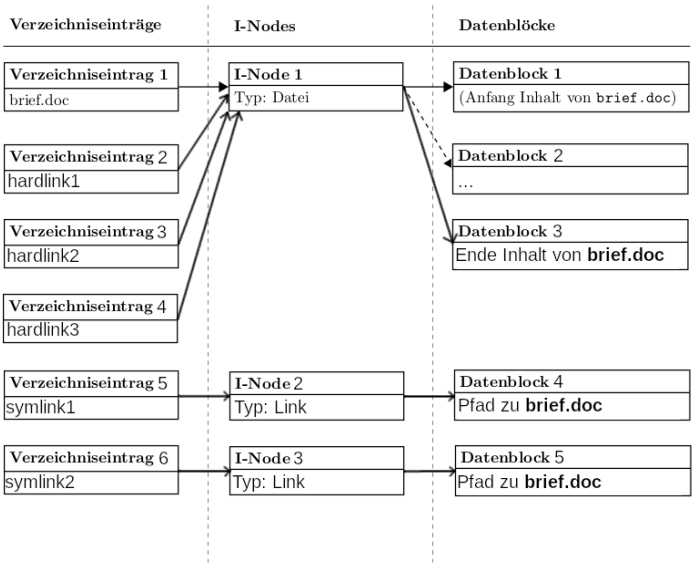
Jeder direkte Zeiger verweist auf einen Block, der Größe 4 KiB = 4096 Byte.
50.000/4096 = 12.20. Das Byte n befindet sich somit in Dateiblock 12 (Dreizehnter Block). Die I-Node enthält 10 direkte Zeiger. Es muss also der einfach indirekte Zeiger verwendet werden, welcher auf 256 weitere Zeiger zeigt. Der dritte dieser Zeiger zeigt dann auf Dateiblock 12. Das 0.Element von Dateiblock 12 ist Byte Nummer 49151 der Datei. Der Zeiger für Byte n zeigt somit auf Element 849 des Blocks.

b)

Im Hauptspeicher wird die File Allocation Table angelegt.
In den Verzeichniseinträgen befindet sich die Adresse des Plattenblocks, in der der Beginn der Datei gespeichert ist. Jeder FAT Eintrag eines Plattenblocks verweist auf den nächsten Plattenblock, in dem der nächste Abschnitt der Datei gespeichert ist.
Somit muss für den wahlfreien Zugriff eine Kette von Verweisen verfolgt werden. $\lfloor n/b \rfloor$ ist die Nummer des Plattenblocks, in der sich Byte n befindet.
Zuerst muss der Block angeschaut werden, in dem der Anfang der Datei liegt und danach müssen $\lfloor n/b \rfloor - 1$ Verweise verfolgt werden.
Die Zugriffszeit müsste also linear zu n wachsen $(n/b) \in O(n)$.

Aufgabe 2

a)



Der Inhalt von brief.doc hört bei Datenblock 3 auf, da 2,5KB Daten gespeichert werden müssen und jeder Block 1KB groß ist.

b)

Jeder Hardlink benötigt einen eigenen Verzeichniseintrag, aber keine eigenen I-Nodes, da sie auf die selbe I-Node wie brief.doc verweisen.
Jeder Symlink benötigt einen Verzeichniseintrag der auf eine eigene I-Node hinweist.

brief.doc ist 2,5 KiB = 2560 Byte groß. Die Datei benötigt somit drei Blöcke.
Die Hardlinks benötigen nur den Speicher, der für die zusätzlichen drei Verzeichniseinträge gebraucht wird.
Die Softlinks benötigen den Speicher, der für die zwei Verzeichniseinträge, I-Nodes und Datenblöcke gebraucht wird.

Somit werden 5 Datenblöcke verwendet.

c)

Chmod verändert nichts an den symbolischen Links, da diese ja nur einen Verweis auf eine Datei beinhalten und Nutzerrechte über die I-Node der Dateien

die auf diese verweisen verwaltet werden sollten.
Eine Anpassung eines Hardlinks verändern alle Hardlinks, da sie alle auf die selbe I-Node weisen in welcher die Zugriffsrechte gespeichert sind.

d)

Die Rechte des symbolischen Links verändern sich nicht. Das ist sinnvoll, da der Link selbst keine Recht besitzt.
Die Rechte werden durch die I-Node der Datei geregelt, auf die der Link zeigt.

Aufgabe 3

a)

Mit

```
ps aux
```

werden alle Prozesse gezeigt.

b)

```
kill -s sigstop <pid>
kill -s sigcont <pid>
kill -s sigterm <pid>
kill -s sigkill <pid>
```

<pid> ist hierbei die ID des Prozesses.

Wenn einfach nur

```
kill <pid>
```

genutzt wird erhält der Prozess standartgemäß das SIGKILL Signal, daher ist hier die option redundant.

c)

SIGTERM terminiert einen Prozess, das Signal kann aber im Code verarbeitet werde. Das kann dann zum Beispiel verwendet werden, um Variablen in einer Datei abzuspeichern und dann manuell das Program zu beenden, statt es sofort zu terminieren.
SIGKILL terminiert ebenfalls einen Prozess, kann jedoch nicht verarbeitet werden.

SIGSTOP pausiert den Prozess und SIGCONT setzt einen pausierten Prozess an der unterbrochenen Stelle fort. Im Beispiel der Aufgabe wird so der Wert von x beibehalten und es wird weitergezählt.

d)

mit

```
screen counter.sh
```

wird counter.sh als screen geöffnet.

So kann mit den Eingaben **Ctrl** + **a** und anschließend **d** der screen *detached* werden, das heißt, der Prozess ist vom Terminal gelöst und läuft im Hintergrund weiter.

Wenn man den Prozess dann wieder im Terminal haben will kann man ein neues Terminal öffnen,

```
screen -list
```

eingeben um alle screen Prozesse anzuzeigen, um dann mit

```
screen -r <Prozessname>
```

den richtigen Prozess fortzuführen.