Institut für Informatik

Arbeitsgruppe Autonome Intelligente Systeme

Freiburg, 10. November 2015

Systeme I: Betriebssysteme
Übungsblatt 3

Aufgabe 1 (1,5 Punkte)

Betrachten Sie die Befehle `du`, `df`, `mount`. Lesen Sie die `man`pages der Befehle und versuchen Sie, ihre Ausgaben zu verstehen.

Finden Sie heraus, mit welchem Befehl oder mit welchen Befehlen (evtl. mit zusätzlichen Parametern) Sie folgende Informationen erhalten können:

a) Belegter Speicherplatz Ihres Homeverzeichnisses mit leicht verständlicher Einheit

b) Welche Dateisysteme an welchen Positionen des Verzeichnisbaumes aktuell eingehängt sind und welchen Typ diese Dateisysteme haben

c) Anzahl der inodes eines Dateisystems

Aufgabe 2 (1+0,5 Punkte)

Es ist in der Praxis üblich, die Größe einer Datenmenge in KB (auch KByte, „Kilobyte“), MB (auch MByte, „Megabyte“) oder GB (auch GByte, „Gigabyte“) anzugeben.

Anders als sonst bezeichnet ein Kilo hier aber in der Regel nicht 1000, sondern $2^{10} = 1024$ Einheiten, ein Mega nicht 1000000, sondern $2^{20} = 1048576$ Einheiten usw. Ein KB entspricht daher genau 2^{10} Byte, ein MB entspricht 2^{20} Byte und ein GB entspricht 2^{30} Byte. Die Notation und Interpretation ist jedoch oft uneinheitlich.

Es gibt Normierungsbestrebungen, die versuchen, die in diesem Zusammenhang häufig auftretende Verwirrung zu beseitigen. So wurden von der IEC 1998 die Bezeichnungen KiB, MiB etc. eingeführt (vgl. `https://de.wikipedia.org/wiki/Binärpräfix`):

SI-Dezimalpräfixe		
k	Kilo	$10^3 = 1.000$
M	Mega	$10^6 = 1.000.000$
G	Giga	10^9
T	Tera	10^{12}

IEC-Binärpräfixe		
Ki	Kibi	$2^{10} = 1.024$
Mi	Mebi	$2^{20} = 1.048.576$
Gi	Gibi	2^{30}
Ti	Tebi	2^{40}

Wir verwenden bei der Angabe von Datei- und Festplattengrößen KiB, MiB und GiB im Sinne von 2^{10}, 2^{20} bzw. 2^{30} Byte.

a) Vervollständigen Sie folgende Tabelle, indem Sie die Größenangaben in Bit und Byte (=8 Bit) umrechnen und in 2er-Potenzen und in Dezimaldarstellung angeben.

Angabe	Angabe in Bits		Angabe in Bytes	
	2er-Potenz	dezimal	2er-Potenz	dezimal
1 Byte	2^3 Bit	8 Bit	2^0 Byte	1 Byte
2048 MiB				
32 Byte				
16 MiBit				
1024 KiBit				

b) Sie haben eine „3,0 TB“-Festplatte gekauft. Was glauben Sie, welche Interpretation von „TB“ der Festplattenhersteller gewählt hat? Wie groß ist der Unterschied (in GiB) zwischen den Interpretationen?

Aufgabe 3 (1+1 Punkte)

In der Vorlesung wurde als eine mögliche Anordnung von Dateien auf der Festplatte die *zusammenhängende Belegung* vorgestellt. Die Blöcke (je 32KiB) einer Festplatte seien mit den Dateien „1“, „2“ und „3“ wie folgt belegt (vollständige Belegung mit • und teilweise Belegung mit ◦ gekennzeichnet):

1•	1 •	1 •						3 •	3 •	3 ◦		2 •	2 •	2 •	2 •

Neue Dateien sollen immer am Anfang der ersten passenden Lücke angelegt werden.

a) Ein Benutzer schreibt eine neue Datei „4“ mit einer Größe von 144 KiB. Da nur ganze Blöcke belegt werden können, weist das Betriebssystem 5 Blöcke mit je 32 KiB zu. Schreiben Sie die neue Belegung auf. Wie nennt man die entstandene Fragmentierung am Ende der Datei „4“? Ist dieser Speicherplatz noch nutzbar?

b) Es wird nun Datei „1“ gelöscht. Schreiben Sie die neue Belegung auf. Wie nennt man die entstandene Fragmentierung?
Wie viele Blöcke sind noch frei? Ist es möglich, eine Datei der Größe 100 KiB zu erstellen?

Aufgabe 4 (2+1 Punkte)

In der Vorlesung haben Sie das FAT32-Dateisystem kennengelernt. Es sei eine FAT-Tabelle mit einer Blockgröße von 32 KiB gegeben. Zusätzlich steht eine Liste aller freien Plattenblöcke und eine Tabelle mit den Verzeichniseinträgen zur Verfügung:

FAT:

Plattenblock 0	
Plattenblock 1	
Plattenblock 2	10
Plattenblock 3	11
Plattenblock 4	7
Plattenblock 5	
Plattenblock 6	3
Plattenblock 7	2
Plattenblock 8	
Plattenblock 9	
Plattenblock 10	12
Plattenblock 11	14
Plattenblock 12	−1
Plattenblock 13	
Plattenblock 14	−1
Plattenblock 15	
⋮	⋮

Liste freier Plattenblöcke:

15	13	1	8	9	5	0	...

Verzeichniseinträge:

Dateiname	Erweiterung	Datei-Attribute	Erster Plattenblock	Dateigröße
`BRIEF`	`TXT`	(...)	`4`	`129 KiB`
`EDITOR`	`EXE`	(...)	`6`	`101 KiB`
⋮	⋮	⋮	⋮	⋮

a) Der Benutzer legt eine neue Datei „`AUFGABE.DOC`“ mit einer Dateigröße von 158 KiB an. Geben Sie die FAT, die Liste der freien Plattenblöcke und die Tabelle der Verzeichniseinträge nach dem Anlegen der Datei an. Die freien Blöcke werden in der Reihenfolge belegt, wie sie in der Liste der freien Plattenblöcke gegeben ist.

b) In der Vorlesung wurde bereits die Größe der FAT32 berechnet. Berechnen Sie nun die maximale Größe des Dateisystems bei einer maximalen Blockgröße von 32 KiB. Geben Sie die Größe dabei in einer leicht lesbaren Einheit an.
Zur Erinnerung: FAT32 bietet 28 Bit zur Addressierung von Blöcken (von den eigentlich vorhandenen 32 Bit sind 4 Bit für andere Zwecke reserviert).

Aufgabe 5 (2,5+2+1,5 Punkte)

Unter Linux können Sie Dateisysteme auf einem *block device* (z.B. einer Festplattenpartition) mit dem Befehl `mkfs.xyz` erstellen, wobei `xyz` für das Dateisystem steht. In dieser Aufgabe verwenden wir der Einfachheit halber eine Datei[1] als Speichermedium, die wir mit Nullen füllen, und erzeugen dann eine FAT12-Partition.

Listing 1: Dateisystem erstellen

```
$ dd if=/dev/zero of=blockdata.img bs=1K count=4096
$ mkfs.fat -F 12 -s 4 -S 512 -v blockdata.img
```

Dieses Dateisystem können wir nun einhängen[2] und mit zwei Dateien füllen (vgl. `man`-Page zu `dd`).

Listing 2: Mounten und Dateien auf Dateisystem erstellen

```
$ mkdir my_mountpoint
$ sudo mount -o uid=$(whoami) blockdata.img my_mountpoint/
$ dd if=/dev/zero of=my_mountpoint/vielenullen_0.bin bs=1K count=7
$ dd if=/dev/zero of=my_mountpoint/vielenullen_1.bin bs=1M count=2
$ ls -lh blockdata.img
-rw-rw-r-- 1 henkolk henkolk 4,0M Nov  6 16:39 blockdata.img
$ du my_mountpoint/
2072    my_mountpoint/
```

Nach dem Aushängen des Dateisystems führen wir nun einen Dateisystem-Check durch und lassen uns dabei zusätzliche Informationen (`-v`) sowie die Dateien (`-l`) ausgeben.

Listing 3: Dateisystemüberprüfung mit fsck

```
$ sudo umount my_mountpoint/
$ fsck.fat -vl blockdata.img
fsck.fat 3.0.28 (2015-05-16)
Checking we can access the last sector of the filesystem
Boot sector contents:
System ID "mkfs.fat"
Media byte 0xf8 (hard disk)
       512 bytes per logical sector
      2048 bytes per cluster
         1 reserved sector
First FAT starts at byte 512 (sector 1)
         2 FATs, 12 bit entries
  FAT_SIZE bytes per FAT (= [...] sectors)
Root directory starts at byte [...] (sector [...])
       512 root directory entries
Data area starts at byte 23040 (sector 45)
      2036 data clusters (4169728 bytes)
32 sectors/track, 64 heads
         0 hidden sectors
      8192 sectors total
Checking file /vielenullen_0.bin (VIELEN~1.BIN)
Checking file /vielenullen_1.bin (VIELEN~2.BIN)
Checking for unused clusters.
blockdata.img: 2 files, USED_CLUSTERS/2036 clusters
```

Hierbei berechnet sich die Blockgröße (“cluster size”) aus der Anzahl der Sektoren pro Block multipliziert mit der Sektorengröße.

[1] Unter Linux werden auch Festplattenpartitionen über `/dev/sdXN` in Form einer Datei angesprochen, z.B. `mkfs.ext4 /dev/sdb2`

[2] Da nur Benutzer mit Rootrechten den Befehl `mount` ausführen dürfen, müssen Sie den Code nicht ausführen und können die Aufgabe mit Hilfe der hier angegebenen Ausgaben lösen.

Beantworten Sie basierend auf den obigen Befehlen die folgenden Aufgaben:

a) Wir führen die Listings 1, 2 und 3 aus.

   1) Berechnen Sie die Größe `FAT_SIZE` (Zeile 13) einer einzelnen *file allocation table* (FAT). Was für eine Funktion könnte die zweite FAT besitzen?
   2) Berechnen Sie die Anzahl der belegten Blöcke `USED_CLUSTERS` (Zeile 24) an. Zu welchen Arten von Fragmentierung in welcher Größe kommt es?

   Bemerkung. Die FAT beinhaltet auch Einträge für Sektoren, in denen keine Daten liegen. So sind beispielsweise auch Einträge für Sektoren vorhanden, in denen die FAT selbst liegt.

b) In welchem Verhältnis stehen nach Ausführen von Listings 1 und 2 die Nutzdaten (tatsächliche Dateiinhalte) zum "Overhead" des FAT12-Dateisystems?
   Betrachten Sie den Befehl `$ dd if=/dev/urandom of=my_mountpoint/zufall.bin bs=2K count=MAX_COUNT`, den wir nach Listing 2 ausführen könnten. Welchen Wert müssen Sie für `MAX_COUNT` wählen, um dieses Verhältnis zu maximieren?

   Bemerkung. Als *Overhead* bezeichnen wir in dieser Aufgabe Bereiche des Speichermediums, die für die Verwaltung benötigt werden und somit nicht für Datei*inhalte* zur Verfügung stehen. Wir zählen freien Speicher nicht zum Overhead.

c) Wir betrachten ein FAT12-Dateisystem mit fester Mediengröße und fester Sektorengröße. Können Sie das Verhältnis von Nutzdaten zur Größe des Speichermediums bei der Dateisystemerstellung beeinflussen? Falls ja, welche Nachteile müssen Sie dafür in Kauf nehmen?

Abgabe: Als PDF-Datei über Ilias bis 16. November 2015, 23:59:00 Uhr.