

Lösung von Übungsblatt 6

Aufgabe 1 (Dateisysteme)

1. Geben Sie an, welche Informationen ein Inode speichert.

Speichert die Verwaltungsdaten (Metadaten) einer Datei, außer dem Dateinamen.

2. Nennen Sie drei Beispiele für Metadaten im Dateisystem.

Metadaten sind u.a. Dateigröße, UID/GID, Zugriffsrechte und Datum.

3. Beschreiben Sie, was ein Cluster im Dateisystem ist.

Dateisysteme adressieren Cluster und nicht Blöcke des Datenträgers. Jede Datei belegt eine ganzzahlige Menge an Clustern.

4. Beschreiben Sie, wie ein UNIX-Dateisystem (z.B. ext2/3), das keine Extents verwendet, mehr als 12 Cluster adressiert.

Durch indirekte Adressierung über zusätzliche Cluster, die ausschließlich Cluster-Nummern enthalten.

5. Beschreiben Sie, wie Verzeichnisse bei Linux-Dateisystemen technisch realisiert sind.

Verzeichnisse sind nur Text-Dateien, die die Namen und Inodes von Dateien enthalten.

6. Nennen Sie einen Vorteil und einen Nachteil kleiner Cluster im Dateisystem im Gegensatz zu großen Clustern.

*Vorteil: Weniger Kapazitätsverlust durch interne Fragmentierung.
Nachteil: Mehr Verwaltungsaufwand für große Dateien.*

7. Geben Sie an, ob DOS/Windows-Dateisysteme Groß- und Kleinschreibung unterscheiden.

Ja Nein

8. Geben Sie an, ob UNIX-Dateisysteme Groß- und Kleinschreibung unterscheiden.

Ja Nein

9. Moderne Betriebssysteme beschleunigen Zugriffe auf gespeicherte Daten mit einem Cache im Hauptspeicher.

Ja Nein

10. Die meisten Betriebssystemen arbeiten nach dem Prinzip...

Write-Back Write-Through

11. Nennen Sie je einen Vorteil und einen Nachteil eines Caches im Hauptspeicher, mit dem Betriebssysteme die Zugriffe auf gespeicherte Daten beschleunigen.

Vorteil: Höhere System-Geschwindigkeit.

Nachteil: Stürzt das System ab, kann es zu Inkonsistenzen kommen.

12. Beschreiben Sie was ein absoluter Pfadname ist.

Ein kompletter Pfad von der Wurzel bis zum Ziel (Datei oder Verzeichnis).

13. Beschreiben Sie was ein relativer Pfadname ist.

Ein Pfad, der nicht mit der Wurzel beginnt.

14. `/var/log/messages` ist ein...

Absoluter Pfadname Relativer Pfadname

15. `BTS_Vorlesung/Vorlesung_05/folien_bts_vorlesung_05.tex` ist ein...

Absoluter Pfadname Relativer Pfadname

16. `Dokumente/MasterThesis/thesis.tex` ist ein...

Absoluter Pfadname Relativer Pfadname

17. `/home/<benutzername>/Mail/inbox/` ist ein...

Absoluter Pfadname Relativer Pfadname

18. Nennen Sie die Information, die der Bootsektor (auch genannt Bootblock) eines Dateisystems speichert.

Im Bootsektor (Bootblock) liegen ausführbarer Maschinencode („Boot-Loader“), der das Betriebssystem starten soll und bei einigen Dateisystemen wie FAT12/16/32 auch Informationen über das Dateisystem.

19. Nennen Sie die Information, die der Superblock eines Dateisystems speichert.

Er enthält Informationen über das Dateisystem, z.B. Anzahl der Inodes und Cluster.

20. Erklären Sie warum manche Dateisysteme (z.B. ext2/3) die Cluster des Dateisystems zu Blockgruppen zusammenfassen.

Die Inodes (Metadaten) liegen dadurch physisch nahe bei den Clustern, die sie adressieren.

21. Beschreiben Sie, was die Dateizuordnungstabelle bzw. File Allocation Table (FAT) ist und welche Informationen diese enthält.

Für jeden Cluster des Dateisystems existiert in der FAT ein Eintrag mit folgenden Informationen über den Cluster:

- *Cluster ist frei oder das Medium an dieser Stelle beschädigt.*
- *Cluster ist von einer Datei belegt und enthält die Adresse des nächsten Clusters, der zu dieser Datei gehört bzw. ist der letzte Cluster der Datei.*

22. Beschreiben Sie die Aufgabe des Journals bei Journaling-Dateisystemen.

Im Journal werden Schreibzugriffe gesammelt, bevor sie durchgeführt werden.

23. Nennen Sie einen Vorteil von Journaling-Dateisystemen gegenüber Dateisystemen ohne Journal.

Nach einem Absturz müssen nur diejenigen Dateien (Cluster) und Metadaten überprüft werden, die im Journal stehen.

24. Nennen Sie die drei Werte, die zum Speichern eines Extents nötig sind.

Start (Clusternummer) des Bereichs (Extents) in der Datei.

Größe des Bereichs in der Datei (in Clustern).

Nummer des ersten Clusters auf dem Speichergerät.

25. Beschreiben Sie den Vorteil des Einsatzes von Extents gegenüber direkter Adressierung der Cluster.

Statt vieler einzelner Clusternummern sind nur die 3 oben genannten Werte nötig. Vorteil: Weniger Verwaltungsaufwand.

26. Beschreiben Sie, was das Defragmentieren macht.

Logisch zusammengehörende Cluster von Dateien werden räumlich beieinander auf dem Speichermedium angeordnet.

27. Beschreiben Sie welche Art der Datenverarbeitung durch Defragmentieren maximal beschleunigt wird.

Eine zusammenhängende Anordnung beschleunigt das fortlaufende Vorwärtslesen der Daten maximal, da (bei Festplatten) keine Warte- und Suchzeiten mehr vorkommen können.

28. Beschreiben Sie in welchem Szenario das Defragmentieren sinnvoll ist.

Wenn die Suchzeiten groß sind.

Aufgabe 2 (Dateisysteme)

Kreuzen Sie bei jeder Aussage zu Dateisystemen an, ob die Aussage wahr oder falsch ist.

Aussage	wahr	falsch
Inodes speichern alle Verwaltungsdaten (Metadaten) der Dateien.		X
Dateisysteme adressieren Cluster und nicht Blöcke des Datenträgers oder Laufwerks.	X	
Je kleiner die Cluster, desto größer ist der Verwaltungsaufwand für große Dateien.	X	
Je größer die Cluster, desto geringer ist der Kapazitätsverlust durch interne Fragmentierung.		X
Unter UNIX haben Dateiendungen schon immer eine große Bedeutung.		X
Moderne Dateisysteme arbeiten so effizient, dass Puffer durch das Betriebssystem nicht mehr üblich sind.		X
Absolute Pfadnamen beschreiben den kompletten Pfad von der Wurzel bis zur Datei.	X	
Das Trennzeichen in Pfadangaben ist bei allen Betriebssystemen gleich.		X
Ein Vorteil der Blockgruppen bei ext2 ist, dass die Inodes physisch nahe bei den Clustern liegen, die sie adressieren.	X	
Eine Dateizuordnungstabelle (FAT) erfasst die belegten und freien Cluster im Dateisystem.	X	
Bei der Master File Table von NTFS ist Fragmentierung unmöglich.		X
Ein Journal im Dateisystem reduziert die Anzahl der Schreibzugriffe.		X
Journaling-Dateisysteme grenzen die bei der Konsistenzprüfung zu überprüfenden Daten ein.	X	
Bei Dateisystemen mit Journal sind Datenverluste garantiert ausgeschlossen.		X
Vollständiges Journaling führt alle Schreiboperation doppelt aus.	X	
Extents verursachen weniger Verwaltungsaufwand als Blockadressierung.	X	

Aufgabe 3 (Mustervergleiche und Datenauswertung)

1. Nennen (oder beschreiben) Sie eine sinnvolle Anwendung für das Kommando `sed`.

Das Kommando ermöglicht Texttransformationen. Man kann mit sed beispielsweise einzelne Zeichen, Zeilen oder Muster Eingabetext verändern oder löschen. Man kann auch Inhalte in einen Eingabetext einfügen.

2. Erzeugen Sie eine Datei `sedtest.txt` mit folgendem Inhalt:

```
Zeile 1
Zeile 2
Zeile 3
Zeile 4
Zeile 5
Zeile 6
```

```
$ echo -e "Zeile 1\nZeile 2\nZeile 3\nZeile 4\nZeile 5\nZeile 6"
> ~/sedtest.txt
```

Fügen Sie mit `sed` 3 Leerzeichen am Anfang jeder Zeile ein (Einrückung).

```
$ sed "s/^/   /" ~/sedtest.txt
```

3. Geben Sie mit `sed` die Zeilen 2 bis 5 der Datei `sedtest.txt` aus.

```
$ sed -n "2,5p" ~/sedtest.txt
```

4. Löschen Sie mit `sed` jede 2. Zeile der Datei `sedtest.txt`.

```
$ sed "n;d;" ~/sedtest.txt
```

5. Erzeugen Sie eine Datei `htmlcode.html` mit folgendem Inhalt:

```
<a href="BTSWS2019/index.html">Betriebssysteme (BTS)</a><p>
<b>Das ist eine <i>HTML-Datei</i></b><br>
<h2>Eine Überschrift<h2>
```

```
$ echo -e "<a href='BTSWS2019/index.html'>Betriebssysteme (BTS)</a><p>" >> ~/htmlcode.html
$ echo -e "<b>Das ist eine <i>HTML-Datei</i></b><br>" >> ~/htmlcode.html
$ echo -e "<h2>Eine Überschrift<h2>" >> ~/htmlcode.html
```

Entfernen Sie mit `sed` alle HTML-Tags aus der Datei `htmlcode.html`.

```
$ sed -e "s/<[^>]*>//g" ~/htmlcode.html
```

6. Erzeugen Sie eine Datei `umlaute.txt` mit folgendem Inhalt:

```
Bäume, Äpfel, Bücher, Übertreibung
Töpfe, Öffentlichkeit, Straße, Spaß
```

```
$ echo "Bäume, Äpfel, Bücher, Übertreibung" >> ~/umlaute.txt
$ echo "Töpfe, Öffentlichkeit, Straße, Spaß" >> ~/umlaute.txt
```

Ändern Sie mit `sed` alle Umlaute aus der Datei `umlaute.txt` in „ae“, „oe“, „ue“, „Ae“, „Oe“, „Ue“ und „ss“.

```
$ sed -e "s/ä/ae/g;s/Ä/Ae/g;s/ö/oe/g;s/Ö/Oe/g;s/ü/ue/g;s/Ü/Ue/g;s/ß/ss/g" ~/umlaute.txt
```

7. Erzeugen Sie eine Datei `bundesliga_08_0405.txt` mit den Ergebnissen des 8. Spieltags der Saison 2004/2005:

```
Schalke      - Bochum      3 : 2 61500 Zuschauer
Bielefeld    - Stuttgart   0 : 2 22700 Zuschauer
Dortmund     - Nürnberg    2 : 2 73500 Zuschauer
Leverkusen   - Hamburg     3 : 0 22500 Zuschauer
Freiburg     - Mainz       1 : 2 24000 Zuschauer
Kaiserslautern - Berlin     0 : 2 30500 Zuschauer
Wolfsburg    - Mönchengladbach 2 : 1 26500 Zuschauer
Rostock      - Hannover    1 : 3 16500 Zuschauer
Bremen       - München     1 : 2 42000 Zuschauer
```

```
$ echo -e "Schalke      - Bochum      3 : 2 61500 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Bielefeld    - Stuttgart   0 : 2 22700 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Dortmund     - Nürnberg    2 : 2 73500 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Leverkusen   - Hamburg     3 : 0 22500 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Freiburg     - Mainz       1 : 2 24000 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Kaiserslautern - Berlin     0 : 2 30500 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Wolfsburg    - Mönchengladbach 2 : 1 26500 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Rostock      - Hannover    1 : 3 16500 Zuschauer" >> ~/bundesliga_08_0405.txt
$ echo -e "Bremen       - München     1 : 2 42000 Zuschauer" >> ~/bundesliga_08_0405.txt
```

8. Nennen (oder beschreiben) Sie eine sinnvolle Anwendung für das Kommando `awk`.

Das Kommando ermöglicht das Suchen und Verarbeiten von Mustern in Eingabetexten. Man kann beispielsweise mit `awk` bestimmte Spalten eines Eingabetextes addieren und das Ergebnis in der gewünschten Form ausgeben.

9. Ermitteln Sie mit `awk` alle Spiele, bei denen mehr als 35000 Zuschauer waren.

```
$ awk '$7 > 35000' ~/bundesliga_08_0405.txt
Schalke      - Bochum      3 : 2 61500 Zuschauer
Dortmund     - Nürnberg    2 : 2 73500 Zuschauer
Bremen       - München     1 : 2 42000 Zuschauer
```

10. Ermitteln Sie mit `awk` alle Spiele, bei denen weniger als 50000 Zuschauer waren und bei denen es einen Sieg der Heimmannschaft gab.

```
$ awk '$4 > $6 && $7 < 50000' ~/bundesliga_08_0405.txt
Leverkusen   - Hamburg     3 : 0 22500 Zuschauer
Wolfsburg    - Mönchengladbach 2 : 1 26500 Zuschauer
```

11. Ermitteln Sie mit `awk` für jedes Spiel die Summe der gefallen Tore.

```
$ awk '{print $4+$6 " " " $1 $2 $3}' ~/bundesliga_08_0405.txt
5 Schalke-Bochum
2 Bielefeld-Stuttgart
4 Dortmund-Nürnberg
```

- 3 Leverkusen-Hamburg
- 3 Freiburg-Mainz
- 2 Kaiserslautern-Berlin
- 3 Wolfsburg-Mönchengladbach
- 4 Rostock-Hannover
- 3 Bremen-München

12. Ermitteln Sie mit `awk` in welcher Stadt die meisten Zuschauer waren und geben das Ergebnis wie folgt aus:

Die meisten Zuschauer waren in STADT (ANZAHL).

```
$ awk -v max=0 'max<$7 {max = $7; stadt=$1} \
END{print"Die meisten Zuschauer waren in",stadt, "("max")"}' \
~/bundesliga_08_0405.txt
Die meisten Zuschauer waren in Dortmund (73500)
```