Inhalt

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 5 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "ucsencs.def" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Kapitel 1
Dateisysteme

In diesem Kapitel lernen Sie die Arbeitsweise der wichtigsten Linux-Dateisysteme sowie den Umgang mit den dazugehörigen Tools kennen.

Was genau macht eigentlich ein Dateisystem, und warum gibt es so viele? Als Linux-Admin haben Sie die Wahl zwischen den Dateisystemen Ext2, Ext3 und Ext4, ReiserFS, XFS, JFS und BtrFS – und das sind nur die gängigsten, universell einsetzbaren Dateisysteme. Dieses Kapitel informiert Sie im Grundlagenteil über die Struktur der Dateisysteme und über die Unterschiede zwischen ihnen. Der Praxisteil erläutert den Umgang mit den dazugehörigen Tools in der täglichen Administrationsarbeit.

1.1 Dateisysteme: von Bäumen, Journalen und einer Kuh

Dateisysteme speichern Daten. Aber sie speichern nicht nur Daten, sondern auch Daten über Daten. Dazu gehören die Größe, der Besitzer oder Informationen darüber, welche Benutzer Zugriffsrechte auf die Daten besitzen. Diese »Daten über Daten« heißen *Metadaten*.

Das Dateisystem speichert die Daten und Metadaten auch nicht nur, es *verwaltet* sie. Essenzielle Verwaltungsaufgaben, die jedes Dateisystem beherrschen muss, sind das

- Finden,
- Einfügen und
- Löschen.

Komplexere Verwaltungsvorgänge wie das *Ändern* setzen sich aus Abfolgen von *Einfügen* und *Löschen* zusammen. Wie schnell, sicher und effizient ein Dateisystem ist, hängt maßgeblich davon ab, wie gut diese grundlegenden Funktionen implementiert sind.

Frühere, heute kaum noch gebräuchliche Dateisysteme verwalteten die Informationen darüber, wo eine bestimmte Datei auf dem Datenträger zu finden war, in einer einfachen Liste. Die Elemente der Liste enthielten Zeiger auf den gesuchten Datenblock. Diese Liste (*File Allocation Table*, FAT) zu verwalten war zwar einfach, aber nicht effizient. Um ein bestimmtes Element zu finden, musste im Extremfall die ganze Liste linear durchsucht werden. Bei einer relativ geringen Anzahl von Dateien sind die Nachteile zu verschmerzen, aber der rasante Anstieg der Festplattenkapazitäten machte bessere Ordnungssysteme erforderlich. Auf sehr

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 7 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

kleinen und entfernbaren Datenträgern wie CDs und einigen USB-Sticks haben die Dateisysteme des alten Typs bis heute überlebt, da ihre geringe Effizienz hier kaum ins Gewicht fällt.

1.1.1 Bäume

Die Organisationsstruktur, mit der moderne Dateisysteme die gespeicherten Daten wiederfinden, ähnelt einem Baum. Der Baum hat eine Wurzel, Gabelungen und Blätter. Die Gabelungen werden *Knoten* genannt.

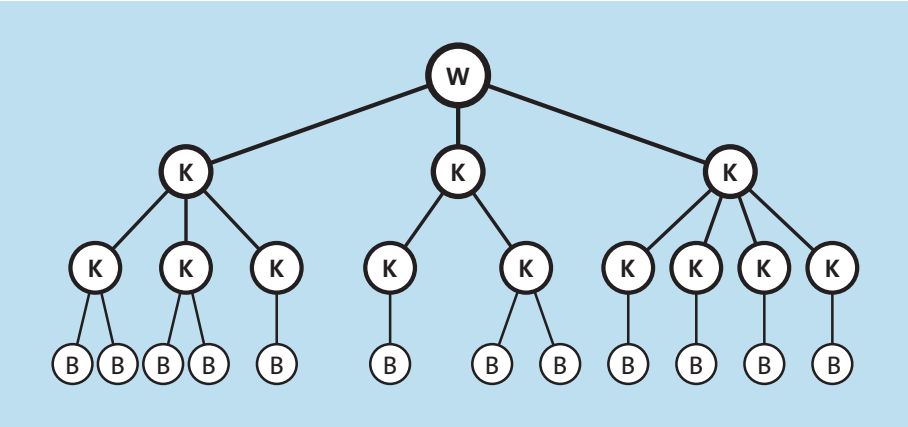

Abbildung 1.1 Schematische Darstellung einer Baumstruktur mit (W)urzel, (K)noten und (B)lättern

Von der Wurzel aus kann man über verhältnismäßig wenige Knoten jedes Blatt erreichen. Die maximale Anzahl der Schritte vom ersten Knoten (die Wurzel wird nicht mitgezählt) bis zu einem Blatt ist die *Höhe* des Baums. Ein Baum mit drei Knotenebenen hat also die Höhe 4 – drei Knotenebenen plus die Blattebene.

Baumarten und Suchstrategien

Um von der Wurzel aus möglichst schnell an jeden beliebigen Punkt des Baums zu gelangen, muss der Baum nicht hoch, sondern möglichst breit sein.[1] Daraus folgt automatisch, dass der Baum möglichst »in der Balance« sein muss, das heißt: Es soll keinen Zweig mit sieben Knoten geben, während ein anderer Zweig nur zwei Knoten hat. Außerdem soll die Höhe des Baums sich in allen Zweigen widerspiegeln. Abbildung 1.2 zeigt als Negativbeispiel einen unbalancierten Baum.

1 In einem Dateisystem kann ein Knoten durchaus 200 Unterknoten haben.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 8 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

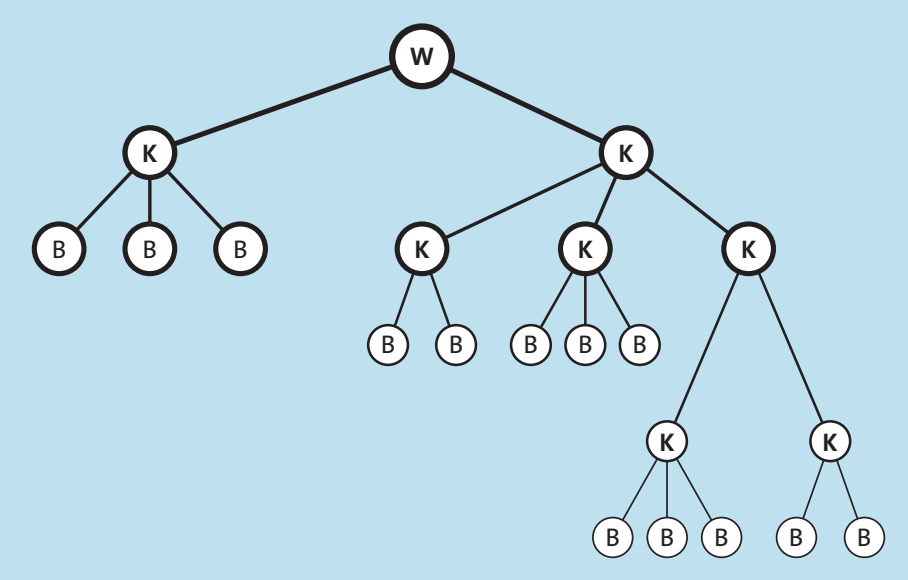

Abbildung 1.2 Eine unbalancierte Baumstruktur

Wurzel, Knoten und Blätter stehen hier natürlich nur metaphorisch für Datenstrukturen. Sie enthalten Schlüssel, anhand derer man sie innerhalb des Baums identifizieren kann. Die Datenstrukturen können neben den Schlüsseln Daten, Metadaten oder beides enthalten. Durch diese Varianz entstehen mehrere Baumarten:

- **B-Bäume**
  B-Bäume sind per Definition balanciert: Alle Blätter liegen auf der gleichen Ebene. Ist die Balance durch Dateioperationen wie Löschen oder Einfügen gefährdet, wird sie durch Neuorganisation von Blättern wiederhergestellt. Dieses Verfahren heißt *Rotation*. Ein weiteres Merkmal von B-Bäumen ist, dass Daten sowohl in den Knoten als auch in den Blättern gespeichert werden. Ext3 und BtrFS benutzen B-Bäume; Abbildung 1.1 ist eine schematische Darstellung eines B-Baums.
- **B+-Bäume**
  In einem B+-Baum werden Daten ausschließlich in den Blättern gespeichert, nicht in den Knoten. Davon profitieren insbesondere Löschoperationen, da Rotationen über Ebenengrenzen hinweg nicht auftreten können. Dateisysteme, die B+-Bäume benutzen, sind beispielsweise ReiserFS 3.x und XFS.
- **H-Bäume**
  Bei einem H-Baum steht die maximale Anzahl der Knoten und Blätter mit dem Anlegen des Dateisystems fest. Das bedeutet einen Verzicht auf Flexibilität, eliminiert aber auch die Notwendigkeit aufwendiger Reorganisationen. Abbildung 1.3 zeigt den schematischen Aufbau eines H-Baums.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 9 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

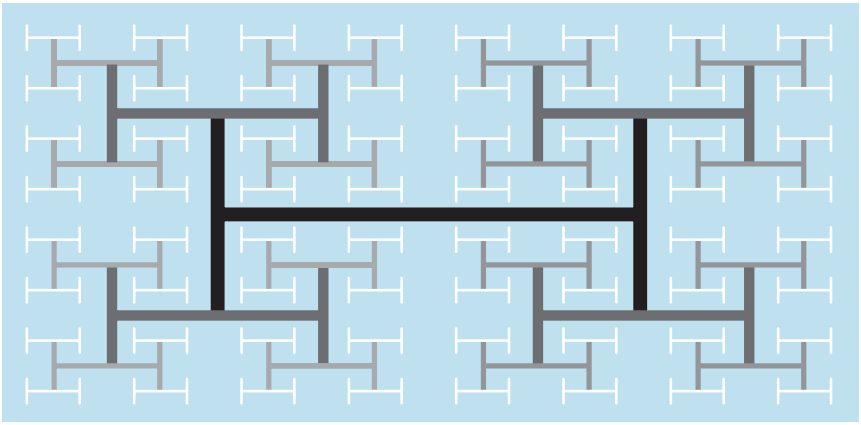

Abbildung 1.3 Schematische Darstellung eines H-Baums

1.1.2 Journale

Wenn Sie eine Datei anlegen, löschen, verschieben oder ändern, sind stets mehrere Datenbereiche auf einer Festplatte involviert: Schlüssel, Metadaten und die Daten selbst, die sich bei großen Dateien auch über mehrere, nicht zusammenhängende Blöcke erstrecken können.

Das Dateisystem hat also ein Problem. Nachdem es die erste Schreiboperation ausgeführt hat, ist der Datenbestand nicht mehr konsistent. Er ist erst dann wieder konsistent, wenn die letzte Schreiboperation erfolgreich abgeschlossen wurde. Fällt innerhalb dieser Zeitspanne der Strom aus, muss das Dateisystem beim Neustart durch eine aufwendige Prüfung und Reparatur wieder einen konsistenten Zustand herstellen. Dieser Vorgang ist auf großen Dateisystemen sehr zeitintensiv. Abhilfe schafft ein *Journaling File System*. Bevor das System die Schreibzugriffe ausführt, schreibt es in das Journal, was es zu tun gedenkt. In bestimmten Zeitabständen wird das Journal abgeschlossen.

Das bedeutet, die »To-do-Liste« im Journal wird mit den tatsächlich durchgeführten Aktionen abgeglichen und aktualisiert. Kommt es zu einem Ausfall, kann zwar immer noch Datenverlust auftreten – nämlich bei den Daten, die zwischen zwei Journal-Abschlusszyklen verändert wurden –, aber Inkonsistenzen werden vermieden, und das System kann ohne langwierige Prüfungen wieder starten.

1.1.3 Und die Kühe? COW-fähige Dateisysteme

Beim Speichern von Daten wird der ursprüngliche Plattenbereich mit den neuen Daten überschrieben. Reicht der Platz nicht mehr aus, werden die zusätzlichen Daten in einen freien Speicherbereich geschrieben. Das dazu notwendige Neupositionieren des Schreib-/Lesekopfs der Platte kostet wertvolle Zeit. Besser ist es, die Daten gleich zusam-

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 10 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

menhängend in einen freien Speicherbereich zu schreiben. So gewinnt man nicht nur Zeit, sondern erhält gratis noch eine Versionsverwaltung, denn die alten Daten sind ja noch auf der Platte. Diese Technik nennt sich »COW« und hat nichts mit wiederkäuenden Paarhufern zu tun, sondern steht für *Copy on Write*. COW-fähige Dateisysteme sind beispielsweise ZFS und BtrFS.

1.2 Praxis

Nach der Lektüre des Grundlagenteils haben Sie sicher schon ein Gespür dafür bekommen, wie die interne Organisationsstruktur eines Dateisystems sein Verhalten in der Praxis beeinflusst. Im Praxisteil lernen Sie den Umgang mit den Verwaltungswerkzeugen der Dateisysteme Ext2/3, ReiserFS und XFS.

1.2.1 Ext2/3-FS aufgebohrt: mke2fs, tune2fs, dumpe2fs, e2label

Die Dateisysteme der Ext-Familie werden von den meisten Linux-Distributionen als Standarddateisystem eingesetzt, weil sie als besonders robust gelten. Bereits beim Anlegen des Ext-Dateisystems können Sie es für seine zukünftigen Aufgaben optimieren.

Ein Ext-Dateisystem anlegen

Wenn Sie ein universell einsetzbares Filesystem benötigen, das gleichermaßen gut mit großen wie kleinen Dateien zurechtkommen soll, legen Sie es einfach mit dem folgenden Kommando an:

```
mke2fs /dev/<Gerätename>
```

Listing 1.1 Ext-Dateisystem ohne weitere Optionen anlegen

`mke2fs` wird abhängig von der Größe der Partition sinnvolle Default-Werte für die Anzahl der Inodes und für die Blockgröße wählen. Es geht dabei davon aus, dass das Dateisystem eine Mischung aus großen, mittleren und kleinen Dateien enthalten wird. Falls Sie allerdings bereits wissen, dass Ihr neues Dateisystem künftig mehrheitlich sehr kleine (Mail-Server) oder sehr große (Videoschnitt, ISO-Images) Dateien aufnehmen können soll, können Sie mit dem Parameter `-T` auf das Layout des Dateisystems Einfluss nehmen (siehe Tabelle 1.1).

Wert	Blockgröße	Inodes pro x Blöcke	Nutzung
small	1 KByte	x = 4	für kleine Partitionen (Default bis 512 MB)
news	4 KByte	x = 1	für viele kleine Dateien

Tabelle 1.1 Optimierung des Dateisystems auf bestimmte Aufgaben

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 11 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Wert	Blockgröße	Inodes pro x Blöcke	Nutzung
largefile	4 KByte	x = 256	für große Dateien, 1 MB pro Inode
largefile4	4 KByte	x = 1.024	für große Dateien, 4 MB pro Inode

Tabelle 1.1 Optimierung des Dateisystems auf bestimmte Aufgaben (Forts.)

Die Routineüberprüfung

Nach einer bestimmten Anzahl von Mountvorgängen oder nach Ablauf einer definierten Zeitspanne – je nachdem, was zuerst eintritt – werden Ext-Dateisysteme einer Routineüberprüfung unterzogen. Auf Systemen, die oft neu gestartet werden, können Sie diese Werte mit dem Kommando `tune2fs` erhöhen. So erfolgt die Prüfung nach 50 Mounts oder 60 Tagen:

```
tune2fs -c 50 -i 60
```

Listing 1.2 Prüfintervall heraufsetzen

Auf- und Abwärtskompatibilität

Das Dateisystem Ext3 (*Third Extended Filesystem*) erweitert seinen Vorgänger Ext2 um die Journaling-Fähigkeit, die Inkonsistenzen zu vermeiden hilft. Beide haben eine hohe Stabilität und Robustheit, die über die Jahre durch schrittweise Verbesserungen und den massenhaften Einsatz auf unzähligen Linux-Systemen erreicht wurde. Beide Systeme sind miteinander kompatibel. Ein Ext3-Dateisystem kann als Ext2 gemountet werden und arbeitet mit Ausnahme des abgeschalteten Journals wie von einem Ext2-System gewohnt. Umgekehrt können Sie ein Ext2-Dateisystem jederzeit mit einem Ritterschlag zum Ext3-System aufwerten. Der folgende Befehl versieht das Ext2-Filesystem auf */dev/sda3* mit einem Journal:

```
tune2fs -j /dev/sda3
```

Listing 1.3 Ext2-System zu Ext3 konvertieren

Prinzipiell funktioniert das sogar, während das Dateisystem gemountet ist – das Journal würde in diesem Fall in einer sichtbaren Datei mit dem Namen *.journal* angelegt. Aber sicherer ist es, die Konvertierung am ausgehängten Dateisystem vorzunehmen.

Journaling

Ext3 kennt zwei Journaling-Methoden, um die Konsistenz der Metadaten zu schützen, und eine dritte, die auch die eigentlichen Nutzdaten mit einbezieht:

- **data=ordered**
  Dies ist die Standardeinstellung, ein Kompromiss zwischen Sicherheit und Geschwindigkeit. Erst nachdem die Nutzdaten einer Schreiboperation auf die Platte geschrieben wurden, werden die Journaleinträge aktualisiert.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 12 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

- **data=writeback**
  Das Dateisystem wartet nicht mit dem Aktualisieren des Journals, bis die Nutzdaten tatsächlich geschrieben wurden. Diese Methode bringt einen Geschwindigkeitsvorteil, kann aber zu Datenverlust führen, wenn zum Zeitpunkt eines Ausfalls das Journal bereits aktualisiert war, aber die Daten noch nicht vollständig geschrieben wurden.
- **data=journal**
  Im Gegensatz zum *Metadata Journaling* der Methoden *ordered* und *writeback* etabliert diese Methode das *Full Journaling*. Daten und Metadaten werden zunächst in das Journal geschrieben. Erst nachdem die Daten von dort aus an ihre endgültige Position im Dateisystem kopiert wurden, wird das Journal aktualisiert. Weil auf diese Weise alle Schreibvorgänge doppelt ausgeführt werden, ist das *Full Journaling* bei Schreiboperationen recht langsam, vermeidet aber wirkungsvoll Dateninkonsistenzen.

Sie können insbesondere die Schreibperformance verbessern, indem Sie das Journal auf einen separaten Datenträger schreiben lassen, etwa einen kleinen RAID-Verbund. Dazu gehen Sie in zwei Schritten vor: Zuerst bereiten Sie das Block-Device vor, auf dem das Journal liegen soll, und im zweiten Schritt formatieren Sie die eigentliche Datenplatte und übergeben als Parameter das Journalgerät. Wenn Ihre Datenplatte */dev/sda1* und Ihre Journalplatte */dev/sdb1* heißt, lauten die beiden Kommandos wie folgt:

```
# Journaldatenträger vorbereiten:
mke2fs -O journal_dev /dev/sdb1

# Datenplatte formatieren, auf Journal-Device verweisen:
mke2fs /dev/sda1 -J device=/dev/sdb1
```

Listing 1.4 Dateisystem mit separatem Journal-Device

Zusätzlich können Sie mit dem Parameter `-L <Bezeichnung>` jedem Dateisystem einen Bezeichner (*Label*) zuweisen. Das ist auch nachträglich mit dem Befehl `e2label <Gerät> <Bezeichner>` möglich. Das *Label* darf nicht mehr als 16 Zeichen lang sein.

dumpe2fs: Informationen über das Filesystem

Mit `dumpe2fs /dev/<Gerätename>` lassen Sie sich Informationen über das Ext-Dateisystem ausgeben. Die Ausgabe ist recht lang, da auch alle Informationen über Superblocks und Blockgruppen ausgegeben werden. In den meisten Fällen reicht es aus, wenn Sie sich die verkürzte Ausgabe mit dem Kommando `dumpe2fs -h /dev/<Gerätename>` ausgeben lassen.

Hier sehen Sie unter anderem den aktuellen Status des Dateisystems und Informationen über den zyklischen *File System Check*:

```
[…]
Filesystem state:         clean
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 13 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

```
[…]
Filesystem created:       Wed Nov 11 10:25:43 2009
Last mount time:          Sun Sep 26 12:51:25 2010
Last write time:          Tue May 25 14:24:51 2010
Mount count:              14
Maximum mount count:      31
Last checked:             Tue May 25 14:24:51 2010
Check interval:           15552000 (6 months)
Next check after:         Sun Nov 21 13:24:51 2010
Lifetime writes:          3086 MB
[…]
```

Listing 1.5 Ausgabe von »dumpe2fs« (Auszug)

Sollten auf Ihrem Dateisystem einige Blöcke als *bad* (schlecht, beschädigt) markiert sein, können Sie sich mit dem Befehl `dumpe2fs -b` ausgeben lassen, welche Blöcke betroffen sind.

Dateisystemprüfung mit »e2fsck«

Wenn Sie den Verdacht haben, dass Ihr Dateisystem über noch nicht lokalisierte *bad blocks* stolpert, rufen Sie einmal `e2fsck -c /dev/<Gerätename>` auf. Das Dateisystem darf dabei nicht gemountet sein. `e2fsck` durchsucht das Dateisystem mithilfe des externen Programms `badblocks` nach Schadstellen. Die IDs der kaputten Blöcke werden einem speziellen Inode hinzugefügt und vom Dateisystem künftig gemieden.

Fehler, die nicht auf schlechten Blöcken beruhen, können Sie ebenfalls bei ausgehängtem Dateisystem mit dem Befehl `e2fsck -p /dev/<Gerätename>` orten. Das Programm wird versuchen, alle gefundenen Fehler ohne weitere Interaktion Ihrerseits zu beheben. Ist das nicht möglich, wird eine Beschreibung des Fehlers ausgegeben, und die Verarbeitung stoppt.

1.2.2 ReiserFS und seine Tools

ReiserFS war 2001 das erste Journaling-Dateisystem, das in den offiziellen Linux-Kernel einzog, damals allerdings noch in experimentellem Zustand. »Produktionsreif« wurde ReiserFS erst später. ReiserFS kann kleine Dateien etwas effizienter speichern als andere Dateisysteme und bietet dadurch auf einem ansonsten gleichen *Volume* etwa 5 % mehr Speicherkapazität. Die höhere Packungsdichte bedingt aber eine etwas geringere Schreibgeschwindigkeit und lässt sich deshalb abschalten, indem in der */etc/fstab* der Parameter `notail` gesetzt wird.

ReiserFS skaliert auf Systemen mit Mehrkernprozessoren nicht optimal, weil nicht alle Teile des Codes auf mehreren Kernen parallel ausgeführt werden können. Entwicklungen, die dieses Problem beheben, sind zwar in Arbeit, aber noch nicht in den offiziellen Linux-Kernel eingeflossen.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 14 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Ein ReiserFS-Dateisystem anlegen

Der folgende Befehl formatiert ein Block-Device mit dem Reiser-Dateisystem:

```
mkfs.reiserfs /dev/<Gerätename>
```

Listing 1.6 Ein ReiserFS-Dateisystem ohne weitere Optionen anlegen

Sie können den Befehl um `-l <Bezeichner>` ergänzen, um dem Dateisystem ein *Label* anzuheften. Wie bei den Ext-Dateisystemen darf das Label nicht mehr als 16 Zeichen umfassen. Auch ReiserFS erlaubt es, das Journal auf einen separaten Datenträger zu legen. Wenn Ihre Daten auf dem Gerät */dev/sda1* und das Journal auf */dev/sdb1* liegen sollen, lautet das Kommando:

```
mkfs.reiserfs /dev/sda1 -j /dev/sdb1
```

Listing 1.7 ReiserFS-Dateisystem mit ausgelagertem Journal anlegen

Sie können das Journal auch nachträglich auf einen anderen Datenträger auslagern. Dazu benötigen Sie den Befehl `reiserfstune`:

```
reiserfstune /dev/sda1 --journal-new-device /dev/sdb1
```

Listing 1.8 Journal nachträglich auf ein anderes Device legen

1.2.3 XFS

XFS wurde in den 90er-Jahren von *Silicon Graphics* entwickelt und steht seit 2001 für Linux zur Verfügung. Schreiboperationen auf das Journal laufen unabhängig von den eigentlichen Datentransaktionen ab, die XFS möglichst lange im RAM puffert (*Delayed Allocation)*. Auf diese Weise wird der Durchsatz erhöht und Fragmentierung beim Schreiben vermieden. Dadurch wird zwar auch die Gefahr eines Datenverlusts bei einem Totalausfall des Systems erhöht, die Konsistenz des Dateisystems ist jedoch sichergestellt.

Ein XFS-Dateisystem anlegen

Wie Ext und ReiserFS kann auch XFS sein Journal auf das gleiche Blockgerät schreiben wie die Nutzdaten oder ein »externes« Journal führen. So legen Sie das Dateisystem an:

```
# XFS mit internem Journal (Journal und Daten auf gleicher Partition)
mkfs.xfs /dev/sda1

# XFS mit externem Journal (Journal auf separater Partition)
mkfs.xfs -l logdev=/dev/sdb1 /dev/sda1
```

Listing 1.9 XFS-Filesystem mit internem und externem Journal

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 15 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Das Dateisystem reorganisieren

Sollte das Dateisystem einmal merklich langsamer werden, können Sie es mit dem Befehl `xfs_fsr` reorganisieren. Geben Sie den Befehl einfach als `root` auf einer Kommandozeile ein – er funktioniert auch online, also während das Dateisystem gemountet ist. `xfs_fsr` ermittelt, welche Dateien besonders stark fragmentiert sind, und beginnt dort mit der Neuorganisation. Per Default bricht das Programm nach zwei Stunden automatisch ab, kann aber manuell neu gestartet werden und fährt an der Stelle fort, an der es aufgehört hat. Sie können auch mit dem Parameter `-t <Laufzeit_in_Sekunden>` eine andere Arbeitsdauer festlegen. Es empfiehlt sich, die Neuorganisation *cron*-gesteuert zu einer Zeit vornehmen zu lassen, in der das Dateisystem wenig belastet ist.

1.2.4 Das Dateisystem vergrößern oder verkleinern

Sie können das Dateisystem bei Bedarf vergrößern oder – im Fall von Ext2/3 und ReiserFS – auch verkleinern. Ein XFS-Dateisystem lässt sich hingegen nicht verkleinern. Beim Vergrößern eines Dateisystems muss die Partition natürlich eine gewisse Reserve aufweisen, die das Dateisystem nutzen kann. Liegt das Dateisystem auf einer logischen Partition, erweitern Sie diese zunächst mit dem `lvextend`-Kommando (siehe Abschnitt **??**, »Rein logisch: Logical Volume Manager ›LVM‹«).

Eine Änderung der Größe ist immer ein recht tiefer Eingriff ins Dateisystem. Obwohl die Tools einen hohen Reifegrad haben, sollten Sie über ein aktuelles Backup verfügen. Vorher noch einen *File System Check* laufen zu lassen, ist ebenfalls eine gute Idee.

Die Größe eines Ext-Dateisystems ändern

Ext-Dateisysteme können Sie online und offline, also im eingehängten oder im nicht eingehängten Zustand, vergrößern, aber nur offline verkleinern. Wenn Sie Ihr Dateisystem auf die maximale Größe der (logischen) Partition ausdehnen möchten, geben Sie als `root` diesen Befehl ein:

```
resize2fs -p /dev/<Gerätename>
```

Listing 1.10 Ext-Dateisystem vergrößern

Der Parameter `-p` erzeugt einen Fortschrittsbalken, während die Vergrößerung läuft. Eine Verkleinerung nehmen Sie vor, indem Sie die Zielgröße des Dateisystems angeben. Denken Sie daran, dass Sie Ext-Dateisysteme nur im ausgehängten Zustand verkleinern können. Falls Ihr System diese Partition zum Starten benötigt, verwenden Sie eine Live-CD wie *Knoppix*.

Wenn Ihr Dateisystem 80 GB groß ist und auf 60 GB schrumpfen soll, lautet das Kommando:

```
resize2fs -p /dev/<Gerätename> 60G
```

Listing 1.11 Ext-Dateisystem verkleinern

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 16 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Die Größe eines Reiser-Dateisystems ändern

Ein Reiser-Dateisystem lässt sich, sofern das Dateisystem nicht gemountet ist, vergrößern und verkleinern. Wie bei den Ext-Dateisystemen können Sie auch eine bestimmte Zielgröße angeben, auf die das Dateisystem vergrößert oder verkleinert werden soll:

```
resize_reiserfs /dev/<Gerätename> 60G
```

Listing 1.12 ReiserFS auf eine bestimmte Zielgröße bringen

Ebenso können Sie das vorhandene Dateisystem auf die maximale Größe aufblähen, die die (logische) Partition hergibt. Dazu lassen Sie einfach die Angabe der Zielgröße weg.

```
resize_reiserfs /dev/<Gerätename>
```

Listing 1.13 ReiserFS maximal vergrößern

Alternativ haben Sie die Möglichkeit, die Differenz zur aktuellen Größe des Filesystems anzugeben:

```
#Dateisystem um 20 GB verkleinern:
resize_reiserfs /dev/<Gerätename> -20G

#Dateisystem um 20 GB vergrößern:
resize_reiserfs /dev/<Gerätename> +20G
```

Listing 1.14 ReiserFS relativ verändern

Die Größe eines XFS-Systems ändern

Ein XFS-System können Sie lediglich vergrößern, nicht verkleinern. Die Änderung der Größe funktioniert nur, während das Dateisystem gemountet ist.

Eine Vergrößerung auf eine von Ihnen definierte Größe ist nicht möglich, XFS vergrößert sich immer auf die gesamte zur Verfügung stehende Partitionsgröße.

Im Unterschied zu den anderen Dateisystemen geben Sie bei XFS nicht das (logische) Gerät, sondern den Mountpunkt an:

```
xfs_growfs /<Mountpunkt> -o remount,resize
```

Listing 1.15 XFS vergrößern

1.2.5 BtrFS

Der Name *BtrFS* deutet auf die verwendeten B-Bäume hin (*B-Tree-FS*), wird aber meist wie »Butter-FS« oder »Better-FS« ausgesprochen. Obwohl die Entwicklung noch nicht vollständig abgeschlossen ist, insbesondere was die Filesystem-Tools angeht, erhält BtrFS bereits viel

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 17 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Lob. Die Wahrscheinlichkeit ist hoch, dass BtrFS in nicht ferner Zukunft die Nachfolge der Ext-Dateisysteme als Quasi-Standard antreten wird.

[!] Zum dem Zeitpunkt, als diese Auflage geschrieben wurde, fand sich im offiziellen Getting-Started-Guide des BtrFS-Wiki[2] immer noch der Warnhinweis, dass man den aktuellsten Kernel verwenden sollte, wenn man BtrFS einsetzt. Weiterhin ist dort der Hinweis zu finden, dass man ein Backup besitzen und darauf vorbereitet sein sollte, es einzusetzen. Seit 2014 gilt das Dateisystem als stabil.

In der Tat ist die Liste der unterstützten Funktionen beeindruckend. Als COW-System unterstützt BtrFS Snapshots. Der Zustand des Dateisystems wird dabei in einem Subvolume eingefroren und kann dort anderweitig genutzt werden, etwa für ein Backup. Solid-State-Speicher werden gesondert unterstützt, außerdem ist die Unterstützung der RAID-Level 0, 1 und 10 implementiert, die Level 5 und 6 gelten immer noch als experimentell, und das schon seit Jahren. Wie XFS kann BtrFS im gemounteten Zustand defragmentiert werden. Außerdem erkennt BtrFS, wenn es auf einer SSD eingesetzt wird, und aktiviert dafür selbstständig sinnvolle Einstellungen.

Die große Stärke von BtrFS ist, dass es komplette Devices verwalten und durch Subvolumes gekapselte Mountpunkte exportieren kann. Im kommerziellen Bereich bietet das sehr ausgereifte, ZFS ähnliche Möglichkeiten. Da die Lizenz von ZFS (Common Development and Distribution License, CDDL) nicht kompatibel mit der Lizenz des Linux-Kernels (GNU Public License, GPL) ist, wird sich ZFS nie im Linux-Kernel befinden. Aus diesem Grund versuchen namhafte Firmen die Entwicklung von BtrFS nach vorn zu treiben, um die Features eines modernen Dateisystems mit integriertem Volumemanager auch unter Linux direkt nutzen zu können.

[+] Beachten Sie bitte, dass unter Debian und Ubuntu das Paket *btrfs-progs* und unter openSUSE das Paket *btrfsprogs* installiert sein sollte, wenn Sie mit BtrFS arbeiten wollen. In CentOS Stream gibt es derzeit keinen BtrFS-Support, was umso erstaunlicher ist, als dass Fedora mit Version 33 dieses Dateisystem als Standard für Desktops einsetzt.

Ein BtrFS-Dateisystem anlegen und die Komprimierung aktivieren

Da die Entwicklung nach wie vor andauert, sollten Sie BtrFS derzeit noch mit Vorsicht genießen. BtrFS zeigt noch Instabilitäten, wenn das Dateisystem sehr voll ist, und die Funktionen, die das System nach einem Crash wieder reparieren, sind noch nicht vollständig implementiert. Das Kommando `btrfsfsck` existiert zwar, aber dahinter verbirgt sich derzeit nur ein Integritäts-Check, echte Reparaturen am Dateisystem können Sie damit noch nicht durchführen. Wenn Sie BtrFS einmal ausprobieren möchten, können Sie es wie jedes andere Dateisystem auf einer freien Partition, im folgenden Beispiel */dev/sdb1*, anlegen.

2 *https://btrfs.wiki.kernel.org/index.php/Getting_started*

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 18 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Dieses Beispiel dient Demonstrationszwecken. Auch wenn es möglich ist, direkt auf das Filesystem innerhalb der Partition zuzugreifen, geht die Konzeption von BtrFS in die Richtung, dass Sie das nicht müssen, sondern stattdessen mit Subvolumes arbeiten, die später in diesem Abschnitt beschrieben werden.

```
mkfs.btrfs /dev/sdb1
```

Listing 1.16 Ein BtrFS-Dateisystem anlegen bzw. eine Partition BtrFS zur Verfügung stellen

Auch das Einhängen des neuen Dateisystems in den Verzeichnisbaum funktioniert wie bei anderen Dateisystemen. Wenn Sie möchten, können Sie BtrFS beim Mountbefehl zusätzlich anweisen, die Datenkomprimierung einzuschalten:

```
mount -o compress /dev/sdb1 /mnt/meinbtrfs
```

Listing 1.17 Das neue Dateisystem mounten und dabei die optionale Komprimierung einschalten

Die Komprimierung verschlingt natürlich einige CPU-Zyklen, aber in Zeiten schneller Multicore-CPUs ist es unwahrscheinlich, dass Sie diese Verzögerung wahrnehmen. Im Gegenteil: Die Komprimierung wird sich eher positiv auf die Verarbeitungsgeschwindigkeit auswirken, weil Schreibzugriffe auf die Platten wesentlich flotter ablaufen. Das gilt natürlich nicht für Daten, die bereits komprimiert sind, wie JPEG-Bilder, die meisten Videos und Musik im MP3-Format. BtrFS erkennt solche bereits komprimierten Dateien und macht gar keinen Versuch, sie noch weiter zu verdichten. Bei gut komprimierbaren Daten wie Text oder Quellcode ist der Gewinn an Geschwindigkeit und Plattenplatz allerdings beträchtlich.

Ein Ext2/3/4-Dateisystem in BtrFS konvertieren

Sie können ein mit Ext2, Ext3 oder Ext4 formatiertes Dateisystem nachträglich in BtrFS konvertieren. Dazu müssen Sie das Dateisystem, im folgenden Beispiel befindet es sich wieder auf der Partition */dev/sdb1*, zunächst aushängen:

```
# zuerst wird das Dateisystem ausgehängt
umount /dev/sdb1

# jetzt erfolgt die Konvertierung
btrfs-convert /dev/sdb1

# danach können Sie das Dateisystem wieder mounten (mit Komprimierung)
mount -o compress /dev/sdb1 /mnt/meinbtrfs
```

Listing 1.18 Eine Ext2/3/4-Partition in BtrFS konvertieren

Sie werden feststellen, dass die Konvertierung recht flott vonstattengeht, selbst auf großen Dateisystemen. Das liegt daran, dass der Konverter die Nutzdaten überhaupt nicht touchiert, sondern lediglich die BtrFS-Metadaten schreibt und einen Snapshot des ursprüng-

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 19 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

lichen Dateisystems erstellt. Wenn Sie entscheiden, dass die Konvertierung in BtrFS keine gute Idee war, können Sie den ursprünglichen Zustand jederzeit mithilfe des Snapshots wiederherstellen. Dabei verlieren Sie allerdings alle Änderungen, die Sie seit der Konvertierung vorgenommen haben.

```
# wieder wird zuerst das Dateisystem ausgehängt
umount /dev/sdb1

# die Konvertierung rückgängig machen
btrfs-convert -r /dev/sdb1

# danach können Sie das Dateisystem wieder mounten
mount /dev/sdb1 /mnt/meinext3fs

# Kontrolle
df -h
```

Listing 1.19 Die Konvertierung rückgängig machen

Wenn Sie sich entschließen, BtrFS zu behalten, und auf die Möglichkeit der Rückkonvertierung keinen Wert legen, dann können Sie den Snapshot – er heißt `ext2_saved` – löschen:

```
btrfs subvolume delete /mnt/meinbtrfs/ext2_saved
```

Listing 1.20 Den Snapshot des alten Dateisystems entfernen

Subvolumes

Subvolumes werden oft als *Namespaces* innerhalb eines BtrFS-Dateisystems beschrieben. Es ist aber einfacher, sich Subvolumes als *virtuelle Dateisysteme* vorzustellen. Bei der Erstellung eines BtrFS-Dateisystems wird bereits ein *root-Subvolume* angelegt. Subvolumes können wieder weitere Subvolumes enthalten und so ineinander verschachtelt werden.

Wir haben bereits in der Partition *sdb1* ein Dateisystem erstellt, das wir nun nach */mnt/btrfs* mounten. Dort erstellen wir ein Verzeichnis *home* und darin ein Subvolume pro User; *opt* soll ein weiteres Subvolume werden. Diese Subvolumes lassen sich selbstverständlich anzeigen, wie Sie in Listing 1.21 sehen. Leider wird das standardmäßig erstellte Subvolume für das Filesystem nicht angezeigt, da es aber immer die ID 5 hat, ist es leicht zu identifizieren.

```
# btrfs subvolume create /mnt/btrfs/home/user1
Create subvolume '/mnt/btrfs/home/user1'
# btrfs subvolume create /mnt/btrfs/home/user2
Create subvolume '/mnt/btrfs/home/user2'
# btrfs subvolume create /mnt/btrfs/opt
Create subvolume '/mnt/btrfs/opt'

# btrfs subvolume list -apt /mnt/btrfs
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 20 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

```
ID      gen     parent  top level       path
--      ---     ------  ---------       ----
257     6       5       5               home/user1
258     7       5       5               home/user2
259     8       5       5               opt
```

Listing 1.21 Erstellung von Subvolumes

Subvolumes verhalten sich innerhalb des Parent-Volumes wie Verzeichnisse, können allerdings nicht mit `rm` gelöscht werden. Dafür muss `btrfs subvolume delete` verwendet werden. Der Parameter ist der gleiche wie beim Erstellen der Subvolumes.

Eine Besonderheit von Subvolumes ist, dass sie auch eigenständig mittels `mount` eingebunden werden können. Dazu kann entweder der Name oder die ID des Subvolumes verwendet werden:

```
# mount -o subvolume=home/user1 /dev/sdb1 /tmp/1
# mount -o subvolid=257 /dev/sdb1 /tmp/1
```

Listing 1.22 Mounten von Subvolumes

Snapshots

Sie können jederzeit von beliebigen Subvolumes Snapshots erstellen. Diese Abbilder enthalten allerdings nur Daten des Subvolumes, für das sie erstellt wurden, nicht für eventuell darunter liegende Subvolumes. Anders als bei LVM-Snapshots, siehe Abschnitt **??** »Backups mit Snapshots«, müssen Sie bei BtrFS keine besonderen Vorbereitungen treffen. Es hat sich als hilfreich erwiesen, den Snapshots die Endung »-snap« zu geben, um sie leichter identifizieren zu können. Snapshots sind beschreibbar!

BtrFS-intern werden Snapshots wie Subvolumes behandelt. Der einzige Unterschied ist, dass sie zu Beginn keine Daten enthalten. Erst wenn sich das Ursprungs-Subvolume oder der Snapshot verändert, wird – gemäß dem Copy-on-Write-Prinzip – Speicherplatz verwendet. Der Parameter `-r` erzeugt beim Erstellen einen nur lesbaren Snapshot:

```
# btrfs subvolume snapshot /mnt/btrfs/opt /mnt/btrfs/opt-snap
# btrfs subvolume snapshot -r /mnt/btrfs/opt /mnt/btrfs/opt-snap-ro
# btrfs mount -o subvolume=opt-snap /dev/sdb1 /tmp/snap
# btrfs mount -o subvolume=opt-snap-ro /dev/sdb1 /tmp/snap-ro
```

Listing 1.23 Erstellung von Snapshots

Defragmentierung

Wie bei allen COW-Dateisystemen ist Fragmentierung bei BtrFS ein Problem. Geänderte Daten dürfen nicht einfach überschrieben werden, sondern wandern in freie Speicherbereiche. Ist der Speicherbereich kleiner als die zu schreibenden Daten, werden diese zerstückelt,

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 21 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

und es kommt zu Fragmentierung. Deshalb müssen BtrFS-Dateisysteme regelmäßig defragmentiert werden. Die Defragmentierung können Sie manuell vornehmen, indem Sie ein Verzeichnis angeben (tatsächlich ein Verzeichnis, nicht ein Subvolume), dessen Daten neu geordnet werden sollen:

```
sudo btrfs filesystem defragment /opt
```

Listing 1.24 Defragmentierung der Daten eines Verzeichnisses

Sie können während der Defragmentierung weiterarbeiten, aber das System wird sich etwas zäher anfühlen. Verzeichnisse auf SSDs müssen nicht defragmentiert werden – die Fragmentierung findet dort zwar auch statt, ist aber unproblematisch, da sie im Gegensatz zu Festplatten nicht zu einer Verringerung des Plattendurchsatzes führt –, eine SSD muss ja keine Schreib-/Leseköpfe neu positionieren. Sie können auch eine Hintergrund-Defragmentierung aktivieren. Dazu fügen Sie einfach in der */etc/fstab* die Mountoption `autodefragment` hinzu.

Mehrere Devices in BtrFS

Wie eingangs geschrieben, sollte man beim Erstellen des BtrFS-Dateisystems davon sprechen, dass Devices BtrFS zur Verfügung gestellt werden. BtrFS bietet Unterstützung für verschiedene RAID-Level, die sich getrennt nach Nutzdaten und Metadaten, die zur Verwaltung von BtrFS verwendet werden, einstellen lassen. Die beiden gebräuchlichsten Fälle sind *Striping* (RAID-0) und *Mirroring* (RAID-1). Wenn man nur Devices ohne Angabe eines RAID-Levels zur Verfügung stellt, sorgt BtrFS für ein Striping der Nutzdaten und ein Mirroring der Metadaten. Um ein Filesystem mit RAID-Level zu mounten, müssen Sie nur ein Device des Verbunds angeben:

```
# mkfs.btrfs -f -d raid1 -m raid1 /dev/sdb /dev/sdc
# mount -o compress /dev/sdb /mnt/btrfs
```

Listing 1.25 Ein BtrFS mit RAID-1 anlegen

Bitte unterschätzen Sie das Volumen der Metadaten nicht! BtrFS kann einige seiner Vorteile nur durch die intensive Nutzung von Metadaten zur Verfügung stellen. Selbst wenn Ihnen das `df`-Kommando freien Speicher anzeigt, heißt das nicht, dass dieser auch wirklich verfügbar ist, da der Speicher von Nutz- und Metadaten gleichermaßen genutzt wird. Mögliche Deduplizierung und Kompression sorgen ebenfalls für ungenaue Anzeigen durch das Betriebssystem. Als Beispiel kopieren wir einmal den Inhalt des */usr*-Verzeichnisses (1,1 GB) in das gerade angelegte Filesystem:

```
# du -hs /usr
1.1G    /usr

# cp -r /usr /mnt/btrfs
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 22 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

```
# df -h /mnt/btrfs
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        2.0G  552M  1.3G  30% /mnt/btrfs
```

Listing 1.26 Speicherauslastung von BtrFS

Um Verwirrung zu vermeiden, stellt BtrFS das `filesystem`-Subkommando zur Verfügung. Mit `show` können Sie sich den Aufbau anzeigen lassen:

```
# btrfs filesystem show /mnt/btrfs
Label: none  uuid: 71c1d2cc-479a-4090-ae7f-482d9ac21c29
        Total devices 2 FS bytes used 535.89MiB
        devid    1 size 2.00GiB used 853.50MiB path /dev/sdb
        devid    2 size 2.00GiB used 833.50MiB path /dev/sdc
```

Listing 1.27 Den Aufbau des Filesystems anzeigen lassen

Das Diskfree-Kommando in Listing 1.26 zeigt Ihnen 552 MB verbrauchten Speicherplatz. Das Kommando `btrfs filesystem show` gibt hingegen aus, dass 853 MiB belegt sind.

Wie der Speicher verwendet wird, zeigt Ihnen `btrfs filesystem df`. Wie Sie sehen können, werden 204 MiB für Metadaten verwendet:

```
# btrfs filesystem df /mnt/btrfs | column -ts,
Data            RAID1: total=620.75MiB   used=500.80MiB
Data            single: total=8.00MiB    used=0.00B
System          RAID1: total=8.00MiB     used=16.00KiB
System          single: total=4.00MiB    used=0.00B
Metadata        RAID1: total=204.75MiB   used=35.08MiB
Metadata        single: total=8.00MiB    used=0.00B
GlobalReserve   single: total=16.00MiB   used=0.00B
```

Listing 1.28 »df«-Subkommando von »btrfs filesystem«

Eine genauere und auch längere Ausgabe bekommen Sie mit `btrfs filesystem usage`:

```
# btrfs filesystem usage /mnt/btrfs
Overall:
    Device size:                   4.00GiB
    Device allocated:              1.65GiB
    Device unallocated:            2.35GiB
    Device missing:                  0.00B
    Used:                          1.05GiB
    Free (estimated):              1.30GiB      (min: 1.30GiB)
    Data ratio:                       1.99
    Metadata ratio:                   1.96
    Global reserve:               16.00MiB      (used: 0.00B)
```


Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 23 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme/dateisysteme" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

```
Data,single: Size:8.00MiB, Used:0.00B
   /dev/sdb        8.00MiB

Data,RAID1: Size:620.75MiB, Used:500.80MiB
   /dev/sdb      620.75MiB
   /dev/sdc      620.75MiB

Metadata,single: Size:8.00MiB, Used:0.00B
   /dev/sdb        8.00MiB

Metadata,RAID1: Size:204.75MiB, Used:35.08MiB
   /dev/sdb      204.75MiB
   /dev/sdc      204.75MiB

System,single: Size:4.00MiB, Used:0.00B
   /dev/sdb        4.00MiB

System,RAID1: Size:8.00MiB, Used:16.00KiB
   /dev/sdb        8.00MiB
   /dev/sdc        8.00MiB

Unallocated:
   /dev/sdb        1.17GiB
   /dev/sdc        1.19GiB
```

Listing 1.29 »usage«-Subkommando von »btrfs filesystem«

1.3 Fazit

Sie haben nun eine Reihe von Dateisystemen kennengelernt und erfahren, dass jedes seine eigenen Stärken und Schwächen hat. Das versetzt Sie in die Lage, für jede Aufgabe das passende Dateisystem zu wählen. Obwohl die Standarddateisysteme wie Ext4 und XFS prinzipiell für alle Einsatzzwecke geeignet sind, können Sie durch den Einsatz eines spezielleren Dateisystems nicht selten ein Plus an Geschwindigkeit oder Sicherheit herausholen.

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 24 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme.ind" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x

Index

Deimeke/Heinlein/Kania/Miesen/van Soest – Das Linux-Administrationshandbuch – Sechste Auflage – Frühjahr 2021 Rheinwerk Verlag, ISBN: 978-3-8362-8088-3
Layout: gp.cls, Version 3.6.008 (9th May, 2021), (C) Dr. Daniel Lauer, Bonn Mit TEX bearbeitet am 24. September 2021, 10:01 Uhr Umlaut-Check: ÄÖÜäöüß. Ligaturen: fl fi
S. 25 Einstellungen: ohne Marginalien – ohne Seitenrand – mit Taben – mit Kapitel-Intro – extra breit – Zoom auf 100.0%
Aktuelle Quelldatei: "dateisysteme.ind" , Aktueller Job: dateisysteme, Aktuelles Encoding: utf8x