Dateisystem-Werkzeuge benutzen

# Handwerk mit Format

Formatierprogramme erledigen längst mehr als nur das Installieren von Dateisystemen. Wie gut sortierte Werkzeugkästen liefern sie für jede Aufgabe das richtige Helferlein. Wer die Performance steigern will, kitzelt über Mount-Optionen das letzte Stückchen Leistung aus Ext 3, JFS, ReiserFS oder XFS heraus. Jörg Reitter

**Langwierig** sind Tests mit Dateisystemen nicht mehr. JFS, ReiserFS und XFS kommen mit großen Partitionen gut zurecht. Nicht länger als ein Wimpernschlag dauert es, wenn die Mkfs-Programme von JFS und XFS eine 1 TByte große Partition formatieren, bei ReiserFS eine halbe Minute. Zeit lässt sich nur Ext 3, das mit rund einer Stunde vergleichsweise lange braucht. Kein Wunder – schließlich handelt es sich im Kern immer noch um Ext 2.

Neben den Programmen zum Formatieren gibt es noch eine Fülle weiterer Utilities für jedes Dateisystem. Manche sollen dabei helfen, die Performance zu steigern. Andere kommen nur zum Einsatz, falls ein Fehler auftritt. Dieser Beitrag stellt die wichtigsten Kommandozeilenprogramme vor. Die Kästen listen die Mount-Optionen auf. Diese kommen entweder auf der Kommandozeile per »mount«-Befehl zum Einsatz oder der Benutzer trägt sie dauerhaft in die Datei »/etc/fstab« ein.

## Ext 2/3: Reich an Optionen

Es ist eine gute Idee, ein Ext 2 mit einem Journal auszustatten und somit auf das Ext-3-Dateisystem zu aktualisieren. Das Upgrade verursacht keine Probleme; zumindest berichten Benutzer nichts davon in den einschlägigen Foren. Vor der Ausnahme von der Regel schützt eine aktuelle Datensicherung.

Um Ext 3 zu nutzen, muss die Ext-3-Unterstützung im Kernel aktiv sein. Wer die Root-Partition mit Ext 3 ausstattet, muss zudem den Ext-3-Treiber einkompilieren. Falls Linux mit einer Initial-RAM-Disk (Initrd) bootet, reicht es, den Treiber als Modul zu übersetzen. Erfüllt das System alle Voraussetzungen, kann das Upgrade starten. Zuerst aktualisiert Root das Ext-2-System mit dem Programm Tune2fs, das im Paket E2fsprogs [1] enthalten ist:

```
tune2fs -j /dev/hda2
```

Die Option »-j« fertigt lediglich das Journal an und legt die Datei ».journal« im Wurzelverzeichnis der Partition ab. Um das Update abzuschließen, ändert der Benutzer in »/etc/fstab« den Dateisystemtyp von »ext2« nach »ext3«. Ausnahmsweise muss der Rechner anschließend neu starten. Falls das Dateisystem wieder als Ext 2 zur Verfügung stehen soll, hängt es der Admin aus und mountet es neu mit dem Typ »ext2«. Soll dies dauerhaft sein, ist der Dateisystemtyp in »/etc/fstab« zu ersetzen.

Je größer das Dateisystem, desto mehr Zeit nimmt die Dateisystemprüfung in Anspruch, die voreingestellt in festen Intervallen stattfindet. Falls der Check trotz Journal zu lange dauert oder die Intervalle zu kurz sind, passen zwei Optionen dieses Verhalten an. Mit der Option »-i« stellt der Anwender die Tage, Wochen oder Monate ein, nach denen E2fsck die Analyse durchführen soll. Der Check kann auch von einer bestimmten Anzahl von Mounts abhängen. Das lässt sich über die Option »-c« einstellen. Das Argument »0« bei beiden Optionen deaktiviert den Dateisystemcheck.

## Formatieren erlaubt

Einen ersten Test führt der Benutzer am besten auf einer kleinen Partition durch. Ohne Optionen aufgerufen ermittelt das Formatierprogramm »mkfs.ext3« alle erforderlichen Parameter selbst. Entsprechend einfach ist es, eine Partition zu formatieren:

```
mkfs.ext3 /dev/sdb2
```

▶

Die Blockgröße leitet das Programm von der Größe der Partition ab. Ist dies nicht gewünscht, übergibt der Anwender die Blockgröße (1, 2 oder 4 KByte) mit dem Schalter »-b« selbst:

```
mkfs.ext3 -b 2048 /dev/sdb2
```

Auch für den jeweiligen Einsatzzweck bringt »mkfs.ext3« Optionen mit. Die Option »-T« erlaubt Einstellungen dazu, wie groß die Dateien im Dateisystem hauptsächlich sein werden. Das Programm ermittelt die optimalen weiteren Einstellungen. Die angenommene Größe variiert zwischen 4 KByte (»news«) und 4 MByte (»largefile4«). Das folgende Kommando nutzt den Mittelweg und geht davon aus, dass die meisten Dateien 1 MByte groß sind:

```
mkfs.ext3 -T largefile
```

## Bad Blocks aufspüren

Es kann nie schaden, vor dem Installieren des Dateisystems die Festplatte zu prüfen. Die Option »-c« weist das Mkfs.ext3-Programm an, den Datenträger auf bad Blocks zu untersuchen. Diese Option sollte gerade bei älterer Hardware grundsätzlich zum Zug kommen. Wer ganz genau wissen will, wie es um den Datenträger steht, gibt sie zweimal hintereinander an. Dann führt Mkfs.ext3 einen Read-write-Test statt nur einen Read-only-Test durch. Der dauert jedoch fast doppelt so lange.

Für das Journal nimmt Mkfs.ext3 ebenfalls Optionen entgegen. Der Anwender wählt, ob das Journal intern im Superblock gespeichert wird (»size = 1024«) oder extern auf einem anderen Device (»device = /dev/hdc2«). Bei der externen Variante muss das Journal bereits existieren, bevor das Dateisystem via Mkfs.ext3 aufgebaut wird.
Dazu sind zwei Schritte nötig. Zuerst formatiert der Anwender die Partition, auf der das Journal liegen soll. Dabei muss er die gleiche Blockgröße (1, 2 oder 4 KByte) verwenden wie auf dem Dateisystem, das von dort das Journal einliest. Dann legt folgender Befehl das Journal für »hdb1« auf dem externen Device »hda9« an:

```
mkfs.ext3 -O /dev/hdb1 /dev/hda9
```

Das Dateisystem auf »hdb1« findet das externe Journal über die Option »-J«:

```
tune2fs -J device=/dev/hda9
```

Sobald alle Optionen vorhanden sind, lässt sich mit der Option »-n« das Anlegen des Dateisystems simulieren. So ist auf den ersten Blick zu erkennen, ob das Optionsset passt. Es gibt noch weitere Optionen, die bei Spezialfällen interessant sind. Damit reserviert der Benutzer Root zum Beispiel mehr als fünf Prozent der Festplattenpartition für sich (»-m«) oder schaltet Funktionen aus, die auf Ext 3 standardmäßig aktiv sind (»-O«).
Wichtige Programme sind auch Dumpe2fs, das den Inhalt des Superblocks und der Block-Gruppen ausgibt, oder E2fsck, das das Dateisystem auf Fehler prüft. Dieses Programm läuft bei einem Ext-3-System wegen des Journals sehr kurz. Einzige Ausnahme: Der Superblock ist beschädigt und erfordert einen Check. Wer ein LVM einsetzt (siehe Beitrag ab Seite 54), vergrößert oder verkleinert das Device per E2fsadm, das Online- und Offline-Resizing beherrscht. Die Dump- und Restore-Utilities von Ext 2 funktionieren auch mit Ext 3 weiter, die Benutzer müssen sich nicht umgewöhnen.

## Magerkost JFS

Die Portierung von JFS ist bereits so weit, dass Tests ohne weiteres möglich sind. Die Untauglichkeit für den Produktiveinsatz (zumindest gegenwärtig und auf Linux) demonstriert die sehr kurze Optionenliste recht gut. Andrerseits erreichen die JFS-Benchmarks Spitzenwerte (siehe Artikel ab Seite 38). Für ausführliche Analysen enthält der Kernelcode eine Debugging-Schnittstelle. Im User-Land steht dem Administrator das Werkzeug Jfs_fscklog aus den Jfsutils [2] zur Verfügung, das den Inhalt der Fsck-Service-Log-Datei extrahiert und auf der Konsole ausgibt.
Bei den Anwenderoptionen ist JFS vergleichsweise mager ausgestattet. So ist es beispielsweise nicht möglich, die Blockgröße einzustellen. Allerdings sind die Manpages sehr gepflegt und offerieren im Gegensatz zu den Dokumenten

**Tabelle 1: XFS-Sysctls**

| Option | Min.-Wert | Default | Max.-Wert | Auswirkung |
|---|---|---|---|---|
| fs.xfs.stats_clear | 0 | 0 | 1 | Auf »1« gesetzt löscht es die Statistik in »/proc/fs/xfs/stat«. |
| fs.xfs.sync_interval | HZ | 30*HZ | 60*HZ | Setzt das Intervall, in dem »xfssyncd« die Metadaten auf die Disk schreibt. |
| fs.xfs.error_level | 0 | 3 | 11 | Setzt den Umfang der Fehlerberichte. Generiert mehr oder weniger detaillierte Informationen und Backtraces für Dateisystem-Shutdowns. Folgende Errorlevel-Werte sind definiert: OFF: 0; LOW: 1, HIGH: 5. |
| fs.xfs.panic_mask | 0 | 0 | 127 | Debugging-Option. Bei bestimmten Fehlern wird die Funktion »BUG()« aufgerufen. |
| fs.xfs.irix_symlink_mode | 0 | 0 | 1 | Kontrolliert, ob symbolische Links mit dem Standardmodus 0777 angelegt werden oder ob der Modus von der »umask«-Vorgabe abhängt (Irix Mode). |
| fs.xfs.irix_sgid_inherit | 0 | 0 | 1 | Kontrolliert die Dateien, die der Benutzer in SGID-Verzeichnissen anlegt. Die Option veranlasst, dass das ISGID-Bit gelöscht wird, sofern die Group-ID der neuen Datei nicht mit der tatsächlichen oder einer der zusätzlichen Group-IDs des Elternverzeichnisses zusammenpasst. |
| fs.xfs.restrict_chown | 0 | 0 | 1 | Wacht, ob unprivilegierte Benutzer das Kommando »chown« benutzen dürfen, um Dateien an andere Benutzer zu übertragen. |
| vm.pagebuf.stats_clear | 0 | 0 | 1 | Auf »1« gesetzt löscht es sofort die Statistik in »/proc/fs/pagebuf/stat«. |
| vm.pagebuf.flush_age | 1*HZ | 15*HZ | 300*HZ | Das Alter, von dem ab der Inhalt der Metadaten-Puffer auf den Datenträger geschrieben wird. |
| vm.pagebuf.flush_int | HZ/2 | HZ | 30*HZ | Das Interval, in dem die Liste der Metadaten-Puffer gescannt wird. |

der anderen Dateisysteme auch eindeutige Beispiele. Einige Mount-Optionen zielen ebenfalls auf die Fehlerbehandlung ab und sind auch eine wichtige Diskussionsgrundlage der Mailingliste.

## Wenige JFS-Anwendungen

Mit dem Befehl »mkfs.jfs« erzeugt der Anwender ein Dateisystem. Über die Option »-c« identifiziert das Programm vor der Formatierung defekte Blöcke. Auch mit JFS ist es möglich, das Journal auf einem anderen Device anzulegen (»-J«). Das Kommando

```
mkfs.jfs -J journal_dev external-journal
```

generiert das Journal auf dem gegebenen Device. Anschließend erfolgt via »device« die Anbindung an das Dateisystem, das danach erzeugt wird:

```
mkfs.jfs -J device=external-journal device
```

Das »device« am Zeilenende entspricht der Partition, auf dem das Dateisystem entsteht. Entweder »journal_dev« oder »external-journal« darf der Benutzer in einem Befehl verwenden. Muss das Dateisystem zu OS/2-Systemen kompatibel sein, ist die Option »-O« dafür zuständig, die Case-insensitive-Unterstützung zu aktivieren.

Drei weitere JFS-Anwendungen stehen zur Verfügung: Mit »jfs_tune« lassen sich unter anderem das Volume-Label und die UUID ausgeben. Per »jfs_debugfs« ersetzen Admins Daten an einem Offset durch einen Hex-String. Das Analyseprogramm »jfs_fscklog« extrahiert den Inhalt des Fsck-Service-Logfile und zeigt diesen an.

## Reiser im Speed-Rausch

ReiserFS v3.6 ist ein stabiles Dateisystem und Spielzeug für Performance-Fans zugleich. Das Paket Reiserfsprogs **[3]** enthält diverse Programme mit Tuning-Optionen und Fehler-Recoverytools. Allerdings enthält das Utility-Paket keine Dump- und Restore-Programme. Immerhin ist es möglich, das Dateisystem zu vergrößern und zu verkleinern.

Seltsam: Obwohl ReiserFS auf Suse Linux das Standarddateisystem ist, findet sich dort keinerlei Information auf der Platte. Zwar existiert das Formatierprogramm »mkfs.reiserfs«. Eine Manpage oder ein paar Hilfedateien im »/usr/share/doc«-Verzeichnis sucht der Anwender aber vergebens.

Auch in der Kernel-Dokumentation befinden sich keine Hinweise. Es bleibt nur eine Möglichkeit: rein ins Internet und

**Ext 3: Mount-Optionen**

Genaue Informationen zum Ext-3-Dateisystem sind unter »/usr/src/linux/Documentation/filesystems/ext3.txt« zu finden.

**data=journal:** Zuerst wandern die Metadaten ins Journal, erst dann die Daten ins Dateisystem (Metadaten- und Daten-Journaling).

**data=ordered:** Zuerst werden die Daten ins Dateisystem geschrieben, erst danach die Metadaten angepasst.

**data=writeback:** Mal so, mal so. Die Funktion »sync()« schreibt die Daten in festen Intervallen. Das Metadaten-Journal wird davor oder danach aktualisiert.

**journal=update:** Update des Ext-3-Journals auf das momentane Format.

**journal=*inum*:** Mit »*inum*« übergibt der Benutzer Root die Inode-Nummer, die das Journal-File repräsentiert. Wird ignoriert, falls bereits ein Journal existiert.

**noload:** Das Journal wird nicht geladen.

**minixdf:** Der Befehl »df« reagiert wie beim Minix-Dateisystem.

**bsddf:** Der Befehl »df« reagiert genauso wie auf dem Betriebssystem Berkeley-Unix (BSD).

**check=none, nocheck:** Führt keinen Check der Bitmaps beim Mounten durch.

**debug:** Der Kernel schickt Debugging-Information an den Syslog-Protokolldienst.

**errors=continue:** Das Mounten erfolgt auch dann, wenn ein Fehler im Dateisystem vorliegt.

**errors=remount-ro:** Bei einem Fehler wird das Dateisystem read-only neu gemountet.

**errors=panic:** Die Maschine hält bei einem Dateisystemfehler an.

**grpid, bsdgroups:** Gibt einem Objekt die gleiche Group-ID wie dem Eltern-Objekt.

**nogrpid, sysvgroups:** Neue Objekte erhalten die Group-ID des Erstellers.

**resuid=*UID*:** Hier steht die User-ID, die reservierte Blöcke benutzen darf.

**resgid=*GID*:** Hier steht die Group-ID, die reservierte Blöcke benutzen darf.

**sb=*Offset*:** Benutze einen alternativen Superblock an dieser Stelle.

Hinweis: Ext 2 ignoriert die Quota-Optionen wie »grpquota«, »noquota«, »quota« und »usrquota«, ohne darüber eine Meldung auszugeben.

die Website des Entwicklers Hans Reiser [3] ansteuern. Das lohnt sich ohnehin, weil die Seite unter anderem auch Patches für besonders Mutige bereithält und die Mitglieder der Mailingliste viele Fragen beantworten.

Ein Dateisystem anlegen gelingt mit dem Befehl »mkreiserfs«. Eine Option die Blockgröße einzustellen existiert zwar, bloß unterstützt ReiserFS v3.6 lediglich 4096 Byte. Dies soll sich erst mit der Version 4 ändern, die irgendwann dieses Jahr herauskommt. Weitere Optionen wie den Hash-Algorithmus (»r5«, »rupasov«, »tea«) oder ob 3.5 oder 3.6 installiert wird, sucht ReiserFS selbst aus, falls diese nicht spezifiziert werden.

Die Optionen für das Journal erstrecken sich auf seine Größe (»-s«), ob es auf einem externen Device liegt (»-j«) und ab welchem Offset es startet (»-o«). Profis geben außerdem die maximale Größe der Transaktion vor. Mit diesen Angaben lässt sich weitgehend gefahrlos spielen, denn Mkreiserfs passt die Größe automatisch an, wenn der Benutzer einen falschen Wert übergibt. Wer die UUID (Universally Unique Identifier) einsetzt, kann sie mit der Option »-u« übergeben. Ohne diese Option erledigt dies Mkreiserfs selbst.

## ReiserFS reparieren

Gleich nach dem Hochfahren tritt das Check- und Reparaturprogramm Reiserfsck auf den Plan. Bei einem Absturz kontrolliert es die Konsistenz des Dateisystems mit der Standardeinstellung »check« und stellt den letzten im Journal gespeicherten Zustand wieder her. Stellt »check« einen schweren Fehler wie die Inkonsistenz des Dateisystems fest, fordert es den Benutzer auf, selbst tätig zu werden. Gut: »check« liefert gleich die richtige Option, die das System reparieren soll, an die Konsole.

Bei kleineren Inkonsistenzen wie ungültigen Verzeichniseinträgen empfiehlt es »fix-fixable«. Hat es das Dateisystem jedoch schwerer erwischt, muss der Benutzer es mit der Option »rebuild-tree« versuchen. Damit baut Reiserfsck den gesamten Dateisystembaum neu auf. Das dauert sehr viel länger und kann dazu führen, dass dieses Dateisystem anschließend komplett zerstört ist.

Der Fehlermeldung »read_super_block: cant't find a reiserfs file system« begegnet der Anwender mit:

```
reiserfsck -rebuild-sb
```

Der Superblock ist in diesem Fall wahrscheinlich korrumpiert, sodass das Dateisystem nicht angesprochen werden kann. »rebuild-sb« stellt den Superblock wieder her, falls sich auf der Partition auch tatsächlich ein Reiser-Dateisystem befindet.

## Dateisystem vergrößern

Das Resize-Programm »resize_reiserfs« ist noch im Betastadium und daher mit Vorsicht zu bedienen. Das Utility beherrscht sowohl die Vergrößerung als auch die Verkleinerung des Dateisystems, das während dieser Aktion nicht gemountet sein darf. Gibt der Admin keine Größe an (»-s«), expandiert das Tool das Dateisystem so weit wie möglich. Die Bedienung ist auf den ersten Blick recht einfach.

```
resize_reiserfs -s -2G /dev/hdb2
```

verkleinert das Dateisystem auf »hdb2« von der ursprünglichen Größe um 2 GByte. Um das Dateisystem um 2 GByte wachsen zu lassen, ist »-2G« durch »+2G« zu ersetzen. Leider ist die Sache nicht ganz so simpel. Per Fdisk muss der Benutzer auch die Partition an die neue Dateisystemgröße anpassen. Bei einer

### JFS: Mount-Optionen

Genaue Informationen zum JFS-Dateisystem sind unter »/usr/src/linux/Documentation/filesystems/jfs.txt« zu finden.

**iocharset=*Name*:** Zeichensatz, den das System nutzen soll, um von Unicode nach Ascii zu konvertieren.

**resize=*Anzahl*:** Vergrößert das Volume um diese Anzahl Blocks online.

**nointegrity:** Verhindert, dass das Journal beschrieben wird. Interessant, wenn ein Volume-Backup zurückzuschreiben ist.

**integrity:** Das Journal protokolliert Änderungen in den Metadaten. Dies sollte der Anwender nach einem Restore mit der Option »nointegrity« zusammen mit einem Remount benutzen.

**errors=continue:** Das Mounten wird ganz normal durchgeführt, auch wenn ein Fehler im Dateisystem vorliegt.

**errors=remount-ro:** Bei einem Fehler wird das Dateisystem read-only neu gemountet.

**errors=panic:** Die Maschine hält bei einem Dateisystemfehler an.

### XFS: Mount-Optionen

Genaue Informationen zum XFS-Dateisystem sind unter »/usr/src/linux/Documentation/filesystems/xfs.txt« zu finden.

**biosize=*Wert*:** Setzt die präferierte Größe für gepufferten I/O (Standard: 64 KByte), wobei »*Wert*« der Logarithmus der gewünschten Größe sein muss.

**ikeep:** Leere Inode-Cluster behalten, statt diese zum freien Speicherplatz hinzuzufügen.

**logbufs=*Wert*:** Setzt die Anzahl von Log-Puffern im Hauptspeicher. Gültig sind die Werte »2« bis »8«. Kann die Leistung erhöhen, reduziert jedoch den freien Speicher.

**logbsize=*Wert*:** Setzt die Größe der Log-Puffer. Gültige Werte sind 16, 32, 64, 128, 256 KByte. Systeme mit mehr als 32 MByte RAM benutzen standardmäßig 32 KByte.

**logdev=*device*:** Aufenthaltsort des externen Logs (Journals).

**noalign:** Mit dieser Option werden Daten-Allokationen nicht an den Rändern von Stripe-Units angereiht.

**noatime:** Aktualisiert die Zeitstempel bei Lesezugriffen nicht.

**norecovery:** Mountet das Dateisystem ohne Log-Recovery. Das Dateisystem muss read-only gemountet werden, sonst funktioniert der Mount nicht.

**osyncisosync:** Führt Timestamp-Updates bei Dateien durch, die das »O_SYNC«-Flag gesetzt haben. Ohne diese Option soll die Performance zwar besser sein, allerdings verlieren die Dateien bei einem Absturz die Timestamp-Informationen.

**quota, usrquota, uqnoenforce:** Aktiviert das User-Quota und erzwingt optional, dass die Limits eingehalten werden.

**grpquota, gqnoenforce:** Aktiviert das Gruppen-Quota und erzwingt optional, dass die Limits eingehalten werden.

**sunit=*Wert* und swidth=*Wert*:** Setzt die Größe der Stripe-Unit in 512 Byte großen Schritten (Block Units). »swidth« muss gesetzt werden und ein Vielfaches vom »sunit«-Wert sein.

Wichtig für Raid-Devices: Falls sich das Disk-Layout ändert, nachdem das Dateisystem angelegt wurde, kann Root so die Informationen im Superblock überschreiben.

**nouuid:** Keine Prüfung per UUID durchführen, ob Dateisysteme doppelt gemountet sind. Hilfreich beim Mounten von LVM-Snapshots.

**ReiserFS: Mount-Optionen**

Genaue Informationen zum ReiserFS-Dateisystem sind auf [www.namesys.com/mount-options.html] zu finden.

**conv:** Damit kann ein ReiserFS v3.6 ein v3.5-Dateisystem mounten. Es benutzt dabei den v3.6-Code für neu angelegte Objekte. Danach funktionieren die Utilities von v3.5 für dieses Dateisystem nicht mehr.

**dontpanic:** Ignoriert Input/Output-Fehler, während das Journal bearbeitet wird. Diese Option ist nur im Modus »raw« verfügbar.

**hash=rupasov:** Sehr schnelle Hash-Funktion für riesige Verzeichnisse und viele ungewöhnliche Dateinamen. Gefährlich zu benutzen, da Hash-Collisions sehr wahrscheinlich sind.

**hash=tea:** Offeriert hohen Zufall und geringe Wahrscheinlichkeit von Hash-Collisions. Senkt die Leistung, sollte jedoch verwendet werden, falls beim »r5«-Hash die Meldung »EHASHCOLLISION« auftritt.

**hash=detect:** Nur sinnvoll, wenn ein altes Dateisystem zum ersten Mal gemountet wird. Erkennt die verwendete Hash-Funktion und schreibt diese in den Superblock.

**hash=r5:** Standard-Hash.

**hashed_relocation:** Versucht, den Block-Allokator zu tunen. Könnte für eine Performance-Steigerung sorgen (muss aber nicht). Siehe auch »no_unhashed_relocation«.

**no_unhashed_relocation:** Versucht, den Block-Allokator zu tunen. Könnte für eine Performance-Steigerung sorgen (muss aber nicht). Siehe auch »hashed_relocation«.

**noborder:** Deaktiviert den »border allocator algorithm«. Könnte für eine Performance-Steigerung sorgen.

**nolog:** Deaktiviert das Journaling. Könnte für eine Performance-Steigerung sorgen. Achtung: Das schnelle Recovery nach einem Crash ist damit ebenfalls deaktiviert.

**notail:** Deaktiviert das »tail«-Feature. Falls im normalen Betrieb Probleme mit einem Programm auftreten (wie etwa Lilo), sollte ReiserFS winzige Dateien nicht direkt im Baum speichern.

**pgc=low, high:** Setzt die Aktivität des Passive Garbage Collector. Funktioniert nur zusammen mit dem Treiber »reiserfs-raw«, der sich nicht im Standardkernel befindet.

**raw:** Mountet das Dateisystem im Modus »raw«. Nicht verfügbar in Standardkernels.

**replayonly:** Spielt die Transaktionen im Journal zurück, ohne das Dateisystem zu mounten. In der Regel benutzt nur »fsck« diese Option.

**resize=*Wert*:** Online-Vergrößerung einer Partition, insbesondere LVM-Devices (Logical Volume Management). Benötigt das Resizer-Utility, das sich im Paket Reiserfs-utils befindet. Download von der Namesys-FTP-Site unter: [ftp://ftp.namesys.com/pub/reiserfsprogs]

Vergrößerung muss dies vorher geschehen, bei einer Verkleinerung erst nachher. Die Größe des Journals sowie die maximale Größe der Transaktion lässt sich mit dem Programm Reiserfstune ändern. Zudem kann der Benutzer das Journal damit auf ein anderes Blockdevice verschieben.

Das Programm Debugreiserfs hilft bei der Fehlereingrenzung. Ohne Optionen aufgerufen gibt es den Superblock des angegebenen Device aus. Interessant in diesem Zusammenhang ist die Option »-p«. Damit findet und extrahiert Debugreiserfs die Metadaten aus dem Dateisystem. Das ReiserFS-Team benutzt sie, um den Fehler zu beseitigen:

```
debugreiserfs -p /dev/hda2/ | ➥
gzip -c > hda2.gz
```

Ein Nachteil für viele Ext-2/3-Anwender, die sich eine Migration auf ReiserFS überlegen: Bis jetzt existiert kein Dump- und Restore-Utility. Die ReiserFS-FAQ empfiehlt Tar für inkrementelle Backups. Anwender der Suse-Pro-Distribution erhalten ein Full-Backup über Yast 2.

## XFS: Für Anspruchsvolle

Neben Ext 3 macht XFS den komplettesten Eindruck. Es bringt nicht nur zahlreiche Optionen mit, auch die Dokumentation ist vorbildlich. Als einziges Dateisystem erlaubt es, Sysctls im Proc-Dateisystem zu setzen (Tabelle 1). Leistungshungrige Anwender verwöhnt XFS mit Optionen für diverse Puffergrößen. Hervorzuheben sind außerdem der Quota-Support im Kernel und die passenden Tools im Paket Xfsprogs [4]. Zudem enthält das Paket XFS-eigene Dump- und Restore-Tools.

## Volle Funktionsvielfalt

Im einfachsten Fall formatiert der Anwender eine Partition mit dem Befehl:

```
mkfs.xfs /dev/sda1
```

Soll das Journal auf einer anderen Partition residieren, ist dies mit folgendem Befehl zu erreichen:

```
mkfs.xfs -l logdev=/dev/sdb2,size=10000b ➥
/dev/sda1
```

formatiert die erste Partition der ersten SCSI-Platte und legt das Journal in die zweite Partition der zweiten Platte.

Bei Blockgrößen ist XFS sehr flexibel, leider aber noch nicht auf Linux. Normalerweise sind Blockgrößen von 512 Byte bis 64 KByte möglich. Auf Linux darf die Größe maximal der Pagesize entsprechen. Und diese liegt auf x86-Linux-Systemen bei 4 KByte. Immerhin ist eine kleinere Größe wählbar.

Die weiteren XFS-Optionen sind nach Dateisystem, Inodes, Journal und Real-Time gruppiert. Jede der Gruppen versorgt der Anwender mit Angaben zum Aufenthaltsort auf der Festplatte und der jeweiligen Größe. So lässt sich beispielsweise die Größe der Stripe-Unit bei Raid-Systemen setzen oder die Anzahl der variablen Inodes als Prozentzahl angeben. Einen Überblick der momentan eingestellten Parameter liefert Mkfs.xfs mit der Option »-N«.

Auch XFS kann der Administrator vergrößern: Xfs_growfs dehnt den Datenteil mit der Option »-d« bis zur maximal möglichen Größe. Mit dem Großbuchstaben »-D« übergibt der Admin die gewünschte Größe in Blöcken. Ebenso verfährt er mit dem Real-Time-Abschnitt des Dateisystems, das auf die Optionen »-r« und »-R« reagiert.

Nicht zuletzt kommen auch Profis auf ihre Kosten. Mit dem Programm Xfs_db ist es möglich, bestimmte Blöcke auszulesen oder zu beschreiben. Beispielsweise liefert die Option »blockuse« die Datei zurück, die einen bestimmten Block benutzt. Die Liste der Xfs_db-Optionen ist riesig und unterstreicht eindrucksvoll, dass Linux dank XFS nun auch mit den großen Unix-Systemen konkurrieren kann. ■

**Infos**

[1] E2fsprogs: [http://e2fsprogs.sourceforge.net/ext2.html]
[2] Jfsutils: [http://www-124.ibm.com/jfs]
[3] Reiserfsprogs: [http://www.namesys.com/download.html]
[4] Xfsprogs: [http://oss.sgi.com/projects/xfs]