Das ext2-Dateisystem

18. Februar 2004

Geschichte

Linux wurde unter Minix entwickelt. Dieses Betriebssystem hatte ein einfaches Dateisystem, das zudem noch sehr gut getestet war. So war das erste Dateisystem, das Linux benutzen konnte, das Dateisystem von Minix. Dieses enthielt jedoch einige empfindliche Einschränkungen, wie zum Beispiel eine maximale größe von 64MB und eine Beschränkung der Dateinamenlänge auf 14 Zeichen. Um diese zu umgehen, wurde 1992 das ext FS (extended filesystem) entwickelt, und in den Kernel 0.96c aufgenommen. Daraus entwickelte sich das heutige ext2 und ext3.

Der Aufbau von ext2

Aus der Sicht des Dateisystems ist das Speichermedium eine aneinandergereihte Kette von Blöcken. Wie und wo die Dateien auf dem Medium organisiert sind, ist Sache des Gerätetreibers.

Die Blockgröße von ext2 kann 1, 2, 4 oder 8 kb (nur auf Alpha-architektur) groß sein. Sie wird beim erstellen des Dateisystems festgelegt und kann im nachhinein nicht verändert werden.

Größere Blöcke ergeben eine bessere performance, da mehr Daten in einem Zug eingelesen werden können. Allerdings kann kein Block von 2 Dateien gleichzeitig belegt werden. Nicht voll ausgenutzte Blöcke sind daher verlorener Speicherplatz.

Wird auf einem Dateisystem mit einer Blockgröße von 1024 byte eine Datei mit 1025 byte abgespeichert, so belegt diese 2 Blöcke und 2048 byte Speicherplatz.

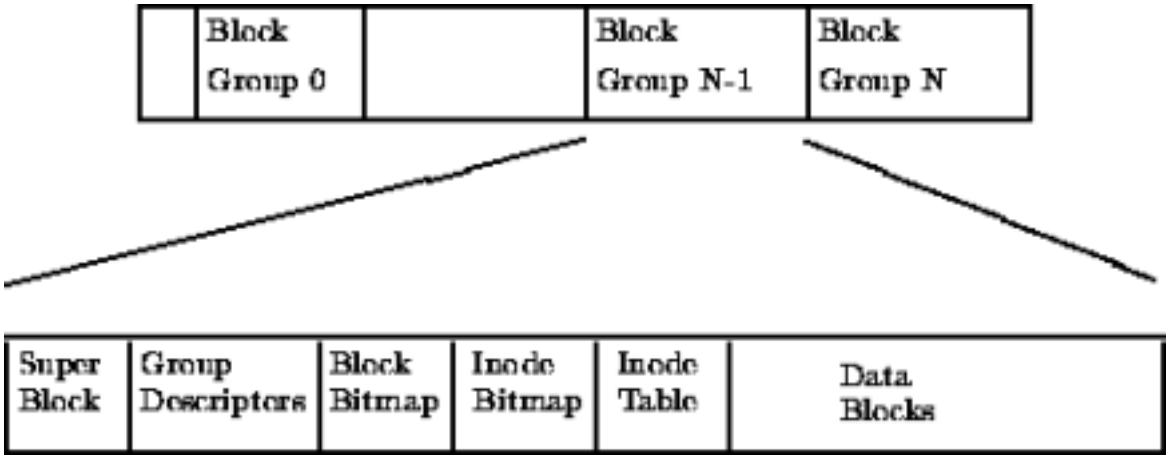
Block groups

Das ext2-Dateisystem unterteilt die logische Partition die es benutzt in *Block Groups* auf. Jede Block Group enthält sowohl für das Dateisystem benötigte Informationen, als auch Daten.

- Da die Verwaltungsstrukturen nahe bei den zugehörigen Datenblöcken liegen, werden Bewegungen des Festplattenkopfes vermieden.
- Da alle wichtigen Verwaltungsstrukturen redundant abgelegt sind, können diese Daten bei Beschädigung der ersten, normalerweise benutzten Block group, einfach aus einer der anderen Block groups gelesen werden.

Jede Blockgruppe besteht aus

- Superblock
- Group Descriptors
- Block Bitmap
- Inode Bitmap
- Inode Table
- Datenblöcke



Der Superblock

Der Superblock behinhaltet Informationen über Art, Zustand und Eigenschaften des Dateisystems. Die Hauptkopie liegt am Anfang des Dateisystems, und wird benötigt, um das Dateisystem mounten zu können.

Da er so wichtig ist, sind Kopien des Superblocks auf das gesamte Dateisystem verteilt. Vom Dateisystem genutzt wird aber nur der Superblock in Blockguppe 0.

Ursprünglich enthielt jede Blockgruppe eine Kopie des Superblocks. Als die Festplatten jedoch größer wurden und damit die Anzahl (und somit der Speicherplatz) der Block Groups anstieg, wurde diese übertriebene Redundanz vermindert.

In modernen Versionen des ext-Dateisystems kann festgelegt werden, daß nur noch in den Block Groups 0, 1, und in den durch 3, 5 und 7 teilbaren Block Groups Kopien abgelegt werden sollen.

Der Supberblock beinhaltet:

- Magic Number
  Anhand der Magic Number können Programme erkennen, daß es sich um ein ext2-Dateisystem handelt.
- Revision Level und Kompatibilitätsfelder
  Das Revision Level ist wie eine Versionsnummer. So kann beim mounten festgestellt werden, welche Features das Dateisystem unterstützt.
- Mount Count, Maximum Mount Count und Datum des letzten Dateisystemchecks
  Diese Felder ermöglichen es den Bootscripten, nach Erreichen von den in *Maximum Mount Count* festgelegten Bootvorgängen, oder nachdem eine bestimmte Zeit zwischen zwei Dateisystemchecks überschritten wurde, das Dateisystem auf Fehler zu überprüfen.
- Blockgröße
  Die Größe der Blocks auf diesem Dateisystem in byte. Diese Zahl kann nach dem Erstellen des Dateisystems nicht mehr verändert werden.
- Anzahl der Blocks pro Blockgruppe
  Die Anzahl der Blocks pro Blockgruppe. Diese Zahl kann nach dem Erstellen des Dateisystems nicht mehr verändert werden
- Freie Blöcke
  Anzahl der freien Blöcke im Dateisystem
- Freie Inodes
  Anzahl der noch freien Inodes im Dateisystem
- Erste Inode
  Die Inode-nummer der ersten Inode des Dateisystems. In einem root-dateisystem wäre das die Inode für das root-Verzeichnis („/")

Die Group Descriptor - Tabelle

Ein Group Descriptor beschreibt alle Block Groups. Er ist redundant in allen Block Groups vorhanden.

Er enthält folgende Informationen:

- Adresse des Blocks Bitmaps
- Adresse der Inode Bitmaps
- Adresse der Inode Tabellen

– Anzahl der freien Blöcke und Inodes

Der Vorteil dieser Einteilung ist, daß die die Metadaten einer Datei nahe bei ihren Daten befinden. So ist ein schneller Zugriff ohne große Kopfbewegungen der Festplatte möglich.

Block- und Inode Bitmaps

Ein Bitmap ist ein ,,Abbild" der freien und belegten Blöcke in der Block Group. Jeds bit in einem Bitmap repräsentiert einen Block (oder im Falle der Inode Bitmap eine Inode). Daran ob das bit gesetzt ist oder nicht kann man erkennen, ob der Block belegt ist oder nicht.

Block Bitmap

Aus dem Block Bitmap ist ersichtlich, welche Blocks im Dateisystem belegt sind

Inode Bitmap

Aus Inode Bitmap ist ersichtlich, welche Inodes im Dateisystem belegt sind

Inodes (index nodes)

Jede Datei im EXT2-Dateisystem hat eine zugehörige Inode. Diese beinhaltet Metadaten [1] über die Datei.

In den Inodes sind folgende Informationen gespeichert:

- Art der Datei
- Dateizugriffsrechte
- Anzahl der links zu der Datei
- UID (user-ID)
- GID (group-ID)
- Dateigröße
- Zeiger auf Datenblöcke der Datei
- Datum des letzten Zugriffs
- Datum der letzten Veränderung
- Erstellungsdatum

Inodes sind in einer Tabelle abgespeichert. Welche Inodes belegt sind, wird in einem Bitmap gespeichert.

[1] Metadaten sind Informationen (Daten) über andere Daten

Art der Datei

- Verzeichnis
- Link
- Block device
- Charakter device
- FIFO

Dateizugriffsrechte Für jede Datei unter UNIX können folgende Berechtigungen gesetzt sein:

- Lesen
- Schreiben
- Ausführen

Diese Berechtigungen sind für jeweils 3 Gruppen abgespeichert:

- Benutzer
- Gruppe
- rest der Welt

Anzahl der links zu der Datei Dazu später mehr.

UID Unter UNIX hat jeder benutzer eine *User-ID* (UID). Hier wird gespeichert, wem die Datei gehört.

GID (group-ID) Hier wird anhand der *Group-ID* (GID) gespeichert, zu welcher Gruppe die Datei gehört.

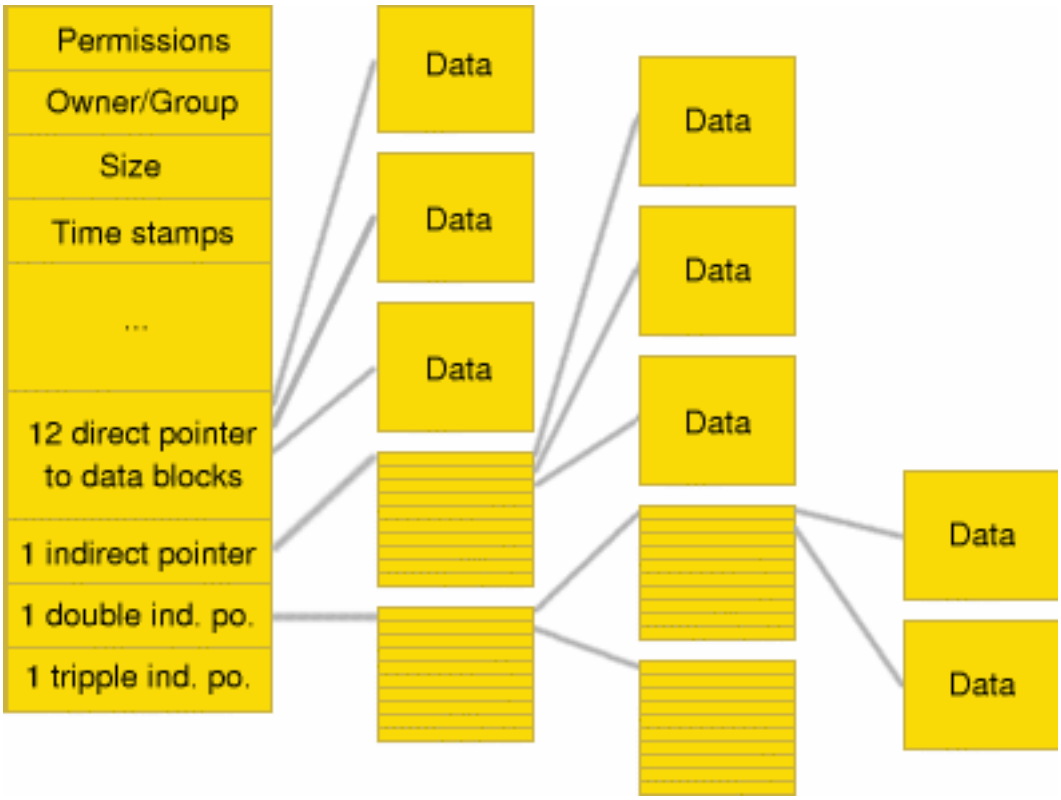
Dateigröße Größe der Datei

Zeiger auf Datenblöcke der Datei Die ersten 12 Zeiger in der Inode zeigen direkt auf die ersten 12 Datenblöcke, in der die zur Inode gehörige Datei gespeichert ist.

Der nächste Zeiger zeigt auf einen indirekten Datenblock, also einem Datenblock, in dem Wiederum Zeiger gespeichert sind, die auf die Daten der Datei zeigen.

Der darauf folgende Zeiger zeigt auf einen doppelt indirekten Datenblock, der auf einen indirekten Datenblock zeigt.

Der letzte Zeiger zeigt auf einen dreifach-indirekten Datenblock, der auf einen doppelt indirekten Datenblock zeigt.



Verzeichnisse

Unter ext2 sind Verzeichnisse Dateien des Typs ,,Directory", in denen eine Zuordnung von in dem Verzeichnis enthaltenen Dateien mit ihren Inodes abgespeichert ist.

Sie enthält für jede Datei jeweils die Inode-nummer, die Länge des Dateinamens und den eigentlichen Dateinamen.

In jedem Verzeichnis gibt es mindesten die Dateien ,,." (eigene Inode-nummer) und ,,.." (die Inode-nummer des übergeordneten Verzeichnisses).

Links

Hard links Ein Link ist ein zweiter Verzeichniseintrag für eine schon vorhandene Datei.

Dazu wird einfach ein Verzeichniseintrag erstellt, der auf eine schon anderswo eingetragene Inode zeigt. Das Link-count-Feld in der Inode wird danach um eins erhöht. Wird eine der beiden Dateien gelöscht, wird der Link-count wieder vermindert. Ist er nach dem löschen einer Datei auf ,,0", sind keiner weiteren

Links vorhanden und der von der Datei belegte Speicherplatz kann freigegeben werden.

Es können keine Links auf Dateien in einem anderen Dateisystem erstellt werden, da die Inode auf die verwiesen wird ja nur im aktuellem Dateisystem vorhanden ist.

Hard links sind außer den schon erwähnten, automatisch angelegten Dateien „." und „.." nicht erlaubt. Zum einen könnte man Endlosschleifen produzieren, schlimmer allerdinge wäre, daß beim rekursivem [2] löschen aus versehen Dateien gelöscht werden könnten, die nicht offensichtlich sind, zum Beispiel wenn sich ein link auf das root-Verzeichnis in dem zu löschenden Verzeichnis befindet.

Ein Hard Link ist eine ganz normale Datei. Legt man einen Hard Link zu einer Datei an, so sind beide Dateien vollkommen gleichberechtigt. Die ursprüngliche Datei ist nicht „Originaler" als die Datei die als zweites angelegt wurde.

Symbolische Links Die beiden beschränkungen der Hard links kann man mit Symbolischen links umgehen. Symbolische links sind Dateien des Typs „link", die einen Pfadnamen enthalten.

Wird eine Symbolischer link geöffnet, wird dessen Inhalt automatisch eingelesen und der Pfadname der darin enthaltenen Datei geöffnet. Dies geschieht vollkommen transparent, Anwendungen bemerken also keinen Unterschied zwischen normalen Dateien und symbolischen Links.

Das ist bei links (Verknüpfungen) unter Windows anders, da die Umsetzung der links hier nicht auf Dateisystemebene sondern durch die Applikation geschieht. Versteht ein Programm das Format des links nicht (z.B. Notepad), öffnet es nicht die Datei auf den die Verknüpfung verweist, sondern den link an sich.

Ist der name des links kleiner als 60 bytes, wird er innerhalb der inode in den Feldern gespeichert, die normalerweise die Pointer auf die Datenblöcke der Datei beinhalten würden. Durch diese Optimierung wird vermieden, daß die Datei einen ganzen Datenblock, also im günstigsten Fall immerhin noch 1024 byte, belegt.

Reservierte Blöcke

Eine beliebige Zahl von Blöcken (meist 2% der partition) können für einen bestimmten User (sinnvollerweise root) reserviert werden. Dadurch wird gewähleistet, daß das System auch dann funktionsfähig bleibt, wenn ein User den gesamten Platz der Partition aufgebraucht hat.

Erweiterbarkeit

Durch den *Feature compatibility*-Mechanismus können Erweiterungen am Dateisystemcode vorgenommen werden, ohne die Abwärtskompatibilät aufgeben zu müssen.

[2] Beim *rekursivem* löschen werden Unterverzeichnisse mit einbezogen

Dazu gibt es im Superblock drei, jeweils 32 bit große Felder in denen abgespeichert ist, wie sich ein Kernel ohne unterstützung bestimmter Features zu verhalten hat.

- Kompatible Features
  Die hier verzeichneten Features sind 100% kompatibel mit älteren Versionen. Auch ein Kernel, der nichts von diesen Features weiß, kann das Dateisystem sicher mounten und beschreiben.

  Das Journaling in ext3 ist so ein Feature, da das Journal einfach in einer Datei des Hauptverzeichnisses gespeichert wird.

- read-only kompatible Features
  Diese Features können von einem älteren Kernel nur lesend benutzt werden. Ein schreibender Zugriff auf das Dateisystem würde dessen Struktur zerstören.

  Ein Beispiel für ein solches Feature ist die schon erwähnte, weniger redundante Verteilung der Superblocks auf die Blockgroups. Da in Versionen, die dieses Feature unterstützen Datenblöcke an Stellen vorkommen können, an denen in früheren Versionen nur der Superblock erlaubt war, würde ein schreibender Zugriff zu inkonsistenzen in der Belegungstabelle (Bitmaps) führen.

- Inkompatible Features
  Durch veränderte Datenstrukturen ist es dem älteren Kernel nicht möglich, auf das Dateisystem zuzugreifen.

  Dieses Feld wird auch von ext3 benutzt um nach einem absturz zu verhindern, daß ein älterer Kernel ein Dateisystem mountet, ohne vorher das Journal zurück zu spielen.

Fragmentierung

Eine Datei ist framentiert, wenn Sie nicht in zusammenhängenden Blöcken auf dem Datenträger gespeichert ist. Wird die Datei gelesen, muss der Festplattenkopf größere Wege zurücklegen um die einzelnen Teile der Datei zusammenzusuchen, was die Zugriffszeit vergrößert.

Wird eine Datei zum Schreiben geöffnet so steht noch nicht fest, welche Größe Sie am Ende annehmen wird. Daher reserviert ext2 nicht nur den Speicherplatz den die Datei gerade benötigt, sondern bis zu 7 Blöcke im vorraus *(Preallocation)*. Wird die Bearbeitung einer neuen Datei beendet, gibt ext2 die nicht benötigten Blöcke wieder frei, verbraucht die Datei alle Blöcke, liegen diese nun in einem Stück auf der Platte. So bleibt der Grad der Fragmentierung einer Datei in akzeptablen Grenzen.

Durch den Aufbau von ext2 (Metadaten werden nahe bei den Daten abgelegt, Preallocation bei schreibzugriffen) entsteht unter ext2 weder eine große Fragmentierung, noch hat diese große Auswirkungen.

Außerdem muss man beachten, daß GNU/Linux ein Multiuser und Multitasking Betriebssystem ist. Es greift also nicht immer nur exklusiv ein Programm,

sondern viele verschiedene Programme gleichzeitig auf unterschiedliche Bereiche der Festplatte zu, der Festplattenkopf wird also sowieso mehr bewegt.

Die Grenzen von ext2

Verzeichnisse Ein Verzeichnis kann nicht mehr als 32.768 Unterverzeichnisse enthalten

Die maximale Anzahl Dateien in einem Verzeichnis beträgt über 130 Trillionen Dateien. Praktisch wird diese Zahl in den aktuellen Versionen jedoch dadurch begrenzt, daß es ab 10000 Dateien zu Performanceproblemen bei Dateioperationen kommen würde.

Dateien Im Dateinamen sind alle Zeichen außer ,,\0" und dem slash ,,/", der unter Unix den Verzeichnistrenner darstellt, erlaubt. Möchte man solche Dateien allerdings unter der Kommandozeile ansprechen, muss man bestimmte Zeichen *Escapen*, also dem Befehlsinterpreter mitteilen, daß es sich bei dem Zeichen nicht um einen Shellefehl, sondern um einen Dateinamen handelt.

Die maximale länge eines Dateinamens beträgt 255 Zeichen

Partitions- und Dateigrößen

Blockgröße	:	1kB	2kB	4kB
max. Dateigröße	:	~16 GB	256 GB	2 TB
max. Dateisystemgr.	:	2 TB	8 TB	16 TB

siehe http://www.suse.de/~aj/linux_lfs.html

Journaling

Journaling ist eine Technik, die hilft die Integrität (also den korrekten Zustand) von Dateisystem zu bewahren.

Bei Dateisystemen ohne Journaling, muss nach einem nicht ordnungsgemäßen herunterfahren des Betriebssystems das gesamte Dateisystem auf Fehler überprüft werden. Dieser Vorgang ist bei den heutigen Plattengrößen sehr zeitwaufwendig (mehrere Stunden) und für ein Produktivsystem nicht akzeptabel.

Journaling speichert alle Änderunge am Dateisystem (z. B. die Umbenennung einer Datei), als Transaktion im Journal. Die Änderung kann dann im Fall eines Absturzes entweder abgeschlossen oder noch nicht abgeschlossen sein. Wenn eine Transaktion zum Absturzzeitpunkt (oder im Normalfall, wenn das System nicht abstürzt) abgeschlossen war, ist garantiert, dass alle an dieser Transaktion beteiligten Blöcke einen gültigen Dateisystemstatus repräsentieren. Diese

Blöcke werden dann ins Dateisystem kopiert. Wenn eine Transaktion zum Absturzzeitpunkt nicht abgeschlossen war, kann nicht garantiert werden, dass die beteiligten Blöcke konsistent sind, daher wird eine solche Transaktion verworfen (das bedeutet, dass die Dateisystemänderung, die diese Transaktion repräsentierte, verloren geht).

Ext3 beherrscht als einziges unter den modernen Linux-Dateisystem auch Data-Journaling. Es kann nicht nur die Integrität des Dateisystems an sich, sondern auch die Integrität der Daten garantieren. Dieser Modues ist allerdings nicht standardmäßig aktiviert. Er hat den Nachteil, daß er relativ langsam ist.

Ext3 ist zu ext2 voll abwärtskompatibel. Ein ext3-Dateisystem kann auch als ext2 gemountet werden. Das Journal wird dann natürlich nicht benutzt. Es erscheint als normale Datei im Hauptverzeichnis.

Der Befehl zum hinzufügen eines Journals lautet

```
tune2fs -j /dev/hdXX
```

und dauert nur wenige sekunden.

Zusammenfassung

Es wird versucht, die Inodes zusammen mit den dazugehörigen Datenblocken in derselben Blockgruppe unterzubringen. Dadurch werden Kopfbewegungen der Festplatte vermieden.

Obwohl ext2 relativ alt ist, kann es sich auch gegen moderne Dateisystem recht gut behaupten. Dies hat es vor allem seiner Erweiterbarkeit zu verdanken durch die es möglich war, aus ext2 ein Journaling Filesystem zu machen.

Performancemäßig ist ext2 ziemlich ausgeglichen und tut sich weder besonders positiv noch besonders negativ hervor, wenn man es mit jüngeren Dateisystem wie XFS, Reiserfs, und JFS vergleicht.

Quellen

http://www.science.unitn.it/~fiorella/guidelinux/tlk/node95.html
http://www.linuxgazette.com/issue93/tag/2.html
http://kris.koehntopp.de/artikel/diplom/node13.html
http://portal.suse.de/sdb/en/2002/06/ext2frag.html
http://de.wikipedia.org/wiki/Ext2
http://web.mit.edu/tytso/www/linux/ext2intro.html
http://www.science.unitn.it/~fiorella/guidelinux/tlk/node95.html
http://www.win.tue.nl/~aeb/linux/lk/lk-8.html
http://tldp.org/LDP/tlk/fs/filesystem.html
http://uranus.it.swin.edu.au/~jn/explore2fs/es2fs.htm
http://web.mit.edu/tytso/www/linux/ext2intro.html
http://www.linux-ag.de/linux/LHB/node177.html

http://www.de-locher.ch/linux/paper/fs/ext.php

/usr/src/linux/Documentation/filesystems/ext2.txt
/usr/src/linux/Documentation/filesystems/ext3.txt