Betriebssysteme-6

Verfasser Kacper Bohaczyk
Datum 26.05.2022

Einführung

Das Dokument beinhaltet das Wissen für einen guten Systemtechniker.

Projektbeschreibung

Arbeitsschritt

File Allocation Table(FAT) ist eine Tabelle mit fester Größe, die zu jeden Cluster des Dateisystems sagen kann, ob er frei ist, das Medium beschädigt ist, oder ob er belegt ist. Bei dem Fall eines belegten Clusters Speicher der FAT die Adresse des nächsten Clusters, oder das EOF(End of File). Es gibt viele FAT-Systeme. Die wichtigsten sind FAT12, FAT16, FAT32(Sie waren die Standart-Dateisysteme des DOS und Windows). Die Zahl im Versionsnamen eines FAT-Dateisystems gibt die Länge der Cluster an. Beispiel: FAT12 (12 Cluster), FAT16 (16 Cluster) oder FAT32 (32 Cluster).
Bereich-1 Bootsektor
Bereich-2 Reservierte Sektoren
Bereich-3 FAT1
Bereich-4 FAT2
Bereich-5 Stammverzeichnis
Bereich-6 Daten

Im Bootsektor befinden sich Informationen über: die Blockgröße des Speichermediums, Anzahl der Blöcke pro Cluster, die Anzahl der Blöcke auf dem Speichermedium, die Beschreibung des Speichermediums und eine Angabe der FAT-Version

Zwischen dem Bootsektor und FAT können optionale reservierte Sektoren sein. Im Stammverzeichnis wird jedes Verzeichnis durch ein Eintrag repräsentiert. FAT12 und FAT16 --> Stammverzeichnis direkt hinter FAT --> feste Größe. FAT 32 --> beliebige Stelle --> variable Größe. Stammverzeichnis enthält einen Dateinamen, sowie einen Datumseintrag und eine Dateigröße. CODE.C --> 1240 Bytes ---> zwei Cluster. Cluster 402 --> End of File. OS.DAT --> analog zur ersten Datei. FILE.TXT --> kleiner als ein Cluster --> Eintrag ihres Clusters (581)--> EOF. Probleme die bei Dateisystemen die auf FAT basieren vorkommen sind: verlorene Cluster und querverbundete Cluster. Mit Hilfe von FAT und den Clusternummern zum nächsten Cluster ergibt sich eine Clusterkette. Eintrag von Cluster 12 in der FAT --> Fehler. Cluster 29 --> Teil der Clusterkette von Datei 4. Behebung von Fehlern --> chkdsk(Dos,Windows) fsck.fat(Linux).

FAT12

FAT12 --> 1980 --> QDOS --> MS-DOS --> Clusternummern 12 Bits lang ---> max. 4096 Cluster. --> Clustergrößen: 512 Bytes, 1kB, 2kB und 4kB --> maximale Dateisystemgröße 16MB

FAT16

FAT16--> 1983 --> 65.524 Cluster verfügbar --> 12 Cluster reserviert --> effektiv nur 65.524 Cluster verfügbar --> Cluster können zwischen 512 Bytes und 256 kB groß sein. Die Clustergröße kann beim Erzeugen des Dateisystems festgelegt werden --> kleinen Partionen große Cluster(in praxis selten Sinnvoll). Primäre Einsatzgebiet --> mobile Datenträger (USB-Sticks) --> bis zu 2GB --> größeren Dateisystemen ist der Einsatz wegen des Speicherverlustes nicht sinnvoll

FAT32

FAT32 --> 1997, Clusternummern --> 32 Bits lang, 4 Bits reserviert --> 268.435.456 Cluster können adressiert werden. Können zwischen 512 Bytes und 32 kB groß sein. Clustergröße kann beim Erzeugen des Dateisystems festgelegt werden. maximale Größe ist 4GB. primärer Einsatzbereich --> mobile Datenträger.

VFAT

VFAT --> 1997 ---> Erweiterung für FAT12/16/32 --> ermöglicht längere Dateinamen --> bis zu 255 Zeichen. Kodierung --> Unicode. maximal 20 Pseudo-Verzeichniseinträge. Die ersten vier Bits haben den Wert 1 bei einem VFAT-Eintrag. Die Großkleinschreibung wird angezeigt, aber ignoriert. VFAT --> speichert für jede Datei einen eindeutigen Dateinamen im Format 8.3. Bei Betriebssystemen ohne der VFAT-Erweiterung ignorieren die Pseudo-Verzeichniseiträge und zeigen nur den verkürzten Dateinamen an --> dadurch können Microsoft-Betriebssystemen ohne der VFAT Erweiterung auf die Dateien mit langen Dateinamen zugreifen. Alle Sonderzeichen und Punkte werden gelöscht und alle Kleinbuchstaben werden durch Großbuchstaben ersetzt --> es werden nur die ersten 6 Buchsstaben beibehalten. Die ersten drei Zeichen nach dem Punkt werden beibehalten und der Rest gelöscht. Falls zwei Dateien denselben Namen haben wird die zweite als --Y EINTRAG~1.pdf gespeichert --> falls eine Datei namens EINTRAG~1.pdf existiert wird hochgezählt also --> EINTRAG~2.pdf.

Journaling-Dateisysteme

Schreibzugriffe --> überführen Daten von einem konsistenten Zustand zu einem neuen konsistenten Zustand. Kommt es während eines Schreibzugriffs zum Ausfall ..Y wird die Konsistenz des Dateisystems überprüft. Dateisystem mehrere Gigabyte groß --> kann mehrere Stunden oder Tage dauern. überspringen der Konsistenzprüfung ist nicht sinnvoll. Eingrenzung der Konsistenzprüfung --> Journal-Dateisysteme --> sammelt Schreibzugriff --> es müssen nach dem Absturz nur die Dateien überprüft werden, die im Journal stehen --> Nachteil: zusätzliche Schreibzugriffe. Es existieren verschiedene Journal-Konzepte.
Metadaten-Journaling --> Cache Schreibstrategie Write-Back --> erfasst nur Änderungen. --> sync --> überträgt die Änderungen im Page Cache auf den Datenspeicher --> Konsistenzprüfung --> wenige Sekunden --> Nachteil: nur Konsistenz der Metadaten nach Absturz garantiert --> Datenverlust weiterhin möglich.
Vollständiges Journaling --> erfasst alle Änderungen an den Metadaten und alle Änderungen an den Cluster der Dateien im Journal --> Konsistenz der Daten garantiert --> alle Schreibzugriffe werden doppelt ausgeführt. Meisten Linux-Dateisysteme --> standardmäßig Kompromiss beider Journaling-Konzepte -->
Ordered Journaling --> erfasst nur Änderungen an den Metadaten --> Änderungen an den Cluster von Dateien werden erst im Dateisystem durchgeführt und danach werden erst die Änderungen an den Metadaten ins Journal geschrieben --> Konsistenzprüfung dauert nur wenige Sekunden --> ähnliche Schreibgeschwindigkeit wie beim Metadaten-Journaling --> nur Konsistenz der Metadaten ist garantiert.

Extent-basierte Adressierungen

Blockadressierung --> Aufwand für die Verwaltungsinformationen. Steigende Speicherkapazität der Datenspeicher --> jede Inode adressiert eine geringe Anzahl an Clusternummern --> große Daten belegen

zahlreiche zusätzliche Cluster zur indirekten Adressierung --> jede Inode kann maximal 48kB direkt adressieren. Lösung --> Extends
Extent-basierter Adressierung --> große Dateiberieche auf zusammenhängende Bereiche --> nur drei Werte anstatt vieler Clusternummern. Beispiele für Dateisysteme die via Extends die Cluster addressieren isnd JFS, XFS, Btrfs, NTFS und ext4.

ext4 --> Fourth Extended Filesystem --> Journaling Dateisystem --> Länge wurde auf 48 Bits vergrößert. Adressierung mit Extents. reduzierter Verwaltungsaufwand. Jede Inode 60 Bytes für einen Extent-Header(12 Bytes und zur Adressierung von vier Extents (jeweils 12 Bytes.) Ext4-Inode kann 512MB direkt adressieren.

NTFS --> Anfang der 1990 --> Nachfolger der FAT-Dateisysteme --> Entwickelt von Microsoft --> neuere Versionen sind zu früheren Versionen abwärtskompatibel --> vergrößerter Funktionsumfang. transparente Kompression und Verschlüsselung via Triple-DES und AES ab Version 2.x --> Unterstützung für Kontingente ab Version 3.x --> maximale Dateisystemgröße von 256TB. Clustergrößen können 512 Bytes bis 64kB groß sein. Dateinamen gehen bis zu einer Länge von 255 Unicode-Zeichen. Aufbau von NTFS --> MFT --> Referenzen die Extents und Cluster einer Datei zuordnen und Metadaten --> Dateigröße, das Datum der Erstellung, das Datum der letzten Änderung und der Datentyp, eventuell noch der Dateiinhalt. Cluster können zwischen 512 und 64 kB groß sein

Copy-on-Write --> ändert bei einem Schreibzugriff nicht den Inhalt der Originaldatei, sondern schreibt den veränderten Inhalt als neue Datei --> Metadaten werden auf die neue Datei angepasst --> bis sie angepasst wird bleibt die Originaldatei enthalten und kann nach einem Systemabsturz weiter verwendet werden --> bessere Datensicherheit --> ältere Versionen bleiben vorgehalten

Quellen

[1] Buch 2022 "Betriebssysteme kompakt_Springer Buch"