File System
la potenza nasce dall'hard disk

Luca Berton
mr.evolution@tiscali.it
Magnetic recording process.
numero settore e ECC (Error Correcting code)

rotazione

asse

settore

traccia

piatto
UNIX File System

A UFS volume is composed of the following parts:
* a few blocks at the beginning of the partition reserved for boot blocks (which must be initialized separately from the filesystem)
* a superblock, containing a magic number identifying this as a UFS filesystem, and some other vital numbers describing this filesystem's geometry and statistics and behavioral tuning parameters
* a collection of cylinder groups. Each cylinder group has the following components:
  * a backup copy of the superblock
  * a cylinder group header, with statistics, free lists, etc, about this cylinder group, similar to those in the superblock
  * a number of inodes, each containing file attributes
  * a number of data blocks

Inodes are numbered sequentially. The first several inodes are reserved for historical reasons, followed by the inode for the root directory. Directory files contain only the list of filenames in the directory and the inode associated with each file. All file metadata is kept in the inode.
<table>
<thead>
<tr>
<th>Developer</th>
<th>Microsoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Name</td>
<td>File Allocation Table</td>
</tr>
<tr>
<td>(12-bit version)</td>
<td>(16-bit version)</td>
</tr>
<tr>
<td>Introduced</td>
<td>1977 (Microsoft Disk BASIC)</td>
</tr>
</tbody>
</table>

**Structures**
- Directory contents: Table
- File allocation: Linked List
- Bad blocks: Cluster tagging

**Limits**
- Max file size: 32 MiB, 2 GiB, 4 GiB
- Max number of files: 4,077, 65,517, 268,435,437
- Max filename size: 8.3, or 255 characters when using LFNs
- Max volume size: 32 MiB, 2 GiB, 4 GiB with some implementations, 2 TiB

**Features**
- Dates recorded: Creation, modified, access
- Date range: January 1, 1980 - December 31, 2107
- Forks: Not natively
- Attributes: Read-only, hidden, system, volume label, subdirectory, archive
- Permissions: No
- Transparent compression: Per-volume, Stacker, DoubleSpace, DriveSpace (No)
- Transparent encryption: Per-volume only with DR-DOS (No)
**scalability**: The largest volume size supported by FAT32 on-disk structures is 2 terabytes. Windows 2000 and later versions further limit the size of a volume that can be formatted with FAT32 to 32 GB. Because the maximum size of a file on FAT32 is limited to 32 bits, it cannot be used to store files larger than 4 GB.

**Data transfer limitations**: FAT does not provide support for alternate data streams, other than the default data stream. Copying data from an NTFS or UDF volume, which supports alternate data streams, to a FAT volume can cause data that is stored in the named streams to be lost. This makes FAT unsuitable for transferring data from an NTFS or UDF volume that contains named streams.

**No incremental-write support**: FAT requires its metadata structures to be overwritten in place during updates to the metadata. Because of this, FAT cannot be used on incremental-write removable media, such as CD-R and DVD-R.

**No defect management support**: FAT does not provide a mechanism for defect management at the file system level. The FAT file system expects the underlying hardware to manage and remap defective blocks on the media. It does, however, support marking some allocation units as "bad" to prevent their use by the file system. FAT cannot be used on removable rewritable media, such as CD-RW, DVD-RW, and DVD+RW. This is because these media types do not support defect management in the hardware layer.

**Performance and recoverability**: FAT does not provide clustering of metadata on the disk. The metadata is typically scattered all over the volume. This scattering of metadata causes performance degradation during runtime and also impacts the ability of the operating system to recover corrupted data.
Full name: New Technology File System
Introduced: July 1993 (Windows NT 3.1)
Partition identifier: 0x07 (MBR)
EBD0A0A2-B9E5-4433-87C0-68B6B72699C7 (GPT)

**Structures**
- Directory contents: B+ tree
- File allocation: Bitmap/Extents
- Bad blocks: Bitmap/Extents

**Limits**
- Max file size: 16 TiB with current implementation (16 EiB architecturally)
- Max number of files: 4,294,967,295 (2^32-1)
- Max filename size: 255 characters
- Max volume size: 256 TiB with current implementation (16 EiB architecturally)
- Allowed characters in filenames: any character except '\0' (NULL) and '/[

**Features**
- Dates recorded: Creation, modification, POSIX change, access
- Date range: January 1, 1601 - May 28, 60056
- Forks: Yes
- Attributes: Read-only, hidden, system, archive
- File system permissions: ACLs
- Transparent compression: Per-file, LZ77 (Windows NT 3.51 onward)
- Transparent encryption: Per-file, DESX (Windows 2000 onward), Triple DES (Windows XP onward), AES (Windows XP Service Pack 1, Windows Server 2003 onward)
Ext2

Developer Rémy Card
Full name Second extended file system
Introduced January 1993 (Linux)

Structures
File allocation I-nodes

Limits
Max file size 2 TiB
Max number of files 1018
Max filename size 255 characters
Max volume size 16 TiB
Allowed characters in filenames Any byte except NUL and '/'

Features
Forks yes

Attributes
File system permissions POSIX
Transparent compression Yes (optional)
Transparent encryption No

Supported operating systems Linux, BSD, Windows (through an IFS), MacOS
Developer: Namesys  
Full name: ReiserFS  
Introduced: 2001 (Linux 2.4.1)

**ReiserFS**

**Structures**
- Directory contents: B+ tree
- File allocation: Bitmap [1]

**Limits**
- Max file size: 8TiB
- Max number of files: 232 (~4 billion)
- Max volume size: 16TiB
- Allowed characters in filenames: All bytes except NUL and '/'

**Features**
- Dates recorded: modification (mtime), metadata change (ctime), access (atime)
- Date range: December 14, 1901 - January 18, 2038
- Forks: Extended attributes
- File system permissions: Unix permissions, ACLs and arbitrary security attributes
- Transparent compression: No
- Transparent encryption: No
- Supported operating systems: Linux
Developer Stephen Tweedie (ext3 design and implementation), Rémy Card (original ext2 design and implementation), Theodore Ts'o (tools and improvements), Andreas Gruenbacher (xattrs and ACLs), Andreas Dilger (online resizing), et al

Full name: Third extended file system

Introduced: November 2001 (Linux 2.4.15)

**Structures**
- Directory contents: Table, Tree
- File allocation: bitmap (free space), table (metadata)
- Bad blocks: Table

**Limits**
- Max file size: 16GiB – 2TiB
- Max number of files: Variable[1]
- Max filename size: 255 bytes
- Max volume size: 2TiB – 32TiB
- Allowed characters in filenames: All bytes except NUL and '/'

**Features**
- Dates recorded: modification (mtime), attribute modification (ctime), access (atime)
- Date range: December 14, 1901 - January 18, 2038
- Forks: Yes
- Attributes: No-atime, append-only, synchronous-write, no-dump, h-tree (directory), immutable, journal, secure-delete, top (directory), allow-undelete
- File system permissions: Unix permissions, ACLs and arbitrary security attributes (Linux 2.6 and later)
- Transparent compression: No
- Transparent encryption: No (provided at the block device level)
- Supported operating systems: GNU/Linux, BSD, Windows (through an IFS)
<table>
<thead>
<tr>
<th>max number of files</th>
<th>(2^{32} - 3 \Rightarrow 4 \text{ Gi} - 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>max number files a dir can have</td>
<td>518701895, but in practice this value is limited by hash function. r5 hash allows about 1 200 000 file names without collisions</td>
</tr>
<tr>
<td>max file size</td>
<td>(2^{31} - 1 \Rightarrow 2 \text{ Gi} - 1)</td>
</tr>
<tr>
<td>max number links to a file</td>
<td>(2^{16} \Rightarrow 64 \text{ Ki})</td>
</tr>
<tr>
<td>max filesystem size</td>
<td>(2^{32} \text{ (4K) blocks} \Rightarrow 16 \text{ Ti})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2^{32} - 3 \Rightarrow 4 \text{ Gi} - 3)</td>
</tr>
<tr>
<td>(2^{32} - 4 \Rightarrow 4 \text{ Gi} - 4) but in practice this value is limited by hash function. r5 hash allows about 1 200 000 file names without collisions</td>
</tr>
<tr>
<td>(2^{60} \text{ - bytes} \Rightarrow 1 \text{ Ei}, but page cache limits this to 8 Ti on architectures with 32 bit int)</td>
</tr>
<tr>
<td>(2^{32} \Rightarrow 4 \text{ Gi})</td>
</tr>
<tr>
<td>(2^{32} \text{ (4K) blocks} \Rightarrow 16 \text{ Ti})</td>
</tr>
</tbody>
</table>
Developer: Silicon Graphics Inc.
Full name: XFS
Introduced: 1994 (IRIX v5.3)

Structures
Directory contents: B+ trees
File allocation: extent based

Limits
Max file size: 9 exabytes
Max number of files
Max filename size: 255 bytes
Max volume size: 9 exabytes
Allowed characters in filenames: All bytes except NUL

Features
Dates recorded: Yes
Forks: Yes (called extended attributes)
File system permissions: Yes
Transparent compression: No
Transparent encryption: No (provided at the block device level)
Supported operating systems: IRIX, Linux, FreeBSD
Reiser4

Developer   Namesys
Full name      ReiserFS 4
Introduced    2004 (Linux)

Structures
Directory contents  Dancing B*-tree

Limits
Max file size  8TiB on x86

Allowed characters in filenames  All bytes except NUL and '/'

Features
Dates recorded     modification (mtime), metadata change (ctime), access (atime)
Date range       64-bit timestamps[1]
Forks            Extended attributes

Attributes
File system permissions  Unix permissions, ACLs and arbitrary security attributes
Transparent compression  Version 4.1 (beta)
Transparent encryption  Version 4.1 (beta)

Supported operating systems  Linux
**Internal JFS (potential) limits**

JFS is a full 64-bit file system. All of the appropriate file system structure fields are 64-bits in size. This allows JFS to support both large files and partitions.

**File system size**

The minimum file system size supported by JFS is 16 Mbytes. The maximum file system size is a function of the file system block size and the maximum number of blocks supported by the file system meta-data structures. JFS will support a maximum file size of 512 terabytes (with block size 512 bytes) to 4 petabytes (with block size 4 Kbytes).

**File size**

The maximum file size is the largest file size that virtual file system framework supports. For example, if the framework only supports 32-bits, then this limits the file size.

Developer  IBM
<table>
<thead>
<tr>
<th>Developer</th>
<th>Mingming Cao, Dave Kleikamp, Alex Tomas, Andrew Morton, others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full name</td>
<td>Fourth extended file system</td>
</tr>
<tr>
<td>Introduced</td>
<td>October 10, 2006 (Linux 2.6.19-rc2)</td>
</tr>
</tbody>
</table>

**Structures**
- Directory contents: Table, Tree
- File allocation: bitmap (free space), table (metadata)
- Bad blocks: Table

**Limits**
- Max volume size: 1024 PiB

**Features**
- Supported operating systems: Linux
Benchmarking Filesystems

COMPUTER: Dell Optiplex GX1
  CPU: Pentium III 500MHZ
  RAM:  768MB
  SWAP: 2200MB

CONTROLLER:
Maxtor Promise ATA/133 TX2 - IN PCI SLOT #1

DRIVES USED:
1] Seagate 400GB ATA/100 8MB CACHE 7200RPM
2] Maxtor 61.4GB ATA/66 2MB CACHE 5400RPM

DRIVE TESTED: The Seagate 400GB.
Benchmarking Filesystems

LIBC VERSION: 2.3.5
KERNEL: linux-2.6.14.4
COMPILER USED: gcc-4.0.3
EXT2: e2fsprogs-1.38/sbin/mkfs.EXT2
EXT3: e2fsprogs-1.38/sbin/mkfs.EXT3
JFS: jfsutils-1.1.8/sbin/mkfs.jfs
REISERFSv3: reiserfsprogs-3.6.19/sbin/mkreiserfs
REISERFSv4: reiser4progs-1.0.5/sbin/(Used patch reiser4-for-2.6.14-1.patch w/ libaal-1.0.5 + reiser4progs-1.0.5)
XFS: xfsprogs-2.6.36/sbin/mkfs.xfs
Benchmarking Filesystems

001] Create 10,000 files with touch in a directory.
002] Run 'find' on that directory.
003] Remove the directory.
004] Create 10,000 directories with mkdir in a directory.
005] Run 'find' on that directory.
006] Remove the directory containing the 10,000 directories.
007] Copy kernel tarball from other disk to test disk.
008] Copy kernel tarball from test disk to other disk.
009] Untar kernel tarball on the same disk.
010] Tar kernel tarball on the same disk.
011] Remove kernel source tree.
012] Copy kernel tarball 10 times.
013] Create 1GB file from /dev/zero.
014] Copy the 1GB file on the same disk.
015] Split a 10MB file into 1000/1024/2048/4096/8192 byte pieces.
016] Copy kernel source tree on the same disk.
017] Cat a 1GB file to /dev/null.
Grazie per l'attenzione ed a voi la parola