FATX

From Free60

FATX is the file system used by the Xbox and the Xbox 360, it is unsupported natively by Windows but has some functionality in Linux. Sometimes called "XTAF" (due to its little endian header), the file system is derived from the age-old MS-DOS file system (http://en.wikipedia.org/wiki/File_Allocation_Table) and can be considered as a cleaned-up version of it.

Note that, while not part of the file system itself, the media which contain this file system do not have a master file table which describes which file system starts where. It is up to the consumer (Xbox 360, geom_xbox360 kernel module (http://www.freebsd.org/cgi/query-pr.cgi?pr=kern/107707) , ...) to know this.

Contents

File System Layout

The file system is divided into 4 parts:

- Header (the BOOT sector on FAT file systems)
- File allocation table (FAT)
- Root directory cluster/data region

All multi-byte values contained in each part are [1 (http://en.wikipedia.org/wiki/Endianness)] big-endian.

Difference Between Retail and Development Kit HDD

The essential difference between retail and devkit HDDs is that devkit HDDs hold a partition table (length of 0x18 bytes) at the start of the drive. The table contains sector and length information for the Content and DEVKIT Partitions (possibly Compatibility partition too). This partition table only exists on Devkit HDDs, retails have the offsets built in to the kernel.

USB Drive Layout

Unlike the hard disks and memory units that the Xbox 360 originally shipped with, the USB drives which can be Xbox 360-configured contain sets of files located in a hidden "Xbox360" folder located in the root of the device. The Data0000-Data0003 files are **always** present no matter what the device size. The Data0000 file houses the Cache/USB System Extended partitions, device performance information, and device geometry (mainly just the total device size). Data0001 contains the file allocation table for the Data partition, which uses the remaining files.

There was some research done in hopes to remove the maximum device size of 16 GB (for Xbox 360 storage) on exploited/development consoles, but it was later discovered that anything a few GB larger than the current

maximum causes the console to crash. http://www.xboxhacker.org/index.php?topic=16913.0

Configuration file

The configuration is the first 2 sectors (0x400 bytes) of the Data0000 and is created when the device is configured. It contains info about the device and is secured with a signature. The layout is as follows:

Offset	Length	Description
0	0x1A8	Console Security Certificate
0x1A8	0x80	Signature (part of the console cert)
0x228	0x14	Device ID
0x23C	4 (UINT32)	Certificate size (0x228)
0x240	8 (UINT64)	Size of device in bytes
0x248	4 (UINT32)	Read speed in KBs
0x24A	4 (UINT32)	Write speed in KBs
0x24E	0x1B2	Padding (0x00)

Please not the above layout assumes the USB drive was configured on a Xbox 360, drives preformatted by Microsoft will be different.

The signature is derived from a SHA1 hash taken from the start of the device id (0x228) till the end of the config (0x1D8 bytes) and is signed with the consoles private key and the matching public parameters are present in the console certificate for verification later on.

The certificate size is used to determine how to verify the configuration file when the device is mounted. If the value is 0x228 then the device was configured by an Xbox and the signature is verified using the params in the console certificate, if the value is 0x100 then the device was pre-configured by Microsoft and is verified using the SATA public key (also used for HDDSS verification).

The last 2 values are written when the device is configured, it is unknown how these are used later on. Perhaps it could be used to determine what connected device would provide best performance for caching.

Partition Header

For each offset, add the offset of the partition.

Offset	Length	Type	Information
0x0	0x4	ascii string	Partition magic (XTAF)
0x4	0x4	unsigned int	Partition ID
0x8	0x4	unsigned int	Sectors per Cluster
0xC	0x4	unsigned int	Root directory cluster

SysExt Partition

The SystemUpdate with Kernel-Version 2.0.12611.0 (Kinect-Dashboard) introduced a new system to hold the Kinect-/Avatar related systemfiles. It creates a Partition named "SysExt" aka SystemExtended to store these files. You could call it a Sub-Partition or Hidden-Partition as it's a "Partition inside another Partition".

Partition Locations

Memory Unit

Offset	Length	Information	Format
0x0	0x7FF000	System Cache	SFCX (Secure File Cache for Xbox)
0x7FF000	end of drive	Data	FATX

Xbox 360 Hard Drive

Offset	Length	Information	Format
0x2000	0x204 - 0x80000	Security Sector	Binary
0x80000	0x80000000	System Cache	SFCX (Secure File Cache for Xbox)
0x80080000	0xA0E30000	Game Cache	SFCX (Secure File Cache for Xbox)
0x10C080000	0xCE30000	SysExt	FATX ("Sub"-Partition)
0x118EB0000	0x8000000	SysExt2	FATX ("Sub"-Partition)
0x120eb0000	0x10000000	Xbox 1 Backwards Compatibility	FATX
0x130eb0000	end of drive	Data	FATX

Development Kit HDD Partition Table

Offset	Length	Information	Value
0x0	0x4	HDD Header (Devkit identification)	0x00020000
0x4	0x4	?	?
0x8	0x4	Content Volume sector	0x00633000 (Raw address: 0xC6600000 = 0x633000 * 0x200)
0xC	0x4	Content Volume length (in sectors)	variable (real length = var * 0x200)
0x10	0x4	Xbox 360 Dashboard Volume sector	0x005B3000 (Raw address: 0xB6600000 = 0x5B3000 * 0x200)
0x14	0x4	Xbox 360 Dashboard Volume length (in sectors)	0x00080000 (0x10000000 = 0x80000 * 0x200)

USB Drive

Offset	Length	Information	Format
0x8000400	0x12000400	System Cache	FATX
0x8115200	0x8000000	SysExt ("Sub"-Partition)	FATX
0x12000400	0xDFFFC00	SysExt2 ("Sub"-Partition)	FATX
0x20000000	End of Files	Data	FATX

"Josh" Sector

The **"Josh" Sector** is located on the 4th sector (0x800). It's purpose is currently unknown. It may be used just to identify the Xbox that previously formatted it.

Offset	Length	Type	Information
0x0	0x4	ascii string	"Josh" magic

The Console Security Certificate then follows, continuing with a pair of entries, the format of which is below:

Offset	Length	Type	Information
0x22C	0x8	(un?)signed long	(First entry) Unknown (ID of some kind?)
0x234	0x14	bytes	(First entry) Unknown (0x14 bytes... SHA1 hash?)
0x248	0x4	(un?)signed int	(First entry) Unknown
0x24C	0x4	(un?)signed int	(First entry) Unknown
0x250	0x8	(un?)signed long	(Second entry) Unknown (ID of some kind?)
0x258	0x14	bytes	(Second entry) Unknown (0x14 bytes... SHA1 hash?)
0x26C	0x4	(un?)signed int	(Second entry) Unknown
0x270	0x4	(un?)signed int	(Second entry) Unknown

The sector then ends with the following format:

Offset	Length	Type	Information

0x274	0x4	(un?)signed int	Unknown
0x278	0x4	(un?)signed int	Unknown
0x27C	0x4	(un?)signed int	Unknown
0x280	0x4	(un?)signed int	Unknown

Another possible entry format is below, this disregards the end-sector format:

Offset	Length	Type	Information
0x22C	0x2	(un?)signed short	Unknown (ID of some kind?)
0x22E	0x2	(un?)signed short	Number of extra ints at end of entry
0x230	0x4	(un?)signed int	Unknown
0x234	0x14	bytes	Unknown (0x14 bytes... SHA1 hash?)
0x248	0x4	(un?)signed int	Unknown
0x24C	0x4 * **Number of extra ints at end of entry**	(un?)signed int	Unknown

I've uploaded my own "Josh" sector to here (http://files.aversionmedia.org/JoshSector.bin) (644 bytes) - Stoker25 19:48, 6 August 2010 (UTC)

All of the info above was gained using this sector, can anybody provide another one to make sure the format matches?

Security Sector

The **Security Sector** is used by Microsoft to verify that the harddrive is an original Microsoft product. The **Security Sector** holds details such as drive sector count, drive serial number and the Microsoft logo (to stop other companies using it)

It begins at the 16th sector on the drive (0x2000).

Offset	Length	Type	Information
0x0	0x14	ascii string	Serial Number
0x14	0x8	ascii string	Firmware Revision
0x1C	0x28	ascii string	Model Number
0x44	0x14	bytes	MS Logo Hash
0x58	0x4	unsigned int	Number of Sectors on drive
0x5C	0x100	bytes	RSA Signature
0x200	0x4	signed int	MS Logo Size
0x204	MS Logo Size	image	MS Logo

If a **Security Sector** from a smaller hard drive is used on a bigger one, the Xbox will only see the number of sectors defined in the **Security Sector**.

Chainmap and directories

To find the offset of the chainmap, take the offset of the position and add 4096 (0x1000) to it. Each entry in the chainmap is either an unsigned int or a unsigned short, depending on the number of clusters inside the partition.

To work out the cluster size, take the Sectors per Cluster value and multiply it by 512 (0x200).

To work out the number of clusters, divide the partitions size by the cluster size and that's have the number of clusters.

If the number of clusters is below 65520 (0xFFF0), then the drive uses 2-byte chainmap entries, otherwise it uses 4-byte chainmap entries. The size of the chainmap is the size of the chainmap entries multiplied by the partitions cluster count.

To get a clusters offset, you need to work out the offset of the file data area. This is determined by taking the chainmap offset and adding it's size to it. You then simply times the cluster index (minus 1) by the cluster size and add it to the file data offset.

To get a clusters chainmap entrys offset, times the cluster index by the chain map entry size and add that to the chainmap offset.

Files

The file contents is stored per cluster as indicated by the chainmap and the starting cluster (see below). If the file is larger than one cluster, it is stored in multiple clusters. If the length of the file is not a multiple of the cluster size, then the last cluster is only partially used. **If the file covers more than 1 cluster, the next cluster must be determined by finding the current cluster's entry in the chainmap and using the value as the next cluster.**

Directories

Directories are stored in a tabular format. Because directories are normal files with the "directory" bit set, they are allocated in the FAT and may therefore cover multiple clusters. This makes it possible to have many files in one directory.

Each entry of the directory table is 64 bytes long.

An entry can be set to all 0xFF bytes, which means that this entry is unused and probably marks the end of the directory contents. Used entries are filled as follows:

Offset	Size	Description
0x00	1	File name length, or 0xE5 for deleted files
0x01	1	File Attributes
0x02	0x2A	file name, padded with either 0x00 or 0xFF bytes
0x2C	4	First cluster of file, null/0x00 for empty files
0x30	4	File size
0x34	2	Creation date
0x36	2	Creation time
0x38	2	Last write date
0x3A	2	Last write time
0x3C	2	Last access time
0x3E	2	Last access time

The file flags and the date and time fields are in the same format as the one used in the FAT file system. For time, bits 15-11 represent the hour, 10-5 the minutes, 4-0 the seconds. For date, bits 15-9 are the year, 8-5 are the month, 4-0 are the day.

The volume bit is unused, the volume label is instead stored in a file "name.txt" in the root directory. The contents of this file is big-endian UTF-16 (Unicode string) starting at offset 0x2

The following characters are found in XTAF file names from actual disk images:

- 0x20, 0x24, 0x2e (SPACE $.)
- 0x30-0x39 (digits 0-9)
- 0x41-0x5a (letters A-Z)
- 0x5f (_)
- 0x61-0x7a (letters a-z)

Unlike the FAT file system, XTAF has no "." and ".." entries in the directory tables. This means that it's only possible to go to the parent directory by remembering its cluster number.

Limitations

Attribute	Limitation
Maximum Filename Length	0x2A (42)

Maximum Path Length	240 Characters
Maximum File Size	4 GB (4294967296 bytes)
Maximum File Per Directory	0x1000 (4096)
Possible Cluster Sizes	4 KB (0x1000 bytes, 0x8 sectors per cluster), 8 KB (0x2000, 0x10 SPC),16 KB (0x4000, 0x20 SPC), 32 KB (0x8000, 0x40 SPC), 64 KB (0x10000, 0x80 SPC)
File/Folder Name Characters	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!#$%&'()-.@[]^_`{}~ **(SPACE)**

Retrieved from "http://www.free60.org/index.php?title=FATX&oldid=2615"
Categories: Xbox System Software | File formats