Audio Engineering Society

Convention Paper

Presented at the 125th Convention
2008 October 2–5 San Francisco, CA, USA

File System Tricks for Audio Production

Michael Hlatky[1], Sebastian Heise[1], and Jörn Loviscach[1]

[1]*Hochschule Bremen (University of Applied Sciences), 28199 Bremen, Germany*

Correspondence should be addressed to Jörn Loviscach (joern.loviscach@hs-bremen.de)

ABSTRACT
Not every file presented by a computer operating system needs to be an actual stream of independent bits. We demonstrate that different types of virtual files and folders including so-called "Filesystems in Userspace" (FUSE) allow streamlining audio content management with relatively little additional complexity. For instance, an off-the-shelf database system may present a distributed sound library through (seemingly) standard files in a project-specific hierarchy with no physical copying of the data involved. Regions of audio files may be represented as separate files; audio effect plug-ins may be displayed as collections of folders for on-demand processing while files are read. We address differences between operating systems, available implementations, and lessons learned when applying such techniques.

1. INTRODUCTION

In general terms, a file system can be considered a special-purpose database which usually organizes physical data on a local storage device for saving, hierarchical organization, manipulation, navigation, access and retrieval. Furthermore, file systems can provide access to physical data on a remote file server by acting as client for a network protocol. However, file systems need not make use of a storage device at all. A file system can be used to organize and represent access to any data or information, whether it be stored or dynamically generated.

Standard audio editing and multimedia authoring software reads and writes data via files stored on local or remote hard disks. Content management operate on these files. Physically copying a file to incorporate it into a project is often disadvantageous: It takes time, consumes disk space and—which remains the vexing part in our age of terabyte disks—makes it hard to keep a sound library in a consistent state, since multiple copies of a single file are hard to track. A user may edit a copy of the file and never update all other copies, let alone change the file name of the edited copy.

Sophisticated asset management systems can be employed to address this issue. However, a range of techniques provided by current computer file systems also alleviate this situation. We survey these techniques, which are underrated and underused in audio production, and provide practical application examples. To this end, we built and evaluated a range of software prototypes.

Section 2 discusses file links such as ".lnk" files used by Microsoft Windows and symbolic links used by Unix-based systems such as Mac OS X, also to be found in Microsoft Windows Vista. "Smart Folders" in Mac OS X and "Search Folders" in Windows Vista represent stored file queries, which are addressed in Section 3. A "Filesystem in Userspace" (FUSE) is the most complex and versatile technique, to be dealt with in Section 4. Here, application software is presented with a standard file system hierarchy; behind the scenes, however, every access to the file system is mapped to routines freely definable by the developer.

2. LINKS

Links allow attaching different addresses to a single block of data. This can be employed to create shortcuts or temporary collections.

2.1. Introduction

File systems typically are hierarchically structured, using directories which can contain files as well as sub-directories. The most common local file systems are HFS Plus on computers running Apple Mac OS X, NTFS on computers running Microsoft Windows, and Ext2 and Ext3 on computers running Linux. Though aged, Microsoft's FAT32 file system is understood by every modern operating system and thus can serve as an exchange medium. However, it limits the maximum file size to four gigabytes, which renders it almost useless for media production.

The resolution of a file name to a physical location on the storage device is usually accomplished through a table which for every file points to the storage allocated on the disk. Depending on the file system's type, this table may also contain additional information, such as access rights, or it may point to additional files containing further metadata (inodes).

The name associated with a regular file is simply a label that directs the operating system to a specific stream of bits. Therefore it is called a hard link. The same stream of bits on the disk may be referenced to by several hard links, that is: It may seemingly appear with different names and/or under different paths. However, a hard link can only point to a physical position within one partition [9]. More than one hard link to a directory is not allowed in most operating systems, preventing endless recursions such as a hard link inside a directory pointing to the directory itself. A notable exception, however, is Mac OS X 10.5.

A soft link (also known as symbolic link, symlink), overcomes these limitations in that it is an additional file or inode, containing the linked-to file's name and path, therefore allowing spanning over different partitions or file systems. Furthermore, soft links also allow multiple links to directories.

In contrast to hard links, soft links can be broken by moving, renaming or deleting the linked-to file, since the soft link does not point to a physical storage location but only refers to a path and a name. In Apple Mac OS X this is optionally overcome with the help of alias files. An alias is a small file that represents another object in a file system through its "fingerprint," stored in the alias file's resource fork. If the original file was moved within the file system, and an open command is performed on the alias, the "Alias Manager" tries to locate the file [2].

On Microsoft Windows systems beginning with Windows 2000, an NTFS Junction Point can be used in a similar way to a Unix soft link, with the limitation that it can only link to a folder within the same partition [26]. To allow compatibility to Unix-like operating systems, NTFS symbolic links were introduced with Windows Vista, which allow pointing to point to local as well as remote files [22].

In addition to these two implementations of soft links, even older versions of the Microsoft Windows operating system support .lnk files. These are short cuts that correspond to the alias files and folders of Mac OS X. Genuine soft links are automatically resolved by the file system, so that any software accessing a soft link will see the target file instead. Windows' .lnk shortcuts, however, are treated as regular files. Thus, application software developers have to read and open the reference specified in a .lnk shortcut. Another difference is that .lnk shortcuts can only refer to a destination via an absolute path, whereas soft links also allow a relative path declaration, rendering soft links more portable. If the linked-to file of a shortcut is moved or renamed, the Windows IShellLink Interface tries to relocate it through the distributed link tracking service or standard search methods [21].

2.2. Applications. Lessons Learned

Links can for instance be applied to collect search results in a folder or to virtually collate the files used in a project. When building software that generates links, we quickly noticed, however, that virtual folders and FUSE systems can handle such applications much more elegantly, see Sections 3 and 4. Creating hundreds of .lnk files to display a search result is a rather clumsy operation, in particular as these .lnk files have to be deleted at some later point.

Links cannot only be used to quickly open files, but also to save files. To this end, one can create a short cut or a symbolic link to a directory. For instance, in response to the query of the user where to place a specific file he or she created, a database application may use this mechanism to present a ready-to-use collection of appropriate folders to upload the file to.

On Apple Mac OS X and Microsoft Windows operating systems, it is advisable to employ alias files and .lnk short cuts to provide access to data that stems from multiple locations, as this is the most robust solution: The files that are being linked to can be renamed or moved within the local system, but file access is still guaranteed. This solution has the drawback, however, that the links will only work within the operating systems in which they have been generated. A project folder containing links to several media assets distributed on several file systems is therefore platform dependent. Furthermore, the application software must be capable of resolving the original files. The developer of the application software has to take this into account, even though the change needed may be as small as setting the DereferenceLinks flag for the standard Windows file dialog.

In a heterogeneous production environment, a platform independent project folder containing genuine soft links will work in principle with both Microsoft Windows Vista and Apple Mac OS X, if supported by appropriate files system drivers. Nonetheless, this has the drawback that the links will break if the original files are moved or renamed. Furthermore, manually creating genuine soft links requires command-line operations or extensions to the standard graphical user interfaces: Apple Finder and Microsoft Explorer.

A shared project folder containing hard links to different files will work with every operating system, however is limited in that the linked-to files may only be located within the same disk partition. Furthermore, again the creation of such links is not readily supported by the standard graphical interfaces.

3. VIRTUAL FOLDERS

A virtual folder is understood to be a folder presented by an operating system that contains data which are located in different places. These data typically consist of search results, so that this technique can readily be applied to Music Information Retrieval.

3.1. Introduction

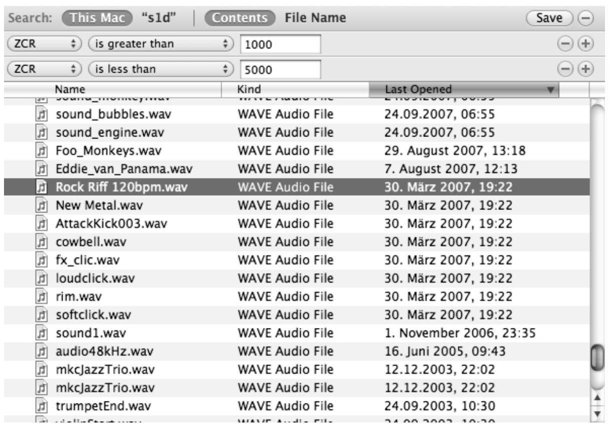
Virtual folders are often referred to as “stored queries.” The operating system executes these queries on the fly when the folder is opened. On accessing a file presented via a virtual folder, an application will automatically receive the true path of the file from the operating system.

By applying custom tags through a “Spotlight Metadata Importer” or through the “Windows Property System,” one can achieve similar results as with links. Whereas links have to be created and deleted like standard files, stored queries are completely dynamic and will automatically be updated when files are added, deleted or renamed in the target directories. This advantage comes at the price of a more advanced software development and of database queries that are executed whenever the stored query is accessed.

On Mac OS X one can tag all files related to a certain project which are to be accessed from one folder, but not physically copied, with a unique Spotlight comment (for instance through a hash function on the project’s name). One could then create a “Smart Folder” [3], setting the folder’s query to the tag assigned beforehand. The same approach can be applied with tags and “Search Folders” under Windows Vista. In both operating systems, the folder will then appear in the defined position within the file system, seemingly containing all files that carry the right tag. Basically, Smart Folders and Search Folders are XML documents instructing the search engine what and where to search for. The query may include properties such as sampling rate, bit depth, and the number of channels.

3.2. Applications

To demonstrate how virtual folders could be leveraged by audio production software, we extended Apple Spotlight by a proprietary property: the zero-crossing rate (ZCR) of WAV audio files. This enables the user to create virtual folders based on the values of this feature, see Figure 1. The feature extraction is executed in the background by the indexing process that is part of Spotlight.



ZCR is greater than 1000
ZCR is less than 5000

Name	Kind	Last Opened
sound_bubbles.wav	WAVE Audio File	24.09.2007, 06:55
sound_engine.wav	WAVE Audio File	24.09.2007, 06:55
Foo_Monkeys.wav	WAVE Audio File	29. August 2007, 13:18
Eddie_van_Panama.wav	WAVE Audio File	7. August 2007, 12:13
Rock Riff 120bpm.wav	WAVE Audio File	30. März 2007, 19:22
New Metal.wav	WAVE Audio File	30. März 2007, 19:22
AttackKick003.wav	WAVE Audio File	30. März 2007, 19:22
cowbell.wav	WAVE Audio File	30. März 2007, 19:22
fx_clic.wav	WAVE Audio File	30. März 2007, 19:22
loudclick.wav	WAVE Audio File	30. März 2007, 19:22
rim.wav	WAVE Audio File	30. März 2007, 19:22
softclick.wav	WAVE Audio File	30. März 2007, 19:22
sound1.wav	WAVE Audio File	1. November 2006, 23:35
audio48kHz.wav	WAVE Audio File	16. Juni 2005, 09:43
mkcJazzTrio.wav	WAVE Audio File	12.12.2003, 22:02
mkcJazzTrio.wav	WAVE Audio File	12.12.2003, 22:02
trumpetEnd.wav	WAVE Audio File	24.09.2003, 10:30

Fig. 1: A proprietary property can be employed in queries (upper left) to create virtual folders (lower right).

There is no limit to the number and the types of features to use; one can also form boolean expressions that take the values of different features into account or comprise additional properties such as bit depth or last access date. Features may be of low level such as the zero-crossing rate or they may be of high level such as music genres, in which case they can be represented through character strings.

3.3. Lessons Learned

In a homogeneous production environment, virtual folders work very well. One cannot, however, use a virtual folder created in Mac OS X on Windows Vista and vice versa. To the other operating system, they show up as XML documents defining a query. Another issue—even with a single operating system—is that the user cannot save files to virtual folders.

4. FILESYSTEMS IN USER SPACE

Representing data through virtual file systems has a long tradition in the Unix world. In audio production, the possibilities range from version control to batch processing.

4.1. Introduction

In 1984, Tom J. Killian created the first implementation of a process file system [18]. This software maps data about the processes running on a computer to a file system hierarchy. As this file system does not correspond to data on a disk, it is called a virtual file system. Further development [17] in virtual file systems led to SUN's VFS in 1985, which allowed Unix system calls to transparently access local UFS and remote NFS file systems.

File systems are among the most complicated items in computer technology, as one must have a deep understanding of the operating system's kernel to develop a file system. This is overcome with "Filesystems in Userspace" [15], kernel modules or drivers available for many operating systems. FUSE allows developers to create file systems without entering kernel space, as the file system code runs in user space, communicating only with the FUSE kernel module. FUSE was officially included in the 2.6.14 Linux kernel, rendering obsolete other implementations of userland file system drivers, such as the Linux Userland Filesystem (LUFS). FUSE was recently ported to Mac OS X by Google, yielding MacFUSE [20]. There are various attempts to port FUSE to Microsoft Windows [1, 13, 14, 34, 37, 38].

Existing FUSE applications that are interesting in an audio production environment comprise Twisted-FLAC [32], a FUSE file system for Mac OS X that converts FLAC audio files to the WAV format on the fly when read and vice versa when written, and MP3FS [10], a read-only FUSE file system that converts FLAC audio files to MP3 on the fly when read. Furthermore, there are various FUSE file system implementations that create hierarchical folder structures containing MP3 files based on their ID3 tags, for instance TagsFS [31] and MusicMeshFS [23].

In addition, many applications of FUSE support versioning, which can immediately be applied to content management, thus relieving non-expert users from details otherwise exposed by typical asset management systems. Examples of versioning file systems using the FUSE kernel module are WaybackFS [36] and Copy-FS [11]. In addition, it is possible to mount SUN's ZFS [25] file system via FUSE [39], which supports versioning.

4.2. Applications

Whereas FUSE systems can be employed for similar applications as the links described in Section 2, they support far more advanced applications. We demonstrate four of them.

First, audio files may contain separate regions, indicated by markers. In the common RIFF file formats such as WAV and AIFF the file is organized in different chunks that can store optional data. For instance the format chunk `fmt␣` chunk provides information about sampling rate and coding format. The cue chunk can be used to store cue points, defined by a sample offset. Through the playlist chunk `plst`, cue points can be set to in- and out-points, defining a region, which also can be assigned a certain name in the label chunk `labl`.

Standard audio editing software such as Sony Sound Forge [28] can define RIFF cue points, link them to different regions, and assign names to them. In our prototype, every region in a RIFF file is represented by a separate (virtual) file whose name equals the region's label. This facilitates for instance working with snippets of a speech of 90 minutes' length. If the regions are changed in the audio editing software, this is immediately reflected in the file system. The standard file browsers require, however, a refresh.

As can be seen in Figure 2, this FUSE application creates a new folder in the same hierarchy as the edited wave file, denoted with the file's name and a "regions" extension. This folder will then contain the regions as (virtual) files. This prototype was developed in C# based on the Dokan Library [13] with our own .NET wrapper. It runs on Microsoft Windows.

Second, we built a prototype that applies basic editing functions to audio files when read from a folder, for instance normalization, phase shift, low cut, gain adjustment, convolution with a given audio file, or

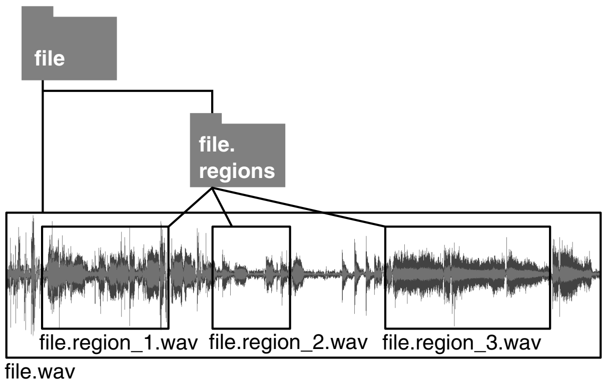


Fig. 2: The WAV region file system maps regions to virtual WAV files.

adding fades. We called this FUSE application "batch folders," indicating that it can easily be used to process a huge amount of audio data in a convenient way. If an audio file is read—which includes copying as well as playback—the processing is applied transparently to the reading software. This method could be called lazy batch processing. It may be helpful if some files of a collection have to be subjected to processing but the precise set is not known in advance.

The functionality of batch folders is shown in Figure 3. If a folder is created within the batch file system, the folder's name defines the processing to be applied. To this end, we compiled a list of keywords. The hierarchy of nested folders is analyzed during file access, granting the generation of processing chains. Folders can also be nested, so that an audio file can for instance first be normalized, then attenuated by 6 dB, faded in over one second and faded out over one second. In a chains, the processing defined by the topmost folder will be applied first, thus in the sequence as one would read (and create) the folders.

The prototype was again developed in C# using the Dokan Library [13]. It creates a background folder for temporal storage of the files.

Third, we built a prototype that represents the presets of VST 2 audio effect plug-ins as nestable folders, see Figure 4. The user can drag (virtual) copies of audio files into such folders. When these copies

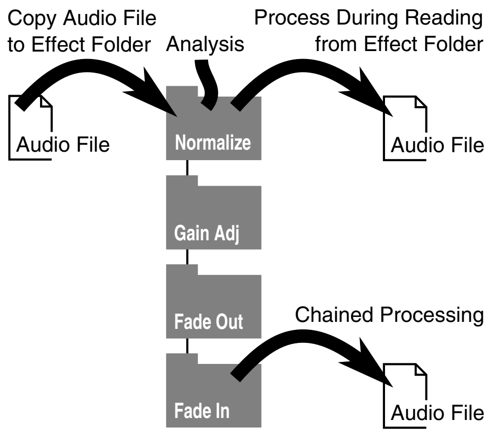


Fig. 3: The audio file is processed while being read from the designated processing folder. Complex batch processing jobs can be applied by nesting different folders.

are read, the effects are executed in real time. This allows building a library of non-destructively edited sounds: When the original file is edited, the copies in the effect file system change as well. In addition, VST audio effects can be used with any software. This approach can again be thought of as lazy batch processing.

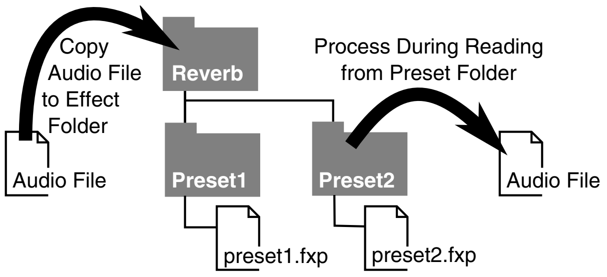

Copy Audio File to Effect Folder
Reverb
Process During Reading from Preset Folder
Audio File
Preset1
Preset2
Audio File
preset1.fxp
preset2.fxp

Fig. 4: The VST file system subjects audio files to effect plug-ins during reading.

The software prototype scans the system for available VST audio effect plug-ins and creates a distinct folder for every discovered effect. The prototype allows creating sub-folders within the effect folders for storage of the plug-ins' preset files. These are loaded if a file is copied to that distinct folder. If a file is copied to the effect folder, the plug-in's default settings will be used for processing.

The VST processing is based on the BASS Audio framework [33], in particular on the VST host contained therein. We developed a proprietary .NET wrapper for the BASS Audio framework. The virtual file system was again developed using the Dokan Library [13].

The VST file system works differently from the batch folder file system introduced above: A VST plug-in is only passed one sample block at a time, which prohibits processing tasks like normalization or fade-outs. In addition, the behavior of the VST file system is markedly different from software such as Automator [4] in Mac OS X, which also supports processing of audio files dragged into a specific folder. In Automator, however, the processed files are no longer connected to the originals; and the processing commences immediately and is not postponed until playback. Furthermore, Automator provides only limited audio file editing functionality such as adding fades, trimming, resampling, and basic audio level compression presets.

Fourth, we addressed the lack of platform-independent virtual folders. As mentioned in Section 3, a Search Folder on Windows Vista or a Smart Folder on Mac OS X will not be readable from another machine with a different operating system. Software manufacturers have undertaken some steps to liberate the operating system's search engine: For instance, SearchLight [16], after installation on the Mac file server, provides a Web interface which lets the user execute Spotlight searches from another machine, providing the query results in the browser window. However, the user must first download the found files to the local machine before he or she can use them in a further application.

Our prototype is based on MySQLFS [24], a FUSE implementation of a database file system, which stores data, inodes and their connections in three simple tables. Data are stored as binary large objects (BLOBs), each of 4 kByte size in one table; the second table, denoted "tree," holds the hierarchy of files and folders; the third table holds data usually to be found in a file's inode, such as the date of last access.

With relatively little effort, it was possible to modify the original MySQLFS source code on a Debian Linux machine in such a way that it resembled the functionality of a "Search Folder." We added a new field to the "tree" table, named "tag," see Table 1. Furthermore, on initialization two folders named "all" and "search" are created. The MySQLFS source code is modified in such a way that on creating a folder inside the "search" directory, a MySQL query is performed in the database, searching for project tags of files stored within the "all" directory matching the new folder's name. A new (virtual, since we are dealing with a FUSE system) inode is then created in the database for every matching file. The inode represents a (virtual) hard link to the file in the newly created folder with no physical copying of data, see Figure 5. The file system can be shared via a standard network protocol, for instance SMB [27]; every remote machine can execute queries by simply creating folders.

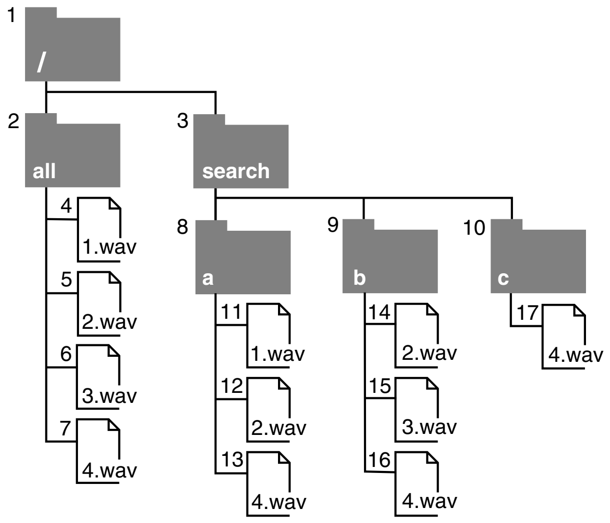


Fig. 5: An exemplary file system: Four audio files that belong to three different projects are stored in the database. The creation of new folders within the search folder triggers a database query to fill the folders with virtual copies of the data.

tree			
inode	parent	name	tag
1		/	
2	1	all	
3	1	search	
4	2	1.wav	a
5	2	2.wav	a, b
6	2	3.wav	b
7	2	4.wav	a, b, c
8	3	a	
9	3	b	
10	3	c	
11	8	1.wav	
12	8	2.wav	
13	8	4.wav	
14	9	2.wav	
15	9	3.wav	
16	9	4.wav	
17	10	4.wav	

Table 1: The table "tree" represents the file system displayed in Figure 5. Four files in the "all" directory are assigned by tags to three different projects. The project folders in the search directory mirror all files belonging to a specific project.

4.3. Lessons Learned

The RIFF file region file system can be put to good use whenever one needs to work with recordings of long durations. To fetch snippets from a long recording into digital audio workstation software, one does not need to load the complete file, search for the part of interest and crop the file to an appropriate length. Furthermore, the snippets of interest need not be stored individually in the file system, which would imply storing partial duplicates of recordings in the file system.

The batch folder file system allows a very quick and convenient access to basic audio processing tasks. The file system interface enables streamlining these functionalities, as the interface is common and well understood. Furthermore, the possibility of allowing VST effect plug-in processing in folders frees even more advanced audio processing tasks from designated audio plug-in hosts, rendering these effects more accessible.

The tweaked MySQLFS resembles the "stored queries" folder's functionality usually found in modern operating systems and provides its functionality in a heterogeneous environment, where for instance Mac OS X and Windows Vista machines are networked, each system providing digital audio workstation software that only runs within that specific operating system. Production environments like this are to be found in many studios, making it extremely difficult to deploy the search functionalities provided by the audio file server's operating system.

The SMB network protocol used to mount the modified MySQLFS on the networked machines is understood by every modern operating system and is widely used in heterogeneous environments. However, the prototype has the drawback of providing only one tenth of the data throughput of regular networked file systems. This is due to the data being stored as BLOBs. Future work will address the storage of the data in a regular file system, limiting the use of the database to search.

We plan to include versioning functionality to the file system, liberating the user from standard file backups. We also plan to employ methods of music information retrieval (MIR) [19] in search for audio assets. Furthermore, a file system presented via FUSE does not have any limits in where its data may stem from. For instance, an existing application of FUSE provides YouTube videos [35] as seemingly local data sources show that there is virtually no limit to what can be mapped to file systems. One could imagine a huge distributed database of audio files that are presented via a FUSE system, always at the appropriate sampling rate and bit depth, structured in folders according to which project they belong to. Non-destructive editing and versioning would also be taken care of by the file system.

All of this functionality could be available on every operating system. We had to learn, however, that the available FUSE implementations for Microsoft Windows systems are still far from a stable state. In contrast, the implementations for Linux and Mac OS X work reliably, which is also indicated by the number of existing applications.

In principle, FUSE systems even allow the system to change the content of a file that is currently open. When doing so, a FUSE system may ignore file locks and even keep the last access time unchanged. This could for instance be applied to let a second audio engineer enhance an audio file which is already in use in a mix; the edits are immediately reflected in the mix. This tends, however, to raise compatibility issues with application software. Apple Logic [6] and Digidesign ProTools [12] preload audio files into RAM and are not aware of changes done to an open file on the disk. In contrast, Sony Vegas [29] updates a changed file within a project if the file was changed within Sony Sound Forge [28], Sony's wave file editor. By locking the file, Vegas does not allow modification to a file in an open project from other applications such as Audacity [7]. With a FUSE system, this restriction can be cracked open.

5. CONCLUSION

In homogeneous production environments where only one operating system is deployed on all computers, the operating system's native file system provides the user with useful tools for workflow improvement. In a heterogeneous environment, where all networked machines work with a central data pool that has to be kept in a consistent state, the operation systems' file system capabilities swiftly come to an end.

Networked Content Management Systems for media assets [5, 8, 30] are widely available on the market today, but are costly, which limits their usage by smaller studios. Relatively inexpensive Storage Area Network (SAN) solutions provide at least a uniform data storage pool, which can be accessed from every networked machine in a heterogeneous production environment. SANs, however, only address part of the problems arising in asset management.

In order to keep a project's assets in a consistent state, files that belong to more than one project still have to be copied physically to a designated folder, resulting in multiple copies of the file, spread over the entire production network. In a heterogeneous production environment, this problem cannot be solved easily with alias files or shortcuts, as these will only be understood from the operating system they were created in. Genuine soft links would solve this problem, at the cost of having to leave every original file at its original location and with the same file name. Furthermore, genuine soft links cannot be generated by the operating systems' graphical user interfaces, which renders them less accessible to non-expert users.

Virtual folders can solve many problems much more elegantly than links can. In particular, they do not require that links are created, modified, and deleted. The operating systems' functions for virtual folders are, however, again limited to the corresponding operating system.

FUSE systems, as demonstrated in this paper, have the broadest ability to provide clean and cross-platform solutions to data management issues. In addition, they can be applied to uncommon but intuitive applications such as lazy batch processing, which can be put to good use in audio production environments. FUSE systems can also be applied to shield the non-expert user from file-system specific tasks such as backups, and provide the user with a familiar interface—a production environment that advances creativity.

6. ACKNOWLEDGMENTS

The authors thank Kristian Gohlke for his assistance with some of the prototypes.

7. REFERENCES

[1] Danilo Almeida. FIFS: a framework for implementing user-mode file systems in Windows NT. In *WINSYM'99: Proceedings of the 3rd conference on USENIX Windows NT Symposium*, pages 13–24, 1999.

[2] Apple. Alias Manager reference. `http://developer.apple.com/documentation/Carbon/Reference/Alias_Manager/Reference/reference.html`, accessed 2008-07-10.

[3] Apple. Working with Spotlight. `http://developer.apple.com/macosx/spotlight.html`, accessed 2008-07-10.

[4] Apple. Automator. `http://www.apple.com/de/macosx/features/300.html#automator`, accessed 2008-07-23.

[5] Apple. Final Cut Server. `http://www.apple.com/de/finalcutserver/`, accessed 2008-07-23.

[6] Apple. Logic. `http://www.apple.com/de/logicstudio/`, accessed 2008-07-23.

[7] Audacity. Free audio editing software. `http://audacity.sourceforge.net/`, accessed 2008-07-23.

[8] Avid. Interplay. `http://www.avid.de/de/products/media-asset-management.asp`, accessed 2008-07-23.

[9] BSD. LN(1): link, ln – make links. BSD General Commands Manual, 2006.

[10] David Collett. MP3FS. `http://mp3fs.sourceforge.net/`, accessed 2008-07-10.

[11] Copy-FS. A copy-on-write, versionned filesystem. `http://n0x.org/copyfs/`, accessed 2008-07-23.

[12] Digidesign. Pro Tools. `http://www.digidesign.com/`, accessed 2008-07-23.

[13] Dokan. User-mode file system for Windows. `http://dokan-dev.net/en/`, accessed 2008-07-10.

[14] ELDOS. Call back file system. `http://www.eldos.com/cbfs/`, accessed 2008-07-10.

[15] FUSE. Filesystem in userspace. `http://fuse.sourceforge.net/`, accessed 2008-07-10.

[16] Gravity Apps. SearchLight. `http://www.gravityapps.com/searchlight/overview/`, accessed 2008-07-23.

[17] John Heidemann. Stackable design of file systems. Technical Report UCLA CSD-950032, 1995.

[18] Tom J. Killian. Processes as files. In *Proceedings of USENIX Summer Conference*, pages 203–207, 1984.

[19] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.

[20] MacFUSE. A user-space file system implementation mechanism for Mac OS X. `http://code.google.com/p/macfuse/`, accessed 2008-07-10.

[21] Microsoft Developer Network. IShellLink::Resolve method. http://msdn.microsoft.com/en-us/library/bb774952(VS.85).aspx, accessed 2008-07-10.

[22] Microsoft Developer Network. Symbolic links. http://msdn.microsoft.com/en-us/library/aa365680.aspx, accessed 2008-07-10.

[23] MusicMeshFS. Virtual file system to organize music files using tags. http://code.google.com/p/musicmeshfs/, accessed 2008-07-23.

[24] MySQLFS. FUSE filesystem using MySQL as a storage. http://sourceforge.net/projects/mysqlfs/, accessed 2008-07-23.

[25] OpenSolaris Community. ZFS. http://opensolaris.org/os/community/zfs/, accessed 2008-07-23.

[26] Mark Russinovich. Windows sysinternals. http://technet.microsoft.com/de-de/sysinternals/bb896768.aspx, accessed 2008-07-10.

[27] Samba. Opening Windows to a wider world. http://us3.samba.org/samba/, accessed 2008-07-23.

[28] Sony. Sound Forge. http://www.sonycreativesoftware.com/soundforge, accessed 2008-07-23.

[29] Sony. Vegas Pro. http://www.sonycreativesoftware.com/vegaspro, accessed 2008-07-23.

[30] Studio Network Solutions. Postmap Operating Environment. http://www.studionetworksolutions.com/products/product_detail.php?pi=7, accessed 2008-07-23.

[31] TagsFS. Music library file system. https://gna.org/projects/tagsfs, accessed 2008-07-23.

[32] TwistedWave. TwistedFLAC. http://twistedwave.com/TwistedFLAC.html, accessed 2008-07-10.

[33] un4seen developments. BASS Audio Library. http://www.un4seen.com/, accessed 2008-07-23.

[34] Universal FUSE. Project homepage. http://ufuse.ikejisoft.com/, accessed 2008-07-10.

[35] Vishpat. YouTubeFS. http://code.google.com/p/youtubefs/, accessed 2008-07-23.

[36] WaybackFS. User-level versioning file system for Linux. http://wayback.sourceforge.net/, accessed 2008-07-23.

[37] WinFUSE. Filesystems with .NET. http://www.suchwerk.net/sodcms_FUSE_for_WINDOWS.htm, 2008-07-10.

[38] WinFUSE. FileSystem in Userland Windows port. http://code.google.com/p/winfuse/, accessed 2008-07-10.

[39] ZFS on FUSE. ZFS filesystem for FUSE/Linux. http://www.wizy.org/wiki/ZFS_on_FUSE, accessed 2008-07-23.