Demo: Universal File System Extended Attributes Namespace

François Revol *

Grenoble University - Laboratoire d'Informatique de Grenoble (LIG);
Laboratoire de Conception et d'Intégration des Systèmes (LCIS)
{firstname}.{sirname}@imag.fr

Abstract

The growing usage of file system extended attributes on many operating systems faces interoperability problems when trying to preserve them across multiple platforms. We propose a generic namespace design and mapping method to maintain an identical view of the global meta-data namespace by each operating system. Additionally, we try to address the API and semantic incompatibilities with a higher level framework.

1. The Problem

Filesystem extended attributes, often abreviated xattrs, or EAs, are a form of file meta-data consisting of name-value pairs, and are used in many operating systems, under many forms and names (Resource Fork, Named Streams), for many purposes, either for security concerns (Access Control Lists, Proof Carrying...), fallback for missing filesystem properties (DOS attributes, POSIX atime) or user-level applications [4], from early adopters like the BeOS, up to recent semantic desktops [5]. However various incarnations of EA concepts are usually incompatible with each other. Some have split namespaces for kernel-private data (ACL...), some have typed values[3], and the API aren't compatible.

The growing usage of incompatible extended attributes conflicts with the need for interoperability, further more in Open-Source operating systems like GNU/Linux which now includes implementations for many foreign filesystems like FAT, NTFS, SMB, BFS and HFS. Each such implementation either uses a naive approach for mapping foreign extended attributes leading to namespace pollution and name clashes [1], or a more complex but unilaterally-imposed (and thus not idempotent) mangling [7] when the file is moved across disks and systems, or sometimes just doesn't expose extended attributes at all due to lack of a clear mapping. A recent proposition for NFS extended attribute support on Linux already asserts incompatibility with the original IRIX implementation [6]. Some old filesystems do not support extended attributes, for which several incompatible backing-store schemes have been devised, some being patented [2].

While reading physical disks from different systems was unlikely in the past, growing usage of networked filesystems and virtualization platforms, using optimized shared folders like VMware, increases the issue since they are meant to be used from multiple operating systems. Other higher-level protocols, for example `rsync` and `subversion`, and archive formats like `tar` and `star` also try to support some form of extended attributes.

1.1 Example

A 0-byte file on a BFS partition seen from the Haiku operating system could carry the following extended attributes:

```
File:  /boot/home/people/François_Revol
Type    Size  Name             Value
'MIMS'    21  "BEOS:TYPE"    "application/x-person"
STRING    14  "META:email"     "revol@free.fr"
STRING     8  "IM:status"      "Offline"
STRING    23  "META:url"   "http://revolf.free.fr/"
RAW       20  "_trk/pinfo_le"  00 BA E3 EC A7 09...
```

After copying this file to an NTFS partition, rebooting to Windows to copy the file to a Samba share running on GNU/Linux, then rebooting to Haiku to read it back from the ext3 filesystem, the extended attributes might become:

```
File:  /unnamed_ext3/home/revol/François_Revol
Type    Size  Name             Value
RAW      264  "linux.user.DosStreams"
      05 00 00 00 00 00 00...   '................'
      00...-42 45 4f 53 5f...   '........BEOS_TYP'
      45 00 53 4d 49 4d 61...   'E.SMIMapplicatio'
```

2. Proposed Solution

Instead of having each vendor define its own reserved namespace prefix for others to implement, we define a common prefix that all vendors should recognize, and map each vendor's native namespace below it. The unified namespace

* Ph.D candidate

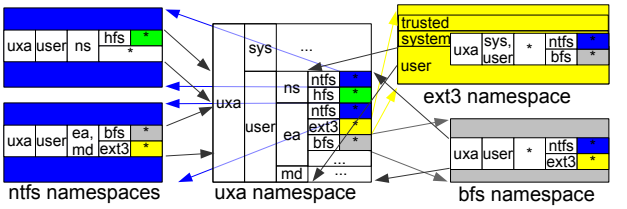

Figure 1. The tao of xattr namespaces

does not try to map extended attribute semantics between platforms, but instead focuses on ensuring correct preservation of the original form, and leaves the interpretation of foreign data to higher layers.

2.1 Namespace Hierarchy

We reserve part of each native EA namespace with an unlikely prefix `uxa.` for Unified eXtended Attributes, to map the global namespace and subdivide it further, while keeping the native names unchanged. Foreign EAs would then appear in a branch of the reserved namespace. When copied from a native filesystem to a foreign one, the mapping is reversed: the remaining part of the native namespace appears in the designated branch of the reserved namespace, and existing EAs copied to the branch corresponding to the foreign filesystem are extracted from the reserved namespace. A shortened unix shell pattern-like representation would be:

`uxa.{sys|user}.{ea|ns|md}.{bfs|ntfs|posix|sun|*}.*`

The proposed hierarchy for the reserved namespace is as follow, using reversed-DNS like notation, though shortened to minimize size and performance penalty on file copying, and shouldn't require special encoding. The root level defines the reserved namespace. The second level separates user-accessible EAs from kernel-only names, though the security implications would likely warrant not using it. The third level indicates if the EA originates from a named stream or real EA, since some platforms support both. Other forms of meta-data (`md`) are accounted for, to handle mapping POSIX `atime` for example. The last designated level names the platform the EA originates from, and indicates the corresponding mangling scheme to use. The next level maps native namespaces from vendors, and assumes UTF-8 encoding. Vendor-specific mappings should ensure preservation of the original name, possibly through percent-escaping as with Uniform Resource Locators. The unified namespace can be mapped partially multiple times, for example a filesystem supporting both EAs and named streams would map the `uxa.*.ns.*` and `uxa.*.{ea|md}.*` in the respective namespaces. Likewise for user-accessible and restricted namespaces.

2.2 Higher Level View

In order to better abstract extended attribute mechanisms, we take model of the traditional OSI network layering [8], and separate the transport and presentation layers. For performance reasons, we propose to split the implementation of the universal xattr namespace. The basic remapping scheme would be implemented in file system layers, typically foreign filesystem kernel modules, while native filesystems will likely not have to mangle the system's native API view of the xattrs. Then higher level views would be available through either a custom library exposing more semantics, or a reimplemented libxattr compatible with the Linux API for easier portability. Other possibilities could include a Java API extension, building on JSR 203.

3. Shortcomings

The mapping doesn't consider filesystems with limited storage capabilities, and assumes small enough attributes. A fallback strategy should probably use backing files like with filesystems missing extended attribute support. Some operating systems support extended attributes only on regular files, not symlinks or directories. ACLs aren't accounted for.

4. Conclusion

The purpose of this early work was to raise the concern about extended attributes interoperability, propose a possible solution, and foster discussion between involved parties, possibly leading to a standardized document like an IETF Request For Comment, with vendors defining the global namespace mapping to their own filesystems and protocols.

References

[1] Ntfs-3g extended attributes. `http://www.tuxera.com/community/ntfs-3g-advanced/extended-attributes/`.

[2] S. M. FRENCH[US/US], D. J. KLEIKAMP[US/US], and T. Y. T. TSO[US/US]. Method and apparatus for emulating alternate data streams across heterogeneous file systems, 03 2008. (IBM) (US patent 2008/0065698 A1).

[3] D. Giampaolo. *Practical file system design with the BE file system*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 1999.

[4] A. W. Leung, M. Shao, T. Bisson, S. Pasupathy, and E. L. Miller. High-performance metadata indexing and search in petascale data storage systems. *Journal of Physics: Conference Series*, 125:012069 (5pp), 2008.

[5] K. Möller and S. Handschuh. Towards a light-weight semantic desktop. In *Proceedings of the Semantic Desktop Design Workshop (SemDeskDesign 2007) at ESWC2007, Innsbruck, Austria*, Innsbruck, Austria, June 2007.

[6] J. Morris. Adding extended attribute support to nfs. `http://namei.org/presentations/linuxcon09_nfsv3xattrs.pdf`, 2009.

[7] A. Tridgell. Wine/samba. `http://www.samba.org/samba/ftp/slides/tridge_wineconf05.pdf`, 2005. (Wineconf).

[8] H. Zimmerman. Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, (28), 1980.