Filename encoding

and case-insensitive filesystems

Gabriel Krisman Bertazi <krisman@collabora.com>

Why an encoding-aware FS?

- Traditional UNIX-like approach: Opaque byte sequences.
- Because the other kids are doing it.
- Real world use cases:
  - Porting from the Windows world.
  - Android exposes case-insensitive tree.
  - Better support for exported filesystems.
- User space hacks are slow and racy.

Why an encoding-aware FS? (2)

- File uniqueness: Doesn’t translate well to real-world languages:

```
[krisman@dilma]$ ls -li toráx
15631222 -rw-r--r-- 1 root root 0 Nov 1 01:47 coração
15631218 -rw-r--r-- 1 root root 0 Nov 1 01:46 coração
15631217 -rw-r--r-- 1 root root 0 Nov 1 01:46 coração
15631215 -rw-r--r-- 1 root root 0 Nov 1 01:46 coração
```

- Lack of normalization is confusing for non-english speakers.
- What is the definition of Case without an encoding?

COLLABORA

Dealing with real world languages in the kernel

- NLS - Native Language Support

NLS Limitations

- Dealing with invalid character sequences.
- Dealing with multi-byte sequences/code points.
  - i.e.: to_upper, to_lower return a single byte char.
- Dealing with encoding evolution over time: stability.
  - i.e.: Unicode: Folding of Unmapped code-points isn’t stable.
- Missing Normalization and partially implemented casefolding.
  - Casefolding is almost ASCII only.

NLS improvements

- Encoding versioning
  - **load_nls_version** (“**utf-8**”, “**10.0.0**”, **flags**);
- Filesystem define NLS behavior:
  - Normalization Type (utf-8: NF(,K)(C,D))
  - Casefold Type (utf-8: CF; ascii: toupper, tolower)
  - Permissiveness mode: Validate, ignore or reject invalid sequences

NLS improvements (2)

- Support Multi-byte code-points
  - New API for comparisons, normalization and casefold
- Support for UTF-8 NFKD normalization and Full-Casefold, based on a decoding trie. Extendable for other Normalization types.
- Backward compatible with existing NLS tables and their users.

New NLS operations important to Filesystems

- nls_load_version()
- nls_validate() - string valid within the charset?
- nls_strncmp() - Consider equivalent sequences
- nls_strncasecmp() -
- nls_normalize() - Get the normalization of the string
- nls_casefold() - Get the casefold of the string

COLLABORA

Making Ext4 encoding-aware (and case-unaware)

- Patches for the kernel, e2fsprogs and xfstests are under review!
  - e2fsprogs
    - https://marc.info/?l=linux-ext4&m=153963794728040&w=2
  - linux:
    - https://www.spinics.net/lists/linux-ext4/msg62602.html

Encoding awareness

- Implemented as an Incompatible Feature.
- Stored on the superblock. Applies for the entire FS.
- Store the encoding type, version and flags.
- Configurable only at mkfs time for now.
  - Disk conversion could require rebuilding htree hashes.
- Name-preserving implementation.
- Implementation doesn’t touch VFS.

Dentry cache details

- Try to make good use of the dentry cache
- Equivalent name sequences don’t create multiple dentries. One per file entry.
  - d_hash() and d_compare() become encoding-aware
- Negative dentries are not cached
  - Requires careful invalidation during later creation

Case-insensitive support

- Requires Encoding support. Otherwise, Casefold == ?
- Per-directory inode attribute
  - On empty directories. Avoid name collisions
  - Finer-grained control is more suitable for users
- Trivial to implement:
  - Can be seen as a special case of the encoding support

Limitations / Issues

- directory encryption not supported
  - lookup based on a hash of the name. Impossible to calculate the same hash starting from an equivalent sequence.
    - Proposal: Store the file using the Hash of the normalization of the name. Does it work?
  - fscrypt has its own ->d_ops
    - Make fscrypt aware of encodings

Limitations / Open Issues

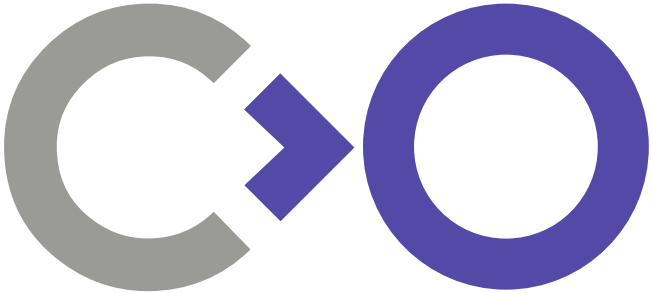
- Default setting (at mkfs) for dealing with invalid sequences
  - Proposal: Make it **permissive**: Consider broken sequences as opaque byte sequences (falls back to previous behavior)
- Userspace breakage due to normalization/casefolding of names?
- Any other?

Code:

- Linux:
  - https://gitlab.collabora.com/krisman/linux -b ext4-ci-directory_v3
- e2fsprogs
  - https://gitlab.collabora.com/krisman/e2fsprogs -b encoding-feature-merge
- xfstests
  - https://gitlab.collabora.com/krisman/xfstests -b encoding
- Patches on linux-ext4.

Acknowledgments:

- Thank you for the reference code, guidance, and/or code reviews
  - Olaf Weber, Ben Myers, Ted Ts’o, Darrick Wong, folks at LSF/MM, Collabora, and those guys at Valve!:)



Thank you!