Linux client update

European AFS & Kerberos Conference
2014

Marc Dionne
Your File System Inc.

From last EAKC

- Current kernel was 3.6
- 3.7 was in in pre-release phase
- putname() was unexported during the conference
- Reported that from 2.6.19 – 3.6, every single release required changes in the OpenAFS source

Fast forward – March 2014

- Current mainline is 3.14-rc8
- 3.14 imminent - expected this coming weekend
- No known issues with Linux mainline and current master and 1.6 branches

Since then

- 3.7
  - key_instantiate API change for key preparsing
  - param.h, ioctl.h: files moved, usage removed
    - Partially replaced later – useful in some cases
  - putname() unexported
    - More detail later

Since then...

- 3.8
  - vmtruncate removal
    - The function had been deprecated for a while in the kernel, now removed
  - session_keyring relocation
    - Moved from a sub structure in the credentials structure up to the main structure. Used for PAGs.
- 3.9
  - hlist API change
    - The API was reworked to not require the caller to provide a temporary element pointer

Since then...

- 3.10
  - sched.h no longer includes aio.h
    - Implicitely relied on it – include it explicitely
  - proc_create_entry replaced by proc_create
- 3.11
  - readdir API change
    - File operation “readdir” replaced by “iterate”
    - Parameters changed, return convention changed (1 = success)

Recent kernels

- 3.12
- 3.13
- 3.14 (not yet final)
- That's right – no changes required!
  - 3.12 introduced a new harmless warning, since fixed
  - More stability in upstream kernel APIs
  - Wider use of helpers and macros to hide underlying changes
  - Hopefully a trend that will continue

Other fixes

- Not unusual for compatibility fixes to require secondary fixes
  - Hard to test every case
  - Adjustments for less widely used versions
  - Side effects can require later fixes

User namespaces

- Kernel feature for containers
- uid/gid can have a different internal (kernel) and external (user space) value
- Need to add conversions – different types used
- Not a new feature, but only recently enabled in a major distribution (Ubuntu)
- Support patches from Anders Kaseorg
- Some improved PAG handling may be interesting in the future

putname/getname

- putname unexported in 3.7
  - but getname still exported – looked like an unintentional effect of patch re-ordering
- First solution was to roughly open-code what putname does, but still use getname
- Logic added to getname/putname to support syscall auditing
  - Names used by the syscall may be needed for audit entries created at syscall exit
  - getname keeps track of names used
  - putname becomes a no-op – the real putnames happen at the end of the syscall

putname/getname

- Problem
  - Our putname was not a noop – freed the name
  - .. causing a double free at end of syscall
  - .. and leading to corruption of the slab cache
  - Only occurred if auditing was active
- Resolution
  - Patch proposed by RedHat, adapted for compile issues and our coding style
  - Provides simplified equivalent of getname/putname without using the kernel versions
- Followup
  - getname unexported in mainline (3.13) to prevent similar cases from occurring

getcwd()

- Issue
  - Users reported “directory does not exist” errors
    - But directory and contents are accessible and usable
  - Caused by getcwd() returning ENOENT
    - But some other APIs succeed (get_current_dir_name)
- Cause
  - In the kernel, each task keeps a direct pointer to the dentry of the current working directory
  - The task holds a reference to this dentry, to keep it from going away
  - getcwd() just follows the dentry and determines its full path
    - .. but returns ENOENT if the dentry is unhashed
  - Some transient situations can cause afs_lookup to return an error, and our dentry revalidation function unconditionally dropped (unhashed) the dentry.

getcwd()

- Fixes
  - Initial fix: use d_invalidate in that situation, which doesn't drop (unhash) the dentry if it has other references such as a working directory
    - Cures the symptom, but has odd behaviour for “fs rm”
  - Better fix: under review, does drop the dentry if we know it's been deleted (lookup returns ENOENT)
    - Restores “fs rm” to original behaviour
  - We could also return an error from revalidate, but would have to make sure that doesn't introduce unwanted side effects

dentry aliases

- Don't intend to go into too much detail, just to give context
- Underlying issue
  - In the kernel, a dentry represents a path to an object (file, directory, symlink, etc.)
  - The Linux VFS expects dentry aliases (multiple dentries) for files, needed for hard link support
  - But it assumes that directories have a single dentry and a single path within a mounted FS
  - Not actually checked in many places, but some checks have been added where aliases were known to cause problems (ex: rmdir)
  - OpenAFS currently presents the entire AFS namespace to Linux as a single mount
  - .. and because of AFS mountpoints, an AFS directory can have multiple paths and would normally require multiple dentries

dentry aliases

- Effects
  - Minor
    - In most cases there are no visible effects for the user
    - But there can be odd behaviour when multiple paths are used simultaneously
  - More serious
    - OpenAFS may create a situation that violates checks in the kernel and result in a panic
- Checks and logic in the kernel evolve over time
    - Always a risk that new patches will trip over aliases
    - Vendors backport selectively, sometimes with changes
    - Can't guarantee that any workaround will keep working in the long run, everywhere

dentry aliases

- Some workarounds in the past – ex: d_automount patches from Andrew Deason
- Probably more short-term workarounds in store for upcoming dentry work in mainline
- Looking to the future
  - Always at risk as long as we have a need to violate a basic VFS assumption
  - More fundamental changes are desirable for the long term
  - Some options proposed, and some ongoing experimental work
  - To be continued..

Thank you