Case Insensitive Trees

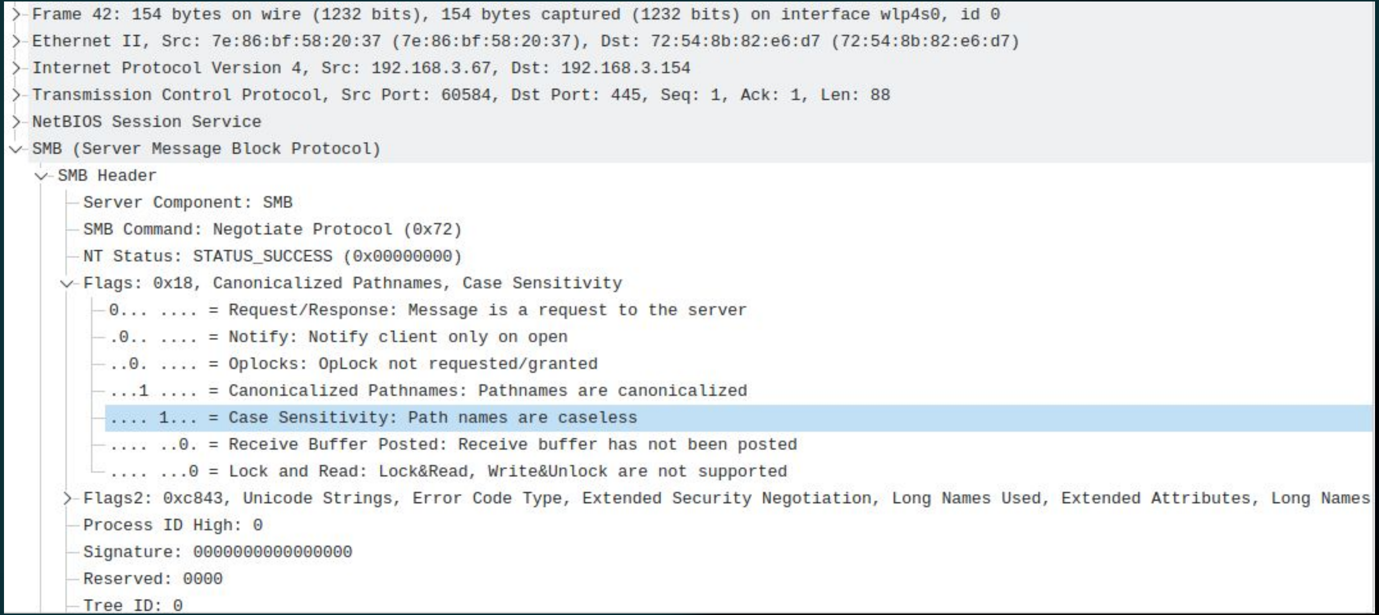
IN CEPHFS

Patrick Donnelly and Günther Deschner
IBM, Inc.

Case sensitivity at SMB protocol level

SMB_FLAGS_CASE_INSENSITIVE in SMB1

- Used during protocol dialect negotiation (NegProt) with deprecated SMB1



```
> Frame 42: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface wlp4s0, id 0
> Ethernet II, Src: 7e:86:bf:58:20:37 (7e:86:bf:58:20:37), Dst: 72:54:8b:82:e6:d7 (72:54:8b:82:e6:d7)
> Internet Protocol Version 4, Src: 192.168.3.67, Dst: 192.168.3.154
> Transmission Control Protocol, Src Port: 60584, Dst Port: 445, Seq: 1, Ack: 1, Len: 88
> NetBIOS Session Service
v SMB (Server Message Block Protocol)
  v SMB Header
      Server Component: SMB
      SMB Command: Negotiate Protocol (0x72)
      NT Status: STATUS_SUCCESS (0x00000000)
    v Flags: 0x18, Canonicalized Pathnames, Case Sensitivity
        0... .... = Request/Response: Message is a request to the server
        .0.. .... = Notify: Notify client only on open
        ..0. .... = Oplocks: OpLock not requested/granted
        ...1 .... = Canonicalized Pathnames: Pathnames are canonicalized
        .... 1... = Case Sensitivity: Path names are caseless
        .... ..0. = Receive Buffer Posted: Receive buffer has not been posted
        .... ...0 = Lock and Read: Lock&Read, Write&Unlock are not supported
    > Flags2: 0xc843, Unicode Strings, Error Code Type, Extended Security Negotiation, Long Names Used, Extended Attributes, Long Names
      Process ID High: 0
      Signature: 0000000000000000
      Reserved: 0000
      Tree ID: 0
```

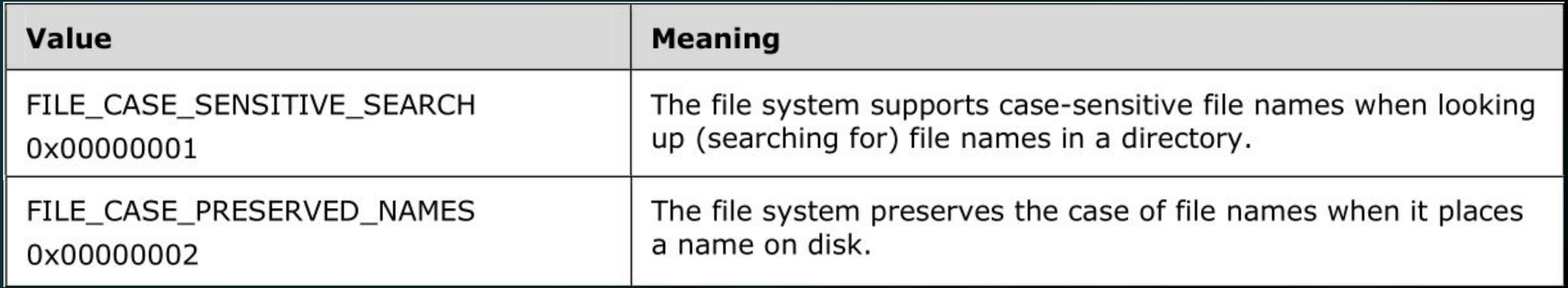
SMB_FLAGS_CASE_INSENSITIVE in SMB2

- SMB protocol specifications ([MS-CIFS], [MS-SMB2]) barely mention this bit:
- Footnote:

“This bit is ignored by Windows systems, which always handle pathnames as Case-insensitive”

FileSystemAttributes

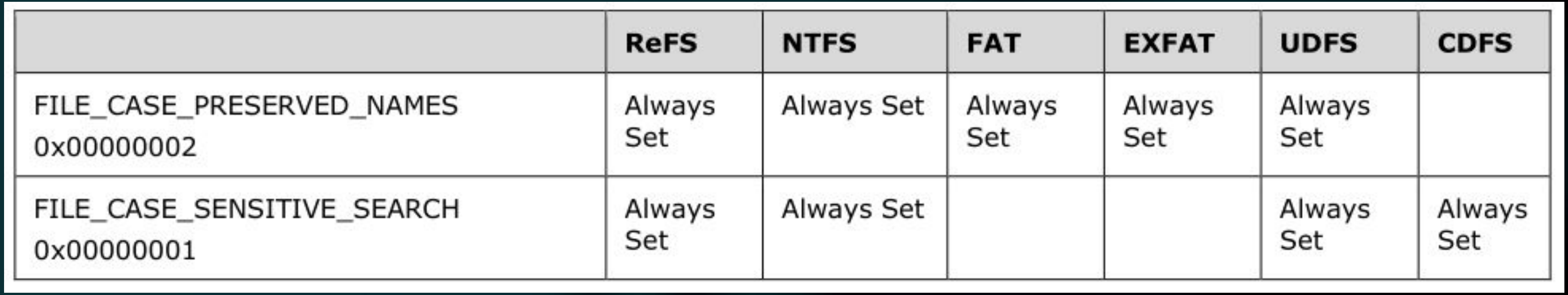
- File system attributes according to [MS-FSCC]:



Value	Meaning
FILE_CASE_SENSITIVE_SEARCH 0x00000001	The file system supports case-sensitive file names when looking up (searching for) file names in a directory.
FILE_CASE_PRESERVED_NAMES 0x00000002	The file system preserves the case of file names when it places a name on disk.

FileSystemAttributes

- Property of Windows filesystems, according to [MS-FSA]:



	ReFS	NTFS	FAT	EXFAT	UDFS	CDFS
FILE_CASE_PRESERVED_NAMES 0x00000002	Always Set	Always Set	Always Set	Always Set	Always Set	
FILE_CASE_SENSITIVE_SEARCH 0x00000001	Always Set	Always Set			Always Set	Always Set

Case sensitivity at SMB protocol level

- => No protocol negotiable way to avoid case sensitivity compatibility issues

Case sensitivity in Samba

Smb.conf options

- Configuration options:
  - Case sensitive = yes|no|***auto***
  - Default case = upper/lower
  - Preserve case = ***yes***|no
- By default Samba is case sensitive and case preserving
- Special case settings provided for large directories:
  - when “case sensitive = yes”, “preserve case = no” then “default case” is applied to all incoming requests
  - Avoids costly casefolding by assuming either all upper or all lowercase filenames

Samba VFS and case sensitivity

- vfs_get_real_filename()

  Allows VFS module to provide returning the correctly case folded filename, e.g. supported by vfs_gluster, vfs_gpfs, etc.

- vfs_fs_capabilities()

  Returns bitmask informing the caller about FS supported case settings, defined in [MS-FSCC]:
  - FILE_CASE_SENSITIVE_SEARCH 0x00000001
  - FILE_CASE_PRESERVED_NAMES 0x00000002

Samba's special cases

- Posix pathnames
  - Kernel SMB client
  - smbclient tool
- SMB(2) Unix Extensions
  - SMB3 POSIX Extensions Specification - The Samba Team
  - Check Volker's talk earlier today: FOSDEM 2025 - SMB3.11 Unix Extensions current status

Samba codepath examples (greatly simplified)

File open without case sensitive FS:

- SMB2_CREATE “FileName”
- stat(“FileName”)
- “FileName” exists:
  - open(“FileName”)
- “FileName” does not exist:
  - OpenDir(parent directory)
  - Compare all entries for a match
  - Found a match:
    - open(“FileName”)

File open with case sensitive FS:

- SMB2_CREATE “FileName”
- stat(“FileName”)
- “FileName” exists:
  - open(“FileName”)

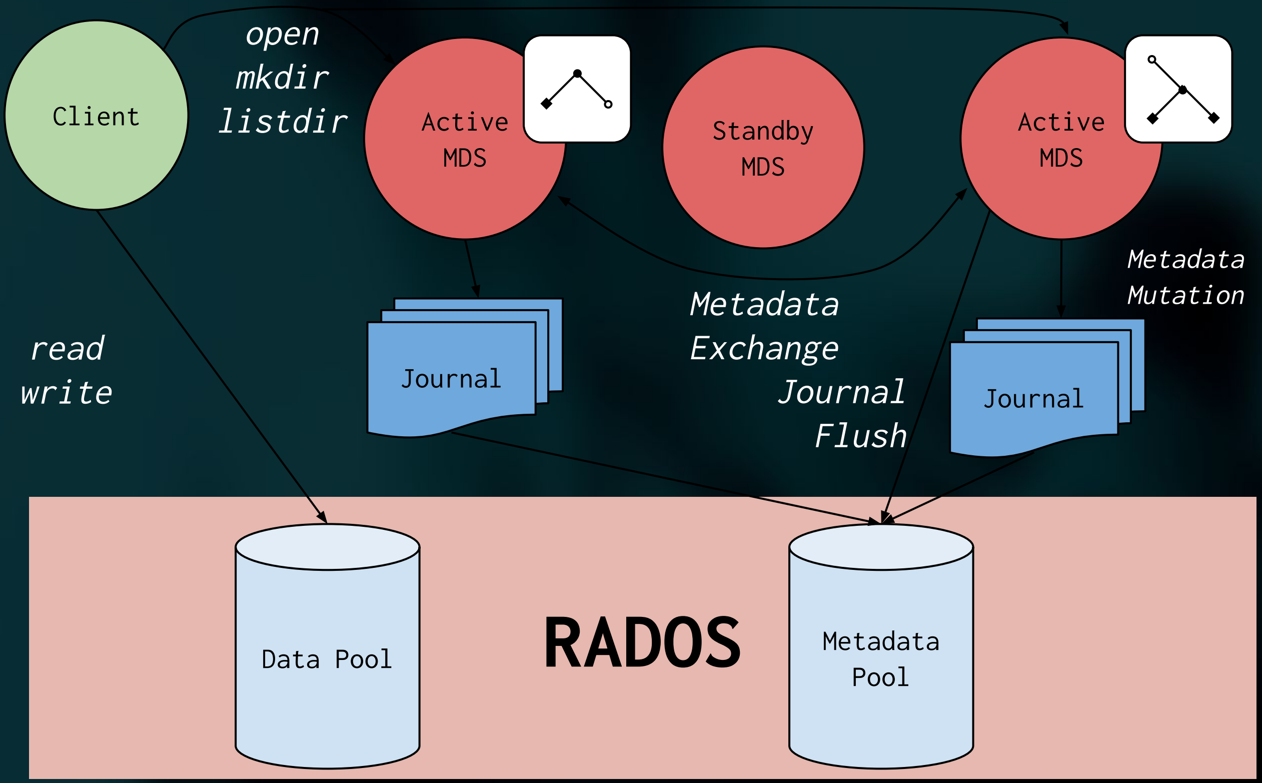
Some experiments with Linux Kernel sources

- Full untar of Linux kernel sources via libcephfs
- 149,342 readdir() operations
- 30.3% of time spent with full directory scans required to complete the operation
- Eliminating case lookups via configuration
    - Case sensitive = yes
    - Default case = lower
    - Preserve case = no
- Execution time went down by almost a third!

Case sensitivity in CephFS

CephFS

- CephFS is a POSIX distributed file system.
- Clients and MDS cooperatively maintain a distributed cache of metadata including inodes and directories
- MDS hands out capabilities (aka caps) to clients, to allow them delegated access to parts of inode metadata
- Clients directly perform file I/O on RADOS



Tagged metadata for directory entries

- Directory entries contain little metadata:
  - Inode #
  - Inode type
  - Dirent name
- A new metadata is attached to directory entries to support encryption, **alternate_name**.
  - MDS does not consider this metadata for any server side RPC; it's simply associating an opaque blob with the dentry.

```
struct dirent {
  ino_t           d_ino;       /* Inode number */
  off_t           d_off;       /* Not an offset; see below */
  unsigned short d_reclen;    /* Length of this record */
  unsigned char  d_type;      /* Type of file; not supported
                                  by all filesystem types */
  char            d_name[256]; /* Null-terminated filename */
 vector<uint8_t> d_alternate_name;
};
```

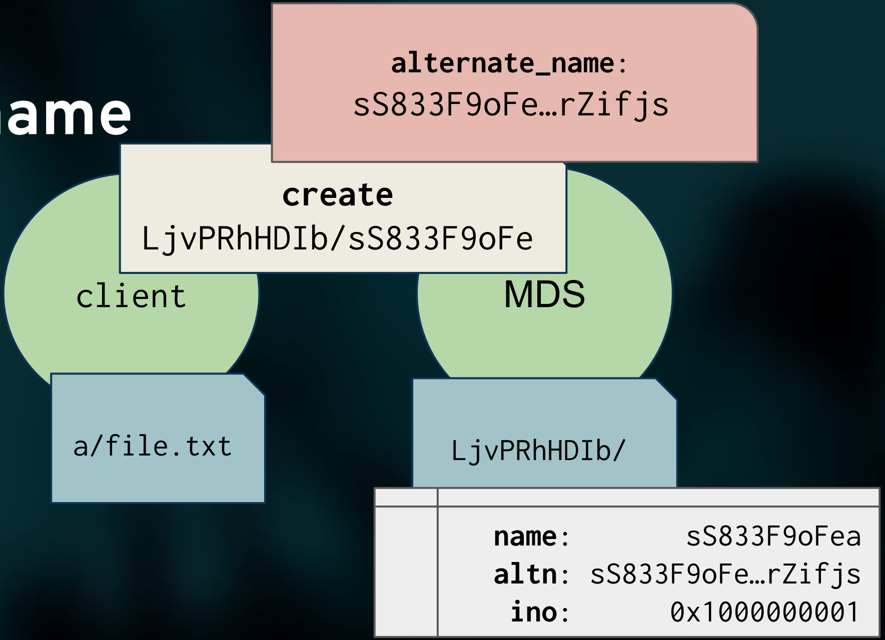
Initial use-case for alternate_name

- Client-side encryption of directory tree requires encoding encrypted names as regular file names sent to the MDS.

```
name: fscrypt(name,key)
```

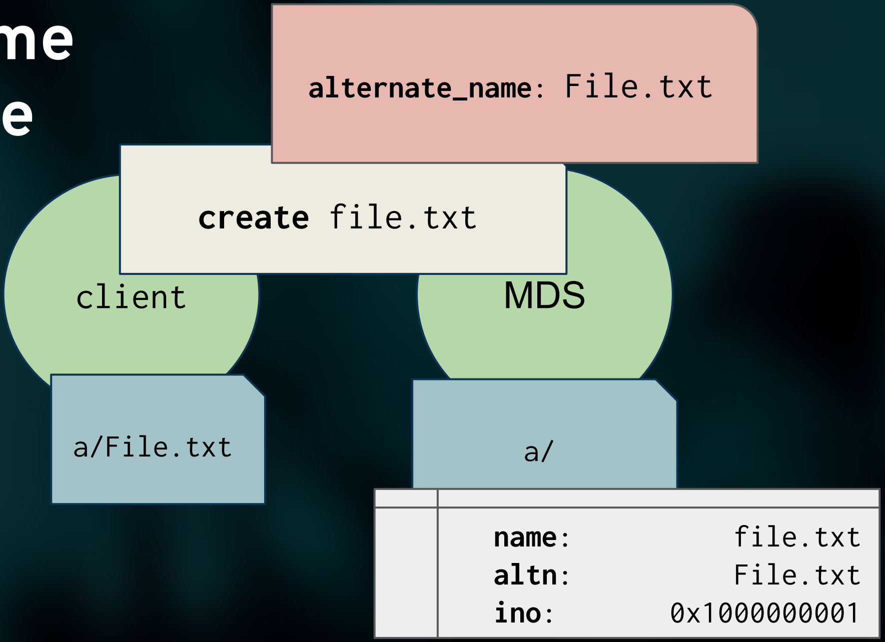
- **Problem:** A dirent name may be up to NAME_MAX (256) chars. Encryption+encoding may cause large names to exceed the max.

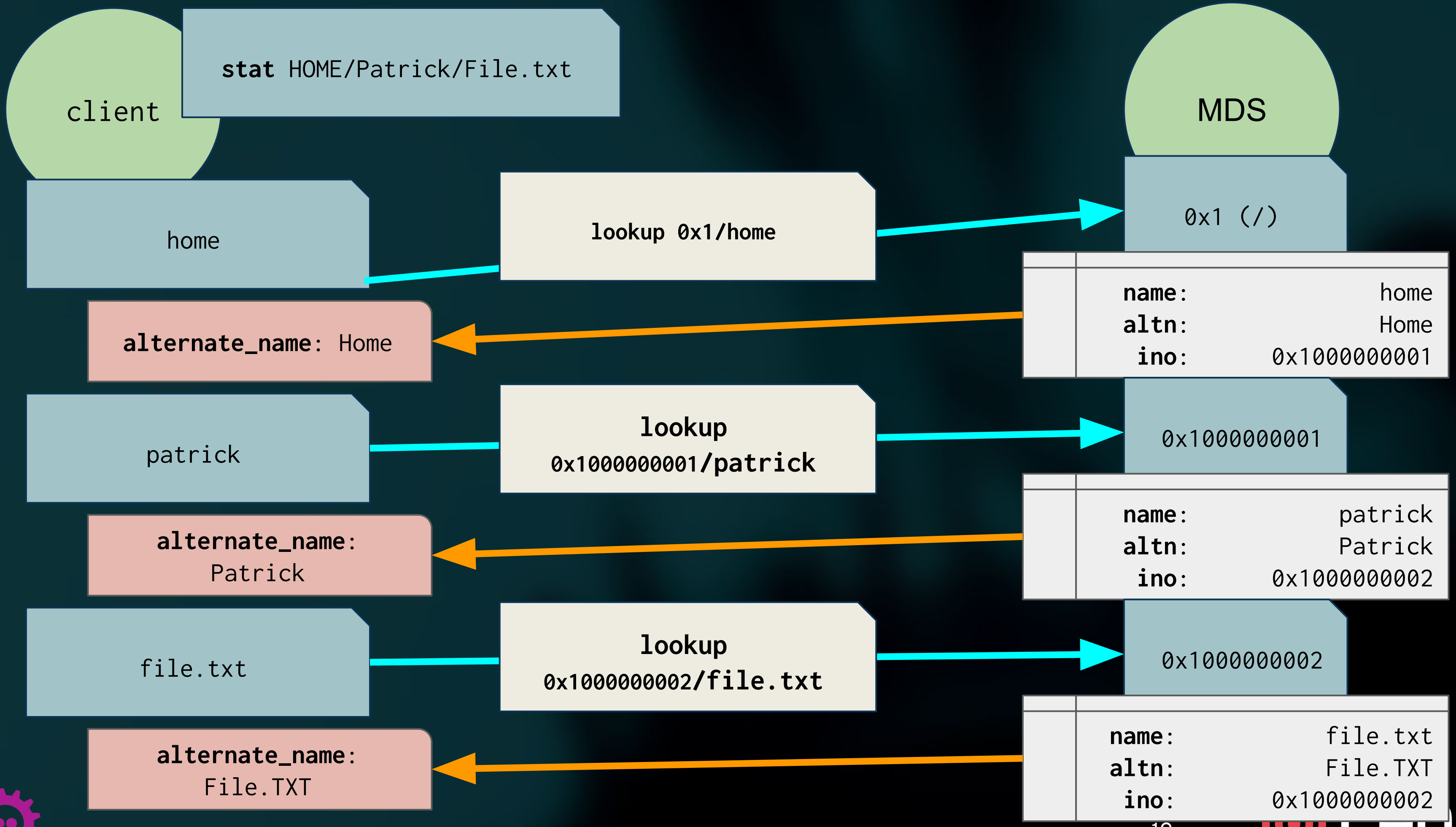
```
name: fscrypt(name,key)[:157]
      .. sha256(fscrypt(name,key)[158:])
alternate_name: fscrypt(name,key)
```

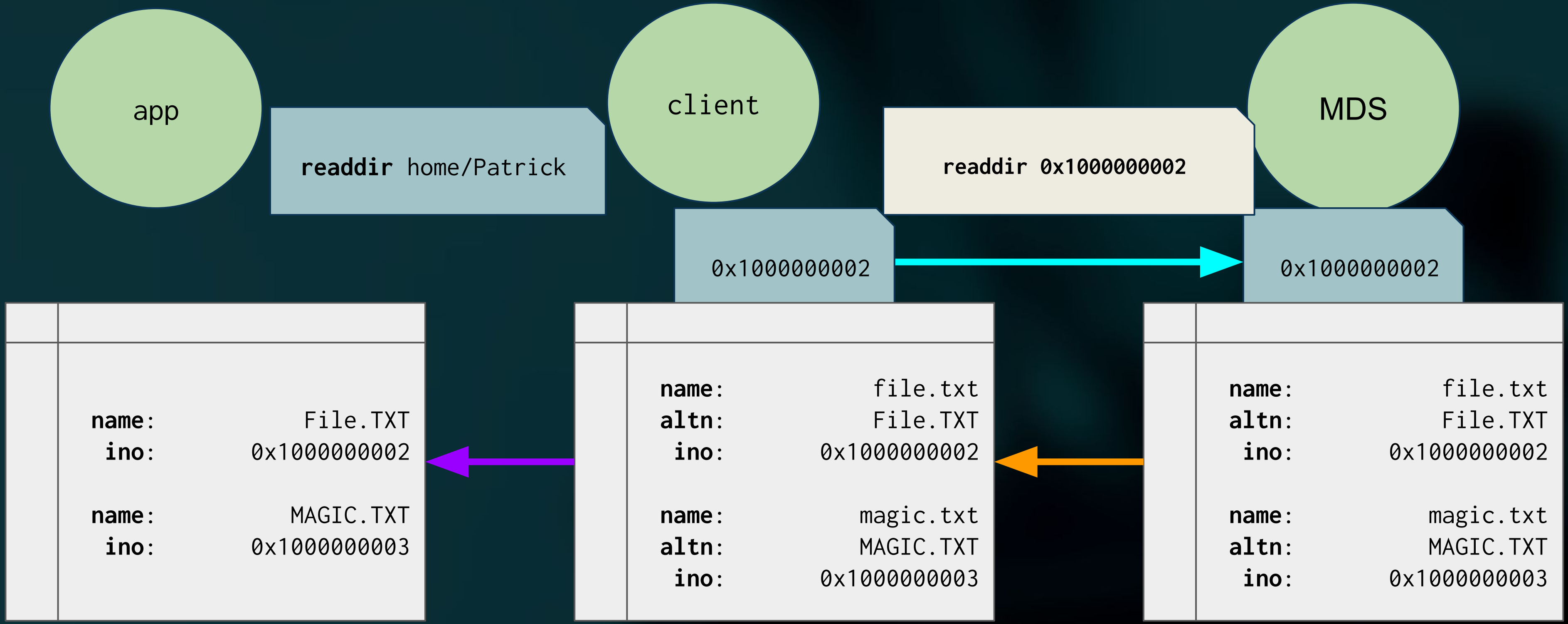


Observation: use alternate_name for preserving the caseful name

- Clients must agree on the transformation of all directory entry names.
- The **alternate_name** metadata can be used to reconstruct the application's actual **caseful** name.
- **Highlights**
  - MDS doesn't need to care about the case transformations of the name.
  - Client generally doesn't care either – it only needs to **unwrap** the directory entry name when presented to the application (i.e. via `readdir`).




client
stat HOME/Patrick/File.txt
MDS
home
lookup 0x1/home
0x1 (/)
name: home
altn: Home
ino: 0x1000000001
alternate_name: Home
patrick
lookup 0x1000000001/patrick
0x1000000001
name: patrick
altn: Patrick
ino: 0x1000000002
alternate_name: Patrick
file.txt
lookup 0x1000000002/file.txt
0x1000000002
name: file.txt
altn: File.TXT
ino: 0x1000000002
alternate_name: File.TXT



```
name:        File.TXT
 ino:    0x1000000002

name:       MAGIC.TXT
 ino:    0x1000000003
```

```
name:        file.txt
altn:        File.TXT
 ino:    0x1000000002

name:       magic.txt
altn:       MAGIC.TXT
 ino:    0x1000000003
```

```
name:        file.txt
altn:        File.TXT
 ino:    0x1000000002

name:       magic.txt
altn:       MAGIC.TXT
 ino:    0x1000000003
```

Charmap API

New virtual xattrs

- ceph.dir.casesensitive
- ceph.dir.normalization
- ceph.dir.encoding
- **ceph.dir.charmap** – read-only view

Requirements to modify:

- Directory is empty
- Directory is not part of a snapshot

```
$ getfattr --only-values -n ceph.dir.charmap . | jq .
{
  "casesensitive": false,
  "normalization": "nfd",
  "encoding": "utf8"
}
```

ceph.dir.normalization

- **nfd**: Form D (Canonical Decomposition)
- **nfc**: Form C (Canonical Decomposition, followed by Canonical Composition)
- **nfkd**: Form KD (Compatibility Decomposition)
- **nfkc**: Form KC (Compatibility Decomposition, followed by Canonical Composition)

The default normalization for a character mapping configuration is `nfd`.

Normalization is not optional!

```
$ setfattr -n ceph.dir.normalization -v nfd .
```

```
Grüßen     -> Gru U+0308 U+00df en
ế (U+1EBF) -> e U+0065 + ◌̂ U+0302 + ◌́ U+0301
```

ceph.dir.casesensitive

- Unicode case folding:
  - https://www.unicode.org/Public/UNIDATA/CaseFolding.txt
  - Locale-independent
  - Normalization is required.

```
$ setfattr -n ceph.dir.casesensitive -v 0 .
```

```
Grüßen      -> gru U+0308 U+00df en
ế (U+1EBF) -> e U+0065 + ◌̂ U+0302 + ◌́  U+0301
```

ceph.dir.encoding

- UTF-8 is the only supported encoding
- We have the option to support other encodings in the future.
    - It’s difficult to support other encodings without correcting NUL-terminated string assumptions for path names.

```
$ setfattr -n ceph.dir.encoding -v utf8 .
```

Subvolume API

```
$ ceph fs subvolumegroup charmap get <volume> <subvolumegroup>
{"casesensitive":false,"normalization":"nfd","encoding":"utf8"}
$ ceph fs subvolume      charmap get <volume> <subvolume> <subvolumegroup>
{"casesensitive":false,"normalization":"nfd","encoding":"utf8"}

$ ceph fs subvolumegroup charmap set <volume> <subvolumegroup> casesensitive 0
# Must be an empty subvolumegroup!
$ ceph fs subvolume      charmap set <volume> <subvolume> <subvolumegroup> casesensitive 0
# Must be an empty subvolume!

$ ceph fs subvolumegroup charmap rm <volume> <subvolumegroup>
# Must be an empty subvolumegroup!
$ ceph fs subvolume      charmap rm <volume> <subvolume>  <subvolumegroup>
# Must be an empty subvolume!
```

Client access guards

New CephFS client feature bit **charmap**.

- **charmap** feature bit prevents older/incompatible clients from creating new directory entries without appropriate name transformations
  - Unlink is okay
  - Rmdir is okay
- You can set the **required_client_features** to prevent incompatible clients from mounting but this **is not recommended.**
  - MDS already protects the directory trees.
  - Setting the bit would prevent kernel clients from accessing the file system.

```
$ ~~ceph fs required_client_features cephfs add charmap~~
added feature 'charmap' to required_client_features
$ ceph fs dump
Filesystem 'cephfs' (1)
…
 required_client_features     {22=charmap}

$ ceph-fuse /cephfs # old version
… client.4563 mds.0 rejected us (missing required features
'0x0000000000400000')
$ ceph fs required_client_features cephfs rm charmap
$ ceph-fuse /cephfs # old version
$ cd /cephfs/dir && getfattr -n ceph.dir.charmap .
ceph.dir.charmap="{\"casesensitive\":true,\"normalization\
":\"nfd\",\"encoding\":\"utf8\"}"
$ ls
Grüßen
$ dd if=/dev/urandom of=File bs=1 count=1
dd: failed to open 'File': Operation not permitted
$ rm Grüßen
$
```

Example

```
$ cd /cephfs/dir
$ ls
$ setfattr -n ceph.dir.casesensitive -v 0 .
$ getfattr --only-values -n ceph.dir.charmap .
{"casesensitive":false,"normalization":"nfd","encoding":"utf8"}
$ touch Grüßen
$ ls .
Grüßen

$ ceph tell mds.a:0 dump tree /dir/ 0 > tree.json

$ < tree.json jq -r '.[0].dirfrags[0].dentries |
                     map(select(.path | test(".*ssen")) |
                         .path) |
                    .[0]'
dir/grüssen
$ < tree.json jq -r '.[0].dirfrags[0].dentries |
                     map(select(.path | test(".*ssen")) |
                         .alternate_name) |
                     .[0]' \
              | base64 -d
Grüßen
```

Closing thoughts

- An extra metadata **`alternate_name`** attached to directory entries has proven to be a useful abstraction beyond its original use-case.
- Samba+CephFS will now enjoy safe native performance for directories enabled with a case-insensitive charmap.

THANK YOU and Q/A

Patrick Donnelly
IBM, Inc.
Ceph engineer
pdonnell@ibm.com

Günther Deschner
IBM, Inc.
Ceph-SMB team manager
gdeschne@redhat.com
gd@samba.org

Pull requests:

- Case-insensitive directory trees:
  - Design document: https://tracker.ceph.com/issues/66373
  - https://github.com/ceph/ceph/pull/60746

Documentation:

- charmap vxattr