NetApp FUSE

Carter Levinson • Danny Yu • Qizhe Wang • Sam Lasky

A quick word from our sponsors...

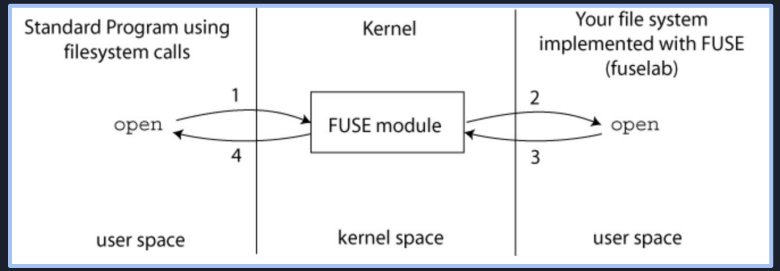
- **What is Net App?**
  - *"Net App is the Data Authority in the Hybrid Cloud"*
  - Works with the design, manufacture, marketing, and technical support of storage and data management solutions
- **Layman's Terms**
  - Think of Net App as a storage company
  - Clients range from Google, Amazon, IBM to name a few
  - Client wants their data stored, doesn't care how it's stored (directly on hardware vs the cloud)
    - NetAPP handles this
  - Provides a service which allows clients to access, store, and handle their data regardless of the storage medium utilized

What is FUSE?

- FUSE: **F**ilesystem in **Use**rspace
- For UNIX or UNIX-like operating systems
  - You can also make it work on Windows systems using wrappers or FUSE drivers specifically implemented for Windows OS
- Allows easier and faster customization of file systems
  - Creation and modification of a file system at the user-level without kernel modification
- Open Source - it's FREE!
  - github.com/libfuse/libfuse
- Written in the C programming language
  - "Low level" language for systems software
- Think back to CS1550 project 4

How FUSE works

- Mount/unmount the virtual file system
  - In Unix/Linux, all files exist under the root directory “/” can attach arbitrary file systems to anywhere in the tree
  - When you mount a file system, you attach the file system to that directory (or the **mount point**)
  - Acts as the root directory of the new filesystem
- A handler program intercepts the request ops
  - Read/write/stat, etc.
- Requests from userspace is redirected from the kernel VFS to FUSE
- FUSE executes the registered handler program and return the response though the kernel VFS back to the userspace
- No “sudo” required



Why use FUSE?

- It’s really useful!
- No need to change the file system application and kernel code
  - Avoids mistakes when editing kernel driver code
  - Less lines of code to edit/write
  - Less work for us woo-hoo!
- Speeds up process of developing new/interesting filesystems
- Adding new syscalls to accommodate user needs
- Portable between different systems

Project Goals

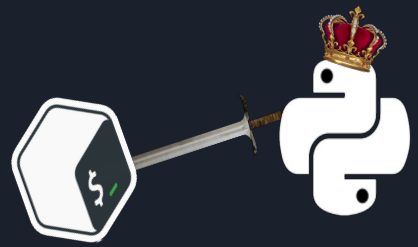
- Implement a handler that considers user storage usage
- Limited storage quota per user
  - e.g. each user can only allocate 5GB for free
  - Additional storage will require charging the user
- Ability to handle multiple read/writes in the system
  - Consider database managing issues
  - e.g. race conditions, read-write conflicts, etc.
- Think about it like the AFS system on Pitt's Linux boxes

The Progress - Data Storage

- Use SQLite as a file based database system
  - Fast!
  - Can make use of well known SQL queries and commands
  - Flexible C language API
- Use Autotools as the build system
  - More portable than Makefiles
  - Allows software to turned into a package and easily installed across *nix systems

The Progress - Testing Infrastructure

- Python v.3.8.10
  - Came down to bash vs python for script implementation
  - Python's simplicity and extensive library (modules) were the deciding factors
    - Easier to interface with the SQLite database with Python's SQLite module(s)
    - Subprocess module is the basis for our testing framework
      - Spawn a new process, process interacts with FUSE via terminal, output stored in a string
      - Resulting string compared to expected output



The Progress - Testing Infrastructure cont.

- Process parameters include **FUSE filesys op(s) followed** by said op's parameters
- It's is simple as that

```
import subprocess
commands = '''
ntapfuse mount base_dir/ mountpath/
echo this is a test! > base_dir/hello
'''
process = subprocess.run(commands,shell = True)
print(process);
```

```
int ntapfuse_getattr (const char *path, struct stat *buf);
int ntapfuse_readlink (const char *path, char *target, size_t size);
int ntapfuse_mknod (const char *path, mode_t mode, dev_t dev);
int ntapfuse_mkdir (const char *path, mode_t mode);
int ntapfuse_unlink (const char *path);
int ntapfuse_rmdir (const char *path);
int ntapfuse_symlink (const char *path, const char *link);
int ntapfuse_rename (const char *src, const char *dst);
int ntapfuse_link (const char *src, const char *dst);
int ntapfuse_chmod (const char *path, mode_t mode);
int ntapfuse_chown (const char *path, uid_t uid, gid_t gid);
int ntapfuse_truncate (const char *path, off_t off);
int ntapfuse_utime (const char *path, struct utimbuf *buf);
int ntapfuse_open (const char *path, struct fuse_file_info *fi);
int ntapfuse_read (const char *path, char *buf, size_t size,
                   off_t off, struct fuse_file_info *fi);
int ntapfuse_write (const char *path, const char *buf, size_t size,
                    off_t off, struct fuse_file_info *fi);
int ntapfuse_statfs (const char *path, struct statvfs *buf);
int ntapfuse_release (const char *path, struct fuse_file_info *fi);
int ntapfuse_fsync (const char *path, int datasync,
                    struct fuse_file_info *fi);
int ntapfuse_setxattr (const char *path, const char *name,
                       const char *value, size_t size, int flags);
int ntapfuse_getxattr (const char *path, const char *name,
                       char *value, size_t size);
int ntapfuse_listxattr (const char *path, char *list, size_t size);
int ntapfuse_removexattr (const char *path, const char *name);
int ntapfuse_opendir (const char *path, struct fuse_file_info *fi);
int ntapfuse_readdir (const char *path, void *buf,
                      fuse_fill_dir_t fill, off_t off,
                      struct fuse_file_info *fi);
int ntapfuse_releasedir (const char *path, struct fuse_file_info *fi);
int ntapfuse_access (const char *path, int mode);
void *ntapfuse_init (struct fuse_conn_info *conn);
```

Future Goals

- Implement a FUSE that works for multiple users on the same filesystem without wrecking the database or the quotas.
- Select a coherent subset of linux file system operations to implement in the database
- Develop testing framework for file systems focused on consistency, persistence and reliability