Using FUSE to Create Your Own Filesystem

James Shewmaker

SCALE 7x

21 Feb 2009

Today's Agenda

- Super Brief Filesystem Basics
- FUSE
  - Basics
  - Examples
  - Future
  - Building Your Own Filesystem
    - Quick and Dirty
    - The Right Way

Filesystem Basics

- Layers
  - Filesystem
    - Block size, free blocklist, free inode list
  - Filename
    - Connects file's name to inode or block
  - Metadata (inodes)
    - Last modified/access/created/changed
  - Data
    - List of blocks that contain the file's content

FUSE Basics

- FUSE is a kernel module that provides userland access to filesystem features
  - Takes care of all of the system library calls for filesystem operations
  - Effectively allows for significant control of what “data” is and how to interact with it
  - Comes in library so you can call with C, Python, etc.

FUSE in practice

- GmailFS (libgmail, python-fuse)
- sshFS (C)
- Various others including http, ftp, etc.
- If you aren't careful, you create a filesystem that needs a filesystem to support it

Necessary FUSE Default Attributes

```
# Set Default Attributes
class MyStat(fuse.Stat):
      def __init__(self):
          self.st_mode = stat.S_IFDIR || 0755
          self.st_ino = 0
          self.st_dev = 0
          self.st_nlink = 2
          self.st_uid = 0
          self.st_gid = 0
          self.st_size = 4096
          self.st_atime = 0
          self.st_mtime = 0
          self.st_ctime = 0
```

FUSE Core Functions

- `def getattr(self, path):`
- `def readdir(self, path, offset):`
- `def mknod(self, path, mode, dev):`
- `def unlink(self, path):`
- `def read(self, path, size, offset):`
- `def write(self, path, buf, offset):`
- `def release(self, path, flags):`
- `def open(self, path, flags):`
- `def truncate(self, path, size):`
- `def utime(self, path, times):`
- `def mkdir(self, path, mode):`
- `def rmdir(self, path):`
- `def rename(self, pathfrom, pathto):`
- `def fsync(self, path, isfsyncfile):`

HelloFS

- Just to prove we can make anything a FS
- /usr/share/doc/python-fuse/examples/hello.py
- Only creates a “hello” file in root of FS
- Contents are 13 bytes: Hello world!
- Use this as a template and build in small steps, testing _every_ change

HelloFS source highlights

```
class HelloFS(Fuse):
    def getattr(self, path):
        st = MyStat()
        if path == '/':
            st.st_mode = stat.S_IFDIR | 0755
            st.st_nlink = 2
        elif path == hello_path:
            st.st_mode = stat.S_IFREG | 0444
            st.st_nlink = 1
            st.st_size = len(hello_str)
```

HelloFS source highlights (2)

```
    def readdir(self, path, offset):
        for r in  '.', '..', hello_path[1:]:
            yield fuse.Direntry(r)

    def open(self, path, flags):
        if path != hello_path:
            return -errno.ENOENT
        accmode = os.O_RDONLY | os.O_WRONLY | os.O_RDWR
        if (flags & accmode) != os.O_RDONLY:
            return -errno.EACCES
```

HelloFS source highlights (3)

```
    def read(self, path, size, offset):
        if path != hello_path:
            return -errno.ENOENT
        slen = len(hello_str)
        if offset < slen:
            if offset + size > slen:
                size = slen - offset
            buf = hello_str[offset:offset+size]
        else:
            buf = ''
        return buf
```

Making HelloFS writable

- Write the “write” function
  - Start with simple “What do I want to do?” questions
    - Overwrite a file
    - Append to a file
    - Write bytes at an offset in a file
  - Remember to stop and regression test each time or you will go postal

Making HelloFS writable (2)

- Don't forget the “close” function
  - This is where things trip up
  - FUSE has no “close” function!
    - “flush”
    - “release”
  - Critical to take the baby steps
    - Simple First
    - Grow Complexity
    - Don't forget to test what used to work

Other Ideas

- Wrote a simple replacement for /dev/tcp:
  - mknod opens up netcat
  - read and write tweaked to read/write bytes to STD IN/OUT
  - Only works for quick and dirty packet slinging
    - Should write in “safer” I/O
    - Better off to use python libs instead of I/O to netcat
  - It takes a lot of effort to reinvent netcat as a filesystem so I'm not finished yet (Marchish?)

Other Ideas (2)

- Favorite filesystem that lacks one feature?
- Create a filesystem that overloads that feature
- Pass regular activity on through
- Examples
  - Add your own transaction logging to FAT
  - Filesystem rootkit tricks (lie about reads)

Other Ideas that I Can't Show

- StegoFS
  - Created in August, 2008
  - Used Forward Error Correction (FEC) and Steganographic techniques encode bytes in video
  - Survives conversion/compression of YouTube
  - Used Imagemajick and ffmpeg
  - High Latency!
  - Unreleased because of patents on similar FEC

For More Info

- Read the examples in /usr/share/doc/examples/fuse
- http://apps.sourceforge.net/mediawiki/fuse/index.php?title=SimpleFilesystemHowto
- This and other presentations at http://bluenotch.com/resources/
- SCALE in Subject: jims@bluenotch.com