FUSE FileSystems
Roll your Own Filesystem

February 2017 OLUG

by Aaron Grothe

What is FUSE?

FUSE is a Filesystem in USEr space.

Quite simply it is a way for users to be able to create file systems without having to do kernel work.

Also don't need root to use FUSE. Just need to be in the fuse group on your system.

Some Examples of FUSE filesystems

- SSHFS - Filesystem over SSH - allows you to mount a remote filesystem via ssh
- GMAILFS - Filesystem that runs on top of Gmail
- ZFS on FUSE - Defunct. A good example of some of the capabilities of FUSE
- ROFS - Read Only File System
- MP3FS - Filesystem that does FLAC to mp3 conversion on the fly
- ZipFS - Filesystem that dynamically unzips files from a zip archive
- There are tons of other filesystems out there as well. If not you can write your own :-)

More Systems based on Fuse

- ExpanDrive - Commercial Software system based on FUSE - multiple back ends
- GlusterFS - Clustered Distributed Filesystem for Gluster which scales to petabytes of data
- EncFS - encrypting filesystem
- HDFS - Hadoop File System

Interfaces for FUSE

There are multiple interfaces available for FUSE

Ruby/Objective-C/Python/PERL/C(++)/Java/Go and others

Writing an interface isn't that difficult - there are at least three of them for Python: python-fuse, python-llfuse, pyfuse - (pyfuse has worked pretty well for me)

OS Availability

FUSE is available on Linux/BSD and Mac OS X. There is/was a Windows port, but it currently appears to be dormant. FUSE is also available on Windows under cygwin.

Most windows people nowadays are using a library named Dokan, which is similar to FUSE.

WinFSP - Windows File System Proxy is similar concept to FUSE, but is not compatible with FUSE.

My Filesystems

So I’ve got two Demo filesystems for tonight.

NewerFS - this is a simple read only filesystem. It allows you to easily find out when files change on a file system.

HashFS - this is another read only filesystem. It dynamically creates a hash file .md5 extension for each file on a filesystem.

NewerFS

NewerFS is written using the C interface to FUSE. The filesystem is 609 lines of code right now. It could be quite a bit shorter if I did some code cleanup.

So we'll do a quick demo and then take a quick look at the code.

HashFS

HashFS is written using the pyfuse Python/Fuse bindings.

It is currently 167 lines of python code, again there is room for improvement here.

So let’s do a quick demo and then we’ll take a look at the code.

Libraries

Either of these filesystems could be written using the other set of libraries.

There is nothing magical to either. Having the hash libs built into python does make the code a bit shorter, but it isn't anything we couldn't work around with librhash.

Stacking FUSE filesystems

One of the capabilities that is pretty cool in FUSE is the ability to stack filesystems.

So for this demo we'll mount hashfs and mount newerfs on top of it.

Note: due to the design of hashfs we can't stack it on top of newerfs, we could of course change this in the design if we wanted to.

Challenges with FUSE

- Speed - you're running a filesystem on top of an underlying filesystem, also you're bringing data into userspace as well
- Stability - can be a bit fickle, especially if your filesystem has a bug in it :-(
- After a certain number of mounts/unmounts your FUSE may go into a weird state, can be in the hundreds of mounts/unmounts and is very hard to reproduce on demand
- Writing to the filesystem can be risky if your code has a bug in it

So is FUSE Production Ready?

I don't use it production. I do know some people who did use it in production. Haven't talked to them in a while, so I don't know if they still are.

Quite simply for me FUSE is a tool that allows me to do very cool things with filesystems ideally without needing root.

What can FUSE do for you?

Ability to create a custom file system.

E.g. you want a file system that copies everything to a remote machine when it is written. Or just moves all files to a hidden trash folder instead of deleting them.

You want to create custom tools for things like forensics.

You want to experiment/prototype a filesystem. You can prove out your concept without having to write a lot of the hard code.

Do I need to write a FUSE?

Probably not. There are hundreds if not thousands of existing FUSE systems out there. Before you write anything I'd probably start with a duckduckgo search.

PyFileSystem

PyFileSystem is another system that is python only that provides some similar capabilities

It is more true to the “python” way but it only works with python so keep that in mind. Already supports the following Filesytems

WebDav, FTP, SFTPFS, S3FS, TahoeLAFS, Zip

Q & A

Any Questions?

Thanks for listening.

Links

Wikipedia page for FUSE

https://en.wikipedia.org/wiki/Filesystem_in_Userspace

###

fuse-rofs - file system

https://github.com/cognusion/fuse-rofs

Links

Python Fuse Filesystem

https://www.stavros.io/posts/python-fuse-filesystem/

###

WinFsp - Windows File system proxy

http://www.secfs.net/winfsp/

Links

PyFileSystem

https://github.com/PyFilesystem/pyfilesystem