Writing a fuse filesystem

Table of Contents

This is intended to be a step-by-step guide to writing a filesystem using fuse. This version uses the fusepy fuse module for python. (In fedora, install python3-fuse or python3-fusepy, depending on version)

This filesystem is a silly use of fuse and filesystems in general, but should give an idea of what writing a filesystem entails.

circlefs - our example filesystem

Our example filesystem will be a simple in-memory filesystem named circlefs. circlefs will store one piece of information: radius

radius can be changed as desired

In addition to radius, the filesystem will provide files containing the circumference and diameter of a circle with the stored radius, as well as a file containing the value of pi. All files will be read-write, resulting in a corresponding change to radius (except for pi, which will obviously be read-only). Modifying any file will update the radius accordingly, with the other files computed from the radius.

Excuse me... is that a skeleton in your closet?

We start out with nothing but the skeleton python code (several additional imports, just so we don't have to cover them later):

circlefs.py

```
#!/usr/bin/env python3.7

import os
import sys
import time
import math
from errno import *
from stat import S_IFDIR, S_IFREG, S_IFLNK
from fuse import FUSE, Operations, FuseOSError

DEBUG = 0

class circlefs(Operations):

    def __init__(self, **kw):
        self.radius = 0
        print("fuse initialized")

def main(mountpoint):
    FUSE(circlefs(), mountpoint, nothreads=True, foreground=True)

if __name__ == '__main__':
    main(sys.argv[1])

# vim: sw=4 ts=4 noexpandtab
```

Houston, we have a filesystem

Let's test it out and see what it does (hint: nothing yet):

```
$ chmod +x circlefs-devel.py
$ mkdir mnt
$ ./circlefs-devel.py mnt
fuse initialized
```

and if we check it out from another window, what does our filesystem look like?

```
$ ls -lnd mnt
drwxr-xr-x 2 0 0 0 Dec 31  1969 mnt

$ ls -aln mnt
total 0
drwxr-xr-x 2    0    0   0 Dec 31  1969 .
drwxrwxr-x 3 1000 1000 183 Sep 13 10:31 ..

$ df mnt ; df -i mnt
Filesystem     1K-blocks  Used Available Use% Mounted on
circlefs               0     0         0    - /var/tmp/circlefs/mnt
Filesystem     Inodes IUsed IFree IUse% Mounted on
circlefs            0     0     0     - /var/tmp/circlefs/mnt
```

So our filesystem really does nothing at this point. But hey... we did it.

Store at least something

Our filesystem should at least contain the radius, since that's the one piece of information we actually store:

Inside our 'class circlefs' block, we'll add a function which will return a directory listing which includes 'radius' (we'll also have to provide '.' and '..'); that's a readdir—the syscall getdents():

```
    def readdir(self, path, fh=None):
        if DEBUG: print("  readdir({})".format(path))

        # we don't need to support any subdirectories...
        if not path == '/':
            raise FuseOSError(ENOENT)

        dirents = ['.', '..', 'radius']
        for dirent in dirents:
            yield dirent
```

(side note: the 'fh' is for filehandle, which would give us persistence between calls, but we don't really need anything that fancy at this stage)

```
$ ./circlefs-devel.py mnt
fuse initialized
```

```
$ ls -alnd mnt
drwxr-xr-x 2 0 0 0 Dec 31  1969 mnt

$ ls -aln mnt
ls: cannot access 'mnt/radius': No such file or directory
total 0
drwxr-xr-x 2    0    0   0 Dec 31  1969 .
drwxrwxr-x 3 1000 1000 183 Sep 13 10:43 ..
?????????? ? ?    ?      ?            ? radius
```

Okay, so now we have a 'radius', but we haven't told it anything about what type of directory entry 'radius', permissions, file size, etc.

Don't you '?' me, young <whatever>!

For information about directory entries, we need 'getattr()'. We might as well add attributes for the root, while we're at it.

We'll add a getattr method to the class:

```
    def getattr(self, path, fh=None):
        if DEBUG: print("  getattr({})".format(path))

        if path in [ '/' ]:
            mode = 0o755 | S_IFDIR
        elif path in [ '/radius' ]:
            mode = 0o644 | S_IFREG
        else:
            raise FuseOSError(ENOENT)

        now = time.time()
        return {
            'st_atime' : now,
            'st_ctime' : now,
            'st_mtime' : now,
            'st_uid' : os.getuid(),
            'st_gid' : os.getgid(),
            'st_size' : 4096,
            'st_mode' : mode,
            'st_blocks' : 1
        }
```

```
$ ./circlefs-devel.py mnt
fuse initialized
```

```
$ ls -alnd mnt
drwxr-xr-x 0 1000 1000 4096 Sep 13 11:01 mnt

$ ls -aln mnt
total 1
drwxr-xr-x 0 1000 1000 4096 Sep 13 11:02 .
drwxrwxr-x 3 1000 1000  208 Sep 13 11:00 ..
-rw-r--r-- 0 1000 1000 4096 Sep 13 11:02 radius
```

Okay, that's a bit more interesting. Let's see what radius contains:

```
$ cat mnt/radius
cat: mnt/radius: Input/output error
```

Well, I suppose that makes sense; we haven't implemented any functions which allow reading files.

What's in the box?

Let's add a method to allow reading files, and a function which lets us get the value of a particular :

```
    # return the radius as a float
    def get_val(self, var):
        if var == '/radius':
            return 1.0 * self.radius
            return math.pi

    # return the radius as a string
    def get_val_str(self, var):
        val = self.get_val(var)
        return "{}\n".format(val)

    def read(self, path, length, offset, fh=None):
        val = self.get_val_str(path)
        if DEBUG: print("  read({}, length: {}, offset: {}, fh: {}): {}".format(path,
length, offset, fh, val))

        return val.encode('utf-8')[offset:length]
```

```
$ cat mnt/radius
0.0
```

Oh, right. We initialized 'radius' to '0'. And we can't write to it yet:

```
$ echo 42 > mnt/radius
bash: mnt/radius: Read-only file system
```

Well, not really, but 'EROFS' is what 'fusepy' returns whenever a function hasn't been implemented yet.

But I want a purple pony!

Let's allow writing to the 'radius':

```
    def write(self, path, buf, offset, fh=None):
        if DEBUG: print("  write({}, '{}', offset: {}".format(path, buf, offset))

        # don't allow writing at anywhere except the beginning of the file
        # that makes no sense
        if offset is not 0:
            raise FuseOSError(EINVAL)

        val = float(buf)
        if DEBUG: print("  => {}".format(val))

        if path == '/radius':
            self.radius = val
        else: # huh?
            raise FuseOSError(EBADF)

        return len(buf)
```

```
$ echo 99 > mnt/radius
bash: mnt/radius: Read-only file system
```

Uh oh. But we just implemented write. What is actually failing?

```
$ echo 99 | strace -fttTvyo /tmp/trace tee mnt/radius
$ egrep EROFS /tmp/trace
295083 11:46:45.504218 openat(AT_FDCWD, "mnt/radius", O_WRONLY|O_CREAT|O_TRUNC, 0666)
= -1 EROFS (Read-only file system) <0.000166>
```

Okay. That makes some sense. Opening the file read-only didn't really require us to check the open options (for 'O_CREAT' for example).

Well, okay. The things I do for you.

Let's implement an 'open()', and check some flags:

```
    def open(self, path, flags):
        if DEBUG: print("  open({})".format(path))

        if (flags & os.O_CREAT):
            raise FuseOSError(EACCES)

        if (flags & os.O_APPEND) or (flags & os.O_EXCL) or (flags & os.O_DIRECT):
            raise FuseOSError(EINVAL)

        if not path in [ '/radius']:
            raise FuseOSError(EPERM)

        if DEBUG: print("successfully opened '{}'".format(path))
        return 0
```

```
$ echo 99 > mnt/radius
bash: mnt/radius: Read-only file system

$ echo 99 | strace -fttTvyo /tmp/trace tee mnt/radius
$ egrep EROFS /tmp/trace
295083 11:46:45.504218 openat(AT_FDCWD, "mnt/radius", O_WRONLY|O_CREAT|O_TRUNC, 0666)
= -1 EROFS (Read-only file system) <0.000166>
```

Let's enable 'DEBUG' on our filesystem (temporarily), so we can see what's failing.

```
  ...
  getattr(/radius)
  open(/radius)
successfully opened '/radius'
```

Hmm. We opened the file, but never tried to write. We must still be missing something.

'fusepy' is not cooperating

Let's implement the dispatcher method, and print out the operations 'fusepy' thinks are getting called:

```
    def __call__(self, op, *args):
        if DEBUG: print(" call: {}".format(op))
        if not hasattr(self, op):
            if DEBUG: print(" op '{}' not found".format(op))
            raise FuseOSError(EFAULT)
        try:
            ret = getattr(self, op)(*args)
        except Exception as e:
            if DEBUG: print(" op '{}' failed with {}".format(op, e))
            raise e
        if DEBUG: print(" {} returned {}".format(op, ret))
        return ret
```

```
  ...
 call: getattr
  getattr(/radius)
 getattr returned {'st_atime': 1568394991.9264038, 'st_ctime': 1568394991.9264038,
'st_mtime': 1568394991.9264038, 'st_uid': 1000, 'st_gid': 1000, 'st_size': 4096,
'st_mode': 33188, 'st_blocks': 1}
 call: open
  open(/radius)
successfully opened '/radius'
 open returned 0
 call: getxattr
 op 'getxattr' failed with [Errno 95] Operation not supported
 call: truncate
 op 'truncate' failed with [Errno 30] Read-only file system
 call: release
 release returned 0
```

Okay, there we go. Remember that 'EROFS' is what 'fusepy' returns whenever a function isn't implemented? This is probably a bug in 'fusepy', since it would make more sense to return 'ENOSYS'.

That wasn't enough? You want more?

At any rate, we'll add 'truncate()'. When truncating, we'll just set 'radius' to 0.

Earlier we decided that writing to anywhere in the file except the beginning made no sense. That's also true for 'truncate()'.

In the manpage for 'truncate(2)', the most reasonable response to such a condition would be 'EINVAL' (specifically in reference to 'ftruncate(2)', but it fits best). We don't have a real file where 'truncate' makes any sense.

```
    def truncate(self, path, length, fh=None):
        if DEBUG: print("  truncate({})".format(path))
        if length:
            raise FuseOSError(EINVAL)
        self.radius = 0.0
        return length
```

While we're at it, let's make 'open()' also set 'radius' to '0' if 'O_TRUNC' is in the flags. Modifying 'open':

```
...
        if not path in [ '/radius']:
            raise FuseOSError(EPERM)

        if (flags & os.O_TRUNC):
            self.radius = 0.0

        if DEBUG: print("successfully opened '{}'".format(path))
        return 0
```

```
$ cat mnt/radius
0.0

$ echo 99 > mnt/radius

$ cat mnt/radius
99.0
```

We also get 'truncate' for "free":

```
$ cat mnt/radius
99.0

$ truncate -s 5 mnt/radius
truncate: failed to truncate 'mnt/radius' at 5 bytes: Invalid argument

$ truncate -s 0 mnt/radius

$ cat mnt/radius
0.0
```

If only we had a wheelbarrow

Now, let's make 'circlefs' more useful by adding 'diameter' and 'circumference', which contain the appropriate (calculated) values, given the current value of 'radius'. We'll also make them writeable,

with the written value actually modifying 'radius' as necessary.

We'll also add 'pi', just for kicks. It'll have to be read-only, of course.

This will require changes to a number of our methods:

```
@@ -35,7 +35,7 @@
         if not path == '/':
             raise FuseOSError(ENOENT)

-        dirents = ['.', '..', 'radius']
+        dirents = ['.', '..', 'radius', 'diameter', 'circumference', 'pi' ]
         for dirent in dirents:
             yield dirent

@@ -44,8 +44,10 @@

         if path in [ '/' ]:
             mode = 0o755 | S_IFDIR
-        elif path in [ '/radius' ]:
+        elif path in [ '/radius' , '/diameter', '/circumference' ]:
             mode = 0o644 | S_IFREG
+        elif path in [ '/pi' ]:
+            mode = 0o444 | S_IFREG
         else:
             raise FuseOSError(ENOENT)

@@ -61,12 +64,18 @@
             'st_blocks' : 1
         }

-    # return the radius as a float
+    # return the values as floats
     def get_val(self, var):
         if var == '/radius':
             return 1.0 * self.radius
+            if var == '/diameter':
+                    return 2.0 * self.radius
+            if var == '/circumference':
+                    return 2.0 * math.pi * self.radius
+            if var == '/pi':
+                    return math.pi

-    # return the radius as a string
+    # return the values as strings
     def get_val_str(self, var):
         val = self.get_val(var)
         return "{}\n".format(val)


    # return the values as strings
```

```
    def get_val_str(self, var):
@@ -90,6 +98,12 @@

         if path == '/radius':
             self.radius = val
+        elif path == '/diameter':
+            self.radius = val / 2.0
+        elif path == '/circumference':
+            self.radius = val / (2.0 * math.pi)
+        elif path == '/pi':
+            raise FuseOSError(EBADF) # nope, can't change pi
         else: # huh?
             raise FuseOSError(EBADF)

@@ -104,9 +118,23 @@
         if (flags & os.O_APPEND) or (flags & os.O_EXCL) or (flags & os.O_DIRECT):
             raise FuseOSError(EINVAL)

-        if not path in [ '/radius']:
+        if not path in [ '/radius', '/diameter', '/circumference', '/pi' ]:
             raise FuseOSError(EPERM)

+        # from manpage for open(2):
+        # EACCES The requested access to the file is not allowed ...
+        if path == '/pi':
+            # nope, can't truncate pi
+            # also can't open with any mode which includes write
+            if flags & os.O_TRUNC:
+                raise FuseOSError(EACCES)
+
+            # see the manpage for open(2), under the 'File access mode' section
+            # for why this is obscure
+            open_mode = flags & (os.O_RDONLY | os.O_WRONLY | os.O_RDWR)
+            if open_mode == os.O_WRONLY or open_mode == os.O_RDWR:
+                raise FuseOSError(EACCES)
+
         if (flags & os.O_TRUNC):
             self.radius = 0.0

@@ -117,6 +145,8 @@
         if DEBUG: print("  truncate({}, {})".format(path, length))
         if length:
             raise FuseOSError(EINVAL)
+        if path == '/pi':
+            return EBADF
         self.radius = 0.0
         return length
```

```
$ ls -aln mnt
total 3
drwxr-xr-x 0 1000 1000 4096 Sep 13 13:28 .
drwxrwxr-x 3 1000 1000  237 Sep 13 13:28 ..
-rw-r--r-- 0 1000 1000 4096 Sep 13 13:28 circumference
-rw-r--r-- 0 1000 1000 4096 Sep 13 13:28 diameter
-r--r--r-- 0 1000 1000 4096 Sep 13 13:28 pi
-rw-r--r-- 0 1000 1000 4096 Sep 13 13:28 radius

$ cat mnt/radius
0.0

$ echo 99 > mnt/radius

$ cat mnt/radius
99.0

$ cat mnt/circumference
622.0353454107791

$ cat mnt/diameter
198.0

$ cat mnt/pi
3.141592653589793

$ for f in mnt/* ; do echo "$f: $(cat $f)" ; done
mnt/circumference: 622.0353454107791
mnt/diameter: 198.0
mnt/pi: 3.141592653589793
mnt/radius: 99.0


$ echo 6.283185307179586 > mnt/circumference

$ for f in mnt/* ; do echo "$f: $(cat $f)" ; done
mnt/circumference: 6.283185307179586
mnt/diameter: 2.0
mnt/pi: 3.141592653589793
mnt/radius: 1.0

$ echo 14 > mnt/pi
bash: mnt/pi: Permission denied
```

Just for kicks, let's also add a directory entry for 'π', as a symlink to 'pi':

```
@@ -36,7 +36,7 @@
         if not path == '/':
             raise FuseOSError(ENOENT)

-        dirents = ['.', '..', 'radius', 'diameter', 'circumference', 'pi' ]
+        dirents = ['.', '..', 'radius', 'diameter', 'circumference', 'pi', 'π' ]
         for dirent in dirents:
             yield dirent

@@ -49,6 +49,8 @@
             mode = 0o644 | S_IFREG
         elif path in [ '/pi' ]:
             mode = 0o444 | S_IFREG
+        elif path in [ '/π' ]:
+            mode = 0o777 | S_IFLNK
         else:
             raise FuseOSError(ENOENT)

@@ -165,6 +167,12 @@
             'f_frsize' : 0,
             'f_namemax' : 255
         }
+    def readlink(self, path):
+        if path == '/π':
+            return 'pi'
+
+        return ''
+

 def main(mountpoint):
     FUSE(circlefs(), mountpoint, nothreads=True, foreground=True)
```

What I wouldn’t give for a holocaust cloak!

Of course, no filesystem would be complete without 'df' causing panic over the filesystem being full, so let’s make it happen:

```
    def statfs(self, path):
        if DEBUG: print("  statfs({})".format(path))
        return {
            'f_bsize' : 42,
            'f_bfree' : 0,
            'f_bavail' : 0,
            'f_blocks' : 1,
            'f_files' : 1,
            'f_ffree' : 0,
            'f_favail' : 0,
            'f_flag' : 0,
            'f_frsize' : 0,
            'f_namemax' : 255
        }
```

```
$ df mnt
Filesystem     1K-blocks  Used Available Use% Mounted on
circlefs               1     1         0 100% /var/tmp/circlefs/mnt

$ df -i mnt
Filesystem     Inodes IUsed IFree IUse% Mounted on
circlefs            1     1     0  100% /var/tmp/circlefs/mnt

$ stat -f mnt
  File: "mnt"
    ID: 0        Namelen: 255     Type: fuseblk
Block size: 42         Fundamental block size: 42
Blocks: Total: 1          Free: 0          Available: 0
Inodes: Total: 1          Free: 0
```

Much better :)

I did all my chores! Can I go out and play now?

Here's our 'completed' filesystem:

circlefs.py

```
#!/usr/bin/env python3.7

import os
import sys
import time
import math
from errno import *
```

```
from stat import S_IFDIR, S_IFREG, S_IFLNK
from fuse import FUSE, Operations, FuseOSError

DEBUG = 0

class circlefs(Operations):

    def __init__(self, **kw):
        self.radius = 0
        print("fuse initialized")

    def __call__(self, op, *args):
        if DEBUG: print(" call: {}".format(op))
        if not hasattr(self, op):
            if DEBUG: print(" op '{}' not found".format(op))
            raise FuseOSError(EFAULT)
        try:
            ret = getattr(self, op)(*args)
        except Exception as e:
            if DEBUG: print(" op '{}' failed with {}".format(op, e))
            raise e
        if DEBUG: print(" {} returned {}".format(op, ret))
        return ret

    def readdir(self, path, fh=None):
        if (DEBUG): print("  readdir({})".format(path))

        # we don't need to support any subdirectories...
        if not path == '/':
            raise FuseOSError(ENOENT)

        dirents = ['.', '..', 'radius', 'diameter', 'circumference', 'pi', 'π' ]
        for dirent in dirents:
            yield dirent

    def getattr(self, path, fh=None):
        if DEBUG: print("  getattr({})".format(path))

        if path in [ '/' ]:
            mode = 0o755 | S_IFDIR
        elif path in [ '/radius' , '/diameter', '/circumference' ]:
            mode = 0o644 | S_IFREG
        elif path in [ '/pi' ]:
            mode = 0o444 | S_IFREG
        elif path in [ '/π' ]:
            mode = 0o777 | S_IFLNK
        else:
            raise FuseOSError(ENOENT)

        now = time.time()
        return {
```

```
            'st_atime' : now,
            'st_ctime' : now,
            'st_mtime' : now,
            'st_uid' : os.getuid(),
            'st_gid' : os.getgid(),
            'st_size' : 4096,
            'st_mode' : mode,
            'st_blocks' : 1
        }

    # return the values as floats
    def get_val(self, var):
        if var == '/radius':
            return 1.0 * self.radius
        if var == '/diameter':
            return 2.0 * self.radius
        if var == '/circumference':
            return 2.0 * math.pi * self.radius
        if var == '/pi':
            return math.pi

    # return the values as strings
    def get_val_str(self, var):
        val = self.get_val(var)
        return "{}\n".format(val)

    def read(self, path, length, offset, fh=None):
        val = self.get_val_str(path)
        if DEBUG: print("  read({}, length: {}, offset: {}, fh: {}): {}".format(path,
length, offset, fh, val))

        return val.encode('utf-8')[offset:length]

    def write(self, path, buf, offset, fh=None):
        if DEBUG: print("  write({}, '{}', offset: {}".format(path, buf, offset))

        # don't allow writing at anywhere except the beginning of the file
        # that makes no sense
        if offset is not 0:
            raise FuseOSError(EINVAL)

        val = float(buf)
        if DEBUG: print("  => {}".format(val))

        if path == '/radius':
            self.radius = val
        elif path == '/diameter':
            self.radius = val / 2.0
        elif path == '/circumference':
            self.radius = val / (2.0 * math.pi)
        elif path == '/pi':
```

```
            raise FuseOSError(EBADF) # nope, can't change pi
        else: # huh?
            raise FuseOSError(EBADF)

        return len(buf)

    def open(self, path, flags):
        if DEBUG: print("  open({})".format(path))

        if (flags & os.O_CREAT):
            raise FuseOSError(EACCES)

        if (flags & os.O_APPEND) or (flags & os.O_EXCL) or (flags & os.O_DIRECT):
            raise FuseOSError(EINVAL)

        if not path in [ '/radius', '/diameter', '/circumference', '/pi' ]:
            raise FuseOSError(EPERM)

        # from manpage for open(2):
        # EACCES The requested access to the file is not allowed ...
        if path == '/pi':
            # nope, can't truncate pi
            # also can't open with any mode which includes write
            if flags & os.O_TRUNC:
                raise FuseOSError(EACCES)

            # see the manpage for open(2), under the 'File access mode' section
            # for why this is obscure
            open_mode = flags & (os.O_RDONLY | os.O_WRONLY | os.O_RDWR)
            if open_mode == os.O_WRONLY or open_mode == os.O_RDWR:
                raise FuseOSError(EACCES)

        if (flags & os.O_TRUNC):
            self.radius = 0.0

        if DEBUG: print("successfully opened '{}'".format(path))
        return 0

    def truncate(self, path, length, fh=None):
        if DEBUG: print("  truncate({}, {})".format(path, length))
        if length:
            raise FuseOSError(EINVAL)
        if path == '/pi':
            return EBADF
        self.radius = 0.0
        return length

    def statfs(self, path):
        if DEBUG: print("  statfs({})".format(path))
        return {
            'f_bsize' : 42,
```

```
            'f_bfree' : 0,
            'f_bavail' : 0,
            'f_blocks' : 1,
            'f_files' : 1,
            'f_ffree' : 0,
            'f_favail' : 0,
            'f_flag' : 0,
            'f_frsize' : 0,
            'f_namemax' : 255
        }
    def readlink(self, path):
        if path == '/π':
            return 'pi'

        return ''


def main(mountpoint):
    FUSE(circlefs(), mountpoint, nothreads=True, foreground=True)

if __name__ == '__main__':
    main(sys.argv[1])

# vim: sw=4 ts=4 noexpandtab
```

Think you're a wise-guy, eh? Let's see you do this, then!

some exercises for the reader:

- update mtime/ctime when the values get updated; retain latest time update
- update atime
- make filesystem size dependent on the radius (or area?)
- add:
  - area: pi * radius^2
  - volume (of a sphere): 4/3 * pi * radius^3
  - pieces and serving (number of pieces to slice the pie/pizza, and area of a serving)
  - use this to provide number of inodes for statfs
    - (hint: don't modify the radius, just one of these two, and calculate the other)
    - bonus: how much of the pie/pizza has already been eaten/is remaining? maybe add 'eat_slice', which subtracts from the number of remaining slices, or size of the remaining area?
- implement some other silly filesystem (squarefs, rectanglefs, ?)

- implement a slightly-less-silly filesystem that actually stores data
- re-implement circlefs inside a real kernel module