MarFS Fuse Specs

One must figure out how to hide the gpfs/POSIX metadata only mountpoint from the user, so it is possible that the fuse process family needs to do a private mount of the gpfs metadata name space using unshare or other mechanisms

I imagine there would be shared code in read/write/open/init etc. with pftool

As you can see, the bulk of the ops are just do the obvious to the POSIX/gpfs file system mount point. Only open/read/write/trunc/unlink are really different from just default behavior. We can make some simplifying assumptions, maybe read only for a while except for pftool. Maybe some limits on write like no packing, maybe no multi-part for a while. Its relatively easy to add function as we go.

It looks like a lot but 70% is just copy from a working fuse example.

The other 30% is copy concepts/routines from pftool.

Fuse consists of the following functions:

int(\* [getattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a7a4c5d8eaf7179d819618c0cf3f73724) )(const char \*, struct stat \*)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

No need for access check, just try the op

appropriate statlike call filling in fuse structure (dont mess with xattrs here etc.)

set\_security\_saved()

return

int(\* [readlink](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ab4ce6e6d69dfde3ec550f22d932c5633) )(const char \*, char \*, size\_t)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

No need for access check, just try the op

Appropriate readlinklike call filling in fuse structure

set\_security\_saved()

return

int(\* [mknod](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1465eb2268cec2bb5ed11cb09bbda42f) )(const char \*, mode\_t, dev\_t)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

Check/act on quota num names

Only S\_IFREG flag is supported, so if its not that, refuse

No need for access check, just try the op mknod()

(all files in marfs are born as just regular files and only get xattrs if they need them upon writing)

set\_security\_saved()

return

int(\* [mkdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a0a38aa6ca60e945772d5d21b0c1c8916) )(const char \*, mode\_t)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

Check/act on quota num files

No need for access check, just try the op

Appropriate mkdirlike call filling in fuse structure

set\_security\_saved()

return

int(\* [unlink](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a8bf63301a9d6e94311fa10480993801e) )(const char \*)

set\_security\_user()

Expand\_path\_info(full)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWMRD UD

Call access syscall to check/act if allowed to unlink for this user

Trash\_file() renames file with all xattrs into trashdir preserves objects and paths

set\_security\_saved()

return

int(\* [rmdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ac59578d18db12f0142ae1ab6e8812d55) )(const char \*)

(we wont trash directories, we will preserve full paths of all files trashed instead)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

No need for access check, just try the op

Appropriate rmdirlike call filling in fuse structure

set\_security\_saved()

return

int(\* [symlink](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ab86022391e56a8ad3211cf754b5b5ebe) )(const char \*, const char \*)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

No need for access check, just try the op

Appropriate symlink call filling in fuse structure

set\_security\_saved()

return

int(\* [rename](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#aa777cbddc91887b117ac414e9a2d3cb5) )(const char \*, const char \*)

set\_security\_user()Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

No need for access check, just try the op

Appropriate rename call filling in fuse structure

set\_security\_saved()

return

int(\* [link](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1b234c43e826c6a690d80ea895a17f61) )(const char \*, const char \*)

for now, I think we should not allow link – its pretty complicated to do

int(\* [chmod](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a7e75d299efe3a401e8473af7028e5cc5) )(const char \*, mode\_t)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

No need for access check, just try the op

Appropriate chmod call filling in fuse structure

set\_security\_saved()

return

int(\* [chown](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a40421f8a43e903582c49897894f4692d) )(const char \*, uid\_t, gid\_t)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

No need for access check, just try the op

Appropriate chown call filling in fuse structure

set\_security\_saved()

return

int(\* [truncate](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a8efb50b9cd975ba8c4c450248caff6ed) )(const char \*, off\_t)

\*\*\* this may not be needed until we implement write in the fuse daemon \*\*\*

set\_security\_user()

Expand\_path\_info(full)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWMRD TD

If this is just a normal md is the file data

And if in expandpathinfo area update in place is allowed, then trunc as the user asks and return

} otherwise if it’s a normal file md Is in the file data

and update in place is not allowd

only trunc the file if the user asked for zero and return

}

since this is an object repo then we need to make sure trunc is zero and don’t honor anything else

call access syscall to check/act if allowed to truncate for this user

Trash\_dup\_file() copies metadata to trash, resets original file zero len and no reserved xattr

set\_security\_saved()

return

int(\* [open](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a14b98c3f7ab97cc2ef8f9b1d9dc0709d) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

set\_security\_user()

Call access to see if we can do this operation on a file (before we go to a lot of trouble)

Expand\_path\_info( full)

If multi-file and if you need to get the chunk list/actual lengths/checksums,etc, then call read\_multi\_obj(pass expanded path info, this adds the multi object info so the structure has it all now) \*\*\* this is not necessary to just read the file, its only necessary if you are checking a checksum per object or have used some kind of compression or encryption that requires you to know the actual size of the object before reading it, etc.

Trunc and create will never come in as options, as fuse will trunc a file before calling open if open/trunc is called by user and mknod then open will be called if it is an open/create from the user. We don’t do append, so really we only accept rdonly, rdrw, and wronly

Check/act on iperms from expanded\_path\_info\_structure

If readonly RM RD

If wronly/rdwr RMWMRDWD

If append we don’t support that so toss an error

We don’t need to check quota, that is done in mknod

Poke the address of the expanded path info which has everything we know about the file into fuse\_file\_info fuse structure (which will be passed by fuse on subsequent use of the FD)

open md file in asked for mode

set\_security\_saved()

return

int(\* [read](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a2a1c6b4ce1845de56863f8b7939501b5) )(const char \*, char \*, size\_t, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

set\_security\_user()

we don’t need to expand\_path because we are passed an FD which has fuse\_file\_info where the expanded path info including xattrs and if appropriate the multi objid info

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM RD (really don’t need this permissions check because we get an FD so it has been checked on open)

No need to call access as we called it in open for read

Case

File has no xattr objid

Just read the bytes from the file and fill in fuse read buffer

File is objtype packed or uni

Make sure start and end are within the object according to (file size and objoffset)

Make sure security is set up for accessing objrepo using table (if needed)

Read bytes from object server and fill in fuse read buffer

File is objtype multipart

Loop through this until you get all the read honored

Make sure start and end are within the object according to (file size and objoffset)

By looking up in the multi objid area in the expand path info structure pointer …

Make sure security is set up for accessing objrepo using table

File is striped

We will implement this later perhaps

}

set\_security\_saved()

return

int(\* [write](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a897d1ece4b8b04c92d97b97b2dbf9768) )(const char \*, const char \*, size\_t, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

\*\*\* this may not be needed until we implement write in the fuse daemon \*\*\*

set\_security\_user()

we don’t need to do an expand path as we are passed in an FD fuse\_file\_info which has everything we know about the file

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWMRDWD

Really don’t need the permissions check as we get an FD, this should have been resolved in open

No need to call access as we called it in open for write

The expand path info will tell us where to write from the iwrite\_repo

Make sure security is set up for accessing objrepo using iwrite\_datarepo if necessary

If first write, check/act on quota bytes

If updateinplace in the expand\_path\_info area is yes and

If we are writing to the metadata file, just do the write and return

}

If first write, it has to start at offset 0, if not fail

If write does not start at previous write ending, fail

If we are writing to object system ( which doesn’t support update in place yet)

If first write allocate space for current obj being written put current object marker in fuse\_file\_info for use by next write or release

If first write or if new file length will make object bigger than iwrite\_chunksize

seal (add all appropriate metadata to mds and to end of object) old ojb get new obj

If first write, add MARobjid with all fields

If “new” obj

If first “new” obj

Write old and new chunknumber, space used, and correctness info for chunk to md file and update MARppost numobjects and chunkinfobytes

We probably need to sit down and walk through open write, write, release

To get the ordering to be correct for each operation and what metadata needs updating gwhen, etc/

Add change MARpost type to multi

Else

Write new chunk to mdfile;

}

Put objid into current obj being written into fuse open\_file\_info

}

Write bytes to object

Trunc file to current last byte

set\_security\_saved()

return

int(\* [statfs](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a4e765e29122e7b6b533dc99849a52655) )(const char \*, struct statvfs \*)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

Call statvfs the metadata file system and fill in structure

Open and read from lazy-fsinfo data file updated by batch process fsinfopath

Update the structure with quota= file system size and used info etc. you will need to calculate free blocks, free blocks for unpriv users, inodes, free inodes,

This is the statvfs structure to fill in

struct statvfs {

unsigned long f\_bsize; /\* file system block size \*/

unsigned long f\_frsize; /\* fragment size \*/

fsblkcnt\_t f\_blocks; /\* size of fs in f\_frsize units \*/

fsblkcnt\_t f\_bfree; /\* # free blocks \*/

fsblkcnt\_t f\_bavail; /\* # free blocks for unprivileged users \*/

fsfilcnt\_t f\_files; /\* # inodes \*/

fsfilcnt\_t f\_ffree; /\* # free inodes \*/

fsfilcnt\_t f\_favail; /\* # free inodes for unprivileged users \*/

unsigned long f\_fsid; /\* file system ID \*/

unsigned long f\_flag; /\* mount flags \*/

unsigned long f\_namemax; /\* maximum filename length \*/

};

we may want to play with the fs block size to see if that would be useful to set to something other than the metadata file sytem info

set\_security\_saved()

return

int(\* [flush](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ad4ec9c309072a92dd82ddb20efa4ab14) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

I don’t think we will have dirty data that we can control

I guess we could call flush on the filehandle that is being written

But the only data we will write is multi-part objects,

All other data would be to some object interface

int(\* [release](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#abac8718cdfc1ee273a44831a27393419) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

set\_security\_user()

don’t need expandpath info because we have fuse\_file\_info

if writing there may be an objid stuffed into a address in fuse open table

seal that object if needed

update any xattrs/trunc file if needed

free the area holding that objid fuse\_file\_info stuff

if reading, there might be a malloced space for read obj mgmt. in fuse open table

close any objects if needed

free the area holding that stuff in fuse\_file\_info

close the metadata file handle

set\_security\_saved()

return

int(\* [fsync](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a92bdd6f43ba390a54ac360541c56b528) )(const char \*, int, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

I don’t know if we do anything here, I don’t think so, we will be in sync at the end of each thread end

Just return

int(\* [setxattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a988ced7091c2821daa208e6c96d8b598) )(const char \*, const char \*, const char \*, size\_t, int)

\*\*\* this may not be needed until we implement user xattrs in the fuse daemon \*\*\*

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

\*\*\* make sure they aren’t setting a reserved xattr\*\*\*

No need for access check, just try the op

Appropriate setxattr call filling in fuse structure

set\_security\_saved()

return

int(\* [getxattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ae21503c64fe2990c8a599f5ba339a8f2) )(const char \*, const char \*, char \*, size\_t)

\*\*\* this may not be needed until we implement user xattrs in the fuse daemon \*\*\*

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

\*\*\* make sure they aren’t getting a reserved xattr\*\*\*

No need for access check, just try the op

Appropriate getxattr call filling in fuse structure

set\_security\_saved()

return

int(\* [listxattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ab4a9c361ce48406f07d5a08ab03f5de8) )(const char \*, char \*, size\_t)

\*\*\* this may not be needed until we implement user xattrs in the fuse daemon \*\*\*

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

No need for access check, just try the op

Appropriate listxattr call

\*\*\* remove any reserved xattrs from list \*\*\*

filling in fuse structure

set\_security\_saved()

return

int(\* [removexattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a5e54de801a0e0d7019e4579112ecc477) )(const char \*, const char \*)

\*\*\* this may not be needed until we implement user xattrs in the fuse daemon \*\*\*

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

\*\*\* make sure they aren’t removing a reserved xattr\*\*\*

No need for access check, just try the op

Appropriate removexattr call filling in fuse structure

set\_security\_saved()

return

int(\* [opendir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1813889bc5e6e0087a936b7abe8b923f) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Stuff some information about the open directory in the dirp

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

No need for access check, just try the op

Appropriate opendir call filling in fuse\_file\_info structure

set\_security\_saved()

return

int(\* [readdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a0f634deda31d1e1c42664585ae820076) )(const char \*, void \*, [fuse\_fill\_dir\_t](http://fuse.sourceforge.net/doxygen/fuse_8h.html#ae2a2054f9852fd6020c26a1bcc7f1042), off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

set\_security\_user()

you get a dirp in fuse\_file\_info so you don’t need to run expand\_path\_info

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

You shouldn’t need to check the permissions as you get a dirp passed in

No need for access check, just try the op

Appropriate readdir call filling in fuse structure (fuse does this in chunks)

set\_security\_saved()

return

int(\* [releasedir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a729e53d36acc05a7a8985a1a3bbfac1e) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

set\_security\_user()

you get a dirp, so you don’t need to run expand\_path\_info

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

No need for access check, just try the op

Appropriate closedir call filling in fuse structure

set\_security\_saved()

return

int(\* [fsyncdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#aba5cc1fe9a63ec152ceb19656f243256) )(const char \*, int, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

don’t think there is anything to do here, we wont have dirty data unless its trash

just return

void \*(\* [init](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#adc6dc71274f185de72217e38d62142c4) )(struct [fuse\_conn\_info](http://fuse.sourceforge.net/doxygen/structfuse__conn__info.html) \*conn)

read up object config/db into tables in memory

use sticky bit or other method to read any secrets needed for repos requiring secrets and maybe obscure in memory

void(\* [destroy](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ac41d37ab860204fe4bd7612f9fb036c5) )(void \*)

called when fuse file system exits

nothing for us to do here, we wont have dirty data when the fuse daemon

I suppose they wait for all threads to finish before leaving, so this should be ok

Just return

int(\* [access](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a2248db35e200265f7fb9a18348229858) )(const char \*, int)

set\_security\_user()

Expand\_path\_info()

Check/act on iperms from expanded\_path\_info\_structure, this op requires RM

No need for access check, just try the op

Appropriate access call filling in fuse structure

set\_security\_saved()

return

int(\* [create](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a97243e0f9268a96236bc3b6f2bacee17) )(const char \*, mode\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

we don’t need this as if its not present, fuse will call mdnod and open

don’t implement

int(\* [ftruncate](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1e492882859740f13cbf3344cf963c70) )(const char \*, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

Fuse docs say that if this isn’t present it will call truncate() so lets not do this

int(\* [fgetattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a573d79862df591c98e1685225a4cd3a5) )(const char \*, struct stat \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

Fuse docs say only called if creat() is called and we didn’t implement that, we implemented mknod instead

int(\* [lock](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1c3fff5cf0c1c2003d117e764b9a76fd) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, int cmd, struct [flock](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ad5968d566dab370974043fcf4271eb25) \*)

don’t support it, either don’t implement or throw error

int(\* [utimens](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a79955861cc5eb006954476607ef28944) )(const char \*, const struct timespec tv[2])

set\_security\_user()

Expand\_path\_info(don’t stat\_xattr)

Check/act on iperms from expanded\_path\_info\_structure, this op requires RMWM

No need for access check, just try the op

Appropriate utimens call filling in fuse structure

set\_security\_saved()

return

int(\* [bmap](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ae3f3482e33a0eada0292350d76b82901) )(const char \*, size\_t blocksize, uint64\_t \*idx)

don’t support its is for block mapping

int(\* [ioctl](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a37f0612d67a6b76bf10fe6a71b0e3b5b) )(const char \*, int cmd, void \*arg, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, unsigned int flags,

void \*data)

if we need an ioctl for something or other

\*\*\*Maybe a way to get fuse deamon to read up new config file

\*\*\* we need a way for daemon to read up new config file without stopping

int(\* [poll](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a2c02838d30391c09dd5213edc61e106a) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, struct fuse\_pollhandle \*ph, unsigned \*reventsp)

either don’t implement or just return

int(\* [flock](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ad5968d566dab370974043fcf4271eb25) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, int op)

don’t implement or throw error

int(\* [fallocate](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a4fa8203e4bfa71d62c15deb5dffe4867) )(const char \*, int, off\_t, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

In reading more about this it is mostly about punching holes in files and preallocating space in a file. I don’t think we want to tackle this at this time.

It does pass in an open file handle though