FUSE Kernel Operations

Function Specifications

Vikas Gera

Market Development Engineering
Sun Microsystems, Inc.

Version 1
September 22, 2006

Document History

Date	Version	Author	Changes
09/22/06	1	Vikas Gera	
10/03/06	2	Vikas Gera	Added missing input parameters to readdir function.

Table of Contents

What is Fuse?

Fuse is a framework for implementing filesystems in user land. It eliminates the need for writing kernel modules for implementing filesystems. It has Linux and FreeBSD ports and Solaris port is planned.

Fuse itself has two parts. A kernel module and a user land resident lib which provides a framework and API (Application Programming Interface) using which a programmer can implement a filesystem residing in user land.

The advantage is that any programming language can be used for implementing a file system. The filesystem author can use all the POSIX and user space libs; this overcomes the constraint of the kernel space having a limited set of API. Also, the filesystem implementer is isolated from Kernel programming and OS intricacies.

An example of FUSE application is that, you can mount Gmail in-box as a directory on your home directory! Or one can create a filesystem in user land which maps Amazon S3 (Simple Secure Storage) onto your local directory.

Target audience

This specification is intended for developers of Kernel portion to support Fuse Library. This specification is intended to remove the need for a kernel developer to refer to the FUSE library or any other source code. And, will help the developer to create a clean room FUSE Kernel module implementation.

Source of this document

This document has been prepared by reading the Fuse Library 2.5.3 (LGPL), BSD Fuse Kernel and *fuse_kernel.h* (Dual licensed as BSD and GPL)

Fuse alternatives

Lufs, WebNFS, HTTPFS and NFSv4 are a few examples of FUSE alternative. Though each differs from the other architecturally, they operate in similar application domain.
NFS v4 is an IETF defined protocol. See: http://www.nfsv4.org
LUFS is the closest FUSE alternative. The FUSE site mentions LUFS.
LUFS is a hybrid user space file system framework supporting a number of file systems transparently for any application. See: http://lufs.sf.net

How Fuse works

Fuse has two parts:
1. User land lib providing the framework for the filesystem writer.
2. Kernel module, which acts as a surrogate filesystem as well as a char device.

The FUSE lib provides a framework to the filesystem writer.

The kernel part bridges the requests generated by other user processes, that access the mounted user filesystem, using system console commands like *ls* and *cat*.

The FUSE Kernel module registers itself with the OS as a filesystem as well as a char device. The char device is used to communicate with the FUSE lib and the userland filesystem framework API (Application Provider Interface).

The userland lib forks a process that changes the *suid*(), then opens the FUSE char device. It then calls the *mount*() syscall passing the file-descriptor (file descriptor) as a parameter. Since the file descriptor is unique per process, the FUSE kernel module's VFS (Virtual File System or Virtual File-system Switch) mount operation callback function (VFS_MOUNT) is called by the syscall mount() via its internal function *domount*().

The userland forked process which opens the FUSE char device and then calls the mount() syscall passing this file-descriptor as a parameter, so that the FUSE kernel can co-relate the char device instance with the mount point. Finally, the forked userland mount caller process, passes back the file-descriptor to its parent using UNIX domain socket, which translates file-descriptor from one process to another; This code can be found in the book, Unix Network Programming by W Richard Stevens (ISBN-81-203-0749-6) pg 306 section 6.10.

Thus the filesystem part of the FUSE kernel module receives the mount path and the file descriptor passed by the FUSE lib as the syscall mount parameter. The FUSE kernel module uses *fget*() in Linux to get the FILE pointer (*getf*() in Solaris is the *fget*() counterpart), from which it gets the vnode pointer and from this finally gets the device (*dev_t*) member. The *dev_t* is an integer containing major and minor device numbers. The minor number is what the FUSE kernel needs to correlate the mount point with the char device instance. This correlation is used to pass the VFS and VNODE operation callbacks generated on the filesystem back to the userland filesystem framework API, via the char device instance.

See the Solaris *namefs* for an equivalent implementation.

Once the mount point to FUSE char device co-relation is established, the FUSE kernel module passes all the VFS and VNODE filesystem callbacks to the FUSE userland framework API. This happens at the userland FUSE lib. It simple calls *read*() and *write*() syscall using the file-descriptor of the opened FUSE char device.

The FUSE lib allocates a header and detail structures and call *readw*(), this gets translated to the FUSE char device *read*() callback. Within the FUSE char device, this call blocks, pending a filesystem request. Solaris implementation could use semaphores for process synchronization.

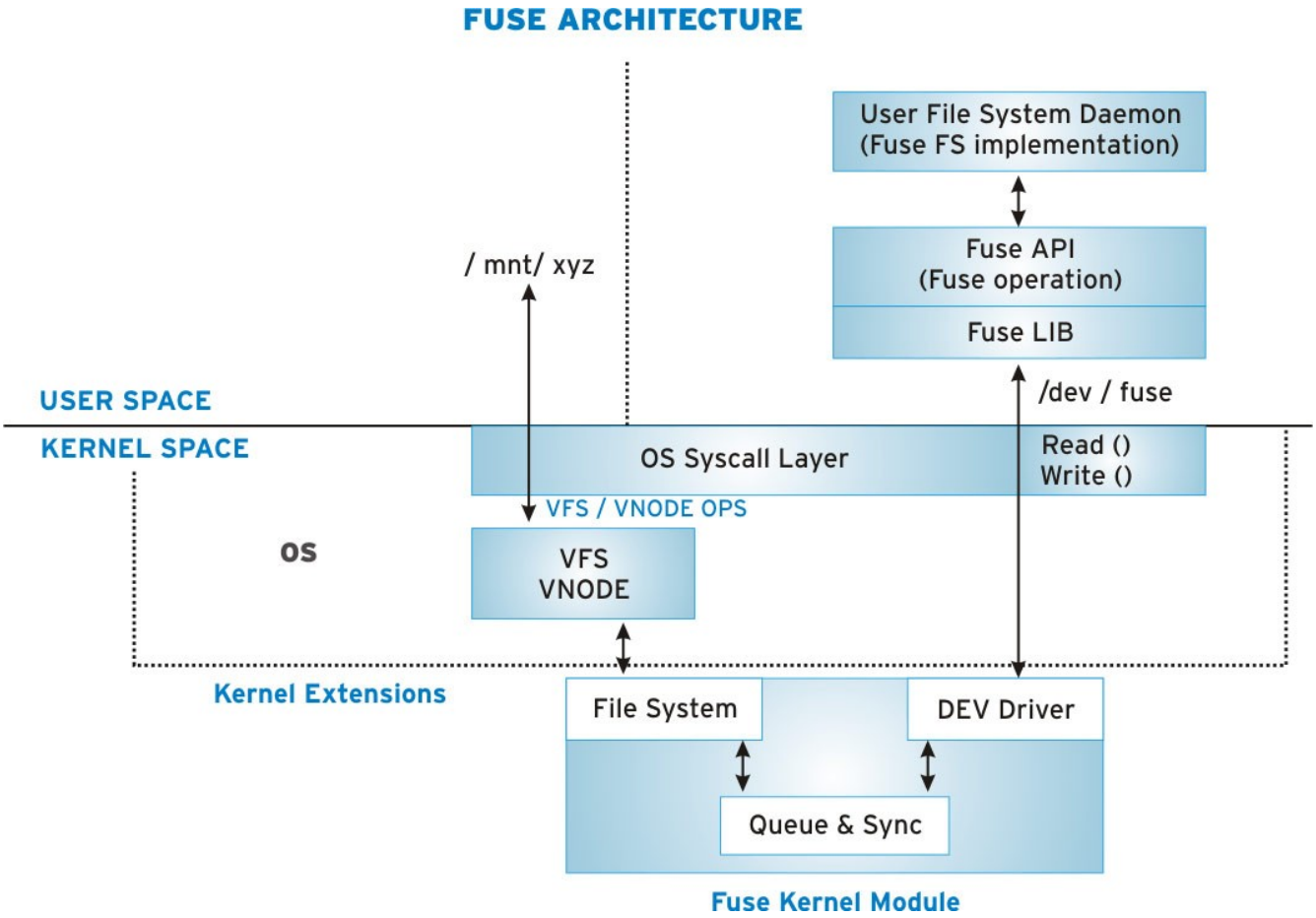
When a filesystem callback is encountered by the FUSE filesystem, it fills in the request structure, header and details and indicates the filesystem operation requested using an

enumeration of FUSE OPCODEs. It places the request in a queue and wakes up the *read*() process and goes to sleep itself.

When the *read*() callback of the FUSE char device is woken up by an incoming filesystem request, it returns back to the userland caller, that is the FUSE lib Framework API. This framework then calls the appropriate callback of the FUSE OPERATION API registered during the initialization process of the FUSE userland framework filesystem implement. on return from the FUSE framework file operation callback, the FUSE lib passes the result back to the FUSE kernel module using the syscall *writew*() which gets translated to *write*() callback of the FUSE char device.

The *write*() callback function of the FUSE char device receives the response from the userland framework API. It then wakes up the sleeping VFS/VNODE filesystem callback. Thus this response is passed backed to the caller process which originally requested a file system operation say the *ls* or the *cat* command.

Fuse architecture diagram



Fuse Kernel can be divided into three logical parts.
1. Filesystem part
2. Device driver part
3. Synchronization and queue part

The Fuse kernel module registers itself as a filesystem as well as a char device.

The char device part of the module is used for communicating the request and getting the response back from user land resident Fuse framework API.

The Filesystem part of the Fuse kernel module, acts as the gateway to plumb the VFS and VNODE callbacks from kernel to the userland filesystem Fuse operations API implemented by the Fuse user filesystem implement.

The synchronization and queue part acts as the glue between the filesystem and device parts of the Fuse kernel module.

FUSE operations specifications

Following table provides the mapping between fuse operations, *vnode* operations and their corresponding OPCODEs:

Note The fuse operation functions are the member of fuse_operations structure callbacks which userland filesystem author implements.

The "*vop_xxxx*" names corresponds to the *vnode* callback functions defined in *vnode.h*

FUSE Operation	***VFS Operation***	***OPCODE***	
getattr	vop_getattr	FUSE_GETATTR	3
fgetattr	No Operation	FUSE_GETATTR	3
chmod	vop_setattr	FUSE_SETATTR	4
chown	vop_setattr	FUSE_SETATTR	4
truncate	vop_setattr	FUSE_SETATTR	4
utime	vop_setattr	FUSE_SETATTR	4
readlink	vop_readlink	FUSE_READLINK	5
symlink	vop_symlink	FUSE_SYMLINK	6
mknod	vop_create	FUSE_MKNOD	8
mkdir	vop_mkdir	FUSE_MKDIR	9
unlink	vop_remove	FUSE_UNLINK	10
rmdir	vop_rmdir	FUSE_RMDIR	11

FUSE Operation	*VFS Operation*	*OPCODE*	
rename	vop_rename	FUSE_RENAME	12
link	vop_link	FUSE_LINK	13
open	vop_open	FUSE_OPEN	14
read	vop_read	FUSE_READ	15
write	vop_write	FUSE_WRITE	16
statfs	vop_statvfs	FUSE_STATFS	17
release	vop_release	FUSE_RELEASE	18
fsync	vop_fsync	FUSE_FSYNC	20
setxattr	No Operation	FUSE_SETXATTR	21
getxattr	No Operation	FUSE_GETXATTR	22
listxattr	No Operation	FUSE_LISTXATTR	23
removexattr	No Operation	FUSE_REMOVEXATTR	24
init	vfs_mount	FUSE_INIT	26
opendir	vop_opendir	FUSE_OPENDIR	27
readdir	vop_readdir	FUSE_READDIR	28
releasedir	vop_closedir	FUSE_RELEASEDIR	29
destroy	No Operation		
access	vop_access	FUSE_ACCESS	34
create	vop_create	FUSE_CREATE	35
ftruncate	vop_open		

Brief Overview of Fuse Library :

Fuse provides a means for a developer to implement a filesystem in userland. The Fuse Library interacts with a kernel module, to marshall the filesystem calls to Fuse operations callback.

The Fuse kernel module has two portions. A character device and a filesystem. These two portions collaborate with each other through a queue mechanism to marshall the filesystem calls to Fuse operations callback via Fuse library.

The Fuse library opens a character device and goes into a loop:

1. Read Device

2. Process (un-marshall IN parameters; call the Fuse operation callback; marshall back the result)

3. Write Device

Marshalling parameters description

The FUSE I/O messages have a header followed by a payload. The payload varies as per the FUSE OP Code. The details are as follows:

FUSE Operation Name	*lookup*	
Description	The function the existence and validity of directory/file.	
OPCODE		1
Corresponding POSIX Call(s)		
IN Parameters		
const char*	path	Path to the directory/file.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 3)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
uint64_t	Inode ID	
uint64_t	Inode generation ID – must be unique for filesystem's lifetime.	

FUSE Operation Name	*lookup*
uint64_t	Cache timeout for the name
uint64_t	Cache timeout for the attributes
uint32_t	Cache timeout in nanoseconds for the name
uint32_t	Cache timeout for attributes in nanoseconds
uint64_t	Inode ID (duplicate/ redundant)
uint64_t	Size
uint64_t	Number of Blocks
uint64_t	Access Time of the Inode
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID
uint32_t	Device ID

FUSE Operation Name	*getattr*	
Description	The function gets the attributes of the directory/file.	
OPCODE	3	
Corresponding POSIX Call(s)	*lstat()*	
IN Parameters		
const char*	path	Path to the directory/file.
OUT Parameters		
struct stat*	stbuf	Structure file stat which would be filled by the call.

FUSE Operation Name	*getattr*
In Message	Message to FUSE Library
uint32_t	Length of message
uint32_t	Opcode of the function (value = 3)
uint64_t	Unique ID of the message
uint64_t	Node ID of the vnode
uint32_t	User ID
uint32_t	Group ID
uint32_t	Process ID
uint32_t	Reserved
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint64_t	Cache timeout for attributes
uint32_t	Timeout in nanoseconds
uint32_t	Reserved
uint64_t	Inode ID of the node
uint64_t	Size of the inode
uint64_t	Number of Blocks
uint64_t	Access Time of the inode.
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID
uint32_t	Device ID

FUSE Operation Name	*fgetattr*	
Description	The function gets the attributes of the directory/file.	
OPCODE	3	
Corresponding POSIX Call(s)	*lstat()*	
IN Parameters		
const char*	path	Path to the directory/file.
OUT Parameters		
struct stat*	stbuf	Structure file stat which would be filled by the call.
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 3)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
uint64_t	Cache timeout for attributes	
uint32_t	Timeout in nanoseconds	
uint32_t	Reserved	
uint64_t	Inode ID of the node	
uint64_t	Size of the inode	
uint64_t	Number of Blocks	
uint64_t	Access Time of the inode.	

FUSE Operation Name	*fgetattr*
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID
uint32_t	Device ID

FUSE Operation Name	*chmod*	
Description	The function creates a hard link to the file/directory specified.	
OPCODE	4	
Corresponding POSIX Call(s)	*chmod()*	
IN Parameters		
const char*	srcPath	Path to the source file/directory.
mode_t	mode	Mode to be set.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 04)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	

FUSE Operation Name	*chmod*
uint32_t	Process ID
uint32_t	Reserved
uint32_t	Validity Flag
uint32_t	Reserved1
uint64_t	File Handler
uint64_t	File Size
uint64_t	Reserved2
uint64_t	Access Time
uint64_t	Modify Time
uint64_t	Reserved3
uint32_t	Access Time (milliseconds)
uint32_t	Modify Time (milliseconds)
uint32_t	Reserved4
uint32_t	Mode
uint32_t	Reserved5
uint32_t	User ID (uid)
uint32_t	Group ID (gid)
uint32_t	Reserved6
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint64_t	Cache timeout for attributes
uint32_t	Timeout in nanoseconds
uint32_t	Reserved
uint64_t	Inode ID of the node
uint64_t	Size of the inode
uint64_t	Number of Blocks
uint64_t	Access Time of the inode.
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)

FUSE Operation Name	*chmod*
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID
uint32_t	Device ID

FUSE Operation Name	*chown*	
Description	The function creates a hard link to the file/directory specified.	
OPCODE	4	
Corresponding POSIX Call(s)	*lchown()*	
IN Parameters		
const char*	srcPath	Path to the source file/directory.
uid_t	uid	Target User ID
gid_t	gid	Target Group ID
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 04)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	

FUSE Operation Name	*chown*
uint32_t	Validity Flag
uint32_t	Reserved1
uint64_t	File Handler
uint64_t	File Size
uint64_t	Reserved2
uint64_t	Access Time
uint64_t	Modify Time
uint64_t	Reserved3
uint32_t	Access Time (milliseconds)
uint32_t	Modify Time (milliseconds)
uint32_t	Reserved4
uint32_t	Mode
uint32_t	Reserved5
uint32_t	User ID (uid)
uint32_t	Group ID (gid)
uint32_t	Reserved6
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint64_t	Cache timeout for attributes
uint32_t	Timeout in nanoseconds
uint32_t	Reserved
uint64_t	Inode ID of the node
uint64_t	Size of the inode
uint64_t	Number of Blocks
uint64_t	Access Time of the inode.
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)

FUSE Operation Name	*chown*
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID
uint32_t	Device ID

FUSE Operation Name	*truncate*	
Description	The function changes the size of file specified.	
OPCODE	4	
Corresponding POSIX Call(s)	*truncate()*	
IN Parameters		
const char*	srcPath	Path to the source file.
off_t	offset	Size to be set.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 04)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint32_t	Validity Flag	
uint32_t	Reserved1	
uint64_t	File Handler	
uint64_t	File Size	

FUSE Operation Name	*truncate*
uint64_t	Reserved2
uint64_t	Access Time
uint64_t	Modify Time
uint64_t	Reserved3
uint32_t	Access Time (milliseconds)
uint32_t	Modify Time (milliseconds)
uint32_t	Reserved4
uint32_t	Mode
uint32_t	Reserved5
uint32_t	User ID (uid)
uint32_t	Group ID (gid)
uint32_t	Reserved6
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint64_t	Cache timeout for attributes
uint32_t	Timeout in nanoseconds
uint32_t	Reserved
uint64_t	Inode ID of the node
uint64_t	Size of the inode
uint64_t	Number of Blocks
uint64_t	Access Time of the inode.
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID

FUSE Operation Name	*truncate*
uint32_t	Device ID

FUSE Operation Name	*ftruncate*	
Description	The function changes the size of file specified.	
OPCODE	4	
Corresponding POSIX Call(s)	*ftruncate()*	
IN Parameters		
const char*	srcPath	Path to the source file.
off_t	offset	Size to be set.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 04)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint32_t	Validity Flag	
uint32_t	Reserved1	
uint64_t	File Handler	
uint64_t	File Size	
uint64_t	Reserved2	
uint64_t	Access Time	
uint64_t	Modify Time	
uint64_t	Reserved3	

FUSE Operation Name	*ftruncate*
uint32_t	Access Time (milliseconds)
uint32_t	Modify Time (milliseconds)
uint32_t	Reserved4
uint32_t	Mode
uint32_t	Reserved5
uint32_t	User ID (uid)
uint32_t	Group ID (gid)
uint32_t	Reserved6
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint64_t	Cache timeout for attributes
uint32_t	Timeout in nanoseconds
uint32_t	Reserved
uint64_t	Inode ID of the node
uint64_t	Size of the inode
uint64_t	Number of Blocks
uint64_t	Access Time of the inode.
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID
uint32_t	Device ID

FUSE Operation Name	*utime*	
Description	The function changes the size of file specified.	
OPCODE	4	
Corresponding POSIX Call(s)	*utime()*	
IN Parameters		
const char*	srcPath	Path to the source file.
struct utimbuf	buf	Time Stamp to be set to the file.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 04)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint32_t	Validity Flag	
uint32_t	Reserved1	
uint64_t	File Handler	
uint64_t	File Size	
uint64_t	Reserved2	
uint64_t	Access Time	
uint64_t	Modify Time	
uint64_t	Reserved3	
uint32_t	Access Time (milliseconds)	
uint32_t	Modify Time (milliseconds)	
uint32_t	Reserved4	
uint32_t	Mode	
uint32_t	Reserved5	

FUSE Operation Name	*utime*
uint32_t	User ID (uid)
uint32_t	Group ID (gid)
uint32_t	Reserved6
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint64_t	Cache timeout for attributes
uint32_t	Timeout in nanoseconds
uint32_t	Reserved
uint64_t	Inode ID of the node
uint64_t	Size of the inode
uint64_t	Number of Blocks
uint64_t	Access Time of the inode.
uint64_t	Time when inode was modified last.
uint64_t	Creation Time
uint32_t	Access time (seconds)
uint32_t	Modified Time (seconds)
uint32_t	Creation Time (seconds)
uint32_t	Mode
uint32_t	Number of links to inode.
uint32_t	User ID
uint32_t	Group ID
uint32_t	Device ID

FUSE Operation Name	*readlink*
Description	The function reads the information about the linked file.
OPCODE	5
Corresponding POSIX Call(s)	*readlink()*

FUSE Operation Name	*readlink*	
IN Parameters		
const char*	path	Path to the directory/file.
OUT Parameters		
char*	buf	String Buffer
size_t	size	Size of returned buffer
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 5)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
const char*	Link name	

FUSE Operation Name	*symlink*
Description	The function creates a symbolic link to the file/directory specified.
OPCODE	6
Corresponding POSIX Call(s)	*symlink()*
IN Parameters	

FUSE Operation Name	*symlink*	
const char*	srcPath	Path to the source directory.
const char*	destPath	Path to the destination directory name.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 06)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
char*	Name of the file/ directory to be linked	
char*	Name of the link.	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
uint64_t	Node Id of the node	
uint64_t	Inode generation: nodeid:gen	
uint64_t	Cache timeout for the name	
uint64_t	Cache timeout for the attributes	
uint32_t	Cache timeout of rname (seconds)	
uint32_t	Cache timeout of attributes (seconds)	

FUSE Operation Name	*mknod*
Description	The function creates the file with the defined mode.
OPCODE	8

FUSE Operation Name	*mknod*	
Corresponding POSIX Call(s)	*mknod()*	
IN Parameters		
const char*	path	Path to the directory/file.
mode_t	mode	File creation mode
dev_t	rdev	Device Descriptor
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 8)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
char*	Name of the node to be created	
mod_t	Creation Mode	
dev_t	Device Descriptor	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
uint64_t	Node Id of the node	
uint64_t	Inode generation: nodeid:gen	
uint64_t	Cache timeout for the name	
uint64_t	Cache timeout for the attributes	
uint32_t	Cache timeout of rname (seconds)	
uint32_t	Cache timeout of attributes (seconds)	

FUSE Operation Name	*mknod*

FUSE Operation Name	*mkdir*	
Description	The function creates the Directory with the defined mode.	
OPCODE	9	
Corresponding POSIX Call(s)	*mkdir()*	
IN Parameters		
const char*	path	Path to the directory.
mode_t	mode	File creation mode
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 9)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
char*	Name of the node to be created	
mod_t	Creation Mode	
uint32_t	Padding	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
uint64_t	Node Id of the node	

FUSE Operation Name	*mkdir*
uint64_t	Inode generation: nodeid:gen
uint64_t	Cache timeout for the name
uint64_t	Cache timeout for the attributes
uint32_t	Cache timeout of rname (seconds)
uint32_t	Cache timeout of attributes (seconds)

FUSE Operation Name	*unlink*	
Description	The function removes a link to the file or directory.	
OPCODE	10	
Corresponding POSIX Call(s)	*unlink()*	
IN Parameters		
const char*	path	Path to the directory.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 10)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
char*	Parth of the node to be unlinked	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	

FUSE Operation Name	*unlink*
uint64_t	Unique ID of the message

FUSE Operation Name	*rmdir*	
Description	The function removes the Directory specified.	
OPCODE	11	
Corresponding POSIX Call(s)	*rmdir()*	
IN Parameters		
const char*	path	Path to the directory.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 11)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
char*	Path of the node to be removed	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	

FUSE Operation Name	*rename*	
Description	The function creates a symbolic link to the file/directory specified.	
OPCODE	12	
Corresponding POSIX Call(s)	*rename()*	
IN Parameters		
const char*	srcPath	Path to the source file/directory.
const char*	destPath	Path to the destination file/directory.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 06)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint64_t	Directory ID	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	

FUSE Operation Name	*link*
Description	The function creates a hard link to the file/directory specified.

FUSE Operation Name	*link*	
OPCODE	13	
Corresponding POSIX Call(s)	*link()*	
IN Parameters		
const char*	srcPath	Path to the source file/directory.
const char*	destPath	Path to the destination file/directory.
OUT Parameters		
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 13)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint64_t	Node ID of source directory	
char*	Name of the target file/ directory.	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
uint64_t	Node Id of the node	
uint64_t	Inode generation: nodeid:gen	
uint64_t	Cache timeout for the name	
uint64_t	Cache timeout for the attributes	
uint32_t	Cache timeout of rname (seconds)	
uint32_t	Cache timeout of attributes (seconds)	

FUSE Operation Name	*open*	
Description	The function changes the size of file specified.	
OPCODE	14	
Corresponding POSIX Call(s)	*open()*	
IN Parameters		
const char*	srcPath	Path to the target file.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
OUT Parameters		
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 04)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint32_t	Flags	
uint32_t	Mode	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	
uint32_t	Error Code	
uint64_t	Unique ID of the message	
uint64_t	File Handler	
uint32_t	Open Flags	
uint32_t	Reserved	

FUSE Operation Name	*read*	
Description	The function changes the size of file specified.	
OPCODE	15	
Corresponding POSIX Call(s)	*open()* *read()*	
IN Parameters		
const char*	srcPath	Path to the target file.
char*	buf	Buffer to fetch the data.
size_t	size	Size of the buffer.
off_t	offset	Position in file from where the data has to be fetched.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
OUT Parameters		
char*	buf	Buffer filled with data.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 15)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint64_t	File Handler	
uint64_t	Offset from where the data should be fetched.	
uint32_t	Size of the data to be fetched.	
uint32_t	Reserved1	
Out Message	Message from FUSE Library to Kernel Module	

FUSE Operation Name	*read*
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
const char*	Buffer

FUSE Operation Name	*write*	
Description	The function changes the size of file specified.	
OPCODE	16	
Corresponding POSIX Call(s)	*open()* *seek()* *write()*	
IN Parameters		
const char*	srcPath	Path to the target file.
const char*	buf	Buffer to be written.
size_t	size	Size of the buffer.
off_t	offset	Position in file from where the data has to be fetched.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
OUT Parameters		
char*	buf	Buffer filled with data.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 16)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	

FUSE Operation Name	*write*
uint32_t	Reserved
uint64_t	File Handler
uint64_t	Offset from where the data should be fetched.
uint32_t	Size of the data to be fetched.
uint32_t	Flags for writing
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint32_t	Size of the buffer written
uint32_t	Reserved

FUSE Operation Name	*release*	
Description	The function releases/closes the directory specified.	
OPCODE	18	
Corresponding POSIX Call(s)	*close()*	
IN Parameters		
const char*	srcPath	Path to the target file.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
OUT Parameters		
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 18)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	

FUSE Operation Name	*release*
uint32_t	User ID
uint32_t	Group ID
uint32_t	Process ID
uint32_t	Reserved
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message

FUSE Operation Name	*init*
Description	The function initializes the file system.
OPCODE	26
Corresponding POSIX Call(s)	
IN Parameters	
OUT Parameters	
In Message	Message to FUSE Library
uint32_t	Length of message
uint32_t	Opcode of the function (value = 26)
uint64_t	Unique ID of the message
uint64_t	Node ID of the vnode
uint32_t	User ID
uint32_t	Group ID

FUSE Operation Name	*init*
uint32_t	Process ID
uint32_t	Reserved
uint32_t	Kernel Version number (please put your Kernel Driver's version)
uint32_t	Kernel Minor Version number (please put your Kernel Driver's minor version)
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint32_t	Kernel Version number (FUSE Lib 2.5.3 returns value 7)
uint32_t	Kernel Minor Version number (FUSE Lib 2.5.3 returns value 5)
uint32_t	Array of reserved numbers [3]
uint32_t	Max. buffer to write

FUSE Operation Name	*opendir*	
Description	The function checks the access permissions for directory and opens the directory specified for reading.	
OPCODE	27	
Corresponding POSIX Call(s)		
IN Parameters		
const char*	srcPath	Path to the target file.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
OUT Parameters		
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
In Message	Message to FUSE Library	

FUSE Operation Name	*opendir*
uint32_t	Length of message
uint32_t	Opcode of the function (value = 27)
uint64_t	Unique ID of the message
uint64_t	Node ID of the vnode
uint32_t	User ID
uint32_t	Group ID
uint32_t	Process ID
uint32_t	Reserved
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message
uint64_t	File Handler
uint32_t	Open Flags
uint32_t	Reserved

FUSE Operation Name	*readdir*	
Description	The function gets the listing of a directory.	
OPCODE	28	
Corresponding POSIX Call(s)	*Opendir()* *readdir()* *closedir()*	
IN Parameters		
const char*	path	Path to the directory/file.
void*	buf	Buffer
fuse_fill_dir_t	Fill Directory Function pointer	
off_t	Offset	
OUT Parameters		
fuse_file_info*	fi	File Info Structure.

FUSE Operation Name	*readdir*
In Message	Message to FUSE Library
uint32_t	Length of message
uint32_t	Opcode of the function (value = 28)
uint64_t	Unique ID of the message
uint64_t	Node ID of the vnode
uint32_t	User ID
uint32_t	Group ID
uint32_t	Process ID
uint32_t	Reserved
uint64_t	File Handler
uint64_t	Offset
uint32_t	Size
uint32_t	Reserved1 (padding)
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message

const char*	Character Buffer
	The Buffer is a collection of a variant of "dirent64" structure defined in /usr/include/sys/dirent.h.
	Here is the structure received from library:
	struct {
	ino64t_t d_ino; /* inode number of entry */
	ino64t_t d_off; /* offset of disk directory entry */
	unsigned short d_namelen; /* length of the file name*/
	unsigned short d_rectype; /* type of record – this is to be ignored for solaris */
	char d_name[0]; /* name of the file – this is variable length */
	}

FUSE Operation Name	*readdir*
	Please note that the whole structure is 8-byte aligned (i.e padding is done to make the size in multiple of 8-byte).

FUSE Operation Name	*releasedir*	
Description	The function releases/closes the directory specified.	
OPCODE	29	
Corresponding POSIX Call(s)	*closedir()*	
IN Parameters		
const char*	srcPath	Path to the target file.
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
OUT Parameters		
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 29)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint64_t	File Handler	
uint32_t	Flags	
uint32_t	Reserved1	
Out Message	Message from FUSE Library to Kernel Module	
uint32_t	Length of message	

FUSE Operation Name	*releasedir*
uint32_t	Error Code
uint64_t	Unique ID of the message

FUSE Operation Name	*fsyncdir*	
Description	The function flushes the user data based on the input datasync parameter.	
OPCODE	30	
Corresponding POSIX Call(s)		
IN Parameters		
const char*	srcPath	Path to the target file.
int	isDataSync	Data Synchronization Flag
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
OUT Parameters		
struct fuse_file_info*	fi	Proprietary structure of FUSE Library.
In Message	Message to FUSE Library	
uint32_t	Length of message	
uint32_t	Opcode of the function (value = 30)	
uint64_t	Unique ID of the message	
uint64_t	Node ID of the vnode	
uint32_t	User ID	
uint32_t	Group ID	
uint32_t	Process ID	
uint32_t	Reserved	
uint64_t	File Handler	
uint32_t	Synchronization Flags	
uint32_t	Reserved1	

FUSE Operation Name	*fsyncdir*
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message

FUSE Operation Name	*access*
Description	This function checks access permissions to a file/directory specified.
OPCODE	34
Corresponding POSIX Call(s)	*access()*
IN Parameters	
OUT Parameters	
In Message	Message to FUSE Library
uint32_t	Length of message
uint32_t	Opcode of the function (value = 34)
uint64_t	Unique ID of the message
uint64_t	Node ID of the vnode
uint32_t	User ID
uint32_t	Group ID
uint32_t	Process ID
uint32_t	Reserved
uint32_t	Mask
uint32_t	Reserved1
Out Message	Message from FUSE Library to Kernel Module

FUSE Operation Name	*access*
uint32_t	Length of message
uint32_t	Error Code
uint64_t	Unique ID of the message

FUSE Operation Name	*create*
Description	This function creates and opens a file with specified mode.
OPCODE	35
Corresponding POSIX Call(s)	*create()*
IN Parameters	
OUT Parameters	
In Message	Message to FUSE Library
uint32_t	Length of message
uint32_t	Opcode of the function (value = 35)
uint64_t	Unique ID of the message
uint64_t	Node ID of the vnode
uint32_t	User ID
uint32_t	Group ID
uint32_t	Process ID
uint32_t	Reserved
uint32_t	Flags
uint32_t	File creation mode.
Out Message	Message from FUSE Library to Kernel Module
uint32_t	Length of message
uint32_t	Error Code

FUSE Operation Name	*create*
uint64_t	Unique ID of the message