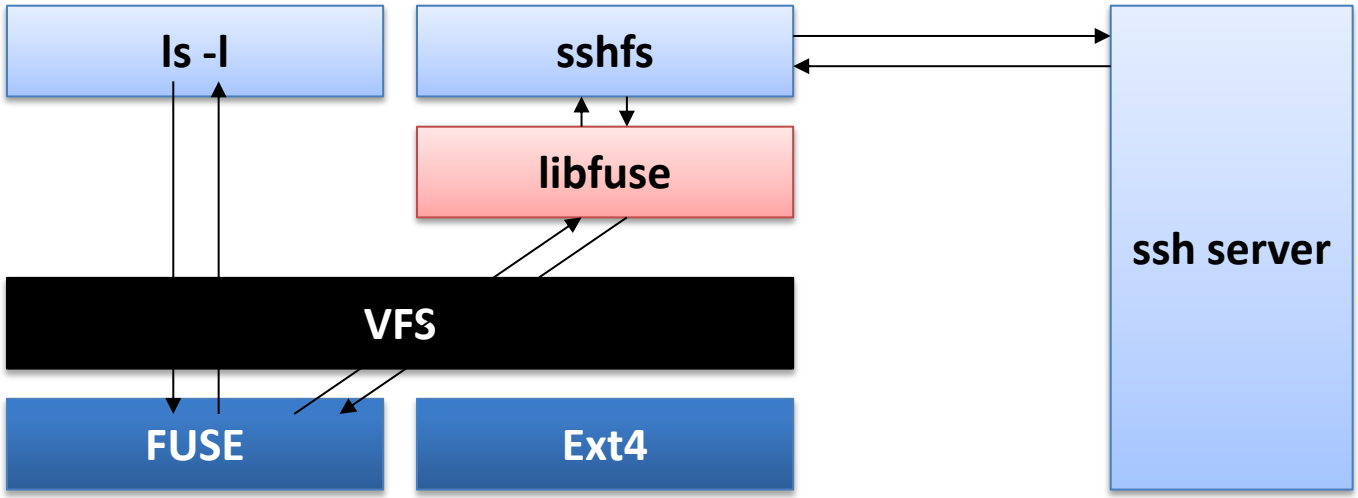
FUSE: File system in user space

Introduction

- Implement simple file system in userspace



- **Compatiblity**
  - Linux, FreeBSD, NetBSD, Mac OS X, OpenSolaris, …
- **Example file systems**
  - Gnome VFS2, unionfs, ftpfs, sshfs, Android sdcard, …

How to use

- **Installation**
  - apt-get install fuse fuse-utils libfuse2 libfuse-dev
  - Source: http://fuse.sourceforge.net/

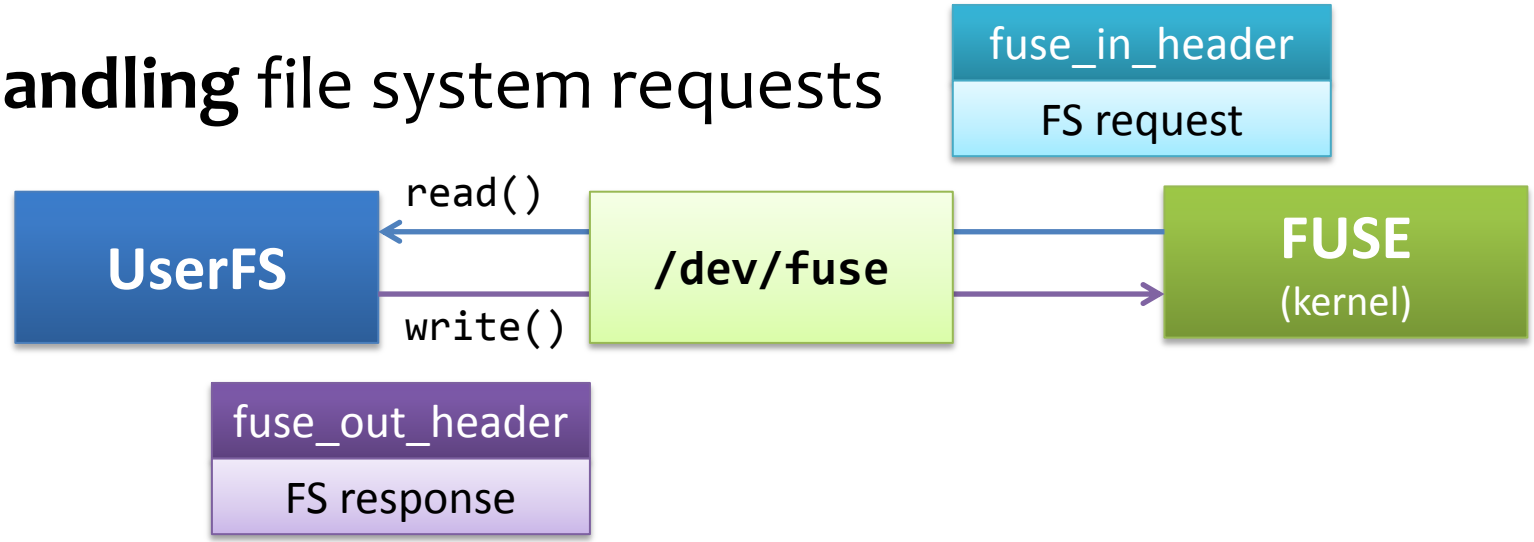
- **Mount / unmount**
  - ***sshfs*** user@host /tmp/fuse
  - ***fusermount*** –u /tmp/fuse

Using SSHFS

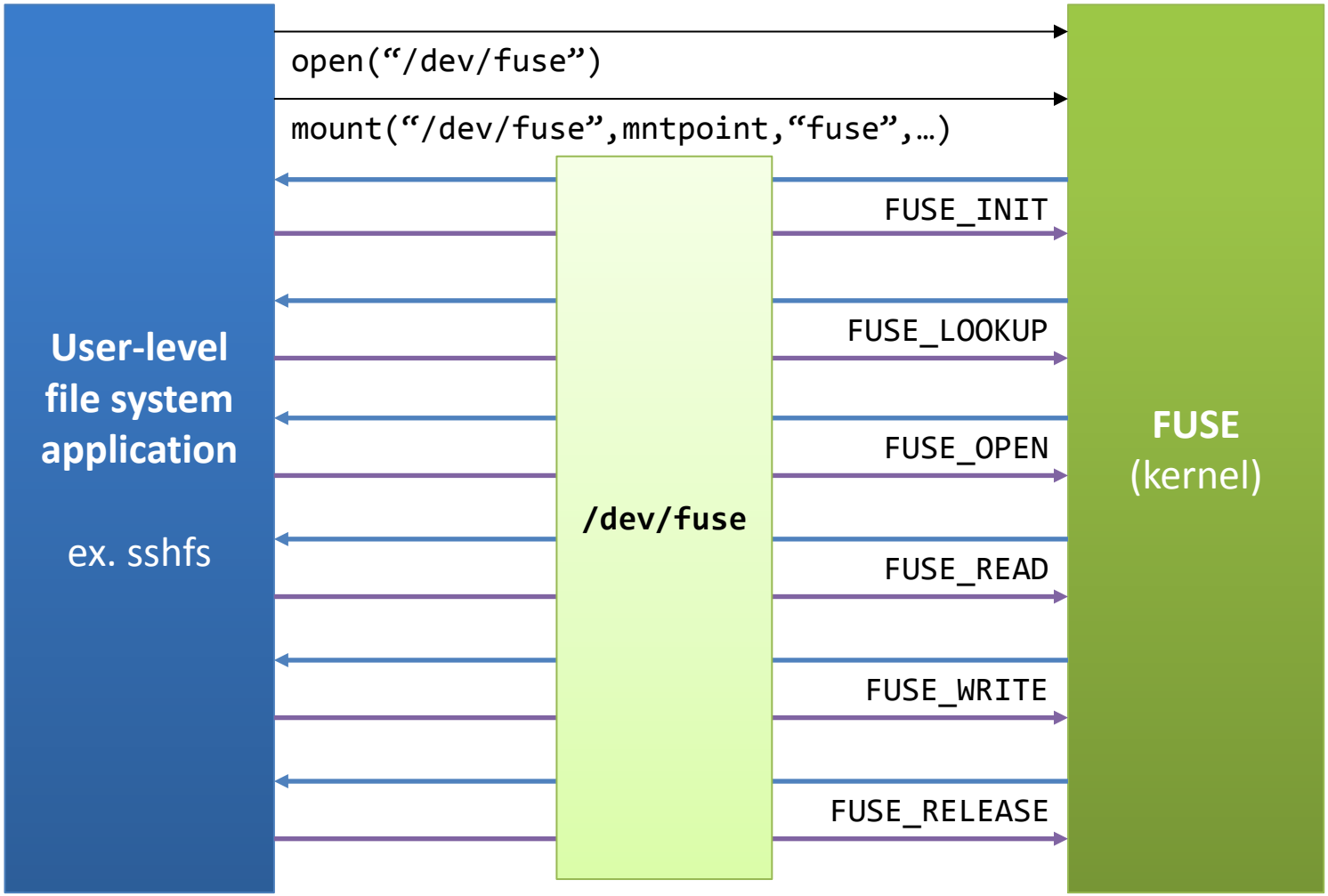
- **Setup ssh server**
  - apt-get install ssh
- **Install** sshfs
  - apt-get install sshfs
- **Mount** sshfs
  - mkdir /home/user/fusemnt
  - sshfs user@localhost:/home/user /home/user/fusemnt
- **Unmount** sshfs
  - fusermount –u /home/user/fusemnt

Low-level FUSE interface

- **FUSE communication channel**
  - */dev/fuse* `fd = open("/dev/fuse", O_RDWR)`
- **Mounting** a file system
  - mount("/dev/fuse", mount_point, "fuse", flags, *opts*)
  - opts: "fd=n,…"
- **Handling** file system requests



FUSE communication example



FUSE message headers

- struct ***fuse_in_header***
  - Describes a file system request
  - enum ***fuse_opcode*** opcode:
    - init
    - open, read, write
    - mknod, mkdir, readlink, symlink, unlink
    - rmdir, rename
    - lookup, forget, getattr, setattr
    - statfs, release, flush
    - opendir, readdir, releasedir
- struct ***fuse_out_header***
  - Return execution result

fuse_in_header	
u32	len
u32	opcode
u64	unique
u64	nodeid
u32	uid
u32	gid
u32	pid
u32	padding

fuse_out_header	
u32	len
s32	error
u64	unique

Directory access

- **FUSE_LOOKUP**
  - Find file "***data***" in directory ***fuse->nodeid***
  - Returns `struct` **`fuse_entry_out`**
- struct ***fuse_open_in***
  - u32 flags
  - Open file identified by `fuse->nodeid`
  - User defines file handle `fh`
- struct ***fuse_release_in***
  - Close file `fh`
  - No return

fuse_entry_out	
u64	nodeid
u64	generation
u64	entry_valid
u64	attr_valid
	fuse_attr

fuse_open_out	
u64	fh
u32	open_flags

fuse_release_in	
u64	fh
u32	flags
u32	release_flags
u64	lock_owner

Read and write

- struct ***fuse_read_in***
  - Next to fuse_in_header
  - Read file (offset, size)
  - Send result to */dev/fuse*
- struct ***fuse_write_in***
  - Identical to *fuse_read_in*
  - File data comes next to *fuse_write_in*
  - Return *fuse_write_out* to */dev/fuse*

fuse_read_in	
u64	fh
u64	offset
u32	size
u32	read_flags
u64	lock_owner
u32	flags
u32	padding

fuse_write_out	
u32	size
u32	padding

Other primitives

- Refer to ***include/fuse_kernel.h***
  - struct fuse_(opcode)_in/out

- **FUSE for Python**
  - http://sourceforge.net/apps/mediawiki/fuse/index.php?title=SimpleFilesystemHowto

- **Other language bindings**
  - C++, Java, C#, Go, Haskell, TCL, Perl, Ocaml, Ruby, Lua, …
  - http://sourceforge.net/apps/mediawiki/fuse/index.php?title=LanguageBindings

libfuse: C binding

- **High-level API**
  - Implements common fuse ⇔ kernel interactions
    - Mount command-line parsing
    - Interpret fuse_in_header and build fuse_out_header
  - Simplifies file system implementation

- **Developing with libfuse**
  - **Include:** <fuse.h>
  - **Compile:** `gcc -Wall `pkg-config fuse --cflags --libs``
  - **Example:** `/usr/share/doc/libfuse-dev/examples/hello.c`
  - **Document:** http://fuse.sourceforge.net/doxygen/index.html

Implementing FS with libfuse

- `struct` **`fuse_operations`**
  - Define and override file system operations
  - Example
    - **`.open`**`(char *path, struct fuse_file_info *fi)`
    - **`.read`**`(char *path,`
      `char *buf, size_t size, off_t offset,`
      `struct fuse_file_info *fi)`
- **`fuse_main`**`(argc, argv, &op, user_data)`
  - Mounts the file system
  - Fetches file system requests and calls defined functions
  - Returns when unmounted: `fusermount -u`