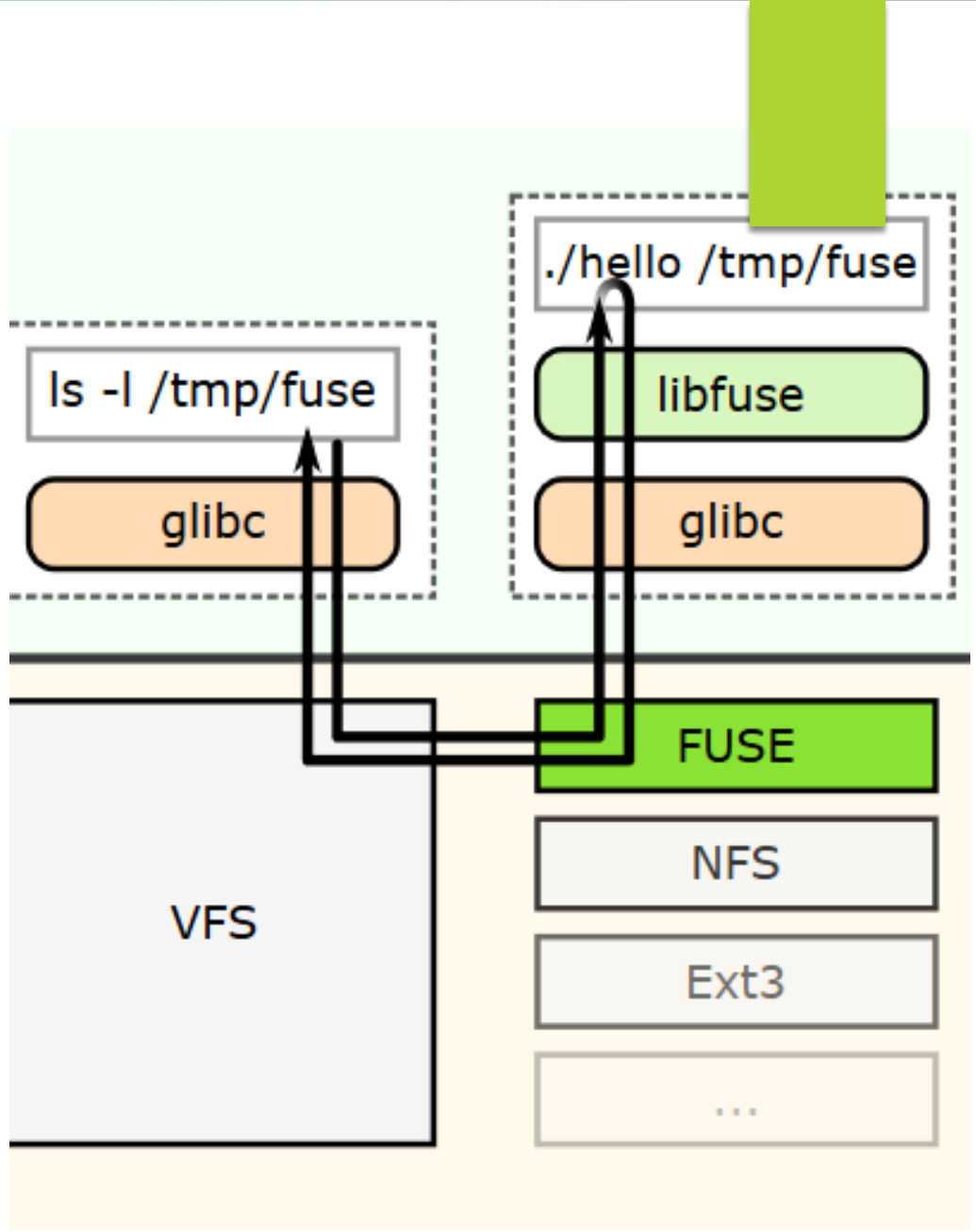
Implementing a file system

JENNICA RAMONES

Research on the FUSE library

- FUSE → Filesystem in Userspace
  - Allows the creation of file systems without coding anything in the kernel space
  - Linux and MacOS
  - Fuse kernel model & libfuse library
  - File system is implemented in userspace while the library ‘bridges’ the code to the actual kernel interfaces



Important Components

- FUSE offers two APIs: a “high-level” and “low-level” API
- fuse_operations structure
  - These pointers will be called by FUSE when something happens on the file system
  - The functions of this structure need to be defined with the pointers of your own implemented functions

```
struct fuse_operations {
        int (*getattr) (const char *, struct stat *);
        int (*readlink) (const char *, char *, size_t);
        int (*getdir) (const char *, fuse_dirh_t,
fuse_dirfil_t);
        int (*mknod) (const char *, mode_t, dev_t);
        int (*mkdir) (const char *, mode_t);
        int (*unlink) (const char *);
        int (*rmdir) (const char *);
        int (*symlink) (const char *, const char *);
        int (*rename) (const char *, const char *);
        int (*link) (const char *, const char *);
        int (*chmod) (const char *, mode_t);
        int (*chown) (const char *, uid_t, gid_t);
        int (*truncate) (const char *, off_t);
        int (*utime) (const char *, struct utimbuf *);
        int (*open) (const char *, struct fuse_file_info *);
        int (*read) (const char *, char *, size_t, off_t,
                     struct fuse_file_info *);
        int (*write) (const char *, const char *, size_t,
off_t,
                      struct fuse_file_info *);
```

My Project

A SIMPLE FILE SYSTEM THAT READS THE CONTENTS OF A DIRECTORY

```c
// gets the attributes of the file, returns 0 on success
static int do_getattr(const char *path, struct stat *st){
  printf("calling getattr()\n");

  // mode specifies its type (file, directory, etc.)
  // and the permission bits of the file
  // nlink specifies the number of hard links
  // size specifies the size of the file in bytes
  if (strcmp(path, "/" == 0)){ // root directory
      st->st_mode = S_IFDIR | 0755;
      st->st_nlink = 2; // two hard links for /. and /..
  }
  else {
    st->st_mode = S_IFREG | 0644;
    st->st_nlink = 1;
    st->st_size = 1024;
  }

    return 0;
}
```

Implementing getattr()

```c
// reads the directory
static int do_readdir(const char *path, void *buffer, fuse_fill_dir_t filler,
off_t offset, struct fuse_file_info *fi){
  printf("reading files\n");
  filler(buffer, ".", NULL, 0);
  filler(buffer, "..", NULL, 0);

  if (strcmp(path, "/") == 0){
    filler(buffer, "file2", NULL, 0);
    filler(buffer, "file100", NULL, 0);
  }
  return 0;
}
```

Implementing readdir()

```
static int do_read(const char *path, char *buffer, size_t size, off_t offset,
struct fuse_file_info *fi){
  char file2Contents[] = "file2's contents";
  char file100Contents[] = "file100's contents";
  char *selectedContents = NULL;

  if (strcmp(path, "/file2") == 0) selectedContents = file2Contents;
  else if (strcmp(path, "/file100") == 0) selectedContents = file100Contents;
  else return -1;

  // copies the contents of the files and returns the number of bytes
  memcpy(buffer, selectedContents + offset, size);
  return strlen(selectedContents) - offset;
}
```

Implementing read()

Debugging

- Couldn’t mount the actual file system
- fuse: missing mountpoint parameter

Resources

- https://github.com/libfuse/libfuse
- http://www.maastaar.net/fuse/linux/filesystem/c/2016/05/21/writing-a-simple-filesystem-using-fuse/