Desenvolvendo Sistemas de Arquivos no FUSE

Nível: Intermediário

Pré-requisitos: familiaridade com a linguagem C
conceitos de sistemas de arquivos

Desenvolvendo Sistemas de Arquivos no FUSE

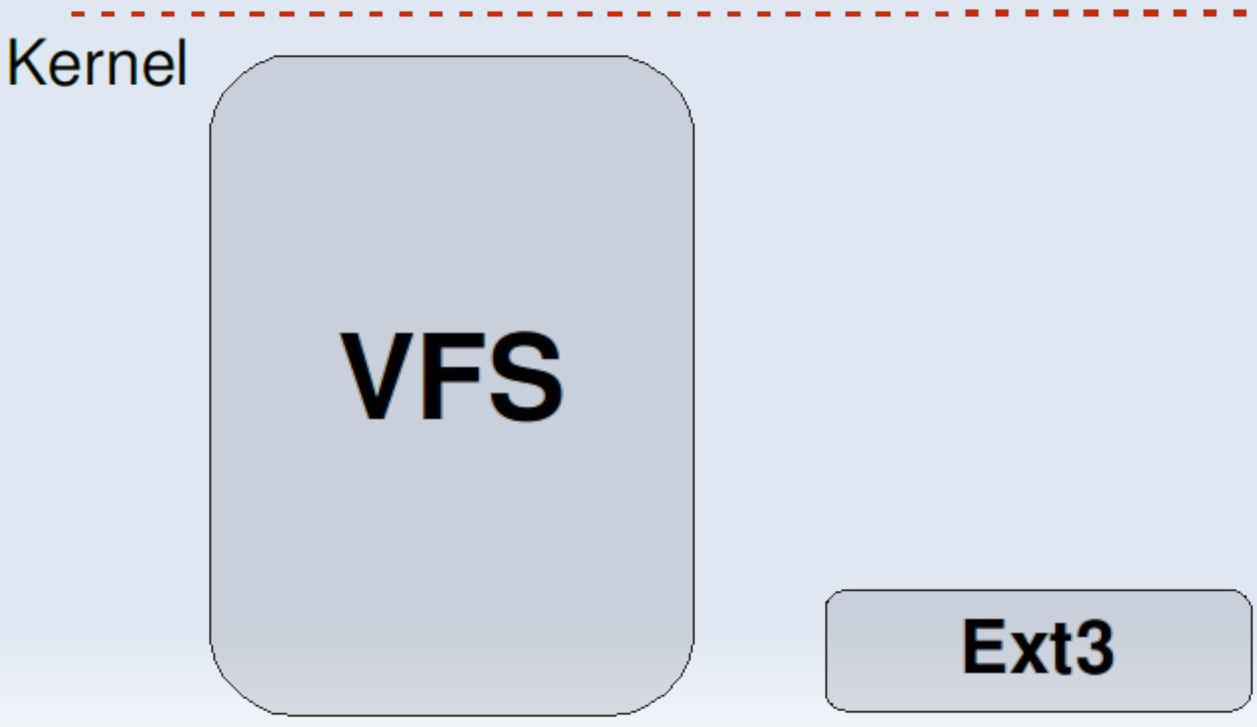
http://fuse.sourceforge.net/

Miklos Szeredi

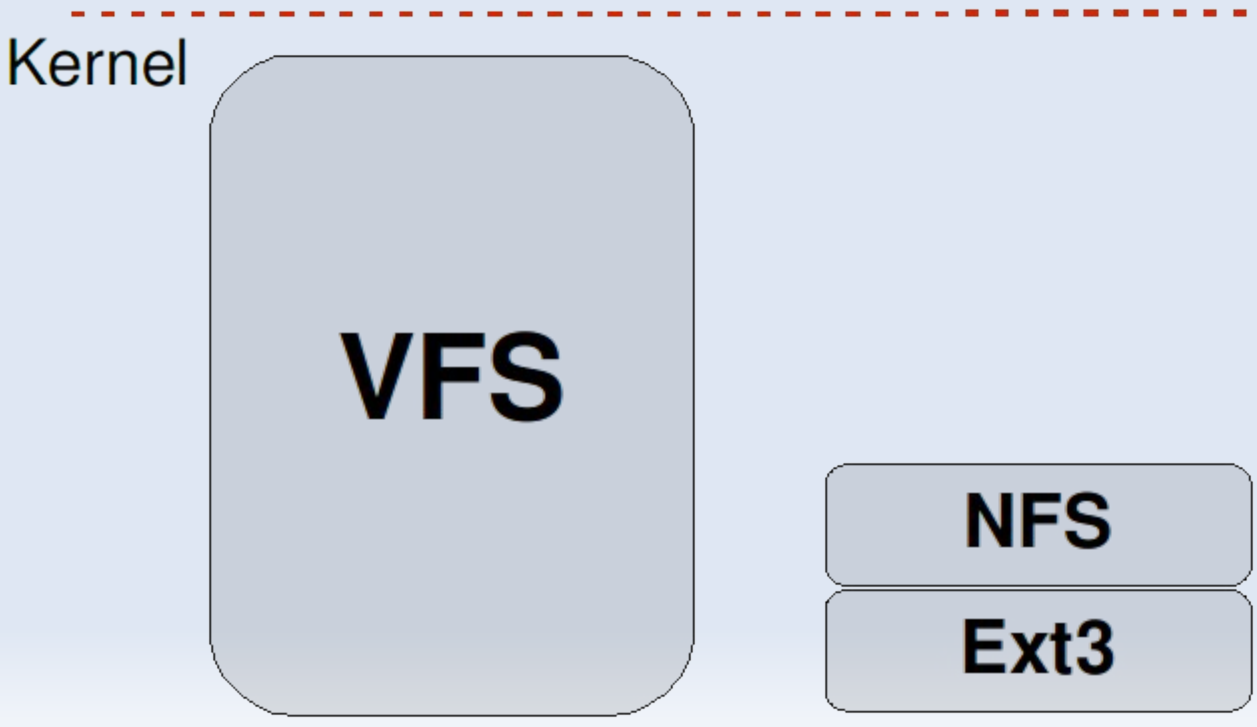
Luis Otávio de Colla Furquim (luisfurquim@gmail.com)

Desenvolvendo Sistemas de Arquivos no FUSE

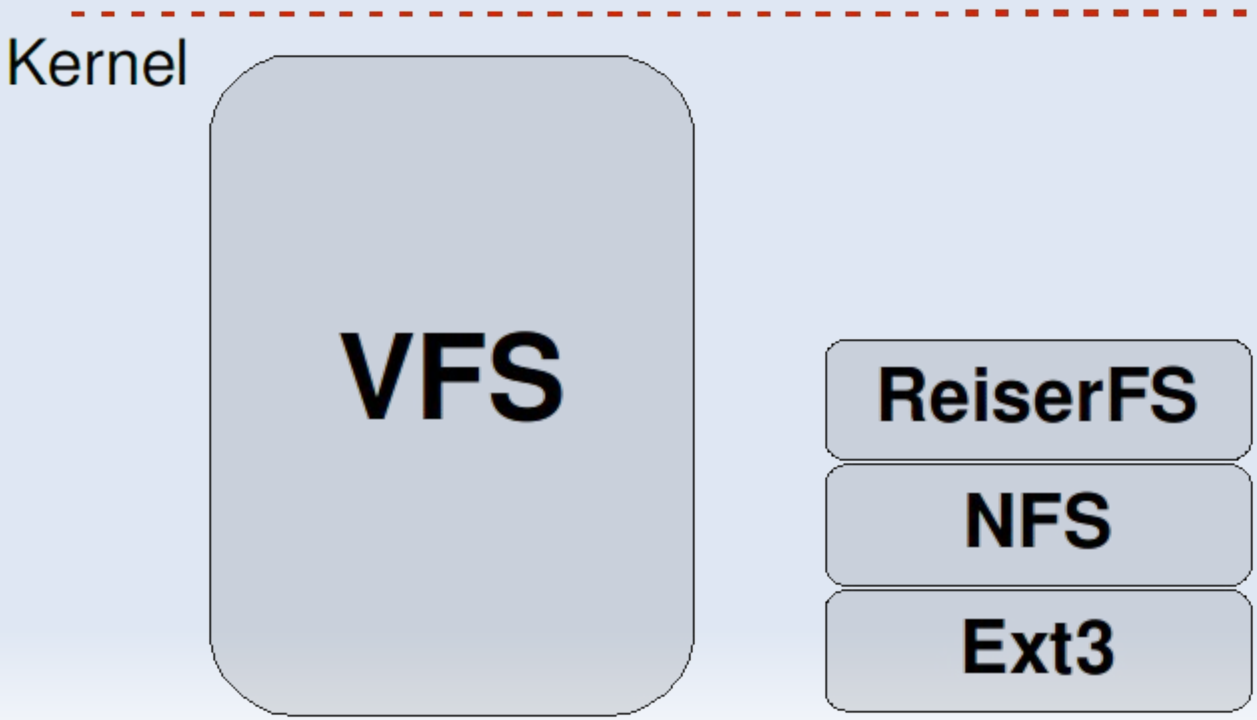
Desenvolvendo Sistemas de Arquivos no FUSE



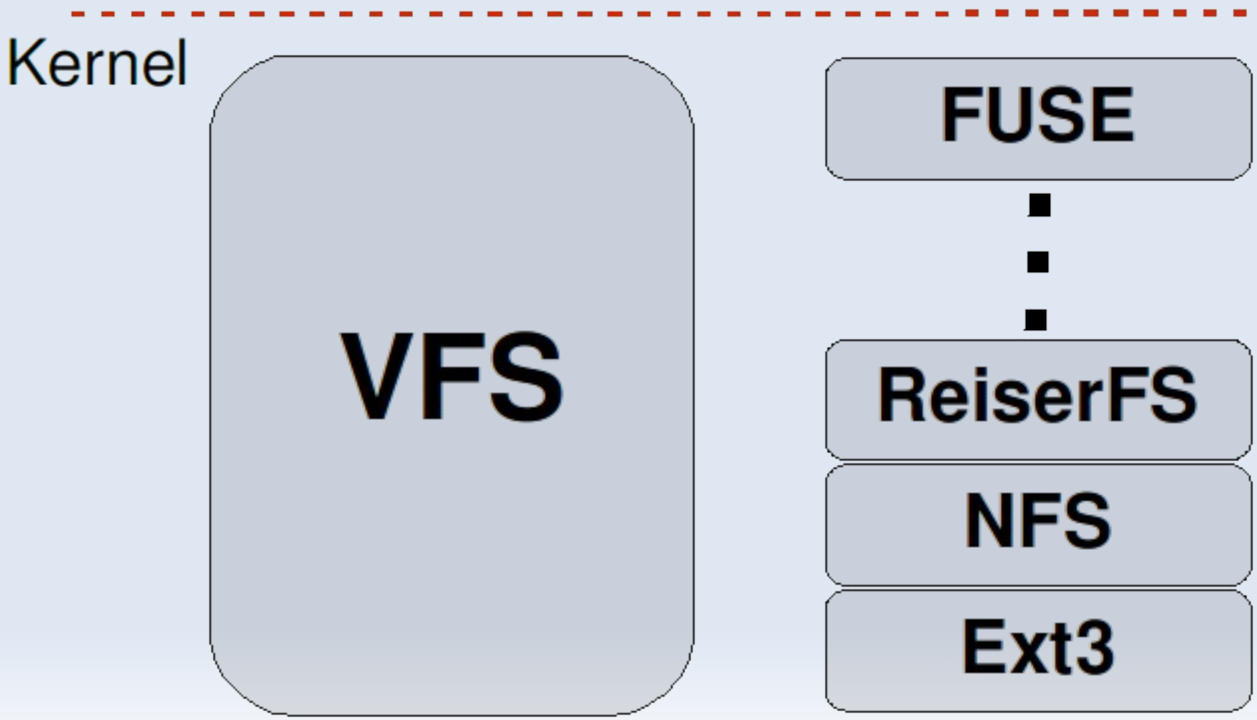
Desenvolvendo Sistemas de Arquivos no FUSE



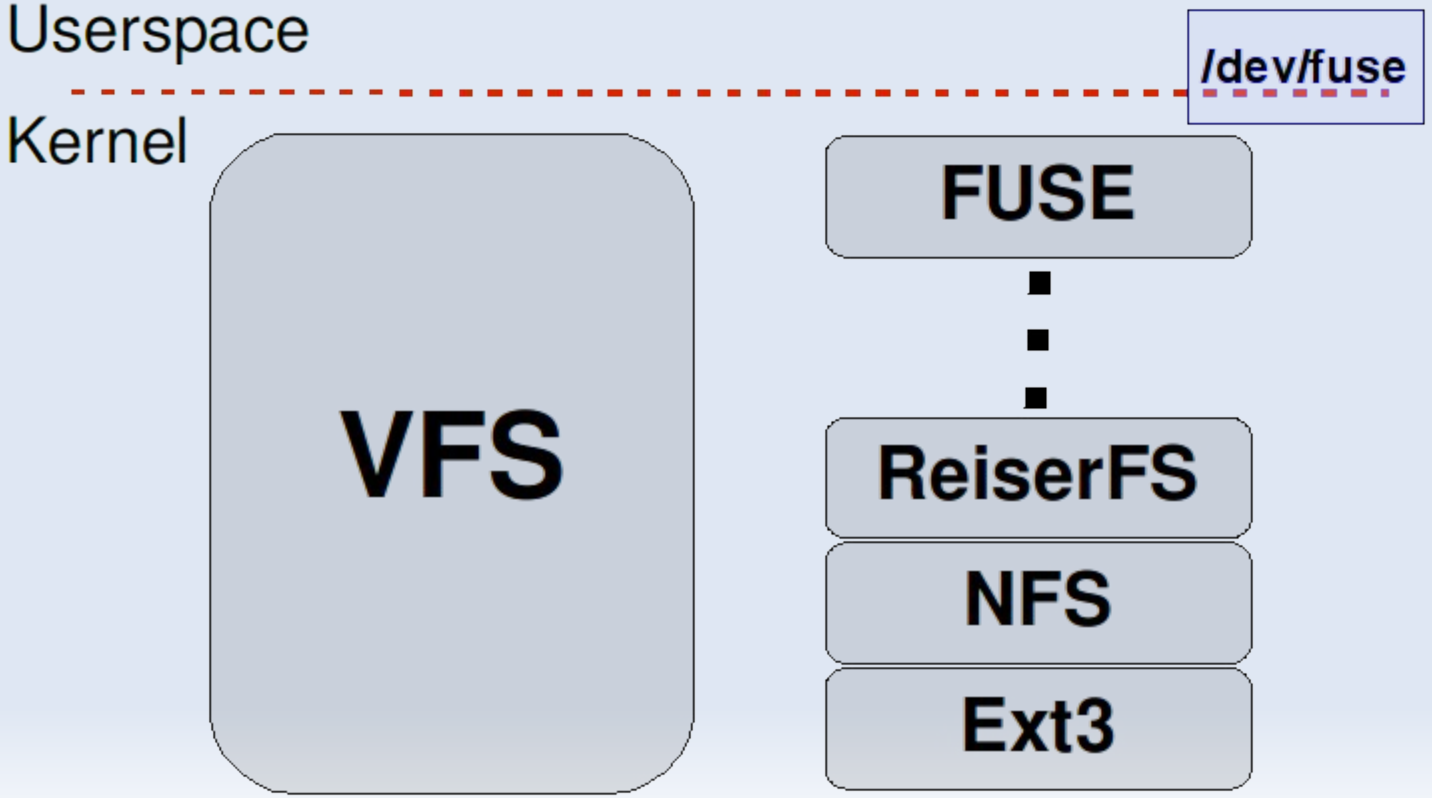
Desenvolvendo Sistemas de Arquivos no FUSE



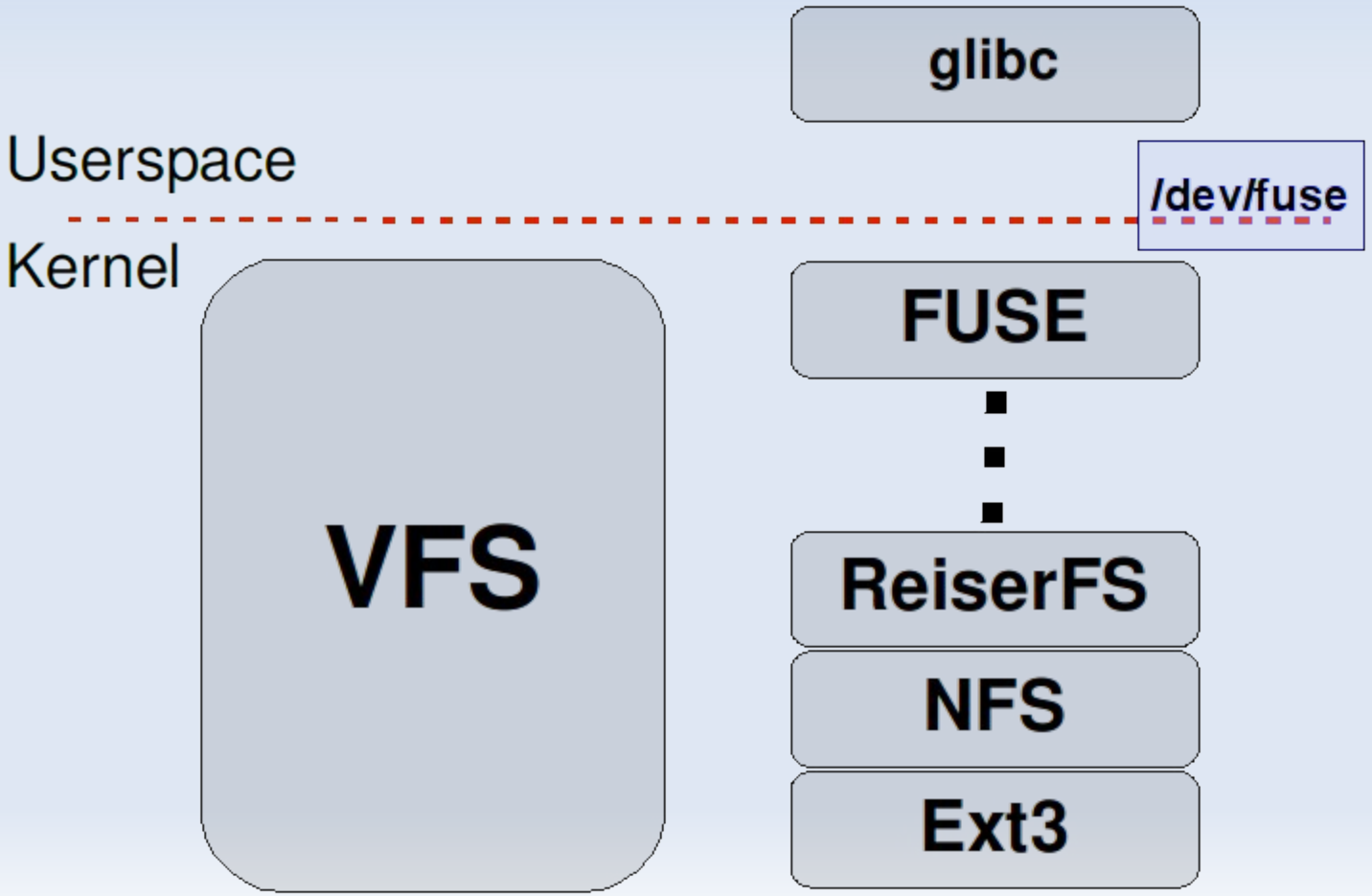
Desenvolvendo Sistemas de Arquivos no FUSE



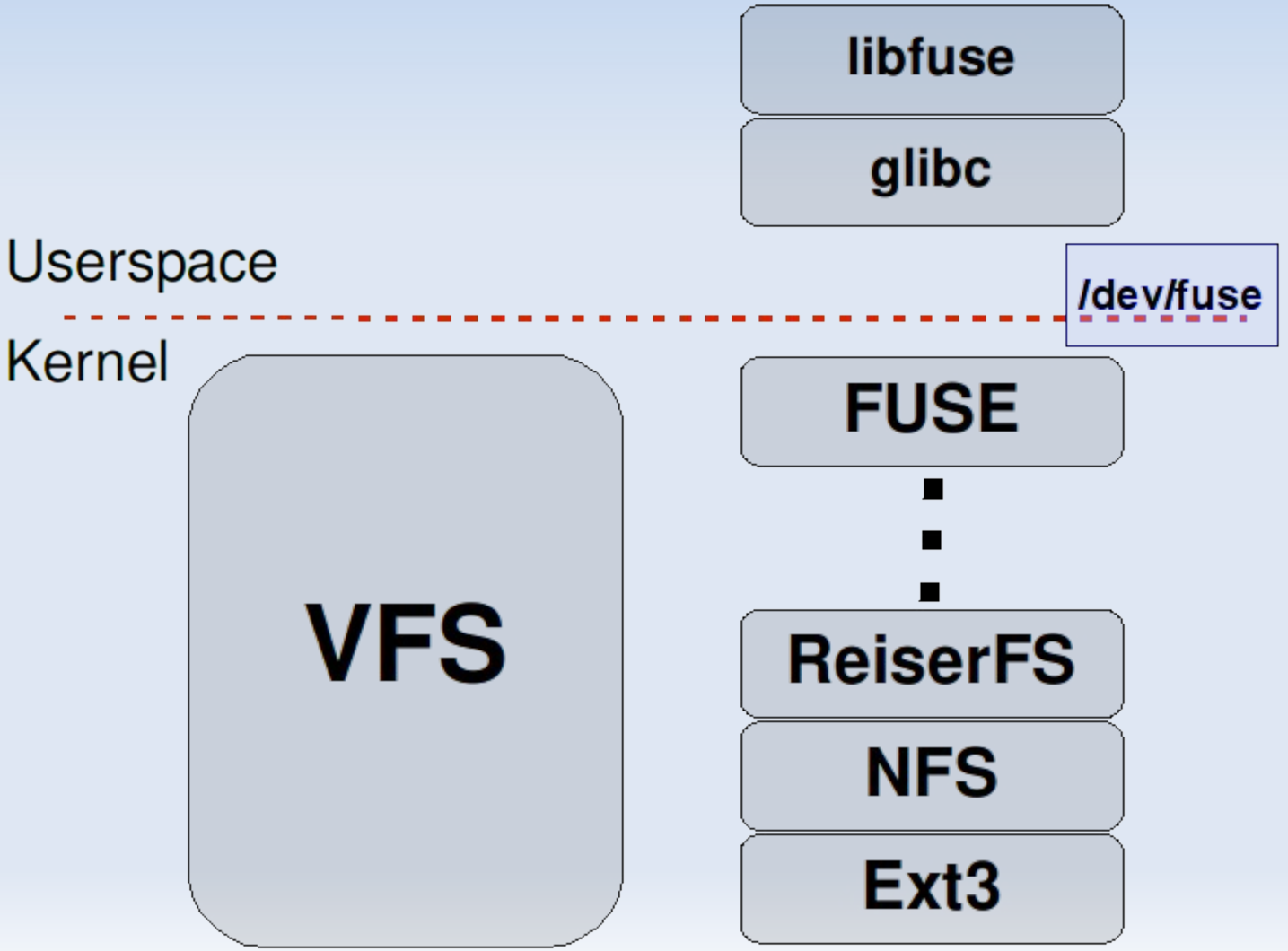
Desenvolvendo Sistemas de Arquivos no FUSE



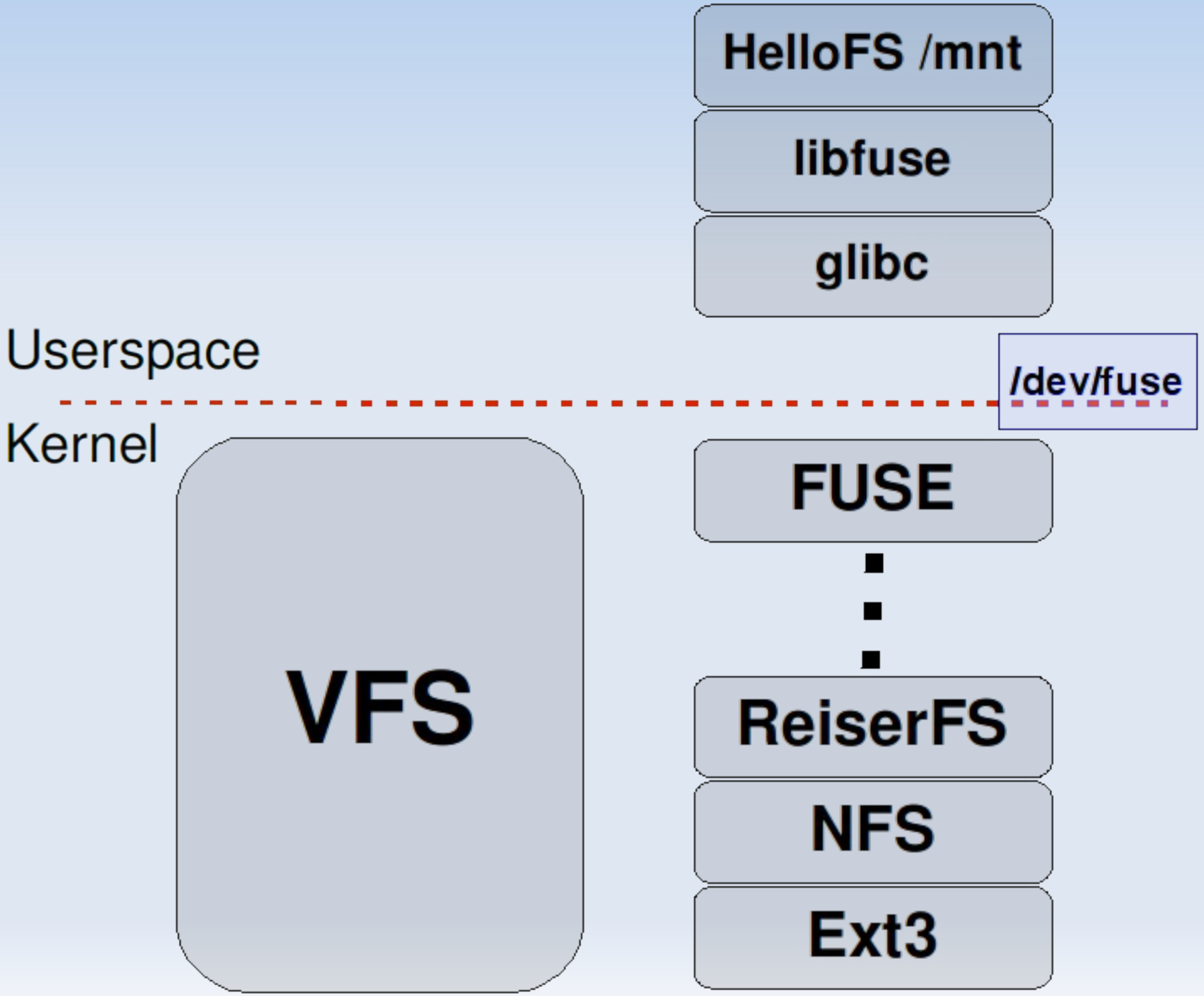
Desenvolvendo Sistemas de Arquivos no FUSE



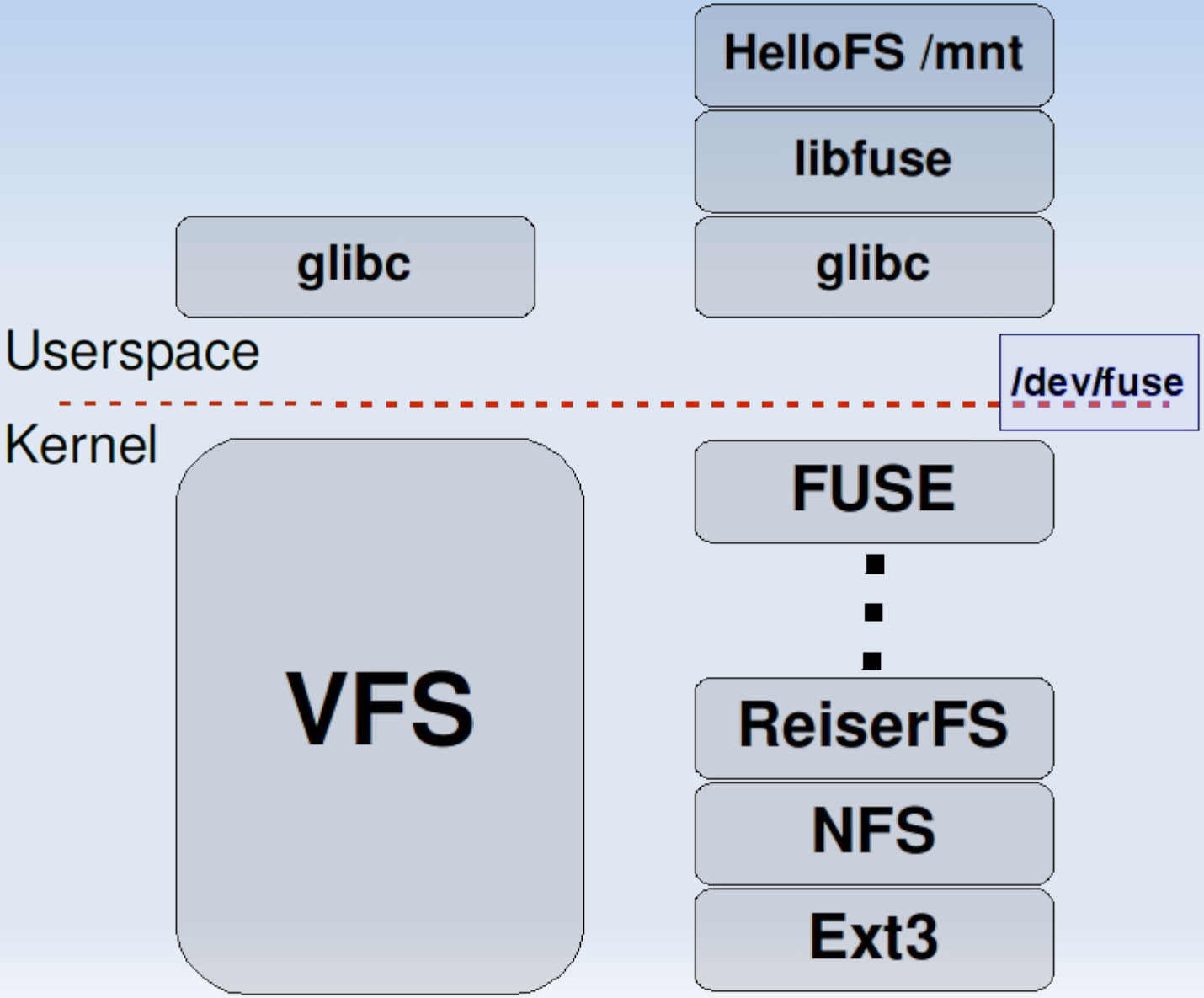
Desenvolvendo Sistemas de Arquivos no FUSE



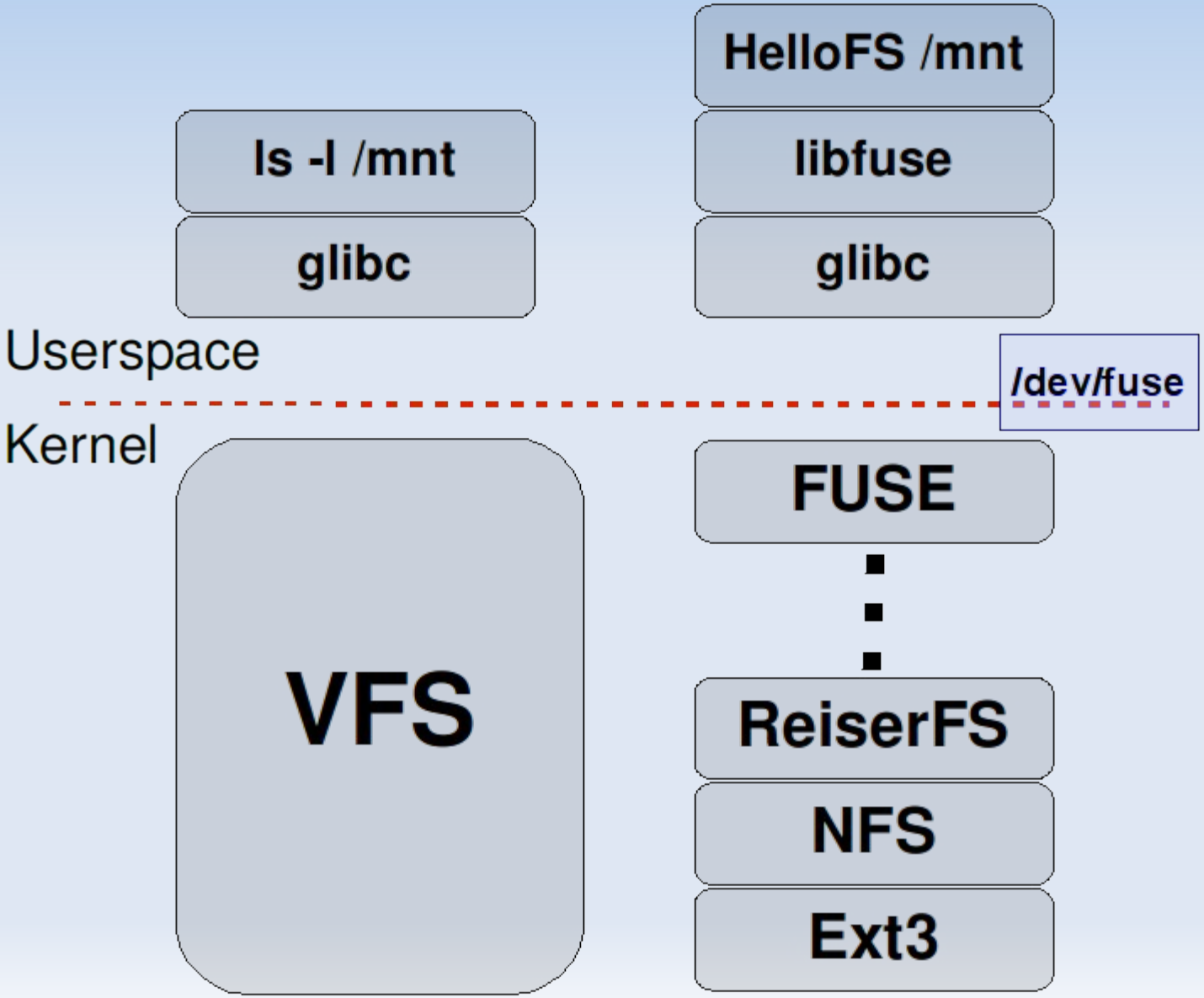
Desenvolvendo Sistemas de Arquivos no FUSE



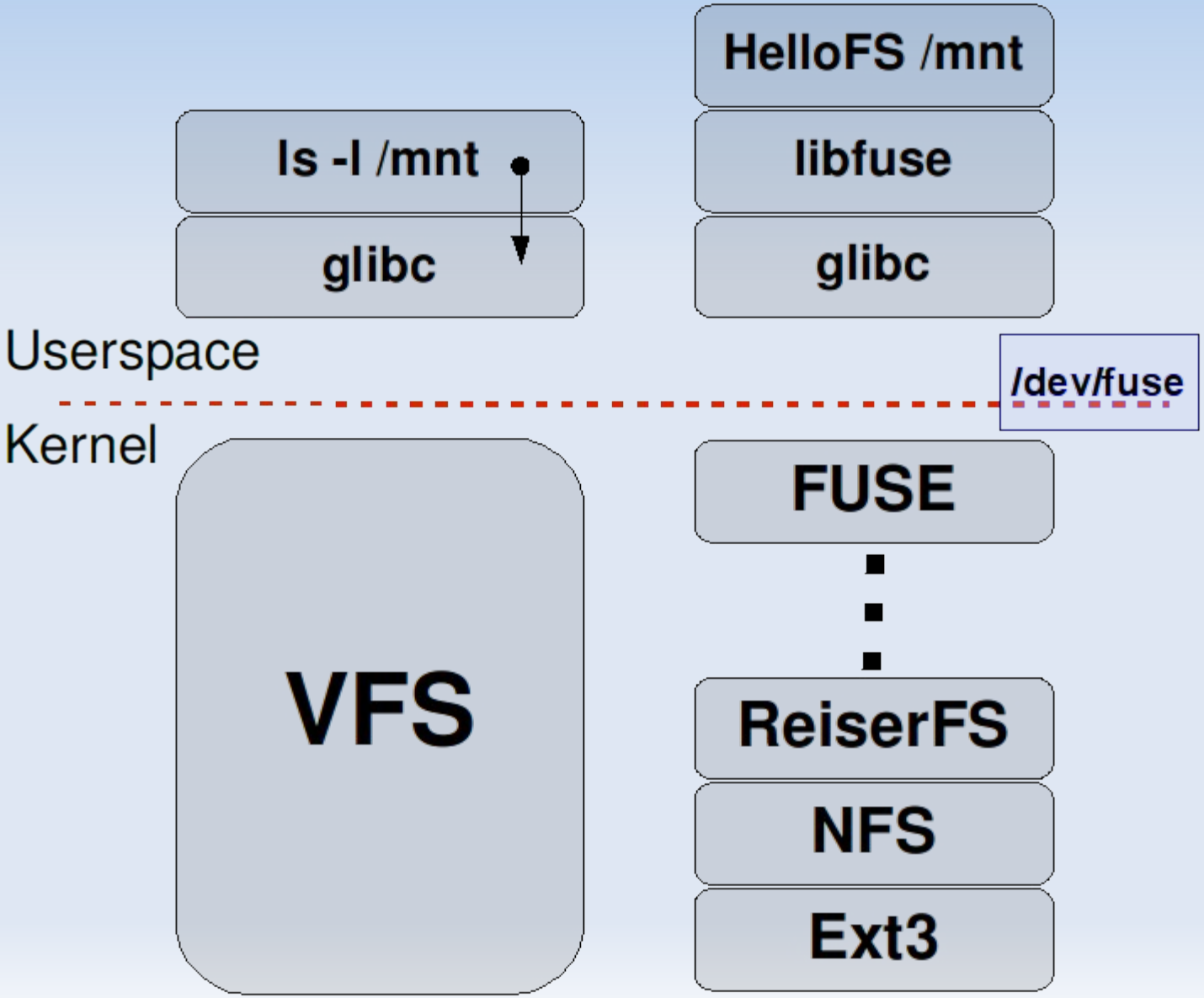
Desenvolvendo Sistemas de Arquivos no FUSE



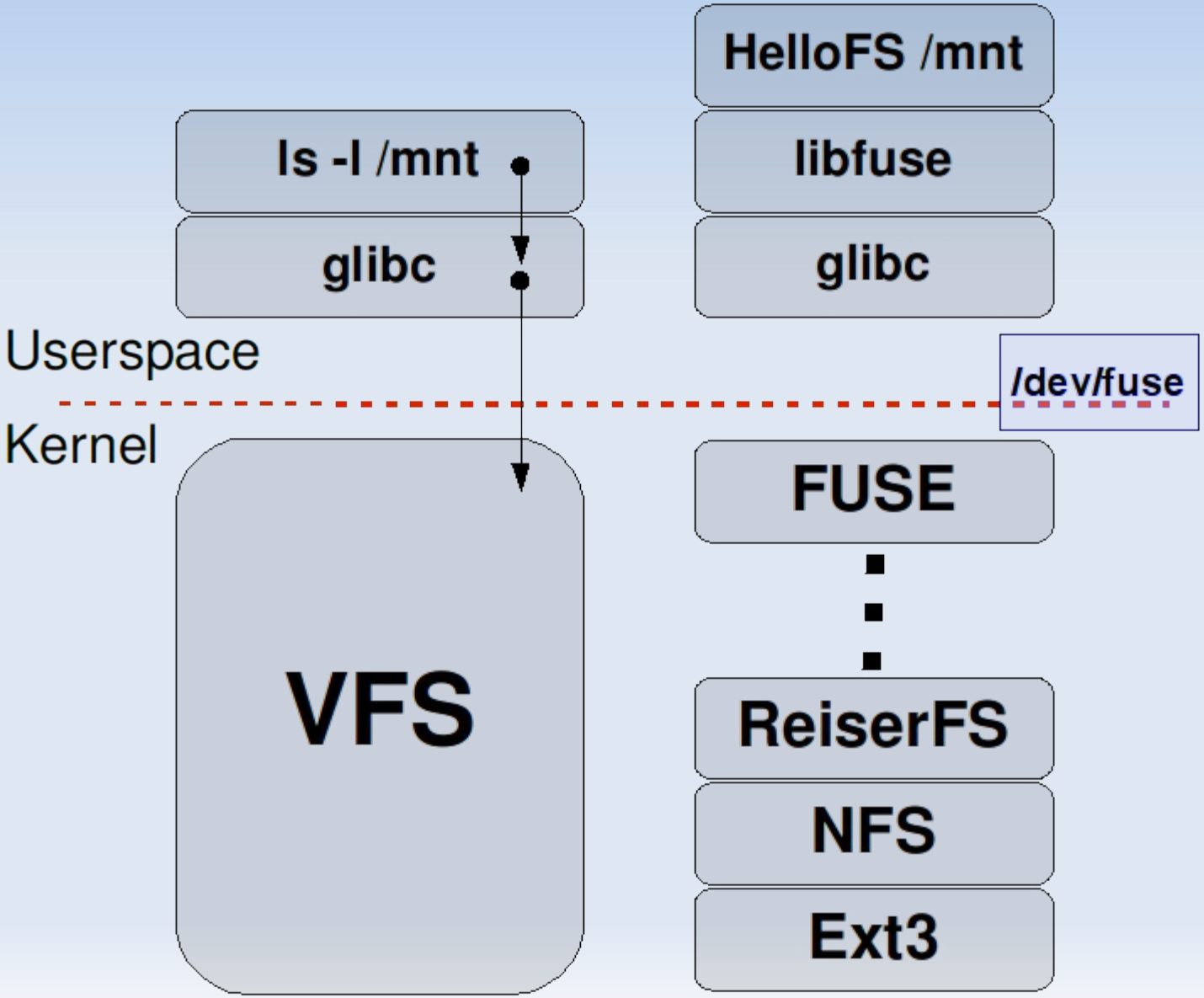
Desenvolvendo Sistemas de Arquivos no FUSE



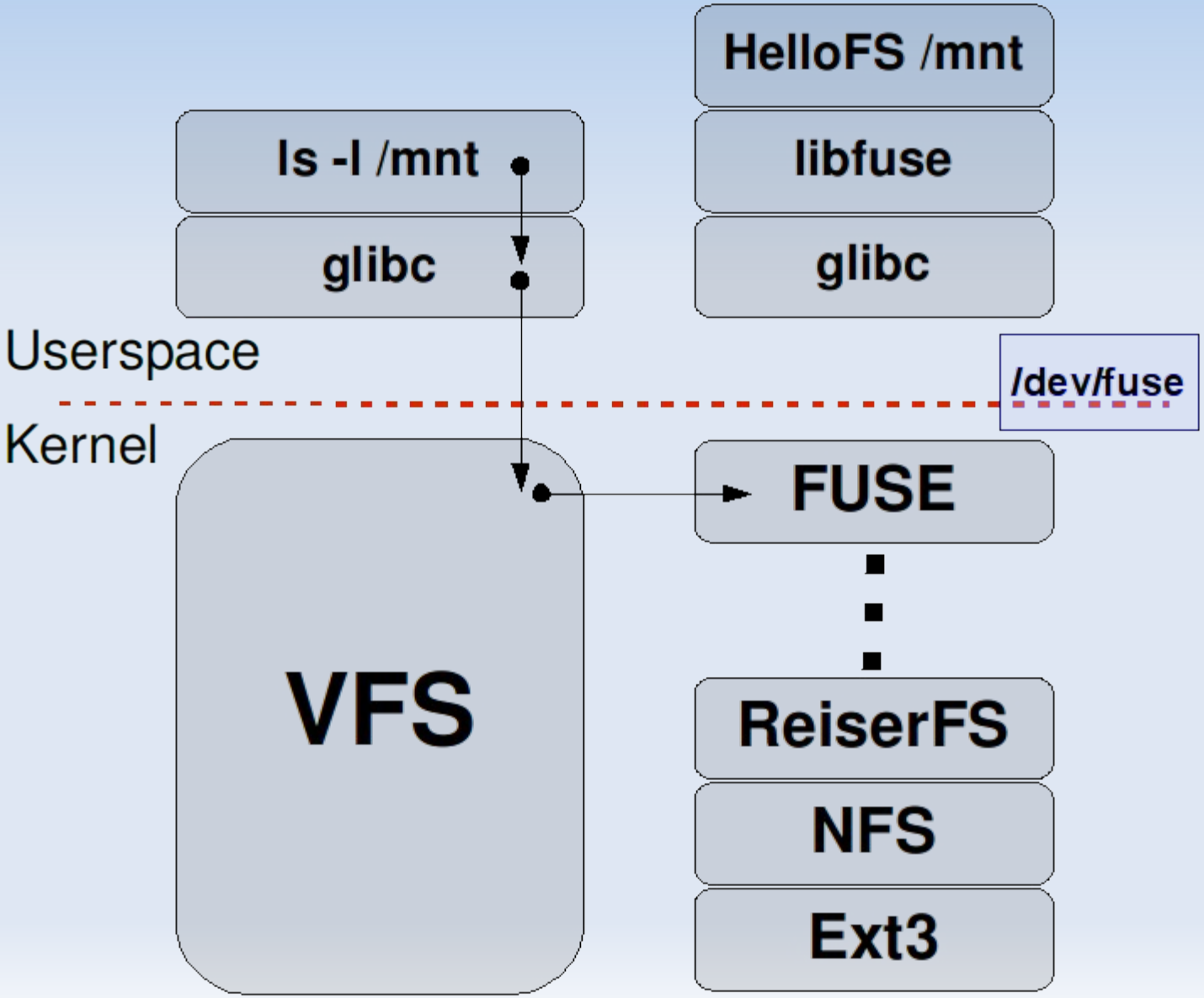
Desenvolvendo Sistemas de Arquivos no FUSE



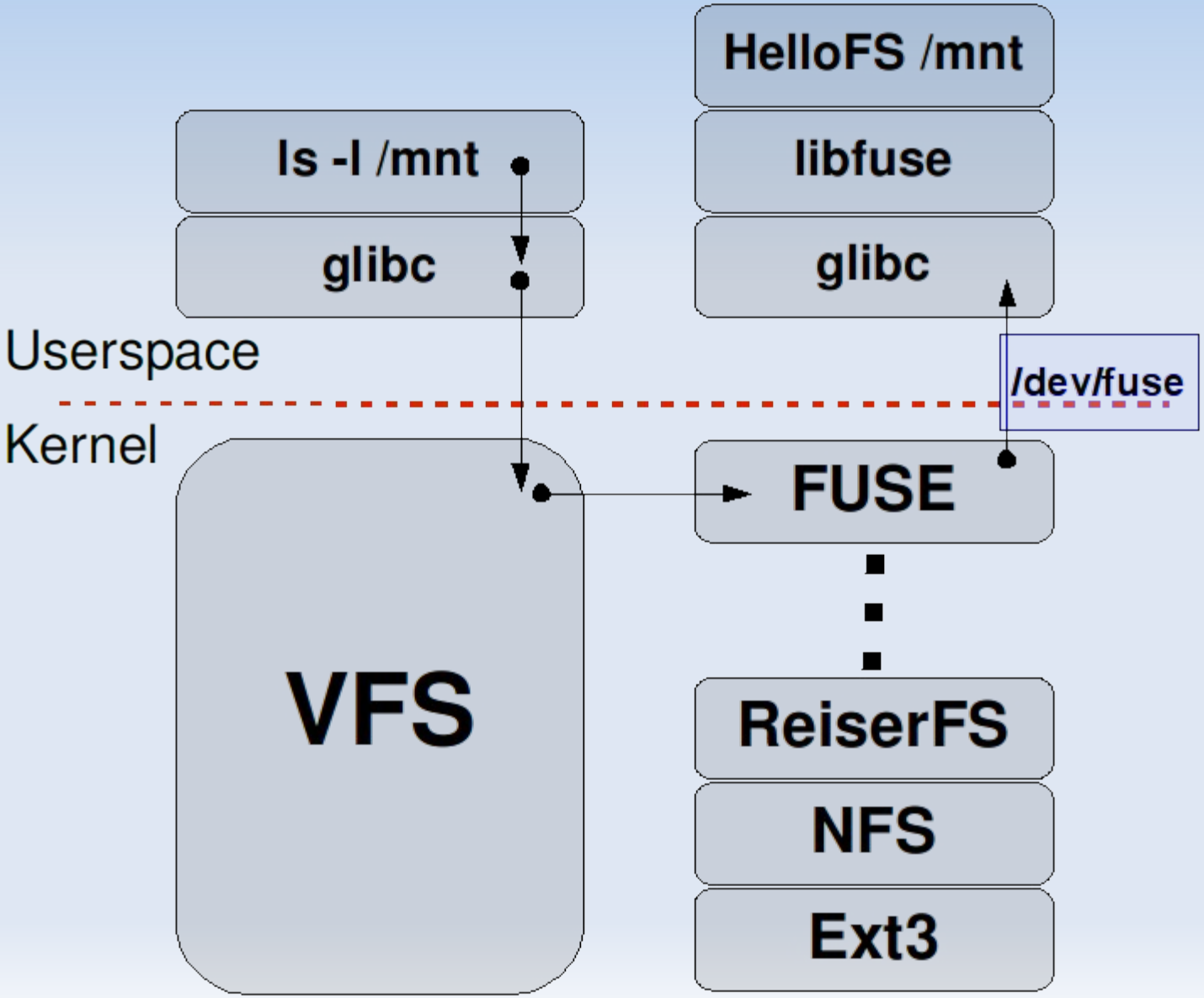
Desenvolvendo Sistemas de Arquivos no FUSE



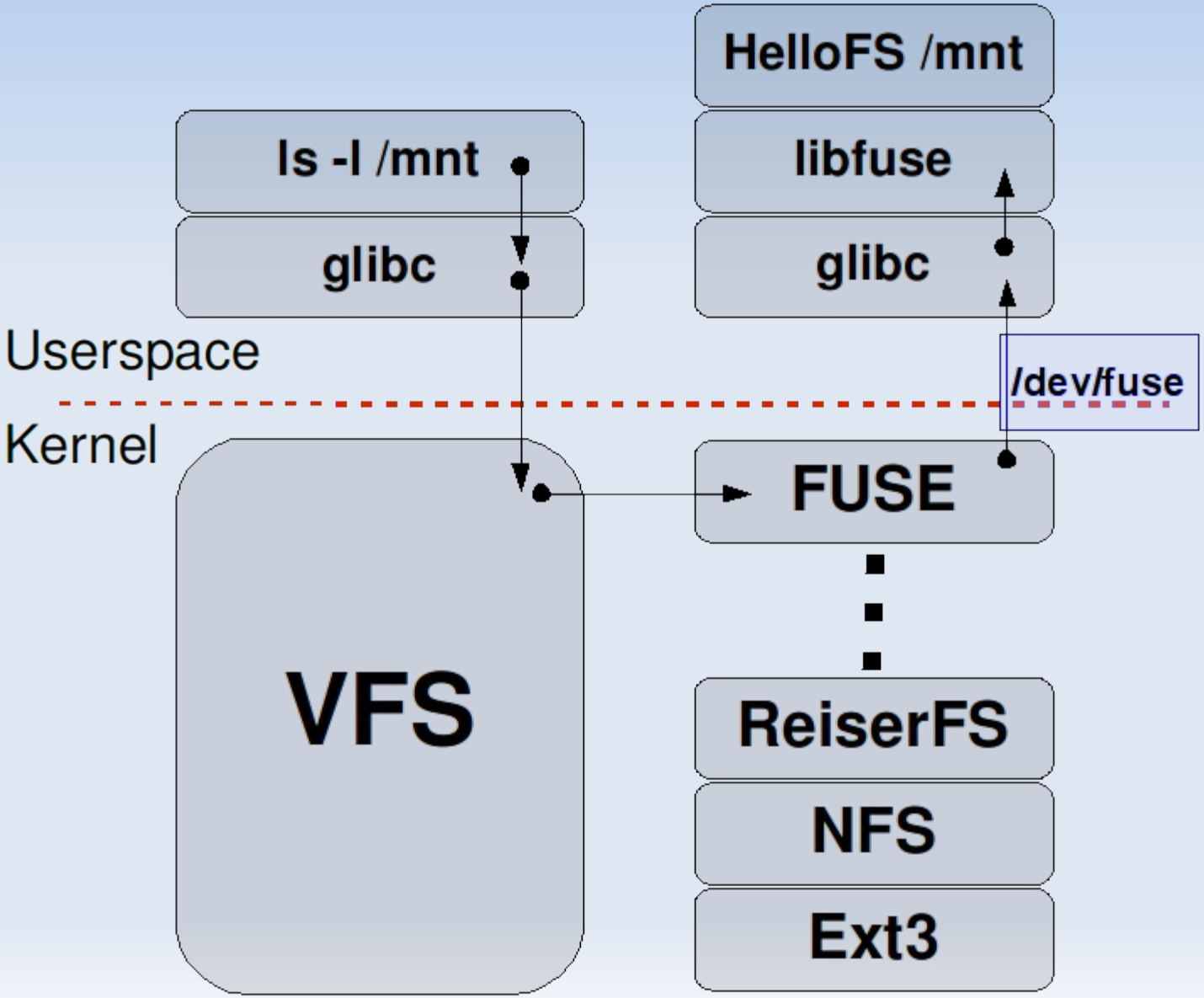
Desenvolvendo Sistemas de Arquivos no FUSE



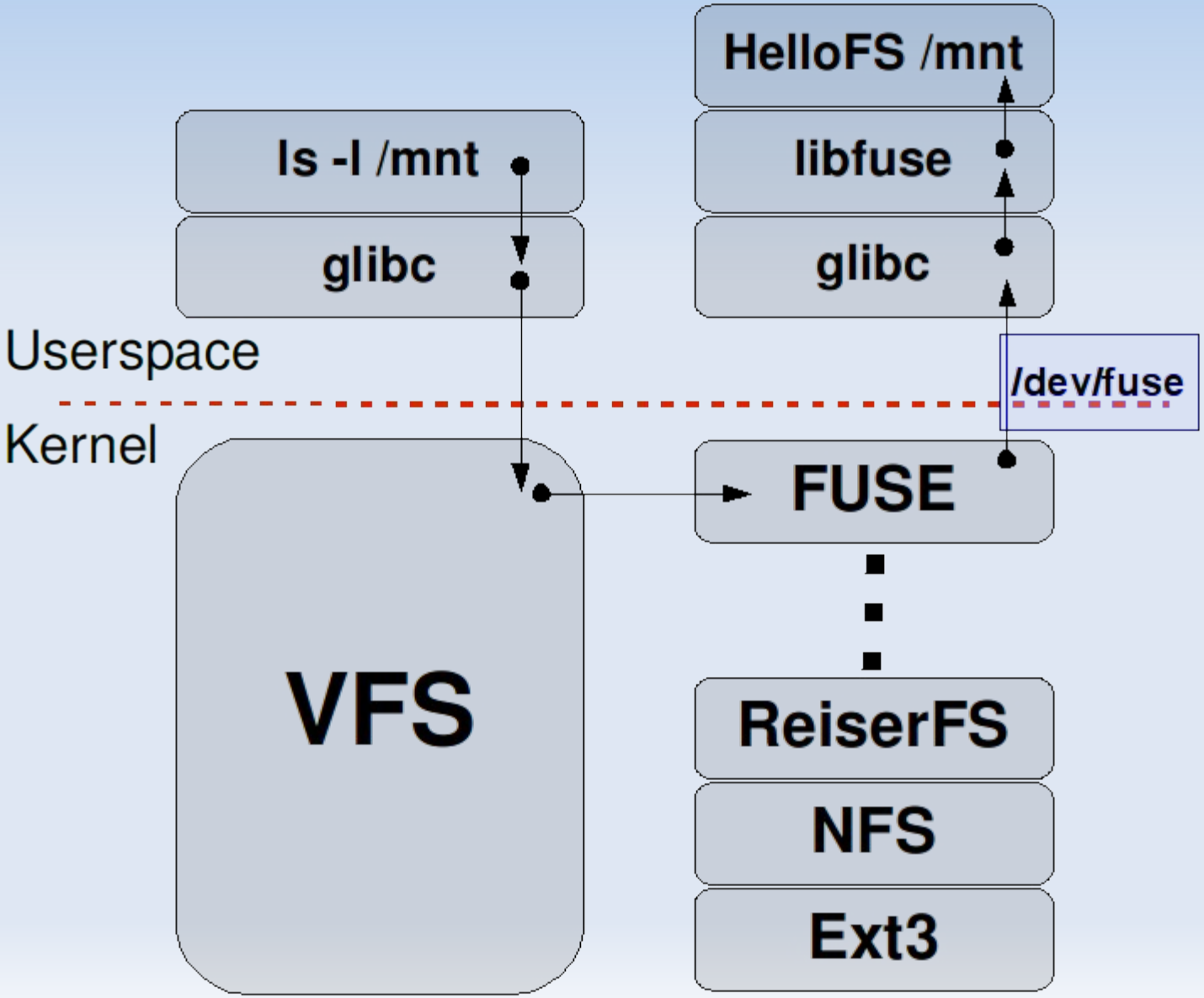
Desenvolvendo Sistemas de Arquivos no FUSE



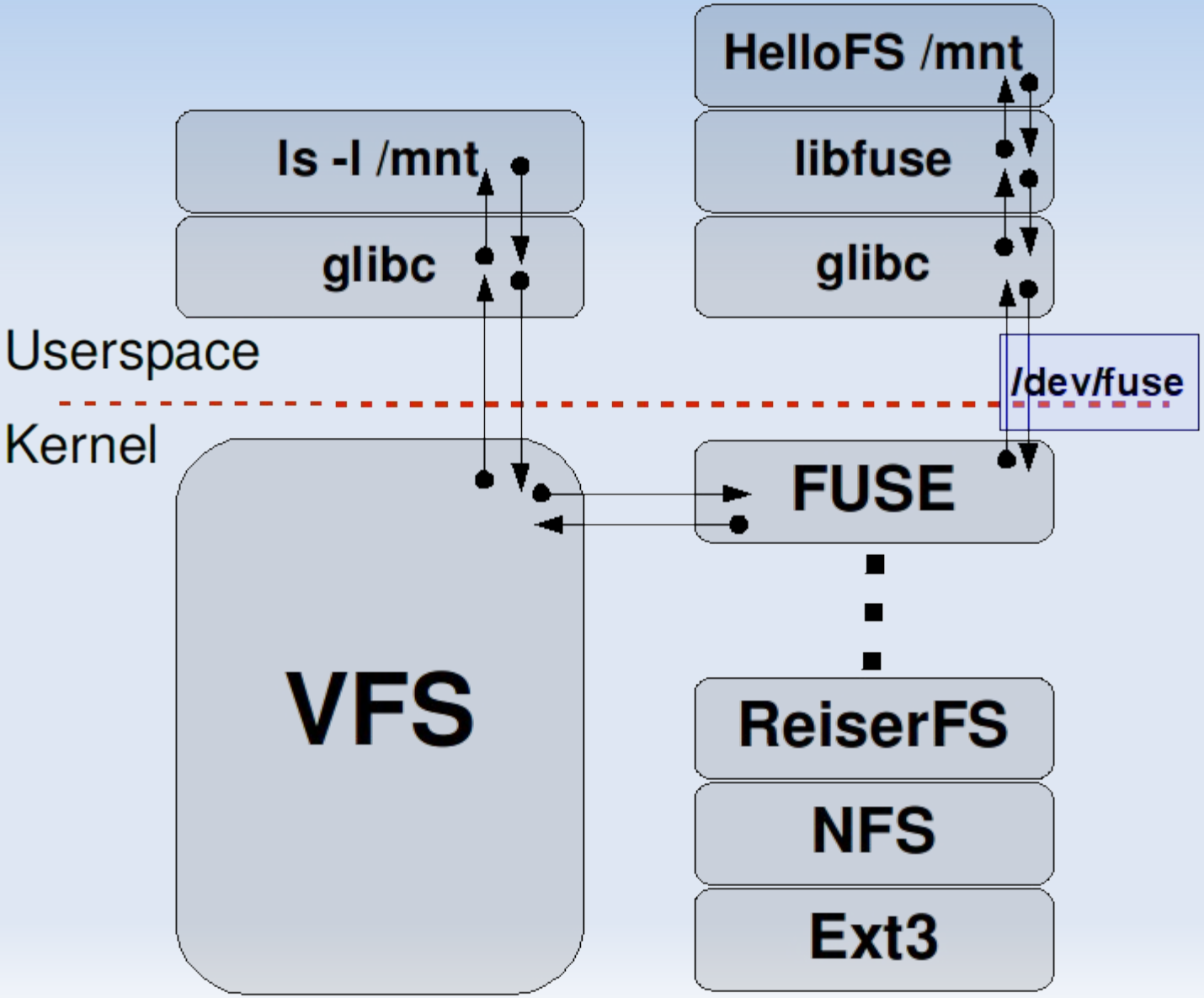
Desenvolvendo Sistemas de Arquivos no FUSE



Desenvolvendo Sistemas de Arquivos no FUSE



Desenvolvendo Sistemas de Arquivos no FUSE



Desenvolvendo Sistemas de Arquivos no FUSE

getattr

open

opendir

statfs

release

readdir

read

releasedir

write

symlink

mkdir

unlink

chmod

rmdir

Desenvolvendo Sistemas de Arquivos no FUSE

- Processar linha de comando
- Inicializar variáveis / hardware
- Inicializar o sistema de arquivos (efetivar a montagem)
- Finalizar o programa

Desenvolvendo Sistemas de Arquivos no FUSE

- **Processar linha de comando**
- **Inicializar variáveis / hardware**
- **Inicializar o sistema de arquivos (efetivar a montagem)**
- **Finalizar o programa**

Desenvolvendo Sistemas de Arquivos no FUSE

Inicializar o sistema de arquivos

fuse.h

fuse_main(argc, argv, &my_oper, NULL);

- argc, argv: opções do FUSE
- my_oper: funções que tratam o FS
- NULL: dados opcionais
- Bloqueia o programa
- Entra em modo DAEMON
  - Perde stdin, stdout, stderr (debug via log)
  - Funções de my_oper iniciam threads (exceto se a opção -s for definida)

Desenvolvendo Sistemas de Arquivos no FUSE

DEBUG

unique: 15, opcode: GETATTR (3), nodeid: 1, insize: 56
  unique: 15, error: 0 (Success), outsize: 112
unique: 16, opcode: GETXATTR (22), nodeid: 1, insize: 65
  unique: 16, error: -38 (Function not implemented), outsize: 16
unique: 17, opcode: OPENDIR (27), nodeid: 1, insize: 48
  unique: 17, error: 0 (Success), outsize: 32
unique: 18, opcode: READDIR (28), nodeid: 1, insize: 80
  unique: 18, error: 0 (Success), outsize: 616
unique: 19, opcode: LOOKUP (1), nodeid: 1, insize: 48
LOOKUP /Maildir
  NODEID: 2
  unique: 19, error: 0 (Success), outsize: 136

Desenvolvendo Sistemas de Arquivos no FUSE

unique: 20, opcode: LOOKUP (1), nodeid: 1, insize: 45

LOOKUP /logs

NODEID: 3

unique: 20, error: 0 (Success), outsize: 136

unique: 27, opcode: READDIR (28), nodeid: 1, insize: 80

unique: 27, error: 0 (Success), outsize: 16

unique: 28, opcode: RELEASEDIR (29), nodeid: 1, insize: 64

unique: 28, error: 0 (Success), outsize: 16

Desenvolvendo Sistemas de Arquivos no FUSE

Processar linha de comando

mount [options] device mountpoint

sshfs [user@]host:[dir] mountpoint [options]

Desenvolvendo Sistemas de Arquivos no FUSE

Processar linha de comando

mount [options] device mountpoint

sshfs [user@]host:[dir] mountpoint [options]

Desenvolvendo Sistemas de Arquivos no FUSE

Processar linha de comando

mount [options] device mountpoint

sshfs [user@]host:[dir] mountpoint [options]

Específicas

-p PORT
equivalent to '-o port=PORT'

-C equivalent to '-o compression=yes'

-1 equivalent to '-o ssh_protocol=1'

-o reconnect
reconnect to server

-o sshfs_sync
synchronous writes

-o no_readahead
synchronous reads (no speculative readahead)

Gerais (FUSE)

-o allow_other
allow access to other users

-o allow_root
allow access to root

-o nonempty
allow mounts over non-empty file/dir

-o default_permissions enable permission checking by kernel

Desenvolvendo Sistemas de Arquivos no FUSE

Processar linha de comando

fuse_opt.h

```
myfs <mountpoint> <myfs_option> [-o <fuse_option1>[,...[,<fuse_optionN>]]]

    struct fuse_args args = FUSE_ARGS_INIT(0, NULL);
    int i;

    for(i = 0; i < argc; i++) {
        if (i == 2) {
            myparameter = argv[i];
        } else {
            fuse_opt_add_arg(&args, argv[i]);
        }
    }
    fuse_main(args.argc, args.argv, my_oper, NULL);
```

Desenvolvendo Sistemas de Arquivos no FUSE

Processar linha de comando

```
char *myparameter;

static int myfs_opt_proc(void *data, const char *arg, int key,
                                        struct fuse_args *outargs)
{
      if (key == FUSE_OPT_KEY_NONOPT && myparameter == NULL) {
            myparameter = strdup(arg);
            return 0;
      }
      return 1;
}

int main(int argc, char *argv[])
{
      struct fuse_args args = FUSE_ARGS_INIT(argc, argv);
      fuse_opt_parse(&args, NULL, NULL, myfs_opt_proc);
      fuse_main(args.argc, args.argv, my_oper, NULL);
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Processar linha de comando

```
char *myparameter;

static int myfs_opt_proc(void *data, const char *arg, int key,
                                        struct fuse_args *outargs)
{
      if (key == FUSE_OPT_KEY_NONOPT && myparameter == NULL) {
            myparameter = strdup(arg);
            return 0;
      }
      return 1;
}

int main(int argc, char *argv[])
{
      struct fuse_args args = FUSE_ARGS_INIT(argc, argv);
      fuse_opt_parse(&args, NULL, NULL, myfs_opt_proc);
      fuse_opt_add_arg(&args, "-oallow_other");
      fuse_main(args.argc, args.argv, my_oper, NULL);
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Funções Callback

getattr

fgetattr

access

readlink

opendir

readdir

releasedir

mknod

mkdir

symlink

unlink

rmdir

rename

link

chmod

chown

truncate

ftruncate

utimens

create

open

read

write

statfs

flush

release

fsync

setxattr

getxattr

listxattr

removexattr

lock

Desenvolvendo Sistemas de Arquivos no FUSE

```
static struct fuse_operations xmp_oper = {
        .getattr        = my_getattr,
        .access         = my_access,
        .readlink       = my_readlink,
        .opendir        = my_opendir,
        .readdir        = my_readdir,
         .
         .
         .
};
```

Desenvolvendo Sistemas de Arquivos no FUSE

Ferramentas de apoio aos callbacks

struct fuse_context :

- **uid_t uid – Id do usuário do processo chamador**
- **gid_t gid – Id do grupo do processo chamador**
- **pid_t pid – Id do processo chamador**
- **void *private_data – Dados do filesystem**

struct fuse_context *fuse_get_context(void); Informações do contexto

struct fuse_file_info:

- **int flags – modo de abertura de arquivo (open/release)**
- **uint64_t fh – file handle, se inicializado em open, estará disponível nas demais operações sobre arquivos abertos**
- **uint64_t lock_owner – Id do dono do lock (lock e flush)**

Desenvolvendo Sistemas de Arquivos no FUSE

Padrões

path – Inicia com '/', mas é relativo ao mountpoint.
Exemplo:
root@darkstar # myfs /foo
root@darkstar # ls /foo/bar
path = ”/bar”

errno – Retornar –errno (MENOS errno)
Exemplo:

```
if ((size=read( ... )<0) {
   return( –errno);
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Funções Callback

Inicialização e término do sistema de arquivos

```
static void *my_init(struct fuse_conn_info *conn)
{
    // inicialização de variáveis, threads, etc
    // uma conexão com um banco mysql, por exemplo
    myconn = mysql_init ( NULL );
    mysql_real_connect (myconn ,"localhost","user", ...);
    return(&myconn);
}

static int my_destroy(void *myconn)
{
    mysql_close(*myconn);
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Atributos do sistema de arquivos

```
static int my_statfs(const char *path, struct statvfs *stbuf)
{
        int res;
        res = statvfs(path, stbuf);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
static int my_statfs(const char *path, struct statvfs *stbuf)
{
  ...
  MYSQL **conn = fuse_get_context()->private_data;
  sprintf(qry,"SELECT MAX_DATA_LENGTH/512, DATA_FREE/512 "
     "FROM INFORMATION_SCHEMA.PARTITIONS "
     "WHERE TABLE_NAME='files' AND TABLE_SCHEMA='myfs'");
  mysql_query(*myconn,qry);
  res = mysql_store_result(*myconn);
  if (row = mysql_fetch_row(res)) {
    memset(&attr,0,sizeof(struct statvfs));
    stbuf->f_bsize      = 512;
    stbuf->f_frsize     = 512;
    stbuf->f_blocks     = row[0];
    stbuf->f_bfree      = row[1];
    ...
    return (0);
  } else {
    return (-EIO);
  }
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Atributos (uid/gid, datas, permissões, etc) do arquivo

```
static int my_getattr(const char *path, struct stat *stbuf)
{
        int res;
        res = lstat(path, stbuf);
        if (res == -1) return -errno;
        return 0;
}
static int my_fgetattr(const char *path, struct stat *stbuf,
                        struct fuse_file_info *fi)
{
        int res;
        (void) path;
        res = fstat(fi->fh, stbuf);
        if (res == -1) return -errno;
        return 0;

}
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
static int my_getattr(const char *path, struct stat *stbuf) {
   MYSQL_RES *res;   MYSQL_ROW  row;
   MYSQL *conn = (MYSQL *) fuse_get_context()->private_data;
   sprintf(qry,"select uid, gid, mode, ctime, mtime, atime, size ”
                  ”from files where path='%s'",path);
   mysql_query(conn,qry);
   res = mysql_store_result(conn);
   if (row = mysql_fetch_row(res)) {
      memset(&attr,0,sizeof(struct stat));
      stbuf->st_uid          = row[0];
      stbuf->st_gid          = row[1];
      stbuf->st_mode       = row[2];
      stbuf->st_ctime       = row[3];
      stbuf->st_mtime      = row[4];
      stbuf->st_atime       = row[5];
      stbuf->st_size        = row[6];
   } else {
      return (-ENOENT);
   }
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Listagem de diretórios

```
static int my_opendir(const char *path, struct fuse_file_info *fi)
{
        DIR *dp = opendir(path);
        if (dp == NULL)
                return -errno;

        fi->fh = (unsigned long) dp;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
#define get_dirp(f) ((DIR *) (uintptr_t) f->fh)
static int my_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
                      off_t offset, struct fuse_file_info *fi)
{
        DIR *dp = get_dirp(fi);  struct dirent *de; struct stat st;

        seekdir(dp, offset);
        while ((de = readdir(dp)) != NULL) {
                memset(&st, 0, sizeof(st));
                st.st_ino = de->d_ino;
                st.st_mode = de->d_type << 12;
                if (filler(buf, de->d_name, &st, telldir(dp)))
                        break;
        }
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
static int my_releasedir(const char *path, struct fuse_file_info *fi)
{
        DIR *dp = get_dirp(fi);
        (void) path;
        closedir(dp);
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
-rw-r--r-- 1 root root   299 2008-09-12 11:08 a.out.h
-rw-r--r-- 1 root root  5365 2008-09-12 11:08 byteswap.h
-rw-r--r-- 1 root root  4881 2008-09-12 11:07 cmathcalls.h
-rw-r--r-- 1 root root 20634 2008-09-12 11:08 confname.h
-rw-r--r-- 1 root root  1609 2008-09-12 11:08 dirent.h
-rw-r--r-- 1 root root  2593 2008-09-12 11:08 dlfcn.h
-rw-r--r-- 1 root root   426 2008-09-12 11:09 elfclass.h
-rw-r--r-- 1 root root   170 2008-09-12 11:08 endian.h
-rw-r--r-- 1 root root  3275 2008-09-12 11:08 environments.h
-rw-r--r-- 1 root root  2129 2008-09-12 11:07 errno.h
-rw-r--r-- 1 root root  2762 2008-09-12 11:08 error.h
-rw-r--r-- 1 root root  5607 2008-09-12 11:08 fcntl2.h
-rw-r--r-- 1 root root  9258 2008-09-12 11:08 fcntl.h
-rw-r--r-- 1 root root  3064 2008-09-12 11:07 fenv.h
-rw-r--r-- 1 root root   190 2008-09-12 11:07 fenvinline.h
-rw-r--r-- 1 root root  1901 2008-09-12 11:07 huge_valf.h
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
-rw-r--r-- 1 root root   299 2008-09-12 11:08 a.out.h
-rw-r--r-- 1 root root  5365 2008-09-12 11:08 byteswap.h
-rw-r--r-- 1 root root  4881 2008-09-12 11:07 cmathcalls.h
-rw-r--r-- 1 root root 20634 2008-09-12 11:08 confname.h
-rw-r--r-- 1 root root  1609 2008-09-12 11:08 dirent.h
-rw-r--r-- 1 root root  2593 2008-09-12 11:08 dlfcn.h
-rw-r--r-- 1 root root   426 2008-09-12 11:09 elfclass.h
-rw-r--r-- 1 root root   170 2008-09-12 11:08 endian.h
-rw-r--r-- 1 root root  3275 2008-09-12 11:08 environments.h
-rw-r--r-- 1 root root  2129 2008-09-12 11:07 errno.h
-rw-r--r-- 1 root root  2762 2008-09-12 11:08 error.h
-rw-r--r-- 1 root root  5607 2008-09-12 11:08 fcntl2.h
-rw-r--r-- 1 root root  9258 2008-09-12 11:08 fcntl.h
-rw-r--r-- 1 root root  3064 2008-09-12 11:07 fenv.h
-rw-r--r-- 1 root root   190 2008-09-12 11:07 fenvinline.h
-rw-r--r-- 1 root root  1901 2008-09-12 11:07 huge_valf.h
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
-rw-r--r-- 1 root root   299 2008-09-12 11:08 a.out.h
-rw-r--r-- 1 root root  5365 2008-09-12 11:08 byteswap.h
-rw-r--r-- 1 root root  4881 2008-09-12 11:07 cmathcalls.h
-rw-r--r-- 1 root root 20634 2008-09-12 11:08 confname.h
-rw-r--r-- 1 root root  1609 2008-09-12 11:08 dirent.h
-rw-r--r-- 1 root root  2593 2008-09-12 11:08 dlfcn.h
-rw-r--r-- 1 root root   426 2008-09-12 11:09 elfclass.h
-rw-r--r-- 1 root root   170 2008-09-12 11:08 endian.h
-rw-r--r-- 1 root root  3275 2008-09-12 11:08 environments.h
-rw-r--r-- 1 root root  2129 2008-09-12 11:07 errno.h
-rw-r--r-- 1 root root  2762 2008-09-12 11:08 error.h
-rw-r--r-- 1 root root  5607 2008-09-12 11:08 fcntl2.h
-rw-r--r-- 1 root root  9258 2008-09-12 11:08 fcntl.h
-rw-r--r-- 1 root root  3064 2008-09-12 11:07 fenv.h
-rw-r--r-- 1 root root   190 2008-09-12 11:07 fenvinline.h
-rw-r--r-- 1 root root  1901 2008-09-12 11:07 huge_valf.h
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
-rw-r--r-- 1 root root  5365 2008-09-12 11:08 byteswap.h
-rw-r--r-- 1 root root  4881 2008-09-12 11:07 cmathcalls.h
-rw-r--r-- 1 root root 20634 2008-09-12 11:08 confname.h
-rw-r--r-- 1 root root  1609 2008-09-12 11:08 dirent.h
-rw-r--r-- 1 root root  2593 2008-09-12 11:08 dlfcn.h
-rw-r--r-- 1 root root   426 2008-09-12 11:09 elfclass.h
-rw-r--r-- 1 root root   170 2008-09-12 11:08 endian.h
-rw-r--r-- 1 root root  3275 2008-09-12 11:08 environments.h
-rw-r--r-- 1 root root  2129 2008-09-12 11:07 errno.h
-rw-r--r-- 1 root root  2762 2008-09-12 11:08 error.h
-rw-r--r-- 1 root root  5607 2008-09-12 11:08 fcntl2.h
-rw-r--r-- 1 root root  9258 2008-09-12 11:08 fcntl.h
-rw-r--r-- 1 root root  3064 2008-09-12 11:07 fenv.h
-rw-r--r-- 1 root root   190 2008-09-12 11:07 fenvinline.h
-rw-r--r-- 1 root root  1901 2008-09-12 11:07 huge_valf.h
```

Desenvolvendo Sistemas de Arquivos no FUSE

Criação de arquivos – MÉTODO OPCIONAL

```
static int my_create(const char *path, mode_t mode, struct fuse_file_info *fi)
{
        int fd = open(path, fi->flags, mode);
        if (fd == -1)
                return -errno;
        fi->fh = fd;  // prover um descritor de arquivos é OPCIONAL
        return 0;
}
```

Flags

O_EXCL - Falha se o arquivo já existir

O_NONBLOCK - Evita que bloqueie por um "longo" tempo

O_NOLINK - Se for link simbólico, abre o link (não o arquivo)

O_TRUNC - Trunca o arquivo

O_SHLOCK (BSD) - Obtem shared lock

O_EXLOCK (BSD) - Obtem lock exclusivo

Desenvolvendo Sistemas de Arquivos no FUSE

Cria (sem abrir) um arquivo

```
static int my_mknod(const char *path, mode_t mode, dev_t rdev)
{
        int res;

        if (S_ISFIFO(mode))
                res = mkfifo(path, mode);
        else
                res = mknod(path, mode, rdev);
        if (res == -1)
                return -errno;

        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

```
Abre  arquivo existente (não cria arquivos)
static int my_open(const char *path, struct fuse_file_info *fi)
{
        int fd = open(path, fi->flags); // sem mode
        if (fd == -1)
                return -errno;

        fi->fh = fd; // fornecer um file handle é OPCIONAL
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Lê dados de um arquivo

```
static int my_read(const char *path, char *buf, size_t size, off_t offset,
                   struct fuse_file_info *fi)
{
        int res;

        (void) path;
        res = pread(fi->fh, buf, size, offset);
        if (res == -1)
                res = -errno;

        return res;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Grava dados num arquivo

```
static int my_write(const char *path, const char *buf, size_t size,
                    off_t offset, struct fuse_file_info *fi)
{
        int res;

        (void) path;
        res = pwrite(fi->fh, buf, size, offset);
        if (res == -1)
                res = -errno;

        return res;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Fecha arquivo / libera recursos

```
static int my_release(const char *path, struct fuse_file_info *fi)
{
        (void) path;
        close(fi->fh);

        return 0;
}
```

OBS: Só é chamado uma vez por arquivo aberto, se a aplicação duplicou o descritor de arquivo (dup, fork, ...) release é chamado quando a aplicação chamar close para o último descritor aberto.

Desenvolvendo Sistemas de Arquivos no FUSE

Lê o conteúdo de um link simbólico

```
static int my_readlink(const char *path, char *buf, size_t size)
{
        int res = readlink(path, buf, size - 1);
        if (res == -1)
                return -errno;

        buf[res] = '\0';
        return 0;
}
```

OBS: readlink da libc copia string sem '\0' para o buf e retorna a quantidade de caracteres lidos, o callback do FUSE exige o terminador na string e o valor de retorno é -errno.

Desenvolvendo Sistemas de Arquivos no FUSE

Cria symlink/hardlink

```
static int my_symlink(const char *from, const char *to)
{
        int res = symlink(from, to);
        if (res == -1)
                return -errno;
        return 0;
}

static int my_link(const char *from, const char *to)
{
        int res = link(from, to);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Remove (deleta) um arquivo

```
static int my_unlink(const char *path)
{
        int res;

        res = unlink(path);
        if (res == -1)
                return -errno;

        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Renomeia/Move um arquivo

```
static int my_rename(const char *from, const char *to)
{
        int res;

        res = rename(from, to);
        if (res == -1)
                return -errno;

        return 0;
}
```

OBS: from e to sempre estão em myfs

Desenvolvendo Sistemas de Arquivos no FUSE

Muda atributos dos arquivo

```
static int my_chmod(const char *path, mode_t mode)
{
        int res = chmod(path, mode);
        if (res == -1)
                return -errno;
        return 0;
}

static int my_chown(const char *path, uid_t uid, gid_t gid)
{
        int res = lchown(path, uid, gid);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Muda atributos de data/hora com precisão de nanosegundos

```
static int my_utimens(const char *path, const struct timespec ts[2])
{
        int res;
        struct timeval tv[2];

        tv[0].tv_sec = ts[0].tv_sec;
        tv[0].tv_usec = ts[0].tv_nsec / 1000;
        tv[1].tv_sec = ts[1].tv_sec;
        tv[1].tv_usec = ts[1].tv_nsec / 1000;

        res = utimes(path, tv);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Cria/deleta diretório

```
static int my_mkdir(const char *path, mode_t mode)
{
        int res = mkdir(path, mode);
        if (res == -1)
                return -errno;
        return 0;
}

static int my_rmdir(const char *path)
{
        int res = rmdir(path);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Verifica se é possível fazer o acesso, especificado em mask, ao arquivo

```
static int my_access(const char *path, int mask)
{
	int res;

	res = access(path, mask);
	if (res == -1)
		return -errno;

	return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Modifica o tamanho de um arquivo pelo nome ou pelo file handle

```
static int my_truncate(const char *path, off_t size)
{
        int res = truncate(path, size);
        if (res == -1)
                return -errno;
        return 0;
}

static int my_ftruncate(const char *path, off_t size,
                        struct fuse_file_info *fi)
{
        int res = ftruncate(fi->fh, size);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Grava atributo extendido

```
static int my_setxattr(const char *path, const char *name, const char *value,
                       size_t size, int flags)
{
        int res = lsetxattr(path, name, value, size, flags);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Lê/lista atributos extendidos

```
static int my_getxattr(const char *path, const char *name, char *value,
                       size_t size)
{
        int res = lgetxattr(path, name, value, size);
        if (res == -1)
                return -errno;
        return res;
}

static int my_listxattr(const char *path, char *list, size_t size)
{
        int res = llistxattr(path, list, size);
        if (res == -1)
                return -errno;
        return res;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Remove atributo extendido

```
static int my_removexattr(const char *path, const char *name)
{
        int res = lremovexattr(path, name);
        if (res == -1)
                return -errno;
        return 0;
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Gerencia trava de arquivo, segundo o padrão POSIX

```
static int my_lock(const char *path, struct fuse_file_info *fi, int cmd,
                   struct flock *lock)
{
        (void) path;

        return ulockmgr_op(fi->fh, cmd, lock, &fi->lock_owner,
                           sizeof(fi->lock_owner));
}
```

Desenvolvendo Sistemas de Arquivos no FUSE

Sincroniza dados em memória com o meio de armazenamento

```
static int my_fsync(const char *path, int datasync, struct fuse_file_info *fi)
{
        (void) path;
        (void) datasync;
        (void) fi;

        return (0);
}
```

OBS: Se datasync == zero, deve sincronizar dados e meta-dados

Se datasync != zero, deve sincronizar somente dados

Desenvolvendo Sistemas de Arquivos no FUSE

Sincroniza dados em memória com o meio de armazenamento

```
static int my_flush(const char *path, struct fuse_file_info *fi)
{
        (void) path;
        (void) fi;

        return (0);
}
```

OBS: Não é chamado quando a aplicação chama fsync.
Só é chamado quando a aplicação chama close.
Pode haver mais de um close por arquivo (dup, fork, ...)
Não há garantia de que seja chamado sequer uma vez.

Desenvolvendo Sistemas de Arquivos no FUSE

```
int fuseargc = 5;
char *fuseargv[6];
fuseargv[0] = "sohofs";
fuseargv[1] = "-f";
fuseargv[2] = "/tmp/fuse";
fuseargv[3] = "-o";
fuseargv[4] = "allow_other,nonempty";
fuseargv[5] = NULL;
return (fuse_main(fuseargc, fuseargv, &my_ops, NULL));
```

it reports the error "Access is denied" when I try to copy a new file into the share folder through SAMBA.

Try to run in debug mode:

```
fuseargv[4] = "allow_other,nonempty,debug";
```

And check what is happening when samba calls access/getattr.

Desenvolvendo Sistemas de Arquivos no FUSE

unique: 2, opcode: GETATTR (3), nodeid: 1, insize: 40
  unique: 2, error: 0 (Success), outsize: 112
WRITE[6] 4096 bytes to 0
  unique: 128, error: -28 (No space left on device), outsize: 16
unique: 129, opcode: WRITE (16), nodeid: 9, insize: 4160
WRITE[6] 4096 bytes to 0
  unique: 129, error: -28 (No space left on device), outsize: 16
unique: 130, opcode: WRITE (16), nodeid: 9, insize: 4160
WRITE[6] 4096 bytes to 0
  unique: 130, error: -28 (No space left on device), outsize: 16

But there are actually free space on the the device, so what are the potential issues.

So, the problem is with the statfs callback function. This function is responsible to tell the system the info about the filesystem and this includes the space in it.