Systems Programming

File System

H. Turgut Uyar Şima Uyar

2009-2014

Topics

System Programming Levels

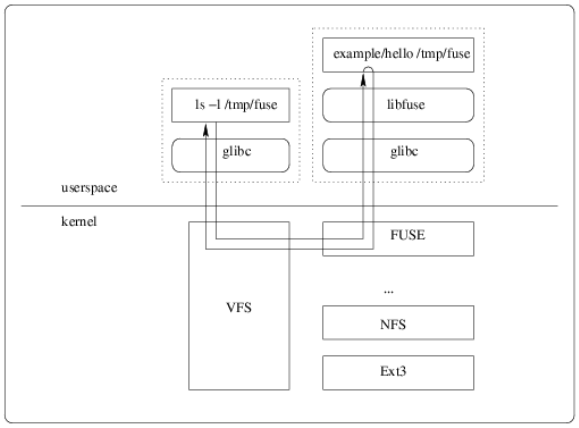
- compiling the kernel:
  pros: best performance, every possible functionality
  cons: risky, time-consuming
- kernel modules:
  pros: very good performance, less risky, fast development
  cons: can not do everything
- user-space:
  pros: even less risky, fast development, can use external libraries
  cons: poorer performance, can do even less

FUSE

- Filesystem in Userspace
- develop a file system in user space on top of a kernel module

- non-native filesystems:
  NTFS, ZFS, ...
- changing data storage:
  SQL, ...
- providing transparent functionality:
  compression, encryption, ...

FUSE Structure



FUSE Development

- similar to device driver development:
  implement system calls
- file related system calls:
  `open, release, read, write, getattr, unlink, ...`
- directory related system calls:
  `readdir, mkdir, rmdir, ...`

Mounting

- associating a file hierarchy with a top-level directory:
  mounting
- requests are relative to top-level directory

example

- file system mounted on `/mnt/fuse`
- `ls /mnt/fuse` → `readdir "/"`
- `mkdir /mnt/fuse/foo` → `mkdir "/foo"`

Example

Hello, world!

- virtual filesystem with only one directory and one file
- name of the file: `hello.txt`
- contents of the file: `Hello, world!`

```
static const char *hello_path = "/hello.txt";
static const char *hello_str = "Hello, world!\n";
```

Example

map system calls to functions: `fuse_operations`

```
static struct fuse_operations hello_oper = {
    .getattr = hello_getattr,
    .readdir = hello_readdir,
    .open    = hello_open,
    .read    = hello_read,
};
```


Directory Listing

- directory listing: `readdir`

```
static int hello_readdir(
    const char *path,
    void *buf,
    fuse_fill_dir_t filler,
    off_t offset,
    struct fuse_file_info *fi
);
```


Example

`readdir`

```
if (strcmp(path, "/") != 0)
    return -ENOENT;

filler(buf, ".", NULL, 0);
filler(buf, "..", NULL, 0);
filler(buf, hello_path + 1, NULL, 0);
```


File Attributes

- reading file attributes: `getattr`

```
static int hello_getattr(
    const char *path,
    struct stat *stbuf
);
```

Example

getattr

```
memset(stbuf, 0, sizeof(struct stat));
if (strcmp(path, "/") == 0)
{
    stbuf->st_mode = S_IFDIR | 0755;
    stbuf->st_nlink = 2;
}
else if (strcmp(path, hello_path) == 0)
{
    stbuf->st_mode = S_IFREG | 0444;
    stbuf->st_nlink = 1;
    stbuf->st_size = strlen(hello_str);
}
else
    res = -ENOENT;
```


Reading Files

- reading from a file: `read`

```
static int hello_read(
    const char *path,
    char *buf,
    size_t size,
    off_t offset,
    struct fuse_file_info *finfo
);
```


Example

read

```
if (strcmp(path, hello_path) != 0)
    return -ENOENT;

len = strlen(hello_str);
if (offset < len)
{
    if (offset + size > len)
        size = len - offset;
    memcpy(buf, hello_str + offset, size);
}
else
    size = 0;

return size;
```


ROFS

read-only file system

- access an underlying directory in read-only mode
- all read accesses are delegated to the underlying directory
- all write accesses are denied
- for example:
- `rw_path`: `/home/itucs/Documents`
- mounted on `/mnt/Documents`

Example

operations

```
struct fuse_operations rofs_oper = {
    .getattr = rofs_getattr,
    .readdir = rofs_readdir,
    .mkdir   = rofs_mkdir,
    .unlink  = rofs_unlink,
    .rmdir   = rofs_rmdir,
    .rename  = rofs_rename,
    .open    = rofs_open,
    .read    = rofs_read,
    .write   = rofs_write,
    .release = rofs_release,
    ...
};
```


Example

path translation

```
char *rPath = malloc(sizeof(char) *
    (strlen(path) + strlen(rw_path) + 1));

strcpy(rPath, rw_path);
if (rPath[strlen(rPath)-1] == '/')
    rPath[strlen(rPath)-1] = '\0';
strcat(rPath, path);

return rPath;
```


Example

directory listing

```
upath = translate_path(path);
dp = opendir(upath);  /* DIR *dp; */
free(upath);
if (dp == NULL)
{
    res = -errno;
    return res;
}

/* fill in the directory info */

closedir(dp);
```


Example

directory info

```
/* struct dirent *de; */
while ((de = readdir(dp)) != NULL)
{
    struct stat st;
    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, 0))
        break;
}
```

Example

file attributes

```
upath = translate_path(path);
res = lstat(upath, st_data);
free(upath);
if (res == -1)
    return -errno;
```


Example

reading from a file

```
upath = translate_path(path);
fd = open(upath, O_RDONLY);
free(upath);
if (fd == -1)
{
    res = -errno;
    return res;
}
res = pread(fd, buf, size, offset);
if (res == -1)
    res = -errno;
close(fd);
```


Example

modification operations

```
static int rofs_mkdir(
    const char *path,
    mode_t mode
);

static int rofs_unlink(const char *path);

/* body */
return -EROFS;
```