Programming Assignment #6
A User-Space File System

Introduction to Operating Systems

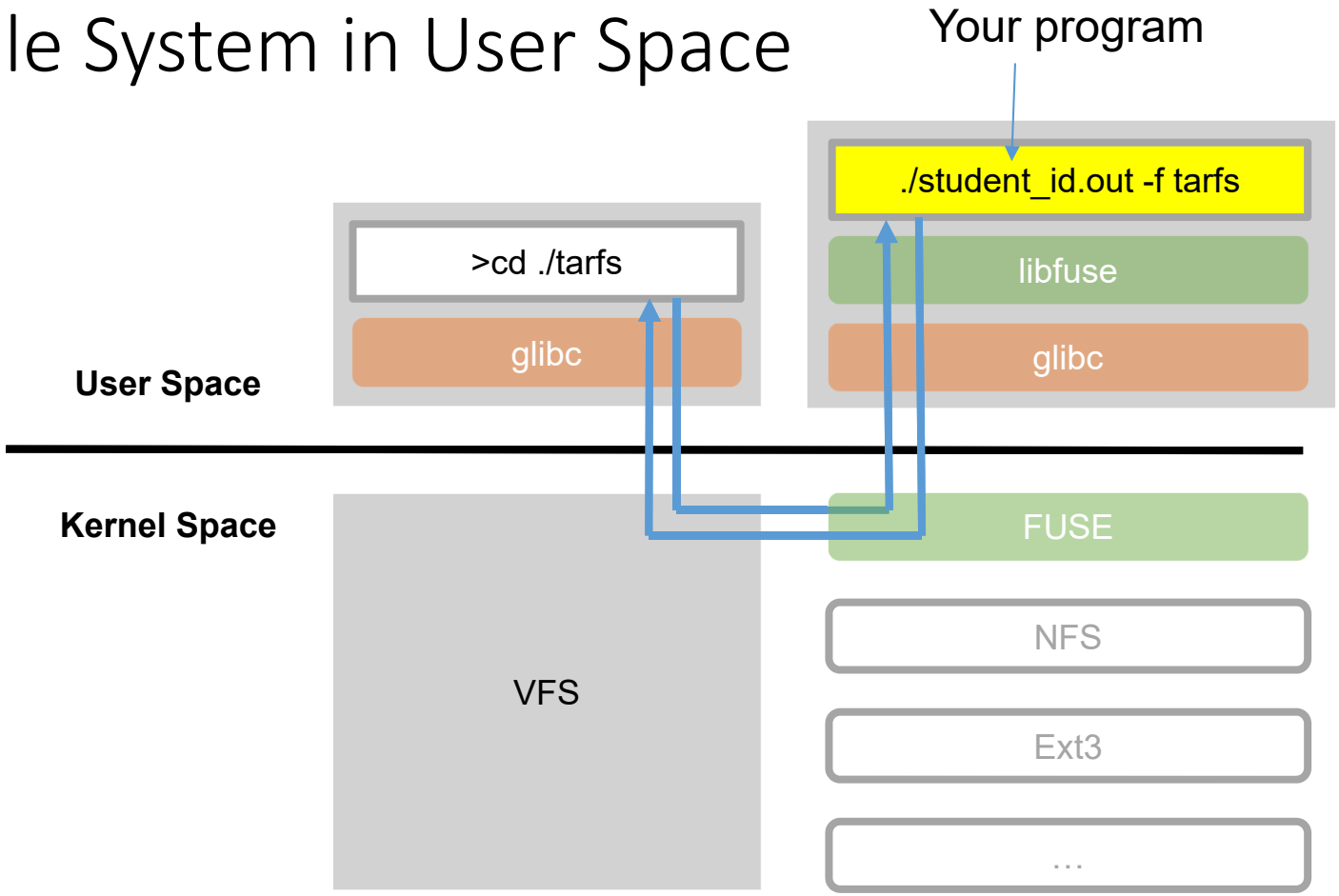
Prof. Li-Pin Chang
CS@NYCU

Objective

- Implementing a user-space file system that mounts a tar file onto a specified directory
- Files in the tar files can be accessed through the system directory tree

- This assignment is based on FUSE of Linux
  - Your program will run as a FUSE server
  - Test your FUSE server from another terminal

FUSE – File System in User Space



- FUSE: a kernel component plus a user-space library
- Purpose: accessing existing files/services through the file system interface
  - E.g., an FTP file system, a zip file system, etc.

The Complete FUSE Operation Set

```
      int(*  getattr )(const char *, struct stat *, struct fuse_file_info *fi)
      int(*  readlink )(const char *, char *, size_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 *, unsigned int flags)
      int(*  link )(const char *, const char *)
      int(*  chmod )(const char *, mode_t, struct fuse_file_info *fi)
      int(*  chown )(const char *, uid_t, gid_t, struct fuse_file_info *fi)
      int(*  truncate )(const char *, off_t, struct fuse_file_info *fi)
      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 *)
      int(*  statfs )(const char *, struct statvfs *)
      int(*  flush )(const char *, struct fuse_file_info *)
      int(*  release )(const char *, struct fuse_file_info *)
      int(*  fsync )(const char *, int, struct fuse_file_info *)
      int(*  setxattr )(const char *, const char *, const char *, size_t, int)
      int(*  getxattr )(const char *, const char *, char *, size_t)
      int(*  listxattr )(const char *, char *, size_t)
      int(*  removexattr )(const char *, const char *)
      int(*  opendir )(const char *, struct fuse_file_info *)
      int(*  readdir )(const char *, void *, fuse_fill_dir_t, off_t, struct fuse_file_info *, enum fuse_readdir_flags)
      int(*  releasedir )(const char *, struct fuse_file_info *)
      int(*  fsyncdir )(const char *, int, struct fuse_file_info *)
 void *(*  init )(struct fuse_conn_info *conn, struct fuse_config *cfg)
     void(*  destroy )(void *private_data)
      int(*  access )(const char *, int)
      int(*  create )(const char *, mode_t, struct fuse_file_info *)
      int(*  lock )(const char *, struct fuse_file_info *, int cmd, struct flock *)
      int(*  utimens )(const char *, const struct timespec tv[2], struct fuse_file_info *fi)
      int(*  bmap )(const char *, size_t blocksize, uint64_t *idx)
      int(*  ioctl )(const char *, unsigned int cmd, void *arg, struct fuse_file_info *, unsigned int flags, void *data)
      int(*  poll )(const char *, struct fuse_file_info *, struct fuse_pollhandle *ph, unsigned *reventsp)
      int(*  write_buf )(const char *, struct fuse_bufvec *buf, off_t off, struct fuse_file_info *)
      int(*  read_buf )(const char *, struct fuse_bufvec **bufp, size_t size, off_t off, struct fuse_file_info *)
      int(*  flock )(const char *, struct fuse_file_info *, int op)
      int(*  fallocate )(const char *, int, off_t, off_t, struct fuse_file_info *)
ssize_t(*  copy_file_range )(const char *path_in, struct fuse_file_info *fi_in, off_t offset_in, const char *path_out, s
  off_t(*  lseek )(const char *, off_t off, int whence, struct fuse_file_info *)
```

Necessary FUSE Operations

```
struct fuse_operations {
        int (*readdir)(const char *, void *, fuse_fill_dir_t, off_t, struct fuse_file_info *);
        int (*getattr)(const char *, struct stat *);
        int (*read)(const char *, char *, size_t, off_t, struct fuse_file_info *);
        //many other functions…
}
```

- The complete FUSE operation set contains many callback functions, but only three are necessary to this assignment
  - **readdir**: Get a list of files and directories that reside in the directory. (Get file names only)
  - **getattr**: Get attributes of a file/directory.
  - **read**: Get the content of a file
- Leave null to the other operations

readdir

```
int readdir(const char *path, void *buffer, fuse_fill_dir_t filler, off_t offset, struct fuse_file_info *fi);
```

- **Arguments**
  - path: Relative path to the file/directory.
  - buffer: You should store file names into this buffer using the provided filler
  - filler: A tool to store the file names into buffer
    - `filler(buffer, “file1.txt”, NULL, 0);`
    - `filler(buffer, “dir1”, NULL, 0);`
  - offset and fi: Not used in this assignment
- **Return values**
  - Always return 0.

getattr

```
int getattr(const char *path, struct stat *st);
```

- **Arguments**
  - path: Relative path to the file/directory.
  - st: You should fill the necessary fields of this structure.
- About structure stat: https://pubs.opengroup.org/onlinepubs/009695399/basedefs/sys/stat.h.html
- Necessary Fields of st: st_uid, st_gid, st_mtime, st_size and st_mode
  - st_mode of the root directory (“/”) should be set to: S_IFDIR | 0444 (act like a read only directory)
  - Other directories: S_IFDIR | accessMode
  - Regular files: S_IFREG | accessMode
- **Return values**
  - Return 0 on success.
  - Return a nonzero value on failure. (If cannot find the specified file/directory)

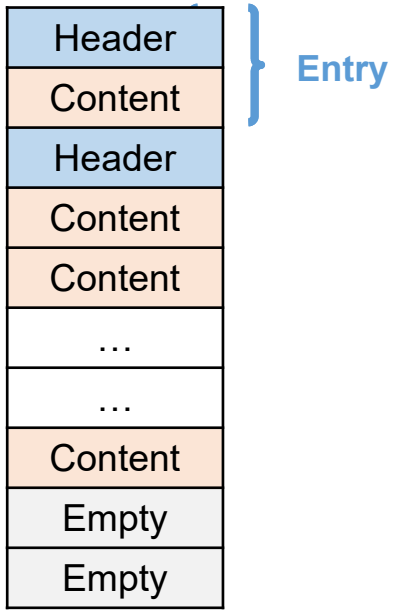
read

```
int read(const char *path, char *buffer, size_t size, off_t offset, struct fuse_file_info *fi);
```

- **Arguments**
  - path: Relative path to the file/directory.
  - buffer: You should store the requested file content into this buffer.
  - size: Max # of chars to store in the buffer. (Should not overrun)
  - offset: Skip *offset* chars from the beginning of the file and then start reading.
  - fi: Not used in this assignment
- **Return values**
  - Return number of bytes read successfully. (Less than or equal to *size*)

Tar File Format

- A tar file contains a series of entries, each of which contains a header and contents
  - One entry per file
  - Header: metadata of a file
  - Contents: contents of the file
- You are responsible for reading and parsing the information of a tar file
- Detailed explanation of tar format: https://www.systutorials.com/docs/linux/man/5-tar/



Skeleton of Your FUSE Server

```
#define FUSE_USE_VERSION 30
#include <fuse.h>
#include <string.h>
int my_readdir(const char *path, void *buffer, fuse_fill_dir_t filler, off_t offset, struct fuse_file_info *fi) { /*do something*/ }
int my_getattr(const char *path, struct stat *st) { /*do something*/ }
int my_read(const char *path, char *buffer, size_t size, off_t offset, struct fuse_file_info *fi) { /*do something*/ }
static struct fuse_operations op;
int main(int argc, char *argv[])
{
    memset(&op, 0, sizeof(op));
    op.getattr = my_getattr;
    op.readdir = my_readdir;
    op.read = my_read;
    return fuse_main(argc, argv, &op, NULL);
}
```

Compiling Your FUSE server

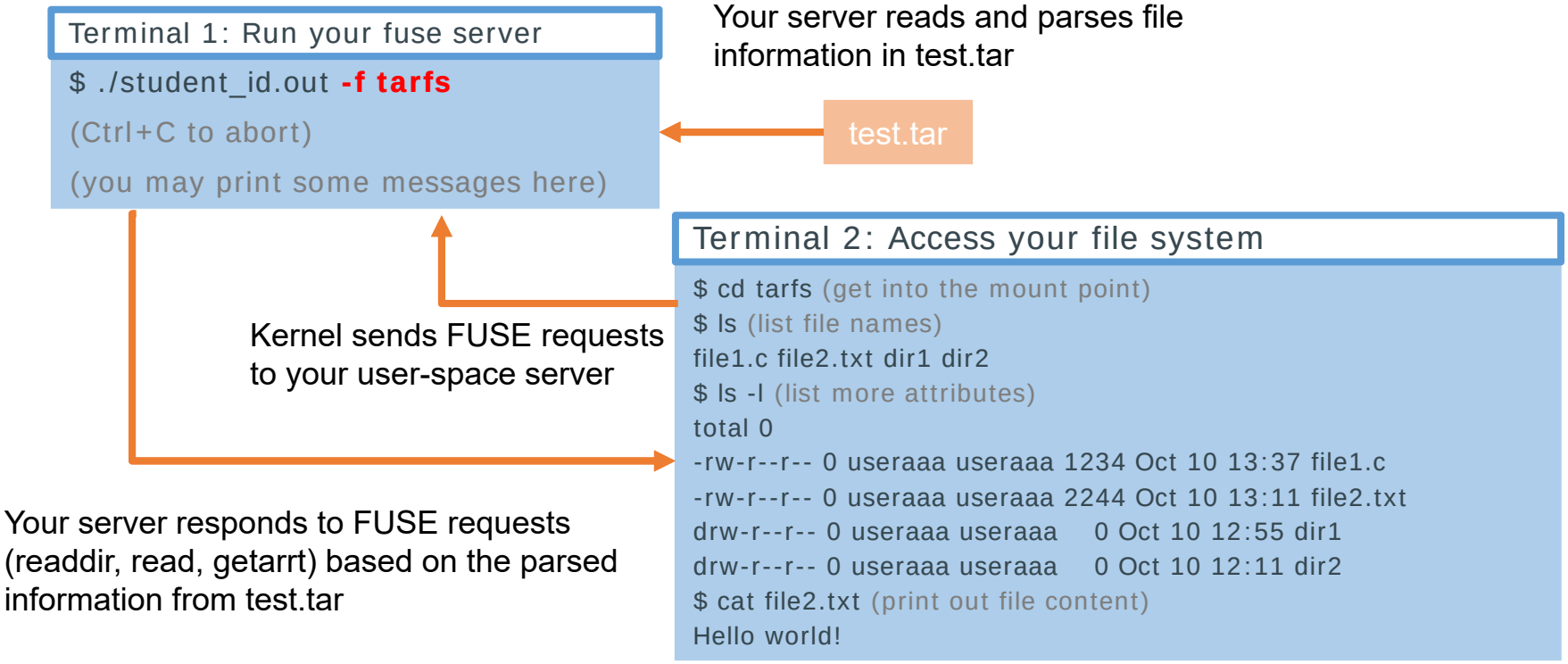
- Install FUSE in your VM Ubuntu 18.04
  - `sudo apt install libfuse-dev`
- Compile

```
$ gcc student_id.c -o student_id.out `pkg-config fuse --cflags --libs`
OR
$ g++ student_id.cpp -o student_id.out `pkg-config fuse --cflags --libs`
```

In Your Working Directory…

- File name of the tar file is fixed: test.tar
  - TA will provide this file
- Put the tar file and your program in the same directory while testing
  1. test.tar
  2. student_id.out (your executable file)
  3. tarfs (Create this empty directory as the mount point)

Running and Testing Your FUSE server



Automatic Testing!!!

- Download the file “hw6.zip”
- Extract the zip in your VM
- in the following files

./answer/1.txt

./answer/2.txt

./answer/5.txt

change “nctuos” to your user account name

```
/home/nctuos/Documents/tarfs
/home/nctuos/Documents/tarfs/dir1/dir2
/home/nctuos/Documents/tarfs
./testcase/1.txt: line 7: cd: largefiles:
/home/nctuos/Documents/tarfs/dir
```

- Run the script: ./demo.sh <pathname of your FUSE server>

Testing Results

```
george@ubuntu:~/Downloads/Documents$ ./demo.sh hw6.out
===== basic case 1 =====
Your answer is correct
===== basic case 2 =====
Your answer is correct
===== basic case 3 =====
Your answer is correct
===== basic case 4 =====
Your answer is correct
===== bonus case 5 =====
Your answer is correct
===== bonus case 6 =====
Your answer is correct
======= Summary =======
[Correct]: 1 2 3 4 5 6
```

All pass

```
george@ubuntu:~/Downloads/Documents$ ./demo.sh hw6.out
===== basic case 1 =====
Your answer is wrong
===== basic case 2 =====
Your answer is wrong
===== basic case 3 =====
Your answer is correct
===== basic case 4 =====
Your answer is correct
===== bonus case 5 =====
Your answer is wrong
===== bonus case 6 =====
Your answer is correct
======= Summary =======
[Correct]: 3 4 6
[ Wrong ]: 1 2 5
```

Some errors

Mandatory: 1,2,3,4
Bonus: 5,6 (see later slices)

Remarks

- If you get a broken mount point during testing, use the following command to force unmount
  - sudo umount -l <your_mount_point>
- Do not use external library to parse tar files; parse on your own!
- Do not untar files from test.tar and copy them to the mount point… this is cheating!!!

Bonus (+10 pt)

- TAR archive will not perform in-place update, instead, it will append the new version of the updated file at the end of archive.
- There will be multiple files with the same file name in the bonus tar file. (Differentiated by last-modify-time)
- You can see the files using tar command

```
$ tar --list --verbose --file=test.tar
-rw-rw-r-- useraaa/useraaa 19 2019-11-26 16:08 blue.txt
-rw-rw-r-- useraaa/useraaa 11 2019-11-26 16:08 red.txt
-rw-rw-r-- useraaa/useraaa 12 2019-11-26 15:58 yellow.txt
-rw-rw-r-- useraaa/useraaa 13 2019-11-26 16:09 blue.txt
-rw-rw-r-- useraaa/useraaa 23 2019-11-26 16:14 blue.txt
-rw-rw-r-- useraaa/useraaa 22 2019-11-26 16:14 red.txt
```

Bonus (Test cases 5 and 6)

- Your FUSE server should report the newest version only

```
$ ls -l
total 0
-rw-rw-r-- 0 useraaa useraaa 23 Nov 26 16:14 blue.txt
-rw-rw-r-- 0 useraaa useraaa 22 Nov 26 16:14 red.txt
-rw-rw-r-- 0 useraaa useraaa 12 Nov 26 15:58 yellow.txt
$ cat blue.txt
sea
sky
cloud
dory
```

Header of your .c or .cpp

/*

Student No.: 31415926

Student Name: John Doe

Email: xxx@yyy.zzz

SE tag: xnxcxtxuxoxsx

Statement: I am fully aware that this program is not supposed to be posted to a public server, such as a public GitHub repository or a public web page.

*/

Credits

- 吴雅柔 rosewu911@gmail.com helped design this assignment