File Systems

A file system needs to satisfy these general requirements.

- We need some way of storing information
  - Independently from a running program, so it can be used at later time
  - Permanent / Persistence
  - Non-volatile storage, so it can be shared with other programs or users
- An infinite variety of data is to be stored
  - The more information the OS knows about the data the more it can facilitate use of the data. E.g. binary files, textual files, executable files
- Naming the data
  - The data needs to be stored and retrieved easily. We need a way to name the data.
  - We must then be able to locate the data using its name.
  - User can easily find, examine, modify, etc data

File System Operations

Create (name)

- Need to specify the **name**, the **file type**
- Create a file descriptor for the file including name, location on disk, and all file attributes
- Specify the size of the file

Delete (name)

- Remove the file. Free the dist blocks used by the file

Move (name)

- Moving a file is performed depending on the before and after locations: e.g., copy + delete the original
- If both locations are on the same device, change information about the file instead

File System Operations

Copy

- Most file systems preserve attributes, e.g., last modified times when a copy is made.
- This way a file can be last modified before it was created

Change attributes

- Some of these should be changeable, others should be secured

```
(base) [mfchiang@ my_solution]$ stat source/one
16777220 12888855230 -rw-r--r-- 1 mfchiang staff 0 2 "Sep 10 18:33:00 2020" "Sep 10 18:32:47 2020"
"Sep 10 18:32:47 2020" "Aug 25 10:48:36 2020" 4096 8 0 source/one
(base) [mfchiang@ my_solution]$ stat source/two
16777220 12888855256 -rw-r--r-- 1 mfchiang staff 0 2 "Sep 10 18:33:00 2020" "Sep 10 18:32:52 2020"
"Sep 10 18:32:52 2020" "Aug 25 10:49:06 2020" 4096 8 0 source/two
```

File System Operations

Read

- The operation works on the files contents
- We need to know where the information is on the device
- Must specify what data to read, how much, and where to put it

- Sequential access
  - Data is retrieved in the same order it is stored
  - There is a current position pointer somewhere
- Direct / Random access
  - The read specifies exactly where it wants to get the data from
  - It could be a byte offset, or or a record number

Write

- Very similar to read but commonly requires the allocation of extra space
- The program “seeks” to the new position
- If a process writes to the empty space then real blocks are allocated and written to

Assignment 2

Filesystem in Userspace (FUSE)

Filesystem in Userspace (FUSE)

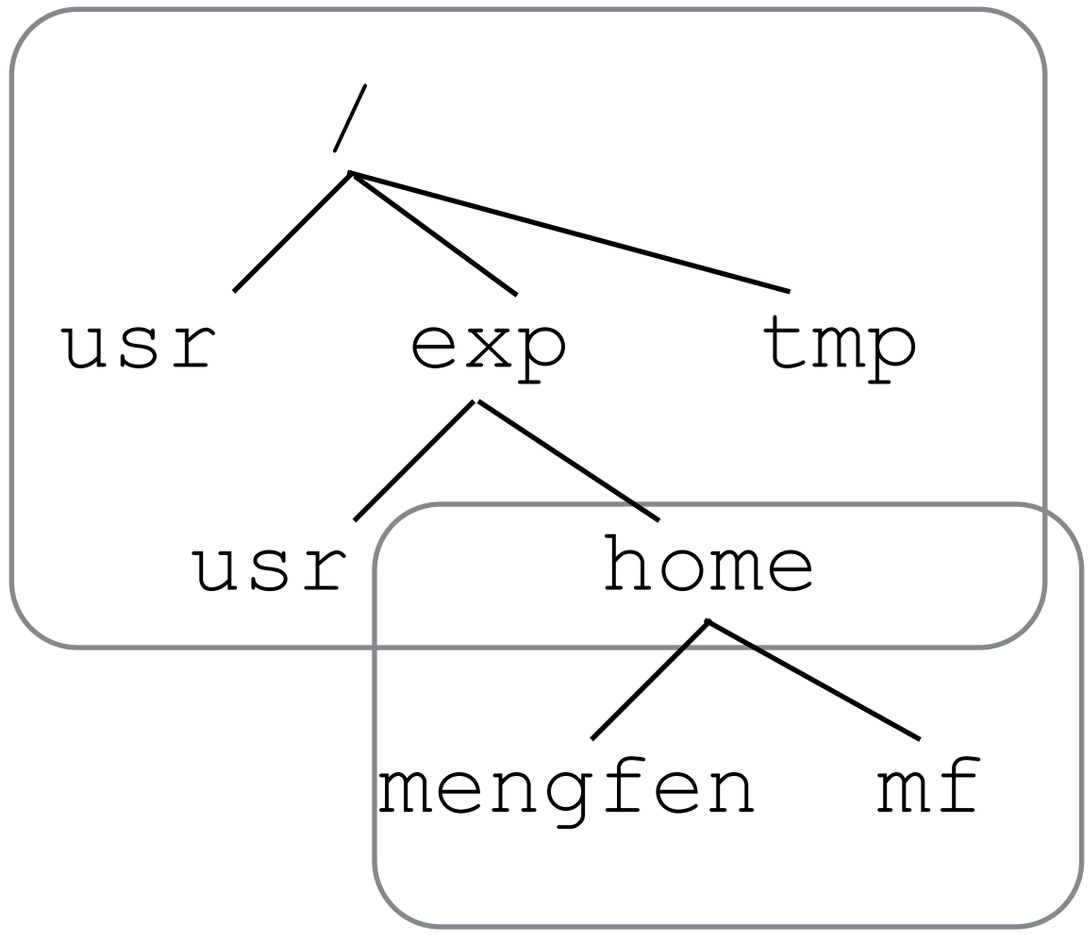
- Linux has a library which allows file systems to be written and used without root privileges - libfuse

- The library is used in conjunction with a kernel module which provides the privileged operations

- The user's file system gets mounted on to an existing directory

Mounting a File System

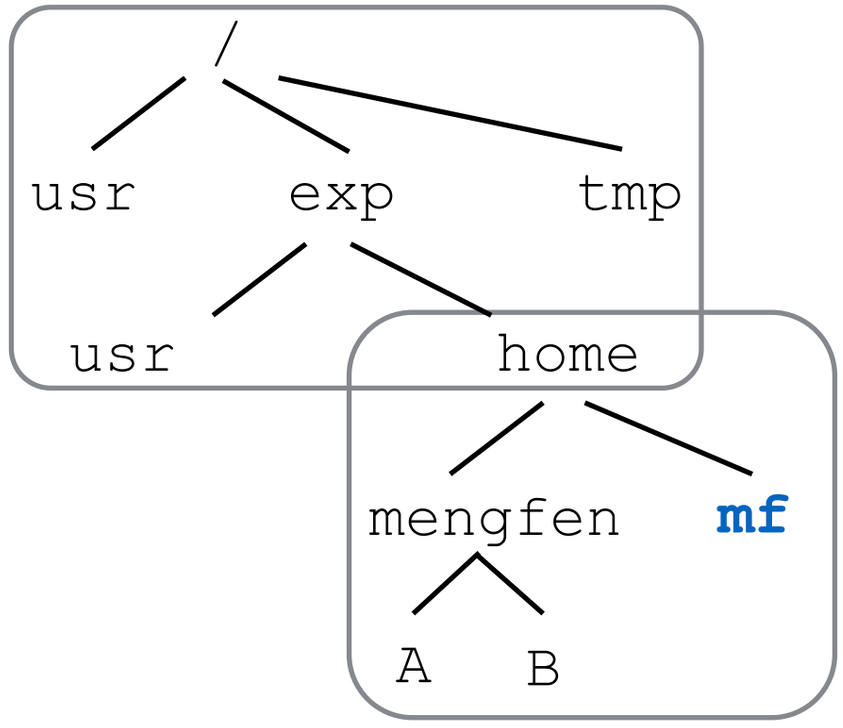
- A file system (Unix) can be thought of as a device or partition which can be connected into the standard hierarchical file system starting from the root "/".



- Multilevel Directories:
  - Tree structured name space

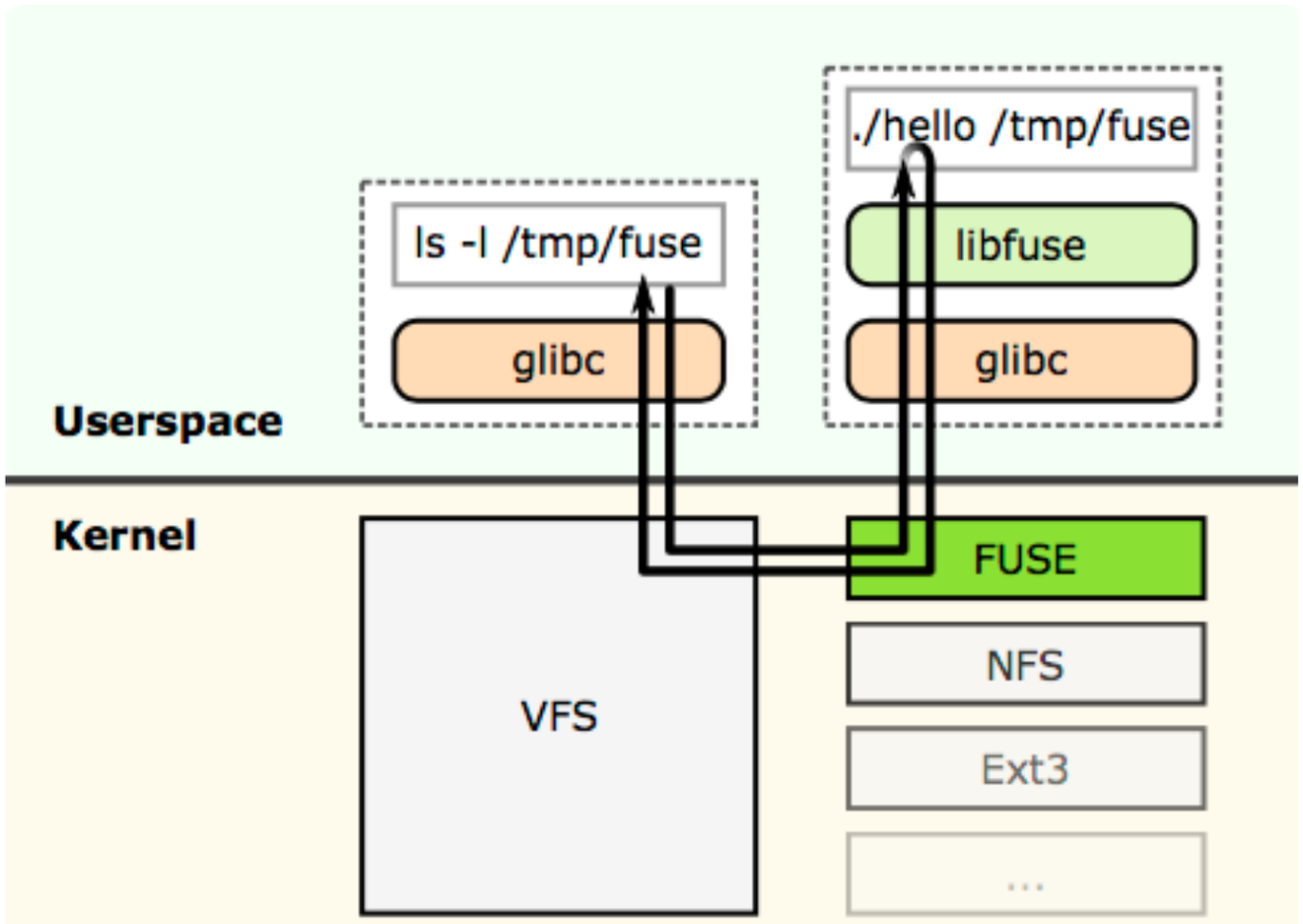
Mounting a File System

- We need a mount point: a directory where we plant the file system
  - Source: `/exp/home/mengfen`
  - Mount point: `/exp/home/mf`



- Any files which are already in the mount point directory then disappear

- They get replaced by the files/directories in the new file system



How it works

from Wikipedia

`./hello` is mounted on `/tmp/fuse`

All the operations on `/tmp/fuse` and its files can be handled by user level code.

Setup

- Ubuntu in the labs or on your own machine
  - Virtual machines, macOS or Windows Subsystem for Linux version 2
  - Ubuntu 18.04 image has been made available at FlexIT (flexit.auckland.ac.nz)
    - Sign in to SSO which will take you to the sign-in page for FlexIT
    - Look for the Ubuntu 18.04 desktop icon

- The Markers will use Ubuntu in the labs

Setup

- Download fuse.py, memory.py and passthrough.py from the A2 files section on Canvas
  - fuse.py is from
    - https://github.com/fusepy/fusepy/blob/master/fuse.py
  - memory.py is from
    - https://github.com/fusepy/fusepy/blob/master/examples/memory.py
  - passthrough.py is from
    - https://github.com/skorokithakis/python-fuse-sample

- You can also get examples from the github site at
  - https://github.com/fusepy/fusepy/tree/master/examples

OMG Python

- Much (much) simpler to do this assignment in Python
- If you don't know how to do something in
  - Ask one of the tutors/lecturers for help
  - Forum: feel free to ask on Piazza or Canvas
  - Python documentation: https://docs.python.org/3.6/
  - StackOverflow, Google
- The standard python 2 or 3.6 on Ubuntu works

passthrough.py

- Neatly enapsulates the methods you may need to override (not all of them)
- Two types of methods
  1. filesystem methods which deal with directories and files
  2. file methods which deal with the contents of files

Logging

- There is some python magic which logs information to the screen as the file system works.

```
class LoggingMixIn:
    log = logging.getLogger('A2')

    def __call__(self, op, path, *args):
        self.log.debug('-> path:%s %s%s', path, op, repr(args))
        ret = '[Unhandled Exception]'
        try:
            ret = getattr(self, op)(path, *args)
            return ret
        except OSError as e:
            ret = str(e)
            raise
```

Part1

- Setup
  - Put `fuse.py, passthrough.py` and `a2fuse1.py` in the same directory
  - Create a directory called "`source`"
  - Download files "`one`", "`two`", and "`three`" from Canvas
  - Create a directory called "`mount`"
- Run "`python a2fuse1.py source mount`"
  - Observe logs and highlight the main functionality, for example

```
DEBUG:fuse.log-mixin:-> getattr /newfile (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1599037397.463398, 'st_ctime':
1599037379.543504, 'st_gid': 20, 'st_mode': 33188, 'st_mtime':
1599037379.543504, 'st_nlink': 1, 'st_size': 12, 'st_uid': 501}
```

Gets attributes for "newfile".

- To stop
  - "`fusermount -u mount`" (Ubuntu) or `umount mount`" (macOS)

Part2

- In the `Memory` class, explain what each method does
  - `__init__`, `getattr`, `readdir`, `open`, `create`, `unlink`, `write`, `read`

- Example: `__init__`

```
def __init__(self):
    self.files = {}
    self.data = defaultdict(bytes)
    self.fd = 0
    now = time()
    self.files['/'] = dict(st_mode=(S_IFDIR | 0o755), st_ctime=now,
                           st_mtime=now, st_atime=now, st_nlink=2)
```

  - Creates an empty dictionary self.files for the files, using the path names as the keys. Each value in the dictionary will be another dictionary.
  - self.data is a dictionary for the files' data. The path names are the keys, the values are the data of that file.
  - Sets the starting value for the file descriptors, these are going to be used as unique file identifiers.
  - Grabs the current time and sets the file attributes for the root of this file system. It is a directory, with creation, modified and accessed times set to now. It has two links.

Part3 – Requirement 1

- Setup
  - Create two directories, `source1` and `source2`
  - Create file `one` in `source1`
  - Create file `two` in `source2`
  - Make sure that `mount` is initially empty

- Enable your FUSE to mount two source directories into a mount point
  - Override the "`_full_path`" method to add logic to return multiple source directories
  - Override the "`readdir`" function to add logic s.t.
    - "`ls`" command in your `mount` can list the content of both `source1` and `source2`
  - Override the "`main`" method because we need an extra parameter
- Run "`python a2fuse2.py source1 source2 mount`"

Part3 – Requirement 2

- Override the File methods to create a totally cached file system
- When a file is opened read all of the contents into memory
- Any reads are extracted directly from memory with no disk access
- Any writes go to memory
- When the file is closed the changes get written to disk

Submission

- Canvas submission system to submit your assignment
  - Zip together A2.txt and a2fuse2.py
- Extra marks
  - 1 mark for including your name and login in both files
  - 1 mark for any files created by the file system having the correct user and group ids
- Contact Info
  - Lecturer (Meng-Fen): meng.chiang@auckland.ac.nz
  - Tutor (Timo): tvan508@aucklanduni.ac.nz
  - Tutor (Luman): lwan353@aucklanduni.ac.nz