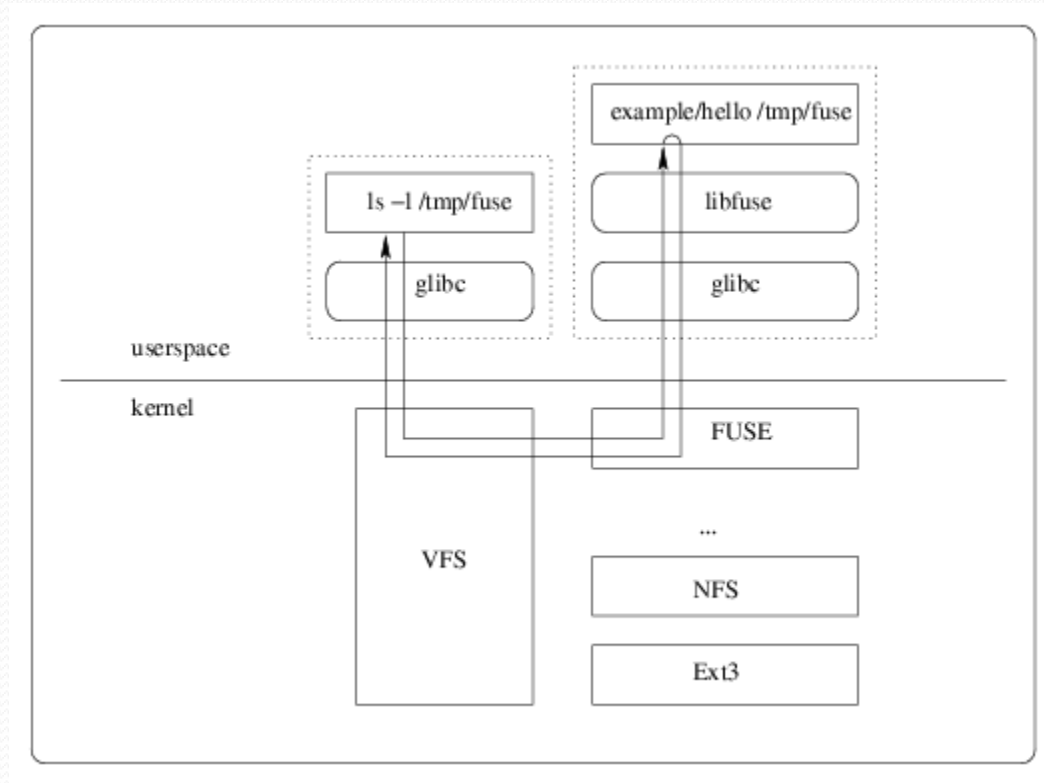


# FILESYSTEM IN USERSPACE (FUSE)

BLG413E – System Programming, Practice Session 4

# Filesystem in userspace (FUSE)

- Simple installation (no need to patch or recompile the kernel)
- Secure implementation
- Usable by non privileged users (via userspace-kernel interface)



# FUSE Development

- **Install:** libfuse-dev
- **To use FUSE library functions:** #include <fuse.h>
- **Main function of FUSE:** fuse\_main(argc, argv, op, user\_data)

## **#define fuse\_main ( argc, argv, op, user\_data )**

Main function of FUSE.

This is for the lazy. This is all that has to be called from the main() function.

This function does the following:

- parses command line options (-d -s and -h)
- passes relevant mount options to the **fuse\_mount()**
- installs signal handlers for INT, HUP, TERM and PIPE
- registers an exit handler to unmount the filesystem on program exit
- creates a fuse handle
- registers the operations
- calls either the single-threaded or the multi-threaded event loop

Note: this is currently implemented as a macro.

### **Parameters:**

<i>argc</i>	the argument counter passed to the main() function
<i>argv</i>	the argument vector passed to the main() function
<i>op</i>	the file system operation
<i>user_data</i>	user data supplied in the context during the init() method


### **Returns:**

0 on success, nonzero on failure

# Hello world in FUSE

- See hello.c
- **Compiling:** gcc hello.c -o hello -D\_FILE\_OFFSET\_BITS=64 -lfuse
  - -lfuse: link FUSE
  - -D\_FILE\_OFFSET\_BITS=64: force all file access calls to use the 64 bit variants. Not setting -D\_FILE\_OFFSET\_BITS=64 would result in different sized types in several structures and function calls. Moreover, FUSE requires to set -D\_FILE\_OFFSET\_BITS=64.

# Hello world in FUSE

- **Mounting:** `./hello <mount_dir>`
  - `grep hello /etc/mtab`  lists all currently mounted file systems along with their initialization options
  - `cat <mount_dir>/hello`
- **Unmounting:** `fusermount -u <mount_dir>`
  - `grep hello /etc/mtab`
- **Running in debug mode:** `./hello -d <mount_dir>`

# Read Only File System (ROFS)

- See rofs.c
- **Compiling:** `gcc rofs.c -o rofs -Wall -ansi -W -std=c99 -g -ggdb -D_GNU_SOURCE -D_FILE_OFFSET_BITS=64 -lfuse`
  - -Wall: enable all basic compiler's warning messages
  - -ansi: enforce ANSI C standards
  - -W: enable some extra warning flags that are not enabled by -Wall
  - -std=c99: use c99 standards
  - -g: produce debugging information in the operating system's native format
  - -ggdb: produce debugging information for use by GDB
  - -D\_GNU\_SOURCE: use GNU standards

# Read Only File System (ROFS)

- **Mounting:** `./rofs test1 test2`

(It is better to use ROFS in debug mode as it gives problems when it is mounted normally)

- `ls -l`
- `grep rofs /etc/mtab`
- **Unmounting:** `fusermount -u test2`
  - `ls -l`
  - `grep rofs /etc/mtab`
- **Running in debug mode:** `./rofs -d test1 test2`



# Read Only File System (ROFS)

- Try to create a file/directory under test1 and see what happens under test2
- Try to create a file/directory under test2
- Try to modify a file under test2