EEL---5737 Principles of Computer System Design Homework #1

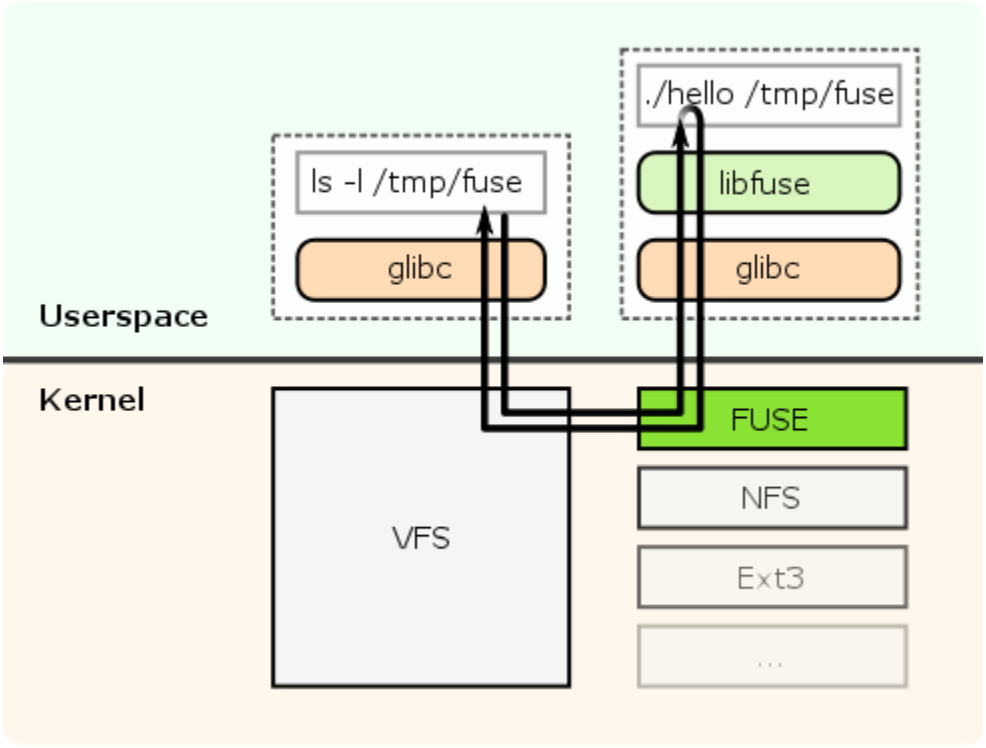
Nirali Patel

Part A

A.1 Summarize: what is FUSE, why it is useful, and how is it architected.

Filesystem in User space(FUSE) is a software interface that allows an unprivileged user to create its own filesystem without editing the Kernel code. It exports filesystems from user programs to OS kernels. The operating system segregates virtual memory into kernel space and user space. A user outside the kernel space would be able to use create and run application where fuse provides a bridge to actual kernel interfaces. FUSE is used for writing virtual file systems. The virtual files do not actually store data, but are translations or view of any existing file system. It can also provide remote access to files located on different hosts for any task.

FUSE ARCHITECTURE:



As the diagram show, the operating system segregates virtual memory into kernel space and user space. It shows how calls execute from the user space via kernel back to the application running. The program running in user space or specifically where the fuse system is being mounted. The diagram shows 'hello world' file mounted on fuse. Related system calls are passed on to kernel. The VFS (Virtual File System) transfers call to Fuse and then the call uses the functions, libraries or references of libfuse. Libfuse works with the binaries that are created in main program and return a reply to kernel and in turn to main operation instruction.

References:

https://github.com/libfuse/libfuse

https://www.youtube.com/watch?v=C2FuPxyip2A&t=513s

https://en.wikipedia.org/wiki/Filesystem_in_Userspace

https://www.cs.nmsu.edu/~pfeiffer/fuse-tutorial/

A.2 What happens when you mount and unmount the FUSE file system?

Mounting

While mounting of a file, the name of the device file for file system is given along with the directory name. This directory will be the mounting point. A connection is created between file containing the mounting point and the file system which is to be mounted. The connection will remain until unmounting is done or daemon (on-going background functions) dies. After mounting the file system, kernel considers the mounting point as the top-level directory of the file systems when their path names are resolved.

Unmounting

Unmounting destroys the connection of file system and makes free the virtual storage space created by mounting the file. Unmounting can also occur automatically if the system shuts down.

A.3 Where in hardware (e.g. memory, disk) is the "hello world" string stored after the echo command runs? Through what abstraction (e.g. virtual memory, file system) is this content accessible? Explain.

"hello world" is stored in the memory after the echo command runs. It uses file system abstraction to access the content of the file system. We can get it back by initiating the path from root directory of file system in which our text file exists.

Part B

B.1 Select one of the system calls shown in the strace output and describe the following components of the system call you chose:

name, arguments, return value; caller, callee; layer of the caller (app, O/S), and layer of the callee.

The system call here is **Open ()**.

open ("/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3

name: open

arguments: "/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC

return value: 3

caller: ls

callee: open()

layer of caller: userspace

layer of callee: userspace

B.2 Describe in one sentence the purpose of the following commands: ls, touch, echo.

ls- ls commands list the contents of a directory.

touch- It is used to create one or more empty file and allows to manipulate the timestamps of any existing file.

echo- It is used to display text/string on standard output or write to a file.

B.3 For the third "strace" (which executes the "echo" command), which functions of the memory.py program are executed and, what is the return value for each function that is called.

Functions executed and their return value:

getattr()

Return Value: 'st_ctime': 1504494177.195423, 'st_mtime': 1504494177.195423, 'st_size': 6, 'st_atime': 1504494177.195423, 'st_nlink': 1, 'st_mode': 33204L

It will return the attributes of the file that is referred.

open()

Return Value: 4

getxattr()

Return Value: ‘ ’(Null)

truncate()

Return Value: None

write()

Return Value: 6 (Length of the String)

B.4 Referring to Figure 2.5 (p. 54), what are three important instruction references that are at play in this exercise? Describe one instance where a change of environment reference occurs.

ls, touch, echo are the instruction references used**.** The commands like ls, touch or echo run in userspace and for this command to be executed, it requires to switch to kernel and back to user space to use the libraries in libfuse. There it encounters an environment change.

B.5 Consider the first command “ls”. Inspect the “open()” system calls at the output of strace (ignore the other system calls). a.) Explain in one sentence, what does open() do? b.) Why are there several “open()” calls that refer to files in pathnames that begin with “/lib” and “/usr/lib”? c.) Which “open()” system call(s) result in call(s) to the FUSE file system?

open() system call

a.) open() : An open system call maps or establishes connection between the file given by the path name to the file descriptor which it returns upon success.

b.) There are many types of the open() calls that refer to /lib and /usr/lib.

The system call has some dependencies and for that the open system calls other dependent files. The system call relies on them to complete its execution.

c.) According to this open() system call, the FUSE file system is being called and composed.

open(".", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3

B.6 Consider the fourth command “cat”. Inspect the “open()” system calls at the output of strace (ignore the other system calls). a.) What is the absolute path name of the file you are creating in the context of the kernel’s file name resolver? b.) What is the absolute path name in the context of the FUSE name resolver?

The absolute path name in the context of the kernel’s file name resolver is

~/fusepy/fusemount/hello.txt

The absolute path name in the context of the FUSE name resolver is

/hello.txt

References:

http://www.youblisher.com/p/31627-fuse/

Principles of Computer System Design (By-Jerome. H. Saltzer and M. Frans Kaashock)

http://www.linux-mag.com/id/7814/

https://www.kernel.org/doc/Documentation/filesystems/fuse.txt

https://linux.die.net/man/3/open