# 

Contents

[Contents 1](#__RefHeading___Toc562_307036362)

[Introduction 1](#__RefHeading___Toc564_307036362)

[Project Goal 1](#__RefHeading___Toc566_307036362)

[System Information 2](#__RefHeading___Toc568_307036362)

[Packages and Tools 2](#__RefHeading___Toc570_307036362)

[Compiling and Running 2](#__RefHeading___Toc572_307036362)

[Design 3](#__RefHeading___Toc574_307036362)

[Integration 3](#__RefHeading___Toc576_307036362)

[Future Improvement 3](#__RefHeading___Toc578_307036362)

[Lack of overlap between this project and the course material 4](#__RefHeading___Toc580_307036362)

[Dependency on extraneous uncontrollable variables 4](#__RefHeading___Toc582_307036362)

[Utilizing a library that has poor programming practices 5](#__RefHeading___Toc584_307036362)

[Summary 6](#__RefHeading___Toc586_307036362)

## Introduction

Filesystems in Userspace, or FUSE, is a concept that allows software engineers to implement custom filesystems in Linux-based distributions without having to edit or manipulate kernel code. FUSE as a concept has been around since at least October 2002 and has since been merged into the Linux kernel since in version 2.6.14.

This project takes an existing Amazon S3-based FUSE implementation and wedges a RC4 enryption layer between the upload and download stages of synchronization.

## Project Goal

The goal of this project is to inject RC4 encryption and decryption between the upload and download layers of a S3-based FUSE filesystem implementation.

## System Information

**Kernel**

$ uname -a  
Linux archlinux 4.15.13-1-ARCH #1 SMP PREEMPT Sun Mar 25 11:27:57 UTC 2018 x86\_64 GNU/Linux

**CPU**

vendor\_id : GenuineIntel  
cpu family : 6  
model : 158  
model name : Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz  
stepping : 9  
microcode : 0x5e  
cpu MHz : 800.026  
cache size : 8192 KB

|  |  |
| --- | --- |
| Software | Version |
| libfuse | 3.2.2 |
| s3fs-fuse | 1.83 |
| openssl | 1.1.0.g-1 |

## Packages and Tools

The following packages and tools are used in this project:

* libfuse (FUSE reference implementation library)
* s3fs-fuse (Amazon S3 FUSE implementation)
* openssl (Encryption library used for RC4 encryption and decryption)
* docker and docker-compose (Container runtime to isolate the project)

## Compiling and Running

Assuming you have an AWS IAM User set up with S3 read/write access granted. Save the credentials in a .env file in the root of the project direcotory.

The .env file should have the following format:

AWS\_S3\_ID=<your-id-here>  
AWS\_S3\_SECRET=<your-secret-here>

Once your .env file is set up, run docker-compose up -d in the project root to build and run an isolated Ubuntu container with your source files bound to /src.

After them image is built and running, run docker-compose exec linux /bin/bash to open a tty session in the ubuntu container in the /src directory.

Once inside, execute ./init.sh <bucket-name> [<mount-dir>], which will write a passwd-s3fs file in /etc, and bind the <bucket-name> bucket to <mount-dir> (default mount directory = /s3mount).

After that, the FUSE filesystem is live and should begin syncing back and forth between the local filesystem and AWS S3 with RC4 encryption.

## Design

The totality of this project was completed in a single function and in around 20 lines of code. There are no specific design characteristics to note here.

The design of the s3fs-fuse library is one that I don't personally agree with, so I don't belive it is my job to try and justify it. Specifically, the code is *extremely* fragile being that almost the entire implementation relies of creating side-effects of globally-defined variables.

## Integration

I was unable to get s3fs-fuse to mount correctly on my operating system. I encountered a bug that prevented the addition or editing files synced from S3 because s3fs-fuse was "unable to set timestamps of the files".

Because I was unable to successfully test this, I also included a small test library that shows just the RC4 encryption/decryption functionality. This library, rc4.h and rc4.cpp, is located in /src and can be compiled and ran using make in the /src directory. After compiling, you may run it by executing ./bin/main.

## Future Improvement

I have scheduled a meeting with my advisor who will then schedule a meeting with the Computer Science Department Head to discuss this at greater length. As previously stated, I feel like this project as a whole is a disservice to students interested in Operating Systems at best and, at worst, one that could potentially turn students away from pursuing a career in operating systems.

With that being said, here are just a few of the the ways that I would recommend improving this project, broken down by the specific issues:

### Lack of overlap between this project and the course material

Based on the work required to complete this project, I think it's fair to say that 90% of the work involves implementing a small encryption layer to an existing codebase. In this project we use RC4 encryption from the openssl library specifically.

The problem with this is that even as of the time of writing this, we have not covered encryption, decryption, or general filesystem security at all. The security section of this course is scheduled *after* this project is due.

What's more, we have never had a lab that introduces us to the openssl library. We have never been introduced to RC4 encryption.

Finally, s3fs-fuse uses automake, which is an incredibly difficult concept to wrap your head around if not familar with it. At the time of writing, we have never covered even basic GNU Make, nor have we been introduced to it formally. I should note that lab 10 is a lab focused on GNU Make, but again, as a class, we've never been introduced to it formally, and Make is *much* simpler than automake.

#### Solution

I would recommend designing labs around the goals of the final project. We have had labs involving process schedulers, interprocess communication, system calls, etc. All of these things would have been an appropriate goal for the final project.

If the goal of the final project is encryption using openssl, it would have been nice to have been introduced to the openssl library in a lab session prior to having needed it in the final.

### Dependency on extraneous uncontrollable variables

This project requires that the student creates an AWS account and an associated S3 bucket in order to begin working. Additionally, this project has a strict dependency on the s3fs-fuse library, which is not written by any legitimate persons or agencies and has the potential to become abandoned at any given moment.

Allow me to unpack those two issues individually:

**Regarding the dependency on AWS:** This is a problem for most undergraduate-level students because AWS is inherantly difficult. So difficult, in fact, that certifications exist for people who are trained in using AWS. A student who is unfamilar with AWS IAM roles and permissions would have an extremely difficult time figuring out how to get that set up. This would be okay if there was some sort of introduction to AWS by the instructor, or if something having to do with AWS was rolled into the lab material, but unfortunately, both are non-existant.

**Regarding the dependency on s3fs-fuse:** Because this is ostensibly somebody's pet-project, that person could very well choose to stop rolling updates and abandon it altogether. This action would brick the entire final completely, which I find unacceptable.

#### Solution

The solution for this would require fundamentally rethinking this project from the ground up. There are many ways to approach this, but one could be to use a [bindfs-based](https://bindfs.org/) FUSE implementation, rather than one that operates over the network and requires a proprietary service like AWS.

If unfamilar, bindfs is a FUSE based implementation that "binds" (mounts) a directory on a system to another location *on the same system*. This is ideal because it would allow students to do the same things that are currently required using s3fs-fuse, but off the network.

### Utilizing a library that has poor programming practices

s3fs-fuse, though apparently popular, is not a library that I would recommend using for educational purposes. There are several things that I would declare "wrong" with the code that I think could potentially promote poor habits in student software engineers. The biggest issue that I have personnally is that the code is almost univerally dependent on side-effects. A majority of the library operates on a single file descriptor that floats around in the global scope. This is an incredibly fragile way of doing things.

Second, this code is extremely poorly documented. A handful of important methods do not have obvious names (for example, FdEntity::RowFlush), and nearly all of the classes, methods, and functions do not have accompanying documentation explaining the purpose, parameters, use, etc.

#### Solution

Limit dependencies on external codebases that don't meet that standards of Wayne State University.

## Summary

The rubric asks that I state what it is that I have learned from this project.

Unfortunately, I personally do not feel like I have accomplished or learned anything from this project. I am extremely dissapointed by the quality of this assignment and feel like my classmates and I deserve better than this.