# Role Based Access Control

...

Scott Linder, Justin Lanyon, Ryan DePrekel

# Introduction

What is the problem?
Design Choices
Overall Design
Implementation
Maintenance
Security
Summary

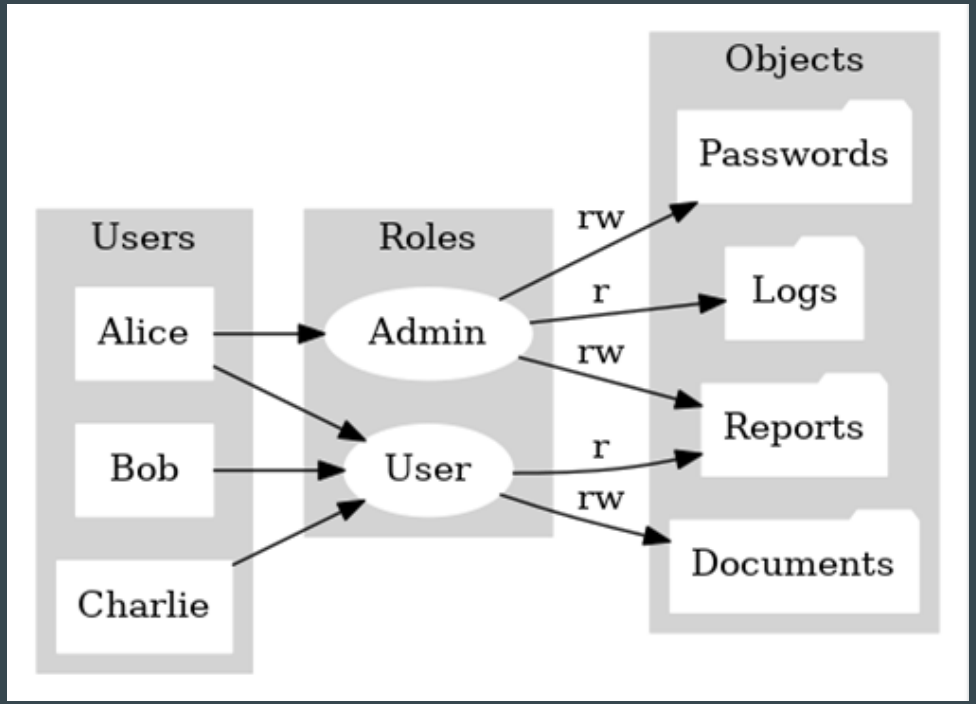
# Background

- Who: Dr. Carr, Computer Science Chair for Western Michigan University
- Problem: Academic learning of Access Control for Dr. Carr's Secure Software Design Class
- Desired Outcome: A file system that enforces mandatory access control (MAC) by using a Role Based Access Control (RBAC) schema.

# What is Access Control?

- Assigning rights (permissions) to users
- Examples
  - Unix permission bits
  - Access Control Lists (ACLs)
  - Role Based Access Control (RBAC)
- Two types
  - Discretionary
  - Mandatory

# Role Based Access Control



# Role Based Access Control

| | Admin | User |
|---|---|---|
| Alice | X | X |
| Bob | | X |
| Charlie | | X |

User to Roles Relationship

| | Passwords | Logs | Reports | Documents |
|---|---|---|---|---|
| Admin | rw | r | rw | |
| User | | | r | rw |

Role to Object Relationship

# Design Decisions

...

# Design Decision #1

File System in User Space (FUSE) vs Develop our own kernel module

## FUSE

- Efficient development
- Well documented
- Doesn't require modifying the kernel
- Better for student usage

## Our own kernel module

- Speed
- Security

# Design Decision #1

File System in User Space (FUSE) vs Develop our own kernel module

## FUSE

- Efficient development
- Well documented
- Doesn't require modifying the kernel
- Better for student usage

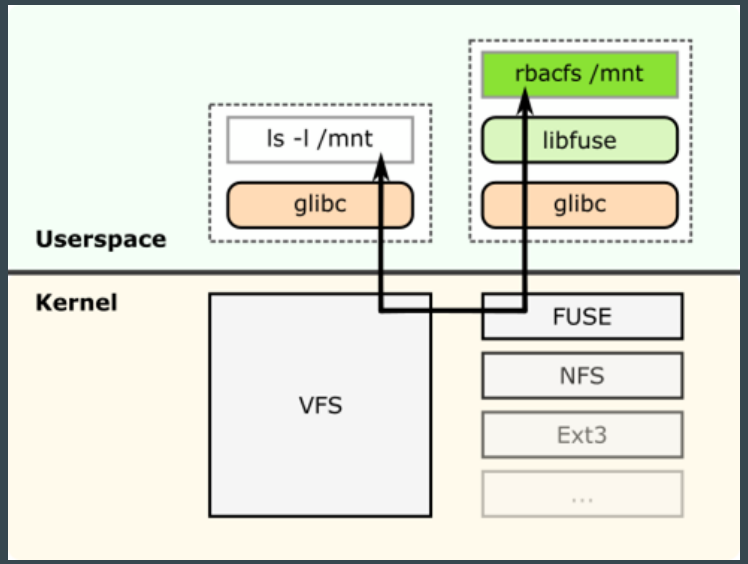
## Our own kernel module

- Speed
- Security

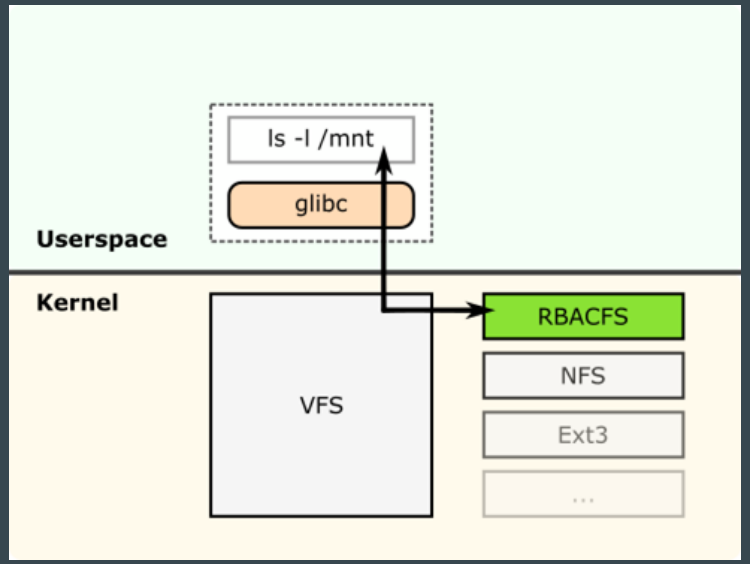
# Design Decision #1

File System in User Space (FUSE) vs Develop our own kernel module

## FUSE




## Our own kernel module




CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# Design Decision #2

Language choice: C vs Python

## C

- libfuse API compatible with C
- Client requested the RBAC to be implemented in C
- Performance
- Student familiarity

## Python

- RBAC grammar already defined by client in Python
- Ease of use
- Readability

# Design Decision #2

Language choice: C vs Python

## C

- libfuse API compatible with C
- Client requested the RBAC to be implemented in C
- Performance
- Student familiarity

## Python

- RBAC grammar already defined by client in Python
- Ease of use
- Readability

# Design Decision #3

Type of RBAC: Flat vs Hierarchical

## Flat

- Simpler implementation
- Easy for students to understand

## Hierarchical

- Simpler usage
- More complex

# Design Decision #3

Type of RBAC: Flat vs Hierarchical

## Flat

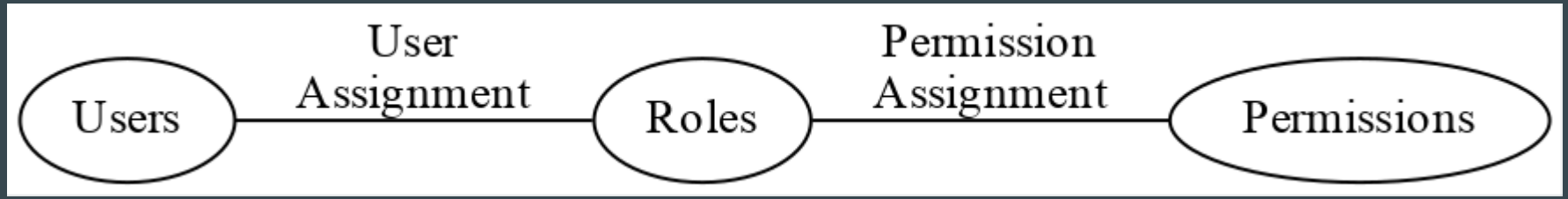
- Simpler implementation
- Easy for students to understand

## Hierarchical

- Simpler usage
- More complex

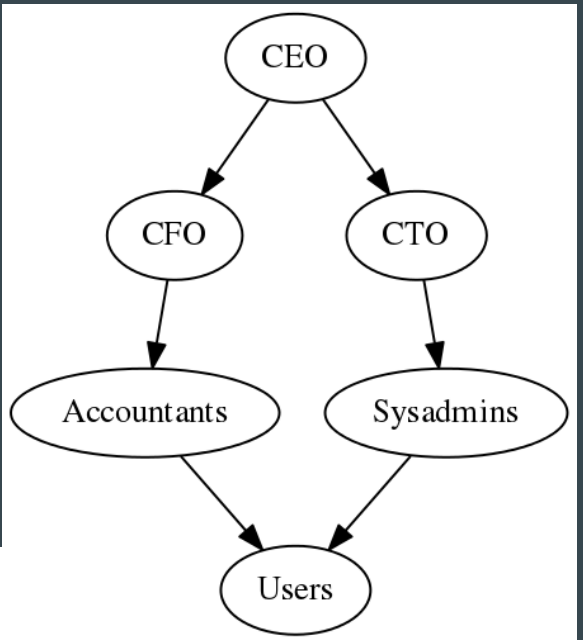
# Design Decision #3

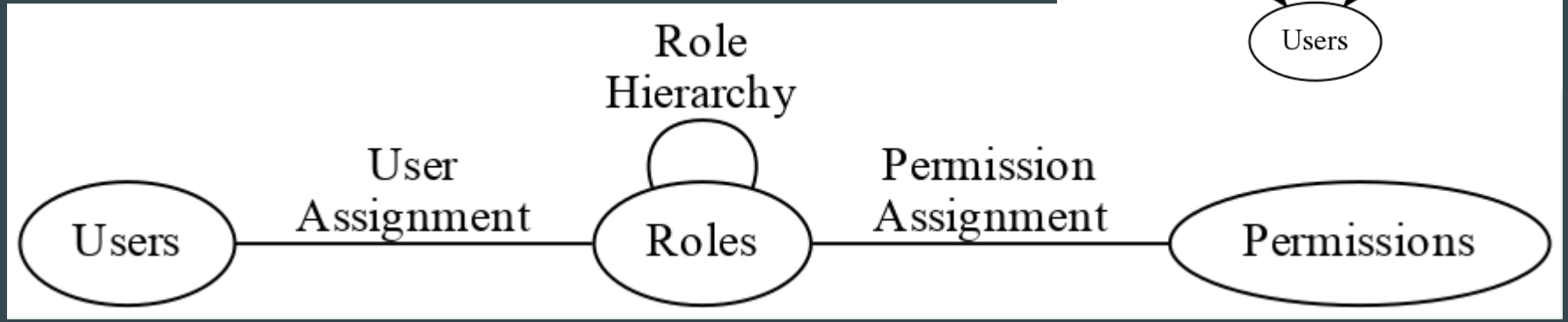
Flat RBAC



# Design Decision #3

Hierarchical RBAC



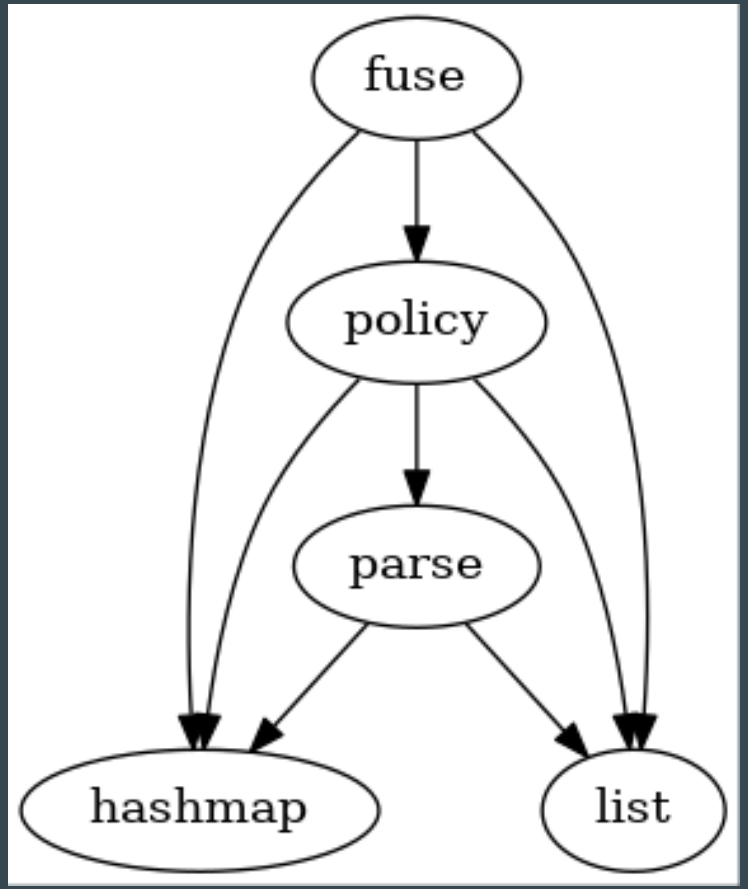
# Overall Design

...

# Overall Design

Each circle represents a module

Each arrow represents a dependency

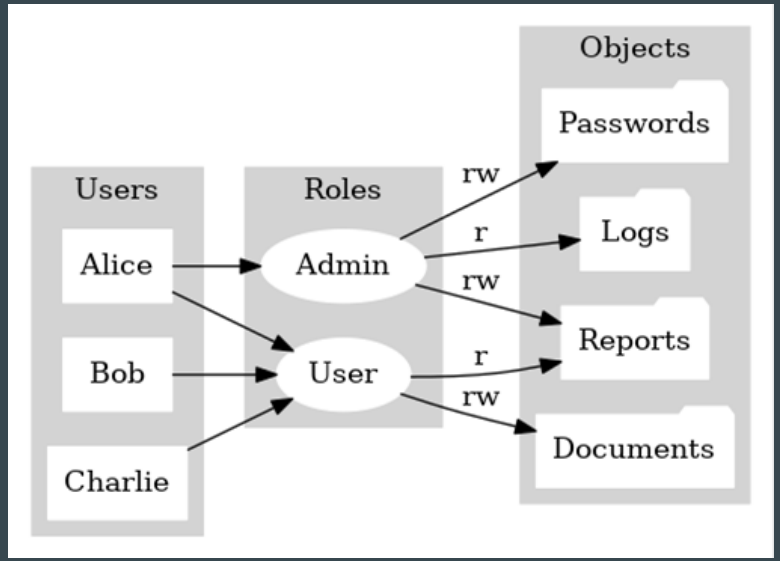


# Overall Design - Parsing

- RBAC grammar already defined by client in Python
- Example definition file:

```
user: admin,user alice
user: user       bob
user: user       charlie

object: admin r,w     /passwords
object: admin r    -r /logs
object: admin r,w  -r /reports
object: user  r    -r /reports
object: user  r,w  -r /documents
```



# Overall Design - Parsing

- Lexer and Parser defined with Flex and Bison, respectively
- Grammar in BNF:

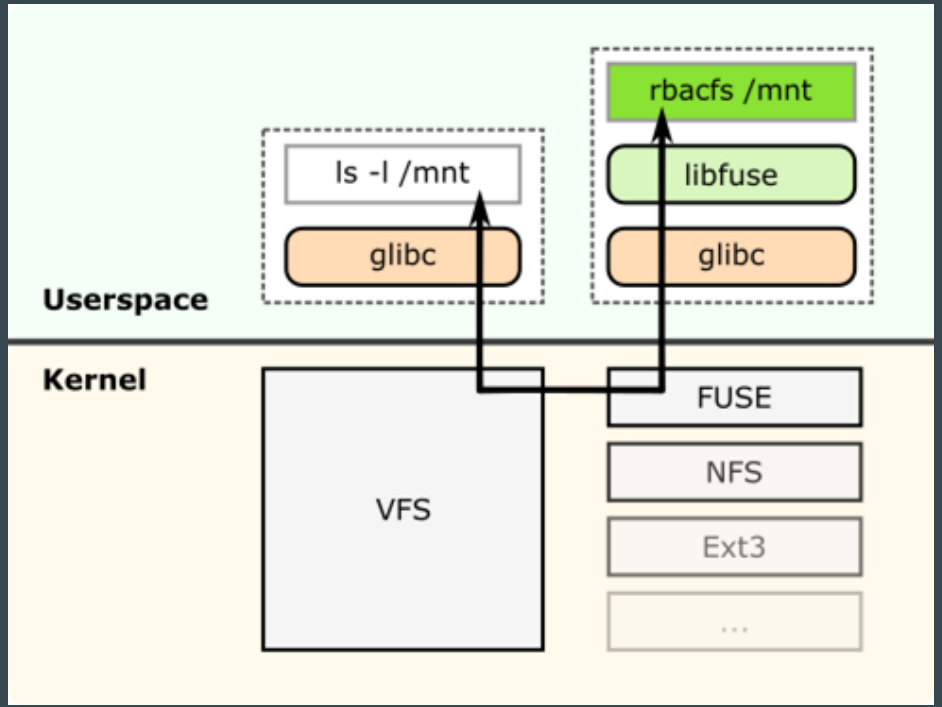
```
<input>                 ::= <def-list>
<def-list>              ::= <def> | <def-list> <def>
<def>                   ::= "user: " <id-list> <id-list>
                          | "object: " <id-list> <perm-list> <opt-recursive>
<path>
<id-list>               ::= <id> | <id-list> "," <id>
<perm-list>             ::= <perm> | <perm-list> "," <perm>
<opt-recursive>   ::= "-r" | ""
<path>                  ::= /[a-zA-Z0-9_.\-/]*
<id>                    ::= [a-zA-Z0-9_.\-/]+
<perm>                  ::= "r" | "w" | "x"
```

# Overall Design - Policy

- The parsed definition file traversed to produce policy data structure

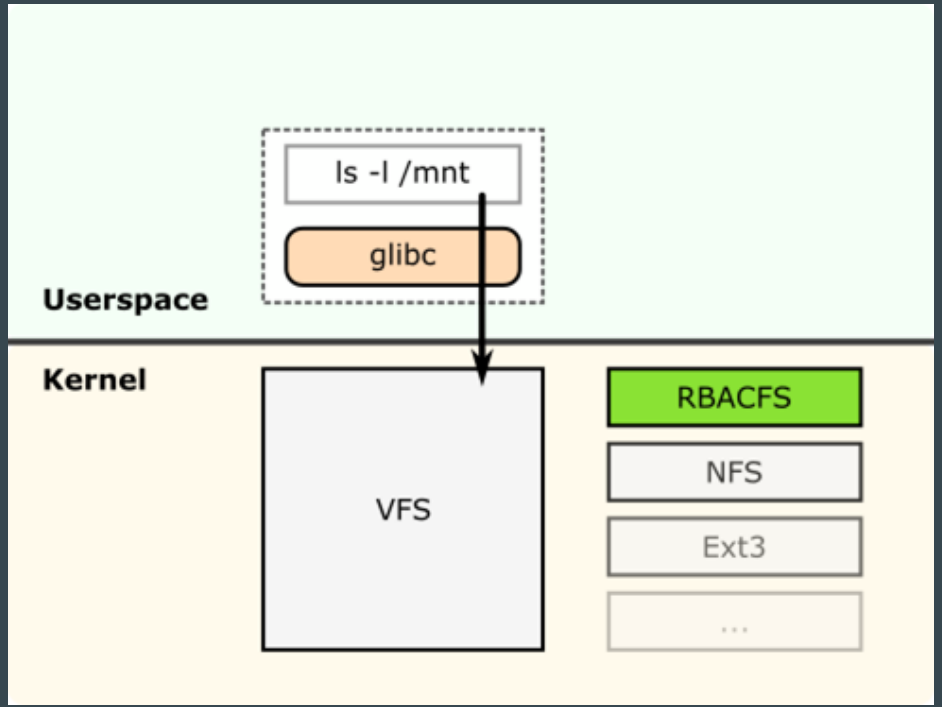
- Designed to efficiently service requests for:
  - Roles given a user
    - `user_roles["user"] = roles`
  - Permissions given an object and role
    - `obj_role_perms["obj"]["role"] = perms`

# Overall Design - FUSE



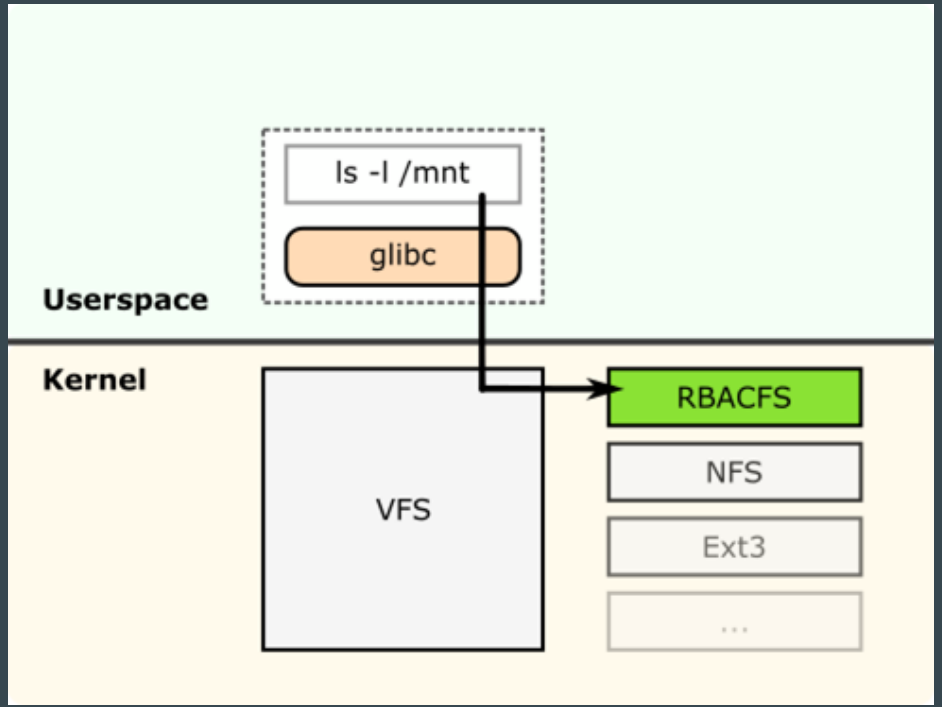

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# Kernel Module - Syscall trace



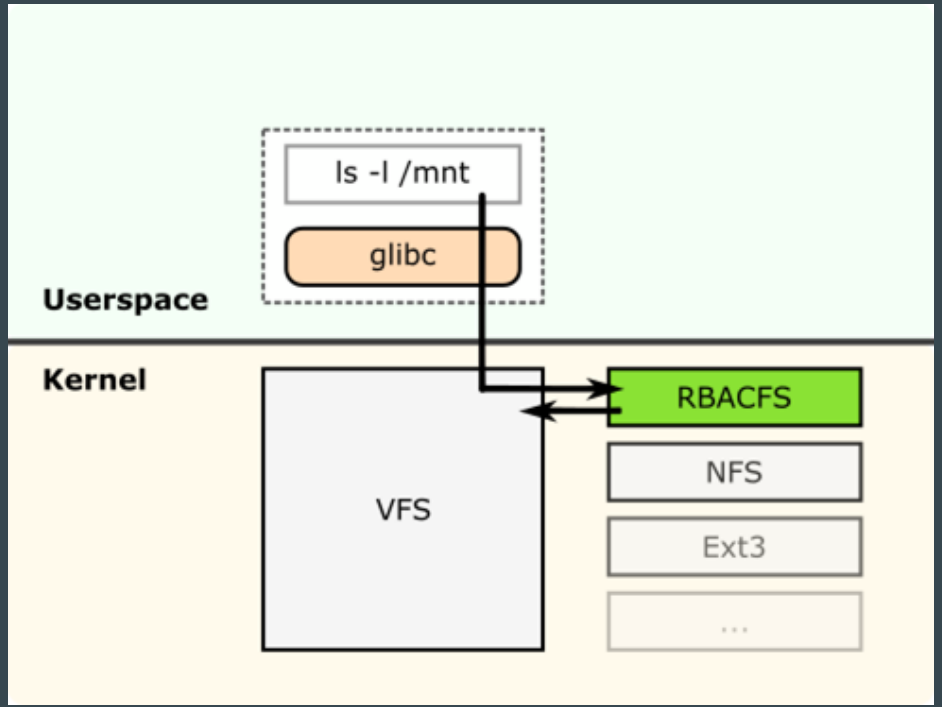

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# Kernel Module - Syscall trace



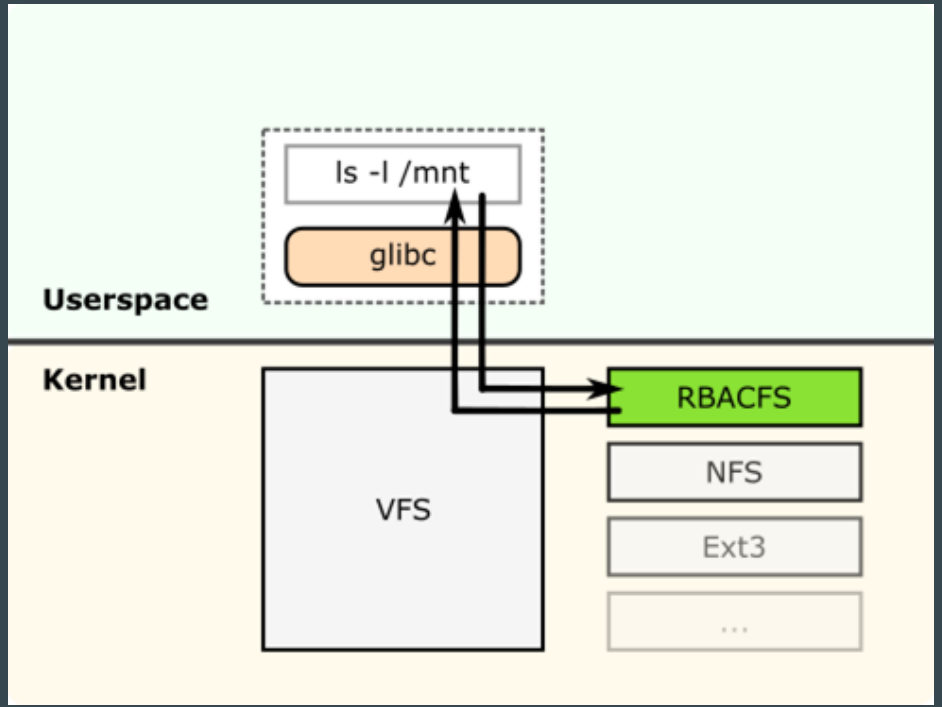

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# Kernel Module - Syscall trace



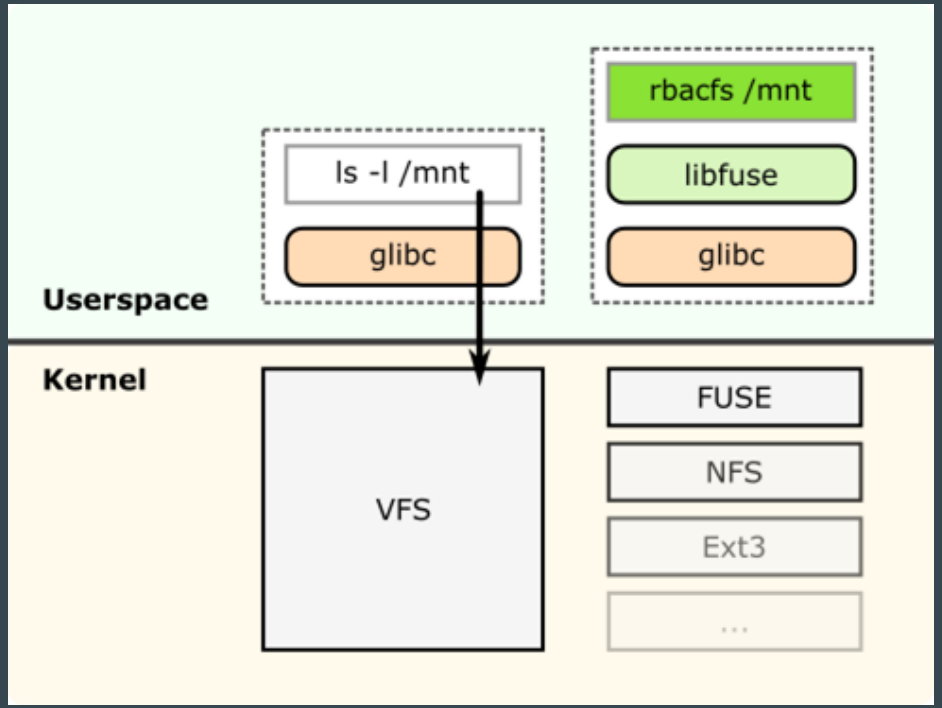

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# Kernel Module - Syscall trace



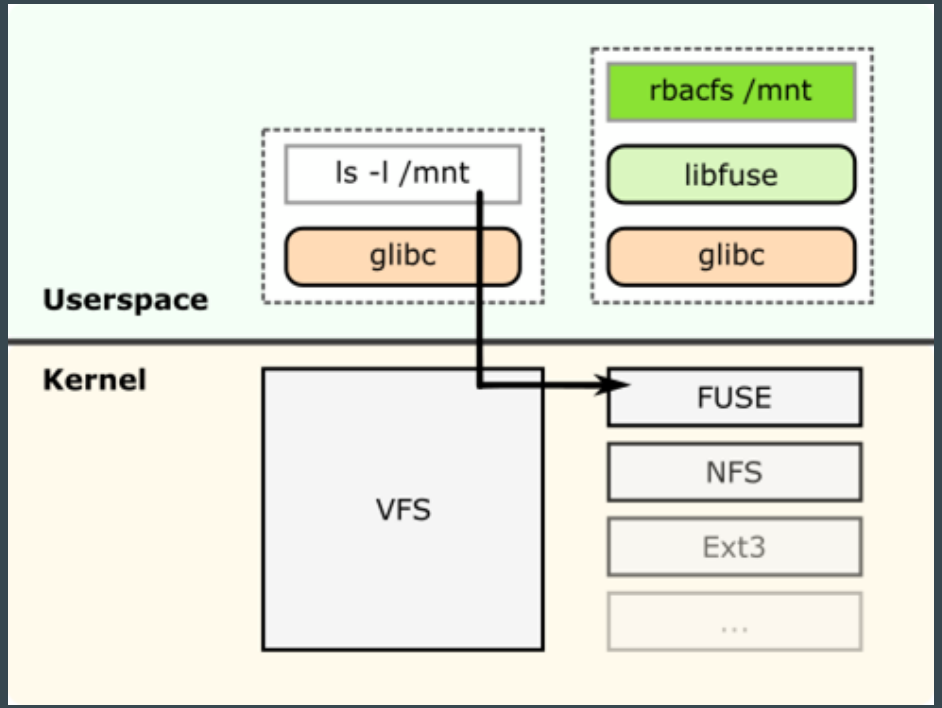

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# FUSE - Syscall trace



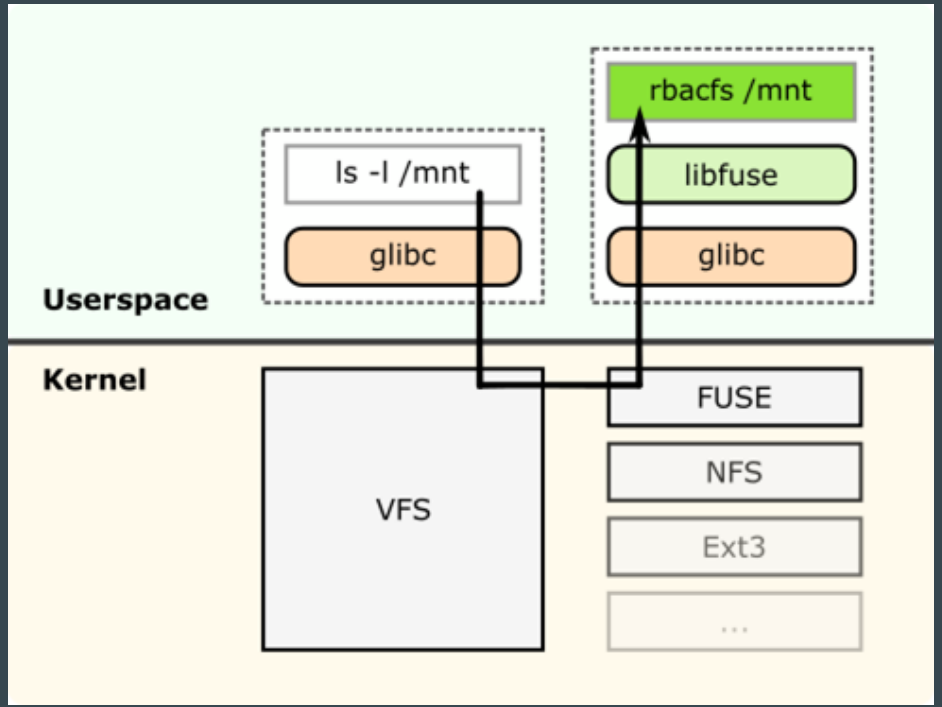

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# FUSE - Syscall trace



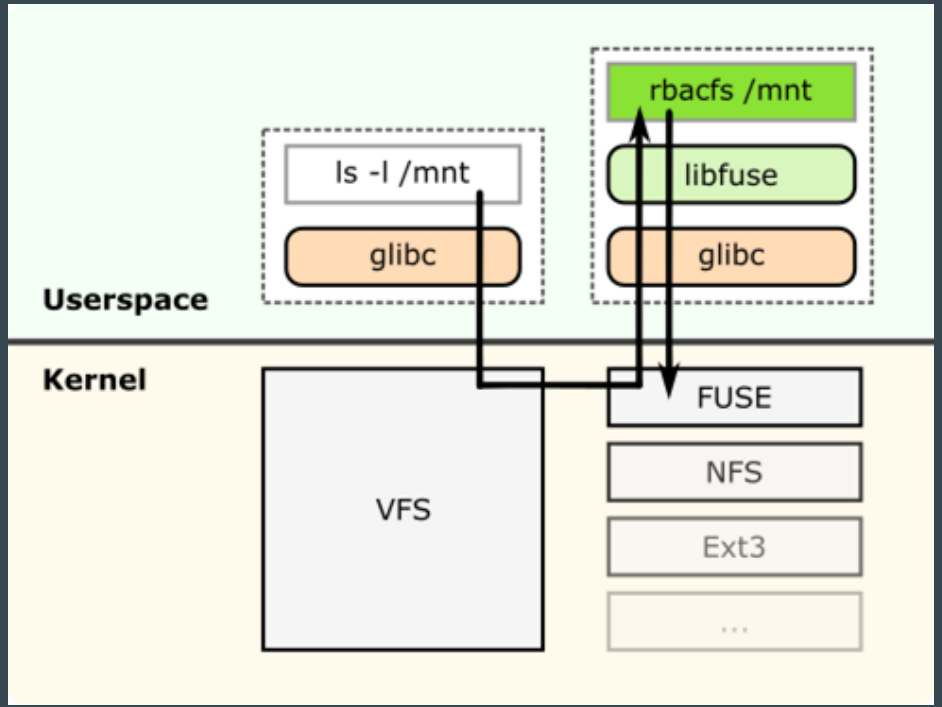

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# FUSE - Syscall trace



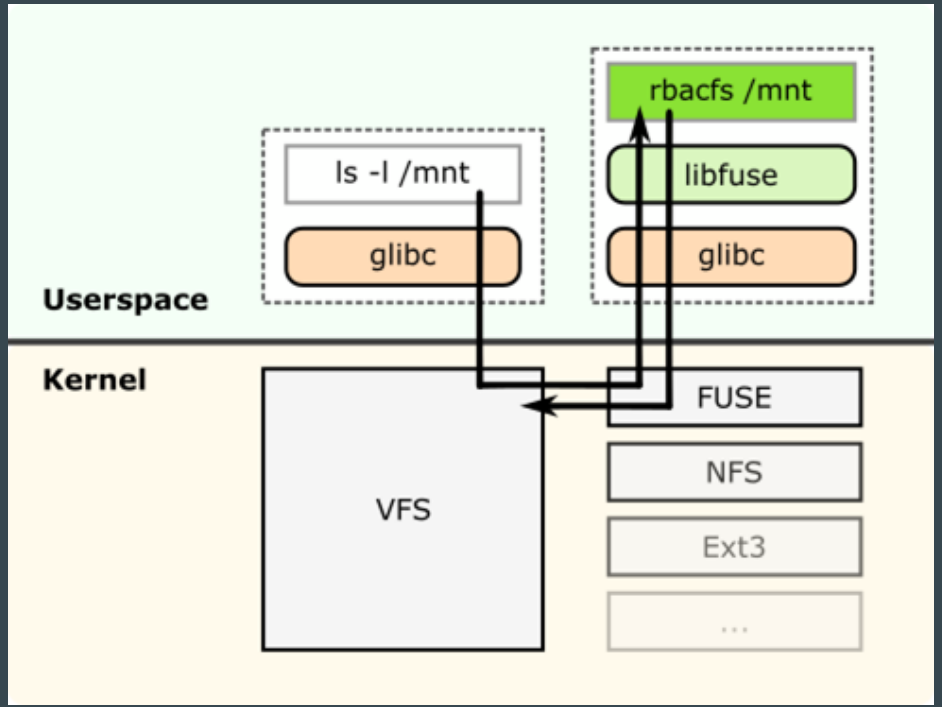

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# FUSE - Syscall trace



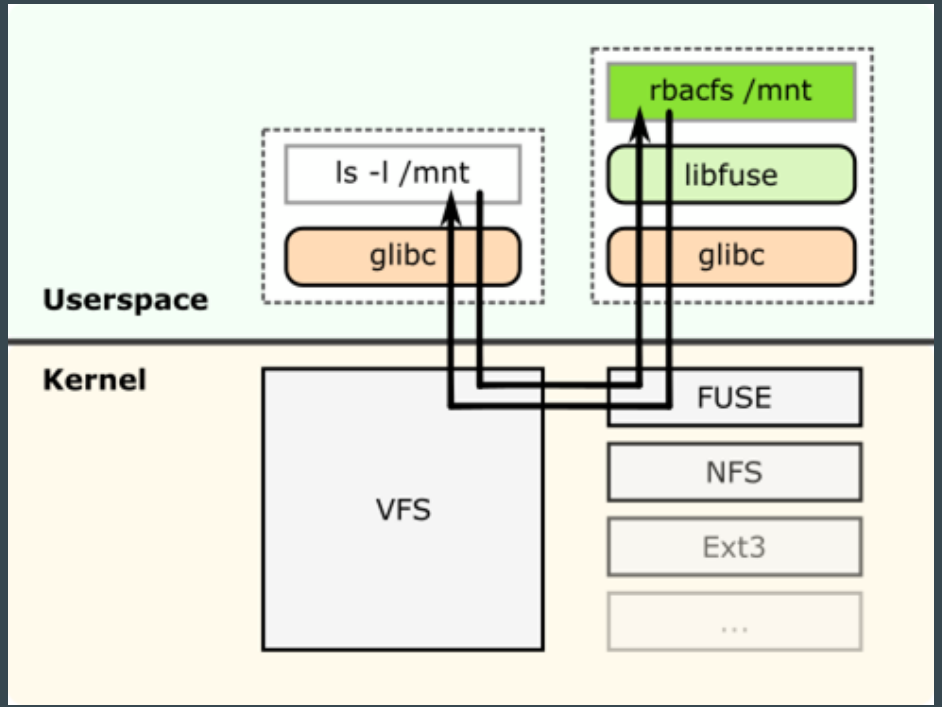

CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# FUSE - Syscall trace




CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# FUSE - Syscall trace




CC BY-SA 3.0 User:Sven https://commons.wikimedia.org/wiki/User:Sven with modifications

# Implementation

...

# Implementation

- Single binary: `rbacfs`
- usage: `./rbacfs -o allow_other mount root rbac.defs`
  - -o allow_other: allow other users to access the filesystem
  - mount: where to mount the filesystem
  - root: the root of the filesystem to shadow
  - rbac.defs: policy definition file
- Policy is immutable once the filesystem is mounted

# Implementation - Example

```
$ ls
mount/    rbac.defs    root/
```

# Implementation - Example

```
$ ls
mount/   rbac.defs   root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
```

# Implementation - Example

```
$ ls
mount/  rbac.defs  root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
$ ls root/
file1  file2
```

# Implementation - Example

```
$ ls
mount/  rbac.defs  root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
$ ls root/
file1  file2
$ ls mount/
```

# Implementation - Example

```
$ ls
mount/  rbac.defs  root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
$ ls root/
file1  file2
$ ls mount/
$ rbacfs mount/ root/ rbac.defs
```

# Implementation - Example

```
$ ls
mount/  rbac.defs  root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
$ ls root/
file1  file2
$ ls mount/
$ rbacfs mount/ root/ rbac.defs
$ ls mount/
file1 file2
```

# Implementation - Example

```
$ ls
mount/  rbac.defs  root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
$ ls root/
file1  file2
$ ls mount/
$ rbacfs mount/ root/ rbac.defs
$ ls mount/
file1 file2
$ cat mount/file1
file1
```

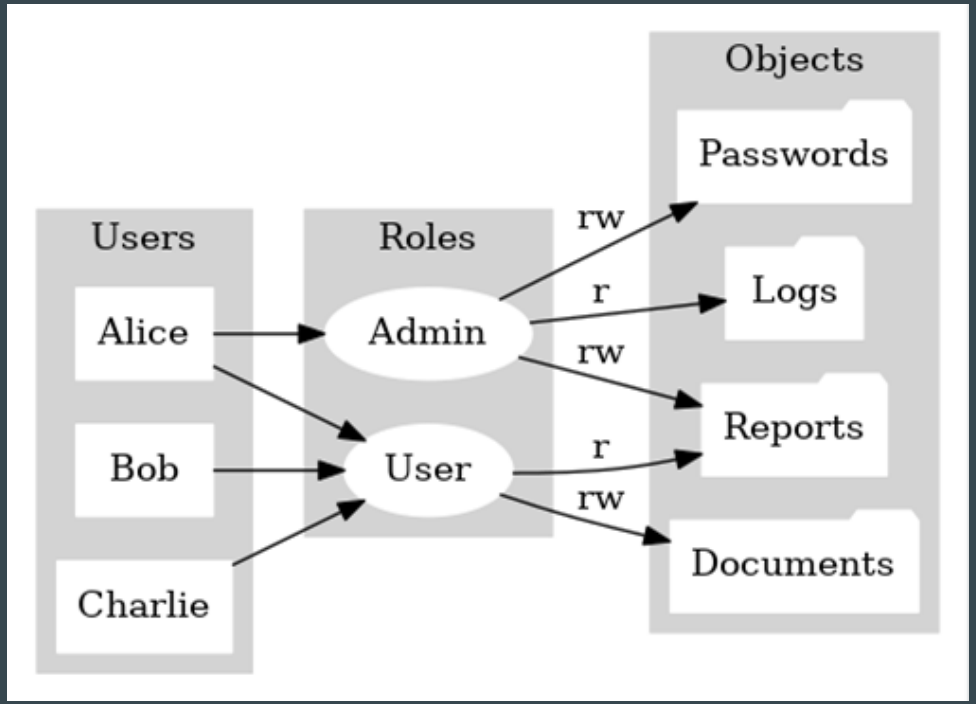
# Implementation - Example

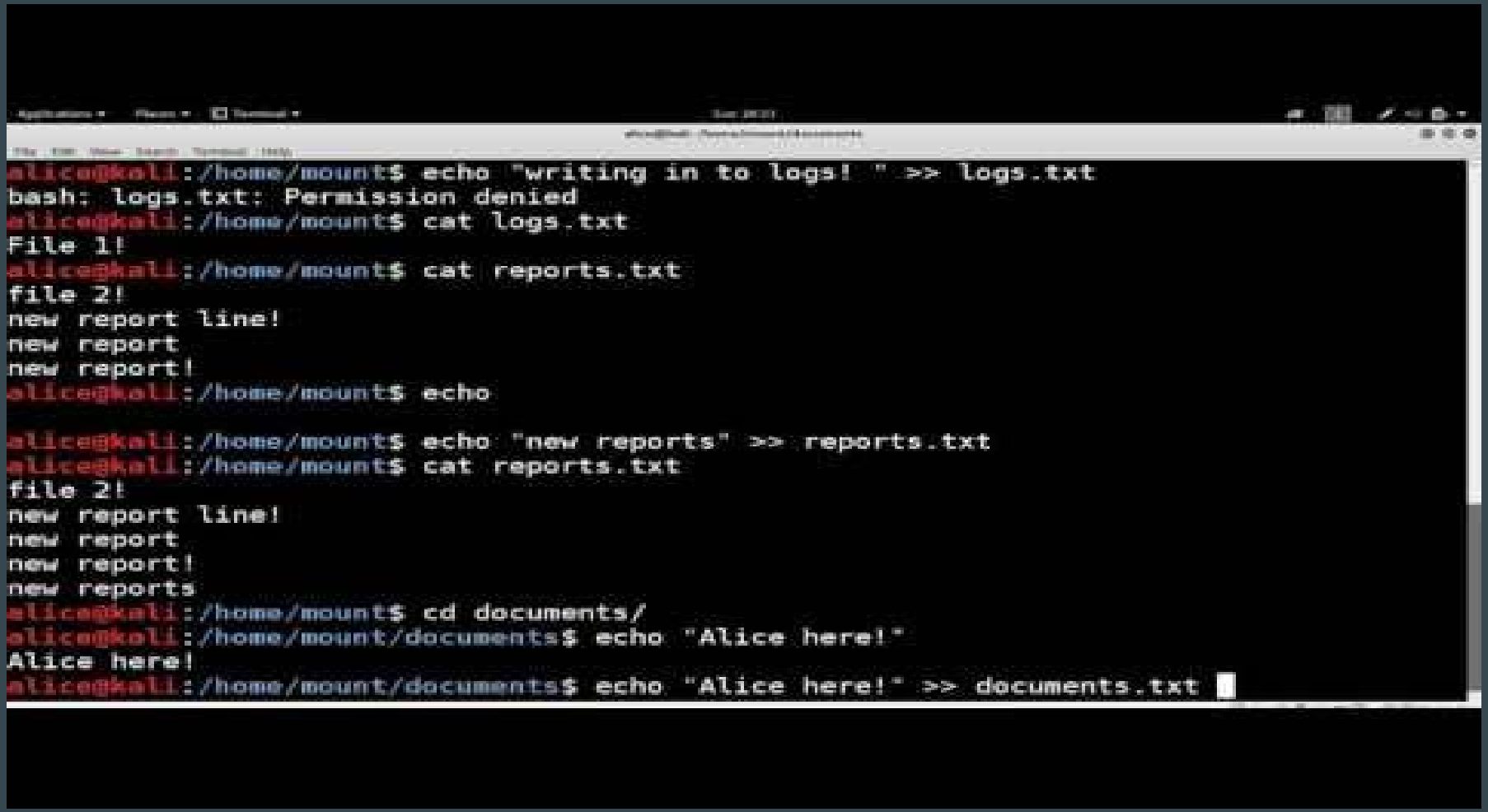
```
$ ls
mount/  rbac.defs  root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
$ ls root/
file1  file2
$ ls mount/
$ rbacfs mount/ root/ rbac.defs
$ ls mount/
file1 file2
$ cat mount/file1
file1
$ cat mount/file2
cat: mount/file2: Permission denied
```

# Implementation - Example

```
$ ls
mount/  rbac.defs  root/
$ cat rbac.defs
user: admin bob
object: admin r /file1
object: admin r /
$ ls root/
file1  file2
$ ls mount/
$ rbacfs mount/ root/ rbac.defs
$ ls mount/
file1 file2
$ cat mount/file1
file1
$ cat mount/file2
cat: mount/file2: Permission denied
```

# Implementation - Complex Example




alice@kali:/home/mount$ echo "writing in to logs! " >> logs.txt
bash: logs.txt: Permission denied
alice@kali:/home/mount$ cat logs.txt
File 1!
alice@kali:/home/mount$ cat reports.txt
file 2!
new report line!
new report
new report!
alice@kali:/home/mount$ echo
alice@kali:/home/mount$ echo "new reports" >> reports.txt
alice@kali:/home/mount$ cat reports.txt
file 2!
new report line!
new report
new report!
new reports
alice@kali:/home/mount$ cd documents/
alice@kali:/home/mount/documents$ echo "Alice here!"
Alice here!
alice@kali:/home/mount/documents$ echo "Alice here!" >> documents.txt

# Maintenance & Security

...

# Maintenance

## Designed with extensibility in mind

- Students can easily make edits
- Not all FUSE functions are implemented

## Unimplemented features include

- Hierarchical RBAC
- Constrained RBAC
- Sessions

# Maintenance - Testing

- Vagrant Virtual Machine
- Unit testing of each module
- Integration testing
  - Scripted
  - For Ubuntu 16.04 and Kali 2016

# Security

- Designed as a proof-of-concept
  - The shadowed root filesystem can still be accessed without mandatory access control
  - Can be made more secure through restricting access to shadowed root
- Sufficient for an academic setting, while easing development and extension

# Summary

Problem
Design Choices
Overall Design
Implementation
Maintenance
Security