When types prevent bugs

Exploring Rust's safety benefits in filesystem development

Florian Meißner

Outline

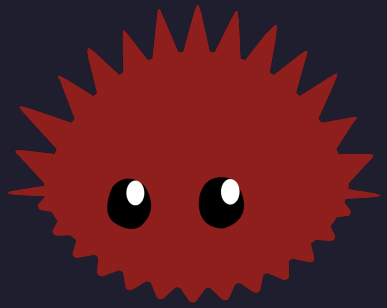
- Basics
- Motivation
- Solution
- Example: hello/hello2
- Comparison
- Outlook
- Sources
- Questions

Basics: Rust

System-level programming language with emphasis on a strong type system.

> Curry-Howard Correspondence implies that, in a sufficiently strict type system, increased expressiveness means increased ability to write provably correct programs.[3]

That means: With a **strong type system**, we can write programs that are (increasingly) **correct** if they **compile**.



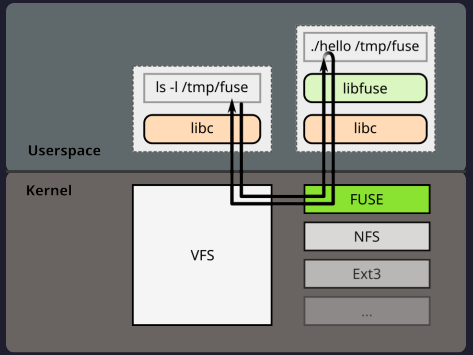
1

Basics: FUSE / libfuse

- **F**ilesystem in **USE**rspace
- => no kernel module needed
- Userspace client implements calls like `open`, `read`, `write`, `readdir` etc.
  - Similar to libc functions

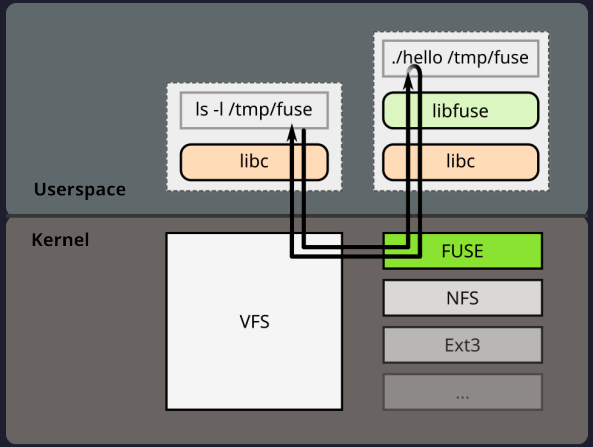
Examples

- Container FS': store folder structure in a single file
- Transparently encrypt files
- Auto-unzip archives
- Restic: mount snapshots



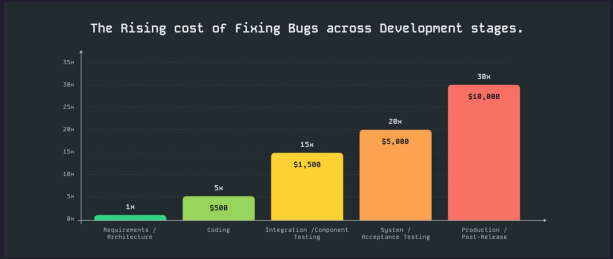
4

Basics: FUSE / libfuse



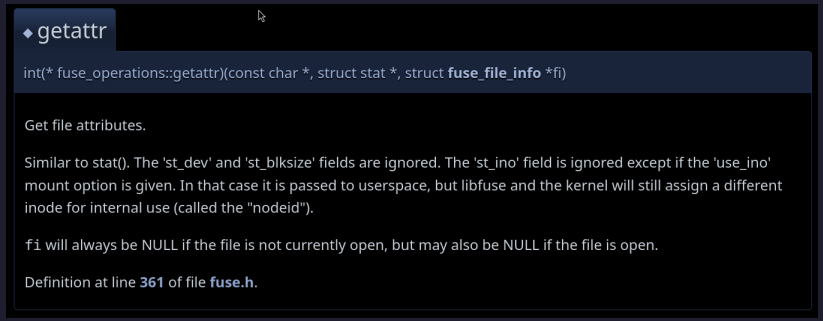
4

Motivation



2

Motivation



◆ getattr

int(* fuse_operations::getattr)(const char *, struct stat *, struct **fuse_file_info** *fi)

Get file attributes.

Similar to stat(). The 'st_dev' and 'st_blksize' fields are ignored. The 'st_ino' field is ignored except if the 'use_ino' mount option is given. In that case it is passed to userspace, but libfuse and the kernel will still assign a different inode for internal use (called the "nodeid").

`fi` will always be NULL if the file is not currently open, but may also be NULL if the file is open.

Definition at line **361** of file **fuse.h**.

Motivation

```
struct stat {
    dev_t      st_dev;      /* ID of device containing file */
    ino_t      st_ino;      /* Inode number */
    mode_t     st_mode;     /* File type and mode */
    nlink_t    st_nlink;    /* Number of hard links */
    uid_t      st_uid;      /* User ID of owner */
    gid_t      st_gid;      /* Group ID of owner */
    dev_t      st_rdev;     /* Device ID (if special file) */
    off_t      st_size;     /* Total size, in bytes */
    blksize_t  st_blksize;  /* Block size for filesystem I/O */
    blkcnt_t   st_blocks;   /* Number of 512 B blocks allocated */

    /* Since POSIX.1-2008, this structure supports nanosecond
       precision for the following timestamp fields.
       For the details before POSIX.1-2008, see VERSIONS. */

    struct timespec  st_atim;  /* Time of last access */
    struct timespec  st_mtim;  /* Time of last modification */
    struct timespec  st_ctim;  /* Time of last status change */
```

Motivation

```c
// `man sys_stat.h.0p`
struct stat {
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* Inode number */
    mode_t    st_mode;    /* File type and mode */
    nlink_t   st_nlink;   /* Number of hard links */
    uid_t     st_uid;     /* User ID of owner */
    gid_t     st_gid;     /* Group ID of owner */
    dev_t     st_rdev;    /* Device ID (if special file) */
    off_t     st_size;    /* Total size, in bytes */
    blksize_t st_blksize; /* Block size for filesystem I/O */
    blkcnt_t  st_blocks;  /* Number of 512 B blocks allocated */
// ...
```

Motivation

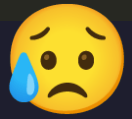
```
// ...
    struct timespec  st_atim;  /* Time of last access */
    struct timespec  st_mtim;  /* Time of last modification */
    struct timespec  st_ctim;  /* Time of last status change */
}
```

What	Constraints
`mode_t st_mode;`	between `0o0000` and `0o7777`; top-most octet has different meaning depending on file type
`dev_t st_dev;`	can be `S_IFREG` (regular file), `S_IFDIR` (directory), `S_IFLNK` (symbolic link), etc…

Not to mention more complex relations, e.g. setting `target` for a file that isn't a symbolic link.

Motivation

```
// […]
stat stat_;
stat_.st_mode = 0o12345;
stat_.st_dev = 34;
```



Motivation

Example: more expressive type system

- `struct inode *iget_locked(struct super_block *sb, unsigned long ino)`
- Callers must check if return value is NULL
- If non-NULL, check if I_NEW is set on i_state field
  - If set, they must initialise the inode
    - Then call unlock_new_inode if init succeeds, inode is refcounted
    - Or call iget_failed if init fails
  - If not set
    - The inode is refcounted and can be used/returned
    - On failure to use, must call iput

Solution

```rust
// rust-bindgen-fuse/src/lib.rs
#[derive(Debug, Clone, Copy, PartialEq,
Eq)]
#[repr(u32)]
pub enum FileType {
    BlockDevice = libfuse::S_IFBLK,
    CharacterDevice = libfuse::S_IFCHR,
    Fifo = libfuse::S_IFIFO,
    RegularFile = libfuse::S_IFREG,
    Directory = libfuse::S_IFDIR,
    SymbolicLink = libfuse::S_IFLNK,
    Socket = libfuse::S_IFSOCK,
}

#[derive(Debug, Clone, Copy, PartialEq,
Eq)]
pub struct FileMode(FileModeRepr);

type FileModeRepr = u32;
```

```rust
#[derive(TypedBuilder)]
#[builder(build_method(into = FileMode))]
pub struct TypedModeBuilder {
    file_type: FileType,

    permissions: FilePermissions,

    #[builder(setter(strip_bool(fallback =
toggle_setuid)))]
    setuid: bool,
    #[builder(setter(strip_bool(fallback =
toggle_setgid)))]
    setgid: bool,
    #[builder(setter(strip_bool(fallback =
toggle_vtx)))]
    vtx_flag: bool,
}
// …
```

Solution

```
let stat = Stat::new_simple(
    TypedModeBuilder::builder()
        .file_type(FileType::RegularFile)
        .permissions(FilePermissions::new(0o444).unwrap())
        .build(),
    1,
    content.len() as i64,
);
```

It is impossible to construct an illegal value of type `FileMode`, `FilePermissions` or `Stat`; and in most cases, during compile-time.

Example

Comparison: fuser crate

- https://docs.rs/fuser
- The most popular, up-to-date, crate for Rust FUSE bindings.
- Is very close to raw C-Bindings.

fuser	This project
Implements the LowLevel FUSE API (inode-based, async)	Implements the "normal" API (path-based, sync)
(Usually) vanilla C datatypes without deeper checks, probably better performance	More complicated mappings between datatypes
Uses references and other safe Rust datatypes, shields against memory corruption and unsafe pitfalls	Additionally, tries to shield against many logic errors

Outlook

- Compile-time bounded integers

```
struct FilePermissions(BoundedU16<0, 0o7777>);

let _ = FilePermissions(33_333); // *compiler error*
```

Strategies:

- atleast two Rust crates expose something like this (**bounded-integer** (https://docs.rs/bounded-integer/latest/bounded_integer/) and **ranged_integers** (https://docs.rs/ranged_integers/latest/ranged_integers/))

Outlook

- Conditional builder

```
TypedModeBuilder::builder()
    .file_type(FileType::SymbolicLink)
    .target(target_path)
    .permissions(FilePermissions::new(0o444).unwrap())
    .build() // OK

TypedModeBuilder::builder()
    .file_type(FileType::RegularFile)
    .target(target_path)
    .permissions(FilePermissions::new(0o444).unwrap())
            .build() // Compile error: no member `target` on this builder
```

Sources

[3] https://www.ps.uni-saarland.de/courses/sem-ws07/notes/0.pdf, p. 53, 21.01.2025

[1] https://rustacean.net/assets/corro.svg

[4] https://en.wikipedia.org/wiki/File:FUSE_structure.svg

[2] https://dev.to/helloquash/did-you-know-the-cost-of-fixing-bugs-increases-exponentially-the-later-the yre-found-in-the-28pl

[5] https://www.youtube.com/watch?v=WiPp9YEBV0Q, "Filesystem in Rust - Kent Overstreet"