Day 22

No More Heap Hoarding

The Story

"Blitzen, do you realize every single `read_to_string()` call triggers a fresh heap allocation? Do you think we've got infinite memory lying around?"

Here, I called it a few times and the process is taking up `1.5GB` of memory, you think this is acceptable?

Blitzen glanced up, nonchalant. "Heap allocations happen, Santa. That's life."

The Story

Santa's eyes narrowed. "Do you want the nightmare of Day 1 all over again? That disaster was just a simple string clone, and it nearly broke us. This? This is worse. Way worse. I don't want to end up manually running `drop()` just to keep things from spiraling into chaos!"

The Story

Blitzen stiffened, finally taking the hint. "Okay, okay. What do you want me to do?"

Prancer cut in cautiously. "Why not cache the file content on every write? And return a reference to the cached content on every read?"

The Story

Blitzen nodded slowly. "Yeah, that'll work. No more redundant allocations, no stale reads, and we don't touch `drop()` at all."

Santa's glare softened—marginally. "Good. Make it lean, make it fast, and make sure it's ready by sunrise. We don't have much time, Christmas is just around the corner."

Your Mission

Santa's files are huge, and they use a lot of memory, To prevent that we need to create a new method named `read_to_str` that will return a string slice instead of an owned value.

Your Mission

Here is what you need to do:

- Create a new field called `content`
- Update the `write` method to update the `content` on every write
- Complete the `read_to_str` method to return a string slice instead of an owned value

Hints

If you're unsure where to start, take a look at these tips:

- Create a new field called `content` to store the file content:

```rust
pub struct TempFile {
    file: File,
    file_path: PathBuf,
    content: Vec<u8>,
}
```

Hints

- Update the field in file writes, use `to_vec()` to turn a `&[u8]` to a `Vec<u8>`:

```
self.content = data.to_vec();
```

- Use the `std::str::from_utf8` function to convert a `&[u8]` to a `&str`:

```
std::str::from_utf8(&self.content).unwrap_or("")
```