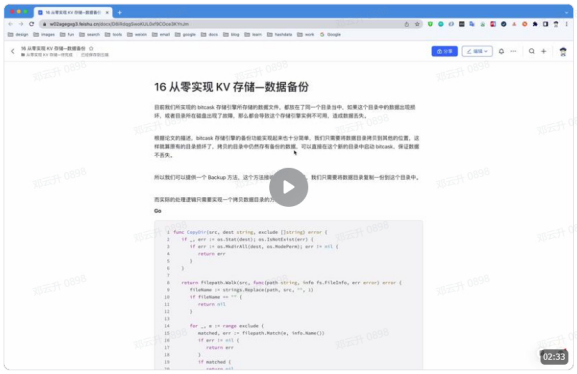
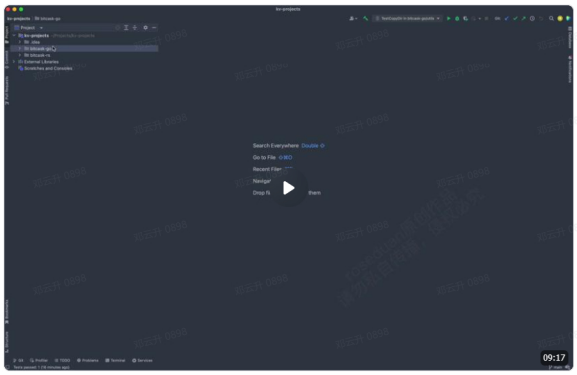
16 从零实现 KV 存储—数据备份

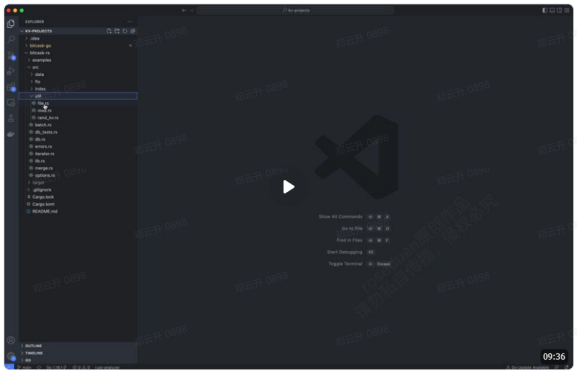
理论讲解



Go 编码



Rust 编码



目前我们所实现的 bitcask 存储引擎所存储的数据文件，都放在了同一个目录当中，如果这个目录中的数据出现损坏，或者目录所在磁盘出现了故障，那么都会导致这个存储引擎实例不可用，造成数据丢失。

根据论文的描述，bitcask 存储引擎的备份功能实现起来也十分简单，我们只需要将数据目录拷贝到其他的位置，这样就算原有的目录损坏了，拷贝的目录中仍然存有备份的数据，可以直接在这个新的目录中启动 bitcask，保证数据不丢失。

所以我们可以提供一个 Backup 方法，这个方法接收一个目标路径，我们只需要将数据目录复制一份到这个目录中。

而实际的处理逻辑只需要实现一个拷贝数据目录的方法。

Go

```go
func CopyDir(src, dest string, exclude []string) error {
    if _, err := os.Stat(dest); os.IsNotExist(err) {
        if err := os.MkdirAll(dest, os.ModePerm); err != nil {
            return err
        }
    }

    return filepath.Walk(src, func(path string, info fs.FileInfo, err error) error {
        fileName := strings.Replace(path, src, "", 1)
        if fileName == "" {
            return nil
        }

        for _, e := range exclude {
            matched, err := filepath.Match(e, info.Name())
            if err != nil {
                return err
            }
            if matched {
                return nil
            }
        }
```

```
        if info.IsDir() {
            return os.Mkdir(filepath.Join(dest, fileName), info.Mode())
        }

        data, err := os.ReadFile(filepath.Join(src, fileName))
        if err != nil {
            return err
        }
        return os.WriteFile(filepath.Join(dest, fileName), data, info.Mode())
    })
}
```

Rust

```rust
// 拷贝数据目录
pub fn copy_dir(src: PathBuf, dest: PathBuf, exclude: &[&str]) -> io::Result<()> {
    if !dest.exists() {
        fs::create_dir_all(&dest)?;
    }

    for dir_entry in fs::read_dir(src)? {
        let entry = dir_entry?;
        let src_path = entry.path();

        if exclude.iter().any(|&x|src_path.ends_with(x)) {
            continue;
        }

        let dest_path = dest.join(entry.file_name());
        if entry.file_type()?.is_dir() {
            copy_dir(src_path, dest_path, exclude)?;
        } else {
            fs::copy(src_path, dest_path)?;
        }
    }
    Ok(())
}
```

需要注意的是，在拷贝的时候，需要将文件锁对应的文件排除掉，在打开 bitcask 存储引擎实例的时候，重新在这个数据目录中申请一个新的文件锁。