BabyCancer: Rust 文件备份工具

黄家俊

December 12, 2025

电子科技大学

目录

Menu

项目介绍

- **项目名称**: BabyCancer
- **编程语言**: Rust
- **项目类型**: 命令行文件备份工具
- **主要功能**:
  - 智能文件备份和过滤
  - 交互式 REPL 界面
  - 多种备份模式（定时、实时）
  - 压缩和归档支持

Menu

文件过滤功能

支持多种文件过滤条件：

- **文件名模式**: 使用正则表达式匹配文件名
- **修改时间**: 根据文件修改日期过滤
- **文件大小**: 按文件大小范围筛选
- **用户权限**: 根据文件所有者过滤
- **文件路径**: 指定特定路径下的文件

示例

```
config --file-name ".*
.txt$" --size 1024 --user admin
```

备份模式

普通备份

- 一次性文件复制
- 保持目录结构
- 支持特殊文件类型

定时备份

- 按指定间隔执行
- 自动循环备份
- 可配置间隔时间

实时备份

- 文件系统监控
- 变更触发备份
- 即时数据保护

归档备份

- TAR 格式打包
- GZIP 压缩支持
- 节省存储空间

Menu

系统架构

REPL 模块	配置模块
命令模块	备份模块

- **REPL 模块**: 交互式命令解析
- **配置模块**: TOML 格式配置管理
- **备份模块**: 核心备份逻辑实现
- **命令模块**: CLI 参数处理

关键技术特性

数据完整性保证

- CRC32 校验和验证
- 文件损坏检测
- 自动清理损坏备份

特殊文件支持

- 符号链接（不跟随）
- FIFO 管道文件
- 字符/块设备文件
- 完整元数据保留

配置持久化

- 进程内配置路径保持
- TOML 格式配置存储
- 灵活的重置机制

Menu

核心代码结构

```rust
// 主要模块
pub mod backup;     // 备份核心逻辑
pub mod command;    // 命令行处理
pub mod config;     // 配置管理
pub mod repl;       // 交互式界面

// 命令定义
#[derive(Subcommand)]
pub enum Commands {
    Backup(backup::BackupArgs),     // 执行备份
    Config(config::ConfigArgs),     // 配置设置
    Reset(config::ResetArgs),       // 重置配置
    Exit,                           // 退出程序
}
```

配置系统实现

```rust
#[derive(Deserialize, Serialize)]
pub struct Config {
    pub path_config: PathConfig,      // 路径配置
    pub file_config: FileConfig,      // 文件过滤配置
    pub output_config: OutputConfig, // 输出配置
}

#[derive(Deserialize, Serialize)]
pub struct OutputConfig {
    pub tar: bool,    // 是否使用TAR打包
    pub gzip: bool,   // 是否启用GZIP压缩
}

// 配置验证和更新机制
trait ValidConfig {
    fn update(&mut self, args: &ConfigArgs) -> Result<(), String>;
    fn reset(&mut self, args: &ResetArgs);
}
```

备份核心逻辑

```rust
fn backup_files(
    source_path: &PathBuf,
    dest_path: &PathBuf,
    file_config: &FileConfig,
    output_config: &OutputConfig,
) -> Result<(), std::io::Error> {
    if output_config.tar {
        let tar_path = if output_config.gzip {
            dest_path.join("backup.tar.gz")
        } else {
            dest_path.join("backup.tar")
        };

        if output_config.gzip {
            let encoder = GzEncoder::new(tar_file, Compression::default());
            let mut tar_builder = tar::Builder::new(encoder);
            tar_builder.follow_symlinks(false);
            tar_builder.append_dir_all(".", source_path)?;
        }
    }
    Ok(())
}
```

Menu

测试覆盖

测试统计: 31 个测试用例，30 个通过，1 个忽略

- **配置功能测试**
  - 配置路径持久化
  - 参数验证和错误处理
  - 重置功能验证
- **备份功能测试**
  - 文件过滤逻辑
  - TAR/GZIP 压缩
  - CRC32 完整性验证
- **REPL 界面测试**
  - 命令解析
  - 交互式工作流程
  - 错误处理机制

测试示例

```
#[test]
fn test_gzip_with_tar() {
    with_clean_config("test_gzip_tar", |config_path| {
        // 配置TAR+GZIP备份
        assert!(repl::execute_line(format!(
            "config -c {} --tar true --gzip true", config_path
        )).is_ok());

        // 执行备份
        assert!(repl::execute_line("backup".to_string()).is_ok());

        // 验证压缩文件存在
        let dest = PathBuf::from("tests/example/dest");
        assert!(dest.join("backup.tar.gz").exists());

        // 验证GZIP格式正确性
        let file = fs::File::open(dest.join("backup.tar.gz")).unwrap();
        let mut gz = flate2::read::GzDecoder::new(file);
        let mut contents = Vec::new();
        gz.read_to_end(&mut contents).unwrap();
        assert!(contents.len() > 0);
    });
}
```

Menu

技术亮点

- **内存安全**: 使用 Rust 语言，零运行时开销
- **并发安全**: 全局状态管理使用 Mutex 保护
- **错误处理**: 完整的 Result 类型错误传播
- **性能优化**:
  - 流式处理大文件
  - CRC32 硬件加速
  - 增量备份支持
- **跨平台**: 支持 Unix 系统特殊文件类型
- **可扩展性**: 模块化设计，易于功能扩展

用户体验

交互式设计

- REPL 模式提供即时反馈
- 配置状态实时显示
- 友好的错误提示信息

灵活配置

- 支持命令行参数和配置文件
- 配置持久化和继承
- 细粒度的重置控制

数据保护

- 备份前数据完整性检查
- 传输过程 CRC32 验证
- 自动清理损坏文件

Menu

项目成果

实现功能

- ✓ 基础备份功能 (40 分)
- ✓ 自定义备份过滤 (+18 分)
- ✓ 定时备份 (+10 分)
- ✓ 实时备份 (+15 分)
- ✓ 文件类型支持 (+10 分)
- ✓ 元数据支持 (+10 分)
- ✓ TAR 打包 (+5 分)
- ✓ GZIP 压缩 (+5 分)

总计: 113 分

额外特性

- CRC32 数据完整性验证
- 全面的测试覆盖 (31 个测试)
- 配置路径持久化机制

技术收获

- **Rust 系统编程**: 掌握了所有权系统和并发编程
- **文件系统操作**: 深入理解 Unix 文件系统特性
- **数据压缩**: 实现了 TAR 归档和 GZIP 压缩集成
- **错误处理**: 构建了健壮的错误处理机制
- **测试驱动开发**: 建立了完整的自动化测试体系
- **配置管理**: 设计了灵活的配置系统架构
- **命令行工具**: 创建了用户友好的 CLI 界面

未来展望

功能增强

- 增量备份算法优化
- 网络远程备份支持
- 加密备份功能
- GUI 图形界面

性能优化

- 并行文件处理
- 内存映射大文件操作
- 压缩算法选择

平台扩展

- Windows 平台支持
- 容器化部署
- WebAssembly 移植

谢谢！

问题与讨论

GitHub: github.com/zhenlu1936/babycancer