加餐｜期中测试：来写一个简单的 grep 命令行

time.geekbang.org/column/article/425013

陈天 · Rust 编程第一课

陈天

Tubi TV 研发副总裁

3669 人已学习

查看详情

课程目录

已更新 40 讲/共 52 讲

开篇词 (1讲)

开篇词｜让Rust成为你的下一门主力语言

前置篇 (3讲)

01｜内存：值放堆上还是放栈上，这是一个问题

02｜串讲：编程开发中，那些你需要掌握的基本概念

加餐｜ Rust真的值得我们花时间学习么？

基础篇 (21讲)

03｜初窥门径：从你的第一个Rust程序开始！

89%

04｜get hands dirty：来写个实用的CLI小工具

05｜get hands dirty：做一个图片服务器有多难？

89%

06｜get hands dirty：SQL查询工具怎么一鱼多吃？

1%

07｜所有权：值的生杀大权到底在谁手上?

08｜所有权：值的借用是如何工作的?

09｜所有权：一个值可以有多个所有者么?

10｜生命周期：你创建的值究竟能活多久?

11｜内存管理：从创建到消亡，值都经历了什么?

加餐｜愚昧之巅：你的Rust学习常见问题汇总

12｜类型系统：Rust的类型系统有什么特点?

13｜类型系统：如何使用trait来定义接口?

14｜类型系统：有哪些必须掌握的Trait?

15｜数据结构：这些浓眉大眼的结构竟然都是智能指针?

16｜数据结构：Vec<T>、&[T]、Box<[T]>，你真的了解集合容器么?

17｜数据结构：软件系统核心部件哈希表，内存如何布局?

18｜错误处理：为什么Rust的错误处理与众不同?

19｜闭包：FnOnce、FnMut 和 Fn，为什么有这么多类型?

20｜4 Steps：如何更好地阅读Rust源码?

21｜阶段实操：构建一个简单的 KV server (1) -基本流程（上）

22｜阶段实操：构建一个简单的 KV server (1) -基本流程（下）

期中周 (2讲)

加餐｜期中测试：来写一个简单的 grep 命令行

加餐｜期中测试：参考实现讲解

进阶篇 (11讲)

23｜类型系统：如何在实战中使用泛型编程?

24｜类型系统：如何在实战中使用 Trait Object?

25｜类型系统：如何围绕 Trait 来设计和架构系统？

17%

加餐｜Rust 2021 版次问世了！

26｜阶段实操：构建一个简单的 KV server (2) - 高级 trait 技巧

11%

27｜生态系统：有哪些常有的 Rust 库可以为我所用？

59%

28｜网络开发：如何使用 Rust 处理网络请求？（上）

2%

29｜网络开发：如何使用 Rust 处理网络请求？（下）

2%

30｜Unsafe Rust：如何用 C++ 的方式打开 Rust？

68%

31｜FFI：Rust 如何和你的语言架起沟通桥梁？

74%

32｜实操项目：使用 PyO3 开发 Python3 模块

并发篇 (2讲)

33｜并发处理：从 atomics 到 Channel，Rust 都提供了什么工具？（上）

34｜并发处理：从 atomics 到 Channel，Rust 都提供了什么工具？（下）

陈天 · Rust 编程第一课

15

15
1.0x

陈天 2021-10-13

00:00

1.0x
讲述：陈天大小：2.18M时长：02:22

你好，我是陈天。

现在 Rust 基础篇已经学完了，相信你已经有足够的信心去应对一些简单的开发任务。今天我们就来个期中测试，实际考察一下你对 Rust 语言的理解以及对所学知识的应用情况。

我们要做的小工具是 rgrep，它是一个类似 grep 的工具。如果你是一个 *nix 用户，那大概率使用过 grep 或者 ag 这样的文本查找工具。

grep 命令用于查找文件里符合条件的字符串。如果发现某个文件的内容符合所指定的字符串，grep 命令会把含有字符串的那一行显示出；若不指定任何文件名称，或是所给予的文件名为 -，grep 命令会从标准输入设备读取数据。

我们的 rgrep 要稍微简单一些，它可以支持以下三种使用场景：

首先是最简单的，给定一个字符串以及一个文件，打印出文件中所有包含该字符串的行：

$ rgrep Hello a.txt

55: Hello world. This is an exmaple text

然后放宽限制，允许用户提供一个正则表达式，来查找文件中所有包含该字符串的行：

$ rgrep Hel[^\\s]+ a.txt

55: Hello world. This is an exmaple text

89: Help me! I need assistant!

如果这个也可以实现，那进一步放宽限制，允许用户提供一个正则表达式，来查找满足文件通配符的所有文件（你可以使用 globset 或者 glob 来处理通配符），比如：

$ rgrep Hel[^\\s]+ a*.txt

a.txt

55:1 Hello world. This is an exmaple text

89:1 Help me! I need assistant!

5:6 Use `Help` to get help.

abc.txt:

100:1 Hello Tyr!

其中，冒号前面的数字是行号，后面的数字是字符在这一行的位置。

给你一点小提示。

对于命令行的部分，你可以使用 clap3 或者 structopt，也可以就用 env.args()。
对于正则表达式的支持，可以使用 regex。
至于文件的读取，可以使用 std::fs 或者 tokio::fs。你可以顺序对所有满足通配符的文件进行处理，也可以用 rayon 或者 tokio 来并行处理。
对于输出的结果，最好能把匹配的文字用不同颜色展示。

```
❯ rgrep "君子" "*.md"
06.md
     5:66  子华使于齐，冉子为其母请粟。子曰：“与之釜。”请益。曰：“与之庾。”冉子与之粟五秉。子曰：“赤之适齐也，乘肥马，衣轻裘。吾闻之也，君子周急不继富。”
    14:10  子谓子夏曰：“女为君子儒，无为小人儒。”
    19:24  子曰：“质胜文则野，文胜质则史。文质彬彬，然后君子。”
    27:38  宰我问曰：“仁者，虽告之曰：‘井有仁焉。’其从之也？”子曰：“何为其然也？君子可逝也，不可陷也；可欺也，不可罔也。”
    28:5   子曰：“君子博学于文，约之以礼，亦可以弗畔矣夫！”
10.md
     7:1   君子不以绀緅饰。红紫不以为亵服。当暑，袗絺綌，必表而出之。缁衣羔裘，素衣麑裘，黄衣狐裘。亵裘长。短右袂。必有寝衣，长一身有半。狐貉之厚以居。去丧，无所不佩。非帷裳，必杀之。羔裘玄冠不以吊。吉月，必朝服而朝。
01.md
     2:37  子曰：“学而时习之，不亦说乎？有朋自远方来，不亦乐乎？人不知而不愠，不亦君子乎？”
     3:38  有子曰：“其为人也孝弟，而好犯上者，鲜矣；不好犯上，而好作乱者，未之有也。君子务本，本立而道生。孝弟也者，其为仁之本与！”
     9:5   子曰：“君子不重则不威，学则不固。主忠信，无友不如己者，过则勿惮改。”
    15:5   子曰：“君子食无求饱，居无求安，敏于事而慎于言，就有道而正焉，可谓好学也已。”
19.md02.md
    13:5   子曰：“君子不器。”
    14:4   子贡问君子。子曰：“先行其言，而后从之。”
    15:5   子曰：“君子周而不比，小人比而不周。”
08.md
     3:33  子曰：“恭而无礼则劳，慎而无礼则葸，勇而无礼则乱，直而无礼则绞。君子笃于亲，则民兴于仁；故旧不遗，则民不偷。”
     5:38  曾子有疾，孟敬子问之。曾子言曰：“鸟之将死，其鸣也哀；人之将死，其言也善。君子所贵乎道者三：动容貌，斯远暴慢矣；正颜色，斯近信矣；出辞气，斯远鄙倍矣。笾豆之事，则有司存。”
     7:31  曾子曰：“可以托六尺之孤，可以寄百里之命，临大节而不可夺也。君子人与？君子人也。”
13.md
     4:59  子路曰：“卫君待子而为政，子将奚先？”子曰：“必也正名乎！”子路曰：“有是哉，子之迂也！奚其正？”子曰：“野哉由也！君子于其所不知，盖阙如也。名不正，则言不顺；言不顺，则事不成；事不成，则礼乐不兴；礼乐不兴，则刑罚不中；刑罚不中，则民无所措手足。故君子名之必可言也，言之必可行也。君子于其言，无所苟而已矣。”
    24:5   子曰：“君子和而不同，小人同而不和。”
    26:5   子曰：“君子易事而难说也：说之不以道，不说也；及其使人也，器之。小人难事而易说也：说之虽不以道，说也；及其使人也，求备焉。”
    27:5   子曰：“君子泰而不骄，小人骄而不泰。”
12.md
     4:57  子夏之门人问交于子张。子张曰：“子夏云何？”对曰：“子夏曰：‘可者与之，其不可者拒之。’”子张曰：“异乎吾所闻：君子尊贤而容众，嘉善而矜不能，我之大贤与，于人何所不容？我之不贤与，人将拒我，如之何其拒人也？”
```

例如这样的输出

如果你有余力，可以看看 grep 的文档，尝试实现更多的功能。

祝你好运！

加油，我们下节课作业讲解见。

给文章提建议

良师益友

Command + Enter 发表

0/2000字

提交留言

精选留言(5)

- 

余泽锋

时间比较紧，先写个初始版本：

```
extern crate clap;

use std::path::Path;
use std::ffi::OsStr;
use std::error::Error;

use clap::{Arg, App};
use regex::Regex;
use tokio::fs::{File, read_dir};
use tokio::io::AsyncReadExt;

#[tokio::main]
async fn main() -> Result<(), Box<dyn Error>> {

    let matches = App::new("rgrep")
                    .version("1.0")
                    .about("Does awesome things")
                    .arg(Arg::with_name("key_word")
                        .index(1))
                    .arg(Arg::with_name("file")
                        .multiple(true)
                        .index(2))
                    .get_matches();
    println!("{:?}", matches);
    let key_word = matches.value_of("key_word").unwrap();
    println!("{}", key_word);
    let file_path = matches.values_of_lossy("file").unwrap();
    println!("{:?}", file_path);

    let re_key_word = format!(r"{}", &key_word);
    println!("re_key_word: {}", &re_key_word);
    let re = Regex::new(&re_key_word).unwrap();

    for file_path in file_path {



        let mut file = File::open(&file_path).await?;
        // let mut contents = vec![];
        let result = tokio::fs::read_to_string(&file_path).await?;

        if let Some(caps) = re.captures(&result) {
            println!("file_path: {:?}", &file_path);
            println!("file: {:?}", &file);
            println!("caps: {:?}", &caps);
```

```
            println!("result: {:?}", &result);
        }
    }

    Ok(())
}
```

作者回复: 嗯，不错。可以进一步优化一下性能以及可测试性。建议看看我的参考代码：https://github.com/tyrchen/geektime-rust/tree/master/mid_term_rgrep

2021-10-17

1

- 

夏洛克Moriaty

磕磕盼盼搞了一天终于实现了这一讲的需求，期中测试算是通过了。自己动手实现的过程中收获了非常多的东西。代码结构前前后后改了许多次，还达不到开发过程中接口不变只是实现变的能力。我把代码仓库链接贴在下面算是献丑了，说实话有点不好意思拿出来哈哈。

https://github.com/LgnMs/rgrep
作者回复: 挺不错的！流程图画的很好啊，可以放到 Readme.md 里

2021-10-14

1

- 

记事本

```
let filename = std::env::args().nth(2).unwrap();
    let query = std::env::args().nth(1).unwrap();
    let case_sensitive = std::env::var("is_sens").is_err();

    let contents = std::fs::read_to_string(filename).unwrap();

    if case_sensitive {
        let mut i = 1;
        for v in contents.lines(){
            if v.contains(&query){
                println!("{}:{}",i,v);
            }
            i+=1;
        }
    }else {
        let c =contents.lines().filter(|item|item.contains(&query)).collect::<Vec<_>>();
        for i in 1..=c.len(){
            println!("{}:{}",i,c[i]);
        }
    }
```

作者回复: 嗯，你可以用 regex 处理，更方便一些。你也可以看看 github 仓库里的代码：https://github.com/tyrchen/geektime-rust/tree/master/mid_term_rgrep

2021-10-13

1

- 

Custer 

1. 最简单的
```rust
use std::error::Error;

use clap::{AppSettings, Clap};
use colored::Colorize;
use tokio::fs;

#[derive(Clap)]
#[clap(version = "1.0", author = "Custer<custer@email.cn>")]
#[clap(setting = AppSettings::ColoredHelp)]
struct Opts {
    find: String,
    path: String,
}

#[tokio::main]
async fn main() -> Result<(), Box<dyn Error>> {
    // 1. 解析参数
    let opts: Opts = Opts::parse();
    let find = opts.find;
    let path = opts.path;
    let length = find.len();

    // 2. 读取文件
    let contents = fs::read_to_string(path).await?;

    // 3. 匹配字符串
    for (row, line) in contents.lines().enumerate() {
        if let Some(col) = line.find(&find) {
            println!(
                "{}:{} {}{}{}",
                row + 1,
                col + 1,
                &line[..col],
                &line[col..col + length].red().bold(),
                &line[col + length..]
            );
        }
    }
    Ok(())
}
```
2. 允许用户提供一个正则表达式，来查找文件中所有包含该字符串的行
```rust
```

```
    // 3. 匹配字符串
    for (row, line) in contents.lines().enumerate() {
        if let Some(re) = Regex::new(find.as_str()).unwrap().find(line) {
            let start = re.start();
            let end = re.end();
            println!(
                "{}:{} {}{}{}",
                row + 1,
                start + 1,
                &line[..start],
                &line[start..end].red().bold(),
                &line[end..]
            );
        }
    }
```

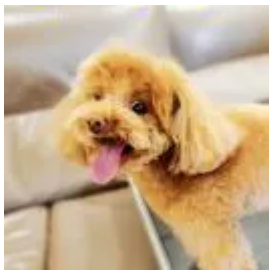
3. 允许用户提供一个正则表达式，来查找满足文件通配符的所有文件(好像并不需要使用globset 或者 glob 就可以处理通配符？）

```rust
...
struct Opts {
    find: String,
    #[clap(multiple_values = true)]
    paths: Vec<String>,
}

#[tokio::main]
async fn main() -> Result<(), Box<dyn Error>> {
    // 1. 解析参数
    let opts: Opts = Opts::parse();
    let find = opts.find.as_str();
    let paths = opts.paths;

    // 2. 循环读取匹配到的文件
    for path in paths {
        println!("{:?}", path);
        let contents = fs::read_to_string(path).await?;

        // 3. 匹配字符串
        ...
    }
    Ok(())
}
```

作者回复: 👍 非常好！

2021-10-15

1

- 

D. D

试着写了一下，实现得比较匆忙。
为了练习之前学过的内容，试了各种写法，应该会有很多不合理的地方。
而且没有做并行化，希望以后有时间可以加上，并把代码重构得更好。
https://github.com/imag1ne/grepr

作者回复: 嗯，写的很不错，尤其是 Display trait 的使用。比我用一个函数处理更好。

impl Display for MatchLine<'_>

2021-10-15

1

收起评论