TinyShell 说明文档

2021 秋 程序设计基础

计11 岳章乔 2021010706

计05 聂子坤 2020010963

计16 张庆林 2021010814

计17 许继茁 2021010

目录

[零、程序的安装与执行 2](#_Toc92638832)

[0.1 Windows 平台 2](#_Toc92638833)

[0.2 Unix/Linux/Mac 平台 2](#_Toc92638834)

[一、基本功能 2](#_Toc92638835)

[1.1. 命令行框架 2](#_Toc92638836)

[1.2. Echo 指令 3](#_Toc92638837)

[1.3 Grep 指令 3](#_Toc92638838)

[1.4. Pwd 指令 4](#_Toc92638839)

[1.5. Cd 指令 4](#_Toc92638840)

[1.6. Tee 指令 5](#_Toc92638841)

[1.7. Cat 指令 5](#_Toc92638842)

[1.8 Cp 指令 6](#_Toc92638843)

[二、扩展功能 6](#_Toc92638844)

[2.1. Grep 指令 6](#_Toc92638845)

[2.2 Echo 指令 7](#_Toc92638846)

[2.3. 解释器框架 7](#_Toc92638847)

[三、异常处理 8](#_Toc92638848)

[四、代码工程结构 8](#_Toc92638849)

[五、功能解说 8](#_Toc92638850)

[六、更新日志 8](#_Toc92638851)

[七、分工 8](#_Toc92638852)

[八、意见反馈 8](#_Toc92638853)

零、程序的安装与执行

在运行TinyShell 前， 在命令行输入 g++ --version，确保 GNU Compiler Collection 已经安装好。

0.1 Windows 平台

1. 打开命令提示符，把工作目录切换到 src 文件夹。

2. 在命令行输入 g++ main.cpp –o main; .\main; 编译cpp文件，然后执行。

0.2 Unix/Linux/Mac 平台

1. 打开 bash，把工作目录切换到 src 文件夹。

2. 在命令行输入 g++ main.cpp; ./a.out; 编译cpp文件，然后执行。

一、基本功能

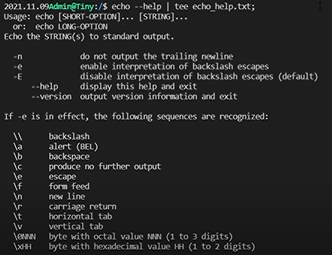
1.1. 命令行框架

命令行首先提示用户输入计算机名，根目录路径和用户名。



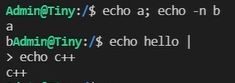
框架实现了命令行参数的标记化（tokenize）,也实现了管道处理——把 “|” 分割符左侧的输出，重定向至分隔符右侧的输入，详见样例。





1.2. Echo 指令

实现了 –n (不在结尾贴上换行符)。



从1.3 开始，测试文件都存在 examples 文件夹里。

1.3 Grep 指令

以下定义匹配行，为行内与模式串有匹配的行。

-c 输出匹配行数

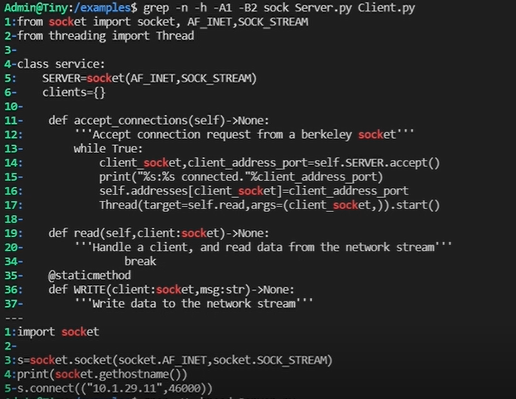


-n 显示行号（行号标为绿色，标点标为蓝色），对匹配行显示 ‘:’，非匹配行的行显示 ‘-’。

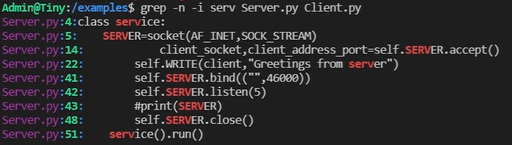
-h 强制不显示文件名（即便是多文件模式）。

-A[i] 显示匹配行后i行。

-B[i] 显示匹配行前 i 行。



-i 忽略模式串与行的大小写差异——如果不考虑大小写差异，那行与模式串匹配，那行就为匹配行。

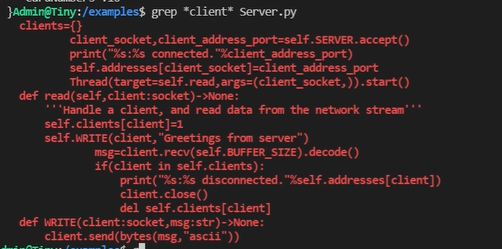


-H 强制显示文件名。



在上面的测试样例，示范了模式串中通配符’.’（一位任意字符）的实现。

以下示范通配符‘\*‘（多位任意字符）的实现。

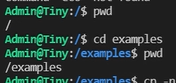


1.4. Pwd 指令

工作正常，样例见 `1.5. Cd 指令’。

1.5. Cd 指令

工作正常。



1.6. Tee 指令

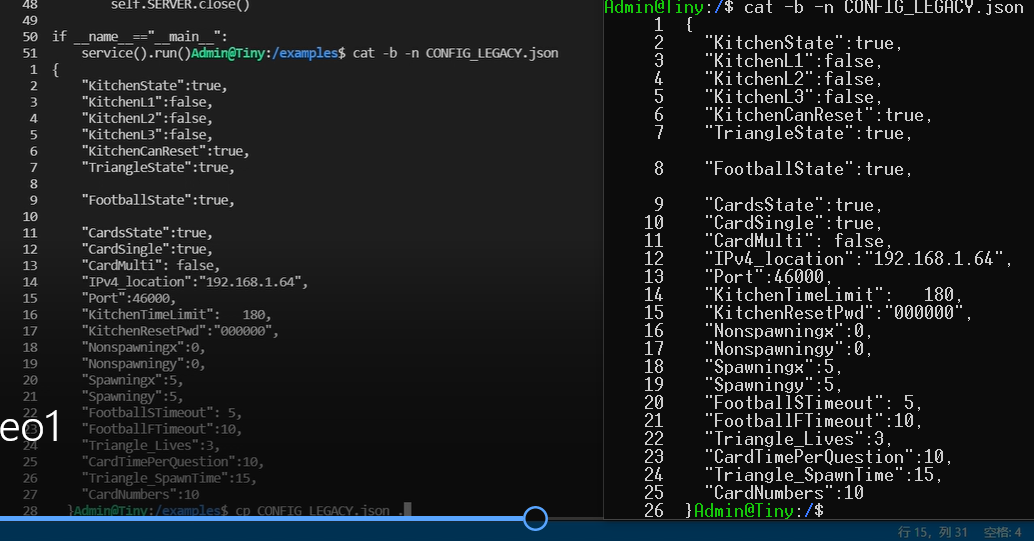
实现情况，见 1.1. 重定向的部分。

-a 如果文件不存在，就新建文件，把标准输入复制到文件里；反之就把标准输入复制在文件末尾。

1.7. Cat 指令

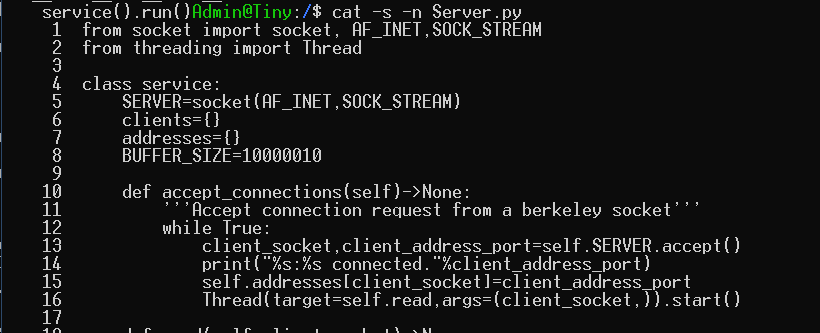
-n 输出行号。

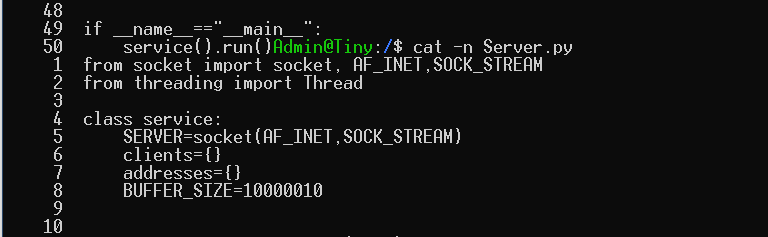
-b 对非空行输出行号，比 –n 的优先级高。



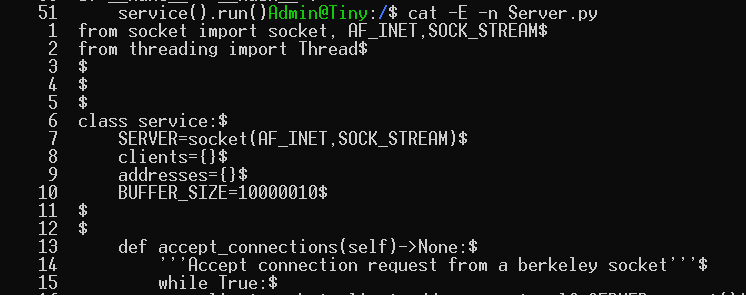
-E 在每行末尾输出‘$’ 号。

-s 输出的时候，合并连续的空行。



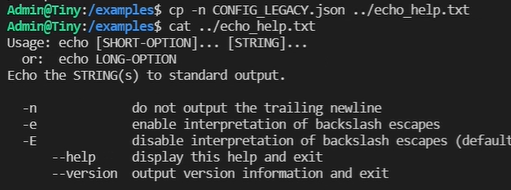






1.8 Cp 指令

-n 如果目的文件存在，就不覆盖；反之就新建文件复制源文件到那个文件。



反之，无论文件是否存在，都把那个文件内容变成是源文件的内容。

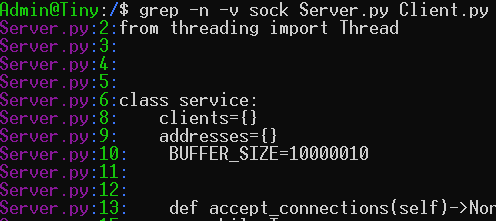
二、扩展功能

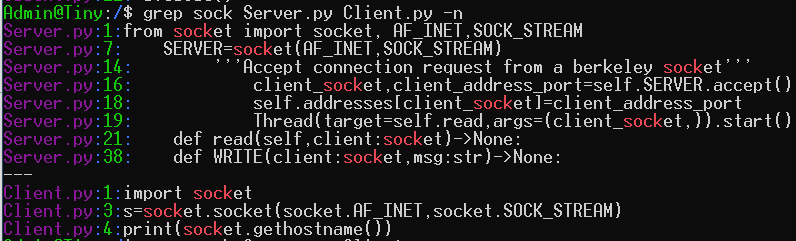
本工程的扩展功能，主要体现在解释器框架、echo指令与grep指令。

2.1. Grep 指令

2.1.1. 实现了‘-v’参数的处理，其功能为反向匹配，即输出不匹配的行。

2.1.2. 即便参数位置不同，也不影响读取。

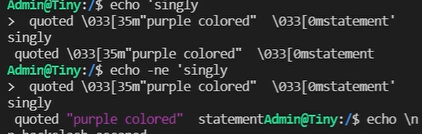




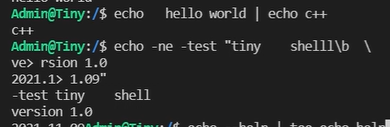
2.2 Echo 指令

-e 把命令行参数里面的特殊字符的表达式，换成相应的特殊字符。

-E -e的逆操作。



实现了把选项连着一起读的功能。



事实上，本工程的echo 与 GNU bash 的 echo 功能一致。

2.3. 解释器框架

实现了对 单引号、双引号、井号、分号的处理。

单引号： 对于引号对以内的所有字符，看作是一个行参数。

双引号： 对于引号内，除了换行符、“`”、“$” 以外，一律看作普通字符处理。

井号： 对于那一行，井号以后的所有字符都不处理，相当于注释。

分号： 表示一条命令的结尾。

另外，对标GNU Bash，如果 “|” 后面是空指令，在下一行提示用户输入指令。

展示略。

三、异常处理

3.1. 解释器框架

对于无效的指令，输出

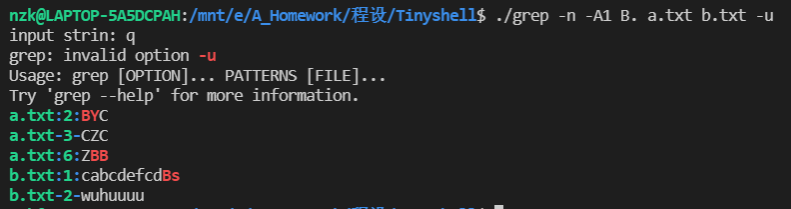
|  |
| --- |
| Command '<错误的指令>' not found |

对于程序运行期间，不涉及内存管理的例外，输出

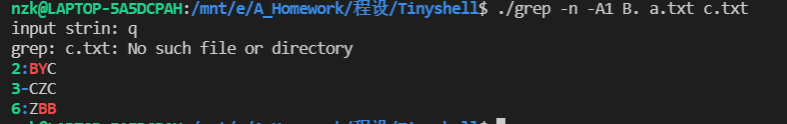
|  |
| --- |
| Error caught while running, see the message below.  <例外的解说>  Please report this bug to yuezq21@mails.tsinghua.edu.cn |

3.2. Grep 指令

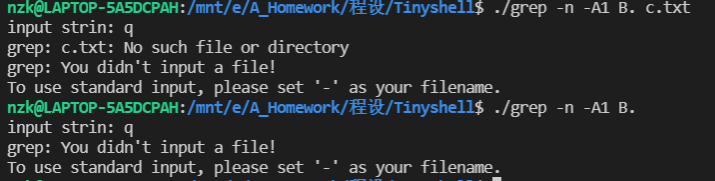
1. 参数错误：提示[invalid option]，但仍在可识别参数的基础上继续匹配。



2. 文件不存在：提示[no such file or directory]，但仍在已有文件的基础上继续匹配。



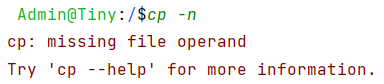
3. 无有效文件：提示使用‘-’作为标准输入。

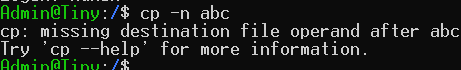


3.3. Cd 指令



3.4. Cp 指令







3.5. 文件处理

在读入文件的时候，如果在读入发现文件不存在，则提示  


四、代码工程结构

五、功能解说

六、更新日志

七、分工

模块一：张庆林

模块二：聂子坤

模块三：许继茁

模块四：岳章乔

八、意见反馈

欢迎对程序运行出现的任何问题，包括但不限于闪退、文件读取异常等，提交反馈到 [nzk20@mails.tsinghua.edu.cn](mailto:nzk20@mails.tsinghua.edu.cn) 或 [yuezq21@mails.tsinghua.edu.cn](mailto:yuezq21@mails.tsinghua.edu.cn) 。