大作业——TinyShell 模拟命令行

〇、声明

本文档旨在说明大作业基本要求。请勿外传。

设置大作业的目的是希望同学们活用课程学到的知识，锻炼解决实际问题的能力，体验团队协作过程，并从过程中学到更多课堂很难涉及的知识。

“程序设计基础”课程教学团队保留对本文档内容的最终解释权。

目录

一、任务概述

命令行（Command Line 或 Command Prompt）是操作系统中通过文本指令与用户交互的一种工作环境。在课程中，运行同学们编写的程序的窗口被称为终端（Terminal），就是一个以图形窗口展现命令行的程序。在常见操作系统中，图形界面只是表象，其基础都是命令行操作。因此，常见操作系统即使在没有图形界面的环境下，也会提供一个命令行，通常被称为控制台（Console）。命令行的核心，是被称为壳（Shell）的命令行解释器。

在不同的操作系统环境下，常用的命令行指令各不相同，但功能有相似性甚至对应关系。比如，在 Windows 环境下，常用的壳程序名为 cmd.exe，其中查看当前目录的指令为 dir；在 Linux 环境下，常用的壳程序名为 bash，其中查看当前目录的指令为 ls。

本次作业希望同学们分模块分组完成一个模拟命令行 TinyShell，可以在其中执行指定的指令。整体任务分为 4 个模块，每位同学负责独立完成其中一个模块。独立部分完成后，再组成四人小组进行合作，将四个模块整合作为最终小组成果。分配方案另行发布。

1.1 diff 指令

diff 指令比较两个文件的内容，详见 2.1 diff 指令说明。注意，整体工程的主函数不在本模块，但自己测试及提交时可以包含适当的主函数。

1.2 grep 指令

grep 指令查找文件内容，详见 2.2 grep 指令说明。注意，整体工程的主函数不在本模块，但自己测试及提交时可以包含适当的主函数。

1.3 tee 指令、cat 指令、cp 指令、cd 指令、pwd 指令

tee 指令写入文件，详见 2.3 tee 指令说明；cat 指令输出文件内容，详见 2.4 cat 指令说明；cp 指令复制文件，详见 2.5 cp 指令说明；cd 指令修改当前目录，详见 2.6 cd 指令说明；pwd 指令显示当前目录，详见 2.7 pwd 指令说明。注意，整体工程的主函数不在本模块，但自己测试及提交时可以包含适当的主函数。

1.4 echo 指令、复合指令、解释器框架

echo 指令输出字符串，详见 2.8 echo 指令说明；复合指令通过“|”将两条指令“串”起来执行，详见 2.9 复合指令说明；解释器框架详见 2.0 解释器框架说明。注意，本模块需要调用 1.1、1.2 及 1.3 的功能来完成整体功能，但自己测试及提交时，需包含相关功能的函数定义（可以为空）才能通过编译。

二、功能任务

2.0 解释器框架

本模块是 TinyShell 的整体框架，包含主函数。

各模块公用的全局变量定义为以下结构和变量：

```
struct Terminal {
    char user[MAXLINE];       // 用户名
    char mach[MAXLINE];       // 计算机名
    char root[MAXLINE];       // 根目录
    char wdir[MAXLINE];       // 工作目录
    char strin[MAXFILE];      // 重定向标准输入
    char strout[MAXFILE];     // 重定向标准输出
};
Terminal gTerm;
```

其中，MAXLINE 和 MAXFILE 可以定义为合适大小的常数。

在程序开始时，提示设置计算机名（其中下划线处为光标位置，下同）：

```
Machine Name: _
```

提示设置根目录：

```
Root Directory: _
```

用户应保证输入的根目录为确实存在的绝对路径，本次作业的所有指令均在此目录下进行。

然后，提示输入用户名（User Name）：

```
Login: _
```

之后，显示命令行提示符，等待用户输入指令。例如：

```
Wang@Tiny:/$ _
```

其中，用户名“Wang”、连接符“@”、计算机名“Tiny”显示为加粗亮绿色，工作目录显示为加粗亮蓝色。如何带颜色输出，可参见2.10 ANSI 颜色输出。初始工作目录为根目录，显示为“/”。

本次作业中规定，一条指令是一行以空格隔开的若干字符串组成。

解释器框架负责获取用户输入的指令，将其拆解为不含有空格的一组字符串（下称为命令行参数），然后以命令行参数作为函数参数调用该指令对应的函数。每个指令函数（复合指令除外）负责解析命令行参数，并执行对应功能。

在本章后续的指令说明中，[xxx]表示名为 xxx 的项目必须出现在指令中；{xxx}表示名为 xxx 的项目为可选项目，且最多只能出现一次；{xxx}*表示名为 xxx 的可选项目，而且可以出现多次；[xxx]*表示名为 xxx 的必选项，而且可以出现多次。指令中出现文件名项目时，“.”表示同级目录，“..”表示上一级目录；文件名可以为绝对路径（相对于 root 目录的路径，如“/Dir1/../File1”等价于真实文件系统中的{root}/File1），也可以是相对路径（相对于 wdir 目录的路径，如“Dir1/../File1”等价于真实文件系统中的{root}/{wdir}/File1）。简单起见，路径和文件名中不允许出现空格，文件夹和文件名只能由大小写字母、数字、_、.、-组成。路径。如无特殊说明，读取文件时，要求文件已经存在，否则提示失败；写入文件时，会创建（如不存在）或覆盖（如已存在）文件。读取文件时，文件名“-”表示从标准输入读取 strin；写入文件时，文件名“-”表示写到标准输出 strout。

2.0.1 正常指令执行样例

例如，如果用户在提示符下输入如下指令：

```
Wang@Tiny:/Dir1$ echo Hello World!_
```

指令将被拆解为字符串“echo”、“Hello”、“World!”，交由 echo 指令函数进一步解析。该指令的功能为依次输出命令行参数，并在其后换行。于是,echo 指令函数执行结束后，重定向标准输出的全局变量 gTerm.strout 中应存储“Hello World!\n”。随后，解释器框架决定将其输出到屏幕上，并重新输出提示符，等待用户进一步输入：

```
Wang@Tiny:/Dir1$ echo Hello World!
Hello World!
Wang@Tiny:/Dir1$ _
```

2.0.2 异常指令执行样例

如果用户输入的指令不合法，将分为两种情况。

1）如果指令不能识别，则由解释器框架提示错误：

```
Wang@Tiny:/Dir1$ eco Hello World!
Command 'eco' not found
Wang@Tiny:/Dir1$ _
```

2）如果指令可识别，但命令行参数不可识别或有误，则由指令函数提示错误：

```
Wang@Tiny:/Dir1$ cat notexistsfilename
cat: notexistsfilename: No such file or directory
Wang@Tiny:/Dir1$ _
```

注意，错误提示并不输出至标准输出，而使用 cerr 直接输出。

试一试，如下代码的执行效果是什么？

```
#include <iostream>
using namespace std;
int main() {
    cerr << "Hello" << endl;
    return 0;
}
```

2.1 diff 指令

diff 指令逐行比较两个文本文件。

2.1.1 指令语法

```
diff --help
diff {-b}{-B}{-I}{-q}{-w}{-I[字符串]}[文件 1][文件 2]
```

diff 指令会逐行比较[文件 1]和[文件 2]的内容，行与行的比较规则参见 2.1.2 选项说明选项说明，输出格式参见 2.1.4 样例样例说明。

2.1.2 选项说明

--help：显示帮助。

-b：不检查空格字符的不同。

-B：不检查空白行。

-i：不检查大小写的不同。

-q：仅显示有无差异，不显示详细的信息。无差异时不显示，有差异时显示“File [文件 1] and [文件 2] differ”。

-w：忽略全部的空格字符。

-I[字符串]：若两个文件在某几行有所不同，但这几行同时都包含了选项中指定字符串，则不显示这两个文件的差异。

更多功能参考：https://www.man7.org/linux/man-pages/man1/diff.1.html

2.1.3 指令函数声明

```
void doDiff(int argc, char * argv[]);
```

其中，argc 表示命令行参数个数，argv 为命令行参数数组。

例如如果用户输入“diff a.txt b.txt”指令，解释器框架应产生 argc=3、argv={"diff", "a.txt", "b.txt"}的参数进行调用，解释器框架负责申请和释放命令行参数所需内存。

如无特殊说明，以下指令函数声明与此类似。

2.1.4 样例

例如，文件 a.txt 和 b.txt 的内容分别如下：

a.txt	b.txt
AXC BYC CZC DRL EEF ZBB	GHH AXC BTT BZZ CZC DRL

输入指令“diff a.txt b.txt”会产生以下输出：

输出	解释
0a1 >GHH 2c3 <BYC --- >BTT >BZZ 5d6 <EEF <ZBB	在 a.txt 的第 0 行和 b.txt 的第 1 行出现不同 添加内容为 GHH 的一行 在 a.txt 的第 2 行和 b.txt 的第 3 行出现不同 将 BYC 的一行修改为 BTT/BZZ 的两行 在 a.txt 的 5 行和 b.txt 的第 6 行出现不同 删除 EEF/ZBB 的两行

其中，a 表示添加 add，c 表示修改 change，d 表示删除 delete；前后两个数字分别表示在两个文件中的行号；后面紧跟具体修改行的

内容。要求输出的行数尽量少。

2.2 grep 指令

grep 指令用于查找文件里符合条件的字符串，并把含有该字符串的那一行显示出来，符合条件的字符串以加粗亮红色显示。

2.2.1 指令语法

```
grep --help
grep {-h}{-H}{-I}{-n}{-c}{-A[行数]}{-B[行数]}[模式串][文件]*
```

在每个[文件]中逐行寻找包含符合[模式串]的行。

[模式串]可以是普通字符串，也可以包含通配符（.或*）。“.”可匹配一个任意字符；“*”可匹配任意数量（包括 0）任意字符。匹配时，应使得匹配到的字符串尽量长；当一行内有多处匹配时，仅匹配第一项即可。

例如：

ab.c 与 abdc 匹配；

ab*c 与 abc、abdc、abedc 均匹配；

在尝试将 ab*c 与 cabcdefcd 匹配时，将匹配其中的 abcdefc。

2.2.2 选项说明

--help：显示帮助

-c：计算符合样式的行数。

-h：在显示符合样式的那一行之前，不标示该行所属的文件名称。

-H：在显示符合样式的那一行之前，标示该行所属的文件名称。

-i：忽略字符大小写的差别。

-n：在显示符合样式的那一行之前，标示出该行的编号。

-A[行数]：除了显示符合范本样式的那一列之外，同时显示该行之后的[行数]行内容。多行连续匹配需要合并输出，不要冗余输出。

-B[行数]：除了显示符合样式的那一行之外，同时显示该行之前的[行数]内容。多行连续匹配需要合并输出，不要冗余输出。

更多功能参考：https://www.man7.org/linux/man-pages/man1/grep.1.html

2.2.3 指令函数声明

```
void doGrep(int argc, char * argv[]);
```

2.2.4 样例

例如，文件 a.txt 如下：

a.txt
AXC BYC CZC DRL EEF ZBB

输入指令“grep *B a.txt”会产生以下输出：

输出	解释
BYC ZBB	*B = {}B *B = {ZB}B

输入指令“grep -n -A1 -B1 B a.txt”会产生以下输出：

输出	解释
1-AXC 2:BYC 3-CZC 5-EEF 6:ZBB	-表示该行无匹配，左边为行号 :表示该行有匹配，左边为行号 行号为加粗亮绿色，-和:为加粗亮蓝色，匹配项为加粗亮红色。

2.3 tee 指令

tee 指令会从标准输入读取数据，将其内容输出到标准输出，同时可以保存至多个文件。

2.3.1 指令语法

```
tee --help
tee {-a}{文件}*
```

将标准输入中的内容输出到标准输出，同时写入{文件}中。

2.3.2 选项说明

--help：显示帮助。

-a：附加到既有文件的后面，而非覆盖它。

更多功能参考：https://www.man7.org/linux/man-pages/man1/tee.1.html

2.3.3 指令函数声明

```
void doTee(int argc, char * argv[]);
```

2.3.4 特殊说明

简化起见，tee 单独使用时，不由键盘输入写入文件的内容，而由标准输入 strin 读取，即仅创建空文件。

tee 与参与复合指令时，参见2.9.2 样例。

2.4 cat 指令

cat 指令用于连接文件并输出到标准输出上。

2.4.1 指令语法

```
cat --help
cat {-b}{-E}{-n}{-s}[文件]*
```

依次输出[文件]中的内容。

2.4.2 选项说明

--help：显示帮助。

-n：由 1 开始对所有输出的行数编号。

-b：和-n 相似，只不过对于空白行不编号，但需要输出空白行。

-s：当遇到有连续两行以上的空白行，就代换为一行的空白行。

-E：在每行结束处显示$。

更多功能参考：https://www.man7.org/linux/man-pages/man1/cat.1.html

2.4.3 指令函数声明

```
void doCat(int argc, char * argv[]);
```

2.4.4 样例

例如，文件 a.txt 和 b.txt 内容如下：

a.txt	b.txt
AXC BYC CZC DRL EEF	GHH AXC BTT BZZ CZC

输入指令“cat -n a.txt b.txt”会产生以下输出：

输出	解释
1 AXC 2 BYC 3 CZC 4 DRL 5 EEF 6 GHH 7 AXC 8 BTT 9 BZZ 10 CZC	行号向第 6 列对齐，其后空两个格 a.txt 内容到此结束。 b.txt 内容从此开始，行号继续。

2.5 cp 指令

cp 指令用于复制文件。

2.5.1 指令语法

```
cp --help
cp {-n}[文件 1][文件 2]
```

读取[文件 1]，将全部内容写入[文件 2]。

2.5.2 选项说明

--help：显示帮助。

-n：不覆盖已存在的文件。

更多功能参考：https://www.man7.org/linux/man-pages/man1/cp.1.html

2.5.3 指令函数声明

```
void doCp(int argc, char * argv[]);
```

2.6 cd 指令

cd 指令用于切换工作目录。

2.6.1 指令语法

```
cd --help
cd [路径]
```

其中，[路径]如以“/”开头为绝对路径，否则为相对路径。另外，“.”表示同一层目录，“..”表示上一层目录。注意不能切换到根目录的上层目录。另外，由于不使用系统相关库的条件下无法检查文件夹是否存在，要求用户自行保证指令输入的目录一定存在。

2.6.2 选项说明

--help：显示帮助。

更多功能参考：https://www.man7.org/linux/man-pages/man1/cd.1p.html

2.6.3 指令函数声明

```
void doCd(int argc, char * argv[]);
```

2.6.4 样例

参见 2.7.4 样例。

2.7 pwd 指令

pwd 指令用于显示工作目录。

2.7.1 指令语法

```
pwd --help
pwd
```

2.7.2 选项说明

--help：显示帮助。

更多功能参考：https://www.man7.org/linux/man-pages/man1/pwd.1.html

2.7.3 指令函数声明

```
void doPwd(int argc, char * argv[]);
```

2.7.4 样例

```
Wang@Tiny:/Dir1$ pwd
/Dir1
Wang@Tiny:/Dir1$ cd ..
Wang@Tiny:/$ pwd
/
Wang@Tiny:/$ cd Dir1/Dir2
Wang@Tiny:/Dir1/Dir2$ pwd
/Dir1/Dir2
Wang@Tiny:/Dir1/Dir2$ cd /Dir1/Dir2/..
Wang@Tiny:/Dir1$ _
```

2.8 echo 指令

echo 指令用于字符串的输出。

2.8.1 指令语法

```
echo --help
echo {-n}{字符串}*
```

依次输出{字符串}到标准输出。简单起见，不识别{字符串}中的转义字符。

2.8.2 选项说明

--help：显示帮助。

-n：表示输出之后不换行。

更多功能参考：https://www.man7.org/linux/man-pages/man1/echo.1.html

2.8.3 指令函数声明

```
void doEcho(int argc, char * argv[]);
```

2.8.4 样例

```
Wang@Tiny:/Dir1$ echo Hello World!
Hello World!
Wang@Tiny:/Dir1$ echo -n Hello World!
Hello World!Wang@Tiny:/Dir1$ _
```

2.9 复合指令

复合指令使用“|”运算符连接两个指令。可嵌套。多指令的衔接可以形成非常灵活的效果。

2.9.1 指令语法

```
[指令 1] | [指令 2]
```

先执行指令 1，再执行指令 2。执行指令 2 时，将指令 1 产生的标准输出作为指令 2 的标准输入。

2.9.2 样例

样例 1：

```
Wang@Tiny:/Dir1$ echo Hello World! | tee a.txt
Hello World!
Wang@Tiny:/Dir1$ _
```

解释：

指令 1“echo Hello World!”在标准输出中产生了“Hello World!\n”的字符串。该字符串并不直接输出，而是交由指令 2“tee a.txt”继续处理。指令 2 将标准输入中的内容输出至标准输出的同时，写入 a.txt。于是产生了上述结果。同时，/Dir1 文件夹下将产生一个 a.txt 文件，内容为“Hello World!\n”；如果/Dir1/a.txt 原本就存在，则内容被覆盖。

样例 2：

```
Wang@Tiny:/Dir1$ echo Hello World! | echo Hello C++!
Hello C++!
Wang@Tiny:/Dir1$ _
```

解释：

指令 1 在标准输出中产生了字符串，但指令 2 并未读取。于是产生了上述结果。

样例 3：

```
Wang@Tiny:/Dir1$ echo Hello World! | pwd | grep ir -
/Dir1
Wang@Tiny:/Dir1$ _
```

解释：

指令 1 在标准输出中产生了字符串，但指令 2 并未读取。指令 2 在标准输出中产生的字符串被指令 3 使用。于是产生了上述结果。

2.10 ANSI 颜色输出

以 ESC（ASCII 码 27，转义字符“\e”）开始的字符串在 ANSI 标准中表达特殊输出格式。本作业中用到的特殊输出格式包括：

转移字符串	含义
\e[0m	恢复默认
\e[91;1m	加粗亮红色
\e[92;1m	加粗亮绿色
\e[94;1m	加粗亮蓝色

例如，如下代码：

```
#include <iostream>
using namespace std;
int main() {
    cout << "\e[91;1mT\e[92;1mi\e[94;1mn\e[0my" << endl;
    return 0;
}
```

其执行效果应该为：

```
Tiny
```

如果在 Windows 环境下看不到上述效果，可尝试在终端中输入如下指令修改注册表，打开 ANSI 颜色输出的支持。

```
reg add HKEY_CURRENT_USER\Console /v VirtualTerminalLevel /t REG_DWORD /d 0x00000001 /f
```

更多信息可参考：https://en.wikipedia.org/wiki/ANSI_escape_code

三、提交要求及计分说明

3.1 组队要求

每位同学独立完成自己所应负责的模块部分，之后同组员合作进行整合。

要求 4 人一组，如出现人数不足的情况可由同学自愿报名参与两组的大作业组队完成部分，并酌情考虑予以加分。分组方案确定后，每组应选出一名队长。

3.2 提交要求

3.2.1 单人模块

每位同学应在网络学堂提交一个.zip 压缩包，压缩包内容应包括但不限于：

1）程序源文件。放置在 src 文件夹下，可以有多级目录，应包括模块全部编译所需文件及测试文件。

2）说明文档。一个 Word 或 Pdf 文档，放置在压缩包根目录下，内容应包括但不限于：学生信息（姓名、学号、班级）、基本功能完成情况、扩展功能说明（可选）。

3）演示视频。放置在压缩包根目录下，内容应包括但不限于：模块基本功能演示、扩展功能演示（可选）。建议长度不超过 3 分钟，大小不超过 100MB。如果上述内容过大无法上传至网络学堂，可分别用一个内含有效链接的.txt 文件代替。

3.2.2 小组提交

每组由队长在网络学堂提交一个.zip 压缩包。组员可以（但不必须）在网络学堂提交作业时说明队长姓名学号，但不要提交附件。发现组员在网络学堂提交附件的，将单独扣分。压缩包内应包括但不限于：

1）程序源文件。放置在 src 文件夹下，可以有多级目录，应包括全部工程编译所需文件。

2）说明文档。一个 Word 或 Pdf 文档，放置在压缩包根目录下，内容应包括但不限于：小组人员（姓名、学号、班级）、基本功能完成情况、扩展功能说明（可选）、分工情况。

3）演示视频。放置在压缩包根目录下，内容应包括但不限于：小组人员展示、基本功能演示、扩展功能演示（可选）。建议长度不超过 3 分钟，大小不超过 100MB。如果上述内容过大无法上传至网络学堂，可分别用一个内含有效链接的.txt 文件代替。

3.3、计分说明

1、要求主体使用 C/C++实现。满分 100 分，个人模块与小组成果各占 50%。完整实现第二章规定的全部任务就可以获得 100%的大作业分值。同小组内的同学获得的小组成果评分相同，但个人模块评分可能不同。

2、允许调用 C/C++语言提供的库以外的其他库，允许调用其他语言编写的函数。但如这样做，应在说明文档中给出说明，并扣除由此

节省的工作量所对等的分值。特别的，如使用操作系统相关的库和头文件（如 system.h、windows.h、unistd.h 等）会进行扣分处理。C++标准库包含的功能可参考：https://en.cppreference.com/w/cpp/header。

3、实现扩展功能或扩展任务（即，除第二章中所提及任务以外的其他有意义的扩展功能）有额外加分，分值视重要程度而定。需要说明的是，第二章中的任务对真实的指令功能进行了简化，推荐扩展功能参考真实的 bash 指令。如果为实现某些扩展功能不得不使用操作系统相关库，推荐使用 Linux 系统配套库，非常不推荐使用 Windows 系统配套的库。

4、“抑制内卷条款”：实现扩展功能或扩展任务并不能获得超过100%的大作业分值，但可以用来补足由于第 2 条规定导致的扣分。请各位同学根据兴趣、能力和时间安排，选择是否实现扩展功能或扩展任务。