操作系统,作业 0

该作业包括一些编程问题,并有机会阅读一些 POSIX 系统调用的在线文档。评分时,我们将在 Gradescope 上编译和测试您的程序。通常,我们将使用如下选项进行编译。在某些情况下,我们需要为特定的编程任务添加一些额外的选项。如果编程问题需要不同的编译器选项,我们会在问题描述中告知您。

gcc -Wall -g -std=c99 -o 程序 program.c

在上面的命令中，-g 选项告诉编译器在可执行文件中包含符号信息,以帮助调试器完成他们的工作。 -Wall 选项会打开很多编译器警告。这可以帮助您查找代码中的错误,即使这些错误不足以阻止编译。 -std=c99 选项启用您可能已经习惯的一些 C99 语言功能。

如果您在不同的系统上开发程序,您需要确保它在 Gradescope 上正确编译和运行 - 您可以在截止日期前无限次提交代码以查看自动评分器的反馈,以及在截止日期,我们只会对您最后一次提交的内容进行评分。此外,请记住,您的程序需要注释并始终如一地缩进,并且您必须记录您的来源并从头开始自己编写至少一半的代码。

1. (16 分)登录到一台 EOS Linux 机器（EB 3 中的一台 Linux 桌面机器,在图书馆,在 111 Lampe（以前称为 Daniels Hall）,或通过 remote.eos.ncsu.edu)和使用联机手册页回答以下每个问题。要阅读特定命令、函数或系统调用的联机手册页,您通常只需键入 man call,其中 call 是您想要阅读的调用。如果两个不同的手册页具有相同的名称,您可以指定手册的部分以确保获得所需的页面。手册的第 1 节是关于 shell 命令的,第 2 节是关于系统调用的。因此,man 1 stat 将告诉您有关 stat shell 命令的信息,而 man 2 stat 将告诉您有关同名系统调用的信息。

   将您的答案写入文本文件,将其转换为名为 hw0 p1.pdf 的 pdf 文件,然后将其提交至 Gradescope 上的 HW0 Q1。

   (a) open() 系统调用可以使用两个参数或三个参数。什么决定操作系统需要两个参数还是三个参数？ (b) 当你调用 read() 系统调用时,它可能会遇到错误。作为一名程序员,您如何区分读取到达文件末尾和遇到某种类型错误的时间？

   (c) fork() 系统调用可能会失败,因为已经有太多的进程。如果发生这种情况,程序员如何知道 fork() 因这个特定原因而失败。 (d) 假设您使用 fork() 创建了几个子进程,然后调用 wait()。对 wait() 的调用将在任何一个子进程终止后返回。在 wait() 返回后,你怎么知道哪个子进程终止了？

2. (10 分) 简要回答下列问题。将您的答案写入文本文件,将其转换为名为 hw0 p2.pdf 的 pdf 文件,然后将其提交至 Gradescope 上的 HW0 Q2。

   (a) (3 pts) 时间局部性和空间局部性有什么区别？ (b)（3 分)以下代码片段是否表现出时间和/或空间局部性,为什么？

   对于 (int i = 0; i < 10; i++) { printf(some_array[i]);

   }

(c) （4 分)为什么中断和陷阱对于现代操作系统设计至关重要?哪些方面
的系统功能依赖于它们?

3. (40 pts) 对于这个问题,你将使用 Unix 系统调用接口实现一个名为 exclude.c 的程序。我们将在没有 C 标准库中的任何函数的情况下执行此操作;这会让我们稍微了解一下 C 库是如何在 Unix 系统上实现的。在内部,这个库必须只使用 C 代码和对操作系统的调用来完成它所做的一切。

一旦您的排除程序开始工作,您应该能够像下面的示例一样运行它：

$ ./exclude input-1.txt output.txt 3

第一个参数给出输入文件的名称,第二个参数给出程序应该写入的输出文件的名称。最后一个参数应该是正整数,给出输入文件中的行数。该程序的工作是将输入文件中的所有内容复制到输出文件中,不包括命令行上数字指示的行。行从 1（输入文件的第一行）开始编号。

在上面的示例中,我们要求程序将除第 3 行之外的所有行从 input-1.txt 复制到 output.txt。作业中提供了一个示例输入文件 input-1.txt。下图显示了它包含的内容以及在运行如上所示的排除程序后输出文件将包含的内容。输出文件看起来就像输入文件，只是缺少第 3 行。

input-1.txt	output.txt
This is a sample input file for the exclude program. You can give a command-line argument to tell it what line to omit when copying to the output.	This is a sample input file for the You can give a command-line argument to tell it what line to omit when copying to the output.

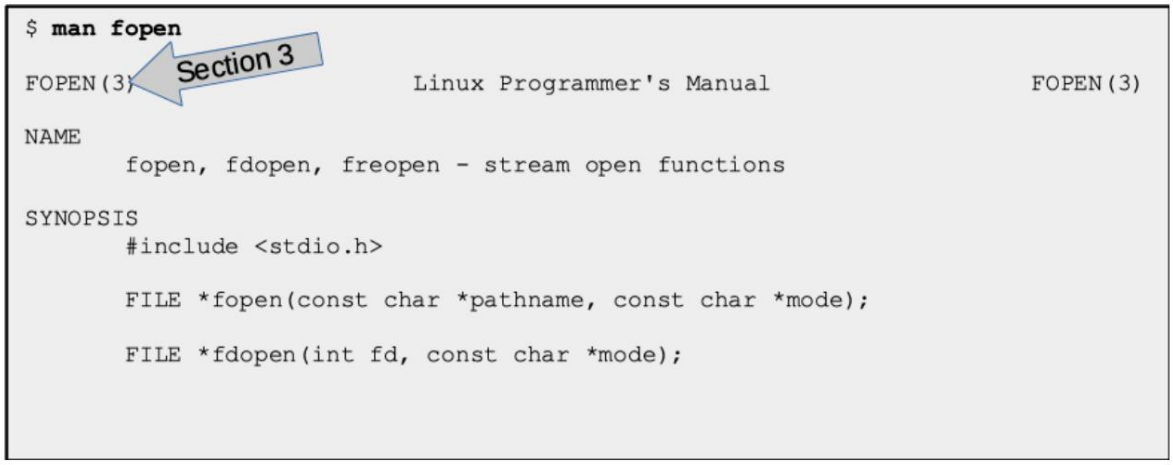
执行

您将仅使用 Unix 系统调用来解决这个问题。您不能使用 C 标准库中的任何函数。这将使您有机会了解系统调用接口是什么样的，以及 C 标准库如何添加功能以使其更便携、更易于使用。

仅使用系统调用接口意味着您不能使用 fopen()、fscanf()、fprintf() 和 fclose() 等函数。您需要改用 open()、read()、write() 和 close() 等系统调用。同样,您不能使用像 atoi() 或 strlen() 这样的函数。如果您需要将字符串转换为整数或测量字符串的长度,则需要编写自己的函数或代码块来执行此操作。这不是太难。

为避免意外使用 C 库中的某些内容,请确保不包含其任何标头。这样,如果您尝试使用标准库函数,编译器就会报错。此处列出了标准库标头： https ://en.cppreference.com/w/c/header如果您不确定某个特定函数,可以使用 man 命令查看其在线文档。喜欢它

如下所示,如果一个函数是 C 标准库的一部分,man 命令会告诉你它在手册的第 3 节中。这应该在靠近顶部的括号中。

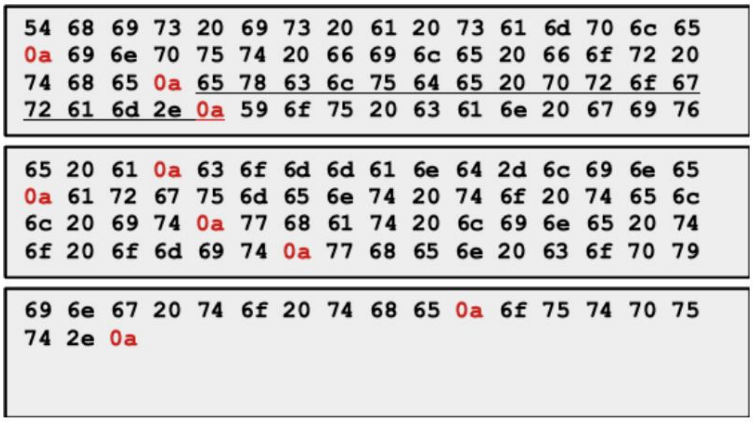


如果没有标准库,您将需要编写自己的代码来将最后一个命令行参数从字符串转换为整数。为此,您可以一次通过该参数一个字符,并将每个字符从 ASCII 字符代码转换为十进制值。因为它是一个十进制数,所以它的最后一位(最右边的)数字是一个的位置。左边的下一个是十位,所以它值十倍。下一个是百位,依此类推。您应该能够编写一个小循环来检查最后一个命令行参数的有效性并将其从 ASCII 字符序列转换为整数值。如果行号左边有一些额外的零就可以了;您不必将此标记为错误。因此,例如,017 将是提供行号 17 的合法方式,但 5x2 和 -904 将不是合法的行号。

您需要使用 read() 和 write() 系统调用来读取和写入文件。这些以字节块的形式读取和写入数据。您将以最多 64 字节的块为单位读取文件。最后一次读取可能不会填满 64 字节缓冲区,因为文件大小可能不是 64 字节的倍数。

read() 和 write() 系统调用不使用以空字符结尾的字符串,它们不关心换行符在文件中的位置。文件中的行只是带有一个换行符标记结尾的字节序列。要排除具有给定数字的行,您需要代码在阅读每个块后查看它并注意行结束的位置。每次传递换行符时,您都可以将其计为一行。为了写入输出文件,您可以复制除最后一个参数指定的一行中的字节之外的所有内容。

下图显示了程序将在上面显示的示例中读取的内容。 input-1.txt 文件的每个字节都显示为两个十六进制数字。红色的是换行符,标记行的结束。带下划线的字节是第三行。对于上面的示例,这些是不应复制到输出文件的字节。



无效参数

给定无效的命令行参数,程序应将以下用法消息打印到标准错误并以退出状态 1 终止。

用法:排除 <输入文件> <输出文件> <行号>

如果用户没有提供正确数量的参数、最后一个参数不是正整数或者程序无法打开给定文件之一,命令行参数将无效。如果给定的数字大于文件中的行数也没关系。这不被认为是错误;它只会产生一个与输入文件相同的输出文件。

如果没有标准库,您将无法使用 fprintf() 或 stderr 来打印此消息,您将无法调用 exit() 来终止程序。要打印使用消息,您可以使用 write() 系统调用将消息的内容发送到错误输出流的文件描述符。幸运的是,unistd.h 头文件定义了一个常量 STDERR FILENO,它给出了标准错误流的文件描述符(它是 2)。

要终止程序,您可以调用 exit() 系统调用。它就像标准库中的 exit() 一样工作,但它是一个 Unix 系统调用而不是标准库函数。

提交

完成后,在 Gradescope 上名为 HW0 Q3 的作业下提交源文件 exclude.c。您不需要为此问题提交任何其他文件,也不要将您的源文件放在存档或子目录中。只需将文件 exclude.c 直接提交到 HW0 Q3 提交柜即可。

4. (40 pts) 你自己的shell程序:stash.c。

shell就是我们所说的Unix系统上的命令行解释器。当您登录到终端窗口并键入“ls”或“cd”等命令时,shell 会读取您的命令,解析它并决定要做什么。 Shell 程序的名称通常以“sh”结尾,例如 sh(bourne shell)、bash(bourne again shell)、csh(C shell)、ksh(korn shell)或 zsh(Z shell)。

对于这个作业,我们将编写我们自己的 shell 程序,名为 stash。我们会说这代表“简单的玩具任务外壳”。它仅具有典型 shell 程序的一些功能,但这足以说明程序如何使用系统调用来启动新程序并等待它们完成。

对于这个问题,您不仅限于使用系统调用。您可以根据需要组合使用 POSIX 系统调用和 C 标准库功能。这通常是您编写程序的方式。您将在需要的地方使用系统调用,并在可能的情况下使用其他更高级别的接口。

命令输入

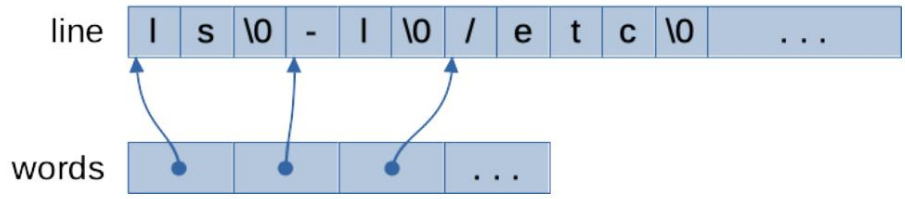
stash.c 程序将使用如下所示的提示提示用户输入命令。
大于号后有一个空格。您在下面看不到它,但您可以在本节后面的示例执行中看到它。

藏起来>

用户可以键入最长为 1024 个字符的命令（不包括行终止或任何空终止）。

存储程序需要将这个输入行分解成单独的单词。第一个词给出了用户正在运行的命令的名称,其余的词是命令行参数,将在命令运行时传递给该命令。在这里,单词只是一系列非空白字符。单词之间或输入行的开头或结尾可能有任意数量的空格。

您的程序将通过在每个单词的末尾放置一个空终止符来修改输入字符串。它将填充一个指针数组,其中一个指针指向每个单词的开头。下图显示了上面显示的示例命令的外观。



运行命令

将用户输入行分解成单词后,您就可以执行命令了。我们的 shell 将支持两种类型的命令。对于大多数命令,shell 会派生一个子进程,然后执行一个单独的程序。一些命令必须在 shell 本身内部实现,因此我们的 shell 将支持一些内置命令,这些命令不需要创建子进程或执行单独的程序。

内置命令

我们的 shell 将支持三个内置命令。下面列出了这些。对于这些,shell 将只检查命令的第一个单词。如果是“cd”,shell 将运行内置的 cd 命令。如果是“exit”,shell 将运行内置的exit 命令。如果输入行为空,shell 将忽略该命令并再次提示用户。

- cd 路径

  cd 命令需要一个参数。它通常是绝对路径或相对路径,但您不需要检查它。您只需将参数传递给 chdir() 系统调用即可更改当前目录。 chdir() 的返回值将告诉您目录是否有效。

  如果用户为 cd 输入了错误数量的参数,或者如果他们给出了从 chdir() 返回错误的路径,程序应该将以下错误消息打印到标准输出并提示用户输入另一个命令。

  命令无效

- 退出状态

  exit 命令以给定的整数值作为其退出状态终止 shell。对于 exit 命令,您可以将给定状态解析为整数,然后使用该值作为参数调用 exit() 。这将以该值作为其退出状态终止 shell。

  如果用户为退出提供了错误数量的参数,或者如果他们提供了无法解析为整数的退出状态,则程序应将以下错误消息打印到标准输出并提示用户输入另一个命令。

  命令无效

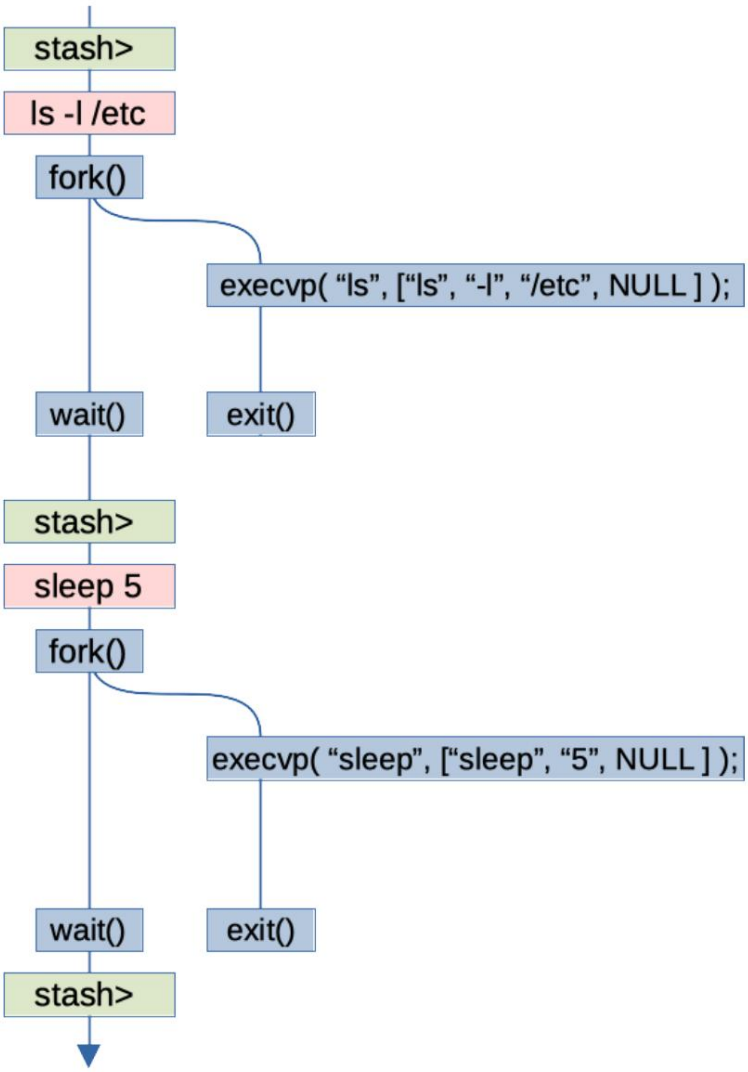
- empty

  如果用户输入一个空行,shell 将忽略它并提示用户输入另一个命令。

外部命令

除了内置命令外,所有输入的命令都将作为单独的程序运行。这意味着 shell 将使用 fork() 创建子进程,然后它将使用 execvp() 在子进程中运行不同的可执行文件。 execvp() 系统调用特别适用于此。它的第一个参数应该是您要运行的可执行文件的名称。此版本的 exec() 会自动查找 PATH 变量中的所有目录以查找具有给定名称的可执行文件,因此它应该能够仅根据可执行文件名称找到所有标准 shell 命令。 execvp() 的第二个参数是一个字符指针数组,就像您在将用户命令解析为单词时构建的那样。您需要将 NULL 指针添加到该数组的末尾。这就是 execvp() 可以判断有多少命令行参数的方式。

下图显示了您将如何运行外部命令。一开始,我们假装用户输入“ls -l /etc”作为第一个命令。 shell 通过创建一个孩子并让它通过 execvp() 运行 ls 命令来运行它。父进程等待子进程完成,然后提示用户输入另一个命令。如果用户输入“sleep 5”之类的命令,shell 将让另一个子进程运行该命令,然后等待该子进程完成。



无效命令

如果 execvp() 无法执行用户给出的命令,程序应将以下错误消息打印到标准输出并提示用户输入另一个命令。在这里,cmd 是他们输入的命令名称,用户命令的第一个单词。

无法运行命令cmd

如果用户为命令输入了无效选项,您无需执行任何特殊操作即可检测到这一点。
在你用 execvp() 运行它之后,每个外部命令都会自动检查它的参数并在它们有任何问题时抱怨。

设计

您的解决方案应包括以下功能。有了这些功能,我发现其余的实现并不太复杂。当然,如果你愿意,你可以添加其他功能。只是

确保至少实施和使用以下内容：

- int parseCommand(char *line, char *words[])
  此函数将用户命令（行）作为输入。如上所述，它将一行分成单独的单词，在单词之间添加空终止符，因此每个单词都是一个单独的字符串，并在单词数组中填充一个指针以指向每个单词的开头。它返回在给定行中找到的单词数。 words 数组的长度应至少为 513 个元素，因此它有足够的空间容纳 1024 个字符的输入行中可能容纳的最大可能数量的单词。

- void runExit(char *words[], int count)
  此函数执行内置的退出命令。 words 数组是指向用户命令中单词的指针列表，count 是数组中的单词数。

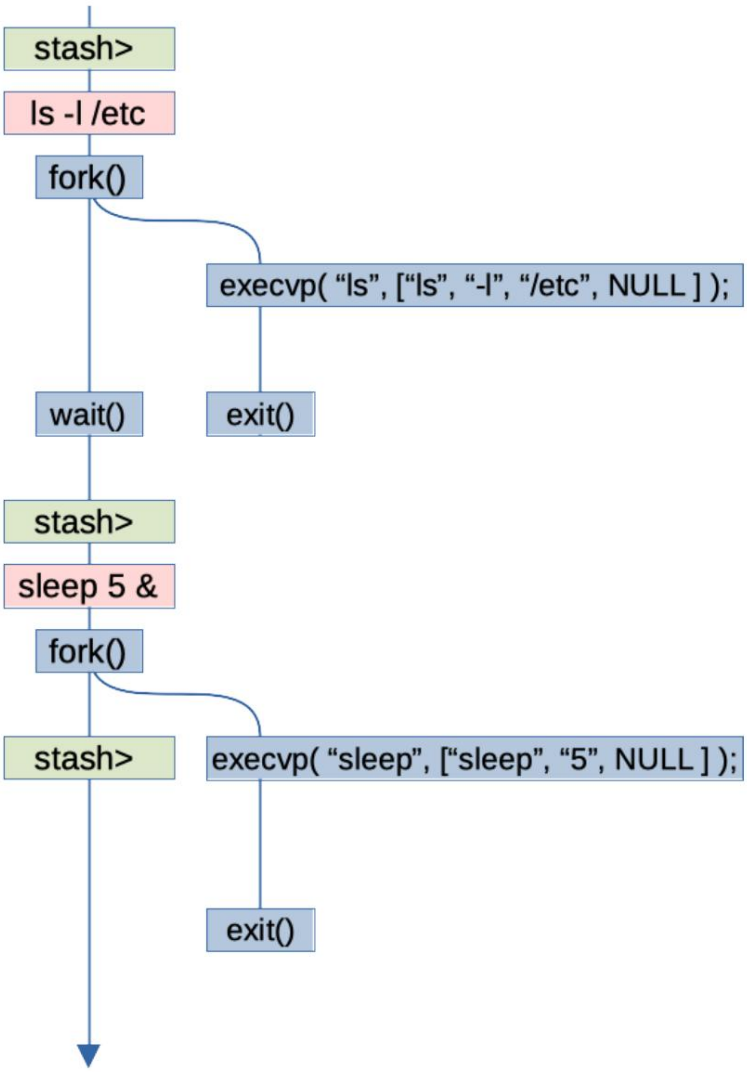
- void runCd(char *words[], int count)
  此函数执行内置的 cd 命令。与 runExit() 一样，参数给出了用户输入的命令中的单词。

- void runCommand(char *words[], int count)
  此函数通过创建子进程并让它调用 execvp() 来运行给定命令来运行（非内置）命令。

额外学分

加分8分，可以实现对后台运行命令的支持。
用户可以通过在末尾放置一个符号来在后台运行命令。符号将是命令的最后一个字；您不必在命令行上寻找属于其他单词一部分的符号。

在后台运行命令意味着 shell 在提示用户输入另一个命令之前不会等待命令完成。它运行命令然后开始尝试立即读取另一个命令。下面显示了与上面相同的命令的执行，但是 sleep 命令在后台运行。在这种情况下，shell 在启动 sleep 命令后立即提示用户输入另一个命令；它不会等待睡眠结束。



当您在后台启动命令时,您将打印出方括号内子进程的进程 ID。这为用户提供了一种在后台命令启动时跟上它们的方法。

因此,如果睡眠命令的进程 ID 为 12345,则在后台运行睡眠命令可能如下所示。

隐藏 > 睡眠 5 & [12345]

后台命令最终将完成。如果你做了额外的功劳,你的 shell 应该报告在用户运行下一个（非内置)命令后完成的任何后台命令。 （这是用户在后台命令完成后运行的第一个命令,不一定是用户输入的下一个命令。)已完成的后台命令的输出应该给出已完成的后台命令的进程 ID,后跟单词“完成”,都在方括号内。例如,如果用户在上面的 sleep 命令完成后运行 date 命令,您可能会得到以下结果（如果 sleep 命令的进程 ID 为 12345）。

藏匿>日期
[12345完成]
2023 年美国东部时间 1 月 11 日星期二 22:12:08

样本执行

下面显示了我们的 shell 的执行。我在下面包含了注释来解释这个例子在做什么（以#开头的行）。当您尝试您的程序时,不要实际键入这些（我们的 shell 不希望能够忽略注释）。

启动我们的 shell 程序。 $ ./藏起来

运行 ls 命令（外部命令）。我在我处理家庭作业 0 的同一 # 目录中执行此操作,因此您可以看到此作业中
的一些其他文件。隐藏> ls

exclude.c input-1.txt input-2.txt Makefile stash stash.c

尝试带有一些参数的 shell 命令。 stash> echo 这是一个有多个参数的
测试 这是一个有多个参数的测试

创建一个目录并尝试 cd 命令。隐藏> mkdir 测试

存储> cp input-1.txt 测试存储> cd 测试

确保我们在刚刚更改的新目录中。我在 EOS linux 机器上用我的帐户做#this。运行此命令时,您会看到不同的
路径名。隐藏> pwd /mnt/ncsudrive/s/sjiao2/MyCSC246/hw0/test

查看我们复制到该目录的文件。 stash> cat input-1.txt 这是排除程序的示例输入文
件。

你可以给一个命令行

告诉它要省略哪一行的参数

复制到输出时。

尝试一个空白命令。它应该被忽略,我们应该 # 得到另一个提示。藏起来>

尝试一些无效的内置命令。藏起来> CD

命令无效

stash> cd 参数过多
命令无效

stash> exit 错误参数

命令无效

```
# 尝试运行一个不存在的命令。 execvp()
# 调用应该在这里失败。
stash> 不存在的命令
无法运行不存在的命令命令

# 尝试退出命令。藏起来>出口200

# 回到常规 shell,确保我们得到了我们指定的退出状态。 $回声$?

200
```

提交

完成后,在名为 HW0 Q4 on Grade scope 的作业下提交源文件 stash.c。您不需要为此问题提交任何其他文件,也不要将您的源文件放在存档或子目录中。只需将文件 stash.c 直接提交到 HW0 Q4 提交柜即可。