外壳膨胀

本练习是关于研究壳扩展。您应该在 Vagrant 中的 Debian VM 上运行它。

`arguments.c` 创建一个包含以下内容的C 程序。 `nano arguments.c` 例如，您可以用于此目的。

```
#include <stdio.h>

int main(int argc, char** argv) {
  for(int i=0; i < argc; i++) {
    printf("Argument #%i: [%s]\n", i, argv[i]);
  }
  return 0;
}
```

用 编译这个 `gcc -Wall arguments.c -o arguments` 。

空白

该程序打印其所有参数，每行一个。该程序从启动它的程序（在本例中为 shell）获取参数。尝试使用以下命令运行该程序：

```
./arguments
./arguments hello
./arguments one two three
```

现在您已经熟悉了该程序的功能，请尝试以下操作：

```
./arguments one       two
./arguments "one two"
./arguments "one      two"
```

根据这些示例，shell 如何处理您键入的行中的空格？

模式匹配

请尝试以下操作：

- `./arguments *` 在包含arguments 程序及其源代码arguments.c 的文件夹中。

- 使用 创建一个空子文件夹 `mkdir empty` ，使用 切换到该子文件夹 `cd empty` ，然后运行 `../arguments *` 。由于您现在位于子文件夹中，因此我们需要在开头加两个点来表示“运行*上面*文件夹中的程序参数”。会发生什么？
- 通过运行返回到包含该程序的文件夹 `cd ..` ，然后执行 `ls` 检查是否返回到正确的文件夹。在此文件夹中，找到三种不同的方法使程序产生以下输出：

```
Argument #0: [./arguments]
Argument #1: [*]
```

名称中含有空格的文件

该命令 `touch FILENAME` 创建一个文件。通过键入 来创建名称中包含空格的文件 `touch "silly named file"` 。如果你去掉引号会发生什么（你可以尝试一下，然后做 `ls` ）？

开始输入 `ls sill` ，然后按 TAB 键自动完成。假设您没有其他名称以*sill*开头的文件，会发生什么？使用此方法让参数程序打印以下内容：

```
Argument #0: [./arguments]
Argument #1: [Hello world!]
```

命令 `rm` （remove）再次删除文件。使用它来删除名称中带有空格的文件，使用多种方法之一让 shell 将空格传递给 `rm` .

外壳变量

在 shell 中， `VARIABLE=VALUE` 将变量设置为值并 `$VARIABLE` 检索其值。例如，要避免输入两次文件名：

```
p=arguments
gcc -Wall $p.c -o $p
```

扩展到 `gcc -Wall arguments.c -o arguments` . 如果要在单词中使用变量，可以使用大括号： `${a}b` 表示变量的值， `a` 后跟字母 `b` ，而 `$ab` will 表示变量的值 `ab` 。

像这样使用双引号变量是一个很好的做法，因为如果您尝试编译一个 `silly name.c` 名称中带有空格的程序，那么

```
program="silly name"
gcc -Wall $program.c -o $program
```

将扩展到

```
gcc -Wall silly name.c -o silly name
```

这会让你的编译器感到困惑，因为你告诉它编译三个名为 的源文件 `silly` ， `name.c` 以及 `name` 一个名为 的程序 `silly` 。正确的是：

```
program="silly name"
gcc -Wall "$program.c" -o "$program"
```

扩展到

```
gcc -Wall "silly name.c" -o "silly name"
```

它可以满足您的需求 - 如果您确实想要一个名称中带有空格的程序！

每次要使用 shell 变量时都用双引号引用它并没有什么坏处，这是一个很好的做法，因为如果变量设置为包含空格的值，它仍然有效。

请注意，我们还必须首先引用设置变量名称，因为

```
program=silly name
```

将翻译为：将变量设置 `program` 为值 `silly` ，然后执行程序 `name` 。尽管您可以分配多个变量，但变量分配仅适用于其后面的第一个参数。

请注意，这也不会按预期工作：

```
file=arguments gcc -Wall "$file.c" -o "$file"
```

这里的问题是，shell 在开始执行命令之前首先读取该行并替换为 `$file` （默认情况下未设置的变量扩展为空字符串）的值，因此您在写入变量之前读取变量的值。省略引号没有帮助：您需要在单独的行上设置变量。