实验一：简单shell的设计和实现

* 实验内容

利用课本第9页程序1-5的框架，实现允许输入命令带参数的简单shell。原来的实现是不能够带参数的。输入命令所能带的参数个数，只受到系统键盘输入缓冲区长度（以及shell输入缓冲区长度）的限制，该缓冲区的缺省长度是4096个字节。

实现时要解决的主要问题有：

1. 正确确理解并使用系统调用fork()，execve()和waitpid()，特别是execve()函数。fork()函数创建一个新的进程。新进程就是所谓的子进程，它是执行fork()函数的进程(父进程)的“克隆”，也就是说，子进程执行的程序与父进程的完全一样。当fork()函数返回值为0时表示处于子进程中；而返回值大于0时表示处于父进程中，此时的返回值是子进程的进程id。因此，fork()的返回值可以用来划分仅仅适合父进程和子进程执行的程序段。fork()函数返回值为-1时表示出错。

如果子进程只是运行与父进程完全一样的程序，那用处是很有限的。要让子进程运行不同于父进程的程序，就必须调用execve函数，它是所有其他exec函数的基础。execve函数把调用它的进程的程序，替换成execve函数的参数所指定的程序。运行execve函数成功后，进程将开始运行新的程序，也就是execve函数的参数所指定的程序。

execve函数原型：

int execve(const char \*path, const char \*argv[],const char \*envp[]);

其中：

path：要执行的程序路径名，比如“/bin/ls”，“cd”，“/usr/bin/gcc”等等。

argv：参数表，比如ls命令中可带的命令行参数-l,-a等。注意，argv的第一个元素必须是要执行的程序（命令）的路径名。

envp：环境变量表，供要执行的命令使用。实参数用NULL或系统环境变量environ均可。注意，因为environ由系统提供，属于外部变量，所以说明时必须用“extern”修饰。

2. 根据简单shell的输入，构造execve函数的参数。

根据程序1-5，数组buf保存用户的输入，包括命令和参数。由于shell命令的命令名和各参数之间是用空格分开，因此可以用空格作为分界符。通过一个循环可以把buf数组中的命令和各个参数依次分离开来，并赋给数组argv的各元素适当的指针值。argv数组的最后一个指针必须是NULL。接着就可以调用execve(argv[0],argv, environ)来执行用户输入的命令。

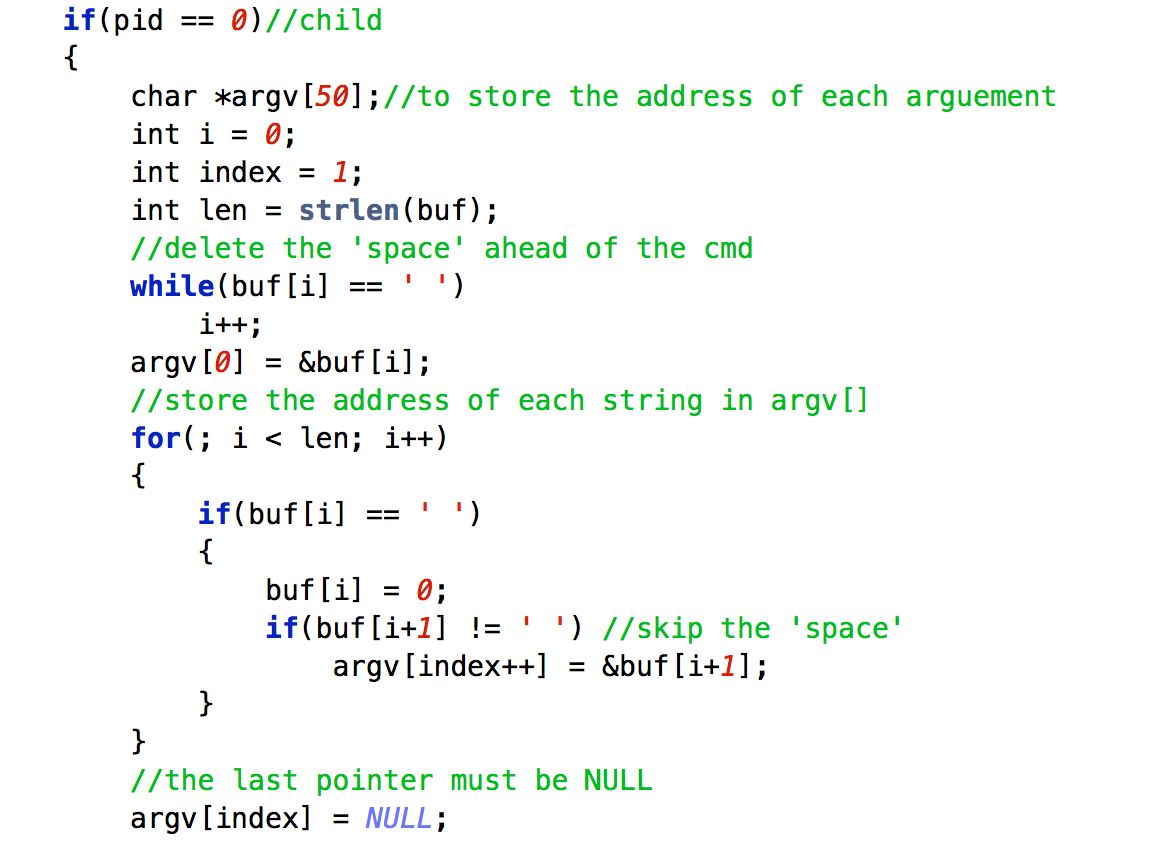
提示：argv数组中各指针所指向的字符串，可以直接利用buf的存储空间，不需要另外分配内存。

3. 正确编译程序。

由于书中例子用到了本书第一版中作者自己定义的error.c文件，因此编译时要记得把error.c文件和源代码文件一起编译，error.c文件可以到cs8的/home/luwei/common目录下复制。可以用如下命令编译：

gcc <源文件名> error.c –o<可执行文件名>

* 设计过程

概括来说，解决的核心问题就是如何将用户的输入字符串 “/bin/ls -l -h”, 以 {“/bin/ls”,“-l”,“-a”, NULL} 的形式存储到 argv[]中，也就是将一个字符串分成多个数组元素。 代码实现如下：

1.首先申请用于存放字符串指针的数组argv，这里设定申请大小为50；

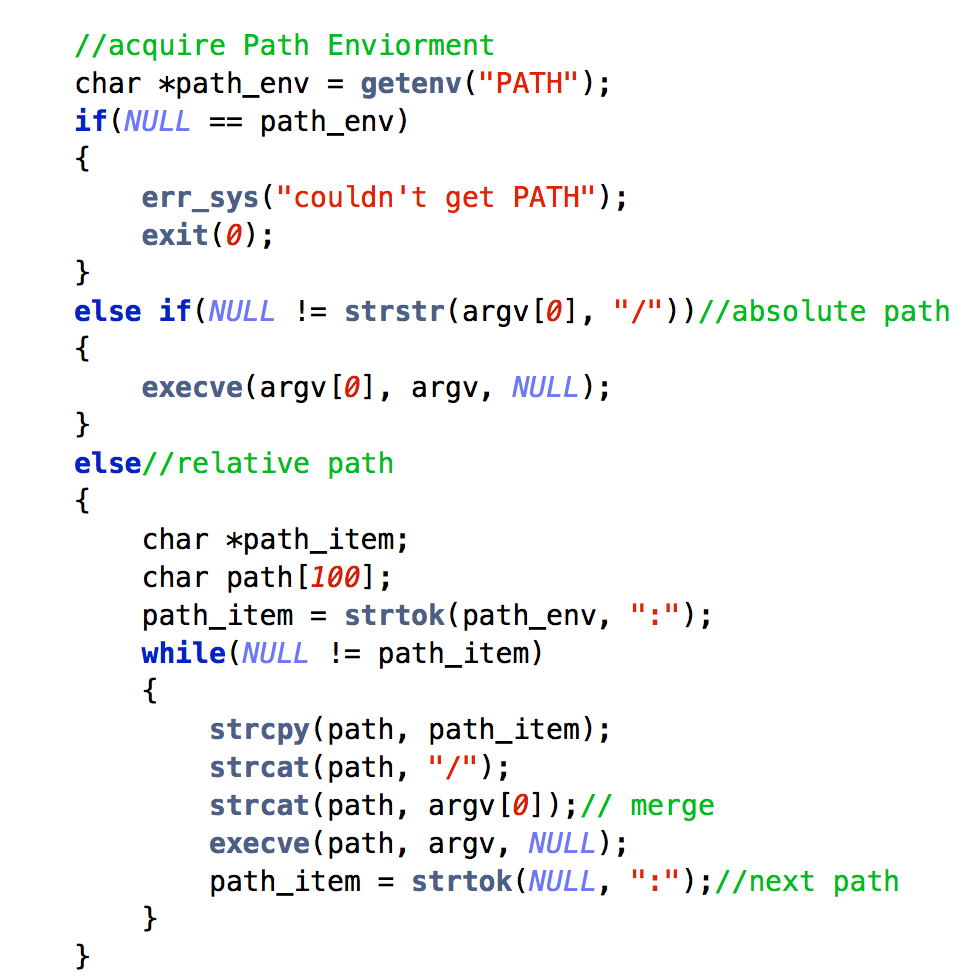
2.为防止输入的命令首部有空格，所以先用while循环去掉首部空格；

3.然后再将每个字符串的地址存入argv中，因为字符串之间以空格分隔开来，连续多个空格时，从第二个开始一概跳过；

4.将argv中最后的字符指针设为NULL。

至此，argv字符指针数组赋值完毕。

书上的程序中，execlp(buf, buf, (char \*)0) 是可以处理 ls -l 这样的命令的，因为 execlp() 会根据系统中的PATH路径（通过命令 echo $PATH 可以看到）来查找程序，也就是最终会查找到 /bin/ls 并执行。

但 execve() 不具备这样的功能，如果要实现类似 execlp() 的功能，则需要自己去遍历PATH路径，在每个PATH目录下尝试运行。

1. 函数getenv(“PATH”)用于获取PATH系统变量；
2. 由于系统的环境变量是以“：”分隔开来的，这里调用函数strtok(path\_env, “:”)讲路径拆分出来；
3. 如果输入的命令已经是绝对路径，如：/bin/ls -a则只需直接执行execve()函数；
4. 否则，我们通过对拆分出来的路径和输入命令的整合，得到绝对路径，再执行execve()函数。

* 实验结果

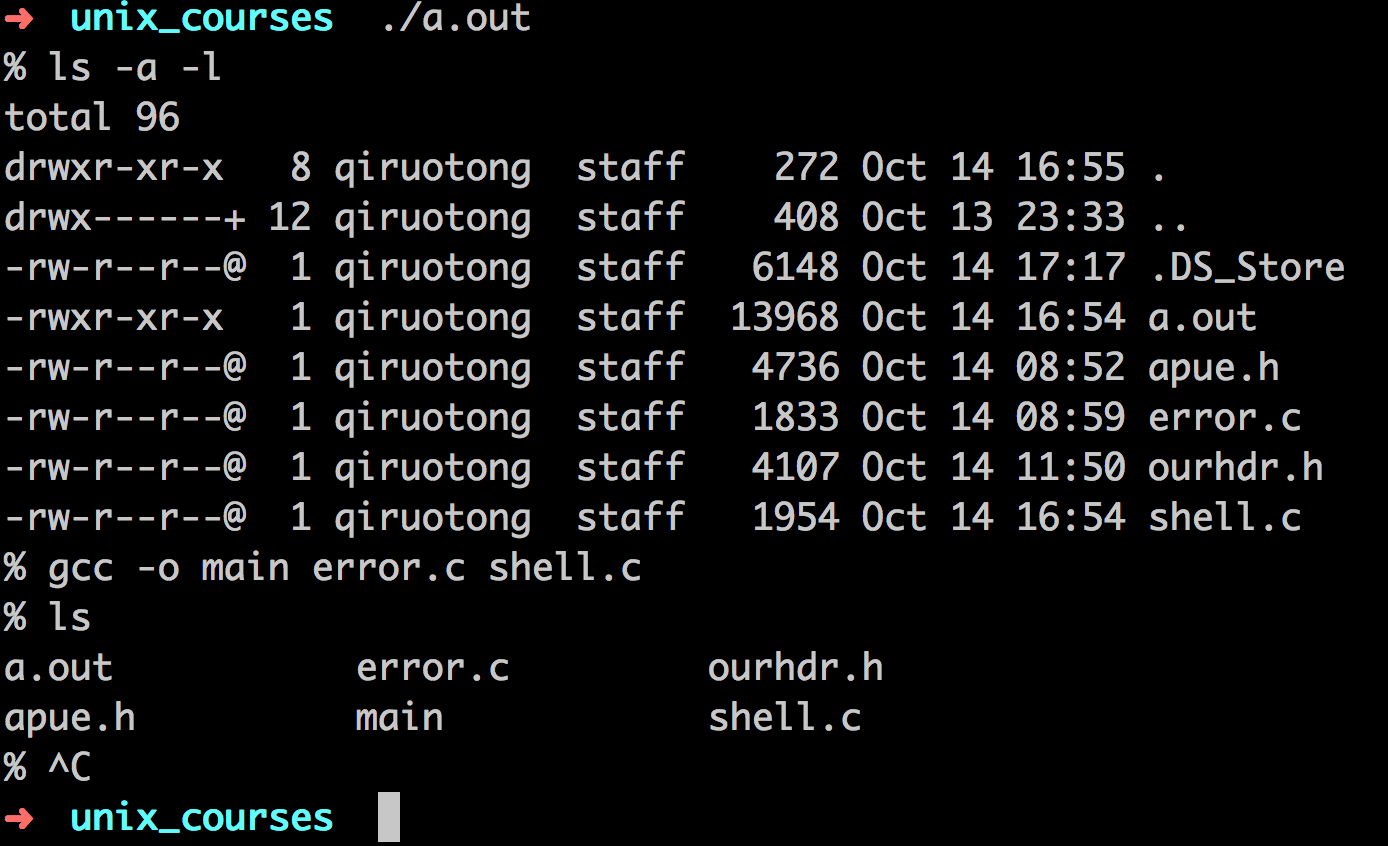
本程序由：**error.c , shell.c, apue.h, ourhdr.h** 这4个文件构成。

编译过程：

1. 执行语句：**gcc shell.c error.c**  进行编译；

2. 编译通过后会生成一个名字为**a.out** 的可执行程序；

3. 执行语句 **./a.out 即可开始输入 命令；**

**4. 如果想退出程序，则执行Ctrl + C 或者 Ctrl + D 即可。**

**结果如上图所示。**

* 个人体会

**很不习惯在shell中编程，很难从windows过渡。各种命令还不是很熟悉。实现这个程序的过程是极为艰辛的。但在shell中编程很爽，编译速度也是windows平台无法比拟的。**