Unix 实验报告

实验： 实验 3 简单 shell 的设计和实现

专业： 计算机科学与技术

班级： 1 班

姓名： 姚怀聿

学号： 22920202204632

2022 年 11 月 5 日

目　　录

一、　实验内容描述 2

二、　设计、实验构思 2

三、　实验结果 7

四、　实验心得与建议 9

五、　完成人姓名及完成时间 10

一、 实验内容描述

实验目的：

1、练习使用编程环境，包括 shell 的命令，vi 和 gcc 编译器。

2、体会进程概念，了解 fork，execve，wait 等系统调用。

实验要求实现允许输入命令带参数的简单 shell。

实验具体要求如下：

1. 除了系统调用 execve，不允许使用其他的 exec 函数。输入应当允许带多个参数(一行内可以表示)，不考虑通配符(即“*”、“?”、“-”等等)的处理。
2. 输入错误命令能提示出错并进入下一轮接收命令状态。
3. 可以用 Ctrl-C 和 Ctrl-\结束简单 shell 的运行。
4. 程序运行正确，提示简洁明确。

二、 设计、实验构思

1. 正确理解并使用系统调用 fork()和 waitpid()

程序中利用 fork()函数创建一个新的进程，即子进程，并在子进程中调用 execve()函数，实现指令的执行。在父进程中，调用 waitpid()函数，等待子进程终止后父进程执行，继续进

行下一轮的指令输入和执行。

实现函数 print_promt()，该函数用于打印输入提示字：

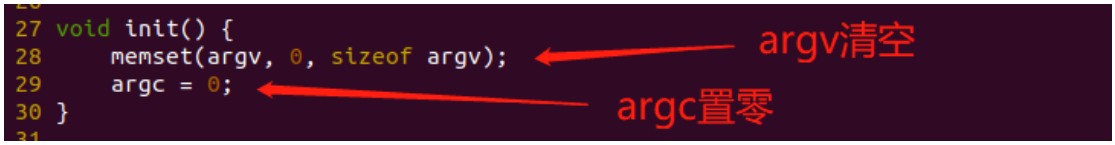
```
18 void print_promt(char* string) {
19     if(getcwd(string + 9, MAXLINE) == NULL) {
20         perror("getcwd error: ");
21         exit(1);
22     }
23     strncat(string, "]$", MAXLINE - 1);
24     printf("%s", string);
25 }
```

运行程序后，首先会打印出一行输入提示字，并显示当前工作路径，然后 while 循环输入，输入的内容存入字符数组 buf 中，调用函数 read_from_buf()，将 buf 中的内容用空格分开，存入字符串数组 argv 中，并将命令和参数的个数保存在变量 argc 中，该函数在下面会具体介绍。之后调用 fork()函数创建子进程，并在子进程中调用 execve()函数执行命令。父进程中调用 waitpid()等待子进程的状态更改。


```
84      print_promt(promt);
85      while(fgets(buf, MAXLINE - 1, stdin) != NULL) {
86          init();
87          read_from_buf(buf);
88          if(!strcmp(argv[0], "cd")) {
89              if(argv[1] == NULL || !strcmp(argv[1], "~")) {
90                  argv[1] = getenv("HOME");
91              }
92              if(chdir(argv[1]) < 0) {
93                  perror("chdir error: ");
94                  exit(1);
95              }
96          }
97          else if(!strcmp(argv[0], "pwd")) {
98              char *curpath;
99              curpath = getcwd(NULL, 0);
100             if(curpath == NULL) {
101                 perror("getcwd error: ");
102                 exit(1);
103             }
104             fprintf(stdout, "%s\n", curpath);
105         }
106         else {
107             if((pid = fork()) < 0) {
108                 perror("fork error: ");
109                 exit(1);
110             } else if(pid == 0) {
111                 redirect(rf, wf);
112                 execve(argv[0], argv, environ);
113                 perror("execve error: ");
114                 exit(1);
115             } else {
116                 if((pid = waitpid(pid, &status, 0)) < 0) {
117                     perror("waitpid error: ");
118                     exit(1);
119                 }
120             }
121         }
122         print_promt(promt);
123     }
124     exit(0);
```

init 函数实现如下：

对全局变量 argv 和 argc 进行初始化




```
27 void init() {
28     memset(argv, 0, sizeof argv);
29     argc = 0;
30 }
```

2. 构造 execve 函数的参数

在 APUE 第三版课本程序上进行修改，字符数组 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隔开的，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隔开，并将其中的每一项取出来赋给字符串数组 argv，int 类型的 argc 用于记录命令和参数的总个数，每次读取前需要先初始化，将 argv 数组清空，argc 置零。注意 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。

该功能用函数 read_from_buf()实现，具体如下：

```
32 void read_from_buf(char *buf) {
33     int type = 0;
34     rf = NULL;
35     wf = NULL;
36     if(buf[strlen(buf) - 1] == '\n') buf[strlen(buf) - 1] = 0;
37     char *token = strtok(buf, " ");
38     while(token != NULL) {
39         if(type == 1) { // rf
40             rf = token;
41             type = 0;
42         } else if(type == 2) {
43             wf = token;
44             type = 0;
45         } else if(!strcmp(token, "<")) type = 1;
46         else if(!strcmp(token, ">")) type = 2;
47         else argv[argc++] = token;
48         token = strtok(NULL, " ");
49     }
50     argv[argc] = NULL;
51 }
52
```

type 用于指示当前参数的类型，0 表示常规，1 表示读文件，

2 表示写文件。rf 和 wf 均为全局变量。

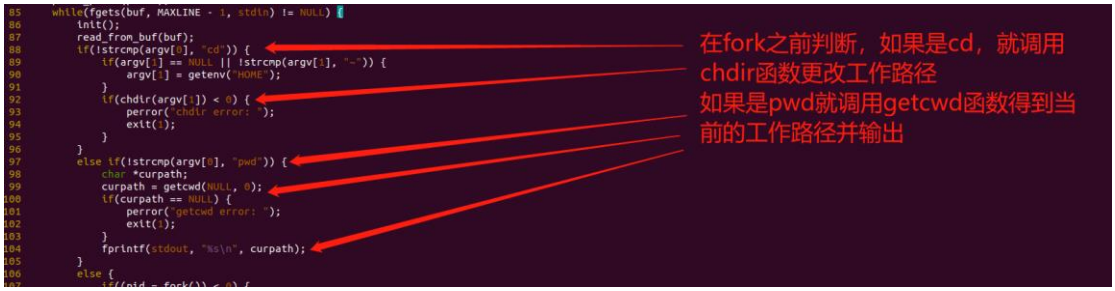
首先对 buf 整体进行处理，将最后一个换行符修改为结束符'\0'。

然后调用函数 strtok()，将 buf 用空格分开。分开后得到的每个参数放入 argv 中，并让 argv 的最后一个单元为 NULL。

3. cd、pwd 和输入输出重定向的实现

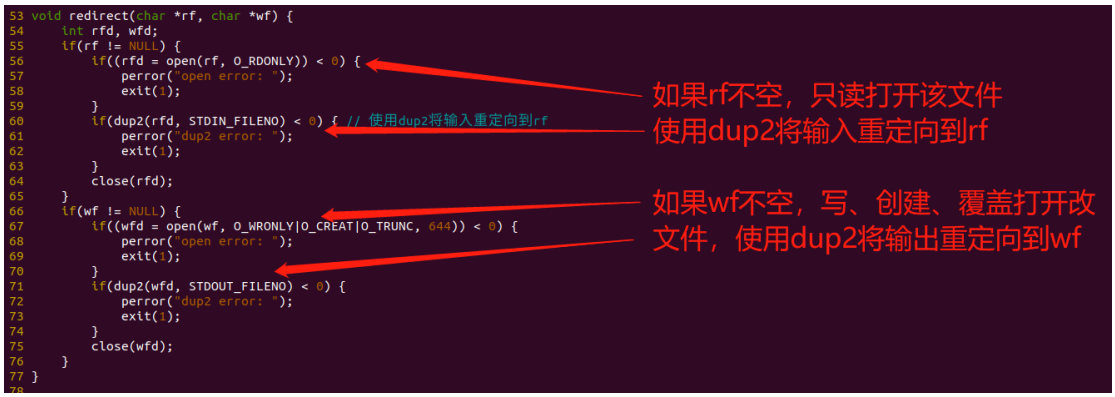
其实到这里，大部分简单 shell 的输入都已经可以实现了，但是该程序还不能实现如 cd、pwd 以及输入输出重定向的操作，于是，我附加实现了该功能。实现 cd 和 pwd 功能需要在 fork 之前实现，而不要在子进程中去实现，否则会有一些小 bug(比如输入 cd 后必须要再按一次回车才能执行成功)。而实现输入输出重定向就需要用到我定义在全局的 rf，wf 以及 dup2 函数了。具体程序如下：

1) cd 和 pwd 的实现



2) 输入输出重定向

该功能使用 redirect()函数实现，具体如下：



至此，shell 基本功能都可以实现了。

三、 实验结果

源程序名	可执行程序名
shell.c	shell

编译生成可执行文件：

使用如下指令：

`make shell`，即可得到可执行文件`shell`

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_3$ ls
shell.c
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_3$ make shell
cc     shell.c   -o shell
```

运行程序：

1. 使用`./shell`运行程序：

```
rongrong@rongrong-virtual-machine:~/Desktop/huaiyuyao/unix/homework/homework_3$ ./shell
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$
```

观察到输入提示字显示正常。

2. 输入`/bin/ls -l -a`：

```
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$/bin/ls -l -a
total 32
drwxrwxr-x 2 rongrong rongrong  4096 11月  5 16:21 .
drwxrwxr-x 6 rongrong rongrong  4096 11月  4 14:40 ..
-rwxrwxr-x 1 rongrong rongrong 18880 11月  5 16:21 shell
-rw-rw-r-- 1 rongrong rongrong  3066 11月  5 15:34 shell.c
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$
```

观察到结果正确。

3. 输入`/bin/gcc -g -c shell.c`

```
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$/bin/gcc -g -c shell.c
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$/bin/ls
shell  shell.c  shell.o
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$
```

观察到结果正确。

4. 输入`/usr/ls -l -a`

```
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$/usr/ls -l -a
execve error: : No such file or directory
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$
```

观察到结果正确，目录/usr/下没有 ls 程序。

5. 输入`cd`

```
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$cd
[Myshell /home/rongrong]$pwd
/home/rongrong
[Myshell /home/rongrong]$cd Desktop/huaiyuyao/unix/homework/homework_3
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$pwd
/home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$
```

观察到结果正确。

6. 输入`pwd`

```
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$cd
[Myshell /home/rongrong]$pwd
/home/rongrong
[Myshell /home/rongrong]$cd Desktop/huaiyuyao/unix/homework/homework_3
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$pwd
/home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$
```

观察到结果正确。

7. 输入`/bin/cat shell.c > /tmp/out`

```
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$/bin/cat shell.c > /tmp/out
[Myshell /home/rongrong/Desktop/huaiyuyao/unix/homework/homework_3]$/bin/diff shell.c /tmp/out
```

使用 diff 命令查看两文件内容是否一致，无输出，说明两文件内容一致，结果正确。

四、 实验心得与建议

实验心得：本次实验实现的一个简单的 myshell 还是有很多的不足，虽然额外实现了 cd、pwd 这些简单的命令，但是还是不能支持真正 shell 的一些功能比如：tab 补全、回溯到上

一条指令等等。这让我体会到只是一个简单的 shell 黑窗口的开发难度也是不容小觑的，linux 能发展到今天这种程度离不开前人的智慧和创造力，我们要站在巨人的肩膀上不断学习、勇攀高峰。

建议：希望老师能够提前发布下一次实验课的任务，这样可以让我们有时间对实验进行预习，使得实验的完成更加顺利。

五、 完成人姓名及完成时间

完成人姓名	完成时间
姚怀聿	2022 年 11 月 5 日