# OS project two 实验报告
# ——UNIX Shell and History Feature

5140309201

黄晟

## 一、 实验要求

在linux中实现一个简单的shell，能够实现程序的后台执行，更改信号量，并且能够显示最近执行的命令的历史特性。

## 二、 实现过程

1、完成setup函数以及程序的执行函数，完成对于新读入的命令的预处理；
2、定义一个二维数组，用以存放历史进程；

```
char history[10][MAX_LINE];
```

3、编写完成打印历史特性函数，定义hiscount来记录当前要打印的命令；

```
void displayHistory()
{
    printf("\n\nCommand History:\n");
    int i;
    int j = 0;
    int hiscount = count;
    for (i = 0; i < 10; ++i){
        printf("%d. ", hiscount);
        while(history[i][j] != '\n' && history[i][j] != '\0'){
            printf("%c",history[i][j]);
            ++j;
        }
        j = 0;
        printf("\n");
        hiscount--;
        if (hiscount == 0) break;
    }
    printf("\n");
}
```

4、为了完成用r来调用最近执行的指令以及rx调用最近的首字母为x的一条命令，在setup函数读取输入之后加条件判断；

```
else if (args[0][0] - 'r' == 0){
    if (args [0][1] == '\0'){
        strcpy(inputBuffer,history[0]);
    }
    else{
        int k;
        for (k = 0; k < 10 && k < count; ++k){
            if (args[0][1] == history[k][0]){
                strcpy(inputBuffer,history[k]);
                break;
            }
        }
        if (k == 10 || k == count){
            printf("\nNo such command in history\n");
            strcpy(inputBuffer,"Wrong command");
        }
    }
}
```

5、同时，我添加了一项类似于收藏夹的功能，即利用‘prefer’快速调用用户想长期的某项指令，若用户想替换当前指令时，可以用“change+新的指令”来替换。Prefer初始自动设置为cal；

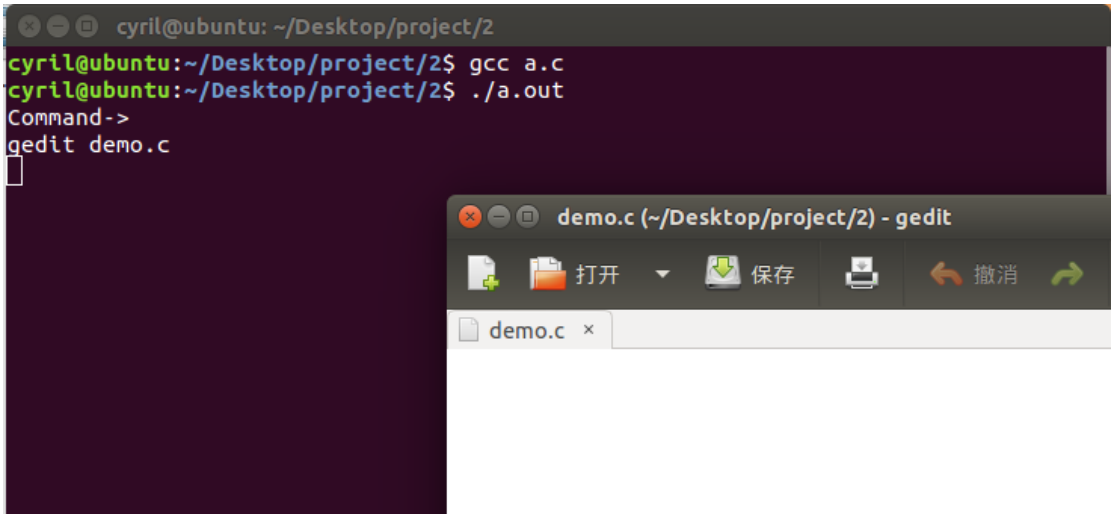
```
else if (strcmp(args[0],"prefer") == 0){
    strcpy(inputBuffer,prefer);
}
else if (args[0][0] == 'c' && args[0][1] == 'h' && args[0][2] == 'a'
&& args[0][3] == 'n' && args[0][4] == 'g' && args[0][5] == 'e'){
    int j = 6;
    while(args[0][j] != '\0' && args[0][j] != '\n'){
        prefer[j-6] = args[0][j];
        ++j;
    }
    inputBuffer[j-6] = '\0';
    return -1;
}
```

6、本实验要求采用“ctrl+c”来调用历史特性，但在实际实验过程中，我发现在用户模式下“ctrl+c”很不稳定，所以为了较好的显示历史特性，我又添加了同步调用历史特性的方法，即手动输入history命令输出历史特性。

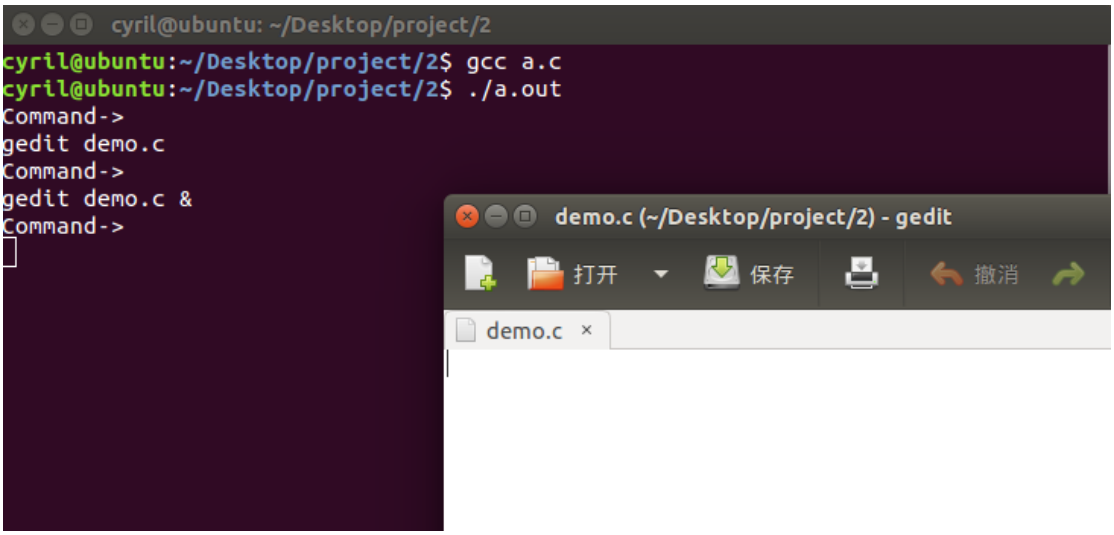
```
if (strcmp(args[0],"history") == 0){
    if (count > 0) displayHistory();
    else printf("\nno history command\n");
    return -1;
}
```

## 三、 实现效果

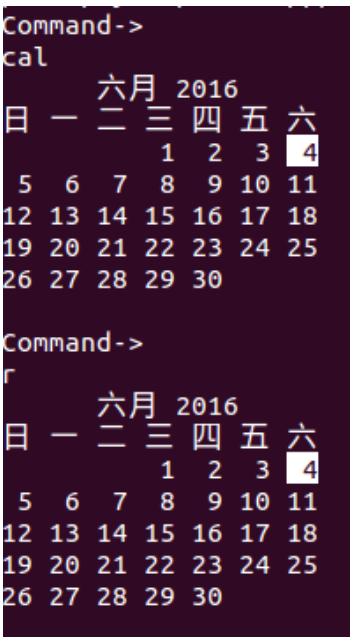
1、在shell中键入不带‘&’命令时，会等待改程序执行完后再键入新的指令；



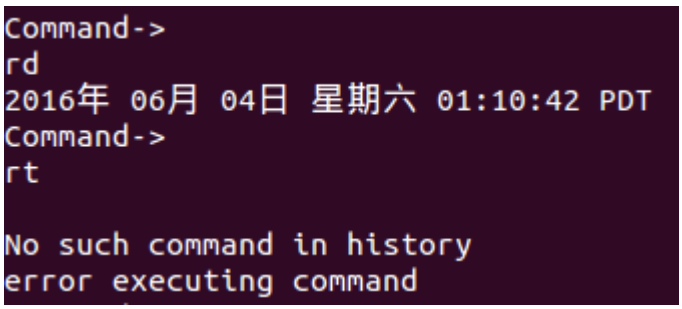

2、在shell中键入带有‘&’命令时，会将该指令后台执行，shell可以接受新的指令；



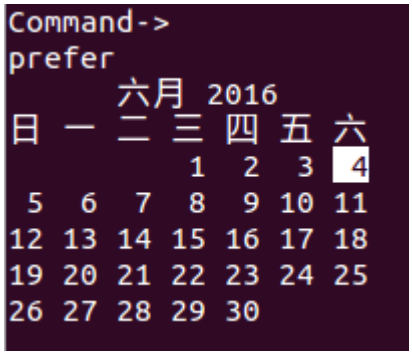

3、单独输入r时，会执行最近执行的指令；



4、 输入r+x时，会搜索并执行最近的以x开头的指令，若没有这样的指令，输出错误提示；

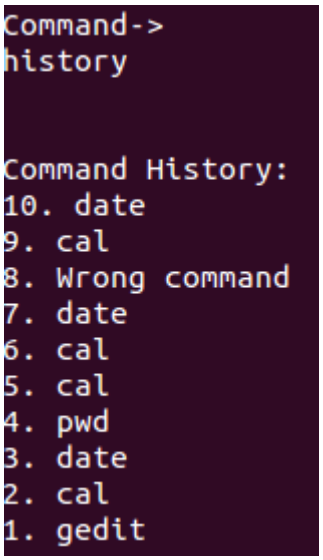



5、 键入prefer后会输出预存指令；




6、 使用change+指令后可以改变prefer，此时再调用prefer会执行新的指令；


7、 键入history后可以显示历史特性；



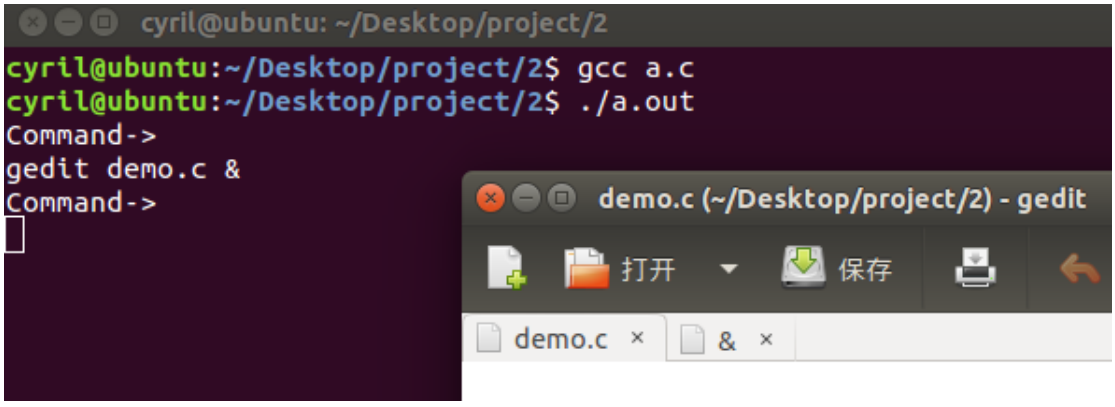

8、 而当使用“ctrl+c”时，shell会变得不稳定，如下图，当我输入异步ctrl+c时，系统会提示接收到两次ctrl+c，因而会同时输出两个历史特性，但只有一个是正确的。

```
Command->
^CCaught Control C
Caught Control C
```

## 四、 心得与体会

1、 最初在创建子进程时，等待是用的 wait(NULL)，这样就导致了执行过一次后执行程序后，之后的程序就都是后台执行的了。为了解决这个问题，首先我是采用了键入 exit 命令，手动退出后台执行程序；但是后来觉着这样做不方便，于是将 wait(NULL)替换成 waitpid(pid,NULL,0)，即等待特定进程。因为 wait(NULL)函数虽然会等待，但是它不关心返回状态，所以系统也不知道后台程序什么时候被终止，即之后的程序都是后台执行了。

2、 在执行后台文件时，会多出来一个文件&，如图所示：




这是因为在预处理指令时，检查到&后立即结束，添加‘\0’，然而由于 i++后，使得指令检测不到‘\0’了，所以要将 inputBuffer[i-1] = '\0';改为 inputBuffer[i] = '\0';即可解决该问题。

```
default:
if (start == -1)
    start = i;
if (inputBuffer[i] == '&'){
    *background = 1;
    inputBuffer[i-1] = '\0';
}
```

3、 在执行同步显示历史特性指令 history 时，会出现如图所示

```
Command->
history
3. history
2. pwd
1. date

Command->
error executing command
```

这是因为 history 不是 shell 固有指令，为了执行 history 并且不再出现错误提示，将 setup 函数的 void 改成 int，检测到输入 history 是返回-1，而在创建子进程时若 setup 函数返回-1，即不执行接下来的 fork 等语句。