**操作系统课程设计实验报告**

——实验一：shell实验

负责人姓名：吴勇宁

学号：14061119

日期：2016.4.8

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 吴勇宁 | 14061119 | 实验一 |
| 2 | 吴勇宁 | 14061119 | 实验二 |
| 3 | 吴勇宁 | 14061119 | 实验三 |
| 4 | 吴勇宁 | 14061119 | 实验四 |

目录

[1.实验目的 4](#_Toc446001831)

[2.需求说明 4](#_Toc446001832)

[2.1基本要求 4](#_Toc446001833)

[2.2 提高要求 4](#_Toc446001834)

[2.3 完成情况 4](#_Toc446001835)

[3.设计说明 5](#_Toc446001836)

[3.1 程序流程图 5](#_Toc446001837)

[3.2基本要求实现说明 5](#_Toc446001838)

[3.3 提高要求实现说明 5](#_Toc446001839)

[4.收获和感想 5](#_Toc446001840)

# 1.实验目的

1.学习Linux相关软件工具的使用（如gcc、gdb和make）；

2.熟悉使用Linux中YACC工具进行语法分析的基本方法；

3.运用man帮助手册查询相关命令；

4.理解并发程序的同步问题；

5.学习POSIX/UNIX系统调用的使用；

6.掌握进程控制和进程间通信的方法；

# 2.需求说明

## 2.1基本要求

1.程序能够正常运行；

2.能够执行fg、bg、cd、history、exit等内部命令；

3.能够执行外部程序命令，命令可以带参数；

4.使用I/O重定向；

5.支持前后台作业，提供作业控制功能，包括打印作业的清单，改变当前运行作业的前台/后台状态，以及控制作业的挂起、中止和继续运行；

## 2.2 提高要求

1.尝试对YACC语法分析的文法进行进一步的修改与完善；

2.尝试在Linux下将Lex和YACC结合起来使用进行词法和语法分析；

3.对其他常用的内部命令进行实现，并可以尝试考虑对通配符的支持与实现 ；

4.实现对管道的支持；

5.考虑并实现组合键ctrl+c命令；

## 2.3 完成情况

【简述实验完成过程】。完成了以下功能：

1.程序能够正常运行；

2.执行fg、bg、cd、history、exit等内部命令；

3.执行外部程序命令，命令可以带参数；

4.使用I/O重定向；

5.支持前后台作业，提供作业控制功能，包括打印作业的清单，改变当前运行作业的前台/后台状态，以及控制作业的挂起、中止和继续运行；

6.支持其他常用的内部命令；

7.对管道的支持；

8.组合键ctrl+c命令；

# 3.设计说明

## 3.1 程序流程图

初始化环境

打印提示符,获取用户输入的命令

解析命令

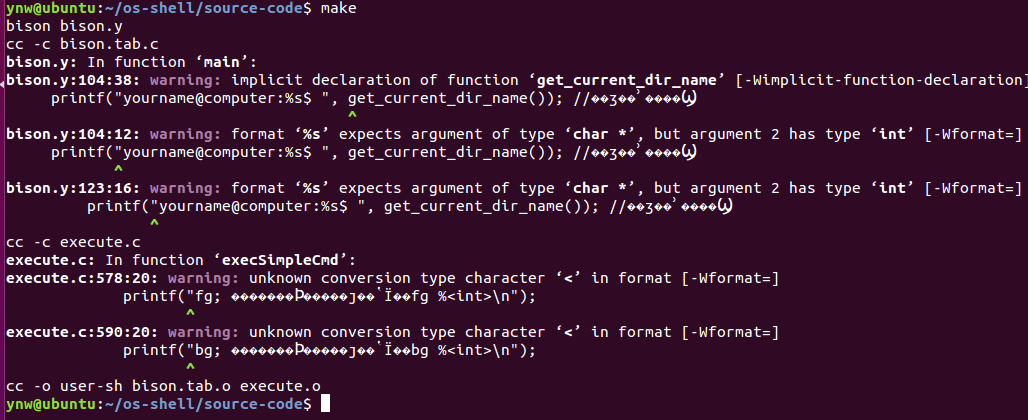
寻找命令文件

执行命令

程序结束

## 3.2基本要求实现说明

1. 解决源代码刚刚克隆下来存在的问题：

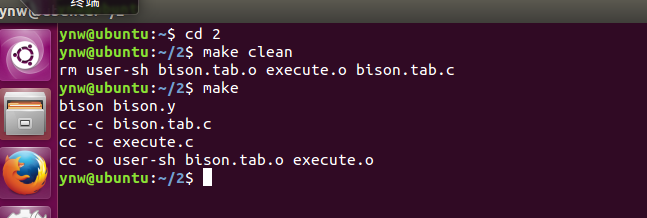
* 将源代码进行编译出现以下问题：

解决方法：

➀将bison.y文件中printf("yourname@computer:%s$ ", get\_current\_dir\_name()); 修 改为printf("user-shell> ");

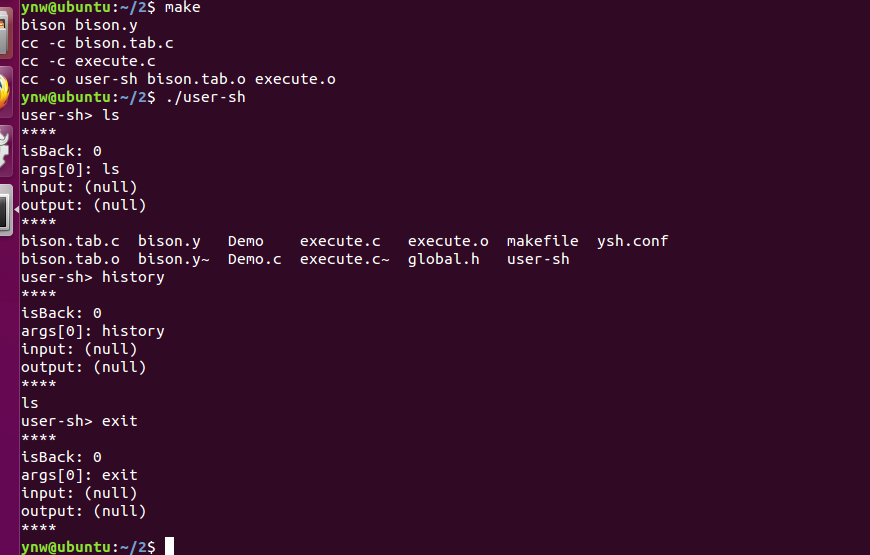
➁将execute.c文件中printf("fg; 参数不合法，正确格式为：fg %<int>\n");修改为printf("fg; 参数不合法，正确格式为：fg %%<int>\n");

➂将execute.c文件中printf("bg; 参数不合法，正确格式为：bg %<int>\n");修改为printf("bg; 参数不合法，正确格式为：bg %%<int>\n");

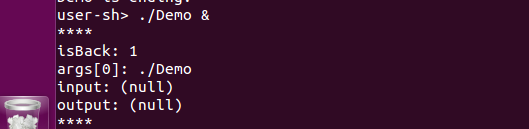


* 解决三个坑的问题：

➀在源代码的目录先使用系统自带的shell验证exit,history,jobs,fg,bg,Ctrl+Z,Ctrl+C的功能和现象，再使用自己编写的shell，即user-sh验证上述命令的功能，首先exit和history已经完成，没有明显的问题。

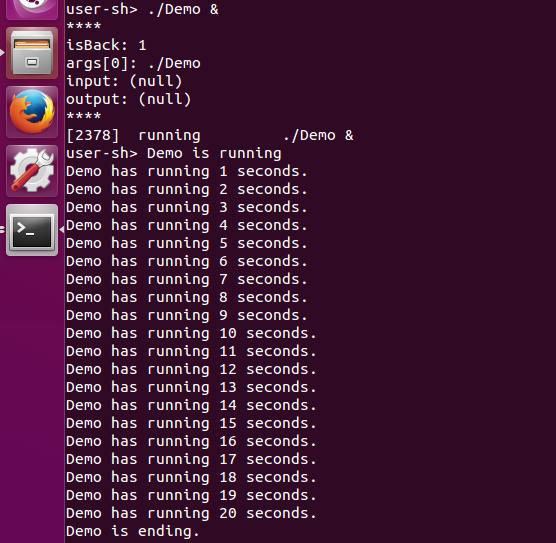


➁在user-sh里进行验证，借助Demo程序验证fg、bg命令时：

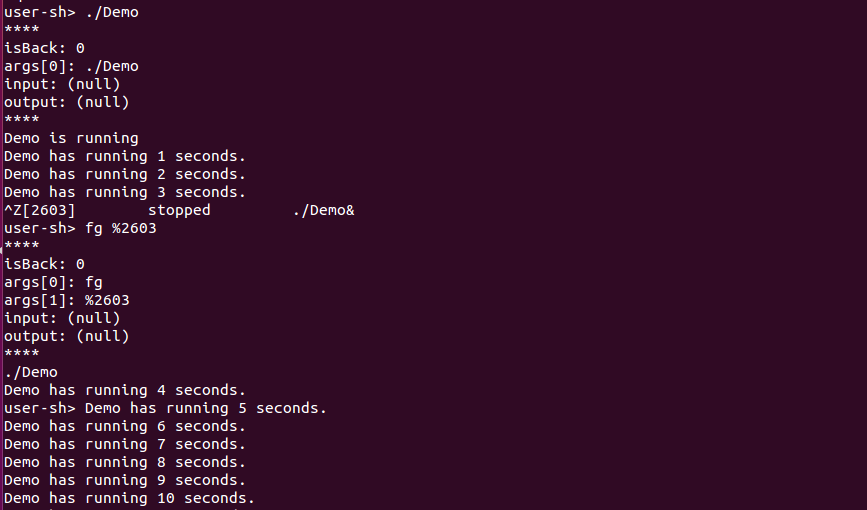


调试的结论：说明在父进程执行完kill(pid, SIGUSR1)命令后子进程被杀死；SIGUSR1、SIGUSR2等信号量默认会导致进程退出；保证子进程在父进程Kill函数执行前运行。

解决办法：在“kill(pid, SIGUSR1);”前增加sleep(1);//父进程kill前面睡一秒，等子进程开始运行

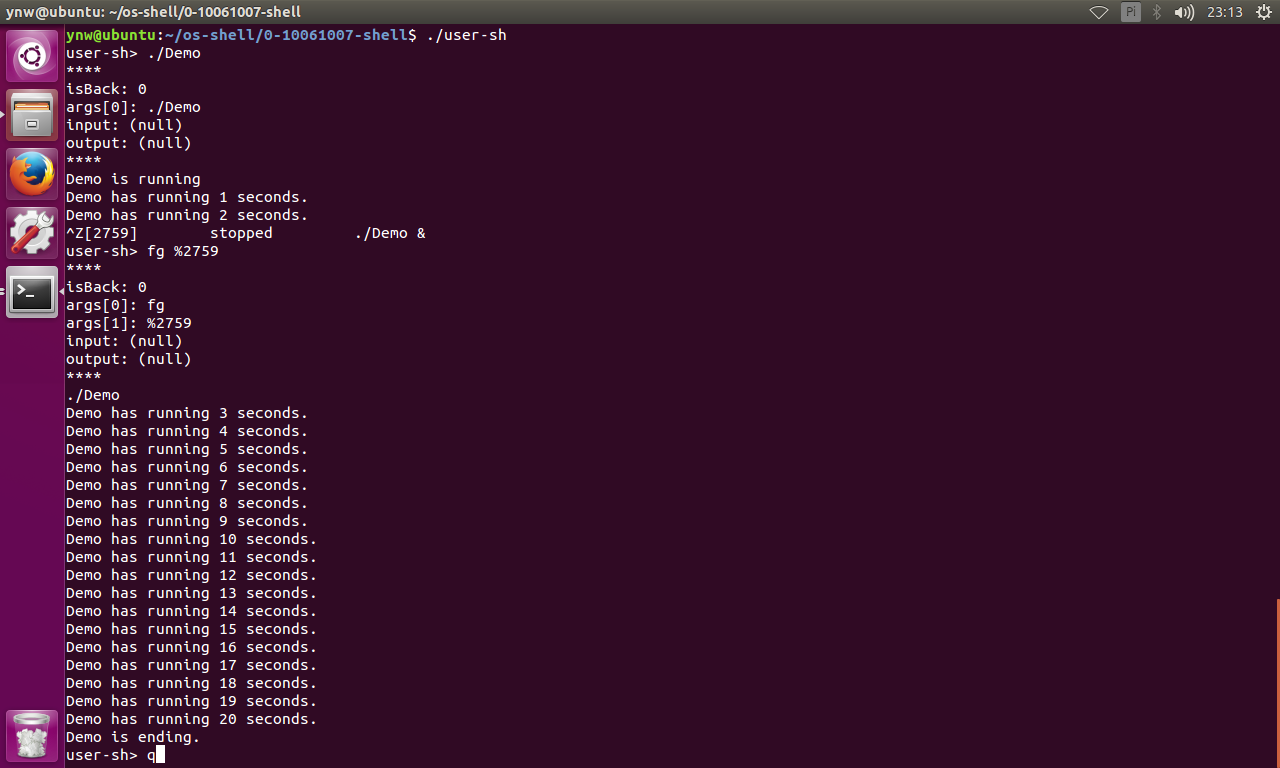


➂运行前台作业，执行ctrl+z命令挂起，再执行fg %<int>命令继续运行前台作业出现命令提示符在作业运行完之前就打印出来的错误，原因是父进程没有等待子进程执行完。

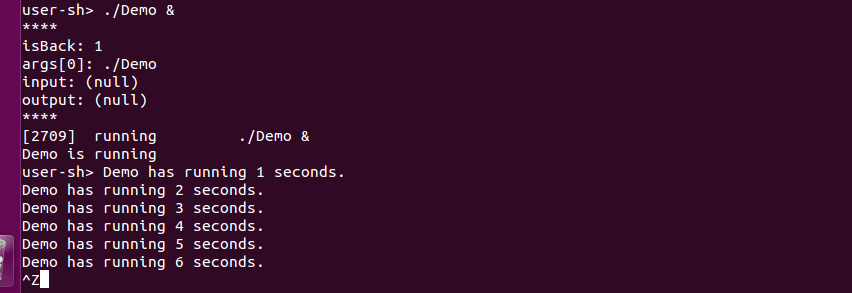


调试结论：wait函数要等待子进程收到开始启动进程的信号后运行才能得到正确的结果。

解决办法：在fg的信号代码里的“kill(now->pid, SIGCONT);”后增加sleep(1);



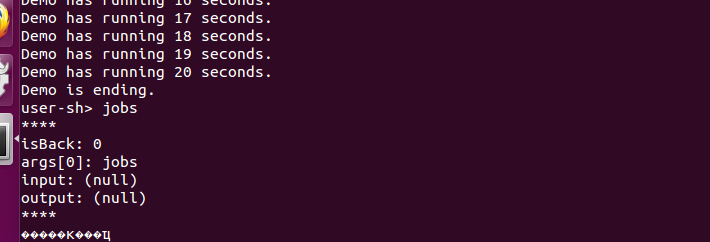
④运行后台作业过程中，键入ctrl+z会导致后台作业停止运行：



原因：子进程会继承父进程的信号处理方式，直到子进程调用exec函数；子进程调用exec函数后，exec将父进程中设置为捕捉的信号变为默认处理方式，其余不变。

方法：使用setpgid函数将子进程分配到独立的进程组，则不会捕获到SIGTSTP信号。setpgid()加入一个现有的进程组或创建一个新进程组。

⑤运行Demo文件后执行Ctrl+Z命令后无法识别其他命令



解决办法：修改bison.y的main函数，在while里添加while((c=getchar())==-1);ungetc(c,stdin);

## 3.3 提高要求实现说明

1. **管道的实现**

1.实现原理

在父进程中使用pipe系统调用创建一个管道，系统调用的参数是包含两个文件描述符数组pfd[0]和pfd[1]。

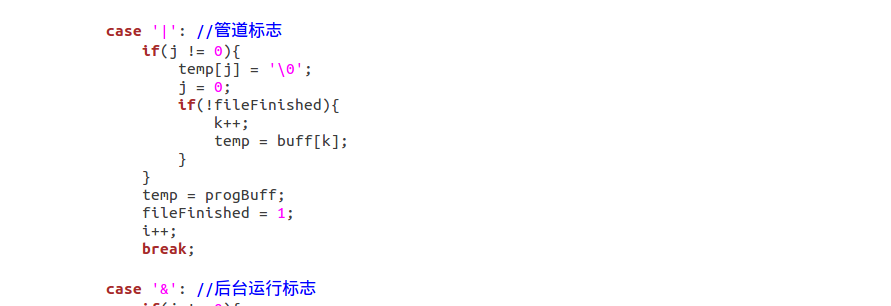
1. 使用pfd[0]作为管道输入描述符，使用pfd[1]作为管道输出描述符。
2. fork产生子进程。
3. 现在父进程和子进程都共享了管道文件描述符。每个进程都关闭管道的一端。
4. 每个进程使用dup2把打开的管道描述符副本给标准输入或者标准输出，然后关闭管道描述符。
5. 两个进程可以使用管道通过标准输入和标准输出进行通信。
6. 具体实现情况

未实现时：

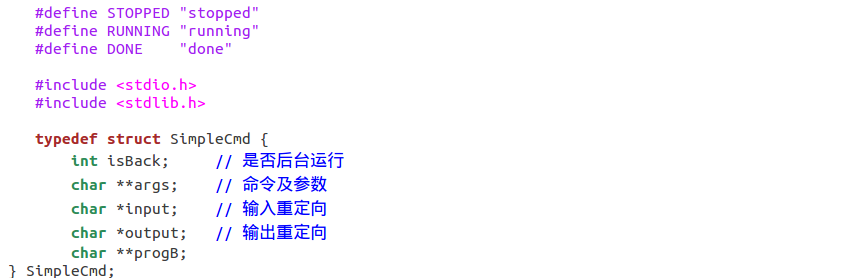


实现方法：

①首先在命令解析中加入对“|”的识别，因为管道的实现与重定向I/O机制基本相同，所以：



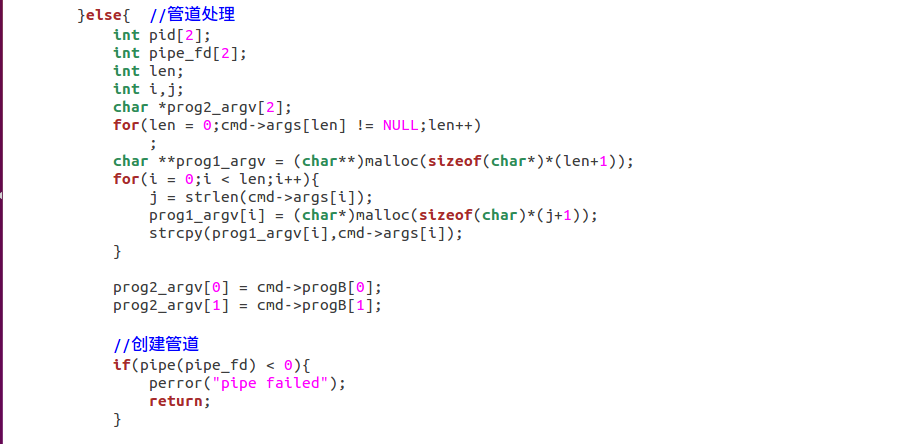
同时我们需要在global.h文件中添加二维数组用来记录管道“|”前个命令和后边的命令：

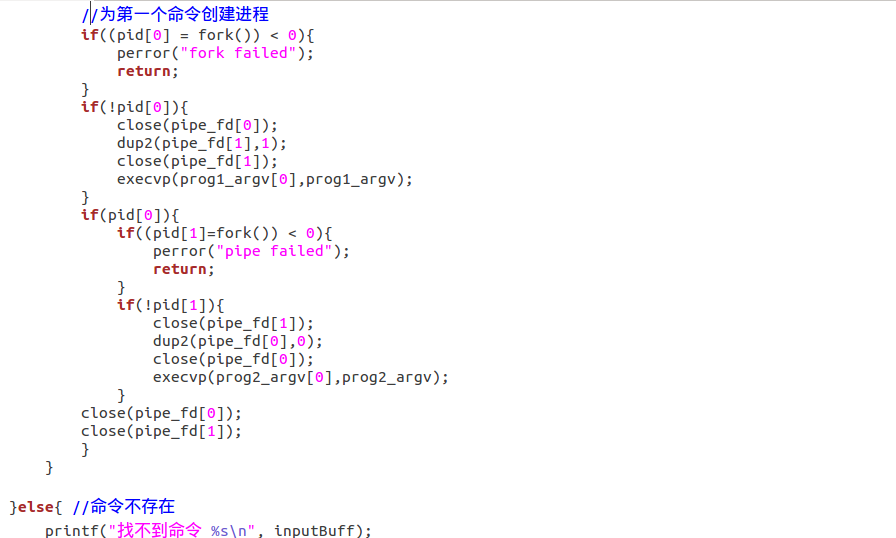


②同样按照重定向机制，管道需要为其变量赋值，在execut.c文件的命令解析中为变量赋值：



③在命令执行execOuterCmd函数中具体实现管道的功能：





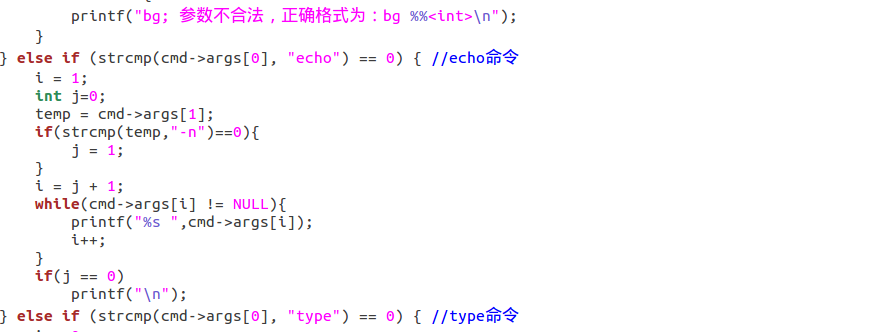
1. **对常用的内部命令进行实现**

**1.echo命令**

功能是将输入的字符串送往标准输出。输出的字符串间以空白字符隔开, 并在最后加上换行号。命令的一般格式为：echo [ -n ] 字符串，其中选项n表示输出文字后不换行。

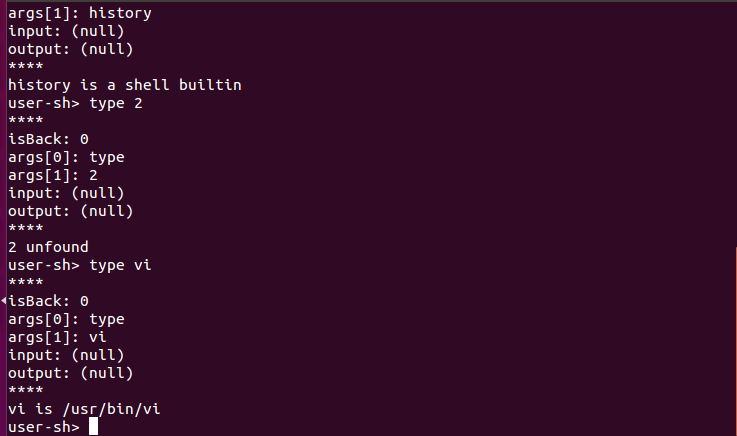


echo命令实现的方法：在execute.c文件执行命令中实现，

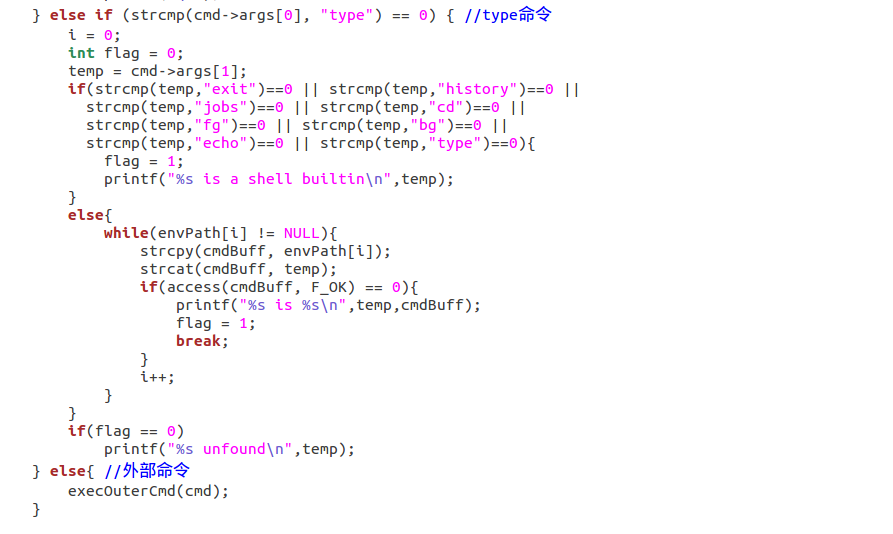


**2.type命令**

type命令被用于判断另外一个命令是否是内部命令。命令的一般格式为：type 参数。如果是内部命令，显示该命令为内部命令。如果是外部命令，给出绝对路径。否则显示命令未找到。

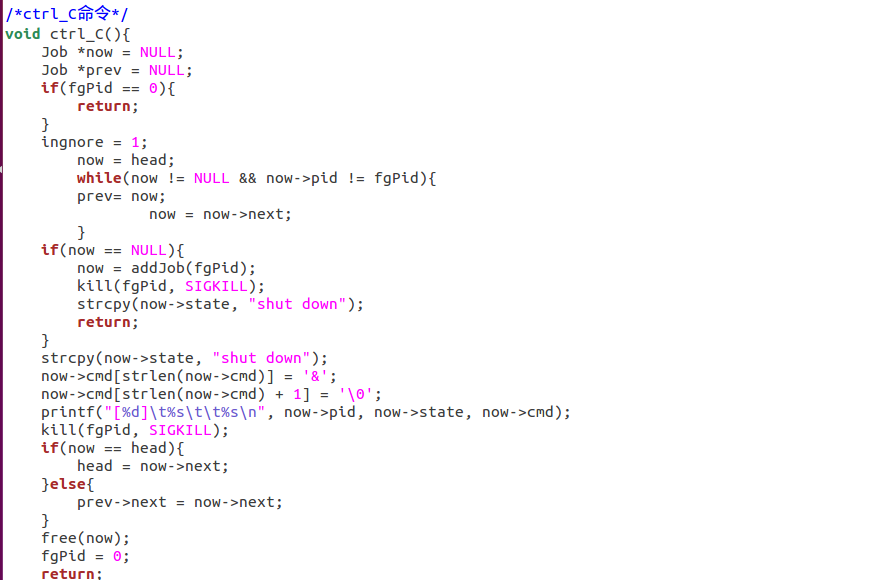


type命令实现的方法：在execute.c文件执行命令中实现，



1. **实现组合键ctrl+c命令**

实现的方式跟Ctrl+z类似：



# 收获和感想

因为操作系统实验从第一周就开始上，也就是说花了六周的时间弄了实验一，然而还是一脸迷茫。我花了两周的时间去弄怎么装实验环境，然而到头来还是把虚拟机装错了，然后再花两个周的时间去理解课件和弄附加实验，还是没弄完，再花了两周的时间去理解shell程序到底要做什么，学会如何使用github、理解源代码等等.....只能说身心疲惫，唯一有点安慰的是自己一步一步的边查资料边理解才明白到底要干嘛！！！然后有些实在是弄不明白的便请教下大神。但是现在不是有些地方还处于迷茫的状态，一些东西还用得不是很熟悉，开始是真的一点不知道要干嘛的，起码知道了思路了，慢慢磨蹭的一点点的把它给弄明白。

然而由于对YACC和Lex语法还不是很清楚且时间有限，所以对于提高要求的前俩个未能写出来。