**操作系统课程设计实验报告**

——实验一：shell实验

负责人姓名：崔博涵

学号：14061127

日期：2016.3.23

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 崔博涵 | 14061127 | 修改bug，编写实验报告 |
| 2 | 史天泽 | 14061121 | Yacc和lex学习并编写.lex文件 |
| 3 | 孟祥鑫 | 14061135 | Yacc和lex学习，加通配符 |
| 4 | 韩世依 | 14061129 | Ctrl c和管道实现 |

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)

5.发现的问题和解决方法……………………………………………………………5

# 1.实验目的

# 学习Linux相关软件工具的使用（如gcc、gdb和make）

# 熟悉使用Linux中YACC工具进行语法分析的基本方法

# 运用man帮助手册查询相关命令

# 理解并发程序的同步问题

# 学习POSIX/UNIX系统调用的使用

# 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

## 程序能够正常运行

## 能够执行*fg、bg、cd、history、exit*等内部命令

## 能够执行外部程序命令，命令可以带参数

## 使用I/O重定向

## 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

## 尝试对YACC语法分析的文法进行进一步的修改与完善

## 尝试在Linux下将Lex和YACC结合起来使用进行词法和语法分析

## 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

## 实现对管道的支持

## 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. 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. 可以使用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. 增加了ctrl+c的命令
7. 修改了第二次ppt上的3个bug。修改了kill函数传信号子进程的signal没来得及接收就被杀死的情况。修改了wait函数在子进程还没收到启动信号就已经执行的情况。修改了*ctrl+z*会导致后台作业停止运行的情况。

# 3.设计说明

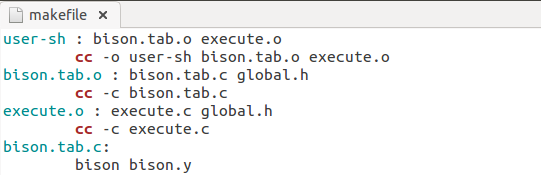
## 3.1 程序流程图

## 3.2基本要求实现说明

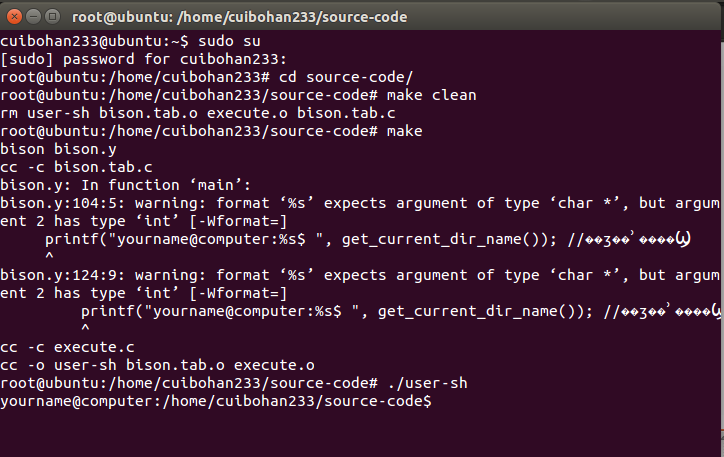
1、程序的正常运行

首先用打开文件夹，执行make clean命令，清空之前的user-sh bison.tab.o execute.o bison.tab.c文件。

然后执行make命令，从下到上，把截图中黑色的文件编译成蓝色的。

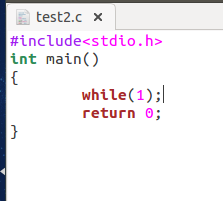
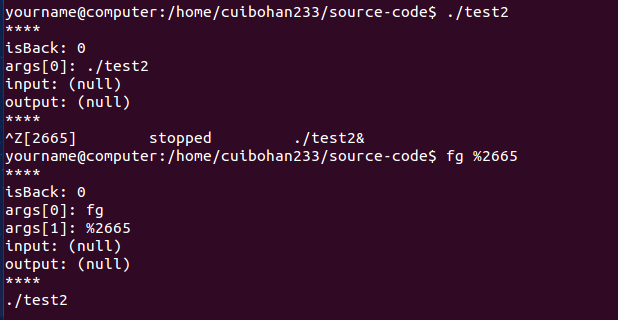


然后运行命令***./user-sh***即可运行该程序。

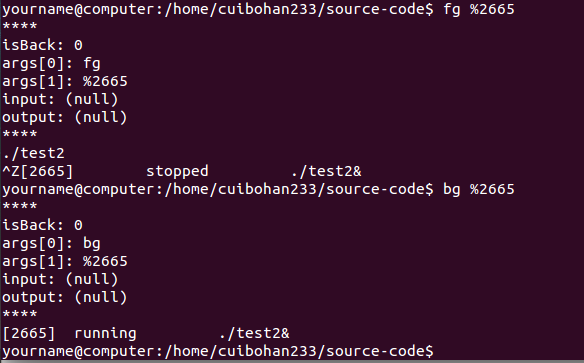


2、fg命令 bg命令

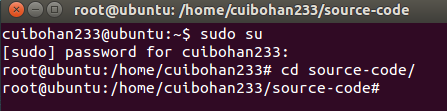
我先执行一个test2.c文件，然后ctrl z把它暂停了，再用fg %它的id把它弄到前台。

bg就是在暂停之后bg %它的id把它弄到后台。

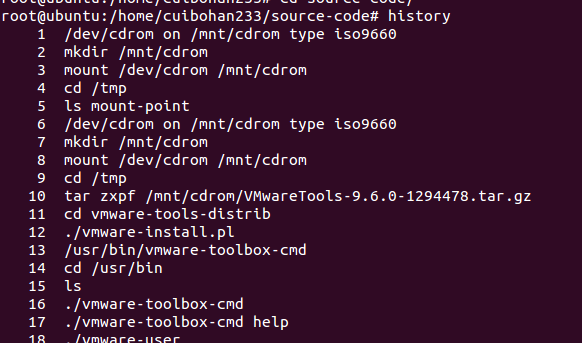


3、cd 是打开命令，cd 文件名就会打开那个文件



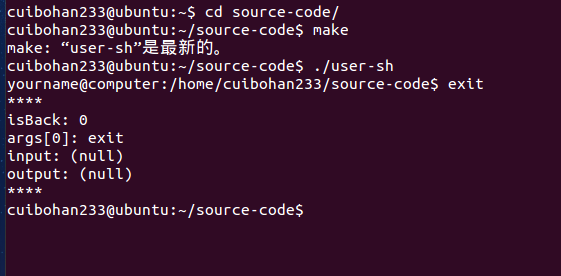
4、history

history是显示输入过什么命令

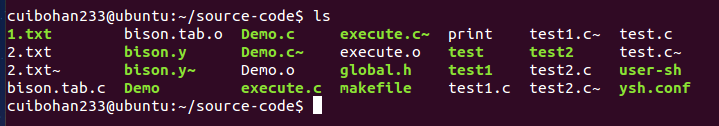


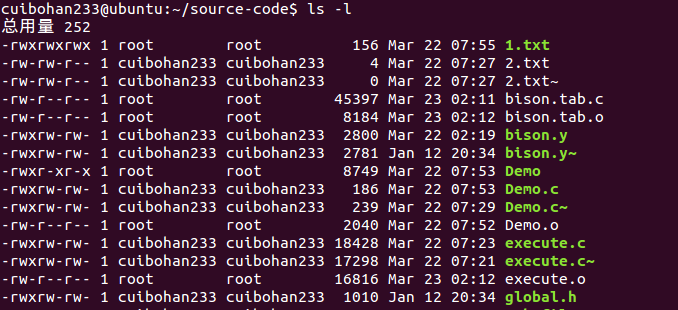
5、exit

exit命令用于退出当前shell，在shell脚本中可以终止当前脚本执行。



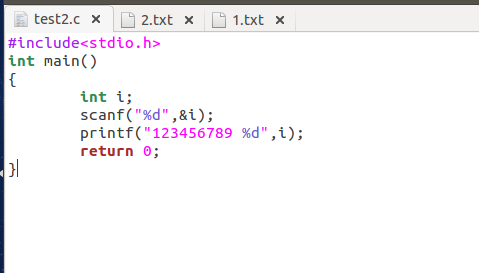
1. 执行外部程序命令ls 和ls –l

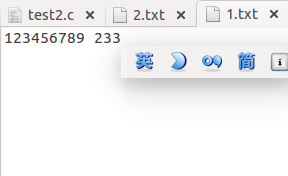




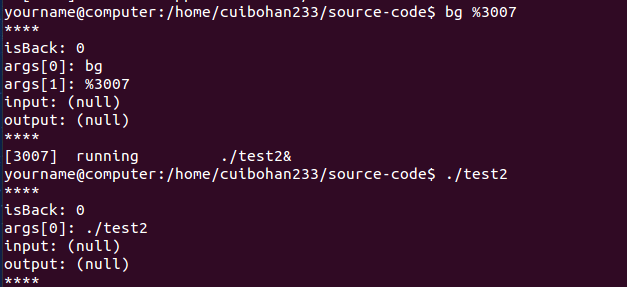
7、使用I/O重定向 输入命令./test2 < 2.txt > 1.txt

就是把2.txt里的内容当作输入，输出到1.txt里。

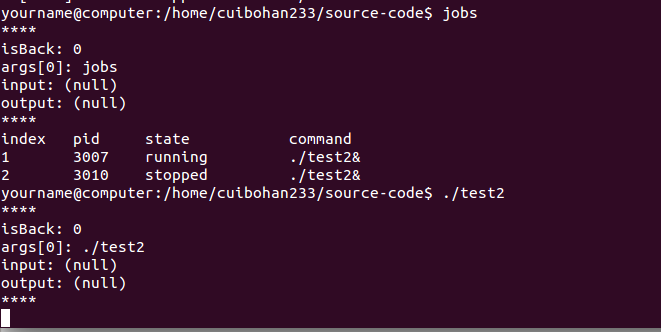




8、支持前后台作业，提供作业控制功能，包括打印作业的清单，改变当前运行作业的前台/后台状态，以及控制作业的挂起、中止和继续运行。



这些是对前后台作业的操作。组合键***ctrl+z***用来挂起一项正在前台运行的作业；***bg***命令可以将已挂起的命令发到后台运行；***fg***命令可以将被挂起的或在后台运行的作业放到前台来运行；组合键***ctrl+c***用来杀死一项正在前台运行的作业



## 3.3 提高要求实现说明

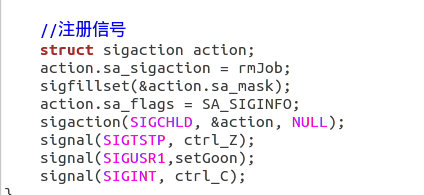
1、修改3个bug

第一个bug：a.在父进程执行完*kill(pid, SIGUSR1)*命令后子进程被杀死

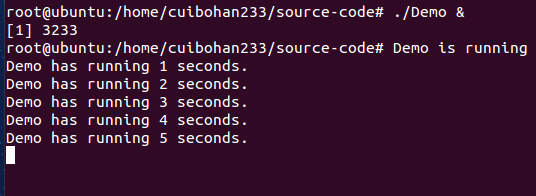
*b.SIGUSR1*、*SIGUSR2*等信号量默认会导致进程退出

c.保证子进程在父进程Kill函数执行前运行

修改方法：在init()函数中加个SIGUSR1的接收信号，这样子进程就能接收到SIGUSR1信号了。



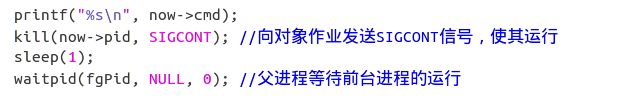
下面是正确结果：（后台的输出打印到前台了）



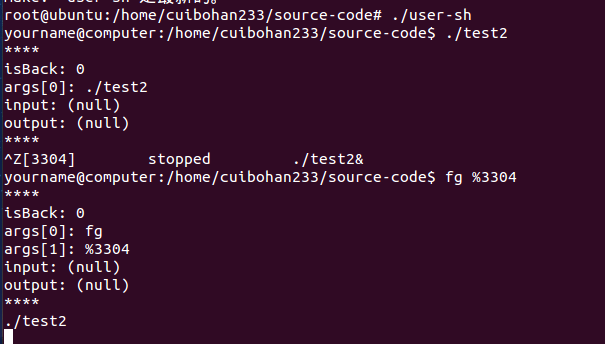
这个bug还有两种改法，1是把所有与SIGUSR1有关的信号注释掉，2是在kill之后加个sleep(1)，等子进程收到信号。

第二个bug：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

修改方法：在kill后加个sleep(1)，等子进程收到信号，这样紫禁城结束之后，waitpid就不返回-1了。

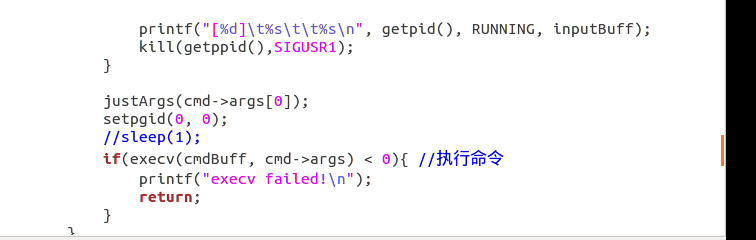


结果是：正常了



第三个bug：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。

解决方法：Ctrl z发送SIGSTOP信号是发给进程组的，整个shell的进程组号都是一样的，所以后台程序接收到SIGSTOP会停止。所以我把子进程的进程组号改为他的进程号，加个setpgid(0, 0)，这样就只停前台了。





2、加ctrl c

添加的代码，//是调试的时候用的，这个和ctrl z大致相同，只是传的信号不同，这个传SIGKILL杀死信号。

/\*组合键命令ctrl+C\*/

void ctrl\_C(){

Job \*now=NULL,\*before=NULL;

if(fgPid == 0){ //前台没有作业则直接返回

return;

}

ingnore = 1;

//printf("ctrl\_c\n");

//SIGCHLD信号产生自ctrl+C

//ingnore = 0;

now = head;

while(now != NULL && now->pid != fgPid)

{

before=now;

now = now->next;

}

if(now == NULL){ //作业不存在，则不进行处理直接返回

kill(fgPid, SIGKILL);

//printf("Terminated.\n");

return;

}

//开始移除该作业

if(now == head){

head = now->next;

}else{

before->next = now->next;

}

//修改前台作业的状态及相应的命令格式，并打印提示信息

printf("[%d]\t%s\t\t%s has been killed.\n", now->pid, now->state, now->cmd);

//free(now);

//printf("Terminated.\n");

kill(fgPid, SIGKILL);

fgPid = 0;

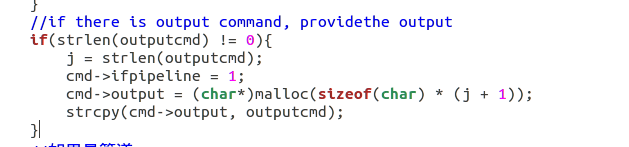
}

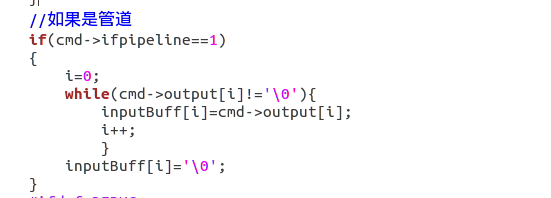
3.管道实现

先修改了命令解析，使管道作为一种命令格式。

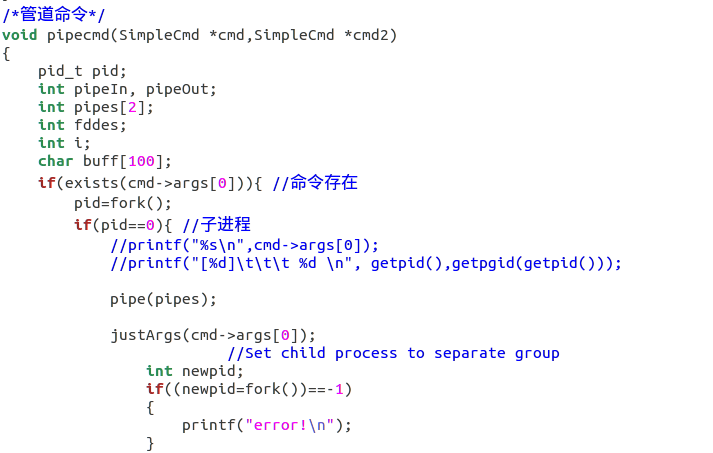


增加了一种处理管道模式的输出重定向的方法





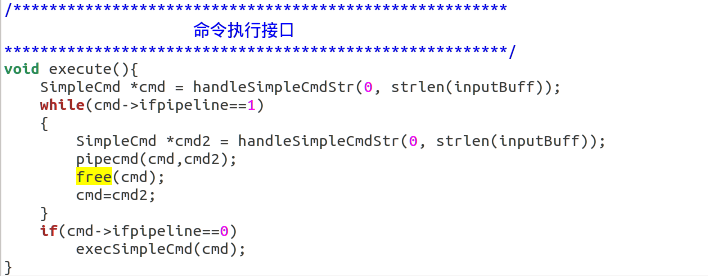
增加了管道命令的pipecmd函数



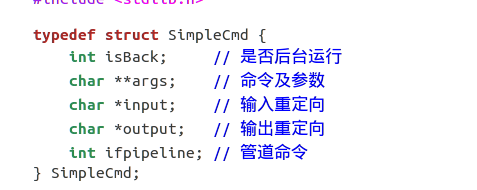




按书上的方法创建子进程完成管道命令。比如关闭读入，把管道写入和标准输出连起来，再关闭写入。最后执行命令，使用while循环，直到最后一个cmd。这样可以处理多个管道的情况。

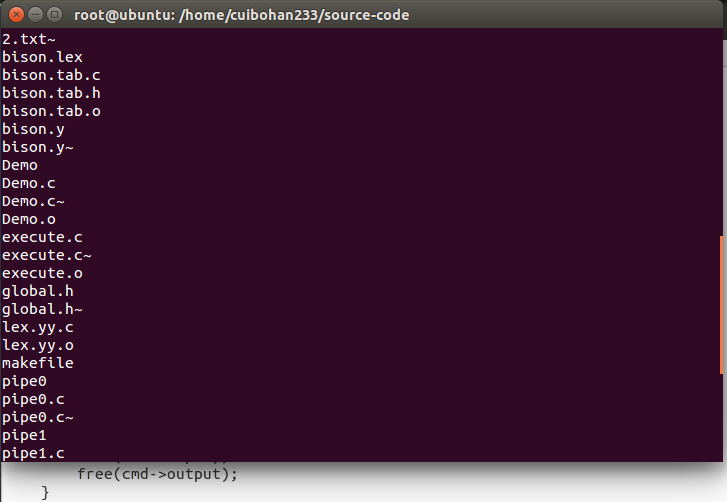


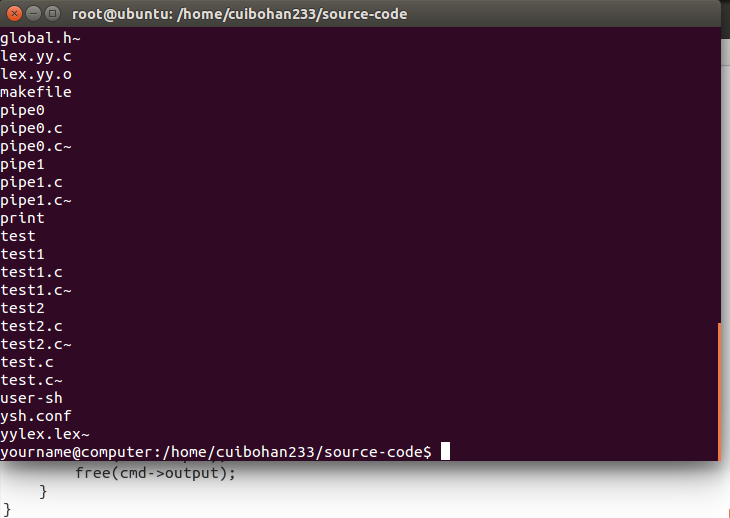
最后在global.h中加入ifpipeline这个属性。



测试样例：



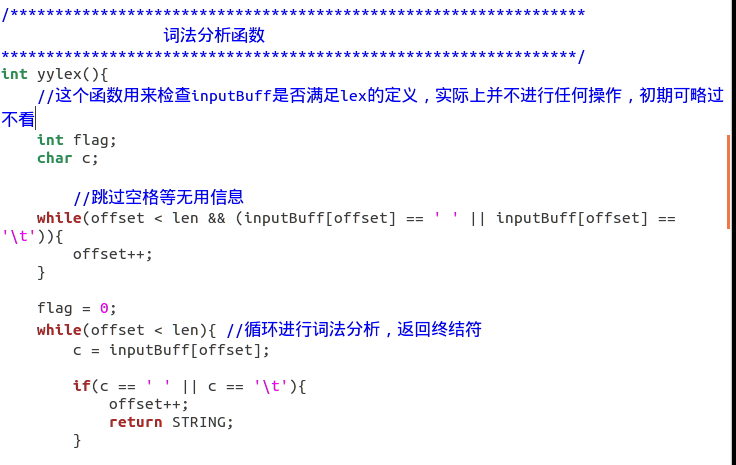


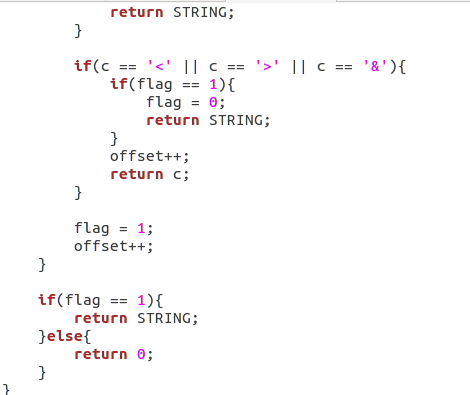


管道命令成功。

4、编写.lex文件

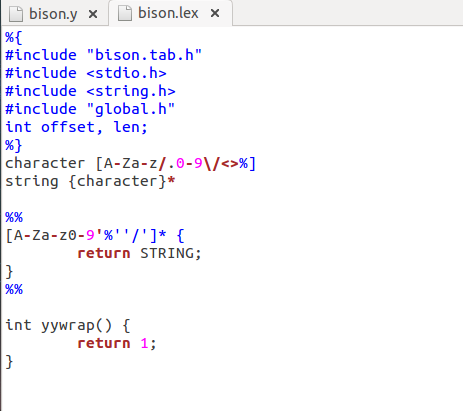
.lex的作用是把bison.y中的词法分析函数简化，用lex语法描述出来。



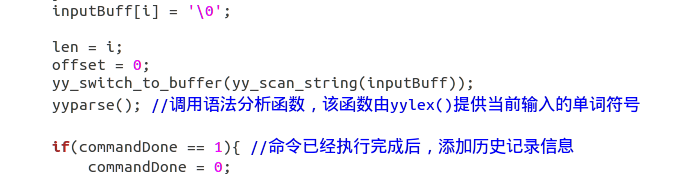


上面是词法分析的C语言版。

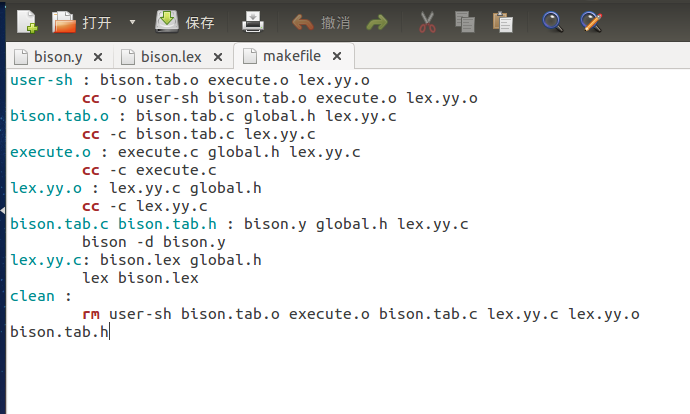
下面是我们写的.lex。



并且在bison.y中加入了一行代码，yy\_switch\_to\_buffer(yy\_scan\_string(inputBuff));表示从inputBuff中输入。



下面是我们的makefile。还是直接make一遍就可以了。



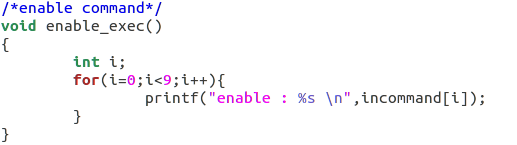
5、增加内部命令

echo、type与enable。

enable：显示可以使用的内部命令

首先定义内部命令的常量数组

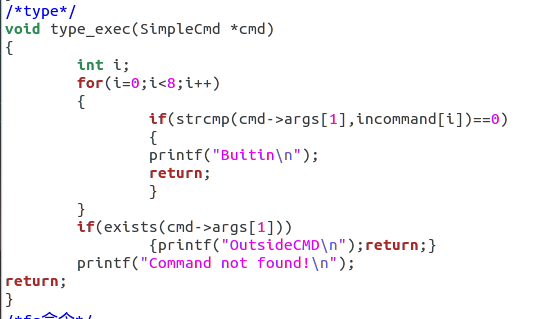






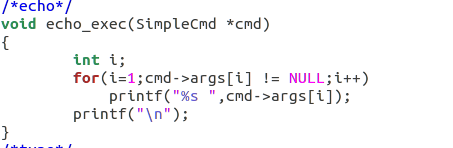
遇到enable只需要打印这些命令即可。

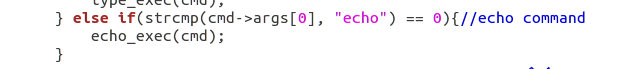
type：打印命令类型，内部命令外部命令和无效命令，比较好实现。





echo：输入什么打印什么

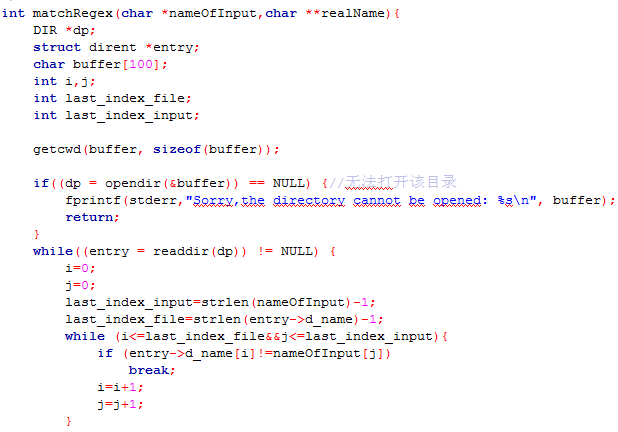


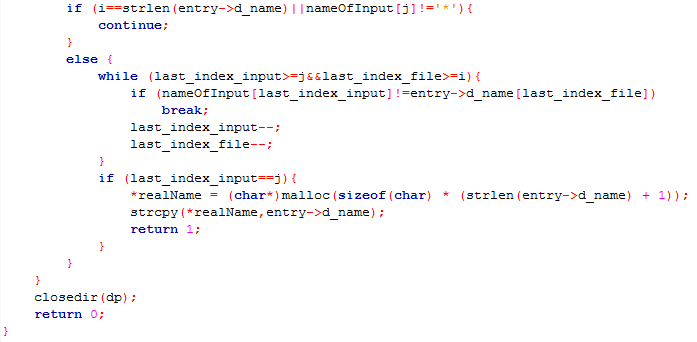


打印cmd中的内容就可以了。

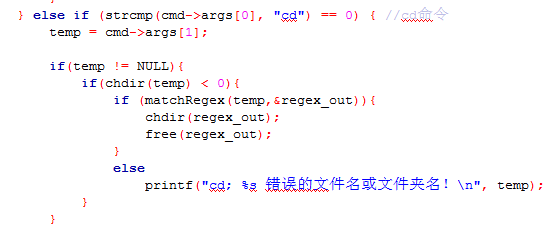
6、增加cd命令中“\*”通配符的实现

在execute.c中添加了一个新的函数match，用来实现“\*”的通配符实现（函数如下图一所示），调用头文件<unistd.h>中的getcwd函数，记录当前路径目录的信息，并给到变量buffer中保存，调用头文件<dirent.h>中的结构体dirent，然后使用opendir函数打开目录&buffer，返回指向目录的结构体，之后调用readdir(dp)函数读取&buffer下的所有文件（包括目录），返回指向目录&buffer下的所有文件的dirent结构体entry。通过对它的遍历，实现对当前目录下的所有文件以及目录信息进行存储。其中dirent结构体中的元素d\_name是指的当前遍历文件的文件名，由此成功提取到每个文件（或目录的）文件名。之后，通过前后匹配的方式，判断当前目录下是否存在与输入（带通配符）形式相同的目录名，如果识别到，则将该文件名存入realName指定的空间中；如果没有找到，则显示无法找到文件。（该函数在execSimpleCmd中调用，如图二，在文件名的比较部分添加新的if情况，如果实现了通配符的匹配，则达到也可以正常运行程序的功能）。





（图一）



（图二）

# 4.收获和感想

第一周刚接到这个ppt，比较茫然，不知道干什么，经过和同学的研究与讨论之后，渐渐明白了我们需要做什么。然后按书上的步骤，一步一步的学，不会的就查，终于对shell有了一定的了解，对linux系统也渐渐熟悉了。然后接到了第二个ppt。ppt讲了bison的语法，以及lex和yacc的结合使用还有3个bug，调bug费了好长时间，有时候是对的，下次再调就错了，可能还是理解不到位吧，但是都调完之后就明白了当时为什么是错的。总之还是要靠自学吧，本来一点都不懂的，学一学觉得还是不算太难的。

# 5.实验遇到的问题和解决办法

问题1：第一个bug，我一直在改变子进程signal的位置，想接收信号。结果一直不行，那天调到2点，电脑没电了还没调完。

解决办法：初始化的时候加上signal，程序还没开始就已经有signal接收的位置了。

管道问题2:close(pipes[0]);

close(STDOUT\_FILENO);

dup2(pipes[1],STDOUT\_FILENO);

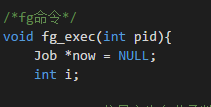
close(pipes[1]);

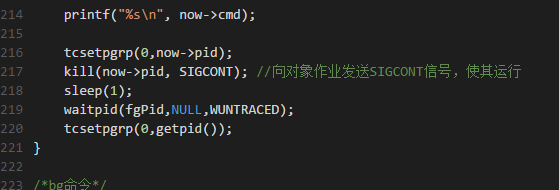
这段代码原来放的位置不对，致使每次建一个子进程又建了一个子进程的时候，这段代码执行了两遍，所以出现了问题。

问题2：在setpgid之后在执行子程序的时候无法scanf

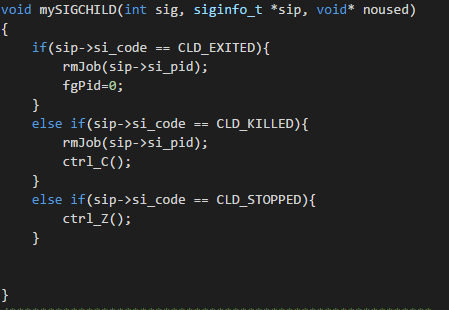
解决办法：使用tcsetpgrp函数，将前台的程序重新赋予前台的权限，在执行之后再还给user-sh，这样便可以使scanf生效。



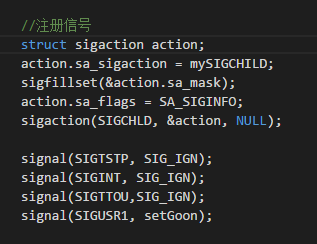




在fg的尾部加了这个，为了使其从后台用fg移到前台的时候也能够运行。



接下来的核心就是这个，将Init里面的信号结构体原来的函数指针从rmjob改为mysigchild，在这个函数里，我们会分辨信号，然后对应执行，否则就会出现ctrlc无法移除Jobs等bug。



然后再Init中将那三种信号屏蔽掉。

接着便大功告成。。

**六．第二组例会记录**

本次实验一共进行了3次例会

第一次例会：

主要进行了任务分配工作

时间定在了3月19日晚，由于白天分配了提高要求，热情的组员们集体要求晚上展开例会。

分配一下工作，并且互相讲一讲对实验的理解，在这次长达3个小时的例会中，我们对基本要求做了深刻的分析和探讨，然后分配了一下任务，崔博涵作为这次实验的主负责人，负责了bug的休整，最后大家的实验结果全部汇总在他的代码中，接着，组员们对所剩不多的任务进行了争取，大体是这样的，韩世依负责ctrlc和管道的实现，史天泽和孟祥鑫负责lex和yacc的优化和编写，并争取实现通配符。

第二次例会：

时间定在了3月22日晚，经过这两天，包括21日晚崔博涵同学和韩世依同学的熬夜改Bug，终于完成了主要框架，完成了ctrlc和基本的Bug修改，但此时，还有scanf的bug和管道等待着我们，在大家进行了近几天的任务回报之后，我们继续进行了接下来的任务分配。

第三次例会：

时间定在了3月25日晚，完成了所有的功能，通配符在最后时刻也成功实现。