**操作系统课程设计实验报告**

——实验一：shell实验

负责人姓名：詹子豪

学号：14061137

日期：2016.3.26

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 詹子豪 | 14061137 | 实验一 |
| 2 | 于福洋 | 14061134 | 实验二 |
| 3 | 于建勋 | 14061125 | 实验三 |
| 4 | 石浩然 | 14061122 | 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，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

本实验要求在linux环境中使用C语言编写一个简单的SHELL命令解释器程序，我们称之为user-sh。其设计类似于目前流行的shell解释程序，如bash、csh、tcsh。User-sh应当具有以下一些重要特征：

1. 能够执行fg、bg、cd、history、exit等内部命令。
2. 能够执行外部程序命令，命令可以带参数。
3. 使用I/O重定向和管道
4. 支持前后台作业，提供作业控制功能，包括打印作业清单，改变当前运行作业的前台/后台状态，在整个实验中还必须做到以下几点：
5. .使用make工具建立工程。
6. 使用gdb或者ddd等调试工具来调试程序。
7. 提供清晰，详细的设计文档和解决放案
8. 锻炼团队成员之间的协作开发能力

## 2.2 提高要求

1. 尝试对YACC的语法和文法进行进一部的修改和完善
2. 尝试在linux下将LEX和YACC结合起来使用进行词法和语法的分析
3. 对其他常用内部命令进行实现，并可以尝试考虑对通配符的实现与支持
4. 实现对管道的支持
5. 参照组合建ctrl+Z命令的实现方法，考虑并实现ctrl+C命令
6. 其他自行提出的改进

## 2.3 完成情况

首先，我们通读了源代码，了解的程序运行的基本过程。然后，我们在自己的电脑上进行了make，发现本实验提供的源代码基本实现了基本要求的部分，但是仍然存在一些小bug，这部分会第4部分中详细解释。然后我们就进行了分工，来实现提高要求的部分。最后，我们将各自负责的部分进行了讨论，然后将改动都整合到一起。

完成了以下功能：

1. 添加了 echo、mkdir、pwd这三个内部命令

2.尝试对YACC的语法和文法进行进一部的修改和完善

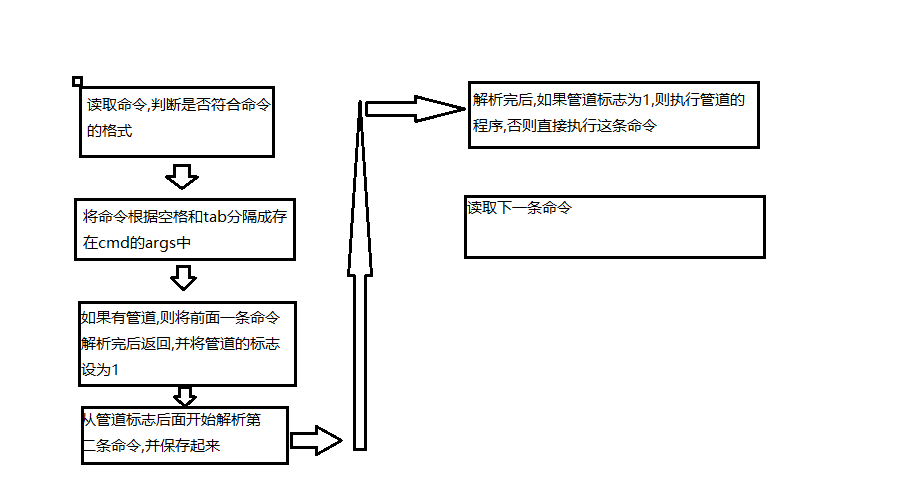
3.尝试在linux下将LEX和YACC结合起来使用进行词法和语法的分析

4.实现了对管道的支持

5.实现了ctrl+C

**3.设计说明**

## 3.1 程序流程图

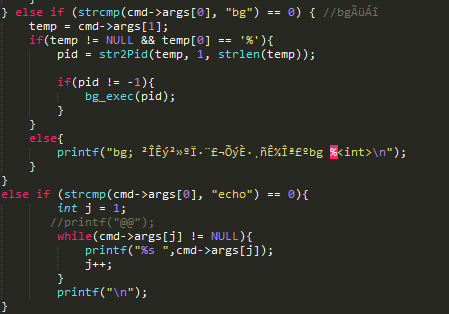


## 3.2基本要求实现说明

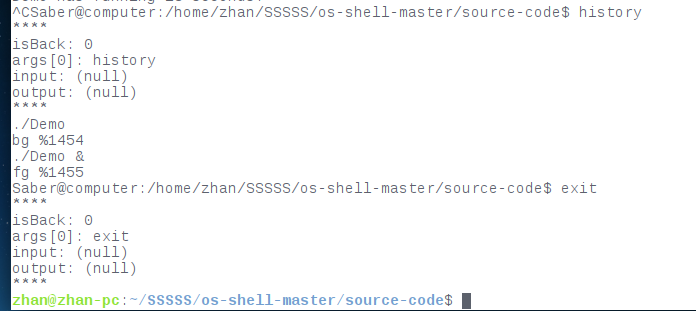
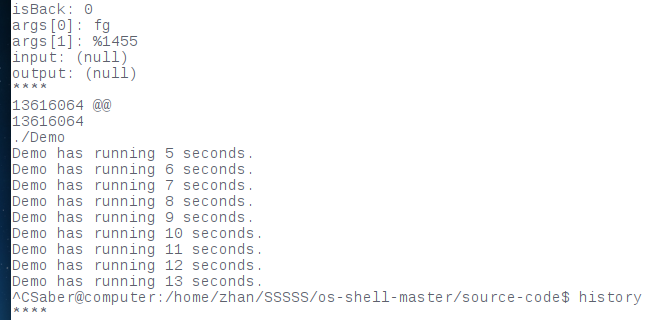
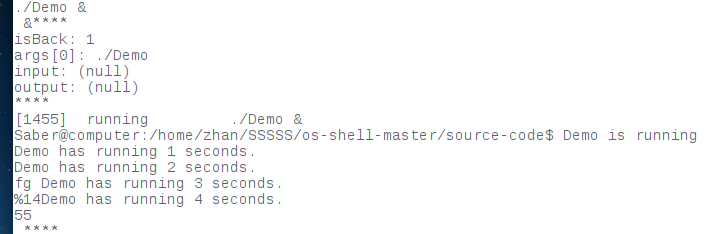
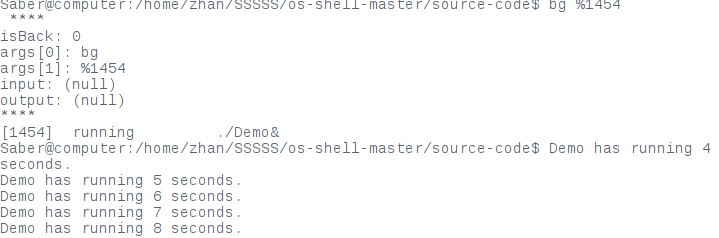
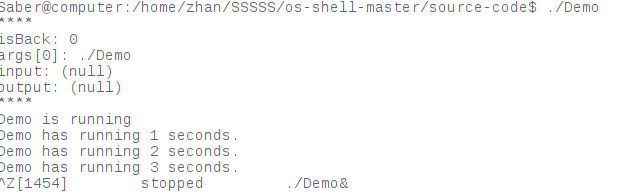
1. 能够执行fg、bg、cd、history、exit等内部命令。

代码:

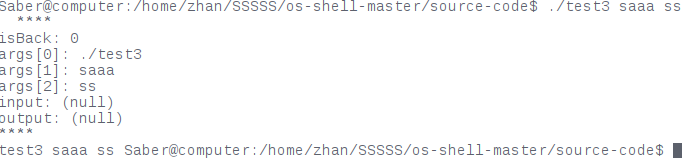


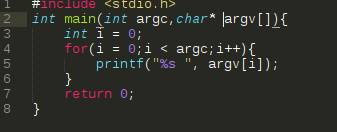


运行截图:



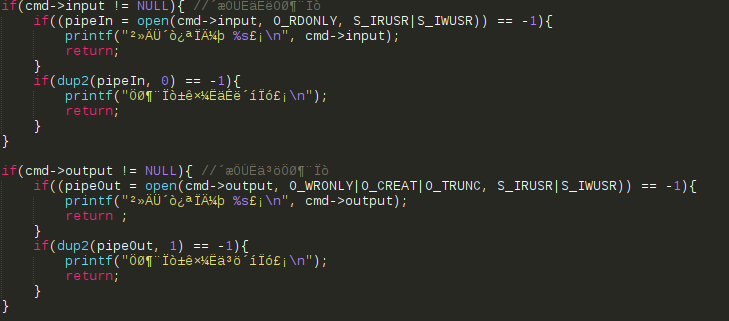
1. 能够执行外部程序命令，命令可以带参数。



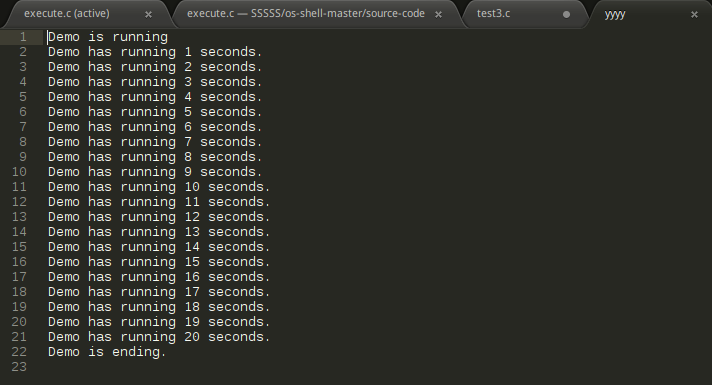
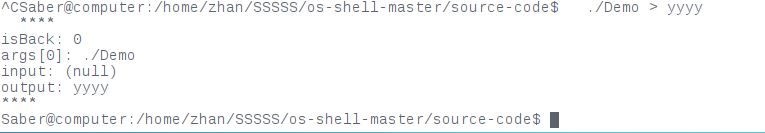
测试程序：

1. 使用I/O重定向和管道

代码：



运行结果：



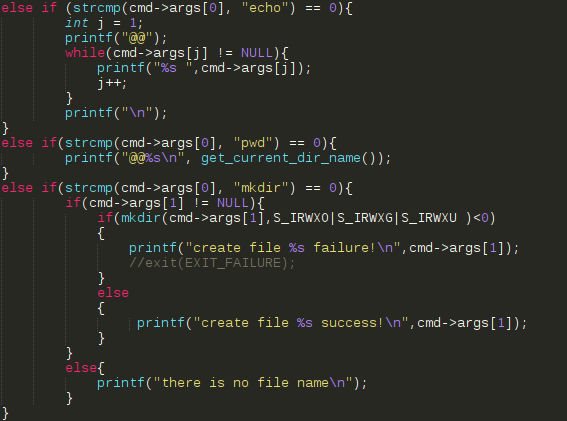
1. 支持前后台作业，提供作业控制功能，包括打印作业清单，改变当前运行作业的前台/后台状态

运行结果见1的截图。

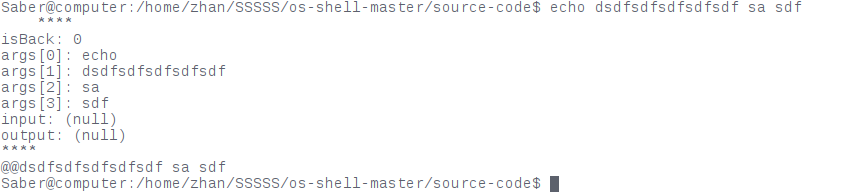
## 3.3 提高要求实现说明

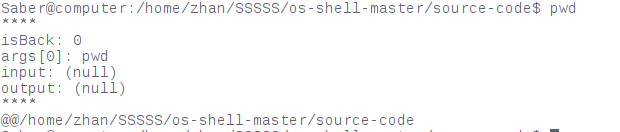
1. 添加了 echo、mkdir、pwd这三个内部命令

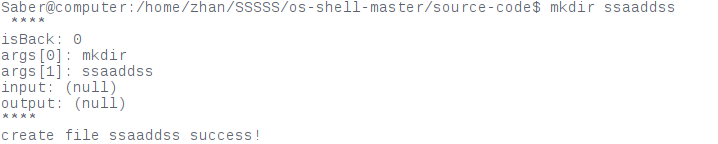
代码截图：

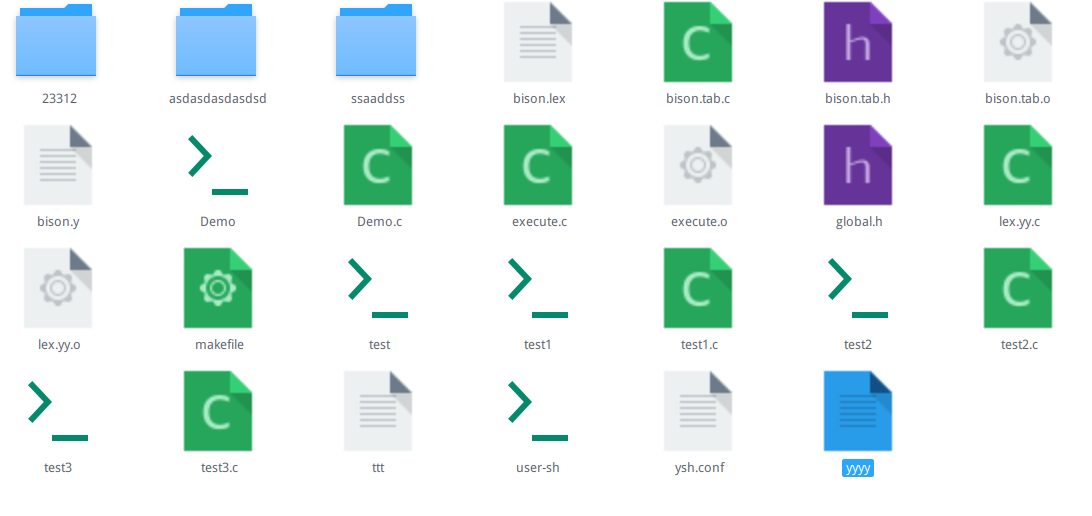


运行结果：



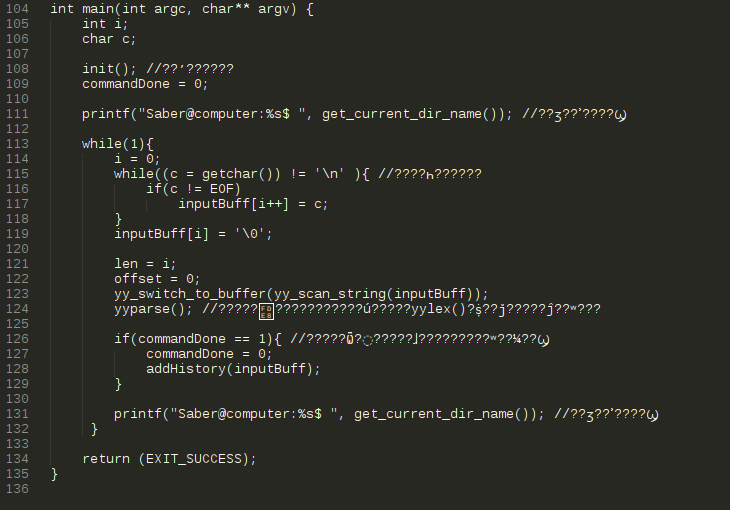
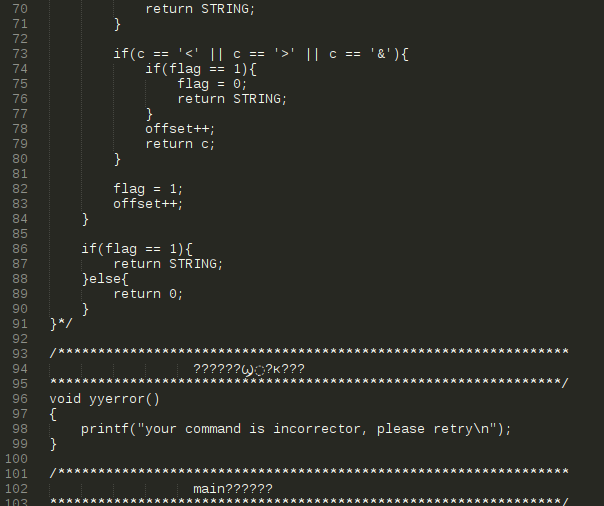
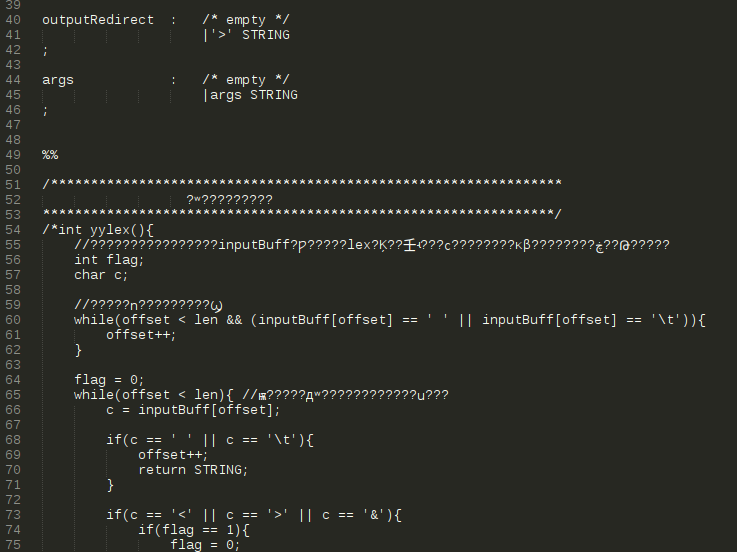
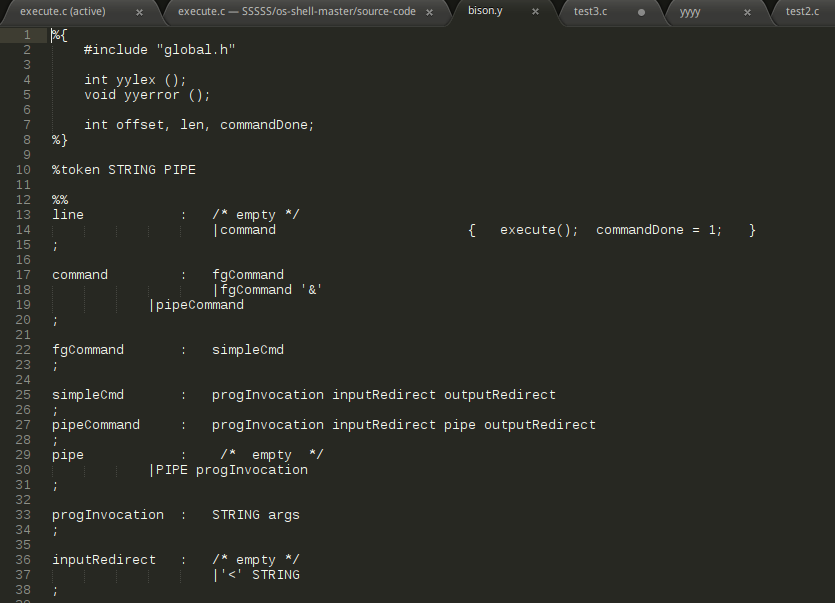






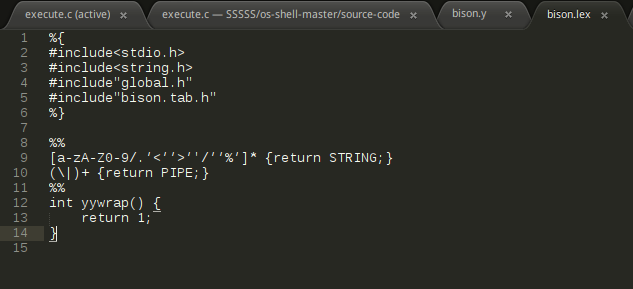
1. 尝试对YACC的语法和文法进行进一部的修改和完善

改后的bison.y：

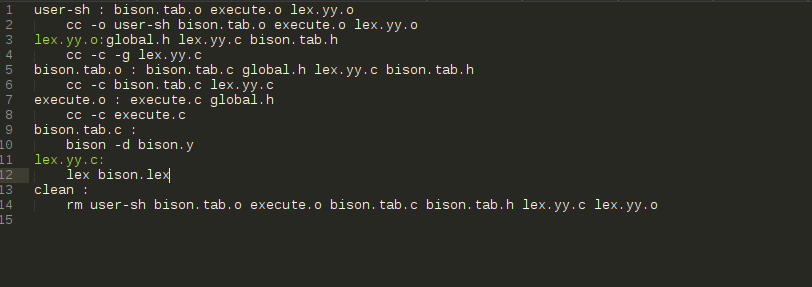


1. 尝试在linux下将LEX和YACC结合起来使用进行词法和语法的分析

新加的bison.lex:

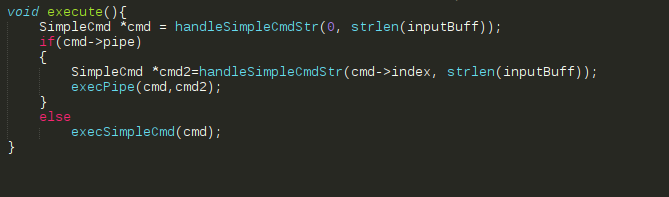
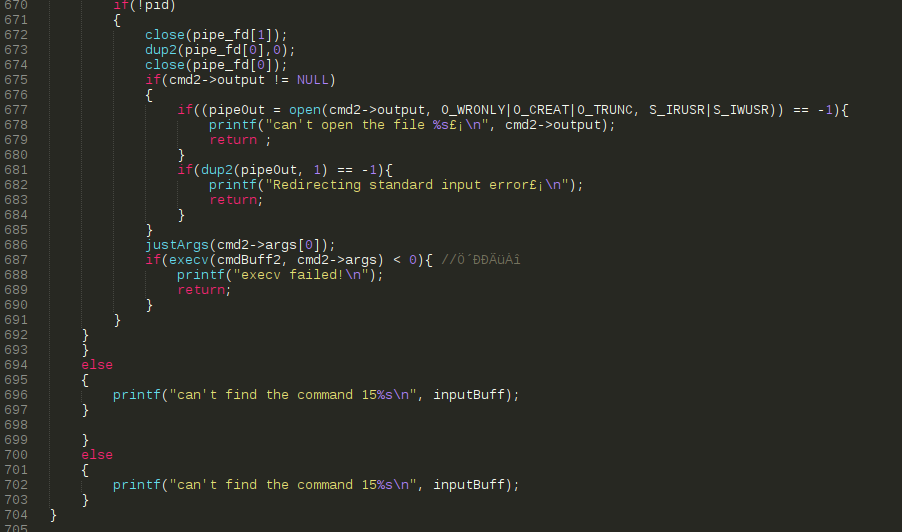
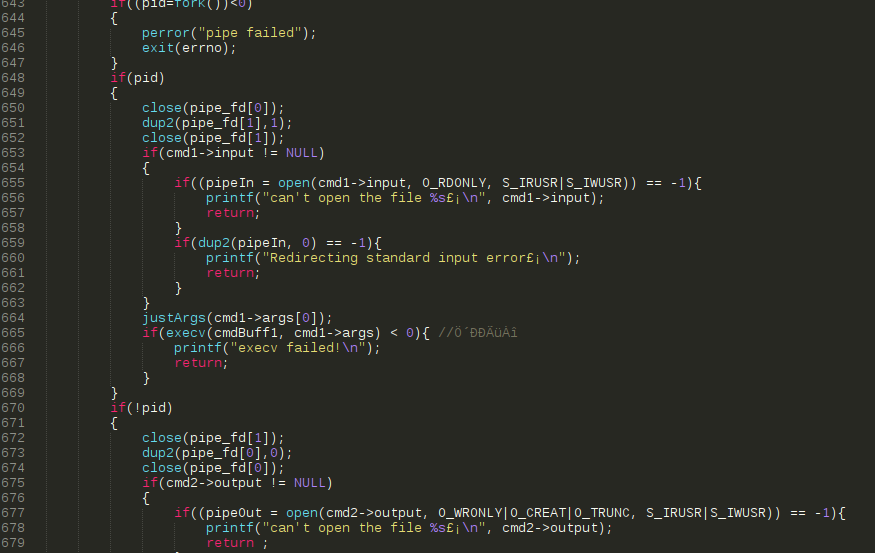
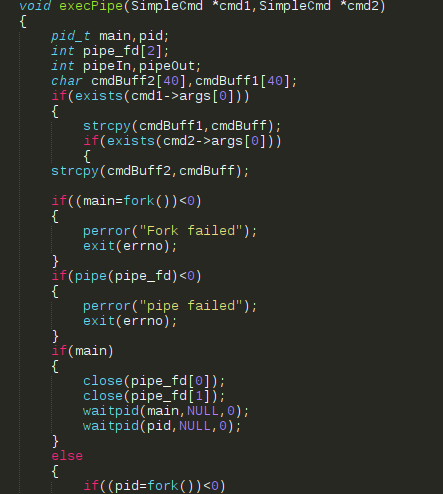
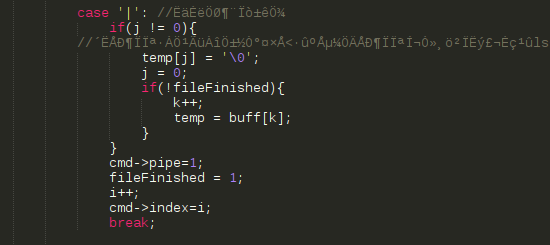


改后的makefile：

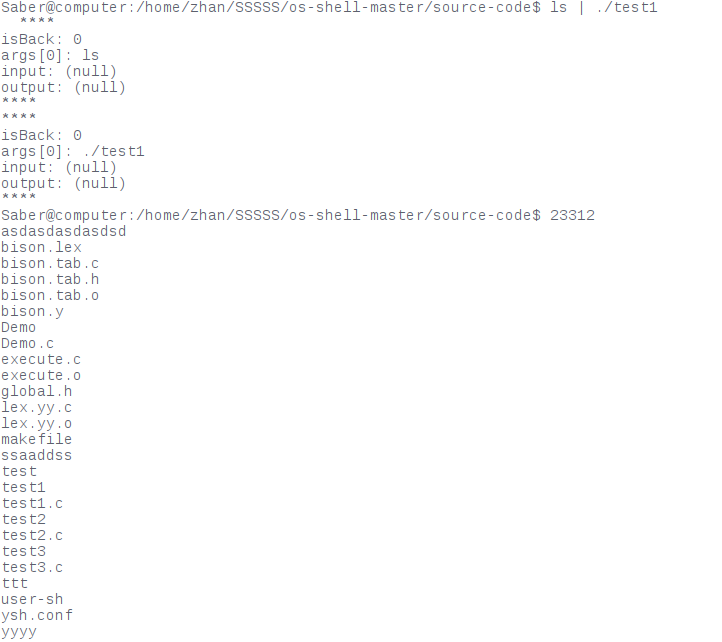


1. 实现对管道的支持

代码：

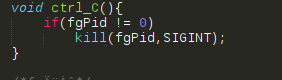


运行结果：

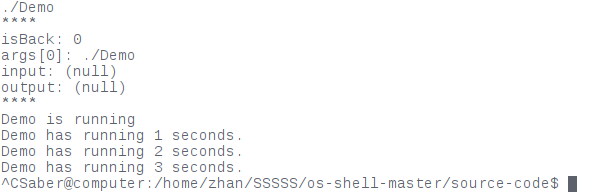


1. 参照组合建ctrl+Z命令的实现方法，考虑并实现ctrl+C命令

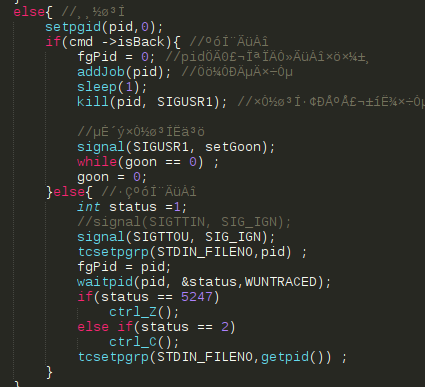
代码：



运行结果：



1. **遇到的问题和解决办法**
2. 直接后台运行外部命令是会陷入死循环。原因是SIGUSR1 、 SIGUSR2等信号量默认会导致进程退出，所以子进程还未开始就退出了，所以让父进程在kill前sleep（1）,就可以让子进程在父进程kill前先运行，子进程就可以收到信号，不会退出了。
3. 运行前台进程，然后ctrl挂起，然后fg，命令提示符就会先打印出来，原因是父进程在子进程结束之前就运行了。因为在fg中先给子进程发信号让他启动后，waitpid（）的时候子进程还没有处理信号，所以在waitpid之前让父进程sleep（1），子进程处理信号开始运行，这样waitpid就能正常的运行。
4. Ctrl+z会导致后台进程停止运行。因为fork（）后，子进程和父进程在同一个进程组里面，所以子进程会接收到ctrl+z然后停止运行。所以我们在fork（）之后，子进程运行时使用setpgid（0,0）将它分到另一个进程组里面。但是这样子进程就会没有终端的控制权，如果使用一个需要输入的命令，那么子进程就会直接退出。所以我们在运行一个前台子进程时，就使用tcsetpgrp将终端控制权给子进程，再调用waitpid让子进程结束后，把控制权给父进程。但是这样还是有问题，控制权给子进程之后，父进程变成后台的进程，接受不到ctrl+z和ctrl+c，就会让系统的shell接收到，然后将user-sh退出，所以我们就根据waitpid的status的值来判断子进程是如何结束的，然后根据返回值自己调用ctrl\_Z和ctrl\_C函数。这样就大致解决了这些问题。



1. 运用lex和yacc共同进行语法和词法分析时输入的字符被读入buffer，不会被yylex捕捉到。后来百度了之后加入yy\_switch\_to\_buffer(yy\_scan\_string(inputBuff));这个函数之后就能将buff里面的值给yylex（）；

# 5.收获和感想

通过这次试验，确实学到了不少东西。现在我的代码已经提交了，坐在这里慢慢总结起来，觉得主要有两个方面的收获。

一是万事开头难，可是只要投入进去了，最后会发现许多事情并不如想象的那么困难。正如shell的试验从开始的无从下手到最后也算是圆满完成的结果，这个一点一滴的积累资料的过程，提高能力的过程我把它当成难得的财富。以后遇到一个完全未知的领域，怎么去从无到有，从不知到了解到解决问题，也会有自己的经验和方法。我想一门实践课决不是为了教会你怎么做一个试验，而应该让人能从中学会一套解决实际问题的方法和方式。

二是对linux这样一个以前没有接触过的系统有了一些基本的理解。学会了在linux下的一些基本操作，对一些系统调用有了一定的理解，比如在这次试验中使用的execvp，dup，pipe，signal等。这样一是对linux环境下的C编程有所熟悉；另一方面对操作系统的运行机制有了结合实践的体会。毕竟课堂上讲的多是理论，现在自己动手编写出自己的程序，感觉当然是很不一样的。