**操作系统课程设计实验报告**

——实验一：shell实验

负责人姓名：张巍

学号：14061186

日期：2016.3.20

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 张巍 | 14061186 | 实验一/三 |
| 2 | 冯炜韬 | 14061197 | 实验二/四 |

目录

[1.实验目的 4](#_Toc446583584)

[2.需求说明 4](#_Toc446583585)

[2.1基本要求 4](#_Toc446583586)

[2.2 提高要求 4](#_Toc446583587)

[2.3 完成情况 4](#_Toc446583588)

[3.设计说明 6](#_Toc446583589)

[3.1 程序流程图 6](#_Toc446583590)

[3.2基本要求实现说明 7](#_Toc446583591)

[3.3 提高要求实现说明 10](#_Toc446583592)

[4.收获和感想 11](#_Toc446583593)

[5.实验遇到的问题及解决方法 12](#_Toc446583594)

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中的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）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后台状态以及控制作业的挂起、终止和继续运行。

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）能够执行fg、bg、cd、history、exit、echo、type、jobs、wait等内部命令且支持管道并能够正确完成输入/输出流的重定向。

3）能够执行外部程序命令，命令可以带参数。

4）能够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后台状态以及控制作业的挂起、终止和继续运行。

6）使用YACC和Lex进行词法和语法分析。

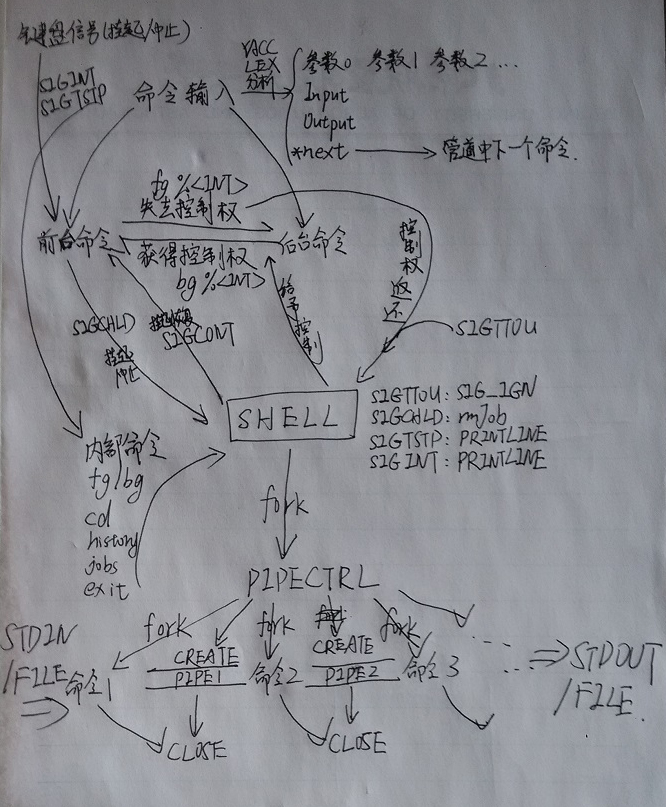
7）支持管道。

8）支持通配符”\*”

9）支持组合键ctrl+c命令。

3.设计说明

3.1 程序流程图



3.2基本要求实现说明

1）fg,bg命令：指令格式为 fg %<int> 以及bg %<int>，根据pid 查找索要更改运行状态的程序，当用户输入错误时会提示正确的指令格式，未找到目标程序时会返回作业不存在。另，在进程的运行状态在前后台切换时，会正确地将控制台交给前台运行的程序。特别地，后台进程无法接受挂起以及终止信号，后台进程组无法控制台进行输入；前台相反．

源代码已经实现了部分功能，这里新做的操作有：将子进程独立成组，前台进程组分发控制权，后台进程组忽略；切换时交接控制权．

2）history命令：将最近由键盘输入的若干条（History\_LEN==10）指令记录下来，在接受到history指令时将其全部输出。

此功能较为简单，不赘述． 

3)cd命令：将读取到的路径与已有路径比对，若存在则移至目标路径，若路径输入错误则返回“错误的文件名或文件夹名”。

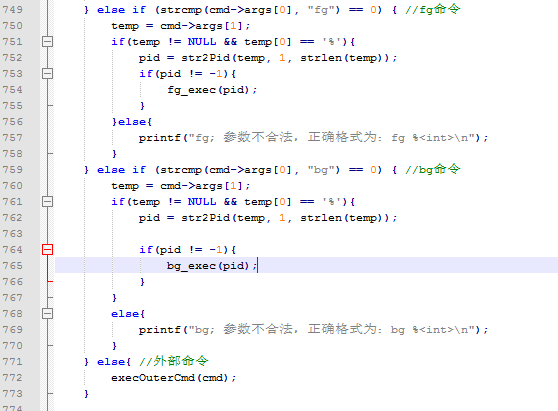
源码已经实现此功能．

4)exit命令：关闭所有的输入输出流以及进行中的进程然后关闭shell。

源码已实现．

5)jobs：若作业队列为空则返回“尚无任何作业”，否则按行输出“作业序号（新作业在前）+作业Pid+作业运行状态+作业名。

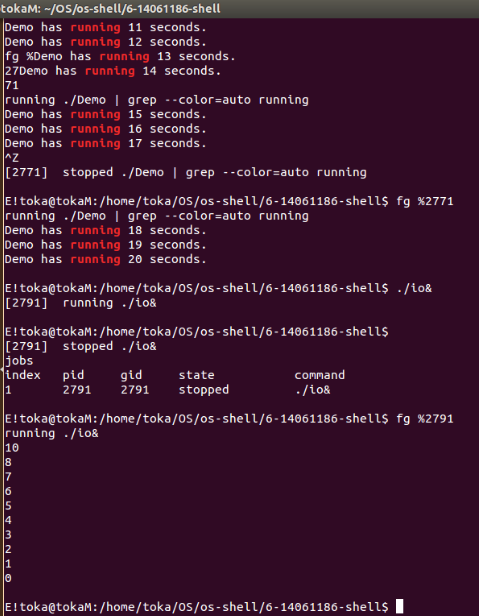
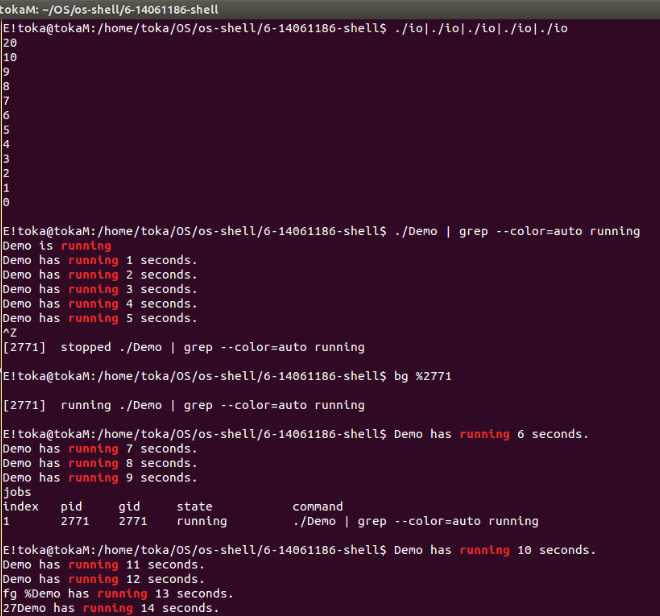
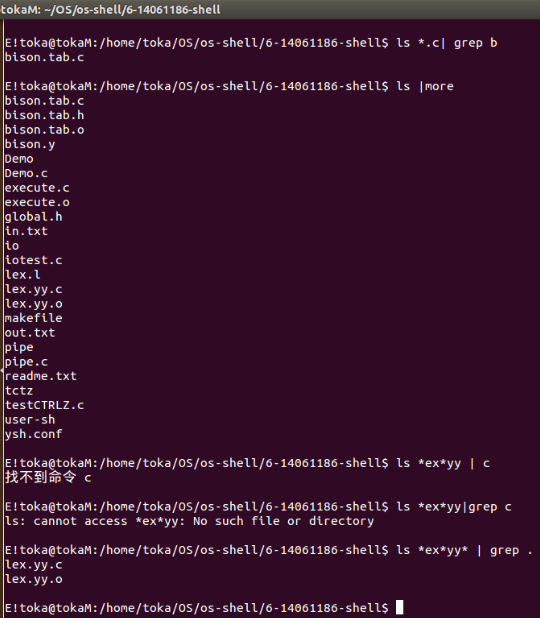
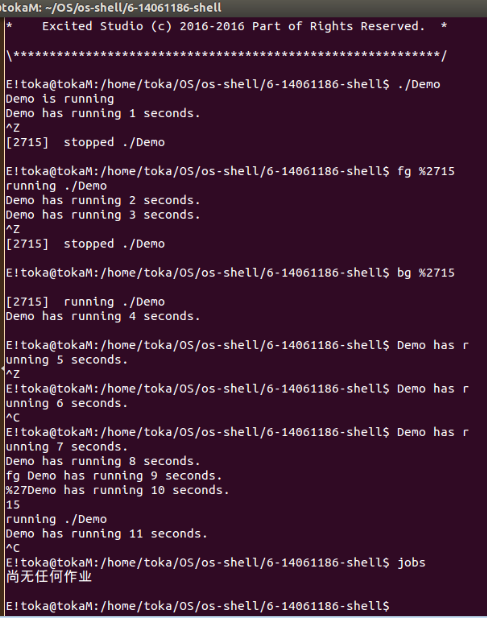
源码实现部分，新增查看进程组，修复一些情况下进程退出作业队列没删除或者进程挂起作业队列状态未正确改变的问题．



6）外部命令：在该外部命令存在的情况下，新建作业加入作业链表并新建进程及进程组执行命令，若为后台命令，则执行后马上继续等待控制台命令输入；若为前台命令则转移输入输出控制权并等待进程完成，执行完毕后等待下一个命令。



运行结果检验如下: (包括提高要求)



3.3 提高要求实现说明

1）Pipe：在命令行输入时以'|'表示管道命令，在判断管道两端均有相应的输入/输出流且定向唯一的情况下，将输入/输出流重定向并将管道相应端口与对应输入/输出流相连，关闭管道时按顺序关闭，应保证关闭全部的字符流。

管道命令中所有外部程序同时执行，设置重定向后，程序自动运作，当所有进程完成退出，控制程序关闭所有管道，结束管道命令．

为了支持管道命令的挂起以及恢复，解析到管道命令时首先新建一个控制子进程，子进程代码在shell内部，这个控制的专用进程再执行命令中的外部程序，并监察这些外部程序的运行状态．所有管道进程以及控制进程均在同一个进程组．当收到挂起信号，整个进程组挂起，恢复命令仅对控制进程进行，控制进程收到恢复信号，再检查外部程序的运行状态，未执行完的重新恢复执行．

2）组合键ctrl+z 和ctrl+c：键盘捕捉到信号时返回中止进程的信号，前台进程组收到信号，挂起时会发送信号给父进程，完成控制权的交接以及其他杂项。父进程设置了信号处理，因此不会被信号所挂起或终止．

3）通配符：这里只实现了\*（代表任意字符而非单个字符），且通配符可以配合管道一起使用，主要用来查找未知完整文件名的文件。在一个正常参数（第二个参数起），中加入＊号，表示该处可能填充任意长度的普通字符，即非操作符．匹配结果为当前目录中所有符合要求的文件或文件夹，以参数扩展的形式传给进程．特别地，使用通配符检索隐藏文件（.开头）必须键入[ . ]，若第一个符号不是点，将不在隐藏文件中检索；再者，对于cd命令，通配符仅返回文件夹匹配结果．

4）YACC及LEX语法分析：根据YACC的归约语法，以及lex的定义，可以构造命令解释模块，完全替换源码中的原有的输入以及解释部分．除了几个操作字符外其他字符的连续串定义为STRING，然后使用这个原子定义相继定义命令中的其他部分，最终定义整个命令．每匹配到一个部分，同时进行解析任务，即匹配到整个命令时，自动生成了对应的命令数据结构．这个获得的数据结构，相当于源码中解释部分得出的结果，可以直接被执行部分的代码利用．

5）echo：将echo后紧随的参数依次输出，这里忽略了引号对于命令行输入结尾判断的影响。

wait：shell等待直到当前shell下没有正在进行中的进程（即不包括已完成的作业以及被中断的作业）

type：可以正确区分目标指令是否为shell的内部命令。

4.收获和感想

整个任务持续时间约⑨天，当然由于组员各自的原因，有相当几天去做别的事情．但是总体感觉还是有一定难度和工作量的，尤其在很多新知识需要自学的情况下．除了新知识，最麻烦的就是细节部分的调试和用户体验的优化，即如何优雅地实现功能．整个流程下来，我们可以窥视到计算机系统的一部分运行规则，离计算的核心又更靠近了一步．虽然有些瑕疵，但是还是得到了重新发明轮子的爽快．

5.实验遇到的问题及解决方法

1. 后台进程可以被挂起:用setpgid将进程放到新的进程组;然后出现新的问题,后台进程回到前台时需要输入:用tcsetpgrp设置后台进程组控制输入,即转移到前台,父进程退居后台,注意,此时父进程监察到前台子进程结束需要夺回控制权,并监听且忽略SIGTTOU信号.

2. 实现管道时，遇到某些读到EOF的进程无法正常结束：这是由于父进程在fork之前就建好了pipe，相当于父进程也有权限向读到EOF的子进程输入，即使该进程的上一级管道已经关闭，父进程仍在等待，解决方法是父进程在fork完之后把相应的管道关闭，只有当所有管道输入关闭时，进程才能正确返回。

3. YACC和lex语法分析：可能遇到的问题是当一个后台进程关闭时标准输入莫名地多出来一个EOF，这里的解决方法比较粗暴，检测一下当前lex的输入是否控制台，如果是的话，直接把EOF替换成空格，实现这一个需要在lex.yy.c中找到yyinput这个宏定义并放在点l文件中重写。另一个问题是对于任意一个完整的定义，由于非终结符不会将一行都匹配完整，假如一行的前部是乱码，后部是正确输入，后部仍会执行非终结符的指令，当然整个指令是不符合顶层定义的，这时候就会出现一个yyerror，同时又会对后部分的正确指令执行，这是一个很不友善的行为。这里的处理方法是遇到yyerror就直接用一个标记标记这一行都不受处理，知道读到回车才消去标记。当然这个做法在某些极端情况还是有例外，但是相对好一些了。