**操作系统课程设计实验报告**

——实验一：shell实验

负责人姓名：金银珠

学号：14231028

日期：2016.4.7

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 金银珠 | 14231028 | yacc&lex |
| 2 | 柴瑞 | 14061182 | yacc&lex、测试 |
| 3 | 刘朋涛 | 14061219 | 内部指令、Ctrl+C、bug修改、管道 |
| 4 | 熊藤攀 | 14061220 | 实验报告、会议记录、管道 |

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.完成基本要求，程序能够正常运行、支持fg/bg等内部指令、支持带参数的外部命令、支持IO重定向、支持作业状态的控制和变化。

2.完成提高要求，根据提高要求更改yacc语法分析部分、添加通配符“\*”“？”、支持管道、支持ctrl+c杀死前台进程。

3.利用本次实验，初步了解linux的一些运行机制。主要是进程的通信机制、信号量控制方法。Yacc、lex工具的使用方法。

# 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。

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，改变当前运行的前台/后台状态，以及控制作业的挂起、中止和继续执行。

## 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分析、语法分析。

对其他常用的内部命令进行实现，并可以尝试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.完成了基本要求并填了一些bug。

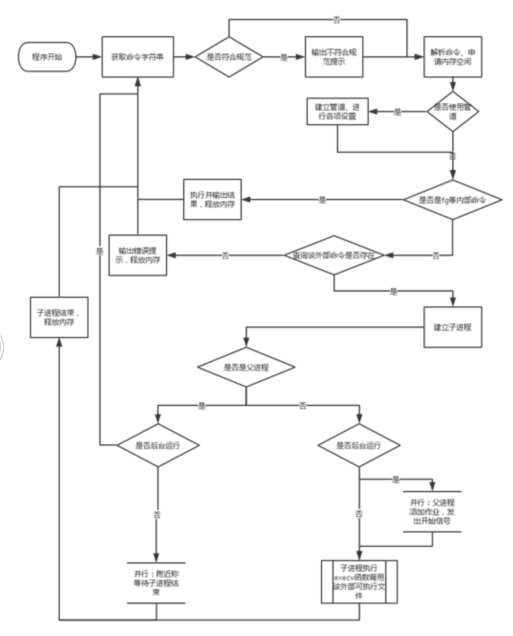
2.完成了对扩充的指令修改对应的yacc&lex的词法分析

3.增加内部命令type、echo、pwd。

4.支持Ctrl+c杀死当前的前台进程。

5.管道（虽然一直研究但未能实现）。

3.程序流程图



## 3.2基本要求实现说明

配以详细的代码和实现步骤截图。

【注：由于最终调整，代码可能有细微差别，但是思路是一样的】

修复bug如下：

1.（1）、错误类型：键入形如”./Demo&”的后台命令，后台命令不会执行。因为父进程错误的、过早的杀死了子进程。为了解决这个问题，修改了父进程对于后台命令的处理代码。

（2）、错误分析：父进程过早的杀死了子进程，这时我们就需要父进程等待子进程的执行，故可用sleep()函数来让父进程等待子进程来实现。

2.（1）错误类型：运行前台作业，键入Ctrl+Z进程被挂起，但是当再次执行fg%<int>命令后继续运行前台作业时，提示符在作业还没有运行完就打印了出来。

（2）错误分析：父进程没有等待子进程执行完，这个问题和第一个错误类型是相似的，这样我们就可以用相似的方法进行解决。我们仍然需要让父进程user-sh等待子进程，这样我们只需要继续sleep（1），但是我觉得应该有更好的方法。

3、（1）错误类型：运行后台作业时，键入Ctrl+Z会导致后台作业也停止运行。

（2）错误分析：原因是后台作业作为子进程继承了父进程的处理方式，直到子进程调用exec函数。所以在键入Ctrl+Z后，父进程执行代码中的信号处理函数，而子进程执行默认操作挂起。所以，我们需要让子进程与父进程执行不同与默认操作的操作，我们需要将子进程变为一个单独的进程，这时我们需要setpgrp()(将一个进程加入一个新的进程组)来实现，并且groupsID是自己的进程号。

4、（1）错误类型：执行后台命令如./Demo &时，在后台进程执行结束时，父进程没有等待他。也就是键入外部指令ps时，该进程仍然存在。即出现了僵尸进程。

（2）错误分析：出现僵尸进程的原因是在子进程结束了，但是他的父进程(还没有结束)没有等待（wait，waitpid）它。所以我们消除僵尸进程，可以让父进程等待子进程。所以我们可以在父进程执行rmjob时，让父进程等待子进程结束时发出的信号，即在rmjob函数里加上一条等待的指令（wait或waitpid）来让父进程等待子进程，代码：if(now->isBack==1){while(waitpid(now->pid,NULL,0));}。在bg\_exec与fg\_exec中改变当时的job的前后台状态（在之前已经加在global.h文件里的job中加入了isBack属性）。

## 3.3 提高要求实现说明

1. 完善lex和yacc的文法，由于加入了管道符’|’，所以lex和yacc的文法需要进行完善，以支持管道命令（可惜最终管道没有实现）。首先是在yylex函数中加入返回’|’的功能，使得’|’能够称为文法中单独出现的符号。

2.实现了type、echo、pwd三个内部指令。

（1）、type

功能：判断一个命令的类型（内部命令或外部命令）。

实现：在源代码里exists()函数已经给出了判断一个命令是否为外部命令，所以我们至于要再判断是否为内部命令，我们可以利用枚举的方法strcmp(cmdFile,"内部命令")==0来判断。

（2）、echo

功能：在标准输出输出一串字符。

实现：只需要printf（“%s”,要输出的字符串）；即可。另外对于echo -n 只需要判断一下就行。

（3）、pwd

功能：输出当前路径。

实现：同echo类似，用getcwd(NULL,0)函数获得当前路径，然后输出即可。

3.实现了Ctrl+C指令。

功能：杀死前台正在执行的进程。

实现：在看到Ctrl+C的功能时，发现和Ctrl+Z很类似。所以就模仿了Ctrl+Z来写。并且在网上了解到Ctrl+C传出的信号时SIGINT。故需要把kill(now->pid, SIGCONT); sleep(1);waitpid(fgPid, NULL, 0);改为kill(now->pid,SIGINT);另外还需要在fg\_exec中加入signal(SIGINT, ctrl\_C); 来设置signal信号，为下一次按下组合键Ctrl+C做准备。

4.管道

教材中对于管道的说明：

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。

实现:管道的部分性质与重定向很类似，我们就模仿重定向的使用方法来实现。在命令解析中，对关键字"|"进行读取。例如A|B,当系统读取到了管道的关键字就会对相应变量进行赋值，保存A.B两个指令。在命令执行时，如果有没有管道操作（cmd->pipes==NULL）,否则表示存在管道操作。此时我利用了教材中对于管道操作所采用的方法完成了我们组在管道操作中的任务。

# 4.收获和感想

通过这次实验，我们完整的阅读了source-code的代码，对于Shell程序的设计有了更加深刻的理解，同时通过自己修改程序中的Bug，提升了自己的调试和编程能力。

以下体会由负责提高要求每个部分的人提供：

YACC和LEX：最初的困惑在于即使是运行完所给的Name的例子之后，也不知道.y和.l的文件是应该如何在解析命令时发挥作用，简言之就是不了解运行机制。在网上找到了一些博客和教程，理解了makefile 的作用。知道了每个不同的文件之间的相互作用。另一个遇到的问题在于编写文件时，对于lex编写规则的表达方式不是非常的明白，就找到了很多的例子来读。在这个过程中，发现yacc和lex真的是功能非常的强大有趣。希望以后可以深入的了解使用二者进行词法分析。

基础部分，几个同学一起合作，互相交流自己对于源码的写法的理解以及各种新的想法。在调试几个基础bug的过程中，我们越来越明白linux的进程生命过程的各个阶段的特点，以及execv这个比较重要的系统调用函数的特性。另外通过对各种信号量的实验，了解到了signal()的用法，以及sigaction的优点。而在调试进程组这个事情中，我们更加深入的理解到了SIGCHLD和SIGTTIN以及SIGTTOU这三个信号量的特性。

管道：我们在管道上花了很长时间，最终终于实现了管道。最初对于管道的困惑，是不知道代码加在什么地方，不能很好地理解管道和进程（父子进程之间的关系，以及他们如何运行的）的运行机制。最主要的还是当时，不知道yacc和lex与管道之间的关系。不能很清晰地明确execute.c中应该实现管道的什么功能。到最后，通过在会议上和负责yacc编写的同学交流，终于有了突破性的进展。于是就开始，初步加一个单管道，以及只能是带一个参数的管道。可是由于不知道的某个原因，在我加管道的时候，把虚拟机搞崩了。之后又接连出现了好多次。也曾经想过放弃，但是考虑到这是个团队工作，自己做不出来，并不仅是自己会付出一定代价（分数不高），团队里的其他人也会跟着受连累。于是我们在网上搜了很多有关管道的讨论、博客。

在学习管道的使用方法的过程中，大家更是深入的理解了进程间通信的各种需要注意的情况，以及管道描述符的使用方法，但是遗憾的是由于工作协调的问题，导致这一项进行的不是很好，希望以后的实验中可以加以改进。

其他提高要求以及debug：在实现Ctrl+C时，我首先了解了Ctrl+C与Ctrl+Z的不同。发现，他们只是发出的信号有所不同。其中Ctrl+C发出的是SIGINT信号，前台进程收到这个信号后被杀死。可是在这其中，我发现了一个bug。仅仅只是实现这些并不能真正地杀死前台进程，相反他会让，想让杀死的前台进程变为一个僵尸进程。于是我通过在每个函数前面加一个输出，来了解数据的传输路径，又在网上查了很多有关这个问题的解决方法。但是，都不尽如人意。功夫不负有心人，最后终于找到了解决方法，就是让父进程在rmjobs时等待子进程发出结束的信号，之后再让父进程继续运行下去。对于加的其他的外部指令，我实现了type，pwd，echo三个外部指令。在实现着三个外部指令时，我并没有花很多时间。而对于一开始的debug，我却非常痛苦。痛苦原因有四：（1）、没有很好的，很深入的了解源代码。（2）、不知道父子进程之间的关系，以及运行机制。（3）、不知道wait()、waitpid()……函数的用法，区别。（4）、没有胆量随意的更改源代码。可是不会debug又心有不甘，于是就只能用时间来弥补。到最后，通过上网搜、同学交流、QQ群上“偷窥”以及自己试着努力改，终于把bug全部修复了。

其他：通过这次的实验，我们充分的体会到了团队合作，定期交流的重要性。以及根据不同同学的能力，分配相应的任务的重要性，要做到知人善任。还有就是人的依赖性以及惰性的可恶。