**操作系统课程设计实验报告**

——实验一：shell实验

负责人姓名：王裕淞

学号：14061199

日期：2016/03/26

小组成员

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 王裕淞 | 14061199 | 主程序改善、管道 |
| 2 | 刘亮 | 14061198 | 内部命令 |
| 3 | 干逸清 | 14061214 | Lex,yacc |
| 4 | 朱瑾 | 14061118 | 报告 |

目录

**1.实验目的及要求4**

1.1目的4

1.2基本要求4

1.3提高要求4

1.4完成情况4

**2.实验内容5**

2.1前期工作5

2.2前后台进程处理5

2.3增加管道6

2.4重构作业6

2.5进程同步7

2.6内存泄露7

**3.实验感想8**

3.1源码太糟糕8

3.2时间太短8

3.3所提供资料太少8

3.4对于实验任务的分割8

3.5对于理论和实践的结合问题8

1. **实验目的及要求**

1.1目的

1）学习Linux相关软件工具的使用；

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

1.2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，提供作业控制工程，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

1.3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将Lex和Yacc结合起来使用进行语法和词法分析

3）对其他常用的内部命令进行实践，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）实现结合词crtl+Z命令

1.4完成情况

1、完成基本要求

2、完成提高要求

1. **实验内容**

2.1前期工作

得到代码后进行编译，产生大量warning，同时makefile非常不方便调试，因此对makefile和源代码都进行了一些调整。

首先makefile可以写得更精简，并且支持了CFLAGS变量控制编译，方便之后使用gdb进行调试。

execute.c中出现了许多未使用的局部变量，删除。字符串字面量中百分号没有转义，添加上。

为global.h添加\_GNU\_SOURCE宏定义消除其他warning。

2.2前后台进程处理

运行代码时发现，被Ctrl+Z挂起的进程，fg运行后依然收到Ctrl+Z停止信号并停止工作。

尝试添加Ctrl+C（参考原Ctrl+Z方式），测试发现按下Ctrl+C时，子进程将受到两次SIGINT，父进程收到一次SIGINT，这个时候我认为源码对信号的处理方式大概是错误的。

对代码进行观察不难发现，代码中没有出现任何与前后台进程相关的系统调用的使用，与实际shell运行时产生的现象完全不符合。实际shell运行时我们可以观察到shell进程创建出的子进程和shell进程不再同一进程组中，因此进程创建和状态改变部分需要有比较大的修改。

我对这个问题的处理分前台和后台，在那之前任何子进程创建后都将重新给它分配进程组。

进入前台运行的方式有直接运行和fg调度，在这两个情况下进入时，通过tcsetpgrp系统调用将对应进程所属进程组提到前台，然后自己进入等待。关于等待的方式问题，由于waitpid系统调用可能被打断而失败，我没有进行采用，而是使用sigsuspend设置信号阻塞，只等待SIGCHLD信号。

进入后台运行主要是&结尾的命令和挂起后bg，实际情况下这两种情况反而没有前台运行需要处理的问题多。

在确定上面的方案前，进行了其他尝试，尤其针对tcsetpgrp。根据手册，这个函数如果在后台进程被调用，并且调用进程没有阻塞或者忽略SIGTTOU信号，则该进程组所有进程将受到SIGTTOU。进程默认对SIGTTOU的处理方法是将自己停止。另一方面，相对子进程，只有shell进程才能明确调用tcsetpgrp来将自己调到前台，所以shell在执行前台子进程而自己在后台进程等待时需要采取额外的手段来保证SIGTTOU对自己无影响。两种方案，一种是对SIGTTOU进行捕获或者忽略，保证tcsetpgrp执行成功，但这样做至少有两点不好：1.不科学，一般来说SIGTTOU和SIGTTIN两个信号不应该随意改变处理方式。2.不安全，waitpid可能在处理SIGTTOU后异常中断，导致程序继续执行。二是对SIGTTOU进行阻塞，我采用了这个办法。

另写SIGCHLD信号处理函数sg\_chld，rmJob从现在开始确定只做字面意义的工作——移除作业。因此，ingnore全局变量不再需要（之前是因为收到SIGCHLD后还要判断是否应当处理或者忽略）。SIGCHLD在子进程停止、结束、继续运行的情况下都会被触发，分别对三种情况进行处理。收到子进程结束，将shell调到前台（此时除SIGCHLD的信号均被阻塞，所以可以成功调用），从作业中删除结束的进程，用waitpid回收该退出的进程。收到子进程停止，只需要将自己调到前台。子进程继续工作，如果是前台继续工作，则在此再次阻塞SIGCHLD信号外的信号进行等待。

这样一来shell进程能够保证安全的阻塞、僵尸进程的回收、前后台的正常切换，虽然这些问题在做管道的时候又再次出现了，但对于运行单个进程的程序，大概是能够满足要求了的。

另外，我删除了子进程和父进程之间同步用的死循环和信号处理，为之后埋下了隐患。

在测试过程中发现&结尾的命令导致了下一条命令前会多一个-1（EOF），没有定位到出现问题的地方，仅仅对输入字符再进行一个判断暂且解决。

2.3增加管道

管道前先对真实shell进行测试，发现重定向会取代管道，那么只要在写程序时先处理管道再处理重定向就行了，问题不大。

在bison.y里新增fgCommand规则，fgCommand可以由多个simpleCmd中间用|隔开。

父进程阻塞前需要创建所有的子进程，原代码的基础上我选择用递归，同时SimpleCmd结构体增加管道输入输出和链表指针，实际上实现管道运行本身并不困难，主要要在合适的时机对pipe产生的两个文件描述符进行关闭（不关闭可能导致管道无法正确接受OEF），因为pipe产生的两个fd一个用于接下来的子进程，另一个用于下一个子进程。在execute函数中创建出命令链表，只需要在原代码的基础上进行小幅度修改完成。其余的修改主要是配合新加的修改，同时发现现有的作业机制完全无法满足一条命令产生多个直接子进程后续的控制，需要对作业这一块进行重构。

2.4重构作业

细节很难描述清楚

概念上一个作业应该有一个或多个进程， 当所有子进程退出时撤销作业或者标记作业为完成，当所有子进程停止时标记作业为停止。fg、bg操作应该针对作业号而不是针对进程号。

首先重新定义jobs链表，用来存放进程号链表、命令内容、状态，并且附带多个相关的函数用来对这些信息进行使用，大概实现有增加一个作业、删除一个作业、查找一个作业。配合这个对代码进行修改，在调试过程中发现了两个问题:

1.父进程子进程需要同步，子进程需要保证在被添加到作业前不能结束。

2.内存泄漏，在命令参数增加的情况下有更大的概率shell异常退出，问题在动态分配上，重查代码发现这个问题一直存在，就是命令执行后cmd结构体的释放时处理有两点不正确。

2.5进程同步

父进程子进程之间的同步很重要，因为不能保证fork后两边的执行进度，可能发生关于一个作业先被尝试删除再被尝试添加的我难题，这可能导致系统调用返回异常，这个时候我是很后悔当时删掉那段同步代码的。

但是，也许有更好的解决方案。查看手册发现，vfork可以保证子进程优先执行，并且在exec之前和父进程共享数据，于是利用vfork解决这个问题，在子进程exec之前利用子进程优先执行并可以修改父进程部分数据，完成作业的添加，至此实验告一段落。

2.6内存泄漏

定位这个问题的过程是艰难的，当时是我free了一个合法指针发生段错误，用了各种手段没有找到直接错误，最后怀疑是内存泄漏时对代码进行重查，发现execute中关于cmd的malloc和free严重不匹配，我觉得有必要把原代码拿出来批评一下

原代码：

for(i = 0; cmd->args[i] != NULL; i++){

free(cmd->args[i]);

free(cmd->input);

free(cmd->output);

}

可以发现，如果循环体只执行一次，那么发生内存泄漏的是cmd结构体本身没被释放以及一个存放了NULL的args[i]和args本身，总共没几个字节，所以侥幸很长时间都没有出现问题。

但是如果循环体执行多次，则不止这个问题，最严重的是对input和output的多次释放，在我的调试过程中，可以断定正是这两个重复释放导致了我的程序在正常free下出现段错误。

修改后，不再出现问题（暂且）。

**3、实验感想**

3.1 源码太糟糕

所提供的源码结构混乱、书写累赘，浪费了大量时间在修改完善源码上；

3.2 时间太短

短期内要对实验要求做深入理解，查阅并学习相关知识，进行大量的阅读和涉猎，并且对于源码的修改和新功能的实现，均需要投入时间与精力，于实验规定的时间内完成有相当大的压力和紧迫感；

3.3 所提供的资料太少

由于实验所需知识有很多超过了课堂内容，需要自学，然而所提供的学习资料却又远远不能满足达到实验要求的程度，对于紧张的时间和繁重的工作量来说，成员需要在寻找资源上花费很多的时间；

3.4 对于实验任务的分割

实验所给出的任务在一定程度上有很强的相关性，对于刚开始合作的组员来说，很难做出合理的分割和任务分配，由此导致的实验进程拖延和组内沟通不畅，不仅给实验的完成造成了一定程度上的障碍，还延缓了组内的互相了解和合理同步提高；

3.5 对于理论和实践的结合问题

理论课上的知识与实验中所遇到的问题往往不在同一方面，理论和时间的结合点很少，并且对于基础较差的同学而言，入门门槛太高，并不利于对于实验的理解和对理论知识的更深一步体会。