



编号： _____39061416_____

版本： _____1.0_____

北京航空航天大学
B E I H A N G U N I V E R S I T Y

Project : The Yalnx Shell

编写:(签名)_____黄建宇_____ 2012年 3月25日

复查:(签名) _____黄建宇_____ 2012年 3月25日

批准:(签名)_____黄建宇_____ 2012年 3月25日



目录

1 小组成员.....	1
2 实验目的与总体结构.....	1
2.1 实验目的.....	1
2.2 需求说明.....	1
2.3 设计说明.....	2
2.3.1 结构设计.....	2
2.3.2 功能设计.....	4
3 实验内容.....	8
3.1 实验步骤.....	8
3.2 实现方法.....	9
1、命令解析.....	9
2、初始化.....	9
3、简单命令.....	9
4、内部命令.....	10
5、外部命令.....	10
6、管道命令.....	10
7、信号处理函数.....	11
3.3 测试和使用说明.....	11
1.登录.....	12
2 cd 命令演示.....	12
3 exit 命令演示.....	13
4 I/O 重定向命令演示	13
5 后台作业演示.....	14
6 管道命令演示.....	14
7 作业控制综合演示.....	14
4 每个人的工作与会议记录.....	17



4.1	会议时间表.....	18
4.2	相关记录.....	18
4.2.1	问题描述.....	18
4.2.2	错误原因.....	20
4.2.3	解决方法.....	21
4.3	其他说明.....	22
4.4	程序清单.....	22
5	实验心得.....	23
5.1	心得：	23
5.2	建议：	24



1 小组成员

每个人的贡献大小以此排名为依据，靠前的为贡献较大者

39061416 黄建宇

2 实验目的与总体结构

了解实验的目的，对于作业的整体设计说明，要求思路清晰，表达明确

2.1 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。

2.2 需求说明

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。

1.user-sh 程序应当具有如下一些重要的特征：

- 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。
- 使用输入/输出重定向和管道（管道功能可选）。
- 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 (Ctrl+Z, Ctrl+C)。

2.除此之外，在整个实验过程中，还须做到以下几点：

- 使用 make 工具建立工程。



- 使用 gdb 或者 ddd 等调试工具来调试程序。
- 提供清晰、详细的设计文档和解决方案。

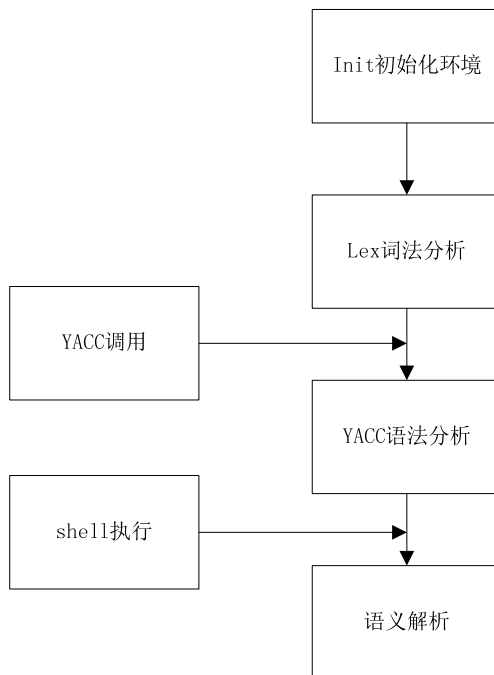
3. 扩展要求

- 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。
- 实现对管道的支持。
- 对多重管道，对管道进程组的作业控制（Ctrl+C, Ctrl+Z）

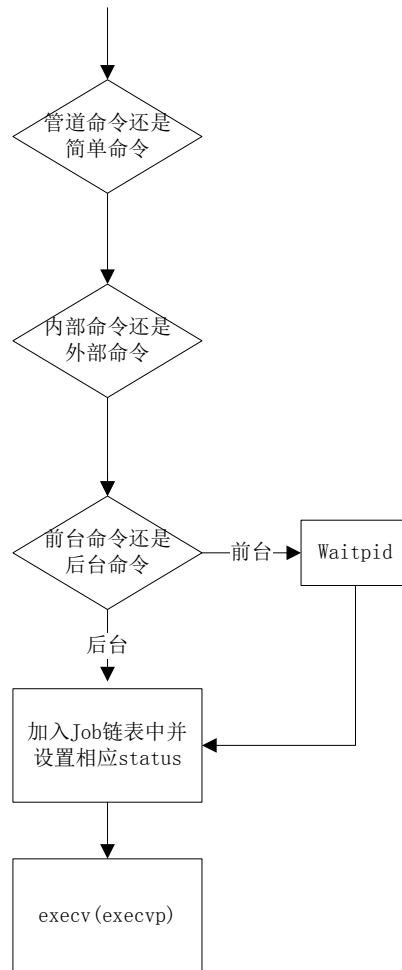
2.3 设计说明

2.3.1 结构设计

- 程序逻辑设计



- 语法解析详细流程图



➤ 相关说明

- 1、**命令解析**：判断命令是简单命令还是管道命令，判断命令是否有输入输出重定向，判断前后台。如果是管道命令，将命令分解。判断命令是否合法。
- 2、对于**简单命令**，判断是否是内部命令。如果是内部命令，则转至相应的处理函数；如果是外部命令，则创建新的进程，运行相应的命令。
- 3、对于**管道命令**，创建管道，创建进程，输入输出重定向到管道，运行新进程。
- 4、**信号控制**：对所有子进程封锁 CTRL+Z 和 CTRL+C 信号，只有父进程能收到信号，收到后，对相应的子进程进行控制。bg 和 fg 命令就是对相应作业的进程节点指示的进程组号发送信号，控制其运行。
- 5、**同步和并发性控制**：必须先建立节点再运行子进程，这点仿照了实验参考书。子进程运行完后会自动向父进程发信号，通过此信号控制进程节点的删除。



2.3.2 功能设计

1、数据结构:

- 作业采用链表存储，每个作业节点存储作业的 id，创建的进程 pid（多个进程时保存第一个 pid），进程组号 groupid，命令行内容 cmd，运行状态 status(FOREGROUND,BACKGROUND,WAITING_FOR_INPUT)。

如以下代码所示:

```
typedef struct Job
{
    int id;
    //int pid[MAXPIPE+1];
    int pid;                //pid number
    int groupid;            //groupid
    char cmd[100];          //command str
    char state[10];         //state
    int status;             //status
    struct Job *next;       //next point
}Job;
```

- 历史记录使用循环链表，共 10 个节点;

如以下代码所示:

```
typedef struct History
{
    int start;              //start position
    int end;               //end position
    char cmds [HISTORY_LEN][100]; //history command
}History;
```



➤ 命令体采用结构体，如下所示

```
typedef struct Command
```

```
{  
    char *args[MAXARG];           //command and parameters  
    char *input;                   //input redirect  
    char *output;                  //output redirect  
}Command;
```

2. 函数功能说明

execute.c

函数名称	函数作用
void pr_ids(char *name)	for debug : present the information of the running process
void errorPrint(char *s)	for error handler, print the error information
void welcomeShell()	Welcome information
void signalHandler_child(int p)	signal handler for SIGCHLD
void waitJob(Job* job)	waits for a job, blocking unless it has been suspended. Deletes the job after it has been executed
void putJobForeground(Job* job, int continueJob)	puts a job in foreground. If continueJob = TRUE, sends the process group a SIGCONT signal to wake it up. After the job is waited successfully, it restores the control of the terminal to the shell
void putJobBackground(Job* job, int continueJob)	puts a job in background, and sends the job a continue signal, if continueJob = TRUE, puts the shell in foreground
int exists(char *cmdFile)	Judge whether the command exists



int str2Pid(char *str,int start,int end)	Transform the String to Number
void justArgs(char *str)	Change the form of some outer commands
void fg_exec(int No)	fg command
void bg_exec(int No)	bg command
void addHistory(char *cmd)	add HistHISory
void getEnvPath(int len,char *buf)	get the environment path
void init()	initial
int handleCommandStr(char *buf,int cmdnum)	Command parser
void exitCmd(Command *cmd)	exit Command
void historyCmd(Command *cmd)	history Command
void jobsCmd(Command *cmd)	jobs Command
void cdCmd(Command *cmd)	cd Command
void fgCmd(Command *cmd)	fg Command
void bgCmd(Command *cmd)	bg Command
void execOuterCmd(Comman	execute outer command



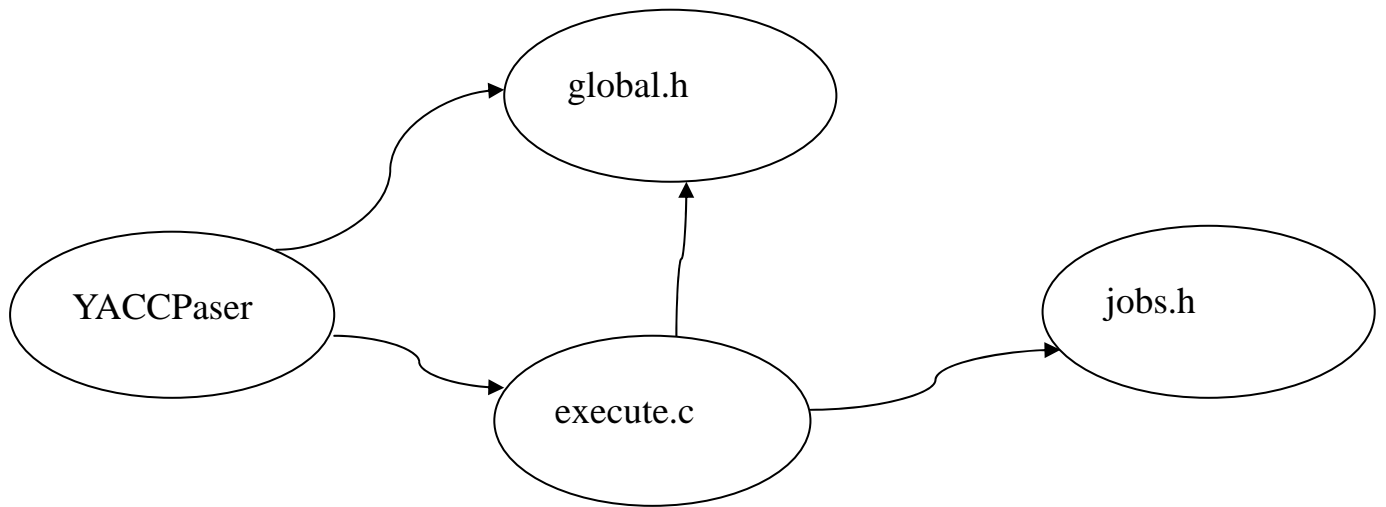
d *cmd,int executionMode)	
int innerCmd(Command *cmd)	inner Command
void execCommand(Command *cmd)	execute the simple command
void execPipeCmd(int executionMode)	execute the pipe command
void execute()	the interface for the yacc command

jobs.h

Job* getJob(int searchValue, int searchParameter)	get the job by searchValue according to the searchParameter
int changeJobStatus(int pgid, int status)	modifies the status of a job
Job* delJob(Job* job)	deletes a no more active process from the linked list
void printJobs()	prints the active processes launched by the shell
Job* insertJob(int pid, pid_t pgid, char* name,int status)	inserts an active process in the linked list



3.接口与调用关系设计



4. 核心逻辑或功能代码

请见 3 实验内容中的 3.2 实现方法以及详细注释。

3 实验内容

详细说明实验的过程，实现方法及遇到的问题

3.1 实验步骤

- (1) 阅读关于 fork、exec、wait 和 exit 系统调用的 man 帮助手册。
- (2) 编写小程序练习使用这些系统调用。
- (3) 阅读关于 tcsetpgrp 和 setpgid 的 man 帮助手册。
- (4) 练习编写控制进程组的小程序，要注意信号 SIGTTIN 和 SIGTTOU。
- (5) 使用 YACC 设计命令行分析器。
- (6) 实现命令行分析器。
- (7) 使用分析器，写一个简单的 shell，它能执行简单的命令。
- (8) 增加对程序在后台运行的支持，增加 jobs 命令。
- (9) 增加输入/输出重定向功能。
- (10) 添加代码支持在后台进程结束时打印出一条信息。



- (11) 添加作业控制特征，主要实现对 Ctrl+C、Ctrl+Z 的响应，实现 fg 和 bg 命令功能。
- (12) 增加对管道的支持。
- (13) 实现上面所有部分并完成集成。
- (14) 测试程序，解决存在的问题。
- (15) 撰写相应的文档和报告。
- (16) 结束。

3.2 实现方法

1、命令解析

对于命令的语法解析，首先使用 yacc 实现，然后在 execute.c 中的 handleCommandStr 函数中继续语义解析，将得到的相应信息存入到 Command 数组信息中。

2、初始化

- 创建历史记录的循环链表。
- 获取系统默认路径以及 cmdenv.conf 所确定的环境路径。
- 将除了 SIGCHLD 之外的所有信号（SIGQUIT, SIGTTOU, SIG_IGN, SIGTTIN, SIGTSTP, SIGINT）全部设为 SIG_IGN, 使得之后 fork 的父进程对 Ctrl+C, Ctrl+Z 不做任何处理；将 SIGCHLD 信号处理重新设定，与 signalHandler_child 函数绑定。

3、简单命令

- 判断是否是内部命令，是则跳至相应的函数进行处理，退出；
- 添加作业节点，进行外部命令的处理。



4、内部命令

➤ cd

系统调用，改变当前目录，修改 yalnix 中的相应记录当前路径变量。

➤ history

遍历历史记录链表，并打印。

➤ exit

退出程序。

➤ jobs

遍历作业链表，打印所有记录为后台的作业。

➤ fg

在作业链表中找到相应的节点，向它的进程组发 SIGCONT 信号，使其运行，并修改作业节点中的状态。父进程等待子进程运行完再运行。

➤ bg

在作业链表中找到相应的节点，向它的进程组发 SIGCONT 信号，使其运行，并修改作业节点中的状态。

5、外部命令

➤ 在默认的路径以及 cmdenv.conf 确定的环境路径里查找，如果没有，则报错返回。

➤ 创建进程，子进程等待父进程加入作业节点，子进程设置除了 SIGCHLD 的所有信号（SIGQUIT, SIGTTOU, SIG_IGN, SIGTTIN, SIGTSTP, SIGINT）为 SIG_DFL（系统默认），判断是否有输入输出重定向，有则重定向，执行新的命令；父进程创建进程节点，添加到相应的作业节点的进程链表中，如果是前台命令则等待子进程运行。

6、管道命令

设命令数为 n

➤ 创建 n-1 个管道，

➤ 进入循环，对每一个命令创建新的进程；在子进程中，设置除了 SIGCHLD



的所有信号 (SIGQUIT, SIGTTOU, SIG_IGN, SIGTTIN, SIGTSTP, SIGINT) 为 SIG_DFL (系统默认), 判断是否有输入输出重定向, 有则重定向, 执行新的命令。如果是第一个命令, 则将标准输出重定向到第一个管道, 如果有输入重定向, 则重定向, 如果是最后一个命令, 则将标准输入重定向到最后一个管道, 如果有输出重定向, 则重定向, 如果是中间的第 i 个命令, 则将标准输入重定向到第 $i-1$ 个管道, 标准输出重定向到第 i 个管道, 运行新的命令; 在父进程中, 关闭相应的管道, 如果是前台程序, 则等待子进程。

7、信号处理函数

SIGCHLD

- 如果判断是否是子进程运行结束所致, 如果不是, 则跳出。否则, 在作业链表和进程链表中查找相应的节点并删除。
- 这是本程序最关键的地方。通过对 `waitpid` 的返回 `status` 进行 `WIFEXITED`、`WIFSIGNALED`、`WIFSTOPPED` 判断子进程的退出原因是正常退出、接收到 `SIGSTOP` (Ctrl+C) 还是接受到 `SIGTSTP` (Ctrl+Z) 退出。这样做可以避免书上所写的程序那样有 `rmJob` 与 `Ctrl_Z` 函数之间的竞争关系。具体请见 `signalHandler_child` 函数。

3.3 测试和使用说明

程序开发环境: gcc、gdb、bison

运行环境: ubuntu 9.04, linux 内核 2.6 以上版本

安装说明: 在 shell 中打开源码所在文件夹, make 编译后, 输入 `./user-sh` 即可安装运行。

测试用例和运行结果分析:



1. 登录

make 编译后，输入./user-sh。

```
james@james-desktop: ~/Desktop/yalnixshell
File Edit View Terminal Help
james@james-desktop:~$ cd '/home/james/Desktop/yalnixshell/'
james@james-desktop:~/Desktop/yalnixshell$ make
cc -c -g execute.c
cc -o user-sh yacc.tab.o execute.o
james@james-desktop:~/Desktop/yalnixshell$ ./user-sh

////////////////////////////////////
Welcome to shell version 1.0
Author:39061416 Huang Jianyu
Website:www.huangjy.info
Mobile Phone:15210965935
Email:hjyahead@gmail.com
////////////////////////////////////

user-sh@/home/james/Desktop/yalnixshell>
```

2 cd 命令演示

输入 cd 或者 cd 相应目录，则目录改变

```
////////////////////////////////////
Welcome to shell version 1.0
Author:39061416 Huang Jianyu
Website:www.huangjy.info
Mobile Phone:15210965935
Email:hjyahead@gmail.com
////////////////////////////////////

user-sh@/home/james/Desktop/yalnixshell>cd
user-sh@/home/james>cd ..
user-sh@/home>cd james
user-sh@/home/james>cd Desktop
user-sh@/home/james/Desktop>cd yalnixshell
user-sh@/home/james/Desktop/yalnixshell>cd
user-sh@/home/james>
user-sh@/home/james>history
cd
cd ..
cd james
cd Desktop
cd yalnixshell
cd
user-sh@/home/james>
```



3 exit 命令演示

输入 exit，则程序退出

```
james@james-desktop:~/Desktop/yalnixshell$ ./user-sh

/////////////////////////////////
Welcome to shell version 1.0
Author:39061416 Huang Jianyu
Website:www.huangjy.info
Mobile Phone:15210965935
Email:hjyahead@gmail.com
/////////////////////////////////

user-sh@/home/james/Desktop/yalnixshell>exit

ByeBye!

james@james-desktop:~/Desktop/yalnixshell$
```

4 I/O 重定向命令演示

输入 cat -l > lsfile，则输出重定向；输入 cat < lsfile，则输入重定向。

```
user-sh@/home/james/Desktop/yalnixshell>ls -l > lsfile
outputFile:lsfile
user-sh@/home/james/Desktop/yalnixshell>cat < lsfile
inputFile:lsfile
total 192
-rwxrw-rw- 1 james james 27235 2012-03-25 14:43 execute.c
-rw-r--r-- 1 james james 32228 2012-03-25 14:43 execute.o
-rwxrw-rw- 1 james james 1487 2012-03-25 11:20 global.h
-rwxrw-rw- 1 james james 4112 2012-03-25 14:31 jobs.h
-rw----- 1 james james 0 2012-03-25 14:43 lsfile
-rw----- 1 james james 703 2012-03-25 14:39 lsfile.txt
-rw-r--r-- 1 james james 268 2012-03-25 10:20 makefile
-rwxr-xr-x 1 james james 42313 2012-03-25 14:43 user-sh
-rw-r--r-- 1 james james 36 2012-03-11 23:54 user-sh.conf
-rw-r--r-- 1 james james 42638 2012-03-25 11:23 yacc.tab.c
-rw-r--r-- 1 james james 2062 2012-03-25 11:23 yacc.tab.h
-rw-r--r-- 1 james james 11500 2012-03-25 11:23 yacc.tab.o
-rw-r--r-- 1 james james 1953 2012-03-25 11:20 yacc.y
user-sh@/home/james/Desktop/yalnixshell>
```




5 后台作业演示

输入 `gedit &` 与 `ls &` 结果如下所示。

```
Welcome to shell version 1.0
Author:39061416 Huang Jianyu
Website:www.huangjy.info
Mobile Phone:15210965935
Email:hjyahead@gmail.com
////////////////////////////////////

user-sh@/home/james/Desktop/yalnixshell>gedit &
user-sh@/home/james/Desktop/yalnixshell>
[1] 7919

[1]+  Done          gedit &

user-sh@/home/james/Desktop/yalnixshell>ls &
user-sh@/home/james/Desktop/yalnixshell>
[1] 7923
a  execute.c  global.h  hist      user-sh    yacc.tab.h
aaa execute.o  hi        jobs.h    user-sh.conf yacc.tab.o
b  fgtest     his       makefile  yacc.tab.c yacc.y

[1]+  Done          ls &

user-sh@/home/james/Desktop/yalnixshell>
```

6 管道命令演示

输入 `ls | cat`，则管道连接结果如下

```
user-sh@/home/james>ls | cat
bochs-2.4.6
Desktop
Documents
Downloads
examples.desktop
forktest01.c
hist
Music
Pictures
Public
Templates
Videos
workspace
user-sh@/home/james>
```

7 作业控制综合演示

1. 输入 `find /`



```
james@james-desktop:~/Desktop/yalnixshell$ ./user-sh
```

```
////////////////////////////////////
Welcome to shell version 1.0
Author:39061416 Huang Jianyu
Website:www.huangjy.info
Mobile Phone:15210965935
Email:hjyahead@gmail.com
////////////////////////////////////
```

```
user-sh@/home/james/Desktop/yalnixshell>find /
```

2.再按下 Ctrl+Z，并输入 jobs

```
/home/james/Desktop/shell/shelltest0.1/execute.c
/home/james/Desktop/shell/shelltest0.1/yacc.tab.c
/home/james/Desktop/shell/shelltest0.1/hello.c
/home/james/Desktop/shell/shelltest0.1/makefile
/home/james/Desktop/shell/shelltest0.1/execute.o
/home/james/Desktop/shell/shelltest0.1/user-sh
/home/james/Desktop/shell/shelltest0.1/record to check
/home/james/Desktop/shell/shelltest0.96
/home/james/Desktop/shell/shelltest0.96/user-sh.conf^Zcome into the signalHandle
r child part!
job->id:1,job->name:find /,pid:7565,groupid:7565,status:70
a job receives a SIGSTP signal
successChangeStatus:1job->status:S

[1]+  stopped      find /
user-sh@/home/james/Desktop/yalnixshell>jobs

Jobs List:
-----
| job no. |                command | groupid | status |
-----
|      1 |                find / |    7565 |      S |
-----
user-sh@/home/james/Desktop/yalnixshell>
```

3.输入 find / | grep a

```
/home/james/Desktop/vmware-tools-distrib/lib/hlp/wwhdata/js/search/pairs/pair14.
js
/home/james/Desktop/vmware-tools-distrib/lib/hlp/wwhdata/js/search/pairs/pair9.j
s
/home/james/Desktop/vmware-tools-distrib/lib/hlp/wwhdata/js/search/pairs/pair6.j
s
/home/james/Desktop/vmware-tools-distrib/lib/hlp/wwhdata/js/search/pairs/pair8.j
s
/home/james/Desktop/vmware-tools-distrib/lib/hlp/wwhdata/js/search/pairs/pair22.
js
/home/james/Desktop/vmware-tools-distrib/lib/hlp/wwhdata/js/search/pairs/pair11.
js^Z
[1]+  stopped      find /
user-sh@/home/james>find / | grep a
```



4.按下 Ctrl+Z, 输入 jobs

```
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.6.27-7-x86_64s
erver-Ubuntu8.10
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.6.27-7-x86_64s
erver-Ubuntu8.10/properties
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.6.27-7-x86_64s
erver-Ubuntu8.10/objects
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.6.27-7-x86_64s
erver-Ubuntu8.10/objects/vmsync.o^Z

[2]+  stopped    find /| grep a
user-sh@/home/james>jobs

Jobs List:
-----
| job no. |                command | groupid | status |
-----
|      1 |                find / |    8798 |    S |
|      2 |          find /| grep a |    8910 |    S |
-----
user-sh@/home/james>
```

5.输入 jobs

```
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.6.27-7-x86_64s
erver-Ubuntu8.10/properties
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.6.27-7-x86_64s
erver-Ubuntu8.10/objects
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.6.27-7-x86_64s
erver-Ubuntu8.10/objects/vmsync.o^Z

[2]+  stopped    find /| grep a
user-sh@/home/james>jobs

Jobs List:
-----
| job no. |                command | groupid | status |
-----
|      1 |                find / |    8798 |    S |
|      2 |          find /| grep a |    8910 |    S |
-----
user-sh@/home/james>fg %2
```

6.按下 Ctrl+C



```
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/properties
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/objects
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/objects/vmhgfs.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/objects/vsock.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/objects/vmblock.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/objects/vmmemctl.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/objects/vmxnet.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/smpPAE-2.4.10-SuSE-7
.3/objects/vmci.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.4.9-49.athlons
mp-RH1091807538
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.4.9-49.athlons
mp-RH1091807538/properties
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.4.9-49.athlons
mp-RH1091807538/objects^C
[2]+  KILLED      find /| grep a
user-sh@/home/james/Desktop/yalnixshell>
```

8. 输入 jobs

```
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.4.18-5-smp-TL8
0S/objects/vsock.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.4.18-5-smp-TL8
0S/objects/vmblock.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.4.18-5-smp-TL8
0S/objects/vmmemctl.o
/home/james/Desktop/vmware-tools-distrib/lib/modules/binary/bld-2.4.18-5-smp-TL8
0S/objects/vmxnet.o^C
[2]+  KILLED      find /| grep a
user-sh@/home/james>jobs

Jobs List:
-----
| job no. |                command | groupid | status |
-----
|      1 |                find / |    8798 |    S |
-----
user-sh@/home/james>
```

4 每个人的工作与会议记录

每次实验至少要开 4 次小组会议,每次会议都要记录以下内容:

- 说明每一位组员前一阶段完成的具体工作,是否按时按量完成任务。
- 说明每一位组员下一阶段需要完成的工作。
- 其他会议内容(如讨论的问题以及解决方案等)



4.1 会议时间表

2011 年 3 月 2 日	Shell 设计	黄建宇
2011 年 3 月 10 日	Shell 文档书写	黄建宇
2011 年 3 月 16 日	Shell 文档审查，修改	黄建宇
2011 年 3 月 18 日	Shell 编写	黄建宇
2011 年 3 月 20-25 日	调试，debug	黄建宇

每次会议都基本完成上次会议所计划内容，并规划下次完成进度。

4.2 相关记录

4.2.1 问题描述

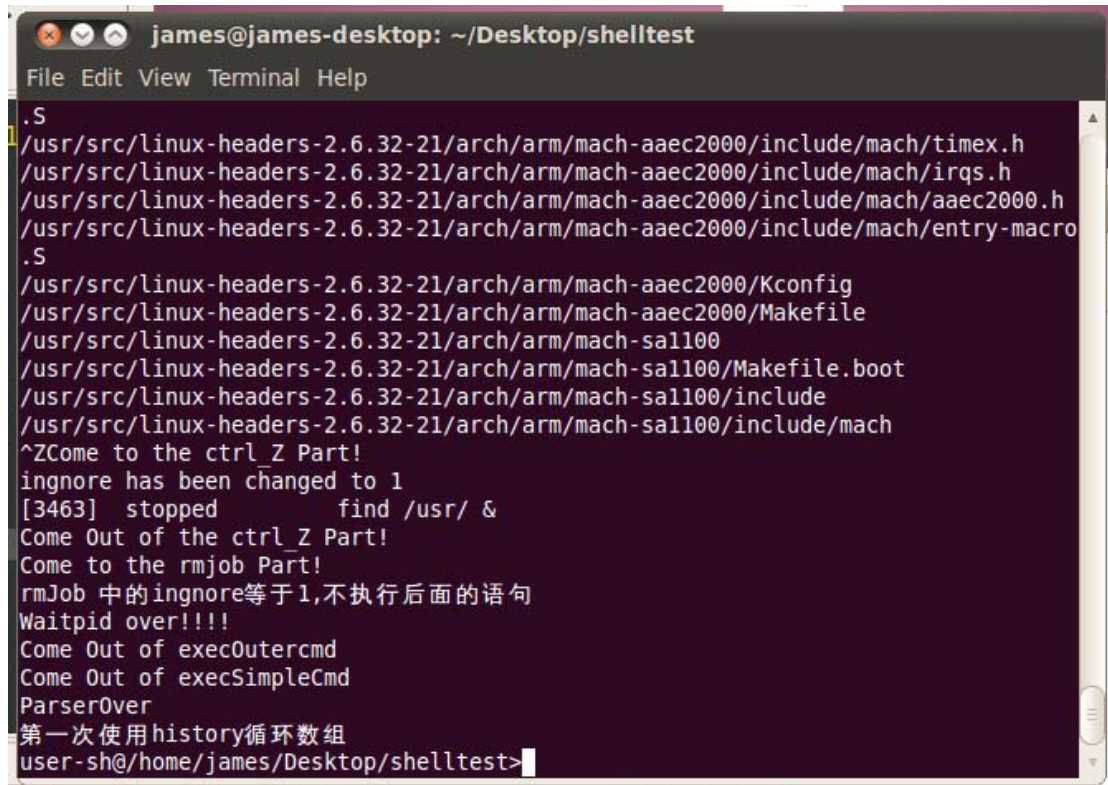
打开文件夹目录，make 即可编译，./user-sh 执行

envPath[0]...是环境变量，作为打印出的调试信息。

在 shelltest 中，输入 find /usr/，如下图所示：

```
james@james-desktop: ~/Desktop/shelltest
File Edit View Terminal Help
james@james-desktop:~$ cd Desktop
james@james-desktop:~/Desktop$ cd shelltest
james@james-desktop:~/Desktop/shelltest$ make
cc -o user-sh yacc.tab.o execute.o
james@james-desktop:~/Desktop/shelltest$ ./user-sh
envPath[0]:/bin/
envPath[1]:/usr/bin/
envPath[2]:/usr/local/bin/
envPath[3]:/sbin/
user-sh@/home/james/Desktop/shelltest>find /usr/
```


这时由于查找的文件太多，查找时间很长，中间键盘按下 **Ctrl+Z** 键，如下图所示：

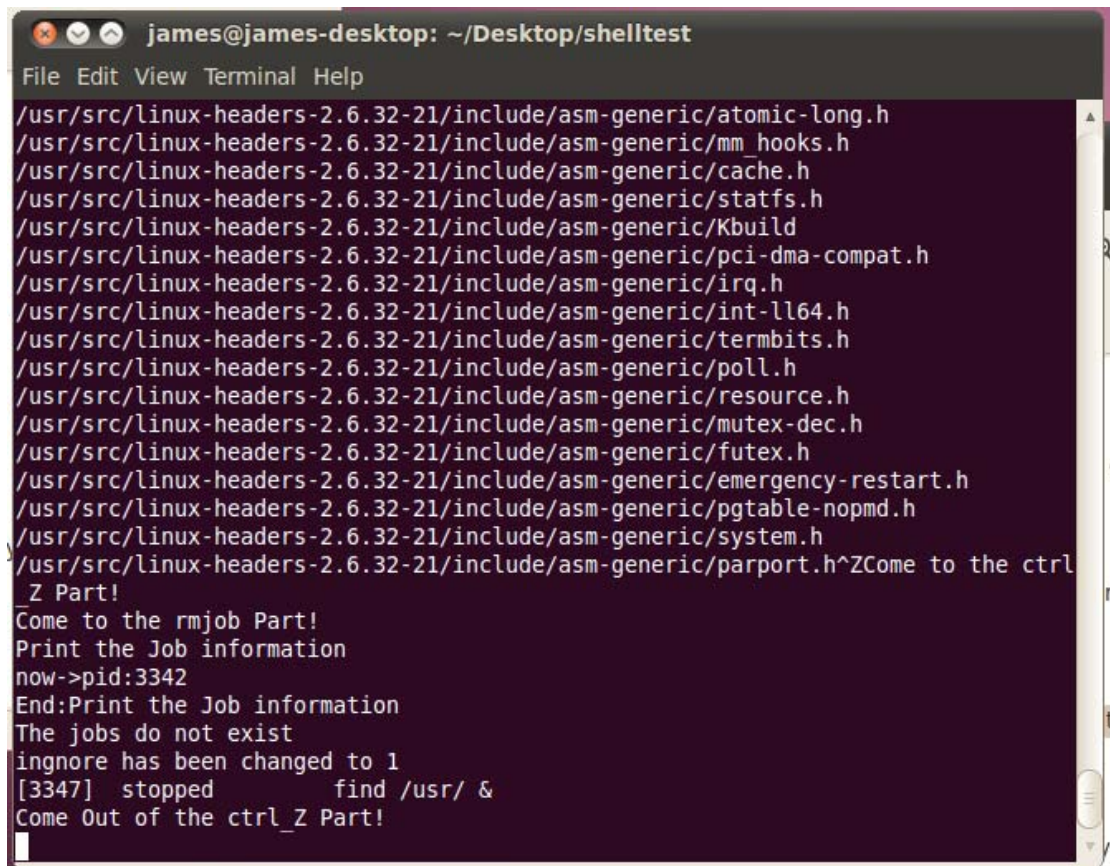


```
james@james-desktop: ~/Desktop/shelltest
File Edit View Terminal Help

.S
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/timex.h
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/irqs.h
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/aaec2000.h
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/entry-macro
.S
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/Kconfig
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/Makefile
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100/Makefile.boot
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100/include
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100/include/mach
^ZCome to the ctrl_Z Part!
ignore has been changed to 1
[3463] stopped      find /usr/ &
Come Out of the ctrl_Z Part!
Come to the rmjob Part!
rmJob 中的ignore等于1,不执行后面的语句
Waitpid over!!!!
Come Out of execOuterCmd
Come Out of execSimpleCmd
ParserOver
第一次使用history循环数组
user-sh@/home/james/Desktop/shelltest>
```

Come to the Ctrl_Z Part 表示进入了 `execute.c` 中的 `Ctrl_Z` 函数，然后 `ignore` 值被改为 1，正常打印出 `jobs` 信息，并正常退出 `Ctrl_Z` 函数（打印 `Come Out of the Ctrl_Z Part`），结束前向子进程发出 `kill SIGSTOP` 信号（程序 `execute.c` 第 270 行），使得子进程向父进程发出 `SIGCHLD` 信号，从而进入 `rmJob` 函数。由于 `ignore=1`，`rmJob` 函数在中间就 `return` 了。

再次输入同样的命令 `find /usr/`，再次在中间按下 **Ctrl+Z** 键，出现如下图所示的打印输出：



```
james@james-desktop: ~/Desktop/shelltest
File Edit View Terminal Help
/usr/src/linux-headers-2.6.32-21/include/asm-generic/atomic-long.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/mm_hooks.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/cache.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/statfs.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/Kbuild
/usr/src/linux-headers-2.6.32-21/include/asm-generic/pci-dma-compat.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/irq.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/int-ll64.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/termbits.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/poll.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/resource.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/mutex-dec.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/futex.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/emergency-restart.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/pgtable-nopmd.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/system.h
/usr/src/linux-headers-2.6.32-21/include/asm-generic/parport.h^ZCome to the ctrl
_Z Part!
Come to the rmjob Part!
Print the Job information
now->pid:3342
End:Print the Job information
The jobs do not exist
ingnore has been changed to 1
[3347] stopped      find /usr/ &
Come Out of the ctrl_Z Part!
```

Come to the ctrl_Z Part 后，（中断重入吗？），进入 rmJob Part（但是好像没有谁发出 SIGCHLD）信号，此时 ignore 并没有改变，导致错误发生。最后才又回到 Ctrl_Z 函数，并且改变 ignore 值为 1，最后打印出“Come Out of the Ctrl_Z Part”。而这时程序好像进入死循环，输入任何字符没有响应，只能 Ctrl+C 退出。

以上两种情况：*同样的输入，不同的输出。*

重复多次这样”find /usr/”，Ctrl+Z，每次结果都不确定。可能先进入 Ctrl_Z 函数，也有可能先进入 rmJob 函数，也有可能 Ctrl_Z 函数执行到一半，“中断重入”到 rmJob 函数。

4.2.2 错误原因



fork 函数之后的 exec: 不“继承”父进程的信号处理函数。

导致子进程 (find /usr/) 对 Ctrl+Z 键产生的 SIGTSTP 信号按照默认的方式处理, 这样就使得子进程 exec (find /usr/) **同时**产生 **SIGTSTP** 并向父进程发出 **SIGCHLD** 信号, 两个处理函数 Ctrl_Z 与 rmJob 处于竞争关系, 使得输出结果不确定。

4.2.3 解决方法

在没有 fork 之前, init 函数中, 对 SIGTSTP 与 Ctrl_Z 函数注册绑定。

fork 之后, 对 pid==0 的子进程加上这么一句:

signal(SIGTSTP,SIG_IGN); (即将 execute.c 中的第 644 行前的注释“//”去掉)

exec 可以继承对信号的屏蔽, 这样就可以将子进程的 exec (find /usr/) 的 SIGTSTP 屏蔽。

处理的过程如下:

1. 输入 find /usr/
2. 按下 Ctrl+Z 键
3. ./user-sh (父进程) 捕获到 SIGTSTP 信号, 执行 Ctrl_Z 处理函数, 而 find /usr/ (子进程) 对该信号屏蔽(就是执行上面的 SIG_IGN)。父进程继续执行 Ctrl_Z 函数, 改变 ignore 值为 1, 打印 jobs 信息, 最后向前台 fgpid 发出 kill(fgPid,SIGSTOP); 的信息。这时 find /usr/ (子进程) 被杀死, 向父进程发出 SIGCHLD 信号, 这样父进程调用 rmJob, 发现 ignore 为 1, 中间 return。这样程序的逻辑就正确了。

每次 Ctrl+Z 执行都是下面的结果, 即先进入函数 Ctrl_Z, 然后进入函数 rmJob



```
james@james-desktop: ~/Desktop/shelltest
File Edit View Terminal Help
.S
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/timex.h
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/irqs.h
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/aaec2000.h
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/include/mach/entry-macro
.S
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/Kconfig
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-aaec2000/Makefile
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100/Makefile.boot
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100/include
/usr/src/linux-headers-2.6.32-21/arch/arm/mach-sa1100/include/mach
^ZCome to the ctrl_Z Part!
ingnore has been changed to 1
[3463] stopped find /usr/ &
Come Out of the ctrl_Z Part!
Come to the rmjob Part!
rmJob 中的ingnore等于1,不执行后面的语句
Waitpid over!!!!
Come Out of execOuterCmd
Come Out of execSimpleCmd
ParserOver
第一次使用history循环数组
user-sh@/home/james/Desktop/shelltest>
```

4.3 其他说明

1 组内成员任务分工说明(注明各个组员的工作量比例)

由于本组只有 1 人，工作量 100%。

2 实验未完成部分说明、实验总结以及其它的意见或建议等。

基本都已完成。

4.4 程序清单

yacc.y	yacc，包含 BNF 语法解析器，yaccler 词法分析器
yacc.tab.c	由 bison 自动生成的函数
yacc.tab.h	由 bison 自动生成的函数
global.h	全局变量以及全局函数
jobs.h	与作业控制相关的函数
execute.c	语义分析时用到的主要函数
cmdenv.conf	命令的环境变量



5 实验心得

5.1 心得：

太痛苦了，本次实验我写了不下 30 个版本，从 0.01 版本一直写到 0.73 版本。

1.我熟悉了 linux 下的编程环境。本来大部分时间都是在 windows 下编程的，虽然大二时上过 linux 的入门课，但是真正用的时候还是不熟悉。

2.我对操作系统中的 **PROCESS** 的概念有了更进一步的了解。进程是构成运行中的操作系统基本逻辑实体。使用进程可以避免不同程序运行时的相互影响。编程的时候，有时候要涉及到进程之间的先后顺序，此时我们一定要采用合适的手段来控制。比如，在我编写的 shell 之中，运行新程序时，必须先要在 shell 中添加相应的作业记录，再运行子进程；否则，就有如下可能：子进程先运行结束，父进程的作业记录还没添加，这时子进程要求父进程删除记录就会出错。此处我走过弯路，后来是采用了全局标志变量，**signal** 和 **kill** 函数来实现的。进程的调度是由操作系统内部实现的，不能简单的假设先建立的先运行，否则会有有一定的概率出错。所以，操作系统中如果涉及到进程之间有操作的先后顺序时，一定要考虑全面，采用合适的方法来实现同步。

3. 我熟悉了操作系统下多线程程序的调试。原本我想在 **eclipse** 下调试，后发现在调试工具下只能运行父进程（可能有子进程的运行，但是我没调出来）。经过研究，我就在程序的不同位置添加了打印的语句。通过判断控制台输出来判断程序的运行情况。正常的话将其注释掉。但是这样又出现了一个新的问题，就是 **tcsetpgrp** 改变终端时可能产生 **SIGTTOU** 或者 **SIGTTIN** 的中断，此时系统默认是暂停程序，这样我就可能错上加错。说到底，还是自己对 **Unix** 操作系统底层的东西不熟悉导致。

4. 我对 shell 本身的功能有了更加深入的了解。**Shell** 是用户与系统内核沟通的中介，它为用户使用操作系统的服务提供了一个命令界面。用户输入的命令通过 shell 解释传给内核执行。虽然我们的 shell 是建立在原来的 shell 之上的，但是，我们还是接触到了一些针对操作系统的编程。例如，进程的创建、进程的通信、输入输出的重定向等。



最后，特别感谢王雷老师精彩的授课，感谢王欢助教辛苦的讲解。最终我高效率高质量的完成的实验的要求。

5.2 建议:

- 1.建议老师多给一些相关的资料，比如初步介绍一下本实验要用到的相关系统调用函数。我是一点一点看那本《Unix 高级环境编程》，看完“作业控制”才感觉豁然开朗。
- 2.实验所附的程序有一些错误，比如上面的 4.2 会议相关记录 中所说的 bug。
- 3.建议给一些测试样例。

一些特殊情况举例（本程序扩展功能（其中管道以及对管道的 **Ctrl+C/Z** 已经在测试和使用说明中体现））:

- 1.内部命令重定向以及管道实现:

history > hist

cat hist

```
user-sh@/home/james>history > hist
outputFile:hist
user-sh@/home/james>cat hist
history
cd
ls
ls -l
user-sh@/home/james>
```

history | cat

```
user-sh@/home/james/Desktop/yalnixshell>history | cat
ls
history | cat
ls
ls -l
find /
fg %1
user-sh@/home/james/Desktop/yalnixshell>
```

- 2.cd 后面没有目录，默认回到系统 HOME 位置



```
////////////////////////////////////
Welcome to shell version 1.0
Author:39061416 Huang Jianyu
Website:www.huangjy.info
Mobile Phone:15210965935
Email:hjyahead@gmail.com
////////////////////////////////////

user-sh@/home/james/Desktop/yalnixshell>cd
user-sh@/home/james>cd ..
user-sh@/home>cd james
user-sh@/home/james>cd Desktop
user-sh@/home/james/Desktop>cd yalnixshell
user-sh@/home/james/Desktop/yalnixshell>cd
user-sh@/home/james>
user-sh@/home/james>history
cd
cd ..
cd james
cd Desktop
cd yalnixshell
cd
user-sh@/home/james>
```

3.后台运行命令需要前台输入（根据《Unix 高级环境编程》中作业控制一节）
cat &: 更改 job 的 status 为 waiting_for_input，并将该进程暂停。（这个实现难度较大，需要考虑定义这种 waiting_for_input 状态）

```
////////////////////////////////////
Welcome to shell version 1.0
Author:39061416 Huang Jianyu
Website:www.huangjy.info
Mobile Phone:15210965935
Email:hjyahead@gmail.com
////////////////////////////////////

user-sh@/home/james/Desktop/yalnixshell>cat &
[1] 7170
user-sh@/home/james/Desktop/yalnixshell>
[1]+  suspended [wants input]    cat &

user-sh@/home/james/Desktop/yalnixshell>jobs

Jobs List:
-----
| job no. |                command | groupid | status |
-----
|      1 |                cat & |    7170 |    W |
-----
user-sh@/home/james/Desktop/yalnixshell>
```