

**实验报告**

**实 验（四）**

题 目 TinyShell

微壳

专 业 计算机类

学　　 号 120L022115

班　　 级 2003007

学 生 王炳轩

指 导 教 师 吴锐

实 验 地 点 线上

实 验 日 期 2022-05-02

**计算学部**

**目 录**

[第1章 实验基本信息 - 4 -](#_Toc102416638)

[1.1 实验目的 - 4 -](#_Toc102416639)

[1.2 实验环境与工具 - 4 -](#_Toc102416640)

[1.2.1 硬件环境 - 4 -](#_Toc102416641)

[1.2.2 软件环境 - 4 -](#_Toc102416642)

[1.2.3 开发工具 - 4 -](#_Toc102416643)

[1.3 实验预习 - 4 -](#_Toc102416644)

[第2章 实验预习 - 5 -](#_Toc102416645)

[2.1 进程的概念、创建和回收方法（5分） - 5 -](#_Toc102416646)

[2.2信号的机制、种类（5分） - 5 -](#_Toc102416647)

[2.3 信号的发送方法、阻塞方法、处理程序的设置方法（5分） - 6 -](#_Toc102416648)

[2.4 什么是shell，简述其功能和处理流程（5分） - 7 -](#_Toc102416649)

[第3章 TinyShell的设计与实现 - 9 -](#_Toc102416650)

[3.1.1 void eval(char \*cmdline)函数（10分） - 9 -](#_Toc102416651)

[3. 1.2 int builtin\_cmd(char \*\*argv)函数（5分） - 10 -](#_Toc102416652)

[3. 1.3 void do\_bgfg(char \*\*argv) 函数（5分） - 11 -](#_Toc102416653)

[3. 1.4 void waitfg(pid\_t pid) 函数（5分） - 12 -](#_Toc102416654)

[3. 1.5 void sigchld\_handler(int sig) 函数（10分） - 13 -](#_Toc102416655)

[第4章 TinyShell测试 - 15 -](#_Toc102416656)

[4.1 测试方法 - 15 -](#_Toc102416657)

[4.2 测试结果评价 - 15 -](#_Toc102416658)

[4.3 自测试结果 - 15 -](#_Toc102416659)

[4.3.1测试用例trace01.txt的输出截图（1分） - 15 -](#_Toc102416660)

[4.3.2测试用例trace02.txt的输出截图（1分） - 15 -](#_Toc102416661)

[4.3.3测试用例trace03.txt的输出截图（1分） - 16 -](#_Toc102416662)

[4.3.4测试用例trace04.txt的输出截图（1分） - 16 -](#_Toc102416663)

[4.3.5测试用例trace05.txt的输出截图（1分） - 16 -](#_Toc102416664)

[4.3.6测试用例trace06.txt的输出截图（1分） - 17 -](#_Toc102416665)

[4.3.7测试用例trace07.txt的输出截图（1分） - 17 -](#_Toc102416666)

[4.3.8测试用例trace08.txt的输出截图（1分） - 17 -](#_Toc102416667)

[4.3.9测试用例trace09.txt的输出截图（1分） - 18 -](#_Toc102416668)

[4.3.10测试用例trace10.txt的输出截图（1分） - 18 -](#_Toc102416669)

[4.3.11测试用例trace11.txt的输出截图（1分） - 18 -](#_Toc102416670)

[4.3.12测试用例trace12.txt的输出截图（1分） - 19 -](#_Toc102416671)

[4.3.13测试用例trace13.txt的输出截图（1分） - 19 -](#_Toc102416672)

[4.3.14测试用例trace14.txt的输出截图（1分） - 21 -](#_Toc102416673)

[4.3.15测试用例trace15.txt的输出截图（1分） - 22 -](#_Toc102416674)

[4.4 自测试评分 - 22 -](#_Toc102416675)

[第4章 总结 - 23 -](#_Toc102416676)

[4.1 请总结本次实验的收获 - 23 -](#_Toc102416677)

[4.2 请给出对本次实验内容的建议 - 23 -](#_Toc102416678)

[参考文献 - 24 -](#_Toc102416679)

# 第1章 实验基本信息

## 1.1 实验目的

理解现代计算机系统进程与并发的基本知识

掌握linux 异常控制流和信号机制的基本原理和相关系统函数

掌握shell的基本原理和实现方法

深入理解Linux信号响应可能导致的并发冲突及解决方法

培养Linux下的软件系统开发与测试能力

## 1.2 实验环境与工具

### 1.2.1 硬件环境

Surface Go 3：x64、Pentium G6500Y @ 1.1GHz、16GB RAM、128GB SSD。

### 1.2.2 软件环境

Windows 11、Windows Subsystem for Linux、Ubuntu 20.04

### 1.2.3 开发工具

Code::Blocks、gcc、vim、edb、gdb

## 1.3 实验预习

上实验课前，必须认真预习实验指导书（PPT或PDF）

了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。

了解进程、作业、信号的基本概念和原理

了解shell的基本原理

熟知进程创建、回收的方法和相关系统函数

熟知信号机制和信号处理相关的系统函数

# 第2章 实验预习

**总分20分**

## 2.1 进程的概念、创建和回收方法（5分）

进程：是一个程序在操作系统中的执行实例，是系统进行资源分配和调度的基本单位，是操作系统结构的基础。在早期面向进程设计的计算机结构中，进程是程序的基本执行实体；在当代面向线程设计的计算机结构中，进程是线程的容器。程序是指令、数据及其组织形式的描述，进程是程序的实体。

进程的创建方法：在shell中，当用户运行新程序时，系统会调用fork复制与当前shell进程相同的一份资源给新程序，然后调用execve装载程序资源。

进程的回收方法：当子进程走完了自己的生命周期后，它会执行exit()系统调用，此时原来进程表中的数据会被该进程的退出码(exit code)、执行时所用的CPU时间等数据所取代，这些数据会一直保留到系统将它传递给它的父进程（必须使用wait或waitpid）为止，此段时间，子进程会成为僵尸进程。之后，子进程由父进程（创建它的进程）回收，当父进程比子进程提前结束时，由系统init进程负责回收。

## 2.2信号的机制、种类（5分）

Linux信号就是一条小消息，它通知进程系统中发生了一个某种类型的事件，类似于异常和中断。从内核发送到（有时是在另一个进程的请求下）一个进程。信号类型是用小整数来标识的：{1-30}。信号中唯一的信息是它的ID和它的到达。

信号可以被进程阻塞（延迟接收处理）、忽略、捕获并自定义处理或执行默认处理行为。

内核为每一个进程维护着两个n位二进制数（n为操作系统的信号数量）：blocks和pendings，这两个二进制数的每一位分别表示对第i个信号的阻塞/发生，因此，信号不能累计，每一种信号最多只能被发送1次直到被接收并清空标志位。

常见的信号种类：



## 2.3 信号的发送方法、阻塞方法、处理程序的设置方法（5分）

**2.3.1 发送方法**

**被动发送信号：**

系统检测到进程发生除零错误、非法内存访问等异常时。

**主动发送信号：**

1、在bash下通过kill命令：

给进程组发送：kill -信号ID -PGID

给单一进程发送：kill -信号ID PID（少一个负号-）

2、通过调用系统函数kill：kill(PID, SIGINT);

3、从键盘发送信号（到前台进程组）：

Ctrl+c:发送SIGINT （终止）

Ctrl+z:发送SIGTSTP （挂起停止）

**2.3.2 阻塞与解除阻塞信号的方法**

隐式阻塞机制

内核默认阻塞与当前正在处理信号类型相同的待处理信号。

显式阻塞和解除阻塞机制

sigprocmask函数及其辅助函数可以明确地阻塞/解除阻塞选定的信号。

辅助函数：

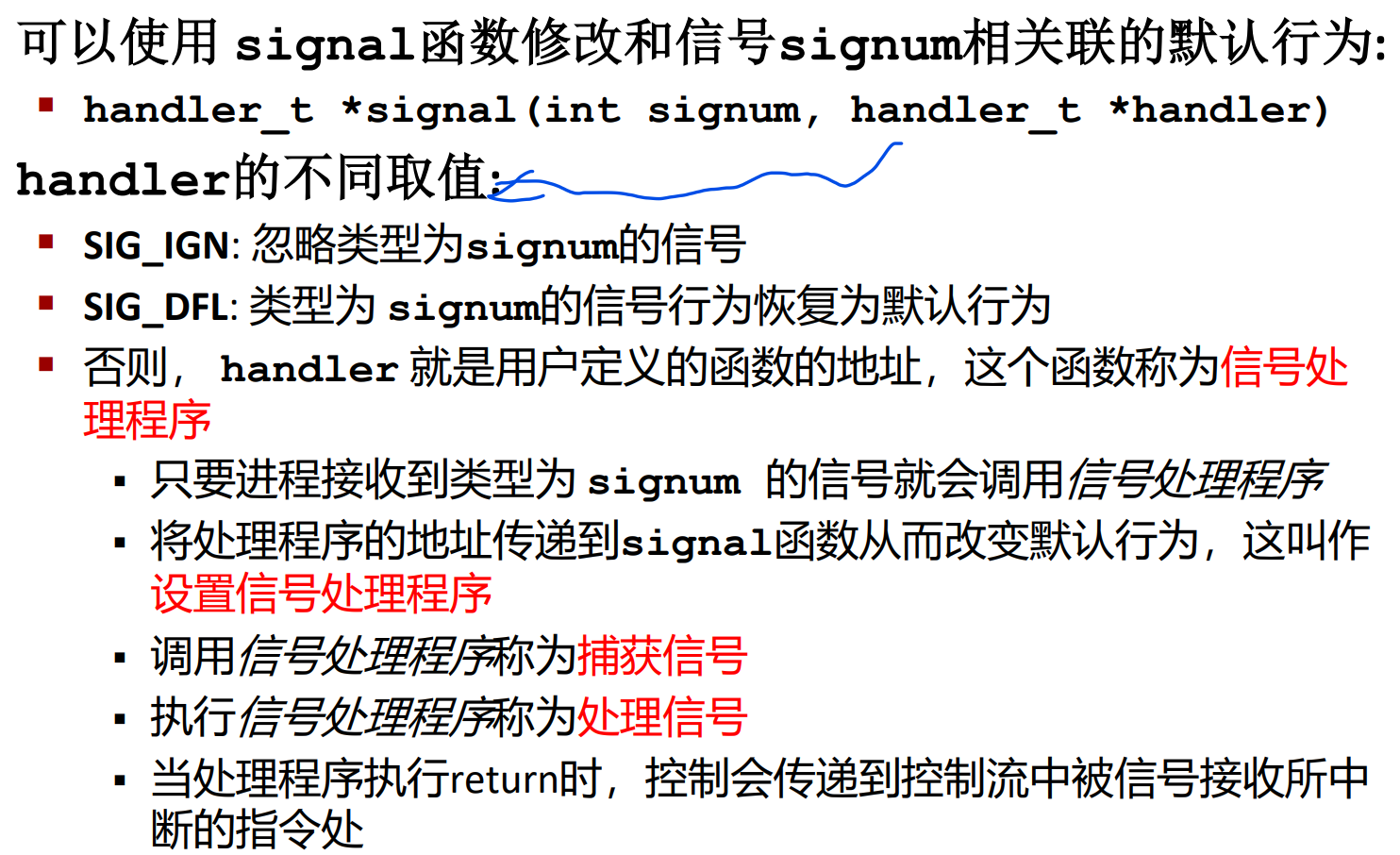
·sigemptyset——初始化set为空集合

·sigflllset——把每个信号都添加到set中

·sigaddset——把指定的信号signum添加到set中

·sigdelset——从set中删除指定的信号

**2.3.3处理程序的设置方法**



## 2.4 什么是shell，简述其功能和处理流程（5分）

Shell俗称壳（用来区别于核），是指“为使用者提供操作界面”的软件（command interpreter，命令解析器）。它类似于DOS下的COMMAND.COM和后来的cmd.exe。它接收用户命令，然后调用相应的应用程序。同时它又是一种程序设计语言。作为命令语言，它交互式解释和执行用户输入的命令或者自动地解释和执行预先设定好的一连串的命令；作为程序设计语言，它定义了各种变量和参数，并提供了许多在高级语言中才具有的控制结构，包括循环和分支。

基本上shell分两大类：

一、图形界面shell（Graphical User Interface shell 即 GUI shell）

例如：应用最为广泛的Windows Explorer（微软的Windows系列操作系统），还有也包括广为人知的Linux shell，其中Linux shell包括X Window Manager（BlackBox和FluxBox），以及功能更强大的CDE、GNOME、KDE、 Xfce。

二、命令行式shell（Command Line Interface shell ，即CLI shell）

例如：sh（Bourne Shell）/csh/tcsh/bash/ksh/zsh/fish等（Unix及类unix）

COMMAND.COM（CP/M系统；MS-DOS、PC-DOS、DR-DOS、FreeDOS等DOS；Windows 9x）

cmd.exe/命令提示符（OS/2、Windows NT、React OS）

Windows PowerShell（支持.NET Framework技术的Windows NT）

shell的功能就是执行用户发出的命令，常用的命令如下：

|  |  |  |
| --- | --- | --- |
| **常用命令** | | |
| **命令** | **备注** | **适用于的命令解释程序** |
| cat [文件名] | 输出文件内容到基本输出（屏幕 or 加>fileName 到另一个文件） | Unix、类unix、Haiku操作系统 |
| cb | 格式化[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81/3969) | Unix、类unix |
| [chmod](https://baike.baidu.com/item/chmod/8818623) | 改变文件的权限 | Unix、类unix |
| [cp](https://baike.baidu.com/item/cp/3906614) | 复制文件 | MULTICS、Unix、类unix、Haiku操作系统 |
| [date](https://baike.baidu.com/item/date/9879981) | 当前的时间和日期 | Unix、类unix、Haiku操作系统 |
| [echo](https://baike.baidu.com/item/echo/35157) $abc | 在变量赋值之后，只需在变量前面加一个$去引用。 | Unix、类unix、Haiku操作系统、Windows Powershell |
| [lint](https://baike.baidu.com/item/lint/2836851) | 语法检查程序 | Unix、类unix、Haiku操作系统 |
| [ls](https://baike.baidu.com/item/ls/2896028) | 列出文件目录 | MULTICS、Unix、类unix、Haiku操作系统、Windows Powershell |
| dir | 列出文件目录 | [Debian GNU/Linux](https://baike.baidu.com/item/Debian%20GNU%2FLinux/2963314)、[Cygwin](https://baike.baidu.com/item/Cygwin/151477" \t "_blank) |
| man | 查询命令 | Unix、类unix、Haiku操作系统 |
| [more](https://baike.baidu.com/item/more/3555120) | 查看文本文件内容 | Unix、类unix、Haiku操作系统 |
| [du](https://baike.baidu.com/item/du/804606) | 查看磁盘空间状况 | Unix、类unix、Haiku操作系统 |
| [uname](https://baike.baidu.com/item/uname/10655076) | 查看当前操作系统名称或版本号 | Unix、类unix |
| [who](https://baike.baidu.com/item/who/4532233)[2] | 你的用户名和终端类型 定义变量 name | Unix、类unix |
| [ps](https://baike.baidu.com/item/ps/8850709) | 查看当前进程状况  =abc? (bash/pdksh) || set name = abc (tcsh) | Unix、类unix、Haiku操作系统 |
| [mv](https://baike.baidu.com/item/mv/2209345) | 改[文件名](https://baike.baidu.com/item/%E6%96%87%E4%BB%B6%E5%90%8D/608450" \t "_blank)/移动文件 | Unix、类unix、Haiku操作系统 |
| [pwd](https://baike.baidu.com/item/pwd/3730074) | 显示目录路径命令 | Unix、类unix |
| exit | [登出](https://baike.baidu.com/item/%E7%99%BB%E5%87%BA/1891322) | Unix、类unix |
| [logout](https://baike.baidu.com/item/logout/496295) | 登出 | Unix、类unix（[FreeBSD](https://baike.baidu.com/item/FreeBSD/413712)操作系统中仅适用于[root](https://baike.baidu.com/item/root/73226)用户） |
| [rm](https://baike.baidu.com/item/rm/2243799) | 删除文件 | Unix、类Unix、Haiku操作系统 |
| echo [文本] | 显示指定文本 | Unix、类unix、Haiku操作系统、DOS、OS/2、Windows、React OS（Windows NT cmd.exe中，[文本]不能为on或off） |
| help [命令] | 列出一个命令的详细信息 | Cygwin、Windows NT |
| [shutdown](https://baike.baidu.com/item/shutdown/10469108) | 关机 | Unix、类unix、Windows NT、React OS |
| [cd](https://baike.baidu.com/item/cd/3516411) | 更改工作目录 | Unix、类unix、Haiku操作系统、OS/2、Windows、React OS；DOS从2.0版起 |
| [chdir](https://baike.baidu.com/item/chdir/10931466) | 更改工作目录 | Unix、类unix、Haiku操作系统、OS/2、Windows、React OS；DOS从2.0版起 |
| [rmdir](https://baike.baidu.com/item/rmdir/6391469) | 删除目录 | Unix、类unix、Haiku操作系统 |
| [mkdir](https://baike.baidu.com/item/mkdir/9500968) | 创建目录 | Unix、类unix、Haiku操作系统、OS/2、Windows、React OS；DOS从2.0版起 |

shell的处理流程包括：解析指令、执行指令序列（调用系统函数）、打印执行结果信息。

# 第3章 TinyShell的设计与实现

**总分45分**

**3.1 设计**

## 3.1.1 void eval(char \*cmdline)函数（10分）

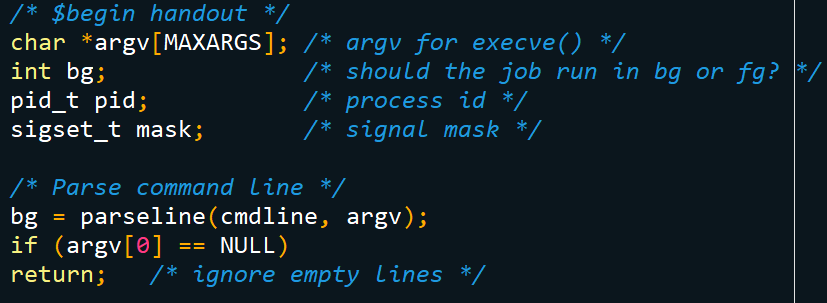
函数功能：执行用户刚刚输入的命令行：如果用户已请求内置命令（退出、作业、bg或fg），则立即执行该命令。否则，派生一个子进程，并在该子进程的上下文中运行该作业。如果作业正在前台运行，等待它终止，然后返回。每个子进程有一个唯一的进程组ID，这样当我们在键盘上键入ctrl-c（ctrl-z）时，我们的后台子进程就不会从内核接收SIGINT（SIGTSTP）。

参 数：char \*cmdline 命令行的字符起始指针。

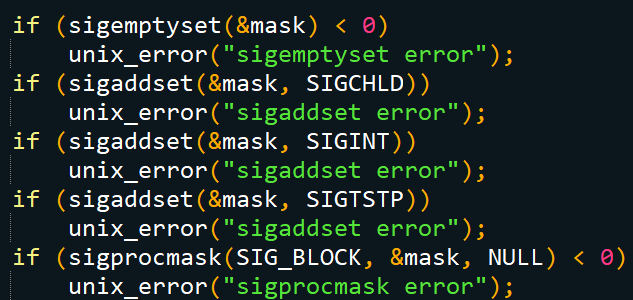
处理流程：初始化、取出指令并判断是否为builtin-command、设置异常信号阻塞、执行fork、对于父进程则等待子进程运行结束回收资源，对于子进程则调用execve执行程序。

要点分析：

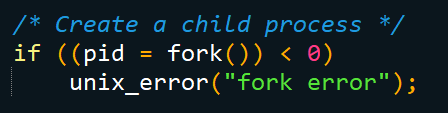
初始化并分析命令：



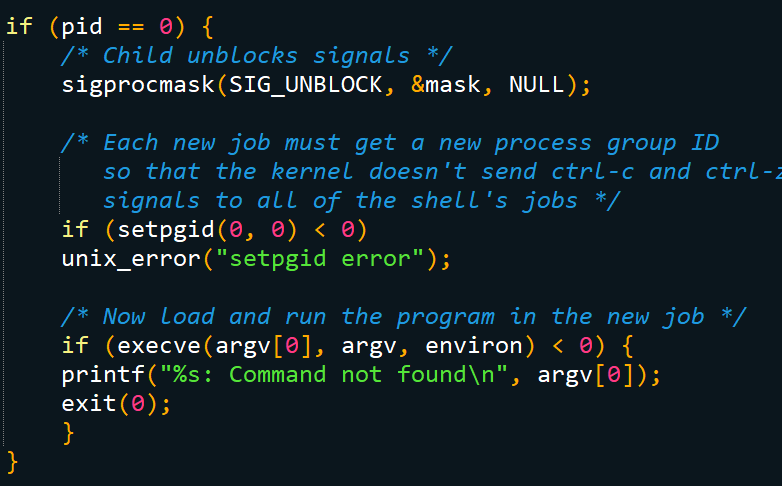
设置信号处理：



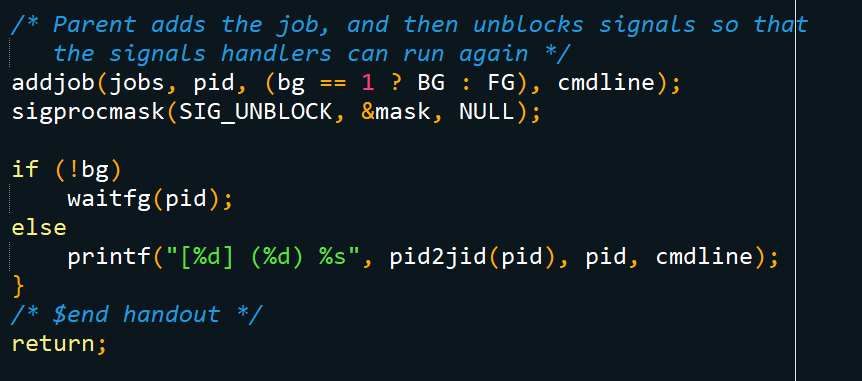
fork：



子进程：



父进程：



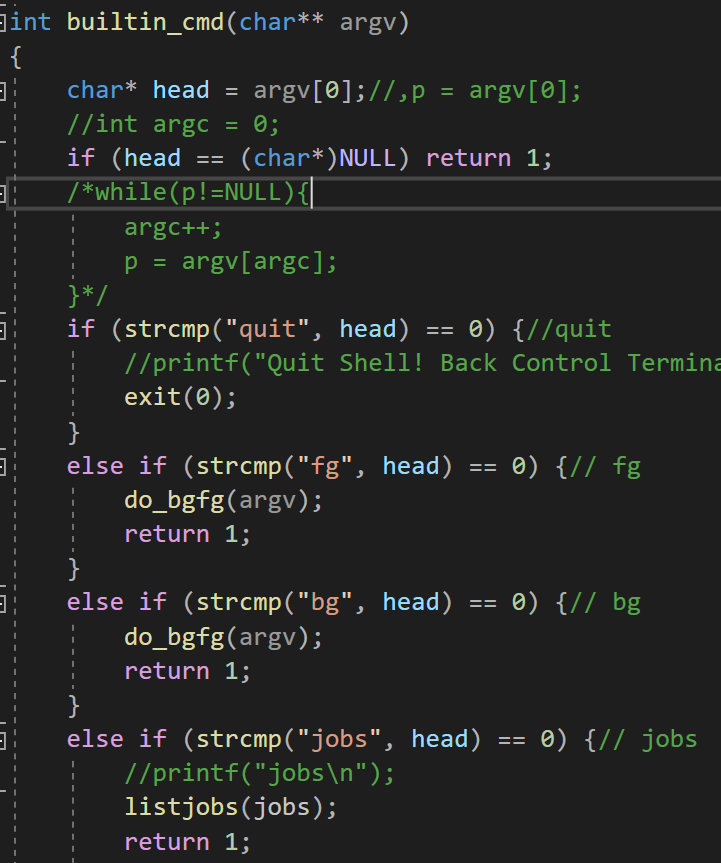
## 3. 1.2 int builtin\_cmd(char \*\*argv)函数（5分）

函数功能：识别并解释内置命令: quit, fg, bg, 和 jobs.

参 数：char \*\*argv 命令行

处理流程：读入命令行，挨个判断调用即可。

要点分析：



## 3. 1.3 void do\_bgfg(char \*\*argv) 函数（5分）

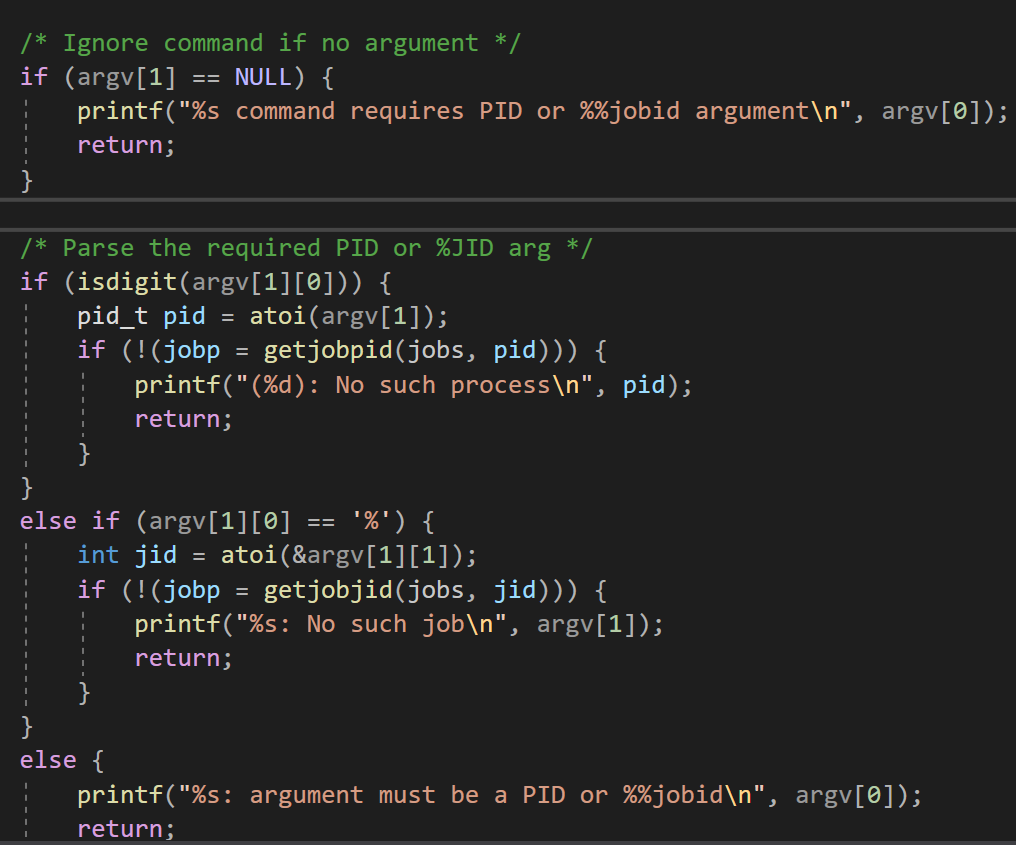
函数功能：**实现内置命令bg 和 fg**

参 数：char\*\* argv

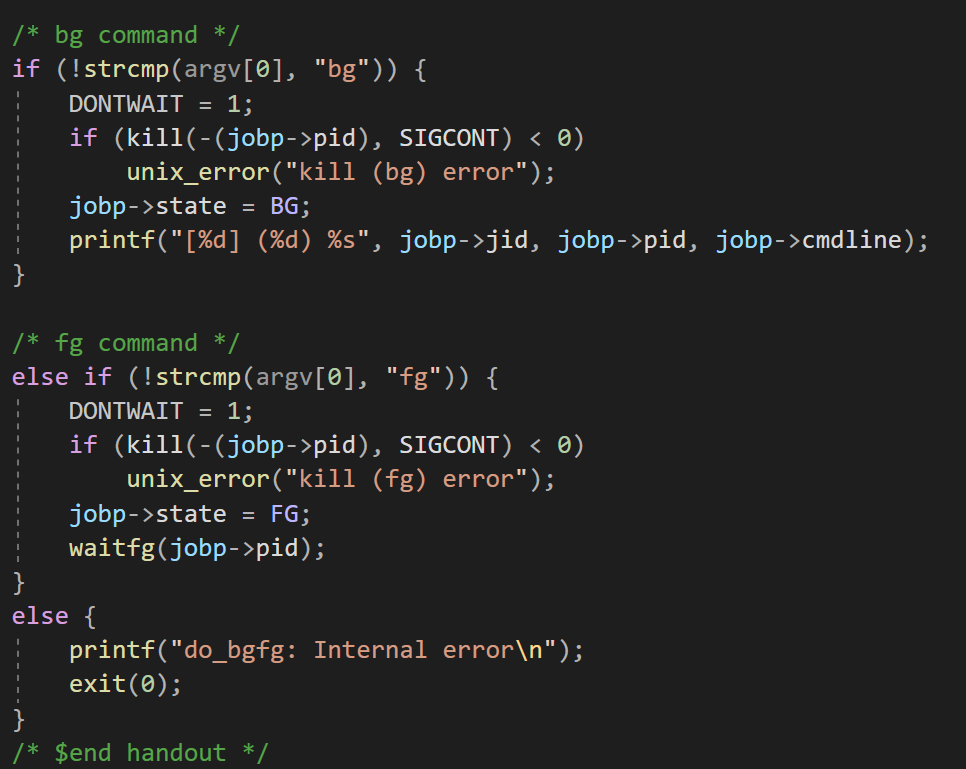
处理流程：先判断是否合法，再执行。

要点分析：

判断合法：



执行：



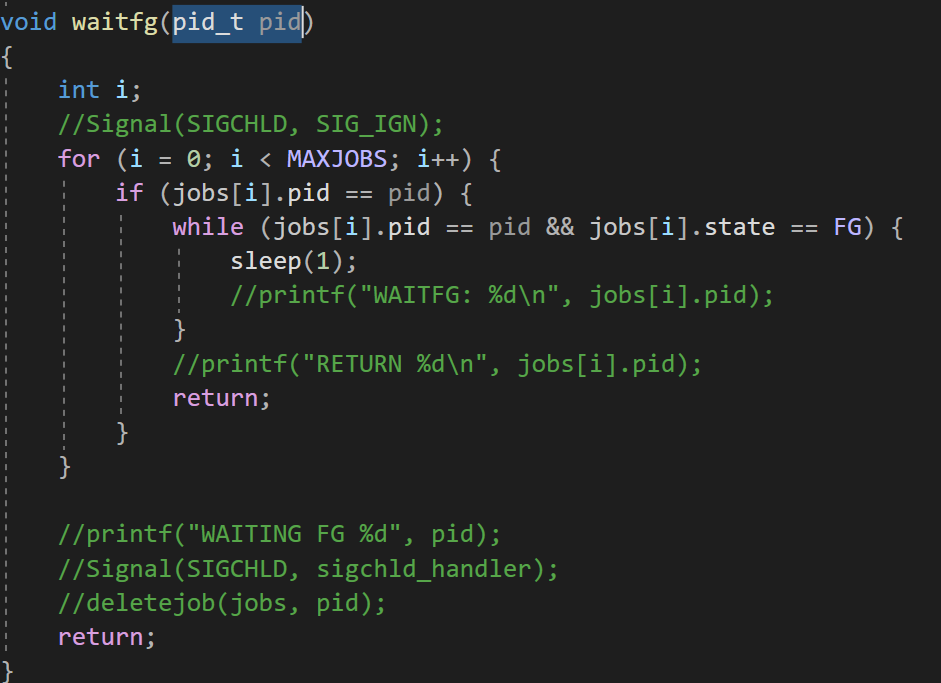
## 3. 1.4 void waitfg(pid\_t pid) 函数（5分）

函数功能：**等待一个前台作业结束**

参 数：pid\_t pid

处理流程：等待PID为pid的前台进程作业结束。

要点分析：



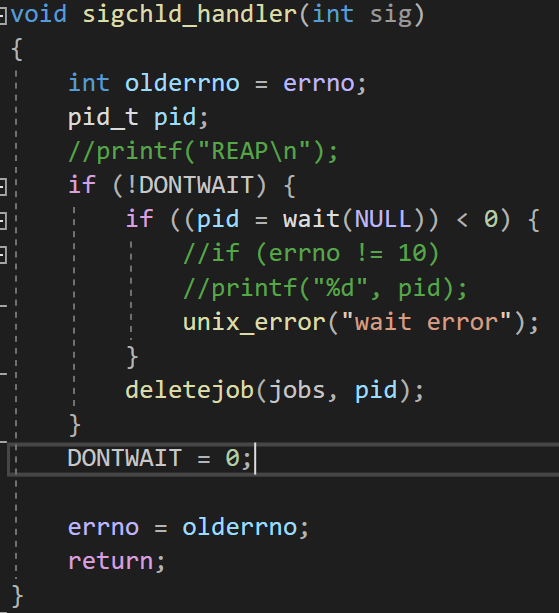
## 3. 1.5 void sigchld\_handler(int sig) 函数（10分）

函数功能：**捕获SIGCHILD信号**

参 数：int sig

处理流程：子进程挂起、继续、终止时收到信号，使用DONTWAIT判断是否为终止，然后执行回收。

要点分析：



**3.2 程序实现（tsh.c的全部内容）（10分）**

**重点检查代码风格：**

1. **用较好的代码注释说明——5分**

已注释

1. **检查每个系统调用的返回值——5分**

已完成

# 第4章 TinyShell测试

**总分15分**

## 4.1 测试方法

针对tsh和参考shell程序tshref，完成测试项目4.1-4.15的对比测试，并将测试结果截图或者通过重定向保存到文本文件(例如：./sdriver.pl -t trace01.txt -s ./tsh -a "-p" > tshresult01.txt)。

## 4.2 测试结果评价

tsh与tshref的输出在一下两个方面可以不同：

（1）PID

（2）测试文件trace11.txt, trace12.txt和trace13.txt中的/bin/ps命令，每次运行的输出都会不同，但每个mysplit进程的运行状态应该相同。

除了上述两方面允许的差异，tsh与tshref的输出相同则判为正确，如不同则给出原因分析。

## 4.3 自测试结果

### 4.3.1测试用例trace01.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.2测试用例trace02.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.3测试用例trace03.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.4测试用例trace04.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.5测试用例trace05.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.6测试用例trace06.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.7测试用例trace07.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.8测试用例trace08.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.9测试用例trace09.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.10测试用例trace10.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.11测试用例trace11.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.12测试用例trace12.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
|  | |  |
| 测试结论 | 相同 | |

### 4.3.13测试用例trace13.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tshref测试结果 | | tsh测试结果 |
| noname:shell>make rtest13  ./sdriver.pl -t trace13.txt -s ./tshref -a "-p"  #  # trace13.txt - Restart every stopped process in process group  #  tsh> ./mysplit 4  Job [1] (30822) stopped by signal 20  tsh> jobs  [1] (30822) Stopped ./mysplit 4  tsh> /bin/ps a  PID TTY STAT TIME COMMAND  1990 tty2 Ssl+ 0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME\_SHELL\_SESSION\_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu  1992 tty2 Sl+ 1:43 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3  2413 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu  30104 pts/0 Ss+ 0:00 bash  30435 pts/1 Ss 0:00 bash  30817 pts/1 S+ 0:00 make rtest13  30818 pts/1 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tshref -a "-p"  30819 pts/1 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tshref -a -p  30820 pts/1 S+ 0:00 ./tshref -p  30822 pts/1 T 0:00 ./mysplit 4  30823 pts/1 T 0:00 ./mysplit 4  30826 pts/1 R 0:00 /bin/ps a  tsh> fg %1  tsh> /bin/ps a  PID TTY STAT TIME COMMAND  1990 tty2 Ssl+ 0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME\_SHELL\_SESSION\_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu  1992 tty2 Sl+ 1:43 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3  2413 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu  30104 pts/0 Ss+ 0:00 bash  30435 pts/1 Ss 0:00 bash  30817 pts/1 S+ 0:00 make rtest13  30818 pts/1 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tshref -a "-p"  30819 pts/1 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tshref -a -p  30820 pts/1 S+ 0:00 ./tshref -p  30829 pts/1 R 0:00 /bin/ps a | | noname:shell>make test13  ./sdriver.pl -t trace13.txt -s ./tsh -a "-p"  #  # trace13.txt - Restart every stopped process in process group  #  tsh> ./mysplit 4  Job [1] (30809) stopped by signal 20  tsh> jobs  [1] (30809) Stopped ./mysplit 4  tsh> /bin/ps a  PID TTY STAT TIME COMMAND  1990 tty2 Ssl+ 0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME\_SHELL\_SESSION\_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu  1992 tty2 Sl+ 1:43 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3  2413 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu  30104 pts/0 Ss 0:00 bash  30435 pts/1 Ss+ 0:00 bash  30804 pts/0 S+ 0:00 make test13  30805 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tsh -a "-p"  30806 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tsh -a -p  30807 pts/0 S 0:00 ./tsh -p  30809 pts/0 T 0:00 ./mysplit 4  30810 pts/0 T 0:00 ./mysplit 4  30813 pts/0 R 0:00 /bin/ps a  tsh> fg %1  tsh> /bin/ps a  PID TTY STAT TIME COMMAND  1990 tty2 Ssl+ 0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME\_SHELL\_SESSION\_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu  1992 tty2 Sl+ 1:43 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3  2413 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu  30104 pts/0 Ss 0:00 bash  30435 pts/1 Ss+ 0:00 bash  30804 pts/0 S+ 0:00 make test13  30805 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tsh -a "-p"  30806 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tsh -a -p  30807 pts/0 S 0:00 ./tsh -p  30816 pts/0 R 0:00 /bin/ps a |
| 测试结论 | 相同 | |

### 4.3.14测试用例trace14.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
| noname:shell>make test14  ./sdriver.pl -t trace14.txt -s ./tsh -a "-p"  #  # trace14.txt - Simple error handling  #  tsh> ./bogus  ./bogus: Command not found  tsh> ./myspin 4 &  [1] (30838) ./myspin 4 &  tsh> fg  fg command requires PID or %jobid argument  tsh> bg  bg command requires PID or %jobid argument  tsh> fg a  fg: argument must be a PID or %jobid  tsh> bg a  bg: argument must be a PID or %jobid  tsh> fg 9999999  (9999999): No such process  tsh> bg 9999999  (9999999): No such process  tsh> fg %2  %2: No such job  tsh> fg %1  Job [1] (30838) stopped by signal 20  tsh> bg %2  %2: No such job  tsh> bg %1  [1] (30838) ./myspin 4 &  tsh> jobs  [1] (30838) Running ./myspin 4 & | | noname:shell>make rtest14  ./sdriver.pl -t trace14.txt -s ./tshref -a "-p"  #  # trace14.txt - Simple error handling  #  tsh> ./bogus  ./bogus: Command not found  tsh> ./myspin 4 &  [1] (30857) ./myspin 4 &  tsh> fg  fg command requires PID or %jobid argument  tsh> bg  bg command requires PID or %jobid argument  tsh> fg a  fg: argument must be a PID or %jobid  tsh> bg a  bg: argument must be a PID or %jobid  tsh> fg 9999999  (9999999): No such process  tsh> bg 9999999  (9999999): No such process  tsh> fg %2  %2: No such job  tsh> fg %1  Job [1] (30857) stopped by signal 20  tsh> bg %2  %2: No such job  tsh> bg %1  [1] (30857) ./myspin 4 &  tsh> jobs  [1] (30857) Running ./myspin 4 & |
| 测试结论 | 相同 | |

### 4.3.15测试用例trace15.txt的输出截图（1分）

|  |  |  |
| --- | --- | --- |
| tsh测试结果 | | tshref测试结果 |
| noname:shell>make test15  ./sdriver.pl -t trace15.txt -s ./tsh -a "-p"  #  # trace15.txt - Putting it all together  #  tsh> ./bogus  ./bogus: Command not found  tsh> ./myspin 10  Job [1] (30876) terminated by signal 2  tsh> ./myspin 3 &  [1] (30878) ./myspin 3 &  tsh> ./myspin 4 &  [2] (30880) ./myspin 4 &  tsh> jobs  [1] (30878) Running ./myspin 3 &  [2] (30880) Running ./myspin 4 &  tsh> fg %1  Job [1] (30878) stopped by signal 20  tsh> jobs  [1] (30878) Stopped ./myspin 3 &  [2] (30880) Running ./myspin 4 &  tsh> bg %3  %3: No such job  tsh> bg %1  [1] (30878) ./myspin 3 &  tsh> jobs  [1] (30878) Running ./myspin 3 &  [2] (30880) Running ./myspin 4 &  tsh> fg %1  tsh> quit | | noname:shell>make rtest15  ./sdriver.pl -t trace15.txt -s ./tshref -a "-p"  #  # trace15.txt - Putting it all together  #  tsh> ./bogus  ./bogus: Command not found  tsh> ./myspin 10  Job [1] (30890) terminated by signal 2  tsh> ./myspin 3 &  [1] (30897) ./myspin 3 &  tsh> ./myspin 4 &  [2] (30899) ./myspin 4 &  tsh> jobs  [1] (30897) Running ./myspin 3 &  [2] (30899) Running ./myspin 4 &  tsh> fg %1  Job [1] (30897) stopped by signal 20  tsh> jobs  [1] (30897) Stopped ./myspin 3 &  [2] (30899) Running ./myspin 4 &  tsh> bg %3  %3: No such job  tsh> bg %1  [1] (30897) ./myspin 3 &  tsh> jobs  [1] (30897) Running ./myspin 3 &  [2] (30899) Running ./myspin 4 &  tsh> fg %1  tsh> quit |
| 测试结论 | 相同 | |

## 4.4 自测试评分

根据节4.3的自测试结果，程序的测试评分为： 满分15分 。

# 第4章 总结

## 4.1 请总结本次实验的收获

学会了使用系统调用fork、execve等。

学会了一些信号与信号处理、捕获信号的方法。

学会使用driver构建一个tinyshell。

## 4.2 请给出对本次实验内容的建议

无

注：本章为酌情加分项。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

