

哈尔滨工业大学

实验报告

实验（四）

题 目 TinyShell
微壳

专 业 计算学部

学 号 120L020701

班 级 2003005

学 生 董琦

指 导 教 师 吴锐

实 验 地 点 宿舍

实 验 日 期 2022/4/29

计算学部

目 录

第 1 章 实验基本信息	- 4 -
1.1 实验目的	- 4 -
1.2 实验环境与工具	- 4 -
1.2.1 硬件环境	- 4 -
1.2.2 软件环境	- 4 -
1.2.3 开发工具	- 4 -
第 2 章 实验预习	- 5 -
2.1 进程的概念、创建和回收方法 (5 分)	- 5 -
2.2 信号的机制、种类 (5 分)	- 5 -
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 -
2.3.1 发送方法:	- 6 -
2.3.2 阻塞方法:	- 6 -
2.3.3 处理程序的设置方法:	- 6 -
2.4 什么是 SHELL, 简述其功能和处理流程 (5 分)	- 7 -
第 3 章 TINY SHELL 的设计与实现	- 8 -
3.1 设计	- 8 -
3.1.1 void eval(char *cmdline) 函数 (10 分)	- 8 -
3.1.2 int builtin_cmd(char **argv) 函数 (5 分)	- 10 -
3.1.3 void do_bgfg(char **argv) 函数 (5 分)	- 10 -
3.1.4 void waitfg(pid_t pid) 函数 (5 分)	- 12 -
3.1.5 void sigchld_handler(int sig) 函数 (10 分)	- 13 -
3.2 程序实现 (TSH.C 的全部内容) (10 分)	- 14 -
第 4 章 TINY SHELL 测试	- 15 -
4.1 测试方法	- 15 -
4.2 测试结果评价	- 15 -
4.3 自测试结果	- 15 -
4.3.1 测试用例 trace01.txt 的输出截图 (1 分)	- 15 -
4.3.2 测试用例 trace02.txt 的输出截图 (1 分)	- 15 -
4.3.3 测试用例 trace03.txt 的输出截图 (1 分)	- 16 -
4.3.4 测试用例 trace04.txt 的输出截图 (1 分)	- 16 -
4.3.5 测试用例 trace05.txt 的输出截图 (1 分)	- 16 -
4.3.6 测试用例 trace06.txt 的输出截图 (1 分)	- 17 -
4.3.7 测试用例 trace07.txt 的输出截图 (1 分)	- 17 -
4.3.8 测试用例 trace08.txt 的输出截图 (1 分)	- 17 -
4.3.9 测试用例 trace09.txt 的输出截图 (1 分)	- 18 -

4.3.10 测试用例 <i>trace10.txt</i> 的输出截图 (1 分)	- 18 -
4.3.11 测试用例 <i>trace11.txt</i> 的输出截图 (1 分)	- 18 -
4.3.12 测试用例 <i>trace12.txt</i> 的输出截图 (1 分)	- 19 -
4.3.13 测试用例 <i>trace13.txt</i> 的输出截图 (1 分)	- 19 -
4.3.14 测试用例 <i>trace14.txt</i> 的输出截图 (1 分)	- 20 -
4.3.15 测试用例 <i>trace15.txt</i> 的输出截图 (1 分)	- 20 -
4.3.16 测试用例 <i>trace16.txt</i> 的输出截图 (1 分)	- 21 -
4.4 自测试评分	- 21 -
第 5 章 总结	- 22 -
5.1 请总结本次实验的收获	- 22 -
5.2 请给出对本次实验内容的建议	- 22 -
参考文献	- 23 -

第 1 章 实验基本信息

1.1 实验目的

- 1.理解现代计算机系统进程与并发的基本知识
- 2.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
- 3.掌握 shell 的基本原理和实现方法
- 4.深入理解 Linux 信号响应可能导致的并发冲突及解决方法
- 5.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

处理器 Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz 8 核
机带 RAM 16.0 GB (15.8 GB 可用)
系统类型 64 位操作系统, 基于 x64 的处理器

1.2.2 软件环境

Windows:

版本 Windows 11 家庭中文版

版本 21H2

安装日期 2022/2/4

操作系统版本 22000.556

体验 Windows 功能体验包 1000.22000.556.0

Ubuntu:

版本 Ubuntu 20.04.3 LTS

类型 64 位

1.2.3 开发工具

Windows: Visual Studio 2019

Ubuntu: VScode gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04)

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）

概念：进程的经典定义是一个执行中程序的实例。进程提供给应用程序的关键抽象：①:一个独立的逻辑控制流；②:一个私有的地址空间。

创建方法：父进程通过 `fork` 函数创建一个新的运行的子进程。其中，`init` 进程是所有用户进程的祖先进程。注意：`execve` 并没有创建新进程。

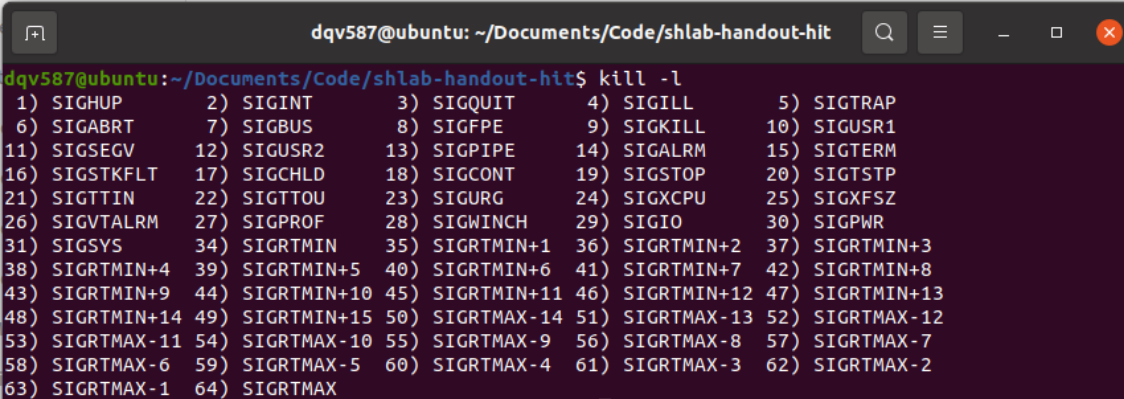
回收方法：当进程终止后，保持在一种已经终止的状态，直到被父进程回收。若父进程终止了，内核则安排 `init` 进程成为孤儿进程的养父母，用于回收这些进程。使用的函数：`wait` 和 `waitpid`。若子进程没有被回收，则它的 PCB 还保留在系统内核，占用着内核资源和进程号。

2.2 信号的机制、种类（5 分）

机制：一个信号就是一条小消息，他通知进程系统中发生了一个某种类型的事件。Linux 信号允许进程和内核中断其他进程。

每种信号类型都对应于某种系统事件。底层的硬件异常是由内核异常处理程序处理的，正常情况下，对用户进程是不可见的。信号提供了一种机制，通知用户进程发生了这些异常。

种类：



```
dqv587@ubuntu: ~/Documents/Code/shlab-handout-hit
dqv587@ubuntu:~/Documents/Code/shlab-handout-hit$ kill -l
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS      8) SIGFPE      9) SIGKILL     10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM     15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD    18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN    22) SIGTTOU    23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF    28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS     34) SIGRTMIN   35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5 60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

2.3 信号的发送方法、阻塞方法、处理程序的设置方法（5 分）

传送一个信号到目的进程是由两个不同步骤组成的。①发送信号。内核通过更新目的进程上下文中的某个状态，发送一个信号到目的进程。一个进程也可以发信号给自己。②接受信号。当目的进程被内核强迫以某种方式对信号的发送做出反应时，它就接收了信号。进程可以忽略这个信号，终止或者通过执行一个称为信号处理程序的用户层函数捕获这个信号。

2.3.1 发送方法：

①用/bin/kill 程序发送信号。/bin/kill 程序可以向另外的进程发送任意的信号。

②从键盘发送信号。例如 Ctrl+C 是发送一个 SIGINT 信号到前台进程组中的每个进程。

③用 kill 函数发送信号。进程通过调用 kill 函数发送信号给进程。

④用 alarm 函数发送信号。进程可以通过调用 alarm 函数向它自己发送 SIGALRM 信号。

2.3.2 阻塞方法：

隐式阻塞机制：内核默认阻塞任何当前处理程序正在处理信号类型的待处理的信号。

显式阻塞机制：应用程序可以使用 sigprocmask 函数和它的辅助函数，明确地阻塞和接触阻塞选定的信号。Sigprocmask 函数改变当前阻塞的信号集合（即改变 blocked 位向量）。

2.3.3 处理程序的设置方法：

```
typedef void (*sighandler_t)(int);  
sighandler_t signal(int signum, sighandler_t handler);
```

利用 signal 函数设置。第一个参数为要修改的信号，第二参数为修改后对应信号处理的行为。若 handler 是 SIG_IGN (ignore), 那么忽略类型为 signum 的信号；若 handler 是 SIG_DFL (default), 那么恢复为默认行为；否则，handler 就是用户定义的函数的地址，称为信号处理程序，只要进程接受到一个类型为 signum 的信号，就会调用这个程序。

例外：SIGSTOP 和 SIGKILL 的默认行为不能修改。

2.4 什么是 shell，简述其功能和处理流程（5 分）

Shell 是一个交互型的应用级程序，它代表用户运行其他程序。Shell 执行一系列的读/求值步骤，然后终止。读步骤读取来自用户的一个命令行。求值步骤解析命令行，并代表用户运行程序。

Shell 利用 `fork` 和 `execve` 运行程序。Shell 解析用户输入的命令行后，若指令是要求运行一个程序，则通过 `fork` 开辟一个新的线程，然后通过 `execve` 函数来运行这个程序。

并且，shell 也负责回收所有的子进程。若命令要求在前台运行程序，那么 shell 程序显式地调用 `waitpid` 函数来等待子进程停止；若命令要求在后台运行程序(即最后一个参数为`&`)，那么 shell 程序通过 `SIGCHLD` 信号处理程序来回收终止的子进程，信号处理程序中依然使用 `waitpid` 这个函数。由于信号处理程序同主程序是并发执行的，所以 shell 可以继续执行其他作业，而不需要等待后台进程停止。所以，一个 shell 程序可以有一个前台作业和多个后台作业。

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：

识别用户输入的命令行，解析指令并执行。若为内置指令则立即执行，若为运行一个新程序则 fork 出一个子进程，然后在子进程中 exevece 这个程序。

参 数：char* cmdline,即用户在 shell 中键入的命令行。

处理流程：

首先调用 parseline 函数将输入的 cmdline 分割为参数，存放于 argv 数组当中，然后根据 argv 数组中参数决定接下来的动作。同时在此步也确定了作业是否在后台执行。

若 argv[0]为 null，则说明输入指令为空，那么则忽略该输入，立即 return；若是程序内置指令，如 quit, jobs, bg 或 fg，则立即执行；若是执行新程序的指令，那么调用 fork 开辟一个子进程，然后在子进程中调用 exevece 来执行程序，并且在父进程中将该任务添加到 jobs 中。

要点分析：

为了防止信号处理程序同 addjob 竞争，导致 deletejob 先于 addjob 之前发生，从而导致作业列表中出现一个不正确的条目。所以需要在调用 fork 之前，阻塞信号，然后在 addjob 之后取消阻塞这些信号，这样保证了在子进程被添加到作业列表之后回收该子进程。另外，由于在子进程中同样堵塞了这些信号，所以在调用 exevece 之前，要接触子进程中阻塞的信号。

被阻塞的信号有：SIGCHLD, SIGINT, SIGTSTP.

```
if (sigemptyset(&mask) < 0)
    unix_error("sigemptyset error");
if (sigaddset(&mask, SIGCHLD))
    unix_error("sigaddset error");
if (sigaddset(&mask, SIGINT))
    unix_error("sigaddset error");
if (sigaddset(&mask, SIGTSTP))
    unix_error("sigaddset error");
```



```

if (sigprocmask(SIG_BLOCK, &mask, NULL) < 0)
    unix_error("sigprocmask error");

/* Create a child process */
if ((pid = fork()) < 0)
    unix_error("fork error");

/* ...

if (pid == 0)
{
    /* Child unblocks signals */
    sigprocmask(SIG_UNBLOCK, &mask, NULL);

    /* Each new job must get a new process group ID
       so that the kernel doesn't send ctrl-c and ctrl-z
       signals to all of the shell's jobs */
    if (setpgid(0, 0) < 0)
        unix_error("setpgid error");

    /* Now load and run the program in the new job */
    if (execve(argv[0], argv, environ) < 0)
    {
        printf("%s: Command not found\n", argv[0]);
        exit(0);
    }
}

/* ...

/* Parent adds the job, and then unblocks signals so that
   the signals handlers can run again */
addjob(jobs, pid, (bg == 1 ? BG : FG), cmdline);
sigprocmask(SIG_UNBLOCK, &mask, NULL);

```

另外，读 `parseline` 源代码后可知，命令行参数的分隔符可以为空格，或者为'。

若命令为前台执行（即最后一个参数不为'&'），那么 `eval` 程序需要显示地等待该子进程停止并且回收。

```

if (!bg)
    waitfg(pid);
else
    printf("[%d] (%d) %s", pid2jid(pid), pid, cmdline);

```

此外，子进程需要重新设置 `pgid`，以为了防止对 `shell` 主进程发送的信号影响到各个子进程。

```

/* Each new job must get a new process group ID
   so that the kernel doesn't send ctrl-c and ctrl-z
   signals to all of the shell's jobs */
if (setpgid(0, 0) < 0)
    unix_error("setpgid error");

```

3.1.2 int builtin_cmd(char **argv)函数（5分）

函数功能：

识别用户输入的指令是否为 shell 内置的指令。若是则立即执行，若不是则返回 0。本实验中内置指令有 fg,bg,quit,jobs 四个。其中 fg,bg 功能由 do_bgfg 执行；quit 指令是 shell 进程立即停止；jobs 是列出当前未结束的子进程及其状态。

参 数：

Parseline 得到的 argv 数组。

处理流程：

通过简单的 if-else 逻辑判断即可，调用 strcmp 判断第一个参数是否与内置指令相等。

若为 bg,fg 指令，则交给 do_bgfg 执行；若为 quit 指令，调用 exit 系统函数退出即可；若为 jobs，则调用 listjobs 输出当前的任务列表；若不为这几个指令，则返回 0。

```
int builtin_cmd(char **argv)
{
    if(!strcmp(argv[0],"bg")||!strcmp(argv[0],"fg"))
    {
        do_bgfg(argv);
        return 1;
    }
    else if(!strcmp(argv[0],"quit"))
    {
        exit(0);
        return 1;
    }
    else if(!strcmp(argv[0],"jobs"))
    {
        listjobs(jobs);
        return 1;
    }
    return 0; /* not a builtin command */
}
```

要点分析：

判断字符串相等需要调用 strcmp 函数，需注意若相等则返回 0。

3.1.3 void do_bgfg(char **argv) 函数（5分）

函数功能：

被 builtin_cmd 调用，执行 bg,fg 指令。需要输入参数，若为单独数字，则是指定进程的 pid，若为 %+ 数字，则是指定进程的 jid。其作用是向停止的进程发送 SIGCONT 信号，若为 fg 则令该进程在前台执行，即调用 waitfg 显式等待该进程停止，若为 bg 则令该进程在后台执行，同时修改任务的状态。

参 数:

依然是 parseline 得到的参数数组。

处理流程:

首先是检查参数是否输入正确，格式是 `bg\fg +pid\%jid`。若格式错误则输出必要的提示信息，若正确则在任务列表中查找对应任务。若查找失败则输出提示信息，查找成功则得到对应任务的结构体。

```
if (argv[1] == NULL)
{
    printf("%s command requires PID or %%jobid argument\n", argv[0]);
    return;
}

/* Parse the required PID or %JID arg */
if (isdigit(argv[1][0]))
{
    pid_t pid = atoi(argv[1]);
    if (!(jobp = getjobpid(jobs, pid)))
    {
        printf("(%d): No such process\n", pid);
        return;
    }
}
else if (argv[1][0] == '%')
{
    int jid = atoi(&argv[1][1]);
    if (!(jobp = getjobjid(jobs, jid)))
    {
        printf("%s: No such job\n", argv[1]);
        return;
    }
}
else
{
    printf("%s: argument must be a PID or %%jobid\n", argv[0]);
    return;
}
```

得到任务的结构体后，首先向该进程发送 SIGCONT 信号，然后根据 `bg`, `fg` 指令，决定是在前台显式等待其终止，还是输出一条任务对应信息后在后台运行。

```
if (!strcmp(argv[0], "bg"))
{
    if (kill(-(jobp->pid), SIGCONT) < 0)
        unix_error("kill (bg) error");
    jobp->state = BG;
    printf("[%d] (%d) %s", jobp->jid, jobp->pid, jobp->cmdline);
}

/* fg command */
else if (!strcmp(argv[0], "fg"))
{
    if (kill(-(jobp->pid), SIGCONT) < 0)
        unix_error("kill (fg) error");
    jobp->state = FG;
    waitfg(jobp->pid);
}
```

要点分析:

这里首先要检验参数的正确格式，然后根据输入参数查找对应任务，这部分只涉及字符串比对和调用函数，较为简单。

然后发送信号需要用到 kill 函数，若 pid 为负值则向该进程组中所有进程发送同样的信号，这里也能体现出为每一个任务设定独特的 pgid 的意义——防止信号影响到其他任务。

最后根据 fg,bg 指令决定是否调用 waitfg 来等待任务结束。

3.1.4 void waitfg(pid_t pid) 函数（5 分）

函数功能：

用于显式地等待唯一的前台任务终止或者停止。在这个函数里我们不进行信号处理。

参 数：

要等待任务进程的 pid。

处理流程：

首先使用一个无限循环来等待进程终止。然后在循环中查询当前任务列表中是否仍有前台任务（这里我们需要假设“任意时刻只能有一个前台任务”限制条件的正确性）。若发现没有前台任务了，那么解除阻塞，接受下一个指令。

```
void waitfg(pid_t pid)
{
    sigset_t mask_all, prev_all, empty_set;
    sigfillset(&mask_all);
    if (sigemptyset(&empty_set) < 0)
        unix_error("sigemptyset error");
    sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
    while(1){
        sigsuspend(&empty_set);
        if(fgpid(jobs) == 0)
            break;
    }
    sigprocmask(SIG_SETMASK, &prev_all, NULL);
    return;
}
```

要点分析：

这里要点有很多哈，也是花时间想很久的地方。

1. 首先，由于要一直阻塞进程，需要一个 while 循环。但是这样很浪费处理器资源，所以可以在循环中使用 pause 或 sleep 函数。但是使用 pause 的话，会出现竞争情况：在 while 后 pause 前收到信号，这样收到信号后再调用 pause，会导致进程永远睡眠；而使用 sleep 的话，我个人测试是很慢，因为必须以 1s 为单位来休眠。因此这里我借鉴了书上的指导，使用了 sigsuspend 函数，这个函数保证了只有在 sigsuspend 函数运行期间接受信号并处理，之后我们检查这个信号是否将前台任务终止或者停止。

2. 然后，为了保证只在 sigsuspend 期间进行信号处理，保证过程的时序正确，我在循环前阻塞了所有信号，只在 sigsuspend 期间解除。这样确保了执行顺序一定是 handler->fgpid。

3.另外，刚开始我写这里的时候，是看了书上较前面的例子，那会没讲信号，于是直接在 `waitfg` 中调用 `waitpid` 等待指定进程停止。但是写完发现，前面的几个测试可以通过，到第八个就不正常了，然后分析半天绕晕了。后来看书的后面，发现这部分已经讲了。

书上方法是用一个全局变量标志前台进程是否停止，这里我们借鉴其思路，用查看是否有前台任务的方法来检查前台任务的状态。然后在 `waitfg` 中不进行子进程的回收，这些内容全部交给信号处理程序 `sigchld_handler` 去做，这样分工明确，代码更有条理性。（至此才发觉所有注意的点 PPT 里都已经暗示过了）

3.1.5 void sigchld_handler(int sig) 函数（10 分）

函数功能：

这个函数用于回收处理子进程，同时还可以监视某个进程被停止信号停止。

当 `sigchld_handler` 被来自子进程的信号激活后，它进行子进程的回收处理，同时还查看导致子进程发出信号的原因，并作出相应输出和动作。

参 数：

`sig` 这个参数书上这里都没有介绍，然后我自己动手 `print` 了一下，这个参数就是这个信号处理函数要处理的信号编号，在函数体中没有用到。

处理流程：

我们使用一个 `while` 循环，来处理终止或停止的子进程。为了防止被其他信号中断，导致未知的错误，我在循环期间阻塞了所有信号。

这里用 `status` 来存放子进程的返回状态，然后用这个 `status` 来判断子进程是正常终止、被信号终止或被信号停止，然后再根据不同情况做不同的动作：若子进程是正常停止，则只要默默地在 `joblist` 中删除就好；若子进程是由于 `S` 信号停止，则获取导致其停止的信号，并作出相应输出，然后从 `joblist` 中删除；若子进程只是停止，那么需要将其运行状态改为停止运行并做信号输出。

```
void sigchld_handler(int sig)
{
    sigset_t mask_all, prev_all;
    pid_t pid;
    int status;
    sigfillset(&mask_all);
    sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
    while((pid=waitpid(-1, &status, WNOHANG|WUNTRACED))>0){
        struct job_t *job=getjobpid(jobs, pid);
        if(WIFEXITED(status)){
            deletejob(jobs, pid);
        }
        else if(WIFSIGNALED(status)){
            if(verbose)
                printf("%d terminated by signal.\n", pid);
            printf("Job [%d] (%d) terminated by signal %d\n", job->jid, job->pid, WTERMSIG(status));
            deletejob(jobs, pid);
        }
        else if(WIFSTOPPED(status)){
            if(verbose)
                printf("%d stopped by signal.\n", pid);
            printf("Job [%d] (%d) stopped by signal %d\n", job->jid, job->pid, WSTOPSIG(status));
            job->state=ST;
        }
    }
    sigprocmask(SIG_SETMASK, &prev_all, NULL);
}
```

要点分析:

1.使用 `while((pid=waitpid(-1,&status,WNOHANG|WUNTRACED))>0)` 循环。这句代码有很多注意的点。①首先用 `while...>0` 的方式，这样可以一次回收多个已经终止的子进程，直至没有需要处理的子进程为止。这是为了防止由于许多子进程终止而导致某些信号被阻塞而被丢弃，从而导致有一些子进程没有被恰当地回收。②传入 `status` 参数。通过这个参数来检查已回收子进程的退出状态，进而根据不同的原因选择不同的处理方式。③使用 `WUNTRACED` 参数。这个参数作用是监视子进程的停止情况，这样子进程停止也会导致这个信号处理函数的调用，并且获取到该子进程的 `pid`，从而可以实现前台任务的状态切换。另外 `WNOHANG` 参数是指示 `waitpid` 不阻塞进程，若无需处理的子进程可以立即返回 0，这里我没有体会到它的用处何在，但是 PPT 上这么写的，我也就这么写了，结果也都正确。

2.在循环体运行前阻塞所有信号。这样避免了不必要的竞争而引发的未知错误，保证了函数的正确性。

3.利用 `status` 参数来获取子进程的状态。这样根据子进程不同的终止或停止原因可以作出相应的动作。

4.对 `job_t` 和 `joblist` 的操作一定要放在这里！另外两个信号处理程序只需要做信号转发工作就好了，原因分析如下。

一开始我把 `deletejob`、修改为停止状态等工作放到了 `sigint_handler`、`sigstp_handler` 当中去做，但是这样出现了第 16 个测试无法通过、卡死的情况。经过分析原因后我明白了：我们用 `Signal` 函数修改信号处理后，只有在 `tsh` 主进程里是作出了这样的修改；当我们 `fork` 并 `execve` 一个子进程后，其信号处理方式恢复到默认情况。这样我们从键盘键入信号的时候，信号是由 `shell` 再转发给我们的 `tsh`，然后 `tsh` 调用这两个信号处理程序转发给当前的前台进程。

然而，如果这个信号是来自于其他地方，比如我们测试 16 中的情况：信号是由前台作业发送给自己的。这样信号是直接传到了前台作业，没有经过那两个信号处理程序转发，从而导致 `joblist` 和 `job_t` 的状态并没有更新，进而发生了错误，比如发送 `SIGTSTP` 后，前台作业停止，而 `tsh` 主进程还认为有前台作业在运行，结果导致程序停止卡死。

我作出对应的修改后，所有测试全部通过！

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

（2）检查每个系统调用的返回值——5 分

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	
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test01 ./sdriver.pl -t trace01.txt -s ./tsh -a "-p" # # trace01.txt - Properly terminate on EOF. #</pre>		<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest01 ./sdriver.pl -t trace01.txt -s ./tshref -a "-p" # # trace01.txt - Properly terminate on EOF. #</pre>	
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	
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test02 ./sdriver.pl -t trace02.txt -s ./tsh -a "-p" # # trace02.txt - Process builtin quit command. #</pre>		<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest02 ./sdriver.pl -t trace02.txt -s ./tshref -a "-p" # # trace02.txt - Process builtin quit command. #</pre>	
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
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test03 ./sdriver.pl -t trace03.txt -s ./tsh -a "-p" # # trace03.txt - Run a foreground job. # tsh> quit</pre>	<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest03 ./sdriver.pl -t trace03.txt -s ./tshref -a "-p" # # trace03.txt - Run a foreground job. # tsh> quit</pre>
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
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test04 ./sdriver.pl -t trace04.txt -s ./tsh -a "-p" # # trace04.txt - Run a background job. # tsh> ./myspin 1 & [1] (98233) ./myspin 1 &</pre>	<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest04 ./sdriver.pl -t trace04.txt -s ./tshref -a "-p" # # trace04.txt - Run a background job. # tsh> ./myspin 1 & [1] (98239) ./myspin 1 &</pre>
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
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test05 ./sdriver.pl -t trace05.txt -s ./tsh -a "-p" # # trace05.txt - Process jobs builtin command. # tsh> ./myspin 2 & [1] (98251) ./myspin 2 & tsh> ./myspin 3 & [2] (98253) ./myspin 3 & tsh> jobs [1] (98251) Running ./myspin 2 & [2] (98253) Running ./myspin 3 &</pre>	<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest05 ./sdriver.pl -t trace05.txt -s ./tshref -a "-p" # # trace05.txt - Process jobs builtin command. # tsh> ./myspin 2 & [1] (98260) ./myspin 2 & tsh> ./myspin 3 & [2] (98262) ./myspin 3 & tsh> jobs [1] (98260) Running ./myspin 2 & [2] (98262) Running ./myspin 3 &</pre>
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
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test06 ./sdriver.pl -t trace06.txt -s ./tsh -a "-p" # # trace06.txt - Forward SIGINT to foreground job. # tsh> ./myspin 4 Job [1] (98276) terminated by signal 2</pre>	<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest06 ./sdriver.pl -t trace06.txt -s ./tshref -a "-p" # # trace06.txt - Forward SIGINT to foreground job. # tsh> ./myspin 4 Job [1] (98356) terminated by signal 2</pre>
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
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test07 ./sdriver.pl -t trace07.txt -s ./tsh -a "-p" # # trace07.txt - Forward SIGINT only to foreground job. # tsh> ./myspin 4 & [1] (98367) ./myspin 4 & tsh> ./myspin 5 Job [2] (98369) terminated by signal 2 tsh> jobs [1] (98367) Running ./myspin 4 &</pre>	<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest07 ./sdriver.pl -t trace07.txt -s ./tshref -a "-p" # # trace07.txt - Forward SIGINT only to foreground job. # tsh> ./myspin 4 & [1] (98376) ./myspin 4 & tsh> ./myspin 5 Job [2] (98378) terminated by signal 2 tsh> jobs [1] (98376) Running ./myspin 4 &</pre>
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
<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test08 ./sdriver.pl -t trace08.txt -s ./tsh -a "-p" # # trace08.txt - Forward SIGTSTP only to foreground job. # tsh> ./myspin 4 & [1] (98392) ./myspin 4 & tsh> ./myspin 5 Job [2] (98394) stopped by signal 20 tsh> jobs [1] (98392) Running ./myspin 4 & [2] (98394) Stopped ./myspin 5</pre>	<pre>dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest08 ./sdriver.pl -t trace08.txt -s ./tshref -a "-p" # # trace08.txt - Forward SIGTSTP only to foreground job. # tsh> ./myspin 4 & [1] (98401) ./myspin 4 & tsh> ./myspin 5 Job [2] (98403) stopped by signal 20 tsh> jobs [1] (98401) Running ./myspin 4 & [2] (98403) Stopped ./myspin 5</pre>
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
<pre> dqvs87@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test09 ./sdriver.pl -t trace09.txt -s ./tsh -a "-p" # # trace09.txt - Process bg builtin command # tsh> ./myspin 4 & [1] (98416) ./myspin 4 & tsh> ./myspin 5 Job [2] (98418) stopped by signal 20 tsh> jobs [1] (98416) Running ./myspin 4 & [2] (98418) Stopped ./myspin 5 tsh> bg %2 [2] (98418) ./myspin 5 tsh> jobs [1] (98416) Running ./myspin 4 & [2] (98418) Running ./myspin 5 </pre>	<pre> dqvs87@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest09 ./sdriver.pl -t trace09.txt -s ./tshref -a "-p" # # trace09.txt - Process bg builtin command # tsh> ./myspin 4 & [1] (98435) ./myspin 4 & tsh> ./myspin 5 Job [2] (98437) stopped by signal 20 tsh> jobs [1] (98435) Running ./myspin 4 & [2] (98437) Stopped ./myspin 5 tsh> bg %2 [2] (98437) ./myspin 5 tsh> jobs [1] (98435) Running ./myspin 4 & [2] (98437) Running ./myspin 5 </pre>
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
<pre> dqvs87@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test10 ./sdriver.pl -t trace10.txt -s ./tsh -a "-p" # # trace10.txt - Process fg builtin command. # tsh> ./myspin 4 & [1] (98449) ./myspin 4 & tsh> fg %1 Job [1] (98449) stopped by signal 20 tsh> jobs [1] (98449) Stopped ./myspin 4 & tsh> fg %1 tsh> jobs </pre>	<pre> dqvs87@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest10 ./sdriver.pl -t trace10.txt -s ./tshref -a "-p" # # trace10.txt - Process fg builtin command. # tsh> ./myspin 4 & [1] (98459) ./myspin 4 & tsh> fg %1 Job [1] (98459) stopped by signal 20 tsh> jobs [1] (98459) Stopped ./myspin 4 & tsh> fg %1 tsh> jobs </pre>
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																																																																																																																																																																																																																												
<pre>dqvs87@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test11 ./sdriver.pl -t trace11.txt -s ./tsh -a "-p" # # trace11.txt - Forward SIGINT to every process in foreground process group # tsh> ./mysplit 4 Job [1] (98474) terminated by signal 2 tsh> /bin/ps a</pre> <table><thead><tr><th>PID</th><th>TTY</th><th>STAT</th><th>TIME</th><th>COMMAND</th></tr></thead><tbody><tr><td>1720</td><td>ttty2</td><td>Ssl+</td><td>0:00</td><td>/usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu</td></tr><tr><td>1722</td><td>ttty2</td><td>Sl+</td><td>23:14</td><td>/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Authority -background none -noretteptty -verbose 3</td></tr><tr><td>1777</td><td>ttty2</td><td>Sl+</td><td>0:00</td><td>/usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu</td></tr><tr><td>91761</td><td>pts/0</td><td>Ss</td><td>0:00</td><td>bash</td></tr><tr><td>95162</td><td>pts/0</td><td>T</td><td>0:00</td><td>make test16</td></tr><tr><td>95163</td><td>pts/0</td><td>T</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"</td></tr><tr><td>95164</td><td>pts/0</td><td>T</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p</td></tr><tr><td>95165</td><td>pts/0</td><td>Z</td><td>0:00</td><td>[tsh] <defunct></td></tr><tr><td>96360</td><td>pts/0</td><td>T</td><td>0:00</td><td>make test16</td></tr><tr><td>96361</td><td>pts/0</td><td>T</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"</td></tr><tr><td>96362</td><td>pts/0</td><td>T</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p</td></tr><tr><td>96363</td><td>pts/0</td><td>Z</td><td>0:00</td><td>[tsh] <defunct></td></tr><tr><td>96573</td><td>pts/0</td><td>T</td><td>0:00</td><td>make test16</td></tr><tr><td>96574</td><td>pts/0</td><td>T</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"</td></tr><tr><td>96575</td><td>pts/0</td><td>T</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p</td></tr><tr><td>96576</td><td>pts/0</td><td>Z</td><td>0:00</td><td>[tsh] <defunct></td></tr><tr><td>98469</td><td>pts/0</td><td>S+</td><td>0:00</td><td>make test11</td></tr><tr><td>98470</td><td>pts/0</td><td>S+</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace11.txt -s ./tsh -a "-p"</td></tr><tr><td>98471</td><td>pts/0</td><td>S+</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tsh -a -p</td></tr><tr><td>98472</td><td>pts/0</td><td>S+</td><td>0:00</td><td>/tsh -p</td></tr><tr><td>98477</td><td>pts/0</td><td>R</td><td>0:00</td><td>/bin/ps a</td></tr></tbody></table>	PID	TTY	STAT	TIME	COMMAND	1720	ttty2	Ssl+	0:00	/usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu	1722	ttty2	Sl+	23:14	/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Authority -background none -noretteptty -verbose 3	1777	ttty2	Sl+	0:00	/usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu	91761	pts/0	Ss	0:00	bash	95162	pts/0	T	0:00	make test16	95163	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"	95164	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p	95165	pts/0	Z	0:00	[tsh] <defunct>	96360	pts/0	T	0:00	make test16	96361	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"	96362	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p	96363	pts/0	Z	0:00	[tsh] <defunct>	96573	pts/0	T	0:00	make test16	96574	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"	96575	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p	96576	pts/0	Z	0:00	[tsh] <defunct>	98469	pts/0	S+	0:00	make test11	98470	pts/0	S+	0:00	/bin/sh -c ./sdriver.pl -t trace11.txt -s ./tsh -a "-p"	98471	pts/0	S+	0:00	/usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tsh -a -p	98472	pts/0	S+	0:00	/tsh -p	98477	pts/0	R	0:00	/bin/ps a	<pre>dqvs87@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest11 ./sdriver.pl -t trace11.txt -s ./tshref -a "-p" # # trace11.txt - Forward SIGINT to every process in foreground process group # tsh> ./mysplit 4 Job [1] (98484) terminated by signal 2 tsh> /bin/ps a</pre> <table><thead><tr><th>PID</th><th>TTY</th><th>STAT</th><th>TIME</th><th>COMMAND</th></tr></thead><tbody><tr><td>1720</td><td>ttty2</td><td>Ssl+</td><td>0:00</td><td>/usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu</td></tr><tr><td>1722</td><td>ttty2</td><td>Sl+</td><td>23:16</td><td>/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Authority -background none -noretteptty -verbose 3</td></tr><tr><td>1777</td><td>ttty2</td><td>Sl+</td><td>0:00</td><td>/usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu</td></tr><tr><td>91761</td><td>pts/0</td><td>Ss</td><td>0:00</td><td>bash</td></tr><tr><td>95162</td><td>pts/0</td><td>T</td><td>0:00</td><td>make test16</td></tr><tr><td>95163</td><td>pts/0</td><td>T</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"</td></tr><tr><td>95164</td><td>pts/0</td><td>T</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p</td></tr><tr><td>95165</td><td>pts/0</td><td>Z</td><td>0:00</td><td>[tsh] <defunct></td></tr><tr><td>96360</td><td>pts/0</td><td>T</td><td>0:00</td><td>make test16</td></tr><tr><td>96361</td><td>pts/0</td><td>T</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"</td></tr><tr><td>96362</td><td>pts/0</td><td>T</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p</td></tr><tr><td>96363</td><td>pts/0</td><td>Z</td><td>0:00</td><td>[tsh] <defunct></td></tr><tr><td>96573</td><td>pts/0</td><td>T</td><td>0:00</td><td>make test16</td></tr><tr><td>96574</td><td>pts/0</td><td>T</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"</td></tr><tr><td>96575</td><td>pts/0</td><td>T</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p</td></tr><tr><td>96576</td><td>pts/0</td><td>Z</td><td>0:00</td><td>[tsh] <defunct></td></tr><tr><td>98479</td><td>pts/0</td><td>S+</td><td>0:00</td><td>make test11</td></tr><tr><td>98480</td><td>pts/0</td><td>S+</td><td>0:00</td><td>/bin/sh -c ./sdriver.pl -t trace11.txt -s ./tshref -a "-p"</td></tr><tr><td>98481</td><td>pts/0</td><td>S+</td><td>0:00</td><td>/usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tshref -a -p</td></tr><tr><td>98482</td><td>pts/0</td><td>S+</td><td>0:00</td><td>/tshref -p</td></tr><tr><td>98487</td><td>pts/0</td><td>R</td><td>0:00</td><td>/bin/ps a</td></tr></tbody></table>	PID	TTY	STAT	TIME	COMMAND	1720	ttty2	Ssl+	0:00	/usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu	1722	ttty2	Sl+	23:16	/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Authority -background none -noretteptty -verbose 3	1777	ttty2	Sl+	0:00	/usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu	91761	pts/0	Ss	0:00	bash	95162	pts/0	T	0:00	make test16	95163	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"	95164	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p	95165	pts/0	Z	0:00	[tsh] <defunct>	96360	pts/0	T	0:00	make test16	96361	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"	96362	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p	96363	pts/0	Z	0:00	[tsh] <defunct>	96573	pts/0	T	0:00	make test16	96574	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"	96575	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p	96576	pts/0	Z	0:00	[tsh] <defunct>	98479	pts/0	S+	0:00	make test11	98480	pts/0	S+	0:00	/bin/sh -c ./sdriver.pl -t trace11.txt -s ./tshref -a "-p"	98481	pts/0	S+	0:00	/usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tshref -a -p	98482	pts/0	S+	0:00	/tshref -p	98487	pts/0	R	0:00	/bin/ps a
PID	TTY	STAT	TIME	COMMAND																																																																																																																																																																																																																									
1720	ttty2	Ssl+	0:00	/usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu																																																																																																																																																																																																																									
1722	ttty2	Sl+	23:14	/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Authority -background none -noretteptty -verbose 3																																																																																																																																																																																																																									
1777	ttty2	Sl+	0:00	/usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu																																																																																																																																																																																																																									
91761	pts/0	Ss	0:00	bash																																																																																																																																																																																																																									
95162	pts/0	T	0:00	make test16																																																																																																																																																																																																																									
95163	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"																																																																																																																																																																																																																									
95164	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p																																																																																																																																																																																																																									
95165	pts/0	Z	0:00	[tsh] <defunct>																																																																																																																																																																																																																									
96360	pts/0	T	0:00	make test16																																																																																																																																																																																																																									
96361	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"																																																																																																																																																																																																																									
96362	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p																																																																																																																																																																																																																									
96363	pts/0	Z	0:00	[tsh] <defunct>																																																																																																																																																																																																																									
96573	pts/0	T	0:00	make test16																																																																																																																																																																																																																									
96574	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"																																																																																																																																																																																																																									
96575	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p																																																																																																																																																																																																																									
96576	pts/0	Z	0:00	[tsh] <defunct>																																																																																																																																																																																																																									
98469	pts/0	S+	0:00	make test11																																																																																																																																																																																																																									
98470	pts/0	S+	0:00	/bin/sh -c ./sdriver.pl -t trace11.txt -s ./tsh -a "-p"																																																																																																																																																																																																																									
98471	pts/0	S+	0:00	/usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tsh -a -p																																																																																																																																																																																																																									
98472	pts/0	S+	0:00	/tsh -p																																																																																																																																																																																																																									
98477	pts/0	R	0:00	/bin/ps a																																																																																																																																																																																																																									
PID	TTY	STAT	TIME	COMMAND																																																																																																																																																																																																																									
1720	ttty2	Ssl+	0:00	/usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd --session=ubuntu																																																																																																																																																																																																																									
1722	ttty2	Sl+	23:16	/usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Authority -background none -noretteptty -verbose 3																																																																																																																																																																																																																									
1777	ttty2	Sl+	0:00	/usr/libexec/gnome-session-binary --systemd --systemd --session=ubuntu																																																																																																																																																																																																																									
91761	pts/0	Ss	0:00	bash																																																																																																																																																																																																																									
95162	pts/0	T	0:00	make test16																																																																																																																																																																																																																									
95163	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"																																																																																																																																																																																																																									
95164	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p																																																																																																																																																																																																																									
95165	pts/0	Z	0:00	[tsh] <defunct>																																																																																																																																																																																																																									
96360	pts/0	T	0:00	make test16																																																																																																																																																																																																																									
96361	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"																																																																																																																																																																																																																									
96362	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p																																																																																																																																																																																																																									
96363	pts/0	Z	0:00	[tsh] <defunct>																																																																																																																																																																																																																									
96573	pts/0	T	0:00	make test16																																																																																																																																																																																																																									
96574	pts/0	T	0:00	/bin/sh -c ./sdriver.pl -t trace16.txt -s ./tsh -a "-p"																																																																																																																																																																																																																									
96575	pts/0	T	0:00	/usr/bin/perl ./sdriver.pl -t trace16.txt -s ./tsh -a -p																																																																																																																																																																																																																									
96576	pts/0	Z	0:00	[tsh] <defunct>																																																																																																																																																																																																																									
98479	pts/0	S+	0:00	make test11																																																																																																																																																																																																																									
98480	pts/0	S+	0:00	/bin/sh -c ./sdriver.pl -t trace11.txt -s ./tshref -a "-p"																																																																																																																																																																																																																									
98481	pts/0	S+	0:00	/usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tshref -a -p																																																																																																																																																																																																																									
98482	pts/0	S+	0:00	/tshref -p																																																																																																																																																																																																																									
98487	pts/0	R	0:00	/bin/ps a																																																																																																																																																																																																																									
测试结论	相同																																																																																																																																																																																																																												

4.3.13 测试用例 trace13.txt 的输出截图 (1 分)

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
<pre> dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ 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] (98567) ./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] (98567) stopped by signal 20 tsh> bg %2 %2: No such job tsh> bg %1 [1] (98567) ./myspin 4 & tsh> jobs [1] (98567) Running ./myspin 4 & </pre>	<pre> dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ 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] (98586) ./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] (98586) stopped by signal 20 tsh> bg %2 %2: No such job tsh> bg %1 [1] (98586) ./myspin 4 & tsh> jobs [1] (98586) Running ./myspin 4 & </pre>
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
<pre> dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ 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] (98629) terminated by signal 2 tsh> ./myspin 3 & [1] (98631) ./myspin 3 & tsh> ./myspin 4 & [2] (98633) ./myspin 4 & tsh> jobs [1] (98631) Running ./myspin 3 & [2] (98633) Running ./myspin 4 & tsh> fg %1 Job [1] (98631) stopped by signal 20 tsh> jobs [1] (98631) Stopped ./myspin 3 & [2] (98633) Running ./myspin 4 & tsh> bg %3 %3: No such job tsh> bg %1 [1] (98631) ./myspin 3 & tsh> jobs [1] (98631) Running ./myspin 3 & [2] (98633) Running ./myspin 4 & tsh> fg %1 tsh> quit </pre>	<pre> dqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ 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] (98649) terminated by signal 2 tsh> ./myspin 3 & [1] (98651) ./myspin 3 & tsh> ./myspin 4 & [2] (98653) ./myspin 4 & tsh> jobs [1] (98651) Running ./myspin 3 & [2] (98653) Running ./myspin 4 & tsh> fg %1 Job [1] (98651) stopped by signal 20 tsh> jobs [1] (98651) Stopped ./myspin 3 & [2] (98653) Running ./myspin 4 & tsh> bg %3 %3: No such job tsh> bg %1 [1] (98651) ./myspin 3 & tsh> jobs [1] (98651) Running ./myspin 3 & [2] (98653) Running ./myspin 4 & tsh> fg %1 tsh> quit </pre>
测试结论	相同

4.3.16 测试用例 trace16txt 的输出截图（1 分）

tsh 测试结果		tshref 测试结果	
<pre>iqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make test16 ./sdriver.pl -t trace16.txt -s ./tsh -a "-p" # # trace16.txt - Tests whether the shell can handle SIGTSTP and SIGINT # signals that come from other processes instead of the terminal. # tsh> ./mystop 2 Job [1] (98667) stopped by signal 20 tsh> jobs [1] (98667) Stopped ./mystop 2 tsh> ./myint 2 Job [2] (98670) terminated by signal 2</pre>		<pre>iqv587@ubuntu:~/Documents/Code/shlab-handout-hit\$ make rtest16 ./sdriver.pl -t trace16.txt -s ./tshref -a "-p" # # trace16.txt - Tests whether the shell can handle SIGTSTP and SIGINT # signals that come from other processes instead of the terminal. # tsh> ./mystop 2 Job [1] (98676) stopped by signal 20 tsh> jobs [1] (98676) Stopped ./mystop 2 tsh> ./myint 2 Job [2] (98679) terminated by signal 2</pre>	
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：15+1（多个 trace16）。（另外泰山服务器上也全部正确）

第 5 章 总结

5.1 请总结本次实验的收获

通过这次 `tiny shell` 实验，我对于书上第八章内容有了较为透彻的理解。之前读书过程中，对于一些问题不求甚解或者根本没意识到自己没有理解。经过这次实验，在查找资料和动手试错的过程中，我逐步理解了书上和 PPT 中每一页所讲的内容和要点，并且熟悉了 `fork`, `exeve`, `signal`, `sigprocmask` 等等函数的调用方式和作用，更加深入地理解了程序间信号的传递以及作用方式。

总之，这次实验使我对于计算机中程序的运行更加深入了。

5.2 请给出对本次实验内容的建议

我猜想由于是时长的原因，这次实验把两个最难的功能给我们实现了，然后替换为了两个较简单的信号处理函数。不过我觉得自己动手做一下 `eval` 和 `do_bgfg` 函数，也会使我们受益匪浅。因此，我建议后面设置这个实验，可以给出这两个函数的大体框架，然后仍然让我们动手去实现，从这个过程中体会一个 `shell` 的执行命令的步骤。

注：本章为酌情加分项。

参考文献

- [1]. [Linux 进程全解 7—父进程 wait / waitip 回收子进程 天糊土的博客-CSDN 博客 父进程 wait](#)
- [2]. [Linux init 进程详解 hxpjava1 的博客-CSDN 博客 init 进程](#)
- [3]. [linux 进程--init 进程（九）_bob62856 的博客-CSDN 博客](#)
- [4]. Kernighan B W, Ritchie D M. The C Programming Language[M]. 2. Dennis Ritchie & Bell Labs, / June 2018.
- [5]. Bryant R E, David R. O'Hallaron. Computer Systems a Programmer's Perspective[M]. 3rd. Carnegie Mellon University, 2016.