

哈尔滨工业大学

实验报告

实 验（七）

题 目 TinyShell

微壳

专 业 计算机类

学 号 1190200122

班 级 1903001

学 生 袁野

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2020-6-4

计算机科学与技术学院

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 -
1.3 实验预习	- 4 -
第 2 章 实验预习	- 7 -
2.1 进程的概念、创建和回收方法（5 分）	- 7 -
2.2 信号的机制、种类（5 分）	- 7 -
2.3 信号的发送方法、阻塞方法、处理程序的设置方法（5 分）	- 8 -
2.4 什么是 SHELL，功能和处理流程（5 分）	- 10 -
第 3 章 TINY SHELL 的设计与实现	- 11 -
3.1.1 VOID EVAL(CHAR *CMDLINE)函数（10 分）	- 11 -
3.1.2 INT BUILTIN_CMD(CHAR **ARGV)函数（5 分）	- 11 -
3.1.3 VOID DO_BGFG(CHAR **ARGV) 函数（5 分）	- 12 -
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 -
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	- 15 -
4.3.2 测试用例 trace02.txt	- 16 -
4.3.3 测试用例 trace03.txt	- 16 -
4.3.4 测试用例 trace04.txt	- 16 -
4.3.5 测试用例 trace05.txt	- 16 -
4.3.6 测试用例 trace06.txt	- 17 -
4.3.7 测试用例 trace07.txt	- 17 -
4.3.8 测试用例 trace08.txt	- 17 -
4.3.9 测试用例 trace09.txt	- 18 -
4.3.10 测试用例 trace10.txt	- 18 -
4.3.11 测试用例 trace11.txt	- 18 -
4.3.12 测试用例 trace12.txt	- 19 -
4.3.13 测试用例 trace13.txt	- 20 -

4.3.14 测试用例 <i>trace14.txt</i>	- 20 -
4.3.15 测试用例 <i>trace15.txt</i>	- 21 -
4.4 自测试评分.....	错误!未定义书签。
第 5 章 总结	- 23 -
5.1 请总结本次实验的收获.....	- 23 -
5.2 请给出对本次实验内容的建议.....	- 23 -
参考文献.....	- 25 -

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

Legion Y7000P 2019 PG0
CPU: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz (12 CPUs), ~2.6GHz
RAM: 16384MB

1.2.2 软件环境

Windows 10 家庭中文版 64-bit
Ubuntu 20.04.2 LTS
VMware® Workstation 16 Player 16.1.0 build-17198959

1.2.3 开发工具

Microsoft Visual Studio Community 2019 版本 16.9.2
Microsoft Visual 1.54.3
GCC 9.3.0

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）

进程概念：一个执行中程序的实例。进程所提供的：我们的程序好像是系统中当前运行的唯一程序一样，我们的程序好像是独占的使用处理器和内存，处理器好像是无间断的执行我们程序中的指令，我们程序中的代码和数据好像是系统中内存中唯一的对象。同一个程序处理不同的数据就是不同的进程。

创建进程：每次用户通过 shell 输入一个可执行目标文件的名字，运行程序时，shell 就会创建一个新的进程。父进程通过调用 fork 函数创建一个新的运行的子进程，对于函数 `int fork(void)`：子进程中，fork 返回 0；父进程中，返回子进程的 PID；新创建的子进程几乎但不完全与父进程相同，子进程得到与父进程虚拟地址空间相同的但是独立的一份副本，子进程获得与父进程任何打开文件描述符相同的副本，最大区别是子进程有不同于父进程的 PID。

当进程终止时，它仍然消耗系统资源，被称为“僵尸 zombie”进程。父进程通过 wait 函数回收子进程，对于函数 `int wait(int *child_status)`：挂起当前进程的执行直到它的一个子进程终止，返回已终止子进程的 PID。如果父进程先于子进程退出，则子进程成为孤儿进程，此时将自动被 PID 为 1 的进程（即 init）接管。孤儿进程退出后，它的清理工作有祖先进程 init 自动处理。

2.2 信号的机制、种类（5 分）

信号的机制：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，类似于异常，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 ID 来标识的，信号中唯一的信息是它的 ID 和它的到达。

发送信号：内核通过更新目的进程上下文中的某个状态，发送一个信号给目的进程。

接收信号：当目的进程被内核强迫以某种方式对信号的发送作出反应时，他就接收了信号。

编号	信号名称	缺省动作	说明
1	SIGHUP	终止	终止控制终端或进程
2	SIGINT	终止	键盘产生的中断 (Ctrl-C)
3	SIGQUIT	dump	键盘产生的退出
4	SIGILL	dump	非法指令
5	SIGTRAP	dump	debug 中断
6	SIGABRT / SIGIOT	dump	异常中止
7	SIGBUS / SIGEMT	dump	总线异常/EMT 指令
8	SIGFPE	dump	浮点运算溢出
9	SIGKILL	终止	强制进程终止
10	SIGUSR1	终止	用户信号,进程可自定义用途
11	SIGSEGV	dump	非法内存地址引用
12	SIGUSR2	终止	用户信号, 进程可自定义用途
13	SIGPIPE	终止	向某个没有读取的管道中写入数据
14	SIGALRM	终止	时钟中断(闹钟)
15	SIGTERM	终止	进程终止
16	SIGSTKFLT	终止	协处理器栈错误
17	SIGCHLD	忽略	子进程退出或中断
18	SIGCONT	继续	如进程停止状态则开始运行
19	SIGSTOP	停止	停止进程运行
20	SIGSTP	停止	键盘产生的停止
21	SIGTTIN	停止	后台进程请求输入
22	SIGTTOU	停止	后台进程请求输出
23	SIGURG	忽略	socket 发生紧急情况
24	SIGXCPU	dump	CPU 时间限制被打破
25	SIGXFSZ	dump	文件大小限制被打破

26	SIGVTALRM	终止	虚拟定时时钟
27	SIGPROF	终止	profile timer clock
28	SIGWINCH	忽略	窗口尺寸调整
29	SIGIO/SIGPOLL	终止	I/O 可用
30	SIGPWR	终止	电源异常
31	SIGSYS / SYSUNUSED	dump	系统调用异常

2.3 信号的发送方法、阻塞方法、处理程序的设置方法（5 分）

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，在键盘上输入 ctrl-c 会导致内核发送一个 SIGINT 信号到前台进程组中的每个进程。Ctrl-z 会发送一个 SIGSTP 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 函数发送信号给其他进程（包括自己）4) 用 alarm 发送信号，进程可以通过调用 alarm 函数在指定 secs 秒后发送一个 SIGALRM 信号给调用进程。

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 如：一个 SIGINT 信号处理程序不能被另一个 SIGINT 信号中断（此时另一个 SIGINT 信号被阻塞）

显示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：

sigempty(set) - 初始化 set 为空集合

sigfill(set) - 把每个信号都添加到 set 中

sigadd(set) - 把指定的信号 signum 添加到 set

sigdel(set) - 从 set 中删除指定的信号 signum

三. 设置信号处理程序

可以使用 signal 函数修改和信号 signum 相关联的默认行为：handler_t
*signal(int signum, handler_t *handler)

handler 的不同取值：

1. SIG_IGN: 忽略类型为 signum 的信号

2. SIG_DFL: 类型为 signum 的信号行为恢复为默认行为

3. 否则， `handler` 就是用户定义的函数的地址，这个函数称为信号处理程序
只要进程接收到类型为 `signum` 的信号就会调用信号处理程序

将处理程序的地址传递到 `signal` 函数从而改变默认行为，这叫作设置信号处理程序。调用信号处理程序称为捕获信号

执行信号处理程序称为处理信号

当处理程序执行 `return` 时，控制会传递到控制流中被信号接收所中断的指令处

2.4 什么是 shell，功能和处理流程（5 分）

一. shell 定义

Shell 是系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行

二. shell 功能

实际上 Shell 是一个命令解释器，它解释由用户输入的命令并且把它们送到内核。不仅如此，Shell 有自己的编程语言用于对命令的编辑，它允许用户编写由 shell 命令组成的程序。Shell 编程语言具有普通编程语言的很多特点，比如它也有循环结构和分支控制结构等，用这种编程语言编写的 Shell 程序与其他应用程序具有同样的效果

三. shell 处理流程

shell 首先检查命令是否是内部命令，若不是再检查是否是一个应用程序（这里的应用程序可以是 Linux 本身的实用程序，如 `ls` 和 `rm`，也可以是购买的商业程序，如 `xv`，或者是自由软件，如 `emacs`）。然后 shell 在搜索路径里寻找这些应用程序（搜索路径就是一个能找到可执行程序的路径列表）。如果键入的命令不是一个内部命令并且在路径里没有找到这个可执行文件，将会显示一条错误信息。如果能够成功找到命令，该内部命令或应用程序将被分解为系统调用并传给 Linux 内核。

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:

解析用户输入的指令, 如果是内置指令就执行, 否则 fork 的一个子进程会在子进程的上下文中运行该作业。当作业在前台运行时, 则等待终止返回, 若是在后台运行则打印进程信息。

参 数:

cmdline 表示用户输入的指令。

处理流程:

首先调用 parseline 函数分析指令, 并同时判断是否需要后台执行。然后调用 builtin_cmd 判断该函数是否为内置函数。如果是内置函数则立即执行, 否则创建一个子进程, 在子进程的上下文中运行命令行要求的函数。

要点分析:

1. 所有的子进程必须有自己的唯一进程组 ID, 否则我们在键盘上输入 ctrl-c(ctrl-z)时, 后台的子进程就会从内核接收 SIGINT(SIGTSTP), 为了避免这种错误, 所以每个子进程必须有唯一的进程组 ID。

2. 我们最开始需要设置阻塞信号, 让信号可以被发送但不会被接收, 这样就能够避免把不存在的子进程添加到了作业列表中。

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

函数功能:

判断用户键入的是否为内置命令以及类型。

参 数:

传入的参数数组 argv。

处理流程:

1. 构造函数判断传入的参数数组 argv 中的实参 argv[0]是否为内置命令, 只需要比较实参与内置命令即可。

2. 紧接着做如下判断:

如果命令为 quit 直接退出。

如果为&直接返回。

如果为 job 命令则调用 listjobs()函数并打印 job 列表。

如果为 fg 或者 bg 则直接调用 do_bgfg()。

要点分析：

需要了解这些命令的实际含义以及采取的行为，直接退出和返回是不同的。

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

函数功能：

执行 bg 和 fg 命令。

参 数：

传入的参数数组 argv。

处理流程：

首先对命令进行解析，判断 bg 或 fg 后边是否有参数，如果没有参数便忽略命令。随后判断后边跟着的是简单的数字还是%加数字的形式，如果是前者那么说明取得进程号，得到进程号之后进一步得到 job；如果是后者则说明%后是任务号，获取 jid 后进一步获得 job。

判断 bg 还是 fg，如果是后台进程则发送 SIGCONT 信号给进程组 PID 的每个进程，并且设置任务的状态为 BG，打印任务的 jid, pid 和命令行；如果是前台进程，则发送 SIGCONT 信号给进程组 PID 的每个进程，并且设置任务的状态为 FG，调用 waitfg(jobp->pid)，等待前台进程结束。

要点分析：

通过命令格式的不同来判断究竟通过 pid 还是 jid 去获得 job。然后再根据 bg 或 fg 进行相应的处理。

3.1.4 void waitfg(pid_t pid) 函数 (5 分)

函数功能：

阻止直到进程 pid 不再是前台进程。

参 数：

前台进程 pid。

处理流程：

通过 while (pid == fgpid(jobs))不断判断传入的 pid 是否是一个前台进程的 pid，是的话则一直循环，不是就跳出循环。循环内部如果不是前台进程的 pid，则使用 sleep(1)使进程休眠，直到收到一个让该进程终止或处理的程序信号。

要点分析：

在 waitfg 函数中，在 sleep 函数附近使用 busy loop。

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

函数功能：

只要子作业终止（变为僵尸），或者因为收到 SIGSTOP 或 SIGTSTP 信号而停止，内核就会向 shell 发送 SIGCHLD。处理程序收集所有可用的僵尸子节点，但不等待任何其他当前正在运行的子节点终止。

参 数：

信号 sig

处理流程：

将所有的信号加入到 mask 阻塞集合中，设置 olderrno 为 errno。

尽可能地回收子进程，其中 WNOHANG | WUNTRACED 表示立即返回，如果等待集合中没有进程被中止或停止返回 0，否则孩子返回进程的 pid。

为了找到 job，需要执行 NewJob = getjobpid(jobs, pid) 得到 pid 的 jib。

依次检查子进程不同的退出情况，WIFSTOPPED(status) 为真的话就说明是由子进程停止引起 waitpid 函数返回，这个时候使用 NewJob->state = ST 将 job 的状态改为 ST，并且进行输出。WIFSIGNALED(status) 为真，说明子进程是因为某一个信号没有被捕获导致终止的，直接输出信息即可。然后执行 deletejob(jobs, pid) 进行回收。

最后清空缓存区并解除阻塞，回复 errno 即可。

要点分析：

由于 deletejob() 函数会对全局变量 jobs 的值进行更改，所以需要阻塞信号来防止这种更改。

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), 并填写完成 4.3 节的相应表格。

4.2 测试结果评价

tsh 与 tshref 的输出在以下两个方面可以不同:

(1) pid

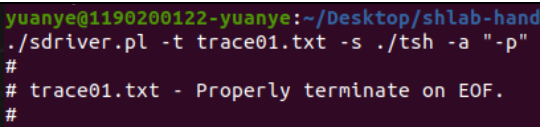
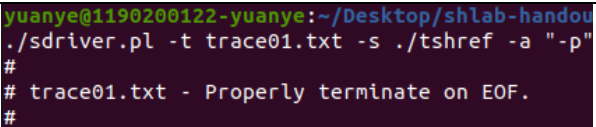
(2) 测试文件 trace11.txt, trace12.txt 和 trace13.txt 中的/bin/ps 命令, 每次运行的输出都会不同, 但每个 mysplit 进程的运行状态应该相同。

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

填写以下各个测试用例的测试结果, 每个测试用例 1 分。

4.3.1 测试用例 trace01.txt

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

tsh 测试结果	tshref 测试结果
<pre>yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./sdriver.pl -t trace02.txt -s ./tsh -a "-p" # # trace02.txt - Process builtin quit command. #</pre>	<pre>yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./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

tsh 测试结果	tshref 测试结果
<pre>yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./sdriver.pl -t trace03.txt -s ./tsh -a "-p" # # trace03.txt - Run a foreground job. # tsh> quit</pre>	<pre>yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./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

tsh 测试结果	tshref 测试结果
<pre>yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./sdriver.pl -t trace04.txt -s ./tsh -a "-p" # # trace04.txt - Run a background job. # tsh> ./myspin 1 & [1] (25317) ./myspin 1 &</pre>	<pre>yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./sdriver.pl -t trace04.txt -s ./tshref -a "-p" # # trace04.txt - Run a background job. # tsh> ./myspin 1 & [1] (25325) ./myspin 1 &</pre>
测试结论	相同

4.3.5 测试用例 trace05.txt

tsh 测试结果	tshref 测试结果
----------	-------------

<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-h ./sdriver.pl -t trace05.txt -s ./tsh -a "-p" # # trace05.txt - Process jobs builtin command. # tsh> ./myspin 2 & [1] (24779) ./myspin 2 & tsh> ./myspin 3 & [2] (24781) ./myspin 3 & tsh> jobs [1] (24779) Running ./myspin 2 & [2] (24781) Running ./myspin 3 & </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-h ./sdriver.pl -t trace05.txt -s ./tshref -a "-p" # # trace05.txt - Process jobs builtin command. # tsh> ./myspin 2 & [1] (24788) ./myspin 2 & tsh> ./myspin 3 & [2] (24790) ./myspin 3 & tsh> jobs [1] (24788) Running ./myspin 2 & [2] (24790) Running ./myspin 3 & </pre>
测试结论	相同

4.3.6 测试用例 trace06.txt

tsh 测试结果	tshref 测试结果
<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-h ./sdriver.pl -t trace06.txt -s ./tsh -a "-p" # # trace06.txt - Forward SIGINT to foreground job. # tsh> ./myspin 4 Job [1] (24800) terminated by signal 2 </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-h ./sdriver.pl -t trace06.txt -s ./tshref -a "-p" # # trace06.txt - Forward SIGINT to foreground job. # tsh> ./myspin 4 Job [1] (24806) terminated by signal 2 </pre>
测试结论	相同

4.3.7 测试用例 trace07.txt

tsh 测试结果	tshref 测试结果
<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$./sdriver.pl -t trace07.txt -s ./tsh -a "-p" # # trace07.txt - Forward SIGINT only to foreground job. # tsh> ./myspin 4 & [1] (24822) ./myspin 4 & tsh> ./myspin 5 Job [2] (24824) terminated by signal 2 tsh> jobs [1] (24822) Running ./myspin 4 & </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$./sdriver.pl -t trace07.txt -s ./tshref -a "-p" # # trace07.txt - Forward SIGINT only to foreground job. # tsh> ./myspin 4 & [1] (24833) ./myspin 4 & tsh> ./myspin 5 Job [2] (24835) terminated by signal 2 tsh> jobs [1] (24833) Running ./myspin 4 & </pre>
测试结论	相同

4.3.8 测试用例 trace08.txt

tsh 测试结果	tshref 测试结果
----------	-------------

<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ m ./sdriver.pl -t trace08.txt -s ./tsh -a "-p" # # trace08.txt - Forward SIGTSTP only to foreground job. # tsh> ./myspin 4 & [1] (24847) ./myspin 4 & tsh> ./myspin 5 Job [2] (24849) stopped by signal 20 tsh> jobs [1] (24847) Running ./myspin 4 & [2] (24849) Stopped ./myspin 5 </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ m ./sdriver.pl -t trace08.txt -s ./tshref -a "-p" # # trace08.txt - Forward SIGTSTP only to foreground job. # tsh> ./myspin 4 & [1] (24874) ./myspin 4 & tsh> ./myspin 5 Job [2] (24876) stopped by signal 20 tsh> jobs [1] (24874) Running ./myspin 4 & [2] (24876) Stopped ./myspin 5 </pre>
测试结论	相同

4.3.9 测试用例 trace09.txt

tsh 测试结果	tshref 测试结果
<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ m ./sdriver.pl -t trace09.txt -s ./tsh -a "-p" # # trace09.txt - Process bg builtin command # tsh> ./myspin 4 & [1] (24883) ./myspin 4 & tsh> ./myspin 5 Job [2] (24885) stopped by signal 20 tsh> jobs [1] (24883) Running ./myspin 4 & [2] (24885) Stopped ./myspin 5 tsh> bg %2 [2] (24885) ./myspin 5 tsh> jobs [1] (24883) Running ./myspin 4 & [2] (24885) Running ./myspin 5 </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ m ./sdriver.pl -t trace09.txt -s ./tshref -a "-p" # # trace09.txt - Process bg builtin command # tsh> ./myspin 4 & [1] (24897) ./myspin 4 & tsh> ./myspin 5 Job [2] (24899) stopped by signal 20 tsh> jobs [1] (24897) Running ./myspin 4 & [2] (24899) Stopped ./myspin 5 tsh> bg %2 [2] (24899) ./myspin 5 tsh> jobs [1] (24897) Running ./myspin 4 & [2] (24899) Running ./myspin 5 </pre>
测试结论	相同

4.3.10 测试用例 trace10.txt

tsh 测试结果	tshref 测试结果
<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ m ./sdriver.pl -t trace10.txt -s ./tsh -a "-p" # # trace10.txt - Process fg builtin command. # tsh> ./myspin 4 & [1] (24921) ./myspin 4 & tsh> fg %1 Job [1] (24921) stopped by signal 20 tsh> jobs [1] (24921) Stopped ./myspin 4 & tsh> fg %1 tsh> jobs </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ m ./sdriver.pl -t trace10.txt -s ./tshref -a "-p" # # trace10.txt - Process fg builtin command. # tsh> ./myspin 4 & [1] (24931) ./myspin 4 & tsh> fg %1 Job [1] (24931) stopped by signal 20 tsh> jobs [1] (24931) Stopped ./myspin 4 & tsh> fg %1 tsh> jobs </pre>
测试结论	相同

4.3.11 测试用例 trace11.txt

测试中 ps 指令的输出内容较多，仅记录和本实验密切相关的 tsh、

mysplit 等进程的部分信息即可。

tsh 测试结果	tshref 测试结果
<pre> yuanye@1190200122-yuanye:~/Desktop/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] (24951) terminated by signal 2 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:38 /usr/lib/xorg/Xorg vt2 -dis playfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session- binary --systemd --systemd --session=ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 24946 pts/0 S+ 0:00 make test11 24947 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace11.txt -s ./tsh -a "-p" 24948 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tsh -a -p 24949 pts/0 S+ 0:00 ./tsh -p 24954 pts/0 R 0:00 /bin/ps a </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/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] (24960) terminated by signal 2 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:38 /usr/lib/xorg/Xorg vt2 -dis playfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session- binary --systemd --systemd --session=ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 24955 pts/0 S+ 0:00 make rtest11 24956 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace11.txt -s ./tshref -a "-p" 24957 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tshref -a -p 24958 pts/0 S+ 0:00 ./tshref -p 24963 pts/0 R 0:00 /bin/ps a </pre>
测试结论	相同

4.3.12 测试用例 trace12.txt

测试中 ps 指令的输出内容较多，仅记录和本实验密切相关的 tsh、mysplit 等进程的部分信息即可。

tsh 测试结果	tshref 测试结果
<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ make test12 ./sdriver.pl -t trace12.txt -s ./tsh -a "-p" # # trace12.txt - Forward SIGTSTP to every process in foreground process group # tsh> ./mysplit 4 Job [1] (25093) stopped by signal 20 tsh> jobs [1] (25093) Stopped ./mysplit 4 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:38 /usr/lib/xorg/Xorg vt2 -dis playfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session- binary --systemd --systemd --session=ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 25088 pts/0 S+ 0:00 make test12 25089 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace12.txt -s ./tsh -a "-p" 25090 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace12.txt -s ./tsh -a -p 25091 pts/0 S+ 0:00 ./tsh -p 25093 pts/0 T 0:00 ./mysplit 4 25094 pts/0 T 0:00 ./mysplit 4 25097 pts/0 R 0:00 /bin/ps a </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout-hit\$ make rtest12 ./sdriver.pl -t trace12.txt -s ./tshref -a "-p" # # trace12.txt - Forward SIGTSTP to every process in foreground process group # tsh> ./mysplit 4 Job [1] (25103) stopped by signal 20 tsh> jobs [1] (25103) Stopped ./mysplit 4 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:39 /usr/lib/xorg/Xorg vt2 -dis playfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session- binary --systemd --systemd --session=ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 25098 pts/0 S+ 0:00 make rtest12 25099 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace12.txt -s ./tshref -a "-p" 25100 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace12.txt -s ./tshref -a -p 25101 pts/0 S+ 0:00 ./tshref -p 25103 pts/0 T 0:00 ./mysplit 4 25104 pts/0 T 0:00 ./mysplit 4 25107 pts/0 R 0:00 /bin/ps a </pre>
测试结论	相同

4.3.13 测试用例 trace13.txt

测试中 ps 指令的输出内容较多，仅记录和本实验密切相关的 tsh、mysplit 等进程的部分信息即可。

tsh 测试结果	tshref 测试结果
<pre> yuan@1190200122-yuany:~/Desktop/shlab-handout\$ 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] (25118) stopped by signal 20 tsh> jobs [1] (25118) Stopped ./mysplit 4 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:39 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xau thority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session= ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 25113 pts/0 S+ 0:00 make test13 25114 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tsh -a "-p" 25115 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tsh -a -p 25116 pts/0 S+ 0:00 ./tsh -p 25118 pts/0 T 0:00 ./mysplit 4 25119 pts/0 T 0:00 ./mysplit 4 25122 pts/0 R 0:00 /bin/ps a tsh> fg %1 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:39 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xau thority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session= ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 25113 pts/0 S+ 0:00 make test13 25114 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tsh -a "-p" 25115 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tsh -a -p 25116 pts/0 S+ 0:00 ./tsh -p 25122 pts/0 R 0:00 /bin/ps a </pre>	<pre> yuan@1190200122-yuany:~/Desktop/shlab-handout\$ 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] (25131) stopped by signal 20 tsh> jobs [1] (25131) Stopped ./mysplit 4 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:39 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xau thority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session= ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 25126 pts/0 S+ 0:00 make rtest13 25127 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tshref -a "-p" 25128 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tshref -a -p 25129 pts/0 S+ 0:00 ./tshref -p 25131 pts/0 T 0:00 ./mysplit 4 25132 pts/0 T 0:00 ./mysplit 4 25141 pts/0 R 0:00 /bin/ps a tsh> fg %1 tsh> /bin/ps a PID TTY STAT TIME COMMAND 2002 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 2005 tty2 Sl+ 1:39 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xau thority -background none -noreset -keeptty -verbose 3 2024 tty2 Sl+ 0:00 /usr/libexec/gnome-session-binary --systemd --systemd --session= ubuntu 2099 tty2 Z+ 0:00 [fcitx] <defunct> 24376 pts/0 Ss 0:00 bash 25126 pts/0 S+ 0:00 make rtest13 25127 pts/0 S+ 0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tshref -a "-p" 25128 pts/0 S+ 0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tshref -a -p 25129 pts/0 S+ 0:00 ./tshref -p 25148 pts/0 R 0:00 /bin/ps a </pre>
测试结论	相同

4.3.14 测试用例 trace14.txt

tsh 测试结果	tshref 测试结果
<pre> yuan@1190200122-yuany:~/Desktop/shlab-handout\$./sdriver.pl -t trace14.txt -s ./tsh -a "-p" # # trace14.txt - Simple error handling # tsh> ./bogus ./bogus: Command not found tsh> ./myspin 4 & [1] (25164) ./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] (25164) stopped by signal 20 tsh> bg %2 %2: No such job tsh> bg %1 [1] (25164) ./myspin 4 & tsh> jobs [1] (25164) Running ./myspin 4 & </pre>	<pre> yuan@1190200122-yuany:~/Desktop/shlab-handout\$./sdriver.pl -t trace14.txt -s ./tshref -a "-p" # # trace14.txt - Simple error handling # tsh> ./bogus ./bogus: Command not found tsh> ./myspin 4 & [1] (25183) ./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] (25183) stopped by signal 20 tsh> bg %2 %2: No such job tsh> bg %1 [1] (25183) ./myspin 4 & tsh> jobs [1] (25183) Running ./myspin 4 & </pre>

测试结论	相同
------	----

4.3.15 测试用例 trace15.txt

tsh 测试结果	tshref 测试结果
<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./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] (25206) terminated by signal 2 tsh> ./myspin 3 & [1] (25208) ./myspin 3 & tsh> ./myspin 4 & [2] (25210) ./myspin 4 & tsh> jobs [1] (25208) Running ./myspin 3 & [2] (25210) Running ./myspin 4 & tsh> fg %1 Job [1] (25208) stopped by signal 20 tsh> jobs [1] (25208) Stopped ./myspin 3 & [2] (25210) Running ./myspin 4 & tsh> bg %3 %3: No such job tsh> bg %1 [1] (25208) ./myspin 3 & tsh> jobs [1] (25208) Running ./myspin 3 & [2] (25210) Running ./myspin 4 & tsh> fg %1 tsh> quit </pre>	<pre> yuanye@1190200122-yuanye:~/Desktop/shlab-handout ./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] (25230) terminated by signal 2 tsh> ./myspin 3 & [1] (25232) ./myspin 3 & tsh> ./myspin 4 & [2] (25234) ./myspin 4 & tsh> jobs [1] (25232) Running ./myspin 3 & [2] (25234) Running ./myspin 4 & tsh> fg %1 Job [1] (25232) stopped by signal 20 tsh> jobs [1] (25232) Stopped ./myspin 3 & [2] (25234) Running ./myspin 4 & tsh> bg %3 %3: No such job tsh> bg %1 [1] (25232) ./myspin 3 & tsh> jobs [1] (25232) Running ./myspin 3 & [2] (25234) Running ./myspin 4 & tsh> fg %1 tsh> quit </pre>
测试结论	相同

第 5 章 评测得分

总分 20 分

实验程序统一测试的评分（教师评价）：

（1）正确性得分：_____（满分 10）

（2）性能加权得分：_____（满分 10）

第 6 章 总结

5.1 请总结本次实验的收获

- 1.理解了 shell 的工作原理。
- 2.对信号的处理过程有了更深入的理解。
- 3.明白了信号处理相关的系统函数的作用

5.2 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

- [1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.
- [2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A 集[C]. 北京：中国科学出版社，1999.
- [3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. <http://www.ie.nthu.edu.tw/info/ie.newie.htm>（Big5）.
- [4] 湛颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.
- [5] KANAMORI H. Shaking Without Quaking[J]. Science, 1998, 279 (5359): 2063-2064.
- [6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science , 1998 , 281 : 331-332[1998-09-23]. <http://www.sciencemag.org/cgi/collection/anatmorp>.