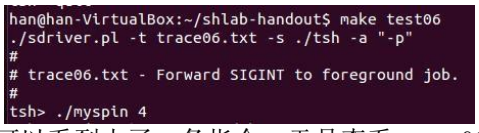
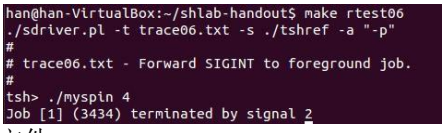
日志2.10和实验报告2.3

1.比较trace06执行不同结果，编程实现sigint_handler捕获INT响应、waitfg()等待、sigchld_handler回收僵死

执行trace06比较不同，得到结果如下：





可以看到少了一条指令，于是查看trace06文件：

```
#
# trace06.txt - Forward SIGINT to foreground job.
#
/bin/echo -e tsh> ./myspin 4
./myspin 4

SLEEP 2
INT
```

可以看到最后一条INT指令没有执行，说明在函数中缺少可以处理INT的分支。

对于TNT指令而言：

在sigint_handler函数中执行接收信号，通过kill(-pid,SIGINT)发送信号给指定进程，如果pid为负，说明给该进程组的所有进程发送信息。

```c
void sigint_handler(int sig)
{
    if (verbose)
    {
        printf("sigint_handler:entering\n");
    }
    pid_t pid;
    pid = fgpid(jobs);
    if (pid)
    {
        if (kill(-pid, SIGINT) < 0)
            unix_error("Signal error");
        if (verbose)
        {
            printf("sigint_handler:Job (%d) killed\n", pid);
        }
    }
    if (verbose)
    {
        printf("sigint_handler:exiting\n");
    }
    return;
}
```

在waitfg函数回收僵尸进程，使用在sigchld_handler中。

```c
void waitfg(pid_t pid)
{
    struct job_t *job;
    job = getjobpid(jobs, pid);
    while (1)
    {
        pause();
        if ((job->pid == 0 && job->state == UNDEF) || (job->pid == pid && job->state == ST))
        {
            if (verbose)
            {
                printf("waitfg: Process (%d) no longer the fg process\n", pid);
            }
            break;
        }
    }
    return;
}
```

在sigchld_handler函数中回收僵死，用while循环来避免信号阻塞的问题，为了回收所有的僵尸进程

```c
void sigchld_handler(int sig)
{
    pid_t pid;
    struct job_t *job;
    int jid;
    int status;

    if(verbose){
        printf("sigchld_handler: entering\n");
    }
    //立即返回，如果等待集合中没有任何子进程被停止或已终止， 那么返回值为 0，或者返回值等于那个被停止或者已终止的子进程的 PID
    while((pid = waitpid(-1, &status, WUNTRACED | WNOHANG)) > 0){
        //获取pid作业的jid
        job = getjobpid(jobs, pid);
        if(!job){
            app_error("can't find the job");
        }
        jid = job->jid;
        //检查已回收子进程的退出状态
        if(WIFEXITED(status)){//子进程通过调用 exit 或者一个返回 (return) 正常终止
            if(deletejob(jobs, pid)){
                if(verbose){
                    printf("sigchld_handler: Job [%d] (%d) deleted\n", jid, pid);
                }
            }else{
                app_error("can't delete job");
            }

            if(verbose){
                printf("sigchld_handler: Job [%d] (%d) terminates OK (status %d)\n", jid, pid, WEXITSTATUS(status));
            }
        }

        if(WIFSIGNALED(status)){//子进程是因为一个未被捕获的信号终止的
            if(deletejob(jobs, pid)){
                if(verbose){
                    printf("sigchld_handler: Job [%d] (%d) deleted\n", jid, pid);
                }
            }else{
                app_error("can't delete job");
            }

            printf("Job [%d] (%d) terminated by signal %d\n", jid, pid, WTERMSIG(status));
        }

        if(WIFSTOPPED(status)){//引起返回的子进程当前是被停止的
            job->state = ST;
            printf("Job [%d] (%d) stopped by signal %d\n", jid, pid, WSTOPSIG(status));
        }
    }
    return;
}
```

通过上述步骤，最终实现INT指令。

2.验证trace06～07，了解接收信号、信号处理、信号阻塞概念

接收信号：当内核从一个异常处理程序返回，准备将控制传递给进程p时，它会检查进程p的未被阻塞的待处理信号的集合，如果这个集合为空，那么内核将控制传递到p的逻辑控制流中的下一条指令。如果集合非空，那么内核选择集合中的某个信号k（通常是最小的k），并且强制p接收信号k。收到这个信号会触发进程的某种行为。一旦进程完成了这个行为，那么控制就传递回p的逻辑控制流中的下一条指令。

信号处理：程序对于所捕获的信号进行处理并最后终止。一般会只捕获一个信号并终止，但也有可能会捕获多个信号，那么就可能会出现待处理信号被阻塞、待处理信号不会排队等待、系统调用被中断的问题。

信号阻塞：将信号保持在未决状态，直到进程结束对此信号的阻塞，才执行递达动作。与忽略不同，忽略是信号处理（传递）的一种，阻塞直到被解除才能传递动作。信号未决是指从信号产生到递达之间的状态。信号递达是实际信号执行的处理过程，有三种状态（忽略、执行默认动作、捕获）。

```
#
# trace06.txt - Forward SIGINT to foreground job.
#
/bin/echo -e tsh> ./myspin 4
./myspin 4

SLEEP 2
INT
```

trace06跟踪文件先调用myspin函数睡眠4秒，然后再调用SLEEP内置指令睡眠2秒，最后调用INT指令，应该输出job[1] terminated by signal 2

```
#
# trace07.txt - Forward SIGINT only to foreground job.
#
/bin/echo -e tsh> ./myspin 4 \046
./myspin 4 &

/bin/echo -e tsh> ./myspin 5
./myspin 5

SLEEP 2
INT

/bin/echo tsh> jobs
jobs
```

trace07跟踪文件是先在后台调用myspin函数睡眠4秒，再调用myspin函数睡眠5秒，再调用SLEEP内置指令睡眠2秒，再调用INT指令，应该输出job [2] terminated by signal 2，再调用job指令输出所有的后台作业，最终得到的结果如下：

```
han@han-VirtualBox:~/shlab-handout$ make test06
./sdriver.pl -t trace06.txt -s ./tsh -a "-p"
#
# trace06.txt - Forward SIGINT to foreground job.
#
maketsh> ./myspin 4
Job [1] (2831) terminated by signal 2
han@han-VirtualBox:~/shlab-handout$ make rtest06
./sdriver.pl -t trace06.txt -s ./tshref -a "-p"
#
# trace06.txt - Forward SIGINT to foreground job.
#
tsh> ./myspin 4
Job [1] (2838) terminated by signal 2
```

```
han@han-VirtualBox:~/shlab-handout$ make test07
./sdriver.pl -t trace07.txt -s ./tsh -a "-p"
#
# trace07.txt - Forward SIGINT only to foreground job.
#
tsh> ./myspin 4 &
[1] (2844) ./myspin 4 &
tsh> ./myspin 5
Job [2] (2846) terminated by signal 2
tsh> jobs
[1] (2844) Running ./myspin 4 &
han@han-VirtualBox:~/shlab-handout$ make rtest07
./sdriver.pl -t trace07.txt -s ./tshref -a "-p"
#
# trace07.txt - Forward SIGINT only to foreground job.
#
tsh> ./myspin 4 &
[1] (2853) ./myspin 4 &
tsh> ./myspin 5
Job [2] (2855) terminated by signal 2
tsh> jobs
[1] (2853) Running ./myspin 4 &
```

可以看到test的结果和rtest的示例结果完全相同，证明实验正确。

3.比较trace08执行不同结果,编程实现sigtstp_handler捕获TSTP响应

执行trace08比较不同，得到如下结果：

```
han@han-VirtualBox:~/shlab-handout$ make test08
./sdriver.pl -t trace08.txt -s ./tsh -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (2917) ./myspin 4 &
tsh> ./myspin 5
tsh> jobs
```

```
han@han-VirtualBox:~/shlab-handout$ make rtest08
./sdriver.pl -t trace08.txt -s ./tshref -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (2926) ./myspin 4 &
tsh> ./myspin 5
Job [2] (2928) stopped by signal 20
tsh> jobs
[1] (2926) Running ./myspin 4 &
[2] (2928) Stopped ./myspin 5
```

查看trace08文件：

```
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
/bin/echo -e tsh> ./myspin 4 \046
./myspin 4 &

/bin/echo -e tsh> ./myspin 5
./myspin 5

SLEEP 2
TSTP

/bin/echo tsh> jobs
jobs
```

可以看出TSTP指令没有执行，直接跳过，导致程序出错。原因是在程序中没有对TSTP指令进行捕捉，所以无法进入可以处理TSTP指令的分支，由此将代码进行修改，结果如下：

```
void sigtstp_handler(int sig)
{
    pid_t pid;
    if (verbose)
        printf("sigtstp_handler: entering\n");
    pid = fgpid(jobs);
    if (pid)
    {
        if (kill(-pid, SIGTSTP) < 0)
            unix_error("Signal error");
        if (verbose)
            printf("sigint_handler: Job [%d] (%d) stopped\n", pid2jid(pid), pid);
    }
    if (verbose)
        printf("sigtstp_handler: exiting\n");
    return;
```

4.验证trace08

trace08跟踪文件先后台调用myspin函数睡眠4秒，再调用myspin函数睡眠5秒，再调用SLEEP内置指令睡眠2秒，再调用TSTP指令，应该输出job [2] stopped by signal 2，再调用job指令输出所有的后台作业，最终得到的结果如下：

```
han@han-VirtualBox:~/shlab-handout$ make test08
./sdriver.pl -t trace08.txt -s ./tsh -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (2983) ./myspin 4 &
tsh> ./myspin 5
Job [2] (2985) stopped by signal 20
tsh> jobs
[1] (2983) Running ./myspin 4 &
[2] (2985) Stopped ./myspin 5
```

```
han@han-VirtualBox:~/shlab-handout$ make rtest08
./sdriver.pl -t trace08.txt -s ./tshref -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (2995) ./myspin 4 &
tsh> ./myspin 5
Job [2] (2997) stopped by signal 20
tsh> jobs
[1] (2995) Running ./myspin 4 &
[2] (2997) Stopped ./myspin 5
```

可以看到test的结果和rtest的示例结果完全相同，证明实验正确。

1.比较trace09~10执行不同结果，编程实现内建命令bg和fg的do_bgfg()处理函数

```
han@han-VirtualBox:~/shlab-handout$ make test09
./sdriver.pl -t trace09.txt -s ./tsh -a "-p"
#
# trace09.txt - Process bg builtin command
#
tsh> ./myspin 4 &
[1] (3021) ./myspin 4 &
tsh> ./myspin 5
Job [2] (3023) stopped by signal 20
tsh> jobs
[1] (3021) Running ./myspin 4 &
[2] (3023) Stopped ./myspin 5
tsh> bg %2
tsh> jobs
[1] (3021) Running ./myspin 4 &
[2] (3023) Stopped ./myspin 5
han@han-VirtualBox:~/shlab-handout$ make rtest09
./sdriver.pl -t trace09.txt -s ./tshref -a "-p"
#
# trace09.txt - Process bg builtin command
#
tsh> ./myspin 4 &
[1] (3032) ./myspin 4 &
tsh> ./myspin 5
Job [2] (3034) stopped by signal 20
tsh> jobs
[1] (3032) Running ./myspin 4 &
[2] (3034) Stopped ./myspin 5
tsh> bg %2
[2] (3034) ./myspin 5
tsh> jobs
[1] (3032) Running ./myspin 4 &
[2] (3034) Running ./myspin 5
```

```
han@han-VirtualBox:~/shlab-handout$ make test10
./sdriver.pl -t trace10.txt -s ./tsh -a "-p"
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
[1] (3049) ./myspin 4 &
tsh> fg %1
tsh> jobs
[1] (3049) Running ./myspin 4 &
tsh> fg %1
tsh> jobs
[1] (3049) Running ./myspin 4 &
han@han-VirtualBox:~/shlab-handout$ make rtest10
./sdriver.pl -t trace10.txt -s ./tshref -a "-p"
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
[1] (3059) ./myspin 4 &
tsh> fg %1
Job [1] (3059) stopped by signal 20
tsh> jobs
[1] (3059) Stopped ./myspin 4 &
tsh> fg %1
tsh> jobs
```

trace09是在bg %2命令之后的反应有所不同。trace10是在fg %1命令之后的反应有所不同。

bg指令可以将一个在后台暂停的命令变成继续执行，fg可以将后台中的命令调制到前台继续运行，而加上%表示作业，不加表示进程。所以可以进行编程得到如下代码：

```c
void do_bgfg(char **argv)
{
    int bg = 0;
    int jid;
    pid_t pid;
    struct job_t *job;
    char *arg = argv[1];
    if(!arg || arg[0]=='&'){
        printf("%s command requires PID or %%jobid argument\n", argv[0]);
        return;
    }
    if(arg[0]!='%' && (arg[0]<'0' || arg[0]>'9')){
        printf("%s: argument must be a PID or %%jobid\n", argv[0]);
        return;
    }
    //get job
    if(arg[0]=='%'){//JID
        jid = atoi(++arg);
        if(!jid){
            printf("%s: No such job\n", argv[1]);
            return;
        }
        job = getjobjid(jobs, jid);
        if(!job){
            printf("%s: No such job\n", argv[1]);
            return;
        }
        pid = job->pid;
    }else{//PID
        pid = atoi(arg);
        job = getjobpid(jobs, pid);
        if(!job){
            printf("(%d): No such process\n", pid);
            return;
        }
    }
     //区分bg还是fg
    if(!strcmp(argv[0],"bg")){
        bg = 1;
    }
    if(bg){
        kill(-pid, SIGCONT);
        job->state = BG;
        printf("[%d] (%d) %s", job->jid, job->pid, job->cmdline);
    }else{
        kill(-pid, SIGCONT);
        job->state = FG;
        waitfg(pid);
    }
    return;
}
```

2.验证trace09~10

```
#
# trace09.txt - Process bg builtin command
#
/bin/echo -e tsh> ./myspin 4 \046
./myspin 4 &

/bin/echo -e tsh> ./myspin 5
./myspin 5

SLEEP 2
TSTP

/bin/echo tsh> jobs
jobs

/bin/echo tsh> bg %2
bg %2

/bin/echo tsh> jobs
jobs
```

```
#
# trace10.txt - Process fg builtin command.
#
/bin/echo -e tsh> ./myspin 4 \046
./myspin 4 &

SLEEP 1
/bin/echo tsh> fg %1
fg %1

SLEEP 1
TSTP

/bin/echo tsh> jobs
jobs

/bin/echo tsh> fg %1
fg %1

/bin/echo tsh> jobs
jobs
```

trace09跟踪文件：增加了一条bg %2指令，即将作业号为2的作业调制后台运行；

trace10跟踪文件：增加了一条fg %1指令，即将作业号为1的作业调制前台运行。

测试结果如下：和标准结果一致，说明代码运行正确。

```
han@han-VirtualBox:~/shlab-handout$ make test09
./sdriver.pl -t trace09.txt -s ./tsh -a "-p"
#
# trace09.txt - Process bg builtin command
#
tsh> ./myspin 4 &
[1] (3153) ./myspin 4 &
tsh> ./myspin 5
Job [2] (3155) stopped by signal 20
tsh> jobs
[1] (3153) Running ./myspin 4 &
[2] (3155) Stopped ./myspin 5
tsh> bg %2
[2] (3155) ./myspin 5
tsh> jobs
[1] (3153) Running ./myspin 4 &
[2] (3155) Running ./myspin 5
```

```
han@han-VirtualBox:~/shlab-handout$ make test10
./sdriver.pl -t trace10.txt -s ./tsh -a "-p"
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
[1] (3126) ./myspin 4 &
tsh> fg %1
Job [1] (3126) stopped by signal 20
tsh> jobs
[1] (3126) Stopped ./myspin 4 &
tsh> fg %1
tsh> jobs
```

3.验证trace11~15并解释与记录

1）由trace11.txt可知trace11做的是./mysplit 4创建子进程并将其挂起4秒，而父进程在挂起2秒后发送SIGINT信号使子进程终止，最后调用/bin/ps a指令打印信息。验证结果相同，实现成功。

```
han@han-VirtualBox:~/shlab-handout$ 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] (3245) terminated by signal 2
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:00 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3240 pts/0    S+     0:00 make test11
 3241 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace11.txt -s ./tsh -a "-
p"
 3242 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tsh -a
 -p
 3243 pts/0    S+     0:00 ./tsh -p
 3248 pts/0    R      0:00 /bin/ps a
```

```
han@han-VirtualBox:~/shlab-handout$ 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] (3254) terminated by signal 2
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:07 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3249 pts/0    S+     0:00 make rtest11
 3250 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace11.txt -s ./tshref -a
 "-p"
 3251 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace11.txt -s ./tshref
 -a -p
 3252 pts/0    S+     0:00 ./tshref -p
 3257 pts/0    R      0:00 /bin/ps a
```

2）trace12.txt文件了解到，trace12做的是./mysplit 4创建子进程并将其挂起4秒，再调用sleep指令休眠2秒，接着调用TSTP指令，再调用jobs指令打印当前在后台的作业，最后调用/bin/ps a指令查看各个进程运行信息。验证结果相同。

```
han@han-VirtualBox:~/shlab-handout$ 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] (3287) stopped by signal 20
tsh> jobs
[1] (3287) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:13 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3282 pts/0    S+     0:00 make test12
 3283 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace12.txt -s ./tsh -a "-
p"
 3284 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace12.txt -s ./tsh -a
 -p
 3285 pts/0    S+     0:00 ./tsh -p
 3287 pts/0    T      0:00 ./mysplit 4
 3288 pts/0    T      0:00 ./mysplit 4
 3291 pts/0    R      0:00 /bin/ps a
```

```
han@han-VirtualBox:~/shlab-handout$ 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] (3297) stopped by signal 20
tsh> jobs
[1] (3297) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:14 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3292 pts/0    S+     0:00 make rtest12
 3293 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace12.txt -s ./tshref -a
 "-p"
 3294 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace12.txt -s ./tshref
 -a -p
 3295 pts/0    S+     0:00 ./tshref -p
 3297 pts/0    T      0:00 ./mysplit 4
 3298 pts/0    T      0:00 ./mysplit 4
 3301 pts/0    R      0:00 /bin/ps a
```

3) 前三行是字符打印，然后调用mysplit函数睡眠4秒，再调用sleep函数睡眠2秒，调用TSTP指令后，调用jobs指令打印在前台运行的作业，再调用/bin/ps a指令，然后执行fg %1指令将作业1由后台移到前台工作，再调用/bin/ps a指令。验证结果相同。

```
han@han-VirtualBox:~/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] (3308) stopped by signal 20
tsh> jobs
[1] (3308) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:15 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3303 pts/0    S+     0:00 make test13
 3304 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tsh -a "-
p"
 3305 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tsh -a
 -p
 3306 pts/0    S+     0:00 ./tsh -p
 3308 pts/0    T      0:00 ./mysplit 4
 3309 pts/0    T      0:00 ./mysplit 4
 3312 pts/0    R      0:00 /bin/ps a
tsh> fg %1
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:15 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3303 pts/0    S+     0:00 make test13
 3304 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tsh -a "-
p"
 3305 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tsh -a
 -p
 3306 pts/0    S+     0:00 ./tsh -p
 3315 pts/0    R      0:00 /bin/ps a
```

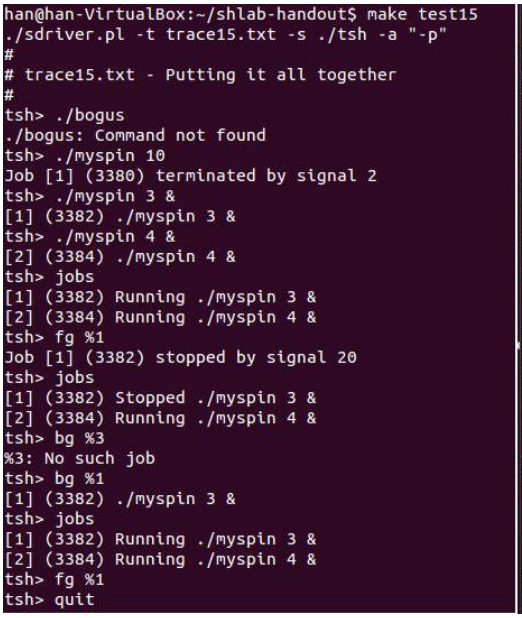
```
han@han-VirtualBox:~/shlab-handout$ make rtest13
./sdriver.pl -t trace13.txt -s ./tshref -a "-p"
#
# trace13.txt - Restart every stopped process in process group
#
tsh> ./mysplit 4
Job [1] (3322) stopped by signal 20
tsh> jobs
[1] (3322) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:16 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3317 pts/0    S+     0:00 make rtest13
 3318 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tshref -a
 "-p"
 3319 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tshref
 -a -p
 3320 pts/0    S+     0:00 ./tshref -p
 3322 pts/0    T      0:00 ./mysplit 4
 3323 pts/0    T      0:00 ./mysplit 4
 3326 pts/0    R      0:00 /bin/ps a
tsh> fg %1
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
  858 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
  863 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
  879 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
  881 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
  891 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
  989 tty7     Ss+    3:16 /usr/bin/X :0 -auth /var/run/lightdm/root/:0 -noliste
n tcp vt7 -novtswitch
 1533 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
 2167 pts/0    Ss     0:00 bash
 3317 pts/0    S+     0:00 make rtest13
 3318 pts/0    S+     0:00 /bin/sh -c ./sdriver.pl -t trace13.txt -s ./tshref -a
 "-p"
 3319 pts/0    S+     0:00 /usr/bin/perl ./sdriver.pl -t trace13.txt -s ./tshref
 -a -p
 3320 pts/0    S+     0:00 ./tshref -p
 3329 pts/0    R      0:00 /bin/ps a
```

4) 处理输入未实现命令、fg和bg参数不正确等错误情况和将作业1、2移到前台再移到后台相关指令。验证结果相同。

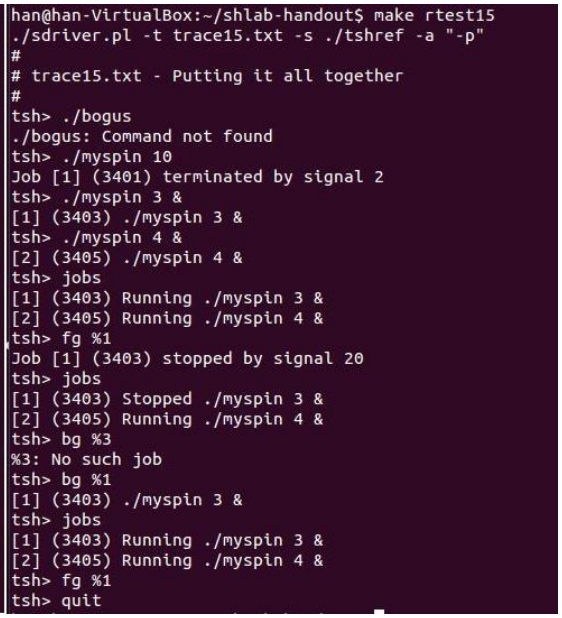
```
han@han-VirtualBox:~/shlab-handout$ make test14
./sdriver.pl -t trace14.txt -s ./tsh -a "-p"
#
# trace14.txt - Simple error handling
#
tsh> ./bogus
./bogus: Command not found
tsh> ./myspin 4 &
[1] (3341) ./myspin 4 &
tsh> fg
fg command requires PID or %jobid argument
tsh> bg
bg command requires PID 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] (3341) stopped by signal 20
tsh> bg %2
%2: No such job
tsh> bg %1
[1] (3341) ./myspin 4 &
tsh> jobs
[1] (3341) Running ./myspin 4 &
```

```
han@han-VirtualBox:~/shlab-handout$ 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] (3360) ./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] (3360) stopped by signal 20
tsh> bg %2
%2: No such job
tsh> bg %1
[1] (3360) ./myspin 4 &
tsh> jobs
[1] (3360) Running ./myspin 4 &
```

5) 前三行是字符打印，然后调用myspin函数睡眠10秒，再调用sleep函数睡眠2秒，调用INT指令后，后台调用myspin指令睡眠3秒，再睡眠四秒；再调用fg %1指令，将作业1从后台移到前台运行，调用sleep函数睡眠2秒，调用TSPT和jobs指令打印相关信息，bg %3指令将作业3在后台运行，bg %1指令将作业1在后台运行，jobs指令打印相关信息，fg %1指令将后台作业移到前台，最后quit指令退出。验证结果相同。



```
han@han-VirtualBox:~/shlab-handout$ 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] (3380) terminated by signal 2
tsh> ./myspin 3 &
[1] (3382) ./myspin 3 &
tsh> ./myspin 4 &
[2] (3384) ./myspin 4 &
tsh> jobs
[1] (3382) Running ./myspin 3 &
[2] (3384) Running ./myspin 4 &
tsh> fg %1
Job [1] (3382) stopped by signal 20
tsh> jobs
[1] (3382) Stopped ./myspin 3 &
[2] (3384) Running ./myspin 4 &
tsh> bg %3
%3: No such job
tsh> bg %1
[1] (3382) ./myspin 3 &
tsh> jobs
[1] (3382) Running ./myspin 3 &
[2] (3384) Running ./myspin 4 &
tsh> fg %1
tsh> quit
```



```
han@han-VirtualBox:~/shlab-handout$ 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] (3401) terminated by signal 2
tsh> ./myspin 3 &
[1] (3403) ./myspin 3 &
tsh> ./myspin 4 &
[2] (3405) ./myspin 4 &
tsh> jobs
[1] (3403) Running ./myspin 3 &
[2] (3405) Running ./myspin 4 &
tsh> fg %1
Job [1] (3403) stopped by signal 20
tsh> jobs
[1] (3403) Stopped ./myspin 3 &
[2] (3405) Running ./myspin 4 &
tsh> bg %3
%3: No such job
tsh> bg %1
[1] (3403) ./myspin 3 &
tsh> jobs
[1] (3403) Running ./myspin 3 &
[2] (3405) Running ./myspin 4 &
tsh> fg %1
tsh> quit
```

总结：

本次实验总体来说难度不大，需要实现job、fg、bg、kill四个内建命令和对执行本地程序的支持，并且还要处理好SIGCHLD、SIGINT、SIGTSTP这几个信号。因为开头没有理解shell的流程，对很多命令的运行情况不熟悉，所以对出来的结果很不解。不过后来慢慢理清了运行顺序和情况、信号的作用、进程的运行情况和周期以及运行过程。这个实验让我对shell和进程的理解进一步加深。