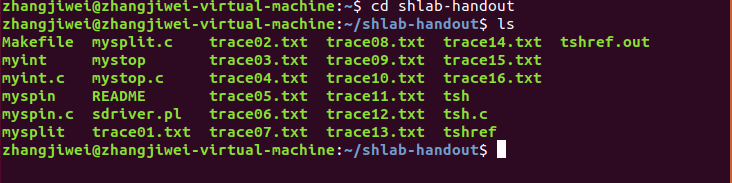
**日志2.8**

**【20%】学会编译tsh.c 调用tsh文件trace.txt的功能验证方法。**

调用系统命令，查看文件。



根据 shlab.pdf 的说明，编译 tsh.c 只需键入指令： unix > make

验证 tracexx.txt 的功能可以通过对比 tsh 的与 tshref.out 来验证，

验证 tsh: unix > ./sdriver.pl − t trace01.txt − s ./tsh − a " − p"

或者输入： unix > make test01

验证 tshref： unix > ./sdriver.pl − t trace01.txt − s ./tshref − a " − p"

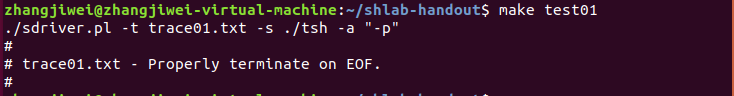
或者输入 unix > make rtest01

通过对比这两个指令得到的结果即可验证自己是否实现了对应功能

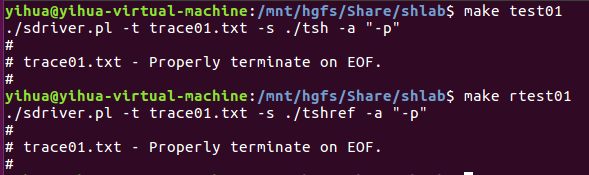
**【20%】用trace01和trace02比较tsh和tshref执行结果并分析。**

验证trace01

输入指令make test01



输入指令make rtest01

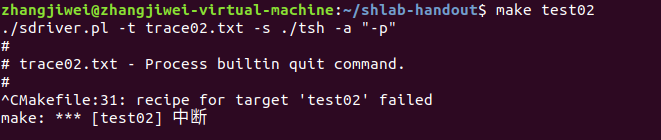


它正确读出文件中数据，然后在读取close的时候，关闭文件，并等待（wait感觉测不出来，貌似直接关闭了，我后来加了很多wait也是一样的

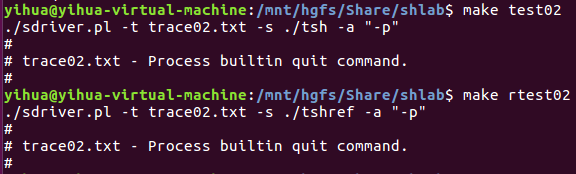
比较：发现效果相同，第一个测试不能进行有效的判别，所以也不能获得任何信息

验证trace02

输入指令 make test02



输入指令 make rtest02

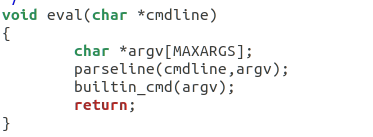


在tsh文件中 需要输入ctrl+c退出

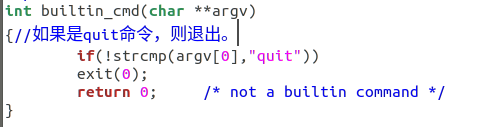
原文件中执行quit指令 目前我们还没有填充函数，tsh不能实现该功能

**【30%】编程实现quit内置命令，补齐文件tsh.c中的函数eval()和函数builtin\_cmd()与quit相关的部分。**

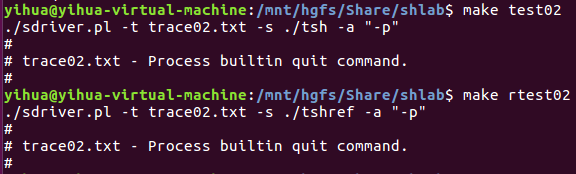
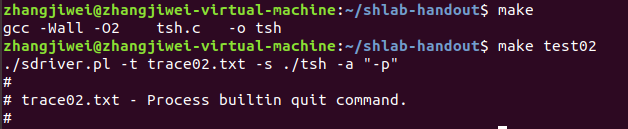
（1）eval()中填充quit部分 首先调用parseline函数，得到具体函数，传参给builtin\_cmd()。



（2）builtin\_cmd()中填充quit部分 使用strcmp函数，判断是否为quit指令，是则退出。

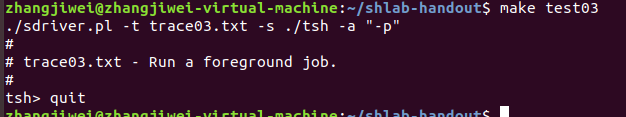


（3）完成trace02功能 直接退出

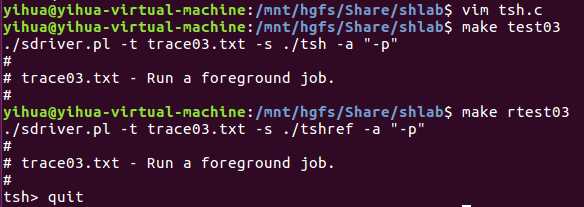


【20%】使用trace03验证quit命令。

输入指令 make test03



输入指令make rtest03



发现：正确的结果多输出了一句tsh> quit 原因是 trace03中使用了echo命令

**【10%】了解eval()与execve执行流程和fork()多进程运行方式。**

eval

eval 函数的作用是评估并解读用户刚刚输入的命令行。如果用户请求了内置 命令（quit, jobs, bg or fg）就立刻执行这个操作。否则 fork 开辟一个子进 程。在子进程中加载并运行该程序。如果一个进程在前台执行，就会在前台进程 执行前一直等待。

execve

execve 函数在当前进程的上下文中加载并运行一个新程序。而且带参数列表 argv 和环境变量列表 envp。当出现错误时，execve 会返回到调用程序。

fork

fork 函数被调用之后会创建一个子进程，同时会有两次返回，一个是子进程 的 pid 一个是 0.两个进程有相同的用户栈，相同的本地变量值，相同的堆，相同 的全局变量值和相同的代码。而且父子进程相互独立，是并发运行的。