1、什么是管道？

2、回顾一下异常的信号机制？？？

3、refresh函数没有像书本那样，注释掉依然可以显示结果。

4、按照书本的代码发现P219实现，发现仍然阻塞在等待函数那里，加上O\_NONBLOCK之后就可以实现了。问题就是，调试了一下代码，为什么阻塞在等待函数那里呢？按道理不应该。反向推测，sigio收到了信号，所以进入了函数。问题是为什么收到信号呢？（再简单验证一下)

做了一下实验，发现还是会阻塞，但是，第一次怎么会进入on\_input里面呢？SIG\_IO这个信号是在什么情况下接受信号进行处理的呢？不懂

现在的问题是：为什么第一次会进入到那个信号？也就是sigio的信号是在什么情况下进行接收的？？？

现在总结一下：

1. 在定时信号的函数没有内容，没有O\_NONBLOCK的情况下：

回车也会被记录到getchar里面，但是会阻塞在getchar那里（仔细想想，其实是阻塞在pause那里），去掉sleep有的时候就会进入main函数。（即输入u+回车进入到定时信号函数里面两次，之后进入main函数parse里面，再次输入q+回车，还是没有退出，但是再次输入别的内容就退出，而且在终端显示这些字符，表明已经退出了，但为什么还停留在程序里面？）

2. printf居然会影响sigio信号。这个需要注意一下。如果在定时信号那里sleep，那么将不会触发return，因为之前test过。

3. 打开O\_NONBLOCK的时候，效果跟没有的是一样的。也就是说，在使用curose库的时候，可能会影响sigio这个信号，所以加上O\_NONBLOCK的时候就会忽略没有结果的内容，达到使用目的。

1. 也就是说，正常情况下在pause那里，输入字符就相应sigio信号，但是有个问题，输入q的时候，为什么不直接退出，而是要继续输入才可以呢？

5、回去看看程序是怎么运行的。

6、fork到底发生什么？这个问题需要深入了解一下。

7、shell程序仍然需要改进的地方：

1. if后面可以跟多个语句，并由最后的那个决定

2. 增加else

3. 嵌套执行if（这个比较困难，思考了一下，这个需要用栈去实现）

4. 如何像常规的shell那样最后才输出呢？（这个比较困难）

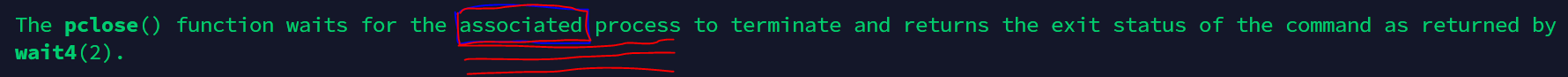
8、回顾一下外部变量和枚举。。。。（无力吐槽）

9、回顾一下编译文件的顺序

10、主机字节序列和网络字节序列的区别（host byte order & network byte order）

11、自己模仿的popen函数mypopen为什么会造成僵尸进程呢？

首先，出现僵尸进程的原因是父进程没有使用wait来接收子进程状态，于是我使用wait，自己的函数myopen就不会出现僵尸进程。查看文档，得知文档如下：



也就是说，pclose是接收“相关”的进程，于是，我做了一个实验，使用原有的popen，再使用wait，接着再使用pclose，就会造成阻塞，于是得证。

至于为什么子进程执行dc会退出呢？如果父进程死亡，子进程是仍然会继续执行（这是孤儿进程），其实仔细想想，如果我关闭管道的输入输出流，那么dc就会退出。

12、守护进程相关搞不太懂。

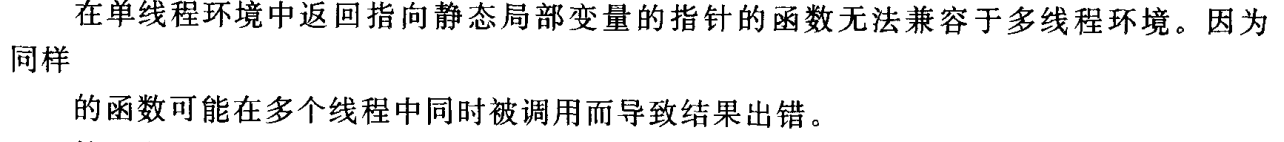
13、书上说信号阻塞的时候是会被丢弃的，可是我试验了一下，并不会被丢弃。

14、accept阻塞出现信号来的时候的错误处理方法。

15、模仿第十二章编写的服务器程序有bug，但是找不到在哪里，总是出现段错误，但是我跟踪进去又找不到错误，非常非常纳闷。

16、programe12里的threadweb总是出现错误，不知道哪里有bug，太困扰人了。

17、书中有说，



这是为什么呢？

1. 看资料介绍，为了实现互斥锁操作，大多数体系结构都提供了swap或exchange指令，该指令的作用是把寄存器和内存单元的数据相交换，由于只有一条指令，保证了原子性，即使是多处理器平台，访问内存的总线周期也有先后，一个处理器上的交换指令执行时另一个处理器的交换指令只能等待总线周期。现在我们把lock和unlock的伪代码改一下（以x86的xchg指令为例）：



但是，我有个疑问，如果一个线程执行到if的时候，另一个线程执行movb指令，就会导致没有线程获得到锁，这个问题我想不懂。

19、Linux的talk命令无法成功。