**Shell**

一、实验目的

通过补全代码，在一个小shell中实现几个功能，熟悉 Unix 系统调用接口和shell

二、实验内容

[Homework: shell (mit.edu)](https://pdos.csail.mit.edu/6.828/2016/homework/xv6-shell.html)

三、实验步骤与实验结论

6.828shell包含两个部分：解析与实施

（1） 将这些命令粘贴到文件t.sh

ls > y

cat < y | sort | uniq | wc > y1

cat y1

rm y1

ls | sort | uniq | wc

rm y

（2）下载sh.c文件

wget <https://pdos.csail.mit.edu/6.828/2018/homework/sh.c>

（3）编译sh.c，得到a.out

gcc sh.c

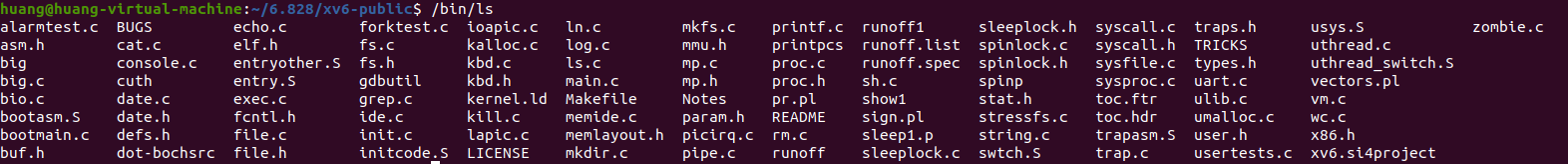
（4）执行简单命令功能

如果要执行一个命令，应该使用exec这个系统调用。这个系统调用有两个参数:

execv(ecmd.argv[0], ecmd.argv); argv[0]是程序名字，argv是参数

execv会停止执行当前的进程，并且以argv[0]应用进程替换被停止执行的进程，进程ID没有改变。

就可以使用/bin/ls来打印目录



（5） I/O重定向功能

每个文件都会有一个自己的文件描述符fd，用于被其他文件使用，还会有一张fd表，用于记录与其他文件的关系，fd从零开始分配

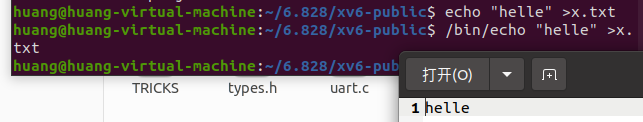
0 -> 标准输入

1 -> 标准输出

2 -> 错误输出

而后，每次open时， 总是从未分配的最小值开始

所以，实现I/O的重定向，只需先关闭对应的fd，再通过open打开需要重定向目标文件



（6）管道

管道实现的是进程之间的通信，是内核当中一段缓存

函数：

pipe(int p[])

建立一个缓冲区，并把缓冲区通过fd形式给程序调用。将p[0]修改为缓冲区的读取端， p[1]修改为缓冲区的写入端。

dup(int old\_fd)

产生并返回与old\_fd指向同一文件的fd。产生的fd总是空闲的最小fd。

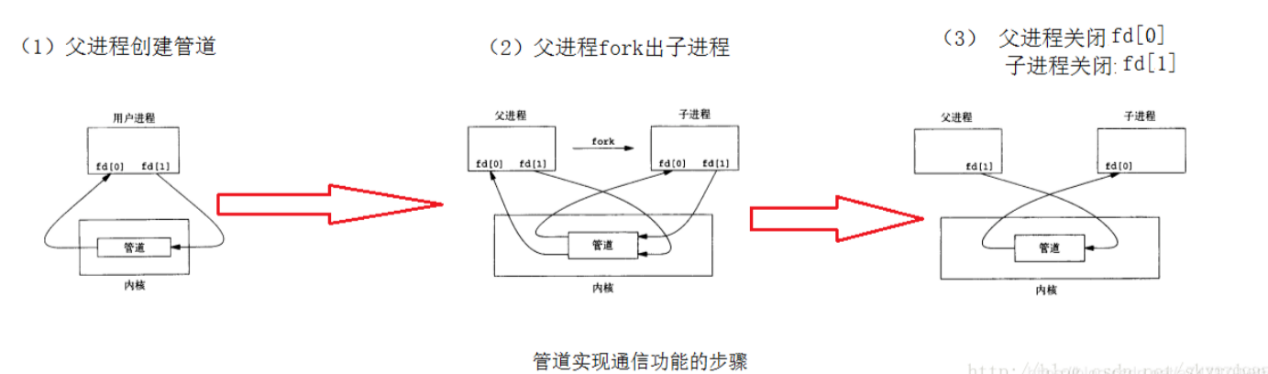
fork()

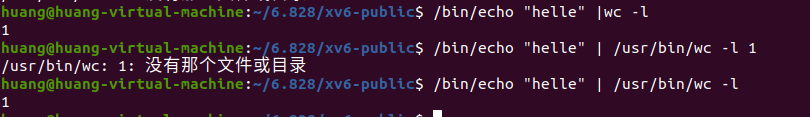
系统调用创建新的进程，来执行命令

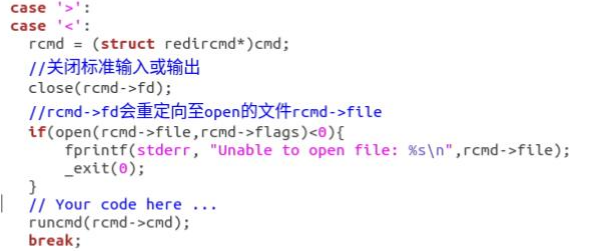
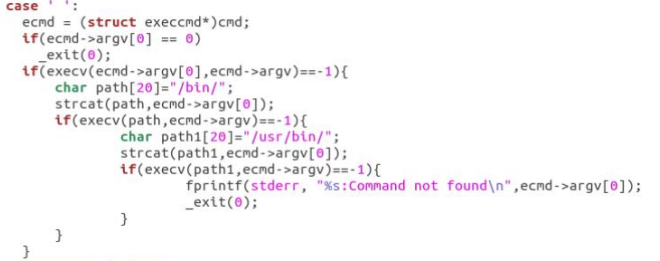
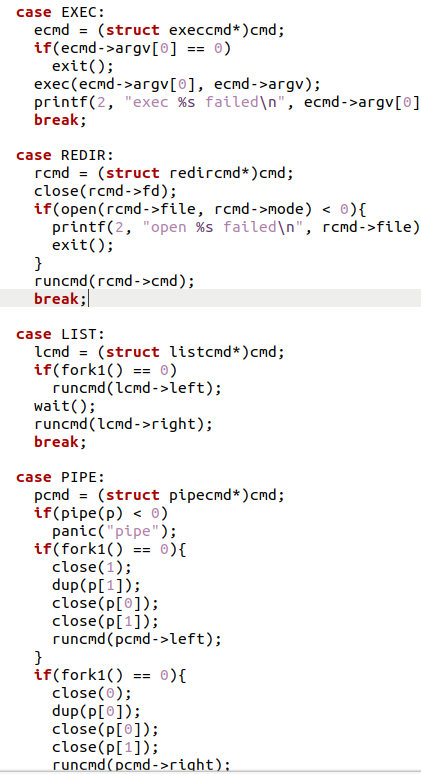
父进程wait两个子进程

子进程1: read from stdin(0) write to right end of pipe(p[1])

子进程2: read from left end of pipe(p[0]) write to right stdout(1)







四、反思总结

开始不理解管道如何运行，后来通过图示明白了流程