华东师范大学数据学院上机实践报告

课程名称： 操作系统　　**年级：**2019 级　　**上机实践成绩：**

指导教师：翁楚良　　**姓名：**周辛娜

上机实践名称： shell 及系统调用　　**学号：** 10195501442　　**上机实践日期：**

上机实践编号：02

一、 目的

1. 安装 MINIX 操作系统(Version:3.3)
2. 学习 Shell，系统编程，实现一个基本的 Shell

二、 内容与设计思想

Shell 主体：

Shell 内置命令：

1. cd
2. History
3. exit
4. mytop

Program 命令：

1.运行程序：利用 fork 调用创建进行子进程，利用 execvp 调用将 minix 程序装 载到该进程，并赋予运行参数，最后 Shell 利用 wait/waitpid 调用等待子进程结束。

2. 重定向

3.管道：若有 n 个子进程组成管道流，Shell 在 fork 前先用 pipe 调用创建 n-1 对 管道描述符，关闭不需要的读写端。Shell 运行 fork 后，每个子进程利用 dup 将前一个管道的读端映射到标准输入，后一个管道的写端映射到标准输出。

4. 后台运行

三、 使用环境

MINIX Vmware 虚拟机软件

四、 实验过程

本次实验主要包含三大部分：shell 主体、内置命令（cd、history、mytop、exit）、program 命令

一．Shell 主体

主体结构是一个 while 循环。Shell 将输入行分解成单词序列，根据命令名称分为二类分别处理，即 shell 内置命令（例如 cd，history，exit）和 program 命令（例如/bin/ 目录下的 ls，grep 等）。识别为 shell 内置命令后，执行对应操作。接受 program 命令后，利用 MINIX 自带的程序创建一个或多个新进程，并等待进程结束。如果末尾包含&参数，Shell 可以不等待进程结束，直接返回。

```
int parseline(char **, char **, int *);         //读取命令函数
int builtin_cmd(char *, char **);               //内建命令函数
int eval(char **, int, struct eval_function *); //语法分析函数
```

这三个函数构成了基本的 shell 框架。

二、内置命令 cd、history、mytop、exit

cd：利用 chdir 系统调用可以移动 Shell 的工作目录。

history：保存 Shell 每次的输入行，打印所需字符串即可。

exit：退出 Shell 的 while 循环，结束 Shell 的 main 函数。exit(0)即可。

mytop：进入/proc/meminfo 中，查看内存信息，页面大小 pagesize，总页数量 total，空闲页数量 free，最大页数量 largest ，缓存页数量 cached 。可计算内存大小：(pagesize * total)/1024，同理算出其他页内存大小。 得到总体内存大小，空闲内存大小，缓存大小。

进入/proc/kinfo 中，查看进程和任务数量。 进入/proc/pid/psinfo 中，例如/proc/107/psinfo 文件中，查看 pid 为 107 的进程信息。每 个参数对应含义依次是：版本 version，类型 type，端点 endpt，名字 name，状态 state，阻塞状态 blocked，动态优先级 priority，滴答 ticks，高周期 highcycle，低 周期 lowcycle，内存 memory，有效用户 ID effuid，静态优先级 nice 等。其中会用 到的参数有：类型，状态，滴答。进程时间 time=ticks/ (u32_t)60。到进程和任务总数量 total_proc，对每一个 proc 的 ticks 累加得到总体 ticks，再计算空闲的 ticks，最终可得到 CPU 使用百分比。

三、program 命令

1.重定向：ls -a -l > result.txt，ls -a -l 是命令 1，应该将对应的执行结果作为输入，输入到 out_file，result.txt 是 out_file。minix 为每个进程赋予键盘输入和控制台输出的文件描述符默认为 0 和 1。子进程装载程序前，利用 close(0 or 1)将默认输入或者输出关闭，再调用 dup(fd)将某个打开文件的文件描述 fd 映射到标准输入或输出。

grep a < result.txt，grep a 是命令 1，result.txt 是 in_file，将 result.txt 中的内容根据 a 进行搜索。具体实现应该是先 open result.txt，得到一个指向 result.txt 的文件描述符，然后要把标准输入指向 result.txt，之后便可以执行命令 1.

2.管道 ls -a -l | grep a，ls -a -l 是命令 1，grep a 是命令 2，按照一般的逻辑，读写通道都调整好以后，只需父进程、子进程各自去执行，正常执行完了以后进程都会终止。但由于这是一个 shell，shell 父进程除非遇到 exit 是不能终止的，所以必须让管道程序成为父进程里的子进程，即在父进程中再 fork 一个子进程，在这个子进程中执行命令 1，原先 fork 出来的子进程执行命令 2，这样，大体框架便出来了。

对于具体的读写通道，谁管谁开，具体来看，对于命令 1，关闭读端，关闭标准输出，再用 dup，将标准输出指向写端，再关闭管道写端，最后执行 execl，当然，用 dup2 也可以，dup2(fd[1],1)，再关 fd[1]。同理，对于 shell 子进程，要执行 grep a，读取管道传过来的数据，然后根据 a 的内容进行查找，所以不需要写端，需要关闭写端，关闭标准输入，再用 dup 使标准输入指向管道读端，最后关掉管道读端，执行命令 1。

3.后台运行 子进程的标准输入、输出映射成 /dev/null。再调用 signal(SIGCHLD,SIG_IGN)，使得 minix 接管此进程。Shell 可以避免调用 wait/waitpid 直接运行下一条命令。

以下为测试结果：

1. cd testing

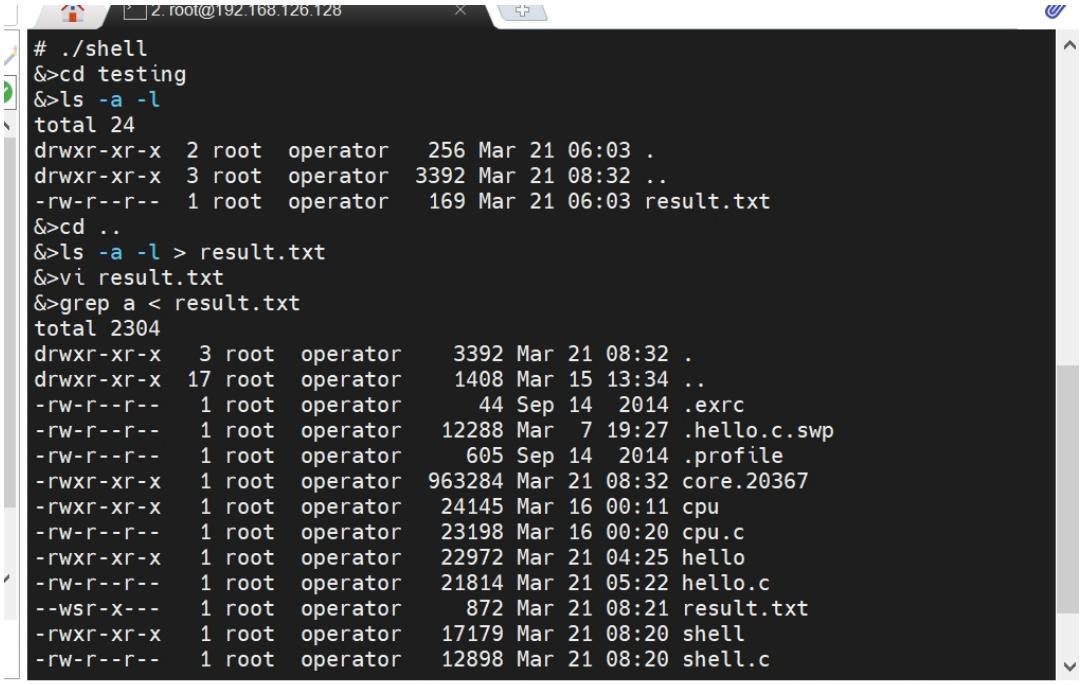
2.ls -a -l

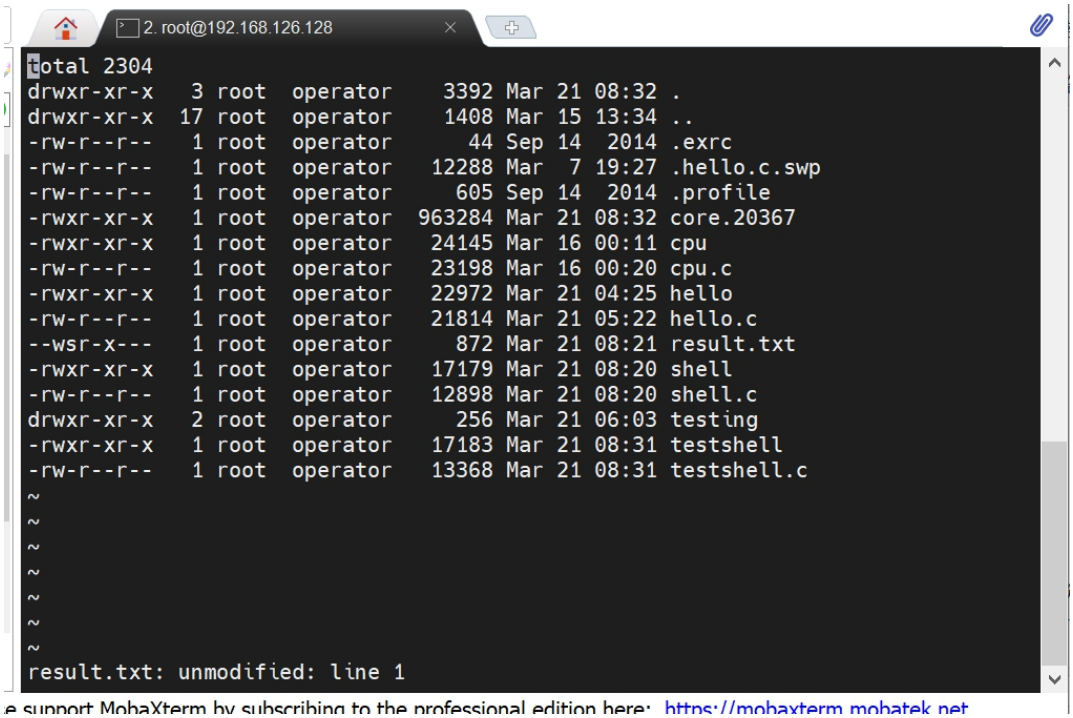
3,cd ..

4.ls -a -l > result.txt

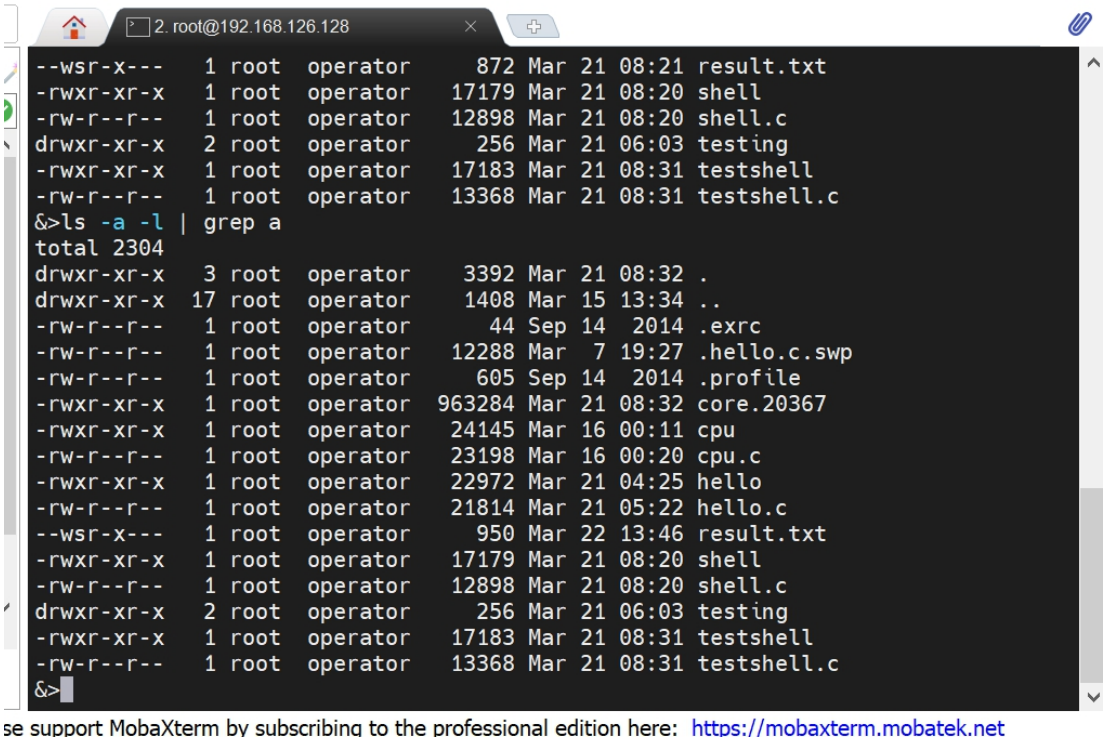
5.vi result.txt

6.grep a < result.txt





7.ls -a -l | grep a

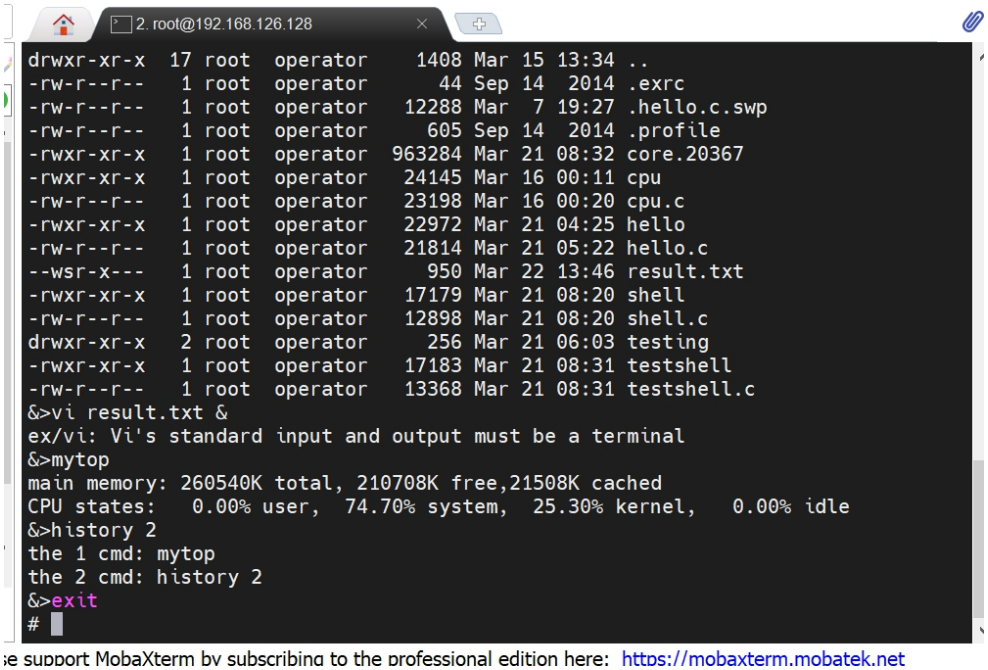


8.vi result.txt &

9.mytop

10.history n

11.exit



五、总结

熟悉了 shell 总体的设计思想以及系统调用的使用方式，对于 shell 作为命令解析器有了更深刻的理解。