# 华东师范大学数据科学与工程学院实验报告

| | | |
|---|---|---|
| **课程名称：**操作系统 | **年级：**2021 | **上机实践成绩：** |
| **指导教师：**翁楚良 | **姓名：**唐小卉 | **学号：**10215501437 |
| **上机实践名称：** lab1 | | **上机实践日期：**2023.3.2 |
| **上机实践编号：** | **组号：** | **上机实践时间：** |

## 一、实验目的

学习 Shell，系统编程，实现一个基本的 Shell。

## 二、实验任务

Shell 能解析的命令行如下：

1. 带参数的程序运行功能。

program arg1 arg2 … argN

2. 重定向功能，将文件作为程序的输入/输出。

1. “>”表示覆盖写

program arg1 arg2 … argN > output-file

2. “>>”表示追加写

program arg1 arg2 … argN >> output-file

3. “<”表示文件输入

program arg1 arg2 … argN < input-file

Shell 能解析的命令行如下：

3. 管道符号“|”

，在程序间传递数据。

programA arg1 … argN | programB arg1 … argN

4. 后台符号& ,表示此命令将以后台运行的方式执行。

program arg1 arg2 … argN &

5. 工作路径移动命令 cd。

6. 程序运行统计 mytop。

7. shell 退出命令 exit。

8. history n 显示最近执行的 n 条指令。

## 三、使用环境

虚拟机：MINIX3，物理机：Windows11

## 四、实验过程

1.实验思路:

(1)内置命令:

• cd:利用系统调用 chdir 函数改变目录，调用 getcwd 函数获取当前工作目录。

• exit:退出 Shell 的 while 循环。

• history:将 Shell 中输入的命令行用二维数组存储，根据参数输出相应的历史命令。

• mytop:在 minix 系统/proc 文件夹中通过 fopen/fscanf 获取进程信息，输出内存使用情况和 CPU 使用百分比。

(2) program 命令:

• 重定向 覆盖写>:调用 open 函数得到文件描述符，清空文件内容，调用 dup2 (fd.

1)函数将文件描述符映射到标准输出。

• 重定向 追加写>>:调用 open 函数得到文件描述符，保留文件内容，调用 dup2 (fd.

1)函数将文件描述符映射到标准输出。

• 重定向 文件输入<:调用 open 函数得到文件描述符，调用 dup2 (fd，0)函数将文件描述符映射到标准输入。

• 后台运行:调用 signal(SIGCHLD,SIG IGN)，将子进程的标准输入、输出映射到/dev/null，Shel1 无需等待子进程结束。

• 管道:调用 pipe 函数创建管道，在子进程中调用 dup2(fd[1],1)函数将管道写端映射到标准输出，进程的输出写入管道。在父进程中，等待子进程结束并回收，调用 dup2(fd01,0)函数将管道读端 fd[0]映射到标准输入，从管道中读入数据并执行。

2.代码讲解：

（1）参数设置：

宏定义中设置了最大输入命令字符数，每条指令的最大长度，历史命令数量（用于hitory），历史命令记录。

（2）Shell 主体：

```
int main(int argc, char **argv)
{
    char c;
    char cmdline[MAXLINE];

    while (1)
    {
        path = getcwd(NULL, 0);
        printf("10215501437Tangxiaohui_shell>%s# ", path);
        fflush(stdout);
        if (fgets(cmdline, MAXLINE, stdin) == NULL)
        {
            continue;
        }
        for (int i = 0; i < M; i++)
        {
            his[his_cnt][i] = cmdline[i];
        }
        his_cnt = his_cnt + 1;
        doCommand(cmdline);
        fflush(stdout);
    }
    exit(0);
}
```

Shell 主体结构是一个 while 循环，不断地接受用户键盘输入行并给出反馈。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。

（3）函数解释：

doCommand 函数：

该函数会调用 parseline 函数获取命令和参数。如果发现是内置函数就会直接返回，如果是其他命令就会执行 builtin_cmd 函数。

Case0：没有出现重定向，管道，后台运行命令。Fork 子进程 execvp 运行

Case1：>。重定向输出，在子进程中调用 open 函数，调用 dup 函数，调用 execvp 执行重定向符号前的指令。

Case2：<。重定向输入。

Case3：|。子进程中调用 pipeline 函数实现管道。

Case4：&。Fork 子进程，调用 signal 让 minix 接管进程，再调用 open，最后 execvp 执行命令。

Case5：>>。重定向追加写。

**（4）parseline 函数**

解析命令行，获得命令和参数，如果最后一个字符是&会判断是否是后台命令。

（5）builtin_cmd 函数

1. cd：因为 Shell 也是一个程序，启动时 minix 会分配一个当前工作目录，利用 chdir 系统调用可以移动 Shell 的工作目录。

2. history：保存 Shell 每次的输入行，打印所需字符串即可。

3. exit：退出 Shell 的 while 循环，结束 Shell 的 main 函数。

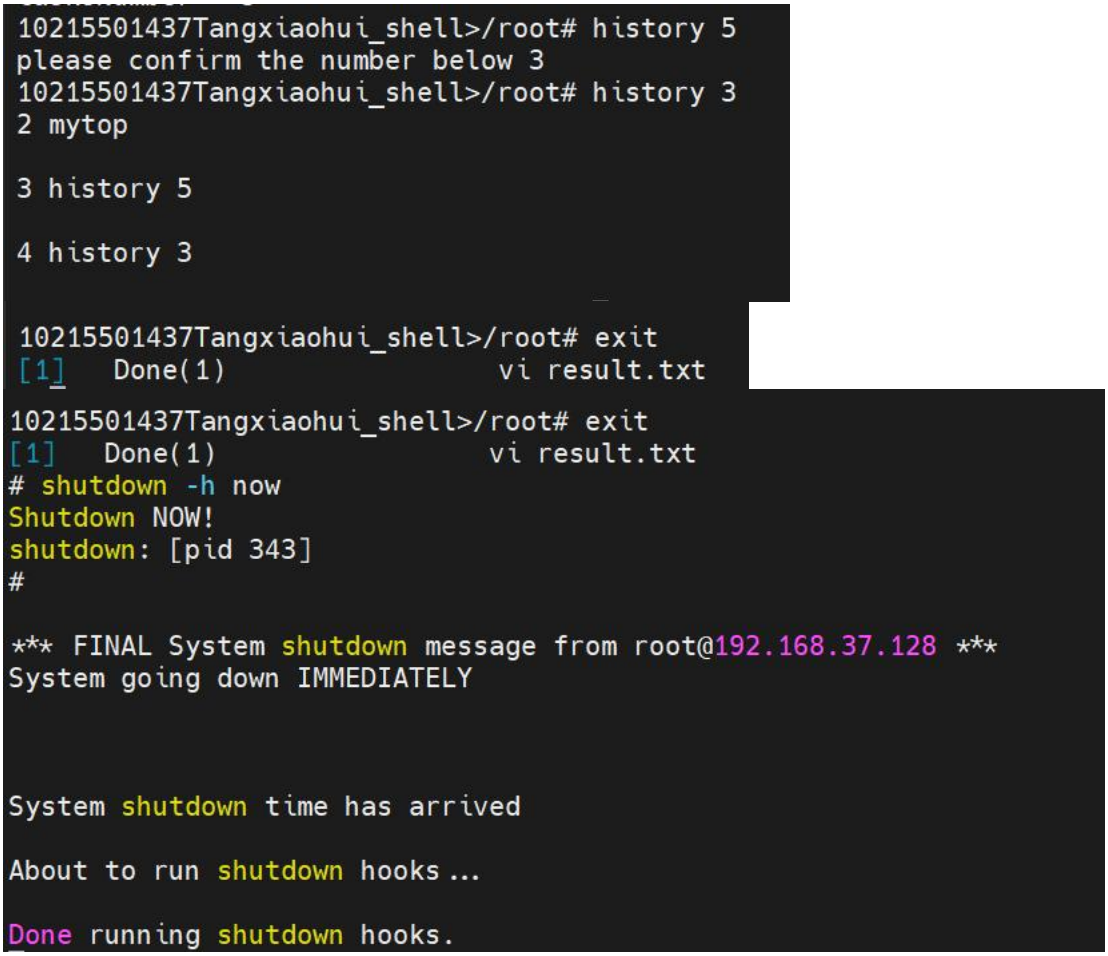
（6）pipe_line 管道函数

调用 pipe 函数创建一个管道 fd[2]，fork 一个子进程，关闭管道读端 fd[0]和文件描述符 1，fd[1]管道写入端，映射到标准输出 1，关闭写端避免堵塞，执行前部分指令，结果输出到管道，父进程中关闭管道读端 fd[1]和文件描述符 0，fd[0]管道读入端，映射到标准输入 0,关闭读端避免堵塞，等待子进程结束，继续执行。

mytop 函数借鉴了博客园的代码。

```
# clang main.c -o main.o
# ./main.o
10215501437Tangxiaohui_shell>/root#
10215501437Tangxiaohui_shell>/root# ls -a -l
total 88
drwxr-xr-x   2 root  operator    576 Mar  9 14:19 .
drwxr-xr-x  17 root  operator   1408 Mar  9 14:17 ..
-rw-r--r--   1 root  operator     44 Sep 14  2014 .exrc
-rw-r--r--   1 root  operator    605 Sep 14  2014 .profile
-rw-r--r--   1 root  operator   9875 Mar  9 14:13 main.c
-rwxr-xr-x   1 root  operator  13239 Mar  9 14:17 main.o
10215501437Tangxiaohui_shell>/root#
```

```
total 88
drwxr-xr-x   2 root  operator    576 Mar  9 14:21 .
drwxr-xr-x  17 root  operator   1408 Mar  9 14:17 ..
-rw-r--r--   1 root  operator     44 Sep 14  2014 .exrc
-rw-r--r--   1 root  operator    605 Sep 14  2014 .profile
-rw-r--r--   1 root  operator   9875 Mar  9 14:13 main.c
-rwxr-xr-x   1 root  operator  13239 Mar  9 14:17 main.o
---sr-S--t   1 root  operator      0 Mar  9 14:21 result.txt
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
result.txt: unmodified, readonly: line 1
10215501437Tangxiaohui_shell>/root# ls -a -l > result.txt
10215501437Tangxiaohui_shell>/root# vi result.txt
10215501437Tangxiaohui_shell>/root# grep a < result.txt
filename=result.txt
drwxr-xr-x   2 root  operator    576 Mar  9 14:21 .
drwxr-xr-x  17 root  operator   1408 Mar  9 14:17 ..
-rw-r--r--   1 root  operator     44 Sep 14  2014 .exrc
-rw-r--r--   1 root  operator    605 Sep 14  2014 .profile
-rw-r--r--   1 root  operator   9875 Mar  9 14:13 main.c
-rwxr-xr-x   1 root  operator  13239 Mar  9 14:17 main.o
---sr-S--t   1 root  operator      0 Mar  9 14:21 result.txt
10215501437Tangxiaohui_shell>/root# ls -a -l | grep a
total 96
drwxr-xr-x   2 root  operator    576 Mar  9 14:21 .
drwxr-xr-x  17 root  operator   1408 Mar  9 14:17 ..
-rw-r--r--   1 root  operator     44 Sep 14  2014 .exrc
-rw-r--r--   1 root  operator    605 Sep 14  2014 .profile
-rw-r--r--   1 root  operator   9875 Mar  9 14:13 main.c
-rwxr-xr-x   1 root  operator  13239 Mar  9 14:17 main.o
---sr-S--t   1 root  operator    404 Mar  9 14:22 result.txt
10215501437Tangxiaohui_shell>/root# vi result.txt&
10215501437Tangxiaohui_shell>/root# mytop
totalMemory  is 522684 KB
freeMemory   is 458104 KB
cachedMemory is 34108 KB
processNumber = 256
tasksNumber = 5
```




## 五、总结

第一次写 shell 有很多的不足和缺点。实验过程是非常艰难的，对于 fork 和信号的处理也不够到位。做完本次实验之后，对 shell 的理解还是不够充分，尤其是 mytop，目前完全的处在我的能力之外了。目前还在攻读 Unix 环境高级编程的用法。