

---

# 实验1 Shell编程

汪楚文 2018202114 04/12/2020

Copyright © 2020- by Wangchuwen. All rights reserved

---

## 实验概述

### [实验目的]

- 掌握编写Linux shell脚本程序的基本方法
- 通过编写Linux shell脚本程序对进程有更深刻的理解

### [基本要求]

编写一个shell脚本程序，功能是输入一个数字，给出系统中以该数字为pid的进程详情,将详情输出到一个文件中。

### [具体要求]

- 执行脚本程序时，将数字作为一个输入；
- 要有基本注释

### [注意问题]

- (1) shell命令行参数；
- (2) shell脚本变量；
- (3) shell的echo、exit、for、if等语句；

### [进一步要求]

- (1) 如何得到进程的详细信息，ps命令只能得到部分信息
- (2) 如果没有以该数字为pid的进程程序该如何处理
- (3) 有一个菜单提供给用户输入
- (4) 添加其他与进程相关的功能，例如kill一个进程

**【重要】**本实现的全部要求都已实现。其中，对于进一步要求，实现方法如下：

- (1) 通过ps aux来得到详细信息
- (2) 输出内存中无\$pid进程，详见下文实验截图（三）
- (3) 菜单界面如下文实验截图（一）
- (4) 本shell程序具有kill \$pid 功能

---

## 实验内容

### [实验环境]

Ubuntu X86\_64

### [实验思路]

根据实验要求，本shell程序的设计如下：

### 一.函数模块

myps函数实现在shell输出目前在内存中运行的全部进程，为了得到进程的详细信息，故在ps命令的基础上，加上了参数-aux

checkpid函数通过awk来检测ps出来的进程是否符合要求，如符合，即将其输出到output.txt文件中，并在shell中输出"👉 已将\$pid的详细状态信息写入output.txt，简要信息如上"。如果内存中不存在符合查询要求的进程，则在shell输出"👎 内存中无\$pid进程"。

killpid函数通过kill -s命令来kill对应进程，如无报错信息说明已成功执行kill命令，输出"✅ 已kill进程\$pid"，否则输出"❌ 操作不允许，kill失败"。

menu函数为菜单设计，其中用了色彩和特效来美化menu界面，并添加了表情增加了图形化程度。menu函数被循环调用。

show函数用来查看output文件，使脚本能在运行时查看output文件

prefer函数用来编辑脚本文件，实现自定义

### 二.程序工作原理

脚本的主体为一个while (1) 结构：

当接收到信号0时，执行break跳出循环，实现退出程序；

当接收到信号1时，执行myps函数，在shell中输出全部process；

当接收到信号2时，执行checkpid函数，输入PID并执行checkpid功能；

当接收到信号3时，执行killpid函数，输入PID并执行killpid功能；

当接收到信号4时，执行show函数，此时可查看output内容；

当接收到信号5时，执行prefer函数，可修改shell程序；

当接收当其他信号时，输出"🙅 Sorry, wrong selection"

**【重要】**执行完每一个信号对应的功能后，shell中会输出"**Hit any key to continue**"此时输入任意键值即可返回菜单。

**【重要】**每一个功能模块中都含有clear操作，这样保证了shell界面始终保持整洁

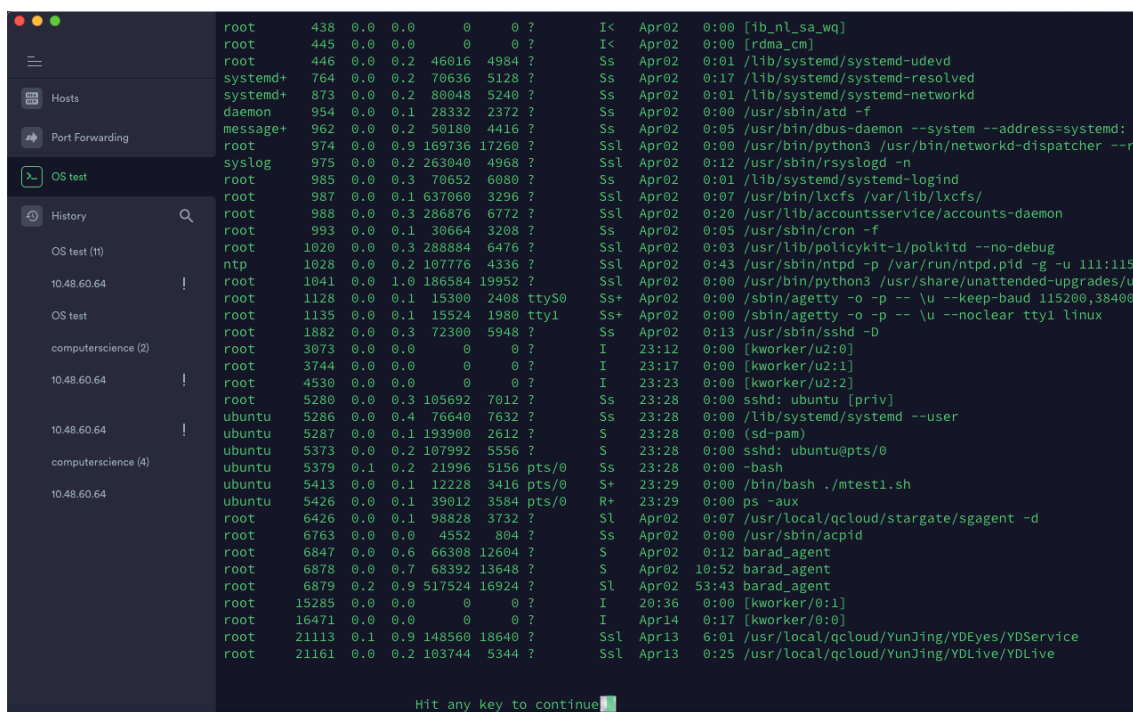
## 实验结果

## 一.菜单界面



“Enter Option: ”

## 二.选择1（显示当前进程）



"Hit any Key to continue"

### 三.选择2查询进程信息，并输入一个伪PID

```
🌲 Input_pid 🌲: 56789
PID TTY          TIME CMD
👤 内存中无56789进程

Hit any key to continue
```

“Hit any Key to continue”

### 四.选择2查询进程信息，并输入一个真实存在的PID

```
🌲 Input_pid 🌲: 21113
PID TTY          TIME CMD
21113 ?          00:06:01 YDService
👤 已将21113的详细状态信息写入output.txt，简要信息如上

Hit any key to continue
```

“Hit any Key to continue”

### 五.选择3，kill进程，并输入一个不可kill的PID

```
🐱 Input_pid 🐱: 21113
./mtest1.sh: line 27: kill: (21113) - Operation not permitted
❌ 操作不允许，kill失败

Hit any key to continue
```

“Hit any Key to continue”

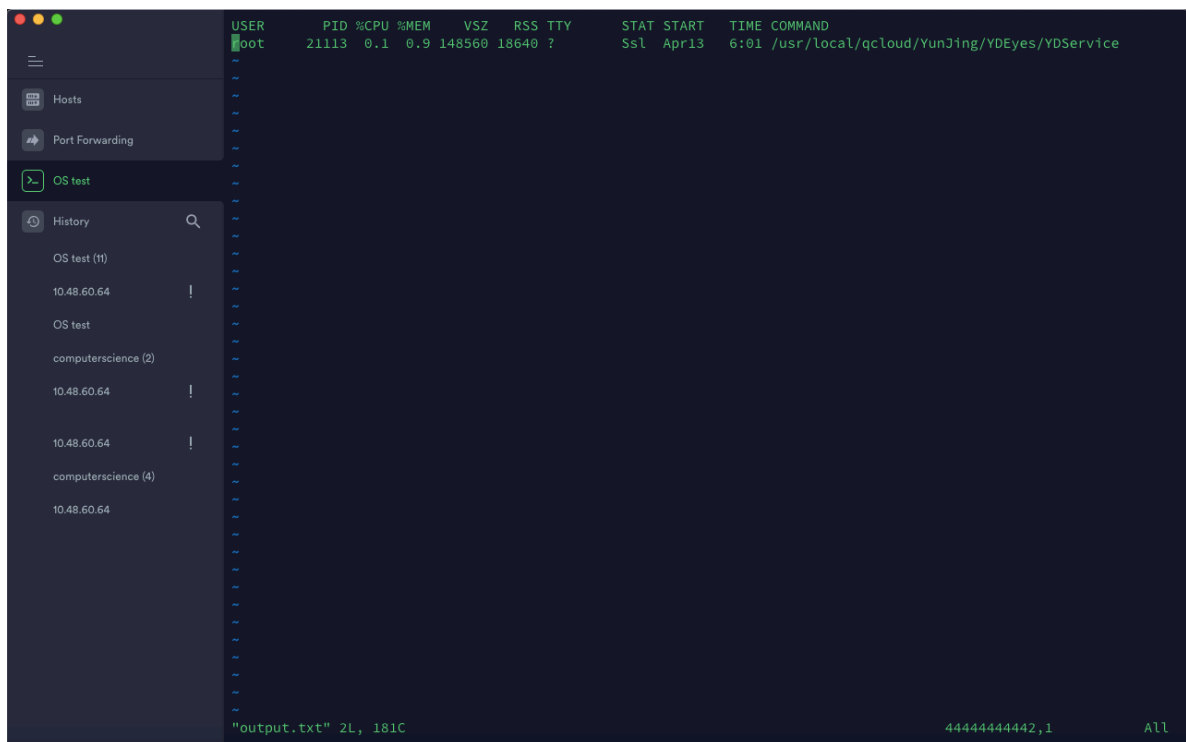
### 六.选择3，kill进程，并输入一个可kill的PID

```
🐱 Input_pid 🐱: 5286
✅ 已kill进程5286

Hit any key to continue
```

“Hit any Key to continue”

### 七.选择4， 查看第四步信息是否已写入output.txt



### 八.选择5，编辑脚本，进入.sh文件



---

## 实验中遇到的问题及解决办法

一.由于当输入的pid不存在时，需要反馈信息给用户。刚写的时候一直想不到怎么去检测输入的pid是否存在。

解决办法：通过ps -p \$pid与if [[ \$? -eq 0 ]]的结合来解决，如果内存中不存在此\$pid，前一个命令便不会正常执行，于是有 \$? -ne 0,故可以在else分支输出“🙄内存中无\$pid进程”来提示用户。

二.在设计交互逻辑的过程中，往往下一步该显示什么比较难以设计。时常用户在只选择一个功能时，程序会把其他功能模块让用户来用0或1来选择是否执行。这种设计非常低效，由于要输入冗余信息，用户体验并不好。

解决办法：将程序进行模块化设计，每一个功能对应一个function。在shell“主函数”中用case \$option in的结构，case的每一个分支对应一个function，这样用户就可以仅选择自己想要执行的模块，且能随时执行任何模块。

三.为了做出更好的菜单效果，通过对echo添加参数来实现不同颜色，闪烁特效，添加表情等美化效果。尽管做得比较美观，但当用户执行了多个操作后，shell程序界面充满着字符，向上翻还能看到一次又一次的菜单被调出，影响了显示的整洁度。

解决办法：把每个函数体的第一行都加上clear操作，于是用户在进行下一个请求时，上次的结果在屏幕上就会被clear，菜单调用前也会clear。于是便做出了超越命令行到达了类似一个独立的app的效果。用户体验大大提升。

四.错误写成ps aux | awk 'NR==1||\$2==\$pid{print}'>output.txt，不知道加 -v。

解决办法：学习了awk命令传参数的方法后，成功改为正确的形式：

```
ps aux | awk -v n=$pid 'NR==1||$2==n{print}'>output.txt
```

五.检查output文件和修改.sh的文件需要退出正在执行的.sh文件，步骤较为繁琐。

解决办法：将查看output文件和.sh文件直接整合到.sh文件的功能选项中，于是用户可以在执行.sh文件的同时查看output.txt，甚至还可以优化.sh程序。

---

## 项目相关文件说明

OStest1.pdf-----实验报告

OS EXP1 - shell.txt-----实验要求原始文件

mtest1.sh-----shell程序源代码

超链接： [点击此处可在github中查看项目：](#) )