实验一
GNU/Linux shell 进阶

2025/03

实验背景与目的

计算机科学与技术专业和软件工程的同学在大二夏季学期进行了《系统开发工具基础》实践课程的学习，熟悉了GNU/Linux shell的基本命令。

本实验在此基础上，继续学习Linux操作系统shell中文件系统、管道、权限、设备、内存、网络相关的命令（程序），①思考这些程序背后调用的操作系统内核的系统调用、开发者考虑的问题，②并为同学们能够在后续实验中进行实验环境部署打下基础。

实验环境

1、软件环境：Linux操作系统

用户名: stu01-stu138　　服务器: 10.140.32.159

端口号: 47001-47138　　初始密码: islouc@25

服务器ssh链接示例：ssh stu01@10.140.32.159 -p 47001

2、要求：

① 请大家拿到后先修改登录口令，不能使用其他人的账号登录，也不得替他人登录练习，**严禁使用账号进行与学习无关或攻击他人及恶意消耗系统资源等行为，一旦发现，按违规处理，后果严重；**

② 大家认真进行练习，练习量将作为实验课达标的依据之一（不评优劣，达到基本练习量即可）

实验内容

1、文件系统相关命令：

ls、cd、pwd、mkdir、touch、cp、mv、rm、cat、 find、grep、more、less、head、tail、ln、stat、file

2、管道：|

3、权限相关命令：chmod、chown、chgrp，特殊权限：suid、sgid、sbit

4、内存相关命令：free、top、ps、pmap、vmstat、sar，

5、内存信息：/proc/pid/maps、/proc/pid/smaps

6、网络相关命令：ifconfig、netstat、curl

7、shell脚本

热身回顾（熟悉的可跳过）

文件系统、管道及权限

① 列出/etc目录下所有以 ".conf" 结尾的文件，并显示它们的详细信息。

② 用 man 查看程序 touch 的使用手册。

③ 在个人目录下新建名为 ouc 的目录和名为自己姓名的文件， 将以下内容写入该文件

```
#!/bin/sh
curl --head --silent http://it.ouc.edu.cn
```

④ 将文件移动到ouc目录中，移动过程中改名为foo，查看foo的工作目录。

⑤ 直接执行foo命令，查看出错信息。用chmod为所有用户增加执行权限，再次运行。

⑥ 将foo的输出结果重定向（>）到result文件。使用vim在result中将已有内容复制3遍。

⑦ 使用cat、more和less命令查看result文件，并观察三个命令的差别。

⑧ 使用管道命令，显示result的内容，并使用sort进行排序。

⑨ 强制删除ouc目录及其中的文件。

热身回顾（熟悉的可跳过）

网络

① 使用 ifconfig 命令查看系统中所有网络接口的信息，并找到当前活动的网络接口

shell脚本

① 编写shell脚本 ex1.sh，提示用户输入用户名，并判断此用户名是否存在。

② 编写一个 Shell 脚本，遍历指定目录下的所有文件，并将文件名以及文件大小输出到一个结果文件中

文件系统及管道

(一) 知识点

1、命令提示符，以 Bourne Again SHell 为例（bash）

打开终端，左侧看到 alice:~$，其中 alice 为当前登录的用户名，~表为当前工作

目录为 home，$表示非 root 用户

2、文件系统

一般为树形文件系统，支持硬链接（区别于Windows）和符号链接

3、管道

管道可以将一个命令的输出作为另一个命令的输入。使用管道可以将多个命令连接在一起，使它们按照顺序依次执行，并将前一个命令的输出作为后一个命令的输入。

4、新命令

（1）date：打印出当前日期时间

（2）echo：字符串输出，格式：echo string。有空格需加上单（或双）引号

（3）which：显示在当前环境下某个命令（或符号链接）的路径

文件系统及管道

(二) 实验过程和要求

① 执行以下命令并说明其功能：ls /home | grep "^test"

②使用管道命令将 "cat" 命令的输出传递给 "head" 命令，显示某个文件的前4行内容。

③ 使用管道命令将/lib目录下的文件按字母顺序对文件名进行排序

④ 思考：为什么两个命令之间不能使用变量或者内存直接传递数据？

权限

(一) 知识点

chmod：更改文件 9 个属性。两种设置方法，一种是数字，一种是符号。

九个基本权限：owner/group/others(拥有者/组/其他) 分别的 read/write/execute 权限。

1、数字 r:4 w:2 x:1

例：chmod 770 filename

2、符号类型改变文件权限

u，g，o 来代表三种身份，a 代表 all。

+， -， = 表示增、删、设定

例：chmod a-x 文件名

chown：自学

chgrp：自学

权限

(二) 实验过程和要求

① 使用man学习chown和chgrp的功能及用法。

② 在桌面下创建一个目录data，使用 chmod 命令将目录data设置为 STICKY BIT 权限，从而使得在该目录下创建的文件只能被其所有者和超级用户删除。说明：该权限位什么含义？

课后探索：

要求课后自行用虚拟机进行练习

① 使用 chown 命令将文件 "file.txt" 的所有者更改为 root。

② 使用 chgrp 命令递归地将目录及其所有子目录下的文件和目录的用户组设置为目标用户组（提示： chgrp -R new_group directory_name）

内存

（一）知识点

1.ps：显示系统中正在运行的进程信息，包括进程的PID、内存使用情况等

例：ps aux

2.top：查看目前程序执行的情景和内存使用的情况

3.free：显示系统的内存使用情况，包括总内存、已使用内存、空闲内存以及缓冲区和缓存的使用情况

4.vmstat：显示系统的虚拟内存统计信息，包括内存的使用情况、页面交换情况等

5.pmap：显示指定进程的内存映射信息，包括进程的内存地址、权限、映射文件等。

例：pmap PID

内存

(二) 实验过程和要求

① 使用 free 命令查看系统的内存使用情况，并解释其中的列名称和含义

② 使用 top 命令查看系统的内存占用情况，并解释其中的内存使用率显示

③ 使用 ps 命令查看当前系统中内存占用最高的进程，并显示其进程ID和内存占用。

④ 使用 pmap 命令查看指定进程的内存映射情况，并解释输出中的地址范围和映射类型

⑤ 使用 vmstat 命令实时监控系统的虚拟内存使用情况，并解释输出中的各列含义。

课后探索：

课后自行用虚拟机进行练习

① 使用 sar 命令查看系统的内存利用率，并解释输出中的各列含义。

网络、binutils

（一）知识点

1.ifconfig：显示机器上所有网卡的详细网络信息，包括ip地址，掩码，网卡接口名称，网关等。

2.netstat：查看网络端口监听信息

3.curl：基于url语法，发送网络请求

4.**binutils**: 二进制工具集，常用的包括：as，ld，nm，strip，objcopy，objdump，readelf

网络、binutils

(二) 实验过程和要求

① 使用 netstat 命令查看当前系统的网络连接状态，并且只显示TCP连接。

② 自学常用的binutils工具。

Shell脚本

(一) 知识点

Shell　script是利用shell的功能所写的一个“程序”，这个程序是使用纯文本文件，将一些Linux　Shell的语法与命令(含外部命令)写在里面，搭配正则表达式、管道命令与数据流重定向等功能，以达到我们所想要的处理目的。

shell 编程注意事项：

1.shell 命名：Shell脚本名称命名一般为英文、大写、小写，后缀以.sh 结尾。

2.shell 编程 首行需要 #!/bin/bash 开头。

3.shell 脚本变量不能以数字、特殊符号开头，可以使用下划线_, 但不能用破折号 -

4.# 在 Bash 中表示注释，而 ! 即使有双引号（"）也有特殊含义

Shell脚本

5.编程格式

```
#!/bin/sh
echo -e "Hello world"
```

第一行 #!/bin/sh 声明该shell script使用的shell名称，linux默认使用的是bash shell

从echo开始，后面表示程序正文

6.运行.sh文件

方法一：当前文件执行.sh 文件

```
./test.sh       # 文件必须含有x执行权限 [文件赋x权限：chmod u+x hello.sh]
sh test.sh    # 文件可以没有x权限
```

方法二：绝对路径执行.sh 文件

./home/test/test.sh　　或　　sh /home/test.test.sh

Shell脚本

7.基础语法

（1）变量赋值：var01=abc

注：①中间不能有空格和特殊字符

②使用 var01 的时候用 $var01

③变量放入“”中仍可以被替换，‘’不行假设$foo 中的内容为 bar

echo "$foo" # 打印 bar

echo '$foo' # 打印 $foo

（2）函数

以第一个参数创建目录，并进入该目录：

$1：表示脚本的第一个参数（$1 到 $9）（$0 表示脚本名）

（3）&&和||：短路运算符，true 是 0，false 是 1

例 false || echo "Oops, fail" # Oops, fail

true || echo "Will not be printed" #

```
mcd()
{
        mkdir -p "$1"
        cd "$1"
}
```

Shell脚本

（4）例

```
#!/bin/bash

echo "Starting program at $(date)" # date会被替换成日期和时间

echo "Running program $0 with $# arguments with pid $$"

for file in "$@"; do
    grep foobar "$file" > /dev/null 2> /dev/null
    # 如果模式没有找到，则grep退出状态为 1
    # 将标准输出流和标准错误流重定向到Null，不关心这些信息
    if [[ $? -ne 0 ]]; then
        echo "File $file does not have any foobar"
        echo "# foobar" >> "$file"
    fi
done
```

$#:参数个数
$$:进程号
$@:所有参数
$?:上个命令的返回值（0表示成功）
-ne 不等于
-eq 等于

Shell脚本

（5）通配与展开

①通配

？ 一个字符

* 0 到多个字符

②展开 { }

convert image.{png,jpg} = convert image.png image.jpg

mv *{.py,.sh} folder =移动所有 *.py 和 *.sh 文件

diff <(ls foo) <(ls bar) 比较文件夹 foo 和 bar 中包含文件的不同

（6）其他脚本

shebang=sharp+bang（#!）表示该脚本的解释器，可包含选项

```
#!/usr/local/bin/python
import sys
for arg in reversed(sys.argv[1:]):
    print(arg)
```

Shell脚本

(二) 实验过程和要求

① 思考：shell 为何知道用 sh 来解析这个文件?

② 编写一个 Shell 脚本，统计一个文件中某个关键词出现的次数。

分析程序（更详细程序查看下载）

```
#!/bin/bash
echo "100" > A.txt
echo "100" > B.txt


transfer() {
    b=$(tail -n 1 $2.txt)
    a=$(tail -n 1 $1.txt)
    ((a -= $3))
    ((b += $3))
    echo "$a" >> $1.txt
    echo "$b" >> $2.txt
    echo "$a + $b = $(($a+$b))"
}

for ((i=0; i<7; i++)); do
    transfer A B 1 $i &
    transfer B A 1 $i &
done
wait
```

程序或系统的正确性如果可以用公式来表达，就称之为不变量。例如

output1 mod input1 = 0

①从这个程序的逻辑来看，不变量是什么？

②从这个程序的执行来看，结果是否正确？

③为什么？