

四 Linux Shell 编程

实验目的

1. 理解 Linux Shell 脚本；
2. 熟悉 Linux Shell 脚本基本语法；
3. 编写系统监控脚本。

实验环境

安装有 Linux 操作系统并连接互联网的计算机。

实验步骤

Shell 是一个用 C 语言编写的程序，它是用户使用 Linux 的桥梁。Shell 是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核的服务。

Linux 中 Shell 编程通常指编写 Shell 脚本。Linux 中 Shell 类型多样，常见有 sh、csh、ksh、rsh、bash 等，不同 Shell 中脚本编写及运行略有差异。本实验中以最常见的 bash 为例。

Shell 脚本中需用到大量 Linux 命令以及正则表达式、管道符、数据流重定向等语法规则，还需要把内部功能模块化后通过逻辑语句进行处理，最终形成日常所见的 Shell 脚本。

1. 第一个 shell 脚本

(1) 编辑脚本文件

新建文件 helloworld.sh，扩展名为 sh（sh 代表 shell），扩展名不影响脚本执行，仅用于识别文件类型。

如下所示为程序代码。

代码清单 1 shell 脚本示例

```
#!/bin/bash

# Function: print "Hello World" to helloworld.txt
# Author: Mr. Yu

echo "Hello World !"
```

Shell 脚本包含 3 部分内容：首先是“#!”是一个约定的标记，表明系统脚本执行所需要的解释器，即使用哪一种 Shell（本例为 bash）；其次是以“#”开头为注释；最后是脚本功能代码。

（2）运行脚本

运行脚本有 2 种方法，一是解释器直接执行，二是作为可执行程序执行。

A. 解释器直接执行

如图 1 所示为执行结果。

```
root@ubuntu:/# cd /test
root@ubuntu:/test#
root@ubuntu:/test# bash helloworld.sh
hello world!
```

图 1 解释器直接执行 shell 脚本

B. 作为可执行程序

如图 2 所示为执行过程及结果。

```
root@ubuntu:/test# ls -l
total 4
-rw-r--r-- 1 root root 32 Mar 11 21:53 helloworld.sh
root@ubuntu:/test#
root@ubuntu:/test# chmod 744 helloworld.sh
root@ubuntu:/test#
root@ubuntu:/test# ./helloworld.sh
hello world!
```

图 2 作为可执行程序执行 shell 脚本

2. 通过脚本获取系统信息

编写脚本，获取系统时间、系统持续运行时间、系统负载、内存使用等信息。

注意：

- （1）脚本命令中嵌套的 Linux 命令以“`”标注，注意不是单引号；
- （2）CPU 信息获取方式，通过 /proc 目录获取系统信息；
- （3）灵活运行管道进行数据过滤。

代码清单 2 获取系统信息脚本

```
#!/bin/bash

#Function: display the infomation of current system
#Author:Mr.Yu

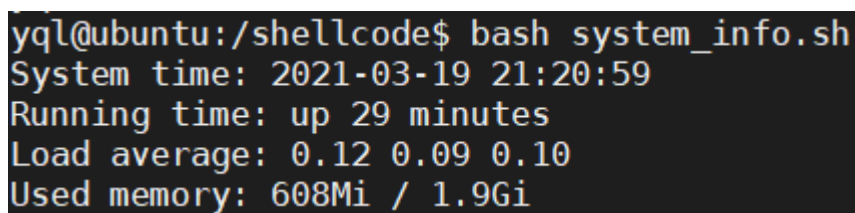
# 1.print the system time
echo System time: `date "+%Y-%m-%d %H:%M:%S"`

# 2.print how long the system has been running
echo Running time: `uptime -p`

# 3.print the load average in the system
echo Load average: `cat /proc/loadavg | awk '{print $1, $2, $3}'`

# 4.print used memory in the system
totalMem=`free -h | grep Mem | awk '{print $2}'`
usedMem=`free -h | grep Mem | awk '{print $3}'`
echo Used memory: $usedMem / $totalMem
```

如图 3 所示运行脚本与获取到系统基本信息结果。



```
yql@ubuntu:/shellcode$ bash system_info.sh
System time: 2021-03-19 21:20:59
Running time: up 29 minutes
Load average: 0.12 0.09 0.10
Used memory: 608Mi / 1.9Gi
```

图 3 获取系统信息

3. if 条件判断

if 条件判断由 if、then、else、fi 关键词组成。如图 4.4 所示为单分支、双分支、多分支语法结构。

代码清单 3 if 条件判断基本结构

if 条件测试	if 条件测试	if 条件测试 1
then 执行动作	then 执行动作 1	then 执行动作 1
fi	else 执行动作 2	elif 条件测试 2
	fi	then 执行动作 2
		fi

4. 获取网卡信息

编写脚本，获取网卡输入输出数据流量，并根据流量判断网卡状态。

注意：

- (1) 网卡名需与实际网卡名相符；
- (2) Linux 条件判断语法，“[条件表达式]”中条件表达式前后有空格；
- (3) Linux 中变量比较运算符。

代码清单 4 获取网卡信息代码

```
#!/bin/bash

#Function: monitor the flow of network
#Author:Mr.Yu

# display IP address
echo IP: `ifconfig ens33 | grep -w inet | awk '{print $2}'`

# get receive bytes 10 seconds ago
inputBytes1=`cat /proc/net/dev | grep ens33 | awk -F: '{print $2}' | awk '{print $1}'`

# get transmit bytes 10 seconds ago
outputBytes1=`cat /proc/net/dev | grep ens33 | awk -F: '{print $2}' | awk '{print $9}'`

echo Input bytes1: $inputBytes1    Output bytes1: $outputBytes1

sleep 10

# get receive bytes 10s later
inputBytes2=`cat /proc/net/dev | grep ens33 | awk -F: '{print $2}'|awk '{print $1}'`

# get transmit bytes 10s later
outputBytes2=`cat /proc/net/dev | grep ens33 | awk -F: '{print $2}'|awk '{print $9}'`

echo Input bytes2: $inputBytes2    Output bytes2: $outputBytes2

# evaluate the network
if [ $inputBytes1 -le $inputBytes2 ]
    then
        echo Network traffic is on the rise.
    else
        echo Network traffic is on the falling.
fi
```

如图 5 所示运行脚本与获取到系统基本信息结果。

```
yql@ubuntu:/shellcode$ bash network_monitor.sh
IP: 192.168.254.132
Input bytes1: 1122077 Output bytes1: 2968620
Input bytes2: 1125493 Output bytes2: 2978661
Network traffic is on the rise.
```

图 5 获取网卡状态

5. for 条件循环

for 循环一般格式为：

代码清单 5 for 循环一般结构

```
for 变量名 in 取值列表
do
    command1
    command2
    ...
done
```

如下代码所示为 for 循环示例。

代码清单 6 for 循环示例

```
for loop in 1 2 3 4 5
do
    echo "The value is: $loop"
done
```

6. 监控 CPU 负载

编写脚本监控 CPU 负载，并通过文本文件保存负载记录。

注意：

- (1) 文件及权限判断方式；
- (2) 重定向方法；
- (3) Linux 中变量比较运算符。

代码清单 7 监控 CPU 负载

```
#!/bin/bash
```

```

#Function: monitor load average of cpu, and write to file
#Author:Mr.Yu

# create file
if [ -f cpu_monitor.txt ]
    then
        touch cpu_monitor.txt
    fi

# modify file permission
if [ -w cpu_monitor.txt ]
    then
        chmod 755 cpu_monitor.txt
    fi

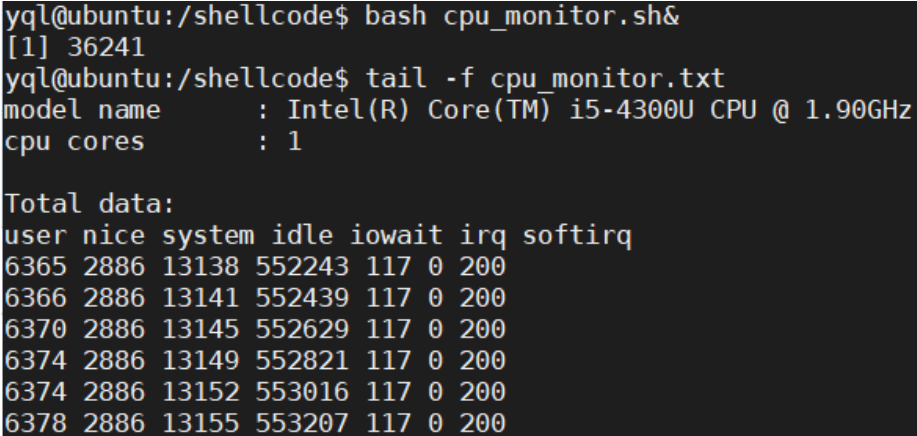
# write cpu infomation
cat /proc/cpuinfo | grep "model name" > cpu_monitor.txt
cat /proc/cpuinfo | grep "cpu cores" >> cpu_monitor.txt

echo " " >> cpu_monitor.txt
echo Total data: >> cpu_monitor.txt
echo user  nice  system  idle  iowait  irq  softirq  >> cpu_monitor.txt

#write cpu infomation every 2s
for ((i=0;i<=50;i++))
do
    cat /proc/stat | grep 'cpu ' | awk '{print $2 " "$3 " "$4 " "$5 " "$6 " "$7 " "$8}' >> cpu_monitor.txt
    sleep 2
done

```

如图 6 所示，后台运行脚本（命令后加 “&” 符号），并通过 “tail -f” 命令动态查看文件内容。



```

yql@ubuntu:/shellcode$ bash cpu_monitor.sh&
[1] 36241
yql@ubuntu:/shellcode$ tail -f cpu_monitor.txt
model name      : Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz
cpu cores       : 1

Total data:
user nice system idle iowait irq softirq
6365 2886 13138 552243 117 0 200
6366 2886 13141 552439 117 0 200
6370 2886 13145 552629 117 0 200
6374 2886 13149 552821 117 0 200
6374 2886 13152 553016 117 0 200
6378 2886 13155 553207 117 0 200

```

图 6 监控 CPU 运行

实验内容

1. 运行以上脚本；
2. 对脚本进行改进，监控系统运行状况。