

## linux学习2

### 1.shell脚本编程

shell脚本作用：软件启动 性能监控 日志分析

shell的本质：内置命令/外部命令

输入命令，先去内置命令里面取查找

再去PATH里面查找

shell脚本语言和C语言一样吗？

编译型语言 c

解释型语言 shell

第一个shell脚本

```
lixin hello!
```

编辑、保存、改权限、运行/排错

shell启动方式

当程序执行 ./hello.sh

指定解释器运行 /bin/dash hello.sh

source和. source hello.sh . hello.sh

### 2.shell脚本语法讲解

定义变量

variable=value 该定义方式变量不能有空格

variable='value' 该定义方式变量允许有空格 不可以输出引用变量，原封不动打印字符

variable="value" 该定义方式变量允许有空格 可以输出引用变量

使用变量

\$variable 自动识别后面变量，可能识别错误

\${variable} 手动识别，做到边界确认，保证识别正确

将命令结果赋值给变量

```
variable= command
```

```
variable=$(command)
```

删除变量

```
unset
```

特殊变量

\$0: 当前脚本的文件名

\$n: 传递给脚本的参数

\$#: 传递给脚本或函数的参数个数

\$\*: 传递给脚本或函数的所有参数

\$?: 上个命令的退出状态或获取函数返回值

\$\$ : 当前shell进程的ID

字符串

并排放

退出当前进程

```
exit
```

对整数进行数学运算

(())

```
#!/bin/bash
var="1234 555"
read -p "input a": a
read -p "input b": b
cc=$((a+b))

mm=`pwd`
nn=$(pwd)
var1="${var}aa"
echo "$var"
echo "$var1"
echo "${mm}"
echo "${nn}"
echo "$a"
echo "${b}yy"

echo "$cc"
exit 111
```

逻辑与/或

command1 && command2 :先判断command1是否成立，成立执行command2

command1 || command2: 先判断command1是否不成立, 不成立执行command2

检测某个条件是否成立

test expression 和 [expression]

-eq : 判断是否相等

-ne: 判断是否不相等

-gt: 判断数值是否大于

-lt: 判断数值是否小于

-ge: 判断是否大于等于

-le: 判断是否小于等于

-z str: 判断字符串是否为空

-n str: 判断字符串str是否非空

=和==: 判断字符串str是否相等

-d filename : 判断文件是否存在, 并且是否未目录文件

command1 | command2 : command1输出信息作为command2输入信息, command1不能有错误

```
#!/bin/bash
read -p "input a:" a
read -p "input b:" b
read -p "input c:" c
read -p "input mulu:" mulu
test $a -eq $b && echo "a=b"
test $a -eq $b || echo "a!=b"
[ $a -eq $b ] && echo "a=b"
[ -z $c ] || echo "$c"
[ -d $mulu ] && echo "$mulu"
```

if语句

if condition

then

statement1

fi

if else语句

```
if condition

then

    statement1

else

    statement2

fi
```

if elif else语句

```
if condition

then

    statement1

elif condition2

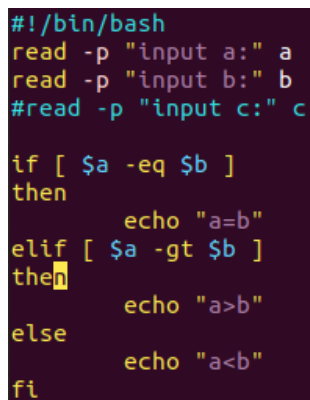
then

    statement2

else

    statement3

fi
```



```
#!/bin/bash
read -p "input a:" a
read -p "input b:" b
#read -p "input c:" c

if [ $a -eq $b ]
then
    echo "a=b"
elif [ $a -gt $b ]
then
    echo "a>b"
else
    echo "a<b"
fi
```

case in 语句（与C语言中switch case一样）

```
case expression in
```

```

pattern1)

    statement1

;;

pattern2)

    statement2

;;

pattern3)

    statement3

;;

*)

    statementn

esac

```

```

#!/bin/bash
read -p "input a:" a
#read -p "input b:" b
#read -p "input c:" c

case $a in
    1)
        echo "a=1"
        ;;
    2)
        echo "a=2"
        ;;
    *)
        echo "a!=1&&a!=2"
        ;;
esac

```

for in 循环

```

for variable in value_list

do

    statements

done

```

其中 value\_list

直接给出具体的值 1 2 3 4 5

直接给出一个取值范围 {1..5}

使用命令的执行结果 \$(ls /bin)

使用shell通配符

使用特殊变量 \$@

```
#!/bin/bash
#for n in $*
#for n in 1 2 3 4 5
#for n in {1..5}
for n in $(ls /bin)
do
    echo $n
done
```

while 循环

while condition

do

statements

done

```
n=0
while (( n < 10 ))
do
    echo "$n"
    n=$(( n+1 ))
done
~
~
```

函数

function name()

{

statements

{return value}

}

```
function name(){  
    echo "lixin"  
}  
name
```