

Shell 笔记

解析器

1. sh HelloWorld.sh
2. bash HelloWorld.sh

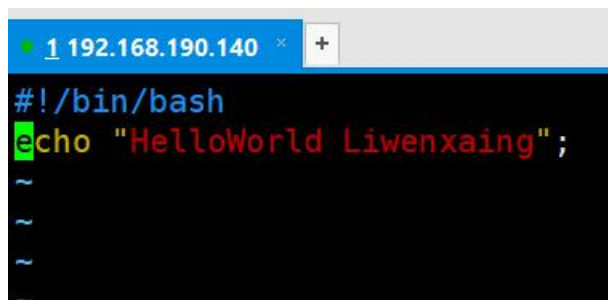
解释：Shell 拥有以上两种解析器

HelloWorld 案例

文件开头内容：

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

```
Echo "HelloWorld";
```

A terminal window with a blue title bar showing the IP address 192.168.190.140. The terminal content shows the execution of a shell script. The first line is the shebang #!/bin/bash. The second line is the command echo "HelloWorld Liwenxaing";. The output of the command is shown on the next line as HelloWorld Liwenxaing. The prompt ~ is visible at the bottom.

```
1 192.168.190.140 x +
#!/bin/bash
echo "HelloWorld Liwenxaing";
~
~
~
```

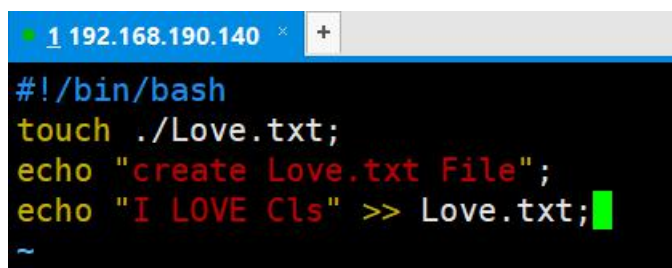
执行权限

```
chmod 777 HelloWorld.sh
```

简单的多命令

文件内容追加：

```
Echo "I LOVE Cls" >> Love.txt
```

A terminal window with a blue title bar showing the IP address 192.168.190.140. The terminal content shows the execution of a shell script. The first line is the shebang #!/bin/bash. The second line is the command touch ./Love.txt;. The third line is the command echo "create Love.txt File";. The fourth line is the command echo "I LOVE Cls" >> Love.txt;. The output of the third command is shown on the next line as create Love.txt File. The prompt ~ is visible at the bottom.

```
1 192.168.190.140 x +
#!/bin/bash
touch ./Love.txt;
echo "create Love.txt File";
echo "I LOVE Cls" >> Love.txt;
~
```

变量

系统变量

\$HOME

\$PWD

\$SHELL

\$USER

使用

Echo \$PWD

自定义变量

A=1

Echo \$A

声明变量的时候不能有空格

删除一个变量

Unset \$A

创建一个只读变量

Readonly A=1

导出变量 提升为全部的变量 供其他 Shell 脚本使用

Export \$A

特殊变量

\$n 一般从 0-9 代表执行该脚本后面跟的参数

\$0 指代的就是当前的脚本文件名称

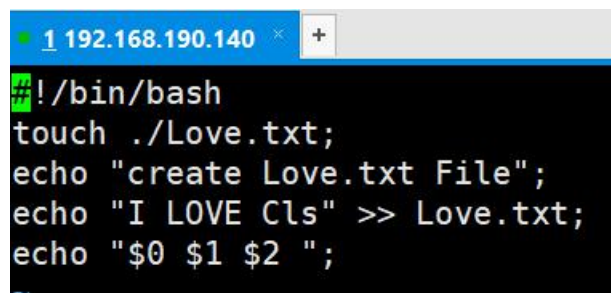
\$? 获取上一次执行的结果是否正确执行返回 0 1

\$* 获取到用户输入的所有值参数

\$@ 和\$*一样不同的是在加上双引号的时候*会将其作为一个整体

使用

Echo "\$0 \$1 \$2";



```
1 192.168.190.140 x +
#!/bin/bash
touch ./Love.txt;
echo "create Love.txt File";
echo "I LOVE Cls" >> Love.txt;
echo "$0 $1 $2 ";
```

注意：只要是在配置过环境变量都可以直接获取到

运算符

`+ - * / %`

之间必须有空格

`Expr 1 + 2`

`Expr 1 * 2`

`Expr `expr 1 + 2` * 4`

或者

`A=${1+2}`

`Echo $A`

或者

`$((1+2))`

条件判断

`[condition]` 操作

`[22 -lt 23] -> $? -> 1`

`[-r Hello.txt] -> $? -> 1`

`-lt` 小于

`=` 等于

`-gt` 大于

`-ge` 大于等于

`-le` 小于等于

`-ne` 不等于

`-r` 是否是可读文件

`-e` 是否存在

`-w` 是否可写

`-f` 是否是文件

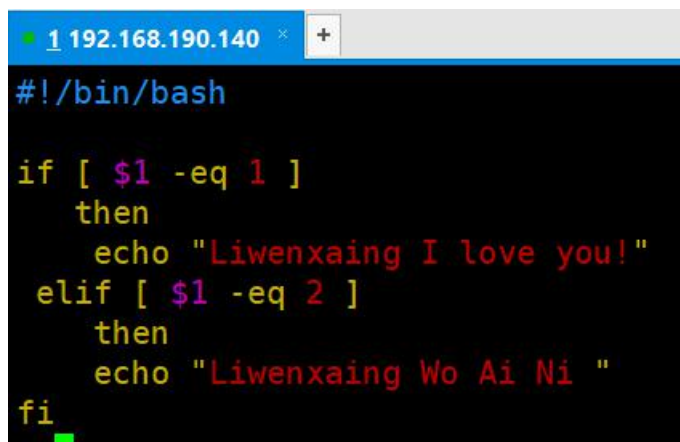
`-d` 是否是目录

流程控制

```
If [ condition ]  
then  
    Echo "xxx";  
Fi
```

```
If [ condition ]  
then  
    Echo "xxx";  
Elif [ condition ]  
then  
    Echo "xxx"  
Fi
```

```
If [ condition ]  
then  
    Echo "xxx";  
fi
```

A terminal window with a title bar showing the IP address 192.168.190.140. The terminal content shows a bash script using if, elif, and fi to check command-line arguments. The prompt is #!/bin/bash. The script has two branches: one for argument 1 printing 'Liwenxaing I love you!' and another for argument 2 printing 'Liwenxaing Wo Ai Ni '.

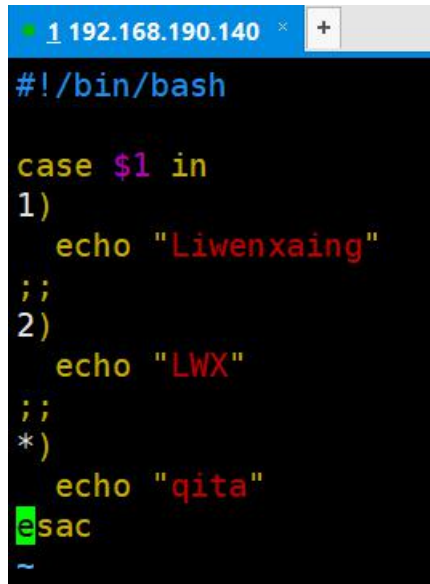
```
1 192.168.190.140 × +  
#!/bin/bash  
  
if [ $1 -eq 1 ]  
then  
    echo "Liwenxaing I love you!"  
elif [ $1 -eq 2 ]  
then  
    echo "Liwenxaing Wo Ai Ni "  
fi
```

Case 语句

语法:

```
Case $1 in  
1)  
    Echo "liwenxaing"  
;;  
2)  
    Echo "LWX"
```

```
;;
*)
    Echo "qita"
Esac
```

A terminal window with a blue title bar showing the IP address 192.168.190.140. The terminal content shows a bash script using a case statement to handle different inputs. The prompt is #!/bin/bash. The script has three cases: 1) echoes 'Liwenxaing', 2) echoes 'LWX', and *) echoes 'qita'. The script ends with esac and a tilde prompt. The text is color-coded: case, \$1, in, 1), 2), *, and esac are in yellow; echo is in green; and the strings are in red.

```
#!/bin/bash

case $1 in
1)
    echo "Liwenxaing"
;;
2)
    echo "LWX"
;;
*)
    echo "qita"
esac

~
```

For 循环

```
For(( i=1;i<=100;i++ ))
Do
    程序
Done
```

```
For variable in var1,var2,var3
Do
    程序
Done
```

```
S=0
I=1
While [ $i -le 100 ]
Do
    程序
Done
```

```

1 192.168.190.140 × +
#!/bin/bash

s=0
for (( i=1; i<=100; i++))
do
    s=$((s+i))
done

echo "总和 : $s"

for variable in $*
do
    echo "Good My Value Is $variable"
done

sum=0
j=0
while [ $j -le 100 ]
do
    sum=$((sum+j))
done
echo "总和: $sum"
~

```

Read 获取控制台输入

-p 提示内容

-t 等待时间

NAME 就是一个变量 获取到用户输入的值 赋值给 NAME 这个变量

```

1 192.168.190.140 × +
#!/bin/bash

read -p 'input your name :' -t 10 NAME

echo $NAME
~

```

函数

系统函数：

Basename 获取到一个路径中的文件名

```
Basename /home/atlwx/test.txt // test.txt
```

Dirname 获取到一个路径中的目录名

```
Dirname /home/atlwx/test.txt // /home/atlwx
```

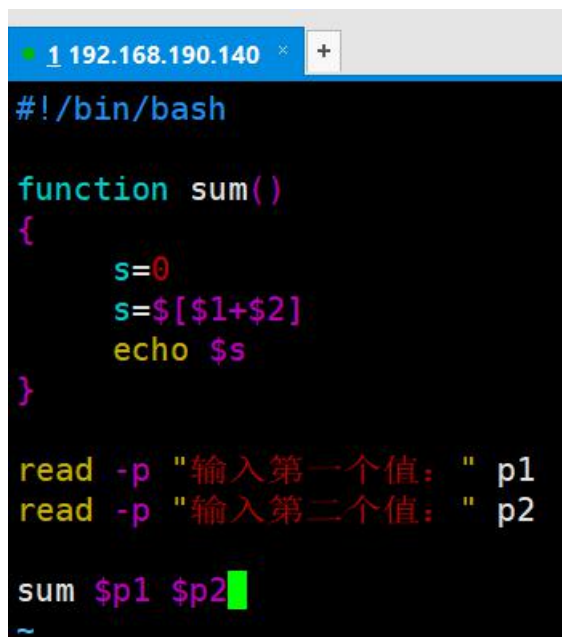
自定义函数

Function sum()

```
{  
    程序  
    [return]  
}
```

// 调用

Sum

A terminal window with a title bar showing '1 192.168.190.140'. The terminal content shows a bash prompt '#!/bin/bash' followed by a function definition: 'function sum() { s=0; s=\$((s+\$1+\$2)); echo \$s; }'. Below the function, there are two 'read' commands: 'read -p "输入第一个值: " p1' and 'read -p "输入第二个值: " p2'. Finally, the function is called with 'sum \$p1 \$p2'. The cursor is at the end of the last line.

```
#!/bin/bash  
  
function sum()  
{  
    s=0  
    s=$((s+$1+$2))  
    echo $s  
}  
  
read -p "输入第一个值: " p1  
read -p "输入第二个值: " p2  
  
sum $p1 $p2
```

过滤文件内容查看

```
Cat a.txt | grep myName
```

CUT 每一行剪切内容

语法:

语法灵活

```
Cut -d " " -f 1
```

Cut -d " " -f 2-

Cut -d : -f 1

Echo \$PATH | Cut -d : -f 1

Echo 123456 | Cut -d 4 -f 1

Echo \$PATH | Cut -d " " -f 1 | Cut -d " " -f 1

```
[root@eureka1 shell]# touch cut.txt
[root@eureka1 shell]# vim cut.txt
[root@eureka1 shell]# cat cut.txt
my name is liwenxaing
my name is canata
my from China
my love my China
your is dog?
oh no my is bigMan
superman is your?
your is sb?
I can't speak English
Can you speak Chinase?
[root@eureka1 shell]# cat cut.txt | grep speck
[root@eureka1 shell]# cat cut.txt | grep speak
I can't speak English
Can you speak Chinase?
[root@eureka1 shell]# cat cut.txt | grep speak | cut -d " " -f 1
I
Can
[root@eureka1 shell]#
```

```
[root@eureka1 shell]# echo $PATH | cut -d : -f 1
/usr/local/jdk8/bin
[root@eureka1 shell]#
```

```
Can you speak Chinase?
[root@eureka1 shell]# cut -d " " -f 2- cut.txt
name is liwenxaing
name is canata
from China
love my China
is dog?
no my is bigMan
is your?
is sb?
can't speak English
you speak Chinase?
[root@eureka1 shell]#
```


Sed 流编辑器

`Sed -e "2d" text.txt` // 代表删除第二行 但是源文件不会受影响

`Sed -e "2a nihao" text.txt` // 代表在第二行之后追加一行 但是源文件不会受影响

`Sed -e "s/wo/ni/g" text.txt` // 全局替换 不加 `g` 就不是全局替换了 但是源文件不会受影响

```
[root@eureka1 shell]# sed -e "s/speak/woaini/g" cut.txt
my name is liwenxaing
my name is canata
my from China
my love my China
your is dog?
oh no my is bigMan
superman is your?
your is sb?
I can't woaini English
Can you woaini Chinase?
[root@eureka1 shell]#
```

```
[root@eureka1 shell]# sed -e "2d" cut.txt
my name is liwenxaing
my from China
my love my China
your is dog?
oh no my is bigMan
superman is your?
your is sb?
I can't speak English
Can you speak Chinase?
[root@eureka1 shell]#
```

```
[root@eureka1 shell]# sed -e "2a nihaoahhh" cut.txt
my name is liwenxaing
my name is canata
nihaoahhh
my from China
my love my China
your is dog?
oh no my is bigMan
superman is your?
your is sb?
I can't speak English
Can you speak Chinase?
[root@eureka1 shell]#
```

