1. 执行多条命令

   使用分号隔开：

```
date; who
```

2. Shell脚本

```
#!/bin/bash
date
who
```

   添加执行权限

```
chmod u+x test.sh
# 通过相对路径或绝对路径的方式引用
./test.sh
```

3. 输出消息

```
echo Hello World
echo "Hello World"
echo 'Hello World'
```

> `""` 中的特殊符号具有意义，如 `$` `!`
>
> `''` 中的内容会被原样输出，不会被解析

   - `echo -n`：不换行输出

4. 变量

   1. 环境变量

```
# 显示环境变量
set
```

      加上 `$` 引用环境变量

```
echo $HOME
```

2. 用户变量

```
var1=hello
var2=11
```

> 等号两侧不能有空格！

以示区分，用户自定义变量使用小写，与环境变量（全大写）进行区分，以免混淆。

```
#!/bin/bash
name=Alice
echo "Hello $name" # Hello Alice
```

3. 命令替换

将命令输出赋值给变量

1. ``
2. $()

```
test=`date`
# or
test = $(date) # 较常用
```

5. 重定向

1. 输出重定向：`command > outputfile`

```
echo "Hello World" > hello
cat hello
```

`>` 会覆盖原文件内容，`>>` 向文件追加内容。

```
echo "Hello World" >> hello
cat hello
```

2. 输入重定向

`command < inputfile`

```
wc < hello
```

> wc：统计文本
>
> 1. 文本行数
> 2. 文本词数
> 3. 文本字节数

内联输入重定向（`<<`），不指定文件，在命令行直接指定输入内容

```
xt in ~/shell λ wc << EOF
heredoc> Hello World
heredoc> Hi
heredoc> EOF
 2  3 15
```

6. 管道

   将一个命令的输出作为一个命令的输入，`command1 | command2`

   > 不是依次执行，而是会同时运行两个命令，在第一个命令产生输出的同时，输出会被立即送给第二个命令。

   ```
   rpm -qa | sort | more
   ```

7. 数学运算

   1. expr

      ```
      expr 1 + 5
      expr 1 \* 5
      ```

表11-1 expr命令操作符

操 作 符	描 述
ARG1 | ARG2	如果ARG1既不是null也不是零值，返回ARG1；否则返回ARG2
ARG1 & ARG2	如果没有参数是null或零值，返回ARG1；否则返回0
ARG1 < ARG2	如果ARG1小于ARG2，返回1；否则返回0
ARG1 <= ARG2	如果ARG1小于或等于ARG2，返回1；否则返回0
ARG1 = ARG2	如果ARG1等于ARG2，返回1；否则返回0
ARG1 != ARG2	如果ARG1不等于ARG2，返回1；否则返回0
ARG1 >= ARG2	如果ARG1大于或等于ARG2，返回1；否则返回0
ARG1 > ARG2	如果ARG1大于ARG2，返回1；否则返回0
ARG1 + ARG2	返回ARG1和ARG2的算术运算和
ARG1 - ARG2	返回ARG1和ARG2的算术运算差
ARG1 * ARG2	返回ARG1和ARG2的算术乘积
ARG1 / ARG2	返回ARG1被ARG2除的算术商
ARG1 % ARG2	返回ARG1被ARG2除的算术余数
STRING : REGEXP	如果REGEXP匹配到了STRING中的某个模式，返回该模式匹配
match STRING REGEXP	如果REGEXP匹配到了STRING中的某个模式，返回该模式匹配
substr STRING POS LENGTH	返回起始位置为POS（从1开始计数）、长度为LENGTH个字符的子字符串
index STRING CHARS	返回在STRING中找到CHARS字符串的位置；否则，返回0
length STRING	返回字符串STRING的数值长度
+ TOKEN	将TOKEN解释成字符串，即使是个关键字
(EXPRESSION)	返回EXPRESSION的值

2. 使用方括号

```
var1=$[1 + 5]
var2=$[1 * 5]
```

> Bash Shell 只支持整数运算，`10 / 3 → 3`

3. 浮点运算：`bc`

```
xt in ~/shell λ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
12 * 1.3
15.6
10 / 3
3
scale=2
10/3
3.33
```

```
var=$(echo "scale=2; 10 / 3" | bc)
var=$(bc << EOF
scale = 2
a = 1.2
b = 2.3
a + b
EOF
)
```

8. 退出脚本

1. 查看状态码

```
echo $?
```

成功结束的命令的退出状态码是0，非 0 表示错误

表11-2　Linux退出状态码

状　态　码	描　　述
0	命令成功结束
1	一般性未知错误
2	不适合的shell命令
126	命令不可执行
127	没找到命令
128	无效的退出参数
128+x	与Linux信号x相关的严重错误
130	通过Ctrl+C终止的命令
255	正常范围之外的退出状态码

2. `exit`

```
exit 5 # 0 ~ 255
exit 300 # -> 44
```