A+

# echo命令详解（一）真的很详细


echo命令是linux中最基础的命令，也是很常用的命令，特别是在写shell脚本的时候，可能会经常被用到，虽然echo命令非常基础，但是功能还算丰富，此处对ech
方法进行总结，并给出示例，方便记忆与回顾。

echo命令的基本用法，很简单，就是echo命令后面跟上要输出的文本，如下。

```
[root@zsythink ~]# echo 123
123
[root@zsythink ~]# echo test message
test message
[root@zsythink ~]# echo "www.zsythink.net"
www.zsythink.net
[root@zsythink ~]#
```



除了基本用法，还可以配合一些选项使用

echo -n 表示不换行输出

```
[root@zsythink ~]# echo "www.zsythink.net"
www.zsythink.net
[root@zsythink ~]# echo -n "www.zsythink.net"
www.zsythink.net[root@zsythink ~]#
```

仅将文本发送到当前选项卡



可以看到，如果不添加-n选项，文本输出以后，指定换行了，而添加了-n选项以后，文本直接连着命令提示符输出了，并没有换行，这样演示效果不明显，我们可以
信息，效果比较明显，示例如下。

```
[root@zsythink ~]# echo "www.zsythink.net" ; echo "zsy.zsythink.net"
www.zsythink.net
zsy.zsythink.net
[root@zsythink ~]# echo -n "www.zsythink.net" ; echo "zsy.zsythink.net"
www.zsythink.netzsy.zsythink.net
[root@zsythink ~]#
[root@zsythink ~]#
```



当使用echo输出命令替换后的内容时，命令执行结果的格式可能会发生变化，比如，如果我们想要输出ifconfig命令执行后的结果，我们可能会使用如下命令

```
[root@zsythink ~]# echo `ifconfig`
eth1 Link encap:Ethernet HWaddr 00:0C:29:60:4C:74 inet addr:172.18.18.128 Bcast:172.18.18.255 Mask:255.255
.255.0 inet6 addr: fe80::20c:29ff:fe60:4c74/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:9445 errors:0 dropped:0 overruns:0 frame:0 TX packets:812 errors:0 dropped:0 overruns:0 carrie
r:0 collisions:0 txqueuelen:1000 RX bytes:2059761 (1.9 MiB) TX bytes:82277 (80.3 KiB) lo Link encap:Local
Loopback inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1/128 Scope:Host UP LOOPBACK RUNNING MTU:65536 M
etric:1 RX packets:12 errors:0 dropped:0 overruns:0 frame:0 TX packets:12 errors:0 dropped:0 overruns:0 ca
rrier:0 collisions:0 txqueuelen:0 RX bytes:720 (720.0 b) TX bytes:720 (720.0 b)
[root@zsythink ~]#
```



但是我们发现，这样输出的文本的格式发生了变化，这样可能不是我们想要的，因为当我们直接执行ifconfig命令时，ifconfig返回的结果是多行的，可是当我们使用
时候，格式发生了变化，如果想要按照命令执行后的原格式输出命令替换后的结果，可以使用如下方法，如下方法在写脚本的时候可能会用到。

```
[root@zsythink ~]# echo "`ifconfig`"
eth1      Link encap:Ethernet  HWaddr 00:0C:29:60:4C:74
          inet addr:172.18.18.128  Bcast:172.18.18.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe60:4c74/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9753 errors:0 dropped:0 overruns:0 frame:0
          TX packets:840 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2123414 (2.0 MiB)  TX bytes:85939 (83.9 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:720 (720.0 b)  TX bytes:720 (720.0 b)
[root@zsythink ~]#
```



我们可以使用-e选项输出转义字符，比如常用的转义字符"\t", 转义字符"\t"表示制表符，作用相当于我们键盘上的tab键。

我们可以使用echo -e输出转义字符，将转义后的内容输出到屏幕上，示例如下

```
[root@zsythink ~]# echo -e  "www.zsythink.net\tzsythink"
www.zsythink.net        zsythink
[root@zsythink ~]#
[root@zsythink ~]#
```



上图示例中，已经将"\t"转义过的制表符输出到了屏幕上，所以，我们只要能够记住这些转义字符，就可以在echo命令中输出它们，那么，我们将常用的转义字符总给出示例。

常用的转义字符如下：

\b 转义后相当于按退格键（backspace），但前提是"\b"后面存在字符，具体效果参考下方示例。

\c 不换行输出，在"\c"后面不存在字符的情况下，作用相当于echo -n，具体效果参考下方示例。

\n 换行，效果看示例。

\f 换行，但是换行后的新行的开头位置连接着上一行的行尾，具体效果查看示例；

\v 与\f相同；

\t 转以后表示插入tab，即制表符，已经在上面举过例子；

\r 光标移至行首，但不换行，相当于使用"\r"以后的字符覆盖"\r"之前同等长度的字符，只看这段文字描述的话可能不容易理解，具体效果查看示例；

\\ 表示插入"\"本身；

使用echo命令输出上述转义字符的示例如下：

使用echo命令输出"\b"转义字符，在"\b"后面存在字符的前提下，"\b"表示删除前一个字符，"\b\b"表示删除前两个字符。

```
[root@cos68ini ~]# echo -e "123\b"
123
[root@cos68ini ~]# echo -e "123\b4567"
124567
[root@cos68ini ~]# echo -e "123\b\b4567"
14567
[root@cos68ini ~]# echo -e "123\b\b\b4567"
4567
[root@cos68ini ~]#
[root@cos68ini ~]#
```



可以看到，上例中，在"\b"后面不存在任何字符时，"\b"并没有转义为"退格键"，当"\b"后面存在字符时，一个"\b"就相当于按一次backspace键。

我们也可以使用\c转义符，表示不换行输出，但是当"\c"后面仍然存在字符时，"\c"后面的字符将不会被输出，如果"\c"后面不存在任何字符时，效果与使用"echo下。

```
[root@cos68ini ~]# echo -e "123\c"
123[root@cos68ini ~]#
[root@cos68ini ~]#
[root@cos68ini ~]# echo -n "123"
123[root@cos68ini ~]#
[root@cos68ini ~]#
[root@cos68ini ~]# echo -e "123\c456"
123[root@cos68ini ~]#
[root@cos68ini ~]#
[root@cos68ini ~]#
[root@cos68ini ~]#
```

"\n"转义后表示换行，下例中，被输出的字符从"\n"处开始另起一行。

```
[root@cos68ini ~]# echo -e "abcdefg\n1234"
abcdefg
1234
[root@cos68ini ~]#
[root@cos68ini ~]#
```



"\f"转移符表示
换行，但是换行后的新行的开头位置连接着上一行的行尾，如下图示例所示，下图中的第三个例子中有两个"\f"。

```
[root@cos68ini ~]# echo -e "123\f456"
123
   456
[root@cos68ini ~]# echo -e "abcdefg\f1234"
abcdefg
      1234
[root@cos68ini ~]# echo -e "abcdefg\f1234\fccccccccc"
abcdefg
      1234
          ccccccccc
[root@cos68ini ~]#
[root@cos68ini ~]#
```



"\v"转义符与"\f"转义符的作用相同。

"\r"转义符表示使用"\r"后面的字符覆盖"\r"之前的同等长度的字符，比较不容易理解，但是看下图示例，就很容易明白了。

```
[root@cos68ini ~]# echo -e "abcdefg\r123"
123defg
[root@cos68ini ~]# echo -e "abc\r123"
123
[root@cos68ini ~]# echo -e "a\r123"
123
[root@cos68ini ~]# echo -e "a\r"
a
[root@cos68ini ~]#
[root@cos68ini ~]#
```



上图中的第一个示例中，"\r"后面的123覆盖了abc，defg没有被覆盖，第二个示例中，因为abc一共有3个字符，123也是有3个字符，所以123覆盖了abc以后，只第三个实例中，123一共有3个字符，a只有一个字符，覆盖以后只剩下123，第四个示例中，"\r"后面并不存在任何字符，所以"\r"前面的字符没有被覆盖。

"\\"经过转义以后，表示"\",示例如下。

```
[root@cos68ini ~]# echo -e "abc\\def"
abc\def
[root@cos68ini ~]#
[root@cos68ini ~]#
```



通过echo命令，还能够输出彩色的文本，或者带有彩色背景的文本，因为篇幅原因， 我们另起一遍文章进行总结。文章链接地址如下：
http://www.zsythink.net/archives/111



常用命令