Linux使用cat命令显示隐藏字符的方法

时间：2015-03-09 来源：系统之家 作者：qipeng

　　Linux系统中，使用普通的文本编辑器查看的时候，有些字符是不显示的，这就需要使用到cat命令来查看，下面小编就给大家介绍下Linux使用cat命令查看隐藏字符的方法，一起来了解下吧。

Linux使用cat命令显示隐藏字符的方法

　　首先，我们创建一个简单的文本文件，写入一些特殊字符。打开终端，运行命令：

　　printf ‘testing\012\011\011testing\014\010\012more testing\012\011\000\013\000even more testing\012\011\011\011\012’ 》 /tmp/testing.txt

　　现在用不同的编辑器软件打开，显示的结果会不同。用简单的cat打开将显示：

　　$ cat /tmp/testing.txt

　　testing

　　testing

　　more testing

　　even more testing

　　如果用nano或者vim打开，将会看到：

　　testing

　　testing^L^H

　　more testing

　　^@^K^@even more testing

　　现在我们给cat加上一些选项参数，以便能显示出特殊字符来。

　　用cat -T命令来显示TAB键的字符^I

　　cat -T /tmp/testing.txt

　　testing

　　^I^Itesting

　　more testing

　　^I

　　even more testing

　　^I^I^I

　　用cat -E命令来显示行尾的结束字符$

　　$ cat -E /tmp/testing.txt

　　testing$

　　testing

　　$

　　more testing$

　　even more testing$

　　$

　　用简单的cat -A命令就可以显示所有不可见的字符：

　　$ cat -A /tmp/testing.txt

　　testing$

　　^I^Itesting^L^H$

　　more testing$

　　^I^@^K^@even more testing$

　　^I^I^I$

　　上面就是Linux使用cat命令显示隐藏字符的方法介绍了，cat命令除了能够显示不可见字符外，还有其他的功能，具体的可以参考系统之家Linux下如何使用cat命令？

分享到：

grep无法查找shell传过来的变量？先注意一下文本格式吧！

昨天，同事告诉我发现一个诡异的问题，grep无法搜索shell中的变量，着实很惊讶。到他所说的服务器上试了下，还真是不行！

大概就是这样一个要求：

①、有个文本为userid.txt，里面每一行一个用户id，类似如下：

Shell

0001

0003

0005

0007

0009

1

2

3

4

5

0001

0003

0005

0007

0009

②、另外还有一个文本为record.txt，里面是所有用户的操作记录，一行一条，并且包含有id，类似如下：

Shell

[12 11 2014 11:03,198 INFO] userId:0001 gilettype:3

[12 11 2014 12:12,198 INFO] userId:0002 gilettype:3

[12 11 2014 13:02,198 INFO] userId:0003 gilettype:1

[12 11 2014 14:33,198 INFO] userId:0001 gilettype:3

[12 11 2014 15:13,198 INFO] userId:0002 gilettype:2

[12 11 2014 16:43,198 INFO] userId:0003 gilettype:1

[12 11 2014 17:32,198 INFO] userId:0001 gilettype:3

[12 11 2014 18:16,198 INFO] userId:0002 gilettype:1

[12 11 2014 19:25,198 INFO] userId:0003 gilettype:2

1

2

3

4

5

6

7

8

9

[12 11 2014 11:03,198 INFO] userId:0001 gilettype:3

[12 11 2014 12:12,198 INFO] userId:0002 gilettype:3

[12 11 2014 13:02,198 INFO] userId:0003 gilettype:1

[12 11 2014 14:33,198 INFO] userId:0001 gilettype:3

[12 11 2014 15:13,198 INFO] userId:0002 gilettype:2

[12 11 2014 16:43,198 INFO] userId:0003 gilettype:1

[12 11 2014 17:32,198 INFO] userId:0001 gilettype:3

[12 11 2014 18:16,198 INFO] userId:0002 gilettype:1

[12 11 2014 19:25,198 INFO] userId:0003 gilettype:2

③、现在他要求循环取出userid.txt中每一行ID值，然后去record.txt去查找并保存结果。

实现这个需求原本很简单，根本难不倒他，只要使用while read + grep 就能搞定。可问题是明明record.txt里面包含这些id，却无法输出结果？？

我顺便写了一个测试脚本测试了下：

Shell

#!/bin/bash

while read userId;

do

echo $userId

grep $userId record.txt

done <userid.txt

1

2

3

4

5

6

#!/bin/bash

while read userId;

do

echo $userId

grep $userId record.txt

done <userid.txt

发现脚本可以打印echo $userId，却无法grep到？？而实际上record.txt里面是有这个id的！还真诡异！

先百度搜索了一下【grep 无法搜索变量】，还真有不少类似问题，比如：http://bbs.chinaunix.net/thread-123113-1-1.html

根据经验，对于这种诡异的问题，我首先会想到是不是系统有问题，要是系统有问题你怎么折腾都是错！

于是把他的文件拷贝到其他服务器，发现居然可以了！！！难道真是系统问题么？

第一台是SUSE Linux，第二台是Centos，难道和系统发行版有关系？

后来，同事在第二台服务器上完成了他的项目。但这个问题却一直留在我的脑子里，挥之不去。

今天，我决定再次研究下这个问题，看看是不是有其他原因。我先在那台SUSE Linux上，手工编写所需文件：

[root@localhost ~]# vim 1.txt

Shell

1111

3333

5555

1

2

3

1111

3333

5555

[root@localhost ~]# vim 2.txt

Shell

1111

2222

3333

4444

5555

6666

1

2

3

4

5

6

1111

2222

3333

4444

5555

6666

[root@localhost ~]# vim test.sh

Shell

#!/bin/bash

cat 1.txt|while read userId;

do

grep $userId 2.txt

done

1

2

3

4

5

#!/bin/bash

cat 1.txt|while read userId;

do

grep $userId 2.txt

done

结果，发现居然可以输出结果！证明这系统没有问题啊！于是再一次测试了一下昨天的脚本，发现还是无法输出。

于是使用 -x 参数 调试一下脚本：

先修改脚本代码：

Shell

#!/bin/bash

cat userid.txt|while read userId;

do

grep $userId record.txt

sleep 3

done

1

2

3

4

5

6

#!/bin/bash

cat userid.txt|while read userId;

do

grep $userId record.txt

sleep 3

done

然后，带 -x 参数执行：

Shell

[root@localhost ~]# sh -x test

+ cat userid.txt

+ read userId

+ grep $'0001\r' record.txt

+ sleep 3

+ read userId

+ grep $'0003\r' record.txt

+ sleep 3

+ read userId

+ grep $'0005\r' record.txt

+ sleep 3

1

2

3

4

5

6

7

8

9

10

11

[root@localhost ~]# sh -x test

+ cat userid.txt

+ read userId

+ grep $'0001\r' record.txt

+ sleep 3

+ read userId

+ grep $'0003\r' record.txt

+ sleep 3

+ read userId

+ grep $'0005\r' record.txt

+ sleep 3

难怪找不到，grep的变量已经变了！0001变成了 $'0001\r' ！

看到\r，立马想到是文本中的换行符，可为毛会输出换行符呢？想到博客以前写的《Linux终端：用cat命令查看不可见字符》，继续改了一下代码：

Shell

#!/bin/bash

cat -A userid.txt|while read userId;

do

grep $userId record.txt

sleep 3

done

1

2

3

4

5

6

#!/bin/bash

cat -A userid.txt|while read userId;

do

grep $userId record.txt

sleep 3

done

执行后恍然大悟：

Shell

[root@localhost ~]# sh -x test

+ cat -A userid.txt

+ read userId

+ grep '0001^M$' record.txt

+ sleep 3

+ read userId

+ grep '0003^M$' record.txt

+ sleep 3

+ read userId

+ grep '0005^M$' record.txt

+ sleep 3

1

2

3

4

5

6

7

8

9

10

11

[root@localhost ~]# sh -x test

+ cat -A userid.txt

+ read userId

+ grep '0001^M$' record.txt

+ sleep 3

+ read userId

+ grep '0003^M$' record.txt

+ sleep 3

+ read userId

+ grep '0005^M$' record.txt

+ sleep 3

原来是dos下的文本格式，问了下同事，他还真是从Windows下导过来的！ — —||

也就是说，userid.txt这个文本的换行符是Windows格式，在Linux下读取会带有^M。

所以解决上述问题，就很明了了，要么转换userid.txt的换行格式，要不就修改代码，去掉多余的字符！

试了下转换格式，发现居然转换不成功，可能是我没找对方法，暂时先不折腾了！

直接如下修改代码，就搞定了:

Shell

#!/bin/bash

cat -A userid.txt|while read userId;

do

#利用cut命令取出 ^ 之前的数字部分：

id=`echo $userId | cut -d"^" -f1`

grep $id record.txt

done

1

2

3

4

5

6

7

#!/bin/bash

cat -A userid.txt|while read userId;

do

#利用cut命令取出 ^ 之前的数字部分：

id=`echo $userId | cut -d"^" -f1`

grep $id record.txt

done

好了，搞了半天原来是dos和unix的换行符问题！o(︶︿︶)o 唉！还是经验不足啊！

网上那些问grep无法搜索变量的朋友，赶紧看看是不是文本格式造成的！现在，让我很纳闷的是，为毛在另一台centos系统可以直接grep？？为什么在SUSE系统就不行？

如果和发行版没关系的话，那造成2个不同结果的原因就只有一个：在我用sz+rz命令将所有文本传送到centos的过程中，文件很可能被自动转格式了！好吧，具体就不深究了，有兴趣的可以试试看。

cut是一个选取命令，就是将一段数据经过分析，取出我们想要的。一般来说，选取信息通常是针对“行”来进行分析的，并不是整篇信息分析的。

（1）其语法格式为：

cut [-bn] [file] 或 cut [-c] [file] 或 cut [-df] [file]

使用说明

cut 命令从文件的每一行剪切字节、字符和字段并将这些字节、字符和字段写至标准输出。

如果不指定 File 参数，cut 命令将读取标准输入。必须指定 -b、-c 或 -f 标志之一。

主要参数

-b ：以字节为单位进行分割。这些字节位置将忽略多字节字符边界，除非也指定了 -n 标志。

-c ：以字符为单位进行分割。

-d ：自定义分隔符，默认为制表符。

-f ：与-d一起使用，指定显示哪个区域。

-n ：取消分割多字节字符。仅和 -b 标志一起使用。如果字符的最后一个字节落在由 -b 标志的 List 参数指示的<br />范围之内，该字符将被写出；否则，该字符将被排除。

（2）cut一般以什么为依据呢? 也就是说，我怎么告诉cut我想定位到的剪切内容呢?

cut命令主要是接受三个定位方法：

第一，字节（bytes），用选项-b

第二，字符（characters），用选项-c

第三，域（fields），用选项-f

（3）以“字节”定位

举个例子吧，当你执行ps命令时，会输出类似如下的内容：

[rocrocket@rocrocket programming]$ who

rocrocket :0 2009-01-08 11:07

rocrocket pts/0 2009-01-08 11:23 (:0.0)

rocrocket pts/1 2009-01-08 14:15 (:0.0)

如果我们想提取每一行的第3个字节，就这样：

[rocrocket@rocrocket programming]$ who|cut -b 3

c

c

c

（4） 如果“字节”定位中，我想提取第3，第4、第5和第8个字节，怎么办?

-b支持形如3-5的写法，而且多个定位之间用逗号隔开就成了。看看例子吧：

[rocrocket@rocrocket programming]$ who|cut -b 3-5,8

croe

croe

croe

但有一点要注意，cut命令如果使用了-b选项，那么执行此命令时，cut会先把-b后面所有的定位进行从小到大排序，然后再提取。可不能颠倒定位的顺序哦。这个例子就可以说明这个问题：

[rocrocket@rocrocket programming]$ who|cut -b 8,3-5

croe

croe

croe

（5） 还有哪些类似“3-5”这样的小技巧，列举一下吧!

[rocrocket@rocrocket programming]$ who

rocrocket :0 2009-01-08 11:07

rocrocket pts/0 2009-01-08 11:23 (:0.0)

rocrocket pts/1 2009-01-08 14:15 (:0.0)

[rocrocket@rocrocket programming]$ who|cut -b -3

roc

roc

roc

[rocrocket@rocrocket programming]$ who|cut -b 3-

crocket :0 2009-01-08 11:07

crocket pts/0 2009-01-08 11:23 (:0.0)

crocket pts/1 2009-01-08 14:15 (:0.0)

想必你也看到了，-3表示从第一个字节到第三个字节，而3-表示从第三个字节到行尾。如果你细心，你可以看到这两种情况下，都包括了第三个字节“c”。

如果我执行who|cut -b -3,3-，你觉得会如何呢？答案是输出整行，不会出现连续两个重叠的c的。看：

[rocrocket@rocrocket programming]$ who|cut -b -3,3-

rocrocket :0 2009-01-08 11:07

rocrocket pts/0 2009-01-08 11:23 (:0.0)

rocrocket pts/1 2009-01-08 14:15 (:0.0)

（6）给个以字符为定位标志的最简单的例子吧!

下面例子你似曾相识，提取第3，第4，第5和第8个字符：

[rocrocket@rocrocket programming]$ who|cut -c 3-5,8

croe

croe

croe

不过，看着怎么和-b没有什么区别啊？莫非-b和-c作用一样? 其实不然，看似相同，只是因为这个例子举的不好，who输出的都是单字节字符，所以用-b和-c没有区别，如果你提取中文，区别就看出来了，来，看看中文提取的情况：

[rocrocket@rocrocket programming]$ cat cut\_ch.txt

星期一

星期二

星期三

星期四

[rocrocket@rocrocket programming]$ cut -b 3 cut\_ch.txt

�

�

�

�

[rocrocket@rocrocket programming]$ cut -c 3 cut\_ch.txt

一

二

三

四

看到了吧，用-c则会以字符为单位，输出正常；而-b只会傻傻的以字节（8位二进制位）来计算，输出就是乱码。

既然提到了这个知识点，就再补充一句，如果你学有余力，就提高一下。

当遇到多字节字符时，可以使用-n选项，-n用于告诉cut不要将多字节字符拆开。例子如下：

[rocrocket@rocrocket programming]$ cat cut\_ch.txt |cut -b 2

�

�

�

�

[rocrocket@rocrocket programming]$ cat cut\_ch.txt |cut -nb 2

[rocrocket@rocrocket programming]$ cat cut\_ch.txt |cut -nb 1,2,3

星

星

星

星

（7）域是怎么回事呢？解释解释:)

为什么会有“域”的提取呢，因为刚才提到的-b和-c只能在固定格式的文档中提取信息，而对于非固定格式的信息则束手无策。这时候“域”就派上用场了。如果你观察过/etc/passwd文件，你会发现，它并不像who的输出信息那样具有固定格式，而是比较零散的排放。但是，冒号在这个文件的每一行中都起到了非常重要的作用，冒号用来隔开每一个项。

我们很幸运，cut命令提供了这样的提取方式，具体的说就是设置“间隔符”，再设置“提取第几个域”，就OK了！

以/etc/passwd的前五行内容为例：

[rocrocket@rocrocket programming]$ cat /etc/passwd|head -n 5

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

[rocrocket@rocrocket programming]$ cat /etc/passwd|head -n 5|cut -d : -f 1

root

bin

daemon

adm

lp

看到了吧，用-d来设置间隔符为冒号，然后用-f来设置我要取的是第一个域，再按回车，所有的用户名就都列出来了！呵呵 有成就感吧！

当然，在设定-f时，也可以使用例如3-5或者4-类似的格式：

[rocrocket@rocrocket programming]$ cat /etc/passwd|head -n 5|cut -d : -f 1,3-5

root:0:0:root

bin:1:1:bin

daemon:2:2:daemon

adm:3:4:adm

lp:4:7:lp

[rocrocket@rocrocket programming]$ cat /etc/passwd|head -n 5|cut -d : -f 1,3-5,7

root:0:0:root:/bin/bash

bin:1:1:bin:/sbin/nologin

daemon:2:2:daemon:/sbin/nologin

adm:3:4:adm:/sbin/nologin

lp:4:7:lp:/sbin/nologin

[rocrocket@rocrocket programming]$ cat /etc/passwd|head -n 5|cut -d : -f -2

root:x

bin:x

daemon:x

adm:x

lp:x

（8）如果遇到空格和制表符时，怎么分辨呢？我觉得有点乱，怎么办？

有时候制表符确实很难辨认，有一个方法可以看出一段空格到底是由若干个空格组成的还是由一个制表符组成的。

[rocrocket@rocrocket programming]$ cat tab\_space.txt

this is tab finish.

this is several space finish.

[rocrocket@rocrocket programming]$ sed -n l tab\_space.txt

this is tab\tfinish.$

this is several space finish.$

看到了吧，如果是制表符（TAB），那么会显示为\t符号，如果是空格，就会原样显示。

通过此方法即可以判断制表符和空格了。

注意，上面sed -n后面的字符是L的小写字母哦，不要看错。

（9）我应该在cut -d中用什么符号来设定制表符或空格呢?

其实cut的-d选项的默认间隔符就是制表符，所以当你就是要使用制表符的时候，完全就可以省略-d选项，而直接用－f来取域就可以了。

如果你设定一个空格为间隔符，那么就这样：

[rocrocket@rocrocket programming]$ cat tab\_space.txt |cut -d ' ' -f 1

this

this

注意，两个单引号之间可确实要有一个空格哦，不能偷懒。

而且，你只能在-d后面设置一个空格，可不许设置多个空格，因为cut只允许间隔符是一个字符。

[rocrocket@rocrocket programming]$ cat tab\_space.txt |cut -d ' ' -f 1

cut: the delimiter must be a single character

Try `cut --help' for more information.

（10）cut有哪些缺陷和不足？

猜出来了吧？对，就是在处理多空格时。

如果文件里面的某些域是由若干个空格来间隔的，那么用cut就有点麻烦了，因为cut只擅长处理“以一个字符间隔”的文本内容