算术运算 行操作

操作历史

目录 [隐藏]

1. 简介

Bash 会保留用户的操作历史，即用户输入的每一条命令都会记录，默认是保存最近的500条命令。有了操作历史以后，就可以使用方向键的 ↑ 和 ↓ ，快速浏览上一条和下一条命令。

退出当前 Shell 的时候，Bash 会将用户在当前 Shell 的操作历史写入 `~/.bash_history` 文件，该文件默认储存500个操作。

环境变量 `HISTFILE` 总是指向这个文件。

```
$ echo $HISTFILE
/home/me/.bash_history
```

2. history 命令

`history` 命令会输出 `.bash_history` 文件的全部内容，即输出操作历史。

```
$ history
...
498 echo Goodbye
499 ls ~
500 cd
```

用户可以使用这个命令，查看最近的操作。相比直接读取 `.bash_history` 文件，它的优势在于所有命令之前加上了行号。最近的操作在最后面，行号最大。

如果想搜索某个以前执行的命令，可以配合 `grep` 命令搜索操作历史。

```
$ history | grep /usr/bin
```

上面命令返回 `.bash_history` 文件里面，那些包含 `/usr/bin` 的命令。

链接

本文源码

代码仓库

反馈

`history` 命令的 `-c` 参数可以清除操作历史，即清空 `.bash_history` 文件。

```
$ history -c
```

3. 环境变量

3.1 HISTTIMEFORMAT

通过定制环境变量 `HISTTIMEFORMAT` ， `history` 的输出结果还可以显示每个操作的时间。

```
$ export HISTTIMEFORMAT='%F %T  '
$ history
1  2013-06-09 10:40:12   cat /etc/issue
2  2013-06-09 10:40:12   clear
```

上面代码中， `%F` 相当于 `%Y - %m - %d` （年-月-日）， `%T` 相当于 `%H : %M : %S` （时:分:秒）。

只要设置 `HISTTIMEFORMAT` 这个环境变量，就会在 `.bash_history` 文件保存命令的执行时间戳。如果不设置，就不会保存时间戳。

3.2 HISTSIZE

环境变量 `HISTSIZE` 设置保存历史操作的数量。

```
$ export HISTSIZE=10000
```

上面命令设置保存过去10000条操作历史。

如果不希望保存本次操作的历史，可以设置 `HISTSIZE` 等于0。

```
export HISTSIZE=0
```

如果 `HISTSIZE=0` 写入用户主目录的 `~/.bashrc` 文件，那么就不会保留该用户的操作历史。如果写入 `/etc/profile` ，整个系统都不会保留操作历史。

3.3 HISTIGNORE

环境变量 `HISTIGNORE` 可以设置哪些命令不写入操作历史。

```
export HISTIGNORE='pwd:ls:exit'
```

上面示例设置， `pwd` 、 `ls` 、 `exit` 这三个命令不写入操作历史。

4. Ctrl + r

输入命令时，按下 `Ctrl + r` 快捷键，就可以搜索操作历史，选择以前执行过的命令。

`Ctrl + r` 相当于打开一个 `.bash_history` 文件的搜索接口，直接键入命令的开头部分，Shell 就会自动在该文件中反向查询（即先查询最近的命令），显示最近一条匹配的结果，这时按下回车键，就会执行那条命令。

5. ！命令

5.1 ！+ 行号

操作历史的每一条记录都有行号。知道了命令的行号以后，可以用 `感叹号 + 行号` 执行该命令。如果想要执行 `.bash_history` 里面的第8条命令，可以像下面这样操作。

```
$ !8
```

5.2 !- 数字

如果想执行本次 Shell 对话中倒数的命令，比如执行倒数第3条命令，就可以输入 `!-3` 。

```
$ touch a.txt
$ touch b.txt
$ touch c.txt

$ !-3
touch a.txt
```

上面示例中， `!-3` 返回倒数第3条命令，即 `touch a.txt` 。

它跟 `! + 行号` 的主要区别是，后者是在 `.bash_history` 文件中从头开始计算行数，而 `!- 数字` 是从底部开始向上计算行数。

5.3 !!

`!!` 命令返回上一条命令。如果需要重复执行某一条命令，就可以不断键入 `!!` ，这样非常方便。它等同于 `!-1` 。

```
$ echo hello
hello

$ !!
echo hello
hello
```

上面示例中， `!!` 会返回并执行上一条命令 `echo hello` 。

有时候，我们使用某条命令，系统报错没有权限，这时就可以使用 `sudo !!` 。

```
# 报错，没有执行权限
$ yum update
```

```
$ sudo !!
sudo yum update
```

上面示例中， `sudo !!` 返回 `sudo yum update` ，从而就可以正确执行了。

5.4 ！+ 搜索词

`感叹号 + 搜索词` 可以快速执行匹配的命令。

```
$ echo Hello World
Hello World

$ echo Goodbye
Goodbye

$ !e
echo Goodbye
Goodbye
```

上面例子中， `!e` 表示找出操作历史之中，最近的那一条以 `e` 开头的命令并执行。Bash 会先输出那一条命令 `echo Goodbye` ，然后直接执行。

同理， `!echo` 也会执行最近一条以 `echo` 开头的命令。

```
$ !echo
echo Goodbye
Goodbye

$ !echo H
echo Goodbye H
Goodbye H

$ !echo H G
echo Goodbye H G
Goodbye H G
```

注意， `感叹号 + 搜索词` 语法只会匹配命令，不会匹配参数。所以 `!echo H` 不会执行 `echo Hello World` ，而是会执行

`echo Goodbye`，并把参数 `H` 附加在这条命令之后。同理，`!echo H G` 也是等同于 `echo Goodbye` 命令之后附加 `H G`。

由于 `感叹号 + 搜索词` 会扩展成以前执行过的命令，所以含有 `!` 的字符串放在双引号里面，必须非常小心，如果它后面有非空格的字符，就很有可能报错。

```
$ echo "I say:\"hello!\""
bash: !\: event not found
```

上面的命令会报错，原因是感叹号后面是一个反斜杠，Bash 会尝试寻找，以前是否执行过反斜杠开头的命令，一旦找不到就会报错。解决方法就是在感叹号前面，也加上反斜杠。

```
$ echo "I say:\"hello\!\""
I say:"hello\!"
```

5.5 !? + 搜索词

`!? + 搜索词` 可以搜索命令的任意部分，包括参数部分。它跟 `! + 搜索词` 的主要区别是，后者是从行首开始匹配。

```
$ cat hello.txt
Hello world ..!

$ !?hello.txt
cat hello.txt
Hello world ..!
```

上面示例中，`!?hello.txt` 会返回最近一条包括 `hello.txt` 的命令。

5.6 !$，!*

`!$` 代表上一个命令的最后一个参数，它的另一种写法是 `$_`。

`!*` 代表上一个命令的所有参数，即除了命令以外的所有部分。

```
$ cp a.txt b.txt
$ echo !$
b.txt

$ cp a.txt b.txt
$ echo !*
a.txt b.txt
```

上面示例中，`!$` 代表上一个命令的最后一个参数（`b.txt`），`!*` 代表上一个命令的所有参数（`a.txt b.txt`）。

如果想匹配上一个命令的某个指定位置的参数，使用 `!:n` 。

```
$ ls a.txt b.txt c.txt

$ echo !:2
b.txt
```

上面示例中，`!:2` 返回上一条命令的第二个参数（`b.txt`）。

这种写法的 `!:$` ，代表上一个命令的最后一个参数。事实上，`!$` 就是 `!:$` 的简写形式。

```
$ ls a.txt b.txt c.txt

$ echo !:$
echo c.txt
c.txt
```

上面示例中，`!:$` 代表上一条命令的最后一个参数（`c.txt`）。

如果想匹配更久以前的命令的参数，可以使用 `!<命令>:n`（指定位置的参数）和 `!<命令>:$` （最后一个参数）。

```
$ ls !mkdir:$
```

上面示例中， `!mkdir:$` 会返回前面最后一条 `mkdir` 命令的最后一个参数。

```
$ ls !mk:2
```

上面示例中， `!mk:2` 会返回前面最后一条以 `mk` 开头的命令的第二个参数。

5.7 !:p

如果只是想输出上一条命令，而不是执行它，可以使用 `!:p` 。

```
$ echo hello

$ !:p
echo hello
```

上面示例中， `!:p` 只会输出 `echo hello` ，而不会执行这条命令。

如果想输出最近一条匹配的命令，而不执行它，可以使用 `!<命令>:p` 。

```
$ !su:p
```

上面示例中， `!su:p` 会输出前面最近一条以 `su` 开头的命令，而不执行它。

6. ^string1^string2

`^string1^string2` 用来执行最近一条包含 `string1` 的命令，将其替换成 `string2` 。

```
$ rm /var/log/httpd/error.log
$ ^error^access
rm /var/log/httpd/access.log
```

上面示例中，`^error^access` 将最近一条含有 `error` 的命令里面的 `error` ，替换成 `access` 。

7. histverify 参数

上面的那些快捷命令（比如 `!!` 命令），都是找到匹配的命令后，直接执行。如果希望增加一个确认步骤，先输出是什么命令，让用户确认后再执行，可以打开 Shell 的 `histverify` 选项。

```
$ shopt -s histverify
```

打开 `histverify` 这个选项后，使用 `!` 快捷键所返回的命令，就会先输出，等到用户按下回车键后再执行。

8. 快捷键

下面是其他一些与操作历史相关的快捷键。

- `Ctrl + p` ：显示上一个命令，与向上箭头效果相同（previous）。
- `Ctrl + n` ：显示下一个命令，与向下箭头效果相同（next）。
- `Alt + <` ：显示第一个命令。
- `Alt + >` ：显示最后一个命令，即当前的命令。
- `Ctrl + o` ：执行历史文件里面的当前条目，并自动显示下一条命令。这对重复执行某个序列的命令很有帮助。

9. 参考链接

- Bash bang commands: A must-know trick for the Linux command line

算术运算 行操作

分享本文 

联系：contact@wangdoc.com