## linux中的set命令: "set -e" 与 "set -o pipefail"

(2012-07-18 14:43:43)

|  |  |
| --- | --- |
| 标签： [linux](http://search.sina.com.cn/?c=blog&q=linux&by=tag)[set](http://search.sina.com.cn/?c=blog&q=set&by=tag) | 分类： [班门弄斧碎碎念](http://blog.sina.com.cn/s/articlelist_2343609277_3_1.html) |

工作中经常在shell脚本中看到set的这两个用法，但就像生活中的很多事情，习惯导致忽视，直到出现问题才引起关注。  
  
1. set -e  
set命令的-e参数，linux自带的说明如下：  
*"Exit immediately if a simple command exits with a non-zero status."*  
也就是说，在"set -e"之后出现的代码，一旦出现了返回值非零，整个脚本就会立即退出。有的人喜欢使用这个参数，是出于保证代码安全性的考虑。但有的时候，这种美好的初衷，也会导致严重的问题。  
  
真实案例：  
脚本a.sh开头使用了"set -e"，且能正常运行。在几个月或更久以后，因需求升级，在脚本中增加了3行hadoop操作：  
*#!/bin/bash  
set -e  
...  
/home/work/.../hadoop dfs -rmr /app/.../dir  
/home/work/.../hadoop dfs -mkdir /app/.../dir  
/home/work/.../hadoop dfs -put file\_1 /app/.../dir/  
...*  
这几行hadoop命令逻辑很简单：在hdfs上清除并新建一个目录，并将一份本地文件推送至这个目录，供后续使用。将这几行单拎出来，在命令行下执行，除了提示待删除的目录不存在，并没有什么问题，文件还是会被推送到指定的地方。  
  
但第一次执行这个脚本的时候，却失败退出了，且导致调用该脚本的程序整体退出，造成了严重的后果。原因是hdfs上还没有这个目录，rmr这一行会返回255，这个值被脚本前方的"set -e"捕捉到，直接导致了脚本退出。  
  
新增的代码本身并没有问题，先删除再新建目录，反而是保证数据安全的比较规范的操作，删除命令本身的容错性，可以保证后续命令正常执行。事实是这个脚本有好几百行，且逻辑比较复杂，在增加这几行代码的时候，开发人员已经不记得这个脚本里还有个"set -e"埋伏着了。  
  
可见设置"set -e"，在脚本开发过程中可能很有帮助，而在开发完成后，特别是对于后期可能有升级的脚本，则可能是埋下了安全隐患。  
  
2. set -o pipefail  
对于set命令-o参数的pipefail选项，linux是这样解释的：  
*"If set, the return value of a pipeline is the value of the last (rightmost) command to exit with a  non-zero  status,or zero if all commands in the pipeline exit successfully.  This option is disabled by default."*  
  
设置了这个选项以后，包含管道命令的语句的返回值，会变成最后一个返回非零的管道命令的返回值。听起来比较绕，其实也很简单：  
  
# test.sh *set -o pipefail  
ls ./a.txt |echo "hi" >/dev/null  
echo $?*  
  
运行test.sh，因为当前目录并不存在a.txt文件，输出：  
*ls: ./a.txt: No such file or directory  
1  #* *设置了set -o pipefail，返回从右往左第一个非零返回值，即ls的返回值1*  
  
注释掉*set -o pipefail* 这一行，再次运行，输出：  
*ls: ./a.txt: No such file or directory  
0  # 没有set -o pipefail，默认返回最后一个管道命令的返回值*

1. **Bash Shell-使用内置的set和shopt命令设置bash选项**

[转载](javascript:;) [yangji008](http://blog.51cto.com/professor) [2014-11-05 14:23](javascript:;) [评论(0)](http://blog.51cto.com/professor/1572113#comment) [6512人阅读](javascript:;)

**使用内置的set和shopt命令设置bash选项**

    set命令可以用来定制shell环境，使用选项“o”来打开或者关闭选项。例如打开选项：set -o 选项，关闭选项目：set +o 选项。

    例如要打开vi交互式命令行编辑，则如下：

|  |
| --- |
| [root@localhost ~]# **set -o**   #查看当前设置情况 allexport       off braceexpand     on emacs           on errexit         off errtrace        off functrace       off hashall         on histexpand      on history         on ignoreeof       off interactive-comments    on keyword         off monitor         on noclobber       off noexec          off noglob          off nolog           off notify          off nounset         off onecmd          off physical        off pipefail        off posix           off privileged      off verbose         off vi              off             #关闭中 xtrace          off [root@localhost ~]# **set -o vi**   #把vi选项打开 [root@localhost ~]# **set -o**      #查看当前设置情况 allexport       off braceexpand     on emacs           off errexit         off errtrace        off functrace       off hashall         on histexpand      on history         on ignoreeof       off interactive-comments    on keyword         off monitor         on noclobber       off noexec          off noglob          off nolog           off notify          off nounset         off onecmd          off physical        off pipefail        off posix           off privileged      off verbose         off vi              on                #已经打开 xtrace          off |

    内置set命令选项

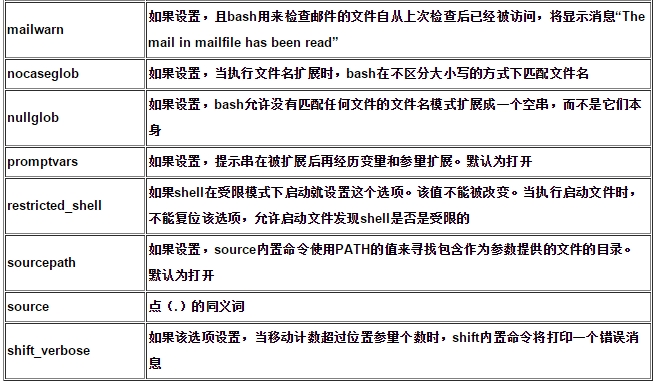
|  |  |  |
| --- | --- | --- |
| 选项名 | 快捷开关 | 含义 |
| allexport | -a | 从这个选项中被设置开始就自动标明要输出的新变量或修改过的变量，直至选项被复位 |
| braceexpand | -B | 打开花括号扩展，它是一个默认设置 |
| emacs |  | 使用emacs内置编辑器进行命令行编辑，是一个默认设置 |
| errexit | -e | 当命令返回一个非零退出状态（失败）时退出。读取初始化文件时不设置 |
| histexpand | -H | 执行历史替换时打开!和!!扩展，是一个默认设置 |
| history |  | 打开命令行历史、默认为打开 |
| ignoreeof |  | 禁止用EOF(Ctrl+D)键退出shell。必须键入exit才能退出。等价于设置shell变量IGNOREEOF=10 |
| keyword | -k | 将关键字参数放到命令的环境中 |
| interactive-comments |  | 对于交互式shell，把#符后面的文本作为注释 |
| monitor | -m | 设置作业控制 |
| noclobber | -C | 防止文件在重定向时被重写 |
| noexec | -n | 读命令，但不执行。用来检查脚本的语法。交互式运行时不开启 |
| noglob | -d | 禁止用路径名扩展。即关闭通配符 |
| notify | -b | 后台作业完成时通知用户 |
| nounset | -u | 扩展一个未设置的变量时显示一个错误信息 |
| onecmd | -t | 在读取和执行命令后退出 |
| physical | -P | 设置时，在键入cd或pwd禁止符号链接。用物理目录代替 |
| privileged | -p | 设置后，shell不读取.profile或ENV文件，且不从环境继承shell函数，将自动为setuid脚本开启特权 |
| verbose | -v | 为调试打开verbose模式 |
| vi |  | 使用vi内置编辑器进行命令行编辑 |
| xtrace | -x | 为调试打开echo模式 |
|  |  |  |

    shopt命令是set命令的一种替代，很多方面都和set命令一样，但它增加了很多选项。可有使用“-p”选项来查看shopt选项的设置。“-u”开关表示一个复位的选项，“-s”表示选项当前被设置。

|  |
| --- |
| [root@localhost ~]# **shopt -p**    #查看当前设置情况 shopt -u cdable\_vars shopt -u cdspell shopt -u checkhash shopt -s checkwinsize shopt -s cmdhist shopt -u dotglob shopt -u execfail shopt -s expand\_aliases shopt -u extdebug shopt -u extglob shopt -s extquote shopt -u failglob shopt -s force\_fignore shopt -u gnu\_errfmt shopt -u histreedit shopt -u histappend shopt -u histverify shopt -s hostcomplete shopt -u huponexit shopt -s interactive\_comments shopt -u lithist shopt -s login\_shell shopt -u mailwarn shopt -u no\_empty\_cmd\_completion shopt -u nocaseglob shopt -u nullglob shopt -s progcomp shopt -s promptvars shopt -u restricted\_shell shopt -u shift\_verbose shopt -s sourcepath shopt -u xpg\_echo  [root@localhost ~]# **shopt -s cdspell**   #把cdspell选项打开 [root@localhost ~]# **shopt -p cdspell**    #打印cdspell设置 shopt -s cdspell [root@localhost ~]# **cd /raot**            #故意拼写错误 /root [root@localhost ~]# **pwd**                 #查看当前目录 /root [root@localhost ~]# cd **/usr/loca/bin**#故意拼写错误 /usr/local/bin [root@localhost bin]# **pwd**               #查看当前目录    /usr/local/bin [root@localhost bin]# **shopt -u cdspell** #把cdspell选项打开 [root@localhost bin]# **shopt -p cdspell** #打印cdspell设置 shopt -u cdspell [root@localhost bin]# **cd /raot**         #故意拼写错误 -bash: cd: /raot: 没有那个文件或目录     #进入目录错误 |

 shopt命令选项

[](http://s3.51cto.com/wyfs02/M02/4D/C9/wKiom1RZwxeiVbE_AAbeDmATTQA475.jpg)

[](http://s3.51cto.com/wyfs02/M00/4D/C8/wKioL1RZw3mDwq5KAALb_ZTv-XM899.jpg)

# ****shopt nullglob, failglob, extglob, globstar的用法****

在读pig的时候发现，它的shell中有这样一个用法：

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. #!/usr/bin/env bash
3. # so that filenames w/ spaces are handled correctly in loops below
4. IFS=
6. shopt -s extglob
7. shopt -s nullglob
9. for f in $PIG\_HOME/lib/\*.jar; do
10. CLASSPATH=${CLASSPATH}:$f;
11. done
13. JYTHON\_JAR=`echo ${PIG\_HOME}/lib/jython\*.jar`
15. for f in $PIG\_HOME/share/pig/lib/\*.jar; do
16. CLASSPATH=${CLASSPATH}:$f;
17. done
19. ...
20. # restore ordinary behaviour
21. unset IFS
22. ...
24. shopt -u nullglob
25. shopt -u extglob

其中nullglob的是什么意思呢？

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. man shopt
2. -s enable
3. -u disable
5. extglob
6. If set, the extended pattern matching features described above under Pathname Expansion are enabled.
7. nullglob
8. If set, bash allows patterns which match no files (see Pathname Expansion above) to expand to
9. a null string, rather than themselves.

（通过 man bash　可以检索到对　“Pathname Expansion”的解释,其中陈述了extglob和nullglob对file 匹配时的用法）

**extglob 的另外用法：**

<http://blog.csdn.net/ciedecem/article/details/18315543>

对于nullglob来说, 就是在使用shell 的通配符\*匹配文件时，如果没有匹配到任何文件时，那就会输出null string，而不是通配符字符本身。 为什么这么说呢？

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. >shopt nullglob
2. nullglob        off
3. >mkdir tmp
4. >cd tmp
5. >for i in \*; do echo "file: $i"; done
6. file: \* 【这里把通配符\*作为字符输出了】
7. >shopt -s nullglob
8. >shopt nullglob
9. nullglob        on
10. >for i in \*; do echo "file: $i"; done
11. 【null string，没有结果输出】

由此可以，在使用\*来匹配一个文件时，如果该目录下并没有这个文件，这个通配符\*就会作为一个字符，被当做一个文件来处理，这是很危险的。因此，在匹配file 时，nullglob是很有必要enable的。

当然也可以使用failglob，在没有匹配时报错。

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. failglob
2. If set, patterns which fail to match filenames during pathname expansion result in an expansion error.

example:

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. >shopt failglob
2. failglob        off
3. >for i in \*; do echo "file: $i"; done
4. file: \*
6. >shopt -s failglob
7. >for i in \*; do echo "file: $i"; done
8. -bash: no match: \*

另外， 单引号可以防止\*号被替换。

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. [login@sonora resources]$ find ../ -name \*.yidf
2. find: missing argument to `-name'
3. [login@sonora resources]$ find ../ -name '\*.yidf'
4. ../resources/slingshot.di.configservice.prod/image/runtime
5. ../resources/slingshot.di.configservice.stage/image/runtime
6. ../resources/common/pkg.yidf
7. ../resources/slingshot.di.configservice.qe/image/runtime

#### globstar的例子

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. globstar
2. If set, the pattern \*\* used in a pathname expansion context will match a files and zero or more directories
3. and subdirectories.  If the pattern is followed by a /, only directories and subdirectories match.

enable **globstar**后，\*\*递归的匹配了所有文件和目录， 如果\*\*后面跟着/(即是\*\*/)，则只匹配目录。  
默认情况下，globstar是关闭的，也就是\*\*与\*是一样的，

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. igi@gentoo ~/test $ tree
2. .
3. ├── a
4. │   ├── 1
5. │   ├── 2
6. │   ├── 3
7. │   ├── 4
8. │   └── 5
9. └── c
10. ├── 2.txt
11. ├── 3.txt
12. ├── 4.txt
13. └── dir
15. 3 directories, 8 files
16. igi@gentoo ~/test $ shopt -s globstar
17. igi@gentoo ~/test $ shopt globstar
18. globstar        on
19. igi@gentoo ~/test $ echo \*
20. a c
21. igi@gentoo ~/test $ echo \*\*
22. a a/1 a/2 a/3 a/4 a/5 c c/2.txt c/3.txt c/4.txt c/dir
23. igi@gentoo ~/test $ echo \*/
24. a/ c/
25. igi@gentoo ~/test $ echo \*\*/
26. a/ c/ c/dir/

可以看到打开globstar后，\*\*递归的匹配了所有文件和目录， 如果\*\*后面跟着/(即是\*\*/)，则只匹配目录。  
问题来了，如果递归显示以.txt结尾到文件，是不是\*\*.txt? 非也，来看看

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. igi@gentoo ~/test $ shopt -s globstar
2. igi@gentoo ~/test $ shopt globstar
3. globstar        on
4. igi@gentoo ~/test $ find . -name '\*.txt'
5. ./a.txt
6. ./c/3.txt
7. ./c/2.txt
8. ./c/4.txt
9. igi@gentoo ~/test $ echo \*\*.txt
10. a.txt
11. igi@gentoo ~/test $ echo \*\*/\*.txt
12. a.txt c/2.txt c/3.txt c/4.txt

看到了吧，\*\*.txt是无法递归的，而\*\*/\*.txt就可以了，同理， foo\*\*这样也不行，\*\*/foo\*这样到才可以。  
这个功能是bash4才有的哦，使用之前，先确认下你到bash版本。

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. >bash -version
2. GNU bash, version 4.1.2(1)-release (x86\_64-redhat-linux-gnu)

[以上参考http://www.igigo.net/archives/category/shell]

但是当enable了failglob之后，它就会把所有出现的\*当做通配符，去匹配文件名了。例如：

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. >shopt failglob
2. failglob        on
3. >let num=3\*3
4. -bash: no match: num=3\*3

即便是touch 3\*3的文件(3\\*3)，这个算术运算也不成功。

但diable该选项之后，就ok了。

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. >shopt -u failglob
2. >shopt  failglob
3. failglob        off
4. >let num=3\*3
5. >echo $num
6. 9

**\*号乘法运算, \*\*冪运算**

**[html]** [view plain](http://blog.csdn.net/ciedecem/article/details/11978277) [copy](http://blog.csdn.net/ciedecem/article/details/11978277)

1. >let num=2\*\*4
2. >echo $num
3. 16