

Linux系统基本操作

1、帮助(man)

man cd man ls

```
BASH_BUILTINS(1)                                General Commands Manual                                BASH_BUILTINS(1)

NAME
  bash, :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs, disown,
  echo, enable, eval, exec, exit, export, false, fc, fg, getopts, hash, help, history, jobs, kill, let, local, logout, mapfile, popd,
  printf, pushd, pwd, read, readonly, return, set, shift, shopt, source, suspend, test, times, trap, true, type, typeset, ulimit, umask,
  unalias, unset, wait - bash built-in commands, see bash(1)

BASH BUILTIN COMMANDS
  Unless otherwise noted, each builtin command documented in this section as accepting options preceded by - accepts -- to signify the
  end of the options. The :, true, false, and test builtins do not accept options and do not treat -- specially. The exit, logout,
  break, continue, let, and shift builtins accept and process arguments beginning with - without requiring --. Other builtins that
  accept arguments but are not specified as accepting options interpret arguments beginning with - as invalid options and require -- to
  prevent this interpretation.
  : [arguments]
    No effect; the command does nothing beyond expanding arguments and performing any specified redirections. A zero exit code is
    returned.
  . filename [arguments]
  source filename [arguments]
    Read and execute commands from filename in the current shell environment and return the exit status of the last command exe-
    cuted from filename. If filename does not contain a slash, file names in PATH are used to find the directory containing file-
    name. The file searched for in PATH need not be executable. When bash is not in posix mode, the current directory is searched
    if no file is found in PATH. If the sourcepath option to the shopt builtin command is turned off, the PATH is not searched.
    If any arguments are supplied, they become the positional parameters when filename is executed. Otherwise the positional
    parameters are unchanged. The return status is the status of the last command exited within the script (0 if no commands are
    executed), and false if filename is not found or cannot be read.
  alias [-p] [name=value] ...
Manual page cd(1) line 1 (press h for help or q to quit)
```

2、创建文件(touch):

touch 或 touch{1...4}.txt;

3、权限

1) 权限解读

r; 可读

w; 可写

x; 可执行

第一个rwx: 当前用户

第二个rwx: 用户组

第三个rwx: 其他用户

```
[root@master zpf]# ll
总用量 808184
-rwxr--r--. 1 root root    71 4月  6 18:34 1.txt
-rwxrwxrwx. 1 root root     0 4月  6 18:06 2.txt
-rw-r--r--. 1 root root     0 4月  6 18:06 3.txt
-rw-r--r--. 1 root root 286170958 11月 22 21:40 apache-hive-2.3.9-bin.tar.gz
-rw-r--r--. 1 root root  9623007 11月 22 20:58 apache-zookeeper-3.5.9-bin.tar.gz
-rw-r--r--. 1 root root 272812222 11月 22 21:43 hbase-2.3.7-bin.tar.gz
-rwxr--r--. 1 root root    31 4月  6 20:22 hello.sh
```

2) 加权限

```
chmod u g o
chmod u+x(可执行) 文件路径
chmod 777 文件路径      加全部权限
权限去除:  chmod u-x/r/w
```

4、查看当前文件:

```
cat 文件路径
```

5、编辑文件

vim是vi的加强版，无文件可添加文件，提示为中文

```
vim test1.sh
```

6、跳转行数 (:set nu)

gg 第一行
G 最后一行
数字+G 固定到数字所对应的那一行(nG)
w可使鼠标往下走

```
1 #!/bin/bash
2 echo '===== $n ====='
3 echo $0 #这是获取当前路径
4 echo $1 #这是第一个参数
5 echo $2 #这是第二个参数
6 echo $# #这是获取参数个数
```

```
~
~
~
~
:set nu
```

7、复制、粘贴、删除内容

1) 复制

复制当前行: yy
复制当前行往下的所有: yG
复制几行内容: 数字+y(ny)

2) 粘贴

粘贴: p

3) 删除:

dd删除当前行
dG删除当前行到所有
nd删除几行

8、shell脚本

1)概述

- 1、shell是一个面向过程的语言
- 2、解释器有bash 和sh,
- 3、bin下面存放启动文件

2) shell操作

1) 头文件: #!/bin/bash

2) 输出(echo): echo "hello world"

3) 查看系统变量的值: echo \$HOME

4) 变量(调用时加'\$符号'): age=18 echo \$age

5) 编辑脚本:

```
#!/bin/bash
echo
'===== $n ====='
echo $0 #这是获取当前路径
echo $1 #这是第一个参数
echo $2 #这是第二个参数
echo $3 #这是第三个参数
echo $# #这是获取参数个数
```

```
#!/bin/bash
echo '===== $n ====='
echo $0 #这是获取当前路径
echo $1 #这是第一个参数
echo $2 #这是第二个参数
echo $# #这是获取参数个数
~
~
```

6) 启动脚本:

命令: sh 文件路径 one two (十个以上的参数<\${10}>)

```
sh test1.sh 12 13
```

```
[root@master zpf]# sh test1.sh 12 13
===== $n =====
test1.sh
12
13
2
[root@master zpf]#
```

7) 运算值的转换:

num=\$((10+10)) 或 num=\$((10+10))

命令: echo \$num

8) 检查语句合法性 (echo \$? 判断语句是否正确) :

```
[ condition ] (condition<语句>前后必须有空格)
[ 23 -ge 22 ]
echo $?
```

```
[root@master zpf]# [ 23 -ge 22 ]
[root@master zpf]# echo $?
0
[root@master zpf]#
```

9) 判断当前文件是否具有可读权限

```
[ -r hello.sh ]  
echo $?
```

```
[root@master zpf]# [ -r hello.sh ]  
[root@master zpf]# echo $?  
0  
[root@master zpf]#
```

10) 分支结构

1、不同：

一般true=1、false=0
Centos linux true=0、false=1

一般判断用 '<、>、='
Centos linux（字符串）'=、!='（数字）'-eq 等于、-ne 不等于、-lt 小于、-gt 大于、-le 小于等于、-ge 大于或等于'

2、单分支：

```
#!/bin/bash  
name="aa"  
if $name="aa"  
then  
echo "这是一个正确的语句"  
else  
echo "这是一个不正确的语句"  
fi
```

3、多分支：

```
#!/bin/bash
if [ $1 -eq 2 ]
then
echo "这是第一个数 or 111"
elif [ $2 -eq 2 ]
then
echo "这是第二个数 or 111"
fi
```

```
#!/bin/bash
if [ $1 -eq 1 ]
then
echo "111"
elif [ $1 -eq 2 ]
then
echo "222"
else
echo "这没有了"
fi
~
[root@master zpf]# sh test2.sh 1
111
[root@master zpf]# sh test2.sh 2
222
[root@master zpf]#
```

4、case:

```
#!/bin/bash
case $1 in
"1")
echo "这是第一个数"
;;    #相当于break
"2")
echo "这是第二个数"
;;    #相当于break
"*) #相当于default
echo "这是其他数"
;;
esac
```

```
#!/bin/bash
case $1 in
"1")
echo "这是第一个数"
;; #相当于break
"2")
echo "这是第二个数"
;; #相当于break
"3") #相当于default
echo "这是其他数"
;;
esac

~
~

[root@master zpf]# sh test3.sh 1
这是第一个数
[root@master zpf]# sh test3.sh 2
这是第二个数
[root@master zpf]# sh test3.sh 3
[root@master zpf]#
```

5、for循环:

```
#!/bin/bash
for i in windows,Linux,Macos
do
echo "操作系统有$i"
done
```

6、注意:

\$*获取当前参数的个数（作为整体输出）
\$@获取当前参数的个数（作为单个输出）

```
#!/bin/bash
echo '=====$*====参数作为整体'
for i in "$*"
do
echo $i
done

echo '=====$@====参数作为单个输出'
for j in "$@"
do
echo $j
done
```

```
sh test4.sh windows Linux MacOS
```

```
#!/bin/bash
echo '=====$*====参数作为整体'
for i in "$*"
do
echo $i
done

echo '=====$@====参数作为单个输出'
for j in "$@"
do
echo $j
done
```

```
[root@master zpf]# vim test4.sh
[root@master zpf]# sh test4.sh Windows Linux MacOS
=====$*====参数作为整体
Windows Linux MacOS
=====$@====参数作为单个输出
Windows
Linux
Macos
[root@master zpf]#
```

7、考试（做一个输出的+）

```
echo $((1+2))
echo [$1+$2]
```

```
[root@master zpf]# echo $((1+2))
3
[root@master zpf]#
```