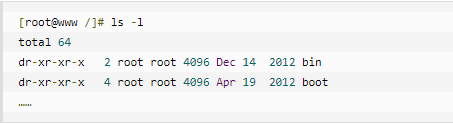
**linux学习文档:** **https://linuxtools-rst.readthedocs.io/zh\_CN/latest/base/index.html**

### linux 文件基本属性

使用ll命令来显示一个文件的属性以及文件所属的用户和组



第一个属性：

* 当为[ d ]则是目录
* 当为[ - ]则是文件；
* 若是[ l ]则表示为链接文档(link file)；
* 若是[ b ]则表示为装置文件里面的可供储存的接口设备(可随机存取装置)；
* 若是[ c ]则表示为装置文件里面的串行端口设备，例如键盘、鼠标(一次性读取装置)。



rwx的顺序不会变，若没有对应的权限，就用-表示；

更改文件属性：

* chgrp：更改文件属组

|  |
| --- |
| chgrp [-R] 数组名 文件名 |

-R：递归更改文件属组，就是在更改某个目录文件的属组时，如果加上-R的参数，那么该目录下的所有文件的属组都会更改。

* chown：更改文件属主，也可以同时更改文件属组

|  |
| --- |
| chown [-R] 属主名 文件名  chown [-R] 属主名：属组名 文件名  chown [-R] :属组名 文件名 |

* chmod：更改文件9个属性

第一种：

r:4 w:2 x:1

使用数字代表权限：

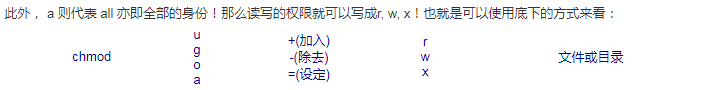
9个权限3个一组，分别是owner（user）/group/others,用每组权限的数字之和表示

赋予全部权限：

|  |
| --- |
| chmod [-R] 777 文件或目录 |

第二种：符号类型

三种身份user/group/others，分别用 u g o来表示



a代表三种身份同时操作

|  |
| --- |
| chmod a-x install.log |

|  |
| --- |
| chmod u+x,g-w,o-w install.log |

|  |
| --- |
| chmod u=r,g=rwx,o=--- install.log |

### 文件和目录管理

#### 绝对路径和相对路径

绝对路径：从根目录 / 写起 例如：/usr/share/doc

相对路径： ../..

#### ls

参数：

-a ：全部的文件，连同隐藏档( 开头为 . 的文件) 一起列出来(常用)

-d ：仅列出目录本身，而不是列出目录内的文件数据(常用)

-l ：长数据串列出，包含文件的属性与权限等等数据；(常用)

|  |
| --- |
| ls –al 目录名称 |

目录名称可以不写，默认显示当前目录下的

#### cd

后可跟相对路径和绝对路径

/ 代表根目录

~代表 /root目录

#### pwd

显示当前目录（print working directory）

#### mkdir

|  |
| --- |
| mkdir [-mp] 目录名称 |

-m ：配置文件的权限喔！直接配置，不需要看默认权限 (umask) 的脸色～

-p ：帮助你直接将所需要的目录(包含上一级目录)递归创建起来！

|  |
| --- |
| mkdir -m 777 –p 1/2/3 |

#### rmdir

删除空目录

|  |
| --- |
| rmdir [-p] 目录名称 |

-p ：连同上一级『空的』目录也一起删除

|  |
| --- |
| rmdir 1/2/3 只删除3  rmdir -p 1/2/3 1、2、3都会被删除 |

#### rm

移除文件和目录

|  |
| --- |
| rm [-fir] 文件或目录 |

-f ：就是 force 的意思，忽略不存在的文件，不会出现警告信息；

-i ：互动模式，在删除前会询问使用者是否动作

-r ：递归删除啊！最常用在目录的删除了！这是非常危险的选项！！！

注意:只有 -r 的时候rm才能删除目录，其他情况不能。

#### cp

复制文件和目录，可以更改文件或目录名

|  |
| --- |
| cp [-adfilprsu] 来源档(source) 目标档(destination) |

-a：相当於 -pdr 的意思，至於 pdr 请参考下列说明；(常用)

-d：若来源档为连结档的属性(link file)，则复制连结档属性而非文件本身；

-f：为强制(force)的意思，若目标文件已经存在且无法开启，则移除后再尝试一次；

-i：若目标档(destination)已经存在时，在覆盖时会先询问动作的进行(常用)

-l：进行硬式连结(hard link)的连结档创建，而非复制文件本身；

-p：连同文件的属性一起复制过去，而非使用默认属性(备份常用)；

-r：递归持续复制，用於目录的复制行为；(常用)

-s：复制成为符号连结档 (symbolic link)，亦即『捷径』文件；

-u：若 destination 比 source 旧才升级 destination ！

#### mv

移动文件或目录，或修改名称

|  |
| --- |
| mv [-fiu] source destination |

-f ：force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖；

-i ：若目标文件 (destination) 已经存在时，就会询问是否覆盖！

-u ：若目标文件已经存在，且 source 比较新，才会升级 (update)

#### touch

修改文件或者目录的时间属性，包括存取时间和更改时间。若文件不存在，**新建**一个。

touch 文件/目录

#### vi

vi 文件 : 新建文件

#### link

ln -s [源文件] [目标文件]

软链接：以路径的形式存在，类似于快捷方式，不占磁盘空间

ln [源文件] [目标文件]

硬链接：已文件副本形式存在，两文件各项属性相同

软链接和硬链接，无论改动哪一个文件，链接文件都会跟着改动！！

### 文件内容查看

#### cat

从第一行开始显示文件内容

|  |
| --- |
| cat [-AbEnTv] |

-A ：相当於 -vET 的整合选项，可列出一些特殊字符而不是空白而已；

-b ：列出行号，仅针对非空白行做行号显示，空白行不标行号！

-E ：将结尾的断行字节 $ 显示出来；

-n ：列印出行号，连同空白行也会有行号，与 -b 的选项不同；

-T ：将 [tab] 按键以 ^I 显示出来；

-v ：列出一些看不出来的特殊字符

#### tac

从最后一行开始显示，即倒着显示，跟cat相反

#### nl

显示行号,输出文件内容

|  |
| --- |
| nl 文件 |

#### more

一页一页翻动

在 more 这个程序的运行过程中，你有几个按键可以按的：

空白键 (space)：代表向下翻一页；

Enter ：代表向下翻『一行』；

/字串 ：代表在这个显示的内容当中，向下搜寻『字串』这个关键字；

:f ：立刻显示出档名以及目前显示的行数；

q ：代表立刻离开 more ，不再显示该文件内容。

b 或 [ctrl]-b ：代表往回翻页，不过这动作只对文件有用，对管线无用。

#### less

一页一页翻动（可以往后翻）

less运行时可以输入的命令：

空白键 ：向下翻动一页；

[pagedown]：向下翻动一页；

[pageup] ：向上翻动一页；

/字串 ：向下搜寻『字串』的功能；

?字串 ：向上搜寻『字串』的功能；

n ：重复前一个搜寻 (与 / 或 ? 有关！)

N ：反向的重复前一个搜寻 (与 / 或 ? 有关！)

q ：离开 less 这个程序；

#### head

取出前面几行

#### tail

取出后面几行

tail -f 10 file 一般用于动态查看日志，显示最后10行

-f: 动态跟踪文件file的增长情况,tail会每隔1秒检查下文件是否增加新内容

-n:输出几行

### 文本处理

#### sed 文本替换

常用：

sed 's/text/replace\_text/' file //替换每一行的第一处匹配的text

sed 's/text/replace\_text/g' file // 全局替换

默认替换后，输出替换后的内容，如果需要直接替换原文件,使用-i:

sed -i 's/text/repalce\_text/g' file

sed -i '.bak' 's/text/repalce\_text/g' file //给源文件添加备份 .bak是备份后缀名

在linux下，不加备份后缀会直接覆盖原文件，但mac下必须加空字符串 ''才能实现不备份直接辅导原文件

sed '/^$/d' file //移除空白行

详细：

sed [options] 'command' file(s) //利用script处理文本文件

options:

-e:用指定的scripit处理文本文件(一般用于多点编辑)

-i: 直接替换源文件

command中的参数：

a\ ：在当前行下面插入文本

i\ ：在当前行上面插入文本

c\ : 把选定的行改为新的文本

d : 删除选择的行

p ：打印，亦即将某个选择的数据印出。通常 p 会与参数 sed -n 一起运行～

s ：取代，可以直接进行取代的工作！通常这个 s 的动作可以搭配正则表示法！例如 ：s/old/new/g 就是啦！

#### grep

搜索文件的内容，而不是搜索文件名

|  |
| --- |
| usage: grep [-abcDEFGHhIiJLlmnOoqRSsUVvwxZ] [-A num] [-B num] [-C[num]]  [-e pattern] [-f file] [--binary-files=value] [--color=when]  [--context[=num]] [--directories=action] [--label] [--line-buffered]  [--null] [pattern] [file ...] |

注意：最后一个参数是文件或文件的集合，不能是文件夹！！

参数：

-e: 指定字符串做为查找文件内容的样式(可指定多个样式)

grep -e 'hello' -e 'world' test.txt

-o: 只输出匹配的文本行

-v: 只输出没有匹配的文本行

-c: 统计文件中包含文本的行数

-n: 打印匹配的行号

-i: 搜索时忽略大小写

-l: 只打印文件名（一般搜索范围会是多个文件）

-r: 递归查询

grep "class" . -r -n //在多级目录递归搜索(在当前文件夹下)

pattern正则表达式主要参数：

\： 忽略正则表达式中特殊字符的原有含义。

^：匹配正则表达式的开始行。

$: 匹配正则表达式的结束行。

\<：从匹配正则表达 式的行开始。

\>：到匹配正则表达式的行结束。

[ ]：单个字符，如[A]即A符合要求 。

[ - ]：范围，如[A-Z]，即A、B、C一直到Z都符合要求 。

.：所有的单个字符。

\* ：有字符，长度可以为0。

### 系统命令

top 实时显示 process 的动态

### shell脚本

#!/bin/bash 指明解释此脚本的shell程序

chmod +x ./test.sh #使脚本具有执行权限

./test.sh #执行脚本

注:一定要写成 ./test.sh ,而不是test.sh, 运行其它二进制的程序也一样，直接写 test.sh，linux 系统会去 PATH 里寻找有没有叫 test.sh 的，而只有 /bin, /sbin, /usr/bin，/usr/sbin 等在 PATH 里，你的当前目录通常不在 PATH 里，所以写成 test.sh 是会找不到命令的，要用 ./test.sh 告诉系统说，就在当前目录找。

#### shell变量

your\_name="runoob.com" 无$符号 ,定义时不加$

echo $your\_name 使用变量 ,使用时加$

或echo ${your\_name}

只读变量：readonly your\_name

只读变量不能再次给他赋值了

删除变量：unset your\_name

#### 字符串

单引号字符串的限制：

单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；

单引号字串中不能出现单引号（对单引号使用转义符后也不行）。

双引号的优点：

双引号里可以有变量

双引号里可以出现转义字符

拼接字符串：

your\_name="qinjx"

greeting="hello, "$your\_name" !"

greeting\_1="hello, ${your\_name} !"

echo $greeting $greeting\_1

获取字符串长度：

echo ${#your\_name}

提取子字符串：

echo ${your\_name:1:4} //从字符串第 **2** 个字符开始截取 **4** 个字符：

查找字符串：

echo `epr index “${your\_name}” y`

#### shell数组

数组名=(值1 值2 ... 值n)

读取数组元素值:

${数组名[下标]}

用@来获取数组里的所有元素：

echo ${array\_name[@]}

获取数组长度：

echo ${#array\_name[@]} 或 ${#array[\*]}

#### shell传递参数

#!/bin/bash

echo '文件名：'$0

echo '第一个参数：'$1

echo '第二个参数：'$2

echo '第三个参数：'$3

运行shell脚本时，./test.sh 1 2 3 会将三个参数传入脚本中

$0表示脚本文件名，$1为传入的第一个参数，以此类推

另外：





#### 运算符

#### 算数运算符

val=`expr 2 + 2` 要用expr，运算符两侧要有空格

常用的有+ - \* / % = == !=

乘法要在\*前加\

val=`expr $a \\* $b`

对于==和!=这样的条件表达式，要放在方括号内，且表达式与方括号之间要有空格！ if[‘2’==’’’2’]是错误的

If [ ‘2’==’2’ ]

then

…

fi

注意：

* 乘号(\*)前边必须加反斜杠(\)才能实现乘法运算；
* 在 MAC 中 shell 的 expr 语法是：$((表达式))，此处表达式中的 "\*" 不需要转义符号 "\" 。

#### 关系运算符

只支持数字，不支持字符串，除非字符串的值是数字

[ $a -eq $b ] 等于

[ $a -ne $b ] 不等于

[ $a -gt $b ] 大于

[ $a -lt $b ] 小于

[ $a -ge $b ] 大于等于

[ $a -le $b ] 小于等于

#### 布尔运算符

! 非运算

-o 或运算

-a 与运算

#### 逻辑运算

&& 逻辑and

|| 逻辑or

#### echo命令

* 原样输出 用单引号
* 输出变量的值 用双引号
* 显示命令执行结果

echo `date` (反引号)

Thu Jul 24 10:08:46 CST 2014

* 换行

echo -e "OK! \n" # -e 开启转义

* 不换行

echo -e "OK! \c" # -e 开启转义 \c 不换行（下面输出会与OK在同一行）

* 将结果输出到文件

echo "It is a test" > myfile

#### 输入/输出重定向

|  |  |
| --- | --- |
| **命令** | **说明** |
| command > file | 将输出重定向到 file。 |
| command < file | 将输入重定向到 file。 |
| command >> file | 将输出以追加的方式重定向到 file。 |
| n > file | 将文件描述符为 n 的文件重定向到 file。 |
| n >> file | 将文件描述符为 n 的文件以追加的方式重定向到 file。 |
| n >& m | 将输出文件 m 和 n 合并。 |
| n <& m | 将输入文件 m 和 n 合并。 |
| << tag | 将开始标记 tag 和结束标记 tag 之间的内容作为输入。 |

一般情况下，每个 Unix/Linux 命令运行时都会打开三个文件：

* 标准输入文件(stdin)：stdin的文件描述符为0，Unix程序默认从stdin读取数据。
* 标准输出文件(stdout)：stdout 的文件描述符为1，Unix程序默认向stdout输出数据。
* 标准错误文件(stderr)：stderr的文件描述符为2，Unix程序会向stderr流中写入错误信息。

so：

* 如果希望 stderr 重定向到 file，可以这样写：

command 2 >> file

* 如果希望将 stdout 和 stderr 合并后重定向到 file，可以这样写：

command >> file 2>&1

从左到右是有顺序的：

先将要输出到stdout的内容重定向到文件，此时文件file就是这个程序的stdout，再将stderr重定向到stdout，也就是文件file

* 若不希望在屏幕显示结果，可以将输出重定向到 /dev/null

command > /dev/null

注：/dev/null 是一个特殊的文件，写入到它的内容都会被丢弃；如果尝试从该文件读取内容，那么什么也读不到。但是 /dev/null 文件非常有用，将命令的输出重定向到它，会起到"禁止输出"的效果。

如果希望屏蔽 stdout 和 stderr，可以这样写：

command > /dev/null 2>&1

关于Curl的输出：若要屏蔽掉执行时间等参数，可以加-s(静默模式)参数即可

### 通配符

#### ?字符

?字符代表单个字符

|  |
| --- |
| # 存在文件 a.txt 和 b.txt  $ ls ?.txt  a.txt b.txt |

？不能匹配空字符

#### \*字符

\*代表任意数量的字符

|  |
| --- |
| # 存在文件 a.txt、b.txt 和 ab.txt  $ ls \*.txt  a.txt b.txt ab.txt |

\*可以匹配空字符

#### […]模式

[...]匹配方括号之中的任意一个字符

[start-end]表示一个连续的范围

|  |
| --- |
| # 存在文件 a.txt 和 b.txt  $ ls [ab].txt  a.txt b.txt  # 存在文件 a.txt、b.txt 和 c.txt  $ ls [a-c].txt  a.txt b.txt c.txt |

### awk

#### 基本用法

|  |
| --- |
| # 格式  $ awk 动作 文件名  # 示例  $ awk '{print $0}' demo.txt |

花括号里是demo.txt文件里**每一行**的处理动作；

$0: 当前行

awk会根据空格和制表符，将每一行分成若干字段，依次用$1、$2、$3代表第一个字段、第二个字段、第三个字段…

**-F** 指定分隔符，按分隔符将每一行分割开来

#### 变量

NF：当前行有多少个字段（字段相当于单词）

$NF：代表最后一个字段

$(NF-1)：代表倒数第二个字段

NR：表示当前处理的是第几行

其他：

|  |
| --- |
| FILENAME：当前文件名  FS：字段分隔符，默认是空格和制表符。  RS：行分隔符，用于分割每一行，默认是换行符。  OFS：输出字段的分隔符，用于打印时分隔字段，默认为空格。  ORS：输出记录的分隔符，用于打印时分隔记录，默认为换行符。 |

print命令里面，如果原样输出字符，要放在双引号里面

#### 函数

|  |
| --- |
| tolower()：字符转为小写。  length()：返回字符串长度。  substr()：返回子字符串。  sin()：正弦。  cos()：余弦。  sqrt()：平方根。  rand()：随机数。 |

#### 条件

|  |
| --- |
| awk '条件 动作' 文件名 |

例如：

|  |
| --- |
| awk -F ':' '/usr/ {print $1}' demo.txt |

print命令前面是一个正则表达式，只输出包含usr的行。

#### if

|  |
| --- |
| awk -F ':' '**{**if ($1 > "m") print $1**}**' demo.txt |

if语句要放到花括号里

if…else

|  |
| --- |
| awk -F ':' '{if ($1 > "m") print $1; else print "---"}' demo.txt |