

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



北京化工大学
BEIJING UNIVERSITY OF CHEMICAL TECHNOLOGY

The Linux Command Line

常用Linux命令

王永红

0x12oot@gmail.com

May 6, 2017



ls命令

ls 命令是 Linux 下最常用的命令。**ls** 命令就是 **list** 的缩写，**ls** 用来打印出当前目录的清单，如果 **ls** 指定其他目录，那么就会显示指定目录里的文件及文件夹清单。通过 **ls** 命令不仅可以查看 Linux 文件夹包含的文件，而且可以查看文件权限（包括目录、文件夹、文件权限），查看目录信息等等。

参数：

- a 列出目录下的所有文件，包括以 . 开头的隐含文件
 - l 详细列出文件的权限、所有者、文件大小等信息
 - h **-human-readable** 以容易理解的格式列出文件大小
(例如 1K 234M 2G)
 - i -inode 印出每个文件的 inode 号
 - t 以文件修改时间排序
 - c 配合 -lt：根据 **ctime** 排序及显示 **ctime**
配合 -l：显示 **ctime** 但根据名称排序
否则：根据 **ctime** 排序
 - R -recursive 同时列出所有子目录层
- 注：**ctime**: 文件状态最后更改的时间

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



北京化工大学
BEIJING UNIVERSITY OF CHEMICAL TECHNOLOGY

ls命令（扩展）

显示彩色目录列表

打开 /etc/.bashrc，加入如下一行：

```
alias ls="ls --color"
```

下次启动 bash 时就可以显示彩色的目录列表了，其中颜色的含义如下：

1. 蓝色 --> 目录
2. 绿色 --> 可执行文件
3. 红色 --> 压缩文件
4. 浅蓝色 --> 链接文件
5. 灰色 --> 其他文件



cd命令

Linux cd 命令可以说是 Linux 中最基本的命令语句，其他的命令语句要进行操作，都是建立在使用 cd 命令上的。

命令格式：

cd [DirName]

命令功能：

切换当前目录至 DirName

举例：

cd /	切换到根目录
cd ~	切换到用户目录
cd ..	切换到上一级目录
cd .	不变
cd ./	不变
cd ../../	切换到上一级目录
cd ../../..	切换到上两级目录
cd -	切换到之前的目录
cd /etc/opt	切换到/etc/opt 目录



pwd 命令

Linux 中用 `pwd` 命令来查看“当前工作目录”的完整路径。简单地说，每当你在终端进行操作时，你都会有一个当前工作目录。在不太确定当前位置时，就会使用 `pwd` 来判定当前目录在文件系统内的确切位置。

命令格式：

`pwd [选项]`

命令功能：

查看“当前工作目录”的完整路径

举例：

`pwd` 默认工作目录的完整路径

`pwd -P` 显示实际路径，而非使用连接（link）路径

`pwd -L` 目录连接链接时，输出连接路径

注：当前目录被删除了，而 `pwd` 命令仍然显示那个目录



mkdir 命令

Linux 中用 `mkdir` 命令用来创建指定的名称的目录，要求创建目录的用户在当前目录中具有写权限，并且指定的目录名不能是当前目录中已有的目录。

命令格式：

```
mkdir [选项] [DirName]
```

举例：

`mkdir test` 在当前目录下创建一个test文件夹

`mkdir -m 777 test` 在当前目录下创建一个权限777的文件夹

`mkdir -p test1/test2` 递归创建文件夹

`mkdir -v test` 文件夹创建成功后有提示

`mkdir -vp test1/test2` 输出信息为：

```
mkdir: created directory 'test1'
```

```
mkdir: created directory 'test1/test2'
```



rm命令

rm命令功能为删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录均删除。对于链接文件，只是删除了链接，原有文件均保持不变。

命令格式：

```
rm [选项] [DirName/FileName]
```

举例：

rm test.txt 删除当前目录下的test.txt文件（询问是否删除）

rm -f test.txt 强行删除test.txt文件，系统不再提示

rm -i *.log 删除任何.log文件，并一一询问

rm -r test 删除test文件夹下面所有文件（包括子目录）

rm -rf test 删除test文件夹下面的所有文件且系统不提示



rmdir命令

`rmdir` 命令从一个目录中删除一个或多个子目录项，删除某目录时也必须具有对父目录的写权限。

命令格式：

```
rmdir [选项] [DirName]
```

参数：

`-p` 递归删除目录`DirName`，当子目录删除后其父目录为空时，也一同被删除。如果整个路径被删除或者由于某种原因保留部分路径，则系统在标准输出上显示相应的信息。

`-v` 显示指令执行过程

举例：

```
rm -vp test/test1
rmdir: removing directory, 'test/test1'
rmdir: removing directory, 'test'
```



mv命令

`mv` 命令是 `move` 的缩写，可以用来移动文件或者将文件改名
(`move (rename) files`)

命令格式：

`mv [选项] [源文件或目录] [目标文件或目录]`

参数：

- b 需覆盖文件，则覆盖前先进行备份，四种策略(如`a.c-->a.c~`)
- f 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，才会更新(`update`)

举例：（作用相同的两条命令）

`mv log1.txt log2.txt log3.txt test`

`mv -t ./test log1.txt log2.txt log3.txt`

注：在跨文件系统移动文件时，`mv`先拷贝，再将原有文件删除，而链至该文件的链接也将丢失。



cp命令

cp命令复制文件或者目录，一般cp是cp -i的别名

命令格式：

```
cp [选项] [源文件或目录] [目标文件或目录]
```

参数：

- b 需覆盖文件，则覆盖前先进行备份，四种策略(如a.c-->a.c~)
- f 如果目标文件无法打开则将其移除并重试
- i 若目标文件 (destination) 已经存在时，就会询问是否覆盖
- n 不覆盖已存在的文件
- r 复制目录及目录内所有项目

举例：

```
cp -s log.log log_link.log
```

复制的 log.log 建立一个连结档 log_link.log



touch命令

touch 命令不常用，一般在使用**make**的时候可能会用到，用来修改文件时间戳，或者新建一个不存在的文件。

命令格式：

touch [选项] [FileName]

举例：

touch log2012.log log2013.log

创建不存在的文件

touch -r log.log log2.log

log2.log时间变为与
log.log一样

touch -t 201211142234.50 log.log

设定指定时间



touch命令（扩展）

-t time 使用指定的时间值 **time** 作为指定文件相应时间戳记的新值。此处的 **time** 规定为如下形式的十进制数：

[[CC]YY]MMDDhhmm[.SS]

这里，**CC** 为年数中的前两位，即“世纪数”；**YY** 为年数的后两位，即某世纪中的年数。如果不给出**CC** 的值，则**touch** 将把年数**CCYY** 限定在**1969--2068** 之内。**MM** 为月数，**DD** 为天将把年数**CCYY** 限定在**1969-2068** 之内。**MM** 为月数，**DD** 为天数，**hh** 为小时数(几点)，**mm** 为分钟数，**ss** 为秒数。此处秒的设定范围是**0--61**，这样可以处理闰秒。这些数字组成的时间是环境变量**TZ** 指定的时区中的一个时间。由于系统的限制，早于**1970** 年**1** 月**1** 日的时间是错误的。



cat命令

cat 命令的用途是连接文件或标准输入并打印。这个命令常用来显示文件内容，或者将几个文件连接起来显示，或者从标准输入读取内容并显示，它常与重定向符号配合使用。

举例：

cat filename 查看文件内容

cat > filename 重新编辑文件内容

cat >> filename 在文件后面继续添加内容

cat -b log1.log log2.log > log.log

把log1.log和log2.log加上行号写入log.log文件

tac命令： tac 是将 cat 反写过来，所以他的功能就跟 cat 相反， cat 是由第一行到最后一行连续显示在屏幕上，而 tac 则是由最后一行到第一行反向在屏幕上显示出来！



n1命令

n1 命令在Linux系统中用来计算文件中行号。**n1** 可以将输出的文件内容自动的加上行号！其默认的结果与 **cat -n** 有点不太一样，**n1** 可以将行号做比较多的显示设计，包括位数与是否自动补齐0等等的功能。

参数：

- b 指定行号指定的方式，主要有两种：
 - b a 表示不论是否为空行，也同样列出行号(类似 **cat -n**)；
 - b t 如果有空行，空的那一行不要列出行号(默认值)；
- n 列出行号表示的方法，主要有三种：
 - n ln 行号在萤幕的最左方显示；
 - n rn 行号在自己栏位的最右方显示，且不加0；
 - n rz 行号在自己栏位的最右方显示，且加0；
- w 行号栏位的占用的位数。
- p 在逻辑定界符处不重新开始计算。



more命令

more命令，类似 cat，cat命令是整个文件的内容显示在屏幕上。more会一页一页的显示方便使用者逐页阅读，基本指令按空白键（Space）往下一页显示，按 b 键往回一页显示，而且还有搜寻字串的功能。

参数：

- +n 从第n行开始显示
- n 定义每屏显示n行
- +/pattern 在文本中搜索该字符串(pattern)，然后从前两行开始显示
- p 通过清除窗口而不是滚屏来对文件进行换页

常用操作命令：

- Enter 向下n行，需要定义，默认为1行
- Ctrl+F/f/Space 向下滚动一屏
- Ctrl+B/b 向上滚动一屏
- q 退出more



less命令

less命令，类似**more**，比**more**更强大，可以使用[PageUp]和[PageDown]来前后翻看文件，**less**的搜索功能也更加强大。**More**命令查看前会加载整个文件，**less**查看前不会加载整个文件。

参数：

- b <缓冲区大小> 设置缓冲区的大小
- i 忽略搜索时的大小写
- m 显示类似**more**命令的百分比
- N 显示每行的行号
- o <文件名> 将**less**输出的内容在指定文件中保存起来
- x <数字> 将“Tab”键显示为规定的数字空格
- /字符串 向下搜索“字符串”的功能
- ?字符串 向上搜索“字符串”的功能



less命令续

常用操作命令：

n	重复前一个搜索（与 / 或 ? 有关）
N	反向重复前一个搜索（与 / 或 ? 有关）
Ctrl+F/f	向前滚动一屏
Ctrl+B/b	向后滚动一屏
Ctrl+D/d	向前滚动半屏
Ctrl+U/u	向后滚动半屏
j	向前滚动一行
k	向后滚动一行
G	移动到最后一行
g	移动到第一行
q/ZZ	退出less命令
v	使用配置的编辑器编辑文件
h	显示帮助文档
&pattern	仅显示匹配模式的行，而不是整个文件



head命令

head命令，用来显示开头或结尾某个数量的文字区块，**head** 用来显示档案的开头至标准输出中。

参数：

-q 隐藏文件名

-v 显示文件名

-c <字节> 显示字节数

-n <行数> 显示的行数

举例：

head -n 5 log.log 显示文件的前5行

head -c 5 log.log 显示文件的前5个字节

head -n -5 log.log 显示文件除了最后5行的内容

head -c -5 log.log 显示文件除了最后5个字节的内容



tail命令

tail命令从指定点开始将文件写到标准输出，使用**tail**命令的-f选项可以方便的查阅正在改变的日志文件，不断刷新。

参数：

- f 循环读取
- q 不显示处理信息
- v 显示详细的处理信息
- c <字节> 显示字节数
- n <行数> 显示的行数

举例：

- tail -n 5 log.log** 显示文件末尾5行
- tail -f test.log** 循环查看文件内容
- tail -n +5 log.log** 从第5行开始显示文件

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



北京化工大学
BEIJING UNIVERSITY OF CHEMICAL TECHNOLOGY

拓展(以-开头的文件)

以-开头的文件在创建或者删除时要加上--选项，例如：

```
touch -- -test.c
```

```
rm -- -test.c
```

```
cat -- -test.c
```



which命令

which命令的作用是，在PATH变量指定的路径中，搜索某个系统命令的位置，并且返回第一个搜索结果。也就是说，使用which命令，就可以看到某个系统命令是否存在，以及执行的到底是哪一个位置的命令。

命令格式：

`which 可执行文件名称`

命令参数：

`-n` 指定文件名长度，指定的长度必须大于或等于所有文件中最长的文件名。

`-p` 与-n参数相同，但此处的包括了文件的路径。

`-w` 指定输出时栏位的宽度。

`-V` 显示版本信息



whereis命令

whereis命令只能用于程序名的搜索，而且只搜索二进制文件（参数-b）、**man**说明文件（参数-m）和源代码文件（参数-s）。如果省略参数，则返回所有信息。

和**find**相比，**whereis**查找的速度非常快，这是因为**Linux**系统会将系统内的所有文件都记录在一个数据库文件中，当使用**whereis**和下面即将介绍的**locate**时，会从数据库中查找数据，而不是像**find**命令那样，通过遍历硬盘来查找，效率自然会很高。

但是该数据库文件并不是实时更新，默认情况下时一星期更新一次，因此，我们在用**whereis**和**locate**查找文件时，有时会找到已经被删除的数据，或者刚刚建立文件，却无法查找到，原因就是因为数据库文件没有被更新。



whereis命令续

参数：

- b 定位可执行文件。
- m 定位帮助文件。
- s 定位源代码文件。
- u 搜索默认路径下除可执行文件、源代码文件、帮助文件以外的其它文件。
- B 指定搜索可执行文件的路径。
- M 指定搜索帮助文件的路径。
- S 指定搜索源代码文件的路径。

举例：

`whereis mysql`

输出：`mysql: /usr/bin/mysql /usr/lib/mysql
/usr/include/mysql /usr/local/mysql`



locate命令

locate可以在搜寻数据库时快速找到档案，数据库由**updatedb**程序来更新，**updatedb**是由**cron daemon**周期性建立的，**locate**命令在搜寻数据库时比由整个由硬盘资料来搜寻资料来得快，但较差劲的是**locate**所找到的档案若是最近才建立或刚更名的，可能会找不到，在内定值中，**updatedb**每天会跑一次，可以由修改/etc/crontab来更新设定值。

locate指定用在搜寻符合条件的档案，它会去储存档案与目录名称的数据库内，寻找合乎范本样式条件的档案或目录录，可以使用特殊字元（如”*”或”?”等）来指定范本样式，如指定范本为kcpa*ner, **locate**会找出所有起始字串为kcpa且结尾为ner的档案或目录，如名称为kcpartner若目录录名称为kcpa_ner则会列出该目录下包括子目录在内的所有档案。

locate指令和**find**找寻档案的功能类似，但**locate**是透过**update**程序将硬盘中的所有档案和目录资料先建立一个索引数据库，在执行**locate**时直接找该索引，查询速度会较快，索引数据库一般是由操作系统管理，但也可以直接下达**update**强迫系统立即修改索引数据库。



locate命令续

参数：

- e 将排除在寻找的范围之外。
- f 将特定的档案系统排除在外，例如我们没有道理要把 proc 档案系统中的档案放在资料库中。
- q 安静模式，不会显示任何错误讯息。
- n 至多显示 n 个输出。
- r 使用正规运算式做寻找的条件。
- o 指定资料库存的名称。
- d 指定资料库的路径
- h 显示辅助讯息
- V 显示程式的版本讯息



find命令

find命令

命令格式：

```
find pathname -options [-print -exec -ok ...]
```

命令参数：

pathname: find命令所查找的目录路径。例如用.来表示当前目录，用/来表示系统根目录。

-print: find命令将匹配的文件输出到标准输出。

-exec: find命令对匹配的文件执行该参数所给出的shell命令。相应命令的形式为'command' {} \;，注意{}和\;之间的空格。

-ok: 和-exec的作用相同，只不过以一种更为安全的模式来执行该参数所给出的shell命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。



find命令示例

`find -name name`

查找文件名为name的文件

`find . -name “*.txt” -print`

在当前目录下查找txt文件并显示

`find ~ -name “*.xml” -print`

在\$HOME目录下查找xml文件并显示

`find -name “[A-Z]*” -print`

查找以大写字母开头的文件

`find -name “P*” -print`

查找以大写字母P开头的文件

`find -name “*s” -print`

查找以小写字母s结尾的文件

`find -name “[a-z]*” -print`

查找以小写字母开头的文件

`find . -perm 755 -print`

查找权限755的文件

`find / -user honghong`

按照文件属主来查找文件

`find / -group hong`

按照文件属组来查找文件

`find /home -nouser`

列出/home内不属于本地用户的文件

`find /home -nogroup`

列出/home内不属于本地组的文件



find命令示例

`find -type d -print`

查找某一类型的文件

(b 块设备文件、d 目录、c 字符设备文件、p 管道文件、l 链接字符文件、f 普通文件)

`find . size 1000c -print`

查找文件长度为n块（1块=512字节）

的文件，带有c时表示文件以字节计

`find . size +1000c -print`

查找文件大于1000c的文件

`find . size +10 -print`

查找文件长度超过10块的文件

`find . -size -5557c -size +5555c -print`

查找文件大小为5556字节的文件

注：

`find -type f -print`

每个输出后添加一个换行符

`find -type f -print0`

每个输出后不会有换行符



find命令示例

按照时间查找文件

- amin n 查找系统中最后N分钟访问的文件
- atime n 查找系统中最后n*24小时访问的文件
- cmin n 查找系统中最后N分钟被改变文件状态的文件
- ctime n 查找系统中最后n*24小时被改变文件状态的文件
- mmin n 查找系统中最后N分钟被改变文件数据的文件
- mtime n 查找系统中最后n*24小时被改变文件数据的文件

find -atime -2 查找超过48小时修改过的文件



find命令之exec

```
find . -type f -exec ls -l {} \;
```

上面的例子中，`find`命令匹配到了当前目录下的所有普通文件，并在`-exec`选项中使用`ls -l`命令将它们列出。

```
find . -type f -mtime +14 -exec rm {} \;
```

在目录中查找更改时间在n日以前的文件并删除它们

```
find . -name “*.log” -mtime +5 -ok rm {} \;
```

在当前目录中查找所有文件名以.log结尾、更改时间在5日以上的文件，并删除它们，只不过在删除之前先给出提示。按y键删除文件，按n键不删除

```
find /etc -name “passwd*” -exec grep “root” {} \;
```

在上面的例子中我们使用`grep`命令。`find`命令首先匹配所有文件名为“`passwd*`”的文件，例如`passwd`、`passwd.old`、`passwd.bak`，然后执行`grep`命令看看在这些文件中是否存在一个`root`用户

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



find命令之exec续

```
find . -name “*.log” -exec mv {} .. \;
```

查找文件移动到指定目录

```
find . -name “*.log” -exec cp {} test3 \;
```

执行cp命令



find命令之xargs

find命令把匹配到的文件传递给xargs命令，而xargs命令每次只获取一部分文件而不是全部，不像-exec选项那样。这样它可以先处理最先获取的一部分文件，然后是下一批，并如此继续下去。避免出现溢出错误。

```
find . -type f -print | xargs file
```

查找系统中的每一个普通文件，然后使用xargs命令来测试它们分别属于哪类文件

```
find / -name "core" -print | xargs echo "" >/tmp/core.log
```

在整个系统中查找内存信息转储文件(core dump)，然后把结果保存到 /tmp/core.log 文件中

```
find . -perm -7 -print | xargs chmod o-w
```

在当前目录下查找所有用户具有读、写和执行权限的文件，并收回相应的写权限



find命令之xargs续

```
find . -type f -print | xargs grep "hostname"
```

用grep命令在所有的普通文件中搜索hostname这个词

```
find . -name \* -type f -print | xargs grep "hostnames"
```

用grep命令在当前目录下的所有普通文件中搜索hostnames这个词

```
find . -name "*.log" | xargs -i mv {} test4
```

使用xargs执行mv

```
find . -type f -atime +0 -print0 | xargs -0 -l1 -t rm -f
```

find后执行xargs提示xargs: argument line too long解决方法

```
find . -name "*.log" | xargs -p -i mv {} ..
```

-p参数会提示让你确认是否执行后面的命令,y执行, n不执行。

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



北京化工大学
BEIJING UNIVERSITY OF CHEMICAL TECHNOLOGY

Linux文件和目录属性

Linux目录结构

Linux文件类型和拓展名

Linux文件属性详解



scp命令

scp是secure copy的简写，是Linux系统下基于ssh登录进行安全的远程文件拷贝命令。scp传输是加密的，可能会稍稍影响传输速度。

命令格式：

```
scp [参数][原路径][目标路径]
```

参数：

- q 不显示传输进度条
- r 递归复制整个目录
- v 详细方式显示输出
- c 选择加密方式
 - 加密方式 (3des-cbc、aes256-ctr、aes192-ctr、aes128-ctr、cast128-cbc、blowfish-cbc、arcfour、arcfour128、arcfour128、arcfour256、aes256-cbc、aes192-cbc、aes128-cbc)
- C 传输是允许压缩，从而加快传输速
- l (limit)限定用户所能使用的带宽，以Kbit/s为单位
- P Port指定ssh端口号
- p 保留文件的修改时间等信息

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



scp命令续

举例：

下载文件 `scp root@192.168.120.204:/opt/soft/nginx-0.5.38.tar.gz /opt/soft/`

下载文件夹 `scp -r root@192.168.120.204:/opt/soft/mongodb /opt/soft/`

上传文件 `scp /opt/soft/nginx-0.5.38.tar.gz root@192.168.120.204:/opt/soft/scptest`

上传文件夹 `scp -r /opt/soft/mongodb root@192.168.120.204:/opt/soft/scptest`



wget命令

wget命令用于从网络上下载资源，没有指定目录，下载资源回默认为当前目录。wget支持断点下传功能，支持FTP、HTTP、HTTPS下载，支持代理服务器，程序小，完全免费。

命令格式：

```
wget [参数][URL地址]
```

参数：

- b 启动后转入后台运行
- d 打印调试输出
- q 安静模式，没有输出
- v 输出详细信息，默认
- c 接着下载没下载完的文件



wget命令续

举例：

```
wget http://www.minjieren.com/wordpress-3.1-zh_CN.zip  
wget -O wordpress.zip http://www.minjieren.com/download.a  
spx?id=1080
```

wget默认会以最后一个符合”/”的后面的字符来命令，对于动态链接的
下载通常文件名会不正确。

错误：下面的例子会下载一个文件并以名称download.aspx?id=1080保
存

```
wget http://www.minjieren.com/download.aspx?id=1080
```

即使下载的文件是zip格式，它仍然以download.php?id=1080命令。

正确：为了解决这个问题，我们可以使用参数-O来指定一个文件名：

```
wget -O wordpress.zip http://www.minjieren.com/download.a  
spx?id=1080
```



tar命令

tar命令用来压缩和解压文件。**tar**本身不具有压缩功能。他是调用压缩功能实现的。

命令格式：

tar [必要参数][选择参数][文件]

举例：

.tar

tar cvf test.tar test

把test文件夹打包成**test.tar**

tar xvf test.tar

解包

.tar.gz

tar zcvf test.tar.gz test

tar zxvf test.tar.gz

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



北京化工大学
BEIJING UNIVERSITY OF CHEMICAL TECHNOLOGY

tar命令续

.tar.bz2

```
tar jcvf test.tar.bz2 test
```

```
tar jxvf test.tar.bz2
```

.Z

```
tar Zcvf test.tar.Z test
```

```
tar Zxvf test.tar.Z
```



rar命令

rar命令用来压缩文件。unrar用来解压文件。

安装：

```
wget http://www.rarlab.com/rar/rarlinux-x64-5.5.b2.tar.gz
```

```
tar zxvf rarlinux-x64-5.5.b2.tar.gz
```

```
cd rar
```

```
sudo install -c -o $USER rar /bin
```

```
sudo install -c -o $USER unrar /bin
```

压缩与解压：

```
rar a test.rar test
```

```
unrar x test.rar
```



chmod命令

chmod用于改变文件或目录的访问权限，用它控制文件或目录的访问权限。

文字设定法：

chmod [who] [+ | - | =] [mode] 文件名

chmod a+x log2012.log 增加文件所有用户组可执行权限

chmod ug+w,o-x log2012.log 同时修改不同用户权限

chmod a-x log2012.log 删 除文件权限

chmod u=x log2012.log 使用“=”设置权限

chmod -R u+x test4 对一个目录及其子目录所有文件添加权限



chmod命令续

数字设定法：

数字与字符对应关系如下：

r=4, w=2, x=1

若要rwx属性则 $4+2+1=7$ ，若要rw-属性则 $4+2=6$ ，若要r-x属性则 $4+1=5$

`chmod 751 file` 给file的属主分配读写执行(7)的权限，给file的所在组分配读执行(5)的权限，给其他用户分配执行(1)的权限

`chmod u=rwx,g=rx,o=x file` 等同于上例

下面三个都是为file赋予所有用户读的权限

`chmod 444 file`

`chmod a-wx,a+r file`

`chmod a=r file`



df命令

df命令显示指定磁盘文件的可用空间。如果没有文件名被指定，则所有当前被挂载的文件系统的可用空间将被显示。

命令参数：

- h 方便阅读方式显示
- H 等于“-h”，但是计算式，1K=1000，而不是1K=1024
- i 显示inode信息
- T 文件系统类型

举例：

```
root@tencent:~# df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/vda1	20G	3.3G	16G	18%	/



du命令

du命令显示每个文件和目录的磁盘使用空间。

参数：

- a 显示目录中个别文件的大小。
- b 显示目录或文件大小时，以byte为单位
- k 以KB(1024bytes)为单位输出。
- m 以MB为单位输出。
- s 仅显示总计，只列出最后加总的值。
- h 以K, M, G为单位，提高信息的可读性。

举例：

- du -h test 以便于阅读的方式查看
- du -ah test 文件和目录都显示
- du | sort -nr | more 按照空间大小进行排序



top命令

top命令详解

```
WYH — ssh root@testest.win — root@testest.win — ssh root@testest.win — 80x24

top - 16:00:02 up 1 day, 18:17, 2 users, load average: 0.00, 0.01, 0.05
Tasks: 75 total, 2 running, 73 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 1.3 sy, 0.0 ni, 98.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1024436 total, 543192 used, 481244 free, 57528 buffers
KiB Swap: 0 total, 0 used, 0 free. 311648 cached Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
7069 root 20 0 31544 13624 1220 R 0.7 1.3 7:58.54 sap1005
1 root 20 0 28564 4128 2540 S 0.0 0.4 0:02.30 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 20 0 0 0 0 S 0.0 0.0 0:04.89 ksoftirqd/0
5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H
6 root 20 0 0 0 0 S 0.0 0.0 0:01.48 kworker/u2:0
7 root 20 0 0 0 0 S 0.0 0.0 1:00.67 rcu_sched
8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
9 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
10 root rt 0 0 0 0 S 0.0 0.0 0:00.76 watchdog/0
```



vmstat命令

vmstat命令用来显示虚拟内存的信息

```
[root@tencent:~# vmstat
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
 r b    swpd   free   buff   cache   si    so    bi    bo    in    cs us sy id wa st
 0 0      0 483404  57200 311516     0     0     5     67   114   216  1  0 98  0  0
root@tencent:~#]
```

字段说明：

Procs (进程) : r: 运行队列中进程数量、b: 等待IO的进程数量

Memory (内存) :

swpd: 使用虚拟内存大小、free: 可用内存大小

buff: 用作缓冲的内存大小、cache: 用作缓存的内存大小

Swap: si: 每秒从交换区写到内存的大小

so: 每秒写入交换区的内存大小、IO: (现在Linux版本块的大小为1024bytes)、bi: 每秒读取的块数、bo: 每秒写入的块数



vmstat命令

系统：

in：每秒中断数，包括时钟中断。

cs：每秒上下文切换数。

CPU（以百分比表示）：

us：用户进程执行时间(user time)

sy：系统进程执行时间(system time)

id：空闲时间(包括IO等待时间)，中央处理器的空闲时间

wa：等待IO时间

注：如果r经常大于4，且id经常少于40，表示CPU的负荷很重。如果pi，po长期不等于0，表示内存不足。如果disk经常不等于0，且在b中的队列大于3，表示io性能不好。



iostat命令

iostat命令方便查看CPU、网卡、tty设备、磁盘、CD-ROM 等等设备的活动情况，负载信息。

```
WYH — ssh root@testest.win — root@testest.win — ssh root@testest.win — 83x24
root@tencent:~# iostat
Linux 3.16.0-4-amd64 (tencent) 05/06/17      _x86_64_      (1 CPU)

avg-cpu: %user  %nice  %system  %iowait  %steal  %idle
        1.08    0.00    0.43    0.16    0.00   98.33

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
vda                 12.19       5.81        66.48    892277   10215328

root@tencent:~#
```



iostat命令续

CPU属性值说明：

%user：CPU处在用户模式下的时间百分比。

%nice：CPU处在带NICE值的用户模式下的时间百分比。

%system：CPU处在系统模式下的时间百分比。

%iowait：CPU等待输入输出完成时间的百分比。

%steal：管理程序维护另一个虚拟处理器时，虚拟CPU的无意识等待时间百分比。

%idle：CPU空闲时间百分比。

备注：如果%iowait的值过高，表示硬盘存在I/O瓶颈，%idle值高，表示CPU较空闲，如果%idle值高但系统响应慢时，有可能是CPU等待分配内存，此时应加大内存容量。%idle值如果持续低于10，那么系统的CPU处理能力相对较低，表明系统中最需要解决的资源是CPU。



iostat命令续

disk属性值说明：

rrqm/s： 每秒进行 merge 的读操作数目。即 rmerge/s

wrqm/s： 每秒进行 merge 的写操作数目。即 wmerge/s

r/s： 每秒完成的读 I/O 设备次数。即 rio/s

w/s： 每秒完成的写 I/O 设备次数。即 wio/s

rsec/s： 每秒读扇区数。即 rsect/s

wsec/s： 每秒写扇区数。即 wsect/s

rkB/s： 每秒读K字节数。是 rsect/s 的一半，因为每扇区大小为512字节。

wkB/s： 每秒写K字节数。是 wsect/s 的一半。

avgrq-sz： 平均每次设备I/O操作的数据大小（扇区）。

avgqu-sz： 平均I/O队列长度。



iostat命令续

await: 平均每次设备I/O操作的等待时间（毫秒）。

svctm: 平均每次设备I/O操作的服务时间（毫秒）。

%util: 一秒中有百分之多少的时间用于 I/O 操作，即被io消耗的cpu百分比

备注：如果 **%util** 接近 100%，说明产生的I/O请求太多，I/O系统已经满负荷，该磁盘可能存在瓶颈。如果 **svctm** 比较接近 **await**，说明 I/O 几乎没有等待时间；如果 **await** 远大于 **svctm**，说明I/O 队列太长，io响应太慢，则需要进行必要优化。如果**avgqu-sz**比较大，也表示有当量io在等待。



ifconfig命令

ifconfig 命令用来查看和配置网络设备。当网络环境发生改变时可通过此命令对网络进行相应的配置。Windows中对应的命令为**ipconfig**命令格式：

ifconfig [网络设备][参数]

举例：

ifconfig 显示网络设备信息（激活状态的）

ifconfig eth0 up 启动指定网卡

ifconfig eth0 down 关闭指定网卡

ifconfig eth0 192.168.120.56

ifconfig eth0 192.168.120.56 netmask 255.255.255.0

ifconfig eth0 192.168.120.56 netmask 255.255.255.0

broadcast 192.168.120.255



ifconfig命令续

`ifconfig eth0 add 33ffe:3240:800:1005::2/64`

为网卡eth0配置IPV6地址 (del为删除)

`ifconfig eth0 hw ether 00:AA:BB:CC:DD:EE`

修改MAC地址

`ifconfig eth0 arp` 启用ARP协议

`ifconfig eth0 -arp` 关闭ARP协议

`ifconfig eth0 mtu 1500`

设置最大传输单元

注：

`inet addr` 用来表示网卡的IP地址，此网卡的IP地址是`192.168.120.204`，广播地址，`Bcast:192.168.120.255`，掩码地址`Mask:255.255.255.0`



ping命令

ping命令用于确定网络和各外部主机的状态；跟踪和隔离硬件和软件问题；测试、评估和管理网络。如果主机正在运行并连在网上，它就对回送信号进行响应。

命令格式：

```
ping [参数][主机名或IP地址]
```

举例：

```
ping baidu.com
```

注：

Linux 下的 ping 和 Windows 下的 ping 稍有区别，Linux 下 ping 不会自动终止，需要按 Ctrl+C 终止或者用参数 -c 指定要求完成的回应次数。

文件目录操作
文件查找
文件和目录属性
文件打包和上传
文件权限设置

磁盘存储相关
性能监控和优化相关
网络命令
其他命令



北京化工大学
BEIJING UNIVERSITY OF CHEMICAL TECHNOLOGY

traceroute命令

traceroute命令用于追踪网络数据包的路由途径，预设数据包大小是40Bytes，用户可另行设置。

命令格式：

```
traceroute [参数][主机名或IP地址]
```

举例：

```
traceroute baidu.com
```

注：

Windows 下对应的命令为 **tracert**



netstat命令

netstat用于显示与IP、TCP、UDP和ICMP协议相关的统计数据，一般用于检验本机各端口的网络连接情况。

常用命令：

```
netstat -antp|grep 8080          查找占用8080端口的netstat -  
np|grep java|wc -l              查看java的并发数
```

查看80端口请求数最高的20个ip（查找攻击源）

```
netstat -anlp|grep 80|grep tcp|awk '{print $5}'|awk -F:  
'{print $1}'|sort|uniq -c|sort -nr|head -n20
```

查看tcp端口的状态

```
netstat -nat |awk '{print $6}'|sort|uniq -c|sort -rn
```



netstat命令续

参数：

- a show both listening and none-listening sockets
默认是不显示listening sockets
- t 仅显示TCP相关， 默认是都显示
- u 仅显示UDP相关， 默认是都显示
- n 拒绝显示别名， 显示数字
- l 仅列出有在Listen(监听)的服务状态
- p 显示建立相关连接的程序名， 需要sudo权限
- r 显示路由表
- c 每隔一段时间(秒)， 执行该netstat命令
- i 显示各个网络接口的状况
- s 按照协议进行统计



netstat命令续

TCP端口状态

1. **LISTENING** 对应netstat的LISTEN，我们开一个80端口的服务，也就是使80端口处于LISTEN状态，这样浏览器就可以与我们的80端口进行连接

2. **ESTABLISHED** 表示两个端口建立连接成功，正在通信

3. **CLOSE_WAIT** 对方主动关闭连接或者网络异常导致连接中断，这时我方的状态就会变为CLOSE_WAIT，此时我方要主动调用close()来关闭连接

4. **TIME_WAIT** 我方主动调用close()断开连接，收到对方确认后变为TIME_WAIT。TCP协议规定TIME_WAIT状态会一直持续2MSL(两倍的分段最大生存期)，以此确保旧的连接状态不会对新连接产生影响。处于TIME_WAIT状态的连接不会被内核释放，所以作为服务器，在可能的情况下，尽量不要主动断开连接，以减少TIME_WAIT状态造成的资源浪费。



ss命令

ss是**Socket Statistics**的缩写。顾名思义，**ss**命令可以用来获取**socket**统计信息，它可以显示和**netstat**类似的内容。但**ss**的优势在于它能够显示更多更详细的有关TCP和连接状态的信息，而且比**netstat**更快更高效。

参数：

- t --tcp 仅显示 TCP 套接字 (sockets)
- u --udp 仅显示 UCP 套接字 (sockets)
- d --dccp 仅显示 DCCP 套接字 (sockets)
- w --raw 仅显示 RAW 套接字 (sockets)
- x --unix 仅显示 Unix 套接字 (sockets)



ln命令

Linux文件系统中，有所谓的链接(link)，我们可以将其视为档案的别名，而链接又可分为两种：硬链接(hard link)与软链接(symbolic link)，硬链接的意思是一个档案可以有多个名称，而软链接的方式则是产生一个特殊的档案，该档案的内容是指向另一个档案的位置。硬链接是存在同一个文件系统中，而软链接却可以跨越不同的文件系统。

软链接：

1. 软链接，以路径的形式存在。类似于Windows操作系统中的快捷方式
2. 软链接可以跨文件系统，硬链接不可以
3. 软链接可以对一个不存在的文件名进行链接
4. 软链接可以对目录进行链接



ln命令续

硬链接：

1. 硬链接，以文件副本的形式存在。但不占用实际空间。
2. 不允许给目录创建硬链接
3. 硬链接只有在同一个文件系统中才能创建

注意：

第一，`ln`命令会保持每一处链接文件的同步性，也就是说，不论你改动了哪一处，其它的文件都会发生相同的变化；

第二，`ln`的链接又分软链接和硬链接两种，软链接就是`ln -s 源文件 目标文件`，它只会在你选定的位置上生成一个文件的镜像，不会占用磁盘空间，硬链接 `ln 源文件 目标文件`，没有参数`-s`，它会在你选定的位置上生成一个和源文件大小相同的文件，无论是软链接还是硬链接，文件都保持同步变化。



ln命令续

参数：

- b 删除，覆盖以前建立的链接
- d 允许超级用户制作目录的硬链接
- f 强制执行
- i 交互模式，文件存在则提示用户是否覆盖
- n 把符号链接视为一般目录
- s 软链接(符号链接)
- v 显示详细的处理过程

举例：

```
ln -s log2013.log link2013      软链接
ln log2013.log link2013          硬链接
```



diff命令

diff命令能比较单个文件或者目录内容。如果指定比较的是文件，则只有当输入为文本文件时才有效。以逐行的方式，比较文本文件的异同处。如果指定比较的是目录的时候，**diff** 命令会比较两个目录下名字相同的文本文件。列出不同的二进制文件、公共子目录和只在一个目录出现的文件。

举例：

<code>diff log2014.log log2013.log</code>	比较两个文件
<code>diff log2013.log log2014.log -y -W 50</code>	并排格式输出
<code>diff log2013.log log2014.log -c</code>	上下文输出格式
<code>diff log2014.log log2013.log -u</code>	统一格式输出
<code>diff test3 test6</code>	比较文件夹不同



grep命令

grep命令用于过滤/搜索的特定字符。可使用正则表达式能多种命令配合使用，使用上十分灵活。（[grep命令详解](#)）

举例：

`ps -ef|grep svn` 查找指定进程

`ps -ef|grep svn -c`

`ps -ef|grep -c svn` 查找指定进程个数

`cat test.txt | grep -f test2.txt`

从文件中读取关键词进行搜索

`cat test.txt | grep -nf test2.txt`

从文件中读取关键词进行搜索且显示行号

`grep '[a-z]\{7\}' *.txt`

显示当前目录下面以.txt 结尾的文件中的所有包含每个字符串至少有7个连续小写字符的字符串的行



WC命令

WC命令用于统计指定文件中的字节数、字数、行数，并将统计结果显示输出。该命令统计指定文件中的字节数、字数、行数。如果没有给出文件名，则从标准输入读取。WC同时也给出所指定文件的总统计数。

参数：

- c 统计字节数。
- l 统计行数。
- m 统计字符数。这个标志不能与 -c 标志一起使用。
- w 统计字数。一个字被定义为由空白、跳格或换行字符分隔的字符串。
- L 打印最长行的长度。



ps命令

ps命令用来显示当前进程的状态。它所提供的查看结果并不动态连续的；如果想对进程时间监控，应该用 top 工具。

ps工具标识进程的5种状态码：

- D 不可中断 uninterruptible sleep (usually IO)
- R 运行 runnable (on run queue)
- S 中断 sleeping
- T 停止 traced or stopped
- Z 僵死 a defunct ("zombie") process

参数：

- a 显示所有进程
- a 显示同一终端下的所有程序
- A 显示所有进程



ps命令续

c	显示进程的真实名称
-N	反向选择
-e	等于“-A”
e	显示环境变量
f	显示程序间的关系
-H	显示树状结构
r	显示当前终端的进程
T	显示当前终端的所有程序
u	指定用户的所有进程
-au	显示较详细的资讯
-aux	显示所有包含其他使用者的行程
-C<命令>	列出指定命令的状况
--lines<行数>	每页显示的行数
--width<字符数>	每页显示的字符数

Linux 目录结构

对于每一个 Linux 学习者来说，了解 Linux 文件系统的目录结构，是学好 Linux 的至关重要的一步，深入了解 Linux 文件目录结构的标准和每个目录的详细功能，对于我们用好 Linux 系统只管重要，下面我们就开始了解一下 Linux 目录结构的相关知识。

当在使用 Linux 的时候，如果您通过 `ls -l /` 就会发现，在 `/` 下包涵很多的目录，比如 `etc`、`usr`、`var`、`bin` ... 等目录，而在这些目录中，我们进去看看，发现也有很多的目录或文件。文件系统在 Linux 下看上去就象树形结构，所以我们可以把文件系统的结构形象的称为树形结构。

文件系统的是用来组织和排列文件存取的，所以她是可见的，在 Linux 中，我们可以通过 `ls` 等工具来查看其结构，在 Linux 系统中，我们见到的都是树形结构；比如操作系统安装在一个文件系统中，他表现为由 `/` 起始的树形结构。Linux 文件系统的最顶端是 `/`，我们称 `/` 为 Linux 的 `root`，也就是 Linux 操作系统的文件系统根目录。Linux 的文件系统的入口就是 `/`，所有的目录、文件、设备都在 `/` 之下，`/` 就是 Linux 文件系统的组织者，也是最上级的领导者。

由于 Linux 是开放源代码，各大公司和团体根据 Linux 的核心代码做各自的操作，编程。这样就造成在根下的目录的不同。这样就造成个人不能使用他人的 Linux 系统的 PC。因为你根本不知道一些基本的配置，文件在哪里。这就造成了混乱。这就是 FHS (Filesystem Hierarchy Standard) 机构诞生的原因。该机构是 Linux 爱好者自发的组成的一个团体，主要是是对 Linux 做一些基本的要求，不至于是操作者换一台主机就成了 Linux 的‘文盲’。

根据 FHS (<http://www.pathname.com/fhs/>) 的官方文件指出，他们的主要目的是希望让使用者可以了解到已安装软件通常放置于那个目录下，所以他们希望独立的软件开发商、操作系统制作者、以及想要维护系统的用户，都能够遵循FHS的标准。也就是说，FHS 的重点在于规范每个特定的目录下应该要放置什么样子的数据而已。这样做好处非常多，因为 Linux 操作系统就能够够在既有的面貌下(目录架构不变)发展出开发者想要的独特风格。

事实上，FHS 是根据过去的经验一直再持续的改版的，FHS 依据文件系统使用的频繁与否与是否允许使用者随意更动，而将目录定义成为四种交互作用的形态，用表格来说有点像底下这样：

	可分享的(shareable)	不可分享的(unshareable)
不变的 (static)	<code>/usr</code> (软件放置处)	<code>/etc</code> (配置文件)
	<code>/opt</code> (第三方协力软件)	<code>/boot</code> (开机与核心档)
可变动的 (variable)	<code>/var/mail</code> (使用者邮件信箱)	<code>/var/run</code> (程序相关)
	<code>/var/spool/news</code> (新闻组)	<code>/var/lock</code> (程序相关)

四种类型：

1. 可分享的：

可以分享给其他系统挂载使用的目录，所以包括执行文件与用户的邮件等数据，是能够分享给网络上其他主机挂载用的目录；

2. 不可分享的：

自己机器上面运作的装置文件或者是与程序有关的 `socket` 文件等，由于仅与自身机器有关，所以当然就不适合分享给其他主机了。

3. 不变的：

有些数据是不会经常变动的，跟随着 `distribution` 而不变动。例如函式库、文件说明文件、系统管理员所管理的主机服务配置文件等等；

4. 可变动的：

经常改变的数据，例如登录文件、一般用户可自行收受的新闻组等。

事实上，FHS 针对目录树架构仅定义出三层目录底下应该放置什么数据而已，分别是底下这三个目录的定义：

- / (root, 根目录): 与开机系统有关；
- /usr (unix software resource): 与软件安装/执行有关；
- /var (variable): 与系统运作过程有关。

一. 根目录 (/) 的意义与内容：

根目录是整个系统最重要的一个目录，因为不但所有的目录都是由根目录衍生出来的，同时根目录也与开机/还原/系统修复等动作有关。由于系统开机时需要特定的开机软件、核心文件、开机所需程序、函式库等等文件数据，若系统出现错误时，根目录也必须要包含有能够修复文件系统的程序才行。因为根目录是这么的重要，所以在 FHS 的要求方面，他希望根目录不要放在非常大的分区，因为越大的分区内你会放入越多的数据，如此一来根目录所在分区就可能会有较多发生错误的机会。

因此 FHS 标准建议：根目录(/)所在分区应该越小越好，且应用程序所安装的软件最好不要与根目录放在同一个分区内，保持根目录越小越好。如此不但效能较佳，根目录所在的文件系统也较不容易发生问题。说白了，就是根目录和Windows的C盘一个样。

根据以上原因，FHS认为根目录(/)下应该包含如下子目录：

目录	应放置档案内容
/bin	系统有很多放置执行档的目录，但/bin比较特殊。因为/bin放置的是在单人维护模式下还能够被操作的指令。在/bin底下的指令可以被root与一般帐号所使用，主要有：cat, chmod(修改权限), chown, date, mv, mkdir, cp, bash等等常用的指令。
/boot	主要放置开机会使用到的档案，包括Linux核心档案以及开机选单与开机所需设定档等等。Linux kernel常用的档名为：vmlinuz，如果使用的是grub这个开机管理程式，则还会存在 /boot/grub/这个目录。
/dev	在Linux系统上，任何装置与周边设备都是以档案的型态存在于这个目录当中。只要通过存取这个目录下的某个档案，就等于存取某个装置。比要重要的档案有 /dev/null, /dev/zero, /dev/tty , /dev/lp*, /dev/hd*, /dev/sd*等等
/etc	系统主要的设定档几乎都放置在这个目录内，例如人员的帐号密码档、各种服务的启始档等等。一般来说，这个目录下的各档案属性是可以让一般使用者查阅的，但是只有root有权力修改。FHS建议不要放置可执行档(binary)在这个目录中。比较重要的档案有：/etc/inittab, /etc/init.d/, /etc/modprobe.conf, /etc/X11/, /etc/fstab, /etc/sysconfig/等等。另外，其下重要的目录有：/etc/init.d/：所有服务的预设启动script都是放在这里的，例如要启动或者关闭iptables的话：/etc/init.d/iptables start、/etc/init.d/ iptables stop /etc/xinetd.d/：这就是所谓的super daemon管理的各项服务的设定档目录。/etc/X11/：与X Window有关的各种设定档都在这里，尤其是xorg.conf或XF86Config这两个X Server的设定档。

/home	这是系统预设的使用者家目录(home directory)。 在你新增一个一般使用者帐号时，预设的使用者家目录都会规范到这里来。比较重要的是，家目录有两种代号： ~ : 代表当前使用者的家目录，而 ~guest: 则代表用户名为 guest 的家目录。
/lib	系统的函式库非常的多，而/ lib 放置的则是在开机时会用到的函式库，以及在/ bin 或/ sbin 底下的指令会呼叫的函式库而已。 什么是函式库呢？你可以将他想成是外挂，某些指令必须要有这些外挂才能够顺利完成程式的执行之意。 尤其重要的是/ lib/modules/ 这个目录，因为该目录会放置核心相关的模组(驱动程式)。
/media	media 是媒体的英文，顾名思义，这个/ media 底下放置的就是可移除的装置。 包括软碟、光碟、DVD等等装置都暂时挂载于此。 常见的档名有： / media/floppy , / media/cdrom 等等。
/mnt	如果妳想要暂时挂载某些额外的装置，一般建议妳可以放置到这个目录中。在古早时候，这个目录的用途与/ media 相同啦。 只是有了/ media 之后，这个目录就用来暂时挂载用了。
/opt	这个是给第三方协力软体放置的目录。 什么是第三方协力软体啊？举例来说， KDE 这个桌面管理系统是一个独立的计画，不过他可以安装到 Linux 系统中，因此 KDE 的软体就建议放置到此目录下了。 另外，如果妳想要自行安装额外的软体(非原本的 distribution 提供的)，那么也能够将你的软体安装到这里来。 不过，以前的 Linux 系统中，我们还是习惯放置在/ usr/local 目录下。
/root	系统管理员(root)的家目录。 之所以放在这里，是因为如果进入单人维护模式而仅挂载根目录时，该目录就能够拥有 root 的家目录，所以我们会希望 root 的家目录与根目录放置在同一个分区中。
/sbin	Linux 有非常多指令是用来设定系统环境的，这些指令只有 root 才能够利用来设定系统，其他使用者最多只能用来查询而已。放在/ sbin 底下的为开机过程中所需要的，里面包括了开机、修复、还原系统所需要的指令。至于某些伺服器软体程式，一般则放置到/ usr/sbin/ 当中。至于本机自行安装的软体所产生的系统执行档(system binary)，则放置到/ usr/local/sbin/ 当中了。常见的指令包括： fdisk , fsck , ifconfig , init , mkfs 等等。
/srv	srv 可以视为 service 的缩写，是一些网路服务启动之后，这些服务所需要取用的资料目录。 常见的服务例如 WWW , FTP 等等。 举例来说， WWW 伺服器需要的网页资料就可以放置在/ srv/www/ 里面。呵呵，看来平时我们编写的代码应该放到这里了。
/tmp	这是让一般使用者或者是正在执行的程序暂时放置档案的地方。这个目录是任何人都能够存取的，所以你需要定期的清理一下。当然，重要资料不可放置在此目录啊。 因为 FHS 甚至建议在开机时，应该要将/ tmp 下的资料都删除。

事实上**FHS**针对根目录所定义的标准就仅限于上表，不过仍旧有些目录也需要我们了解一下，具体如下：

目录	应放置文件内容
/lost+found	这个目录是使用标准的 ext2/ext3 档案系统格式才会产生的一个目录，目的在于当档案系统发生错误时，将一些遗失的片段放置到这个目录下。 这个目录通常会在分割槽的最顶层存在，例如你加装一个硬盘于/ disk 中，那在这个系统下就会自动产生一个这样的目录/ disk/lost+found
/proc	这个目录本身是一个虚拟文件系统(virtual file system)喔。 他放置的资料都是在内存当中，例如系统核心、行程资讯(process)（是进程吗?）、周边装置的状态及网络状态等等。因为这个目录下的资料都是在记忆体（内存）当中，所以本身不占任何硬盘空间。比较重要的档案（目录）例

	如: /proc/cpuinfo, /proc/dma, /proc/interrupts, /proc/ioports, /proc/net/*等等。呵呵, 是虚拟内存吗[guest]?
/sys	这个目录其实跟/proc非常类似, 也是一个虚拟的档案系统, 主要也是记录与核心相关的资讯。包括目前已载入的核心模组与核心侦测到的硬体装置资讯等等。这个目录同样不占硬盘容量。

除了这些目录的内容之外, 另外要注意的是, 因为根目录与开机有关, 开机过程中仅有根目录会被挂载, 其他分区则是在开机完成之后才会持续的进行挂载的行为。就是因为如此, 因此根目录下与开机过程有关的目录, 就不能够与根目录放到不同的分区去。那哪些目录不可与根目录分开呢? 有底下这些:

- /etc: 配置文件
- /bin: 重要执行档
- /dev: 所需要的装置文件
- /lib: 执行档所需的函式库与核心所需的模块
- /sbin: 重要的系统执行文件

这五个目录千万不可与根目录分开在不同的分区。请背下来啊。

二. /usr 的意义与内容:

依据 FHS 的基本定义, /usr 里面放置的数据属于可分享的与不可变动的 (shareable, static), 如果你知道如何透过网络进行分区的挂载(例如在服务器篇会谈到的NFS服务器), 那么/usr确实可以分享给局域网络内的其他主机来使用喔。

/usr 不是 user 的缩写, 其实 usr 是 Unix Software Resource 的缩写, 也就是 Unix 操作系统软件资源所放置的目录, 而不是用户的数据啦。这点要注意。FHS 建议所有软件开发者, 应该将他们的数据合理的分别放置到这个目录下的次目录, 而不要自行建立该软件自己独立的目录。

因为是所有系统默认的软件(distribution发布者提供的软件)都会放置到/usr底下, 因此这个目录有点类似Windows 系统的C:\Windows\ + C:\Program files\这两个目录的综合体, 系统刚安装完毕时, 这个目录会占用最多的硬盘容量。一般来说, /usr的次目录建议有底下这些:

目录	应放置文件内容
/usr/X11R6/	为X Window System重要数据所放置的目录, 之所以取名为X11R6是因为最后的X版本为第11版, 且该版的第6次释出之意。
/usr/bin/	绝大部分的用户可使用指令都放在这里。请注意到他与/bin的不同之处。(是否与开机过程有关)
/usr/include/	c/c++等程序语言的档头(header)与包含档(include)放置处, 当我们以tarball方式 (*.tar.gz 的方式安装软件)安装某些数据时, 会使用到里头的许多包含档。
/usr/lib/	包含各应用软件的函式库、目标文件(object file), 以及不被一般使用者惯用的执行档或脚本(script)。某些软件会提供一些特殊的指令来进行服务器的设定, 这些指令也不会经常被系统管理员操作, 那就会被摆放到这个目录下啦。要注意的是, 如果你使用的是X86_64的Linux系统, 那可能会有/usr/lib64/目录产生
/usr/local/	系统管理员在本机自行安装自己下载的软件(非distribution默认提供者), 建议安装到此目录, 这样会比较便于管理。举例来说, 你的distribution提供的软件较旧, 你想安装较新的软件但又不想移除旧版, 此时你可以将新版软件安装于

	/usr/local/目录下，可与原先的旧版软件有分别啦。你可以自行到/usr/local去看看，该目录下也是具有bin, etc, include, lib...的次目录
/usr/sbin/	非系统正常运作所需要的系统指令。最常见的就是某些网络服务器软件的服务指令(daemon)
/usr/share/	放置共享文件的地方，在这个目录下放置的数据几乎是不分硬件架构均可读取的数据，因为几乎都是文本文件嘛。在此目录下常见的还有这些次目录： /usr/share/man: 联机帮助文件 /usr/share/doc: 软件杂项的文件说明 /usr/share/zoneinfo: 与时区有关的时区文件
/usr/src/	一般原始码建议放置到这里，src有source的意思。至于核心原始码则建议放置到/usr/src/Linux/目录下。

三. /var 的意义与内容：

如果/usr是安装时会占用较大硬盘容量的目录，那么/var就是在系统运作后才会渐渐占用硬盘容量的目录。因为/var目录主要针对常态性变动的文件，包括缓存(cache)、登录档(log file)以及某些软件运作所产生的文件，包括程序文件(lock file, run file)，或者例如MySQL数据库的文件等等。常见的次目录有：

目录	应放置文件内容
/var/cache/	应用程序本身运作过程中会产生的一些暂存档
/var/lib/	程序本身执行的过程中，需要使用到的数据文件放置的目录。在此目录下各自的软件应该要有各自的目录。举例来说，MySQL的数据库放置到/var/lib/mysql/而rpm的数据库则放到/var/lib/rpm去
/var/lock/	某些装置或者是文件资源一次只能被一个应用程序所使用，如果同时有两个程序使用该装置时，就可能产生一些错误的状况，因此就得要将该装置上锁(lock)，以确保该装置只会给单一软件所使用。举例来说，刻录机正在刻录一块光盘，你想一下，会不会有两个人同时在使用一个刻录机烧片？如果两个人同时刻录，那片子写入的是谁的数据？所以当第一个人在刻录时该刻录机就会被上锁，第二个人就得要该装置被解除锁定(就是前一个人用完了)才能够继续使用
/var/log/	非常重要。这是登录文件放置的目录。里面比较重要的文件如/var/log/messages, /var/log/wtmp(记录登入者的信息)等。
/var/mail/	放置个人电子邮件信箱的目录，不过这个目录也被放置到/var/spool/mail/目录中，通常这两个目录是互为链接文件。
/var/run/	某些程序或者是服务启动后，会将他们的PID放置在这个目录下
/var/spool/	这个目录通常放置一些队列数据，所谓的“队列”就是排队等待其他程序使用的数据。这些数据被使用后通常都会被删除。举例来说，系统收到新信会放置到/var/spool/mail/中，但使用者收下该信件后该封信原则上就会被删除。信件如果暂时寄不出去会被放到/var/spool/mqueue/中，等到被送出后就被删除。如果是工作排程数据(crontab)，就会被放置到/var/spool/cron/目录中。

由于FHS仅是定义出最上层(/)及次层(/usr, /var)的目录内容应该要放置的文件或目录数据，因此，在其他次目录层级内，就可以随开发者自行来配置了。

四. 目录树(directory tree) :

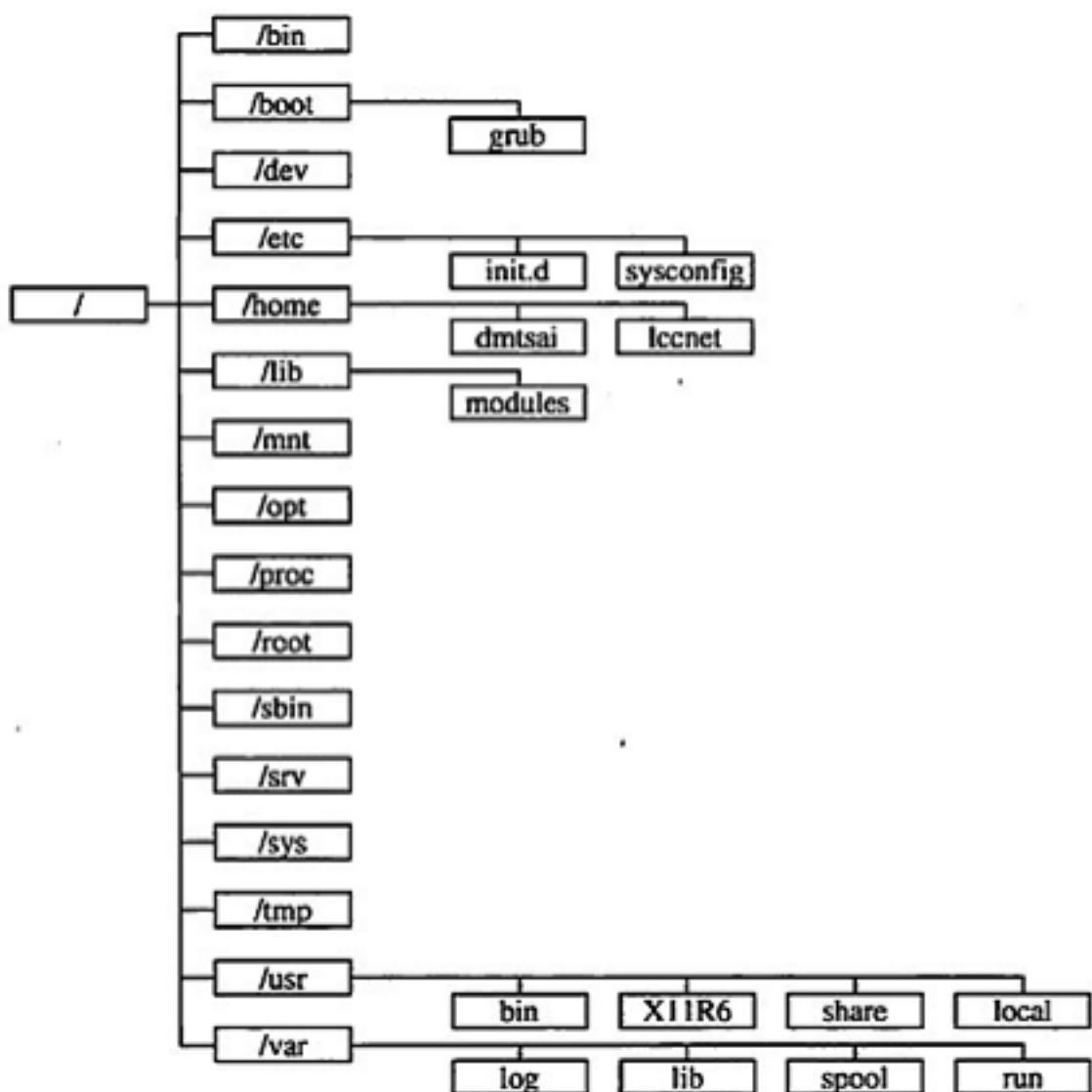
在Linux底下，所有的文件与目录都是由根目录开始的。那是所有目录与文件的源头，然后再一个一个的分支下来，因此，我们也称这种目录配置方式为：目录树(directory tree)，这个目录树的主要特性有：

目录树的启始点为根目录 (/, root)；

每一个目录不止能使用本地端的 partition 的文件系统，也可以使用网络上的 filesystem。举例来说，可以利用 Network File System (NFS) 服务器挂载某特定目录等。

每一个文件在此目录树中的文件名(包含完整路径)都是独一无二的。

如果我们将整个目录树以图的方法来显示，并且将较为重要的文件数据列出来的话，那么目录树架构就如下图所示：



五. 绝对路径与相对路径

除了需要特别注意的FHS目录配置外，在文件名部分我们也要特别注意。因为根据档名写法的不同，也可将所谓的路径(path)定义为绝对路径(absolute)与相对路径(relative)。这两种文件名/路径的写法依据是这样的：

绝对路径：

由根目录(/)开始写起的文件名或目录名称，例如 /home/dmtsai/.bashrc；

相对路径：

相对于目前路径的文件名写法。例

如 ./home/dmtsai 或 http://www.cnblogs.com/home/dmtsai/ 等等。反正开头不是 / 就属于相对路径的写法

而你必须要了解，相对路径是以你当前所在路径的相对位置来表示的。举例来说，你目前在 /home 这个目录下，如果想要进入 /var/log 这个目录时，可以怎么写呢？

```
cd /var/log    (absolute)
cd ../var/log (relative)
```

因为你在 /home 底下，所以要回到上一层(../)之后，才能继续往 /var 来移动的，特别注意这两个特殊的目录：

. : 代表当前的目录，也可以使用 ./ 来表示；
.. : 代表上一层目录，也可以 ../ 来代表。

这个 . 与 .. 目录概念是很重要的，你常常会看到 cd .. 或 ./command 之类的指令下达方式，就是代表上一层与目前所在目录的工作状态。

实例1：如何先进入 /var/spool/mail/ 目录，再进入到 /var/spool/cron/ 目录内？

命令：

```
cd /var/spool/mail
cd ../cron
```

说明：

由于 /var/spool/mail 与 /var/spool/cron 是同样在 /var/spool/ 目录中。如此就不需要在由根目录开始写起了。这个相对路径是非常有帮助的，尤其对于某些软件开发商来说。一般来说，软件开发商会将数据放置到 /usr/local/ 里面的各相对目录。但如果用户想要安装到不同目录呢？就得要使用相对路径。

实例2：网络文件常常提到类似 ./run.sh 之类的数据，这个指令的意义为何？

说明：

由于指令的执行需要变量的支持，若你的执行文件放置在本目录，并且本目录并非正规的执行文件目录 (/bin, /usr/bin 等为正规)，此时要执行指令就得要严格指定该执行档。./ 代表本目录的意思，所以 ./run.sh 代表执行本目录下，名为 run.sh 的文件。

Linux文件类型与拓展名

Linux文件类型和Linux文件的文件名所代表的意义是两个不同的概念。我们通过一般应用程序而创建的比如file.txt、file.tar.gz，这些文件虽然要用不同的程序来打开，但放在Linux文件类型中衡量的话，大多是常规文件（也被称为普通文件）。

一. 文件类型

Linux文件类型常见的有：普通文件、目录文件、字符设备文件和块设备文件、符号链接文件等，现在我们进行一个简要的说明。

1. 普通文件

我们用 ls -lh 来查看某个文件的属性，可以看到有类似-rwxrwxrwx，值得注意的是第一个符号是 -，这样的文件在Linux中就是普通文件。这些文件一般是用一些相关的应用程序创建，比如图像工具、文档工具、归档工具……或 cp工具等。这类文件的删除方式是用rm 命令。另外，依照文件的内容，又大略可以分为：

1>. 纯文本档(ASCII):

这是Linux系统中最的一种文件类型，称为纯文本档是因为内容为我们人类可以直接读到的数据，例如数字、字母等等。几乎只要我们可以用来做为设定的文件都属于这一种文件类型。举例来说，你可以用命令： cat ~/.bashrc 来看到该文件的内容。（cat 是将一个文件内容读出来的指令）。

2>. 二进制文件(binary):

Linux系统其实仅认识且可以执行二进制文件(binary file)。Linux当中的可执行文件(scripts，文字型批处理文件不算)就是这种格式的文件。刚刚使用的命令cat就是一个binary file。

3>. 数据格式文件(data):

有些程序在运作的过程当中会读取某些特定格式的文件，那些特定格式的文件可以被称为数据文件 (data file)。举例来说，我们的Linux在使用者登录时，都会将登录的数据记录在 /var/log/wtmp 那个文件内，该文件是一个data file，他能够透过last这个指令读出来！但是使用cat时，会读出乱码～因为他是属于一种特殊格式的文件？

2. 目录文件

当我们在某个目录下执行，看到有类似 drwxr-xr-x，这样的文件就是目录，目录在Linux是一个比较特殊的文件。注意它的第一个字符是d。创建目录的命令可以用 mkdir 命令，或cp命令，cp可以把一个目录复制为另一个目录。删除用rm 或rmdir命令。

3. 字符设备或块设备文件

如您进入/dev目录，列一下文件，会看到类似如下的：

```
[root@localhost ~]# ls -al /dev/tty
crw-rw-rw- 1 root tty 5, 0 11-03 15:11 /dev/tty
[root@localhost ~]# ls -la /dev/sda1
brw-r---- 1 root disk 8, 1 11-03 07:11 /dev/sda1
```

我们看到/dev/tty的属性是 crw-rw-rw-，注意前面第一个字符是 c，这表示字符设备文件。比如猫等串口设备。我们看到 /dev/sda1 的属性是 brw-r----，注意前面的第一个字符是b，这表示块设备，比如硬盘，光驱等设备。

这个种类的文件，是用mknod来创建，用rm来删除。目前在最新的Linux发行版本中，我们一般不用自己来创建设备文件。因为这些文件是和内核相关联的。

与系统周边及储存等相关的一些文件，通常都集中在/**dev**这个目录之下！通常又分为两种：

区块(**block**)设备档：

就是一些储存数据，以提供系统随机存取的接口设备，举例来说，硬盘与软盘等就是啦！你可以随机的在硬盘的不同区块读写，这种装置就是成组设备！你可以自行查一下/**dev/sda**看看，会发现第一个属性为[b]！

字符(**character**)设备文件：

亦即是一些串行端口的接口设备，例如键盘、鼠标等等！这些设备的特色就是一次性读取的，不能够截断输出。举例来说，你不可能让鼠标跳到另一个画面，而是滑动到另一个地方！第一个属性为 [c]。

4. 数据接口文件(**sockets**):

数据接口文件（或者：套接口文件），这种类型的文件通常被用在网络上的数据承接了。我们可以启动一个程序来监听客户端的要求，而客户端就可以透过这个**socket**来进行数据的沟通了。第一个属性为 [s]，最常在/**var/run**这个目录中看到这种文件类型了。

例如：当我们启动MySQL服务器时，会产生一个**mysql.sock**的文件。

```
[root@localhost ~]# ls -lh /var/lib/mysql/mysql.sock
srwxrwxrwx 1 mysql mysql 0 04-19 11:12 /var/lib/mysql/mysql.sock
```

注意这个文件的第一个字符是 s。

5. 符号链接文件：

当我们查看文件属性时，会看到有类似 **lrwxrwxrwx**，注意第一个字符是l，这类文件是链接文件。是通过**ln -s 源文件名 新文件名**。上面是一个例子，表示**setup.log**是**install.log**的软链接文件。怎么理解呢？这和**Windows**操作系统中的快捷方式有点相似。

符号链接文件的创建方法举例：

```
[root@localhost test]# ls -lh log2012.log
-rw-r--r-- 1 root root 296K 11-13 06:03 log2012.log
[root@localhost test]# ln -s log2012.log linklog.log
[root@localhost test]# ls -lh *.log
lrwxrwxrwx 1 root root 11 11-22 06:58 linklog.log -> log2012.log
-rw-r--r-- 1 root root 296K 11-13 06:03 log2012.log
```

6. 数据输送文件 (**FIFO, pipe**) :

FIFO也是一种特殊的文件类型，他主要的目的在解决多个程序同时存取一个文件所造成的问题。 **FIFO**是**first-in-first-out**的缩写。第一个属性为[p]。

二. Linux文件扩展名

1. 扩展名类型

基本上，Linux的文件是没有所谓的扩展名的，一个Linux文件能不能被执行，与他的第一栏的十个属性有关，与档名根本一点关系也没有。这个观念跟**Windows**的情况不相同喔！在**Windows**底下，能被执行的文件扩展名通常是 .com .exe .bat 等等，而在Linux底下，只要你的权限当中具有x的话，例如 [-rwx-r-xr-x] 即代表这个文件可以被执行。

不过，可以被执行跟可以执行成功是不一样的～举例来说，在`root`家目录下的`install.log` 是一个纯文本档，如果经由修改权限成为 `-rwxrwxrwx` 后，这个文件能够真的执行成功吗？当然不行～因为他的内容根本就没有可以执行的数据。所以说，这个`x`代表这个文件具有可执行的能力，但是能不能执行成功，当然就得要看该文件的内容。

虽然如此，不过我们仍然希望可以藉由扩展名来了解该文件是什么东西，所以，通常我们还是会以适当的扩展名来表示该文件是什么种类的。底下有数种常用的扩展名：

`*.sh`：脚本或批处理文件 (`scripts`)，因为批处理文件为使用 `shell` 写成的，所以扩展名就编成 `.sh`

`*Z, *.tar, *.tar.gz, *.zip, *.tgz`：经过打包的压缩文件。这是因为压缩软件分别为 `gunzip`, `tar` 等等的，由于不同的压缩软件，而取其相关的扩展名！

`*.html, *.php`：网页相关文件，分别代表 `HTML` 语法与 `PHP` 语法的网页文件。`.html` 的文件可使用网页浏览器来直接开启，至于 `.php` 的文件，则可以透过 `client` 端的浏览器来 `server` 端浏览，以得到运算后的网页结果。

基本上，`Linux`系统上的文件名真的只是让你了解该文件可能的用途而已，真正的执行与否仍然需要权限的规范才行。例如虽然有一个文件为可执行文件，如常见的`/bin/ls`这个显示文件属性的指令，不过，如果这个文件的权限被修改成无法执行时，那么`ls`就变成不能执行。

上述的这种问题最常发生在文件传送的过程中。例如你在网络上下载一个可执行文件，但是偏偏在你的 `Linux` 系统中就是无法执行！呵呵！那么就是可能文件的属性被改变了。不要怀疑，从网络上传送到你的 `Linux` 系统中，文件的属性与权限确实是会被改变的。

2. Linux文件名长度限制：

在`Linux`底下，使用预设的`Ext2/Ext3`文件系统时，针对文件名长度限制为：

单一文件或目录的最大容许文件名为 255 个字符

包含完整路径名称及目录 (/) 之完整档名为 4096 个字符

是相当长的档名！我们希望`Linux`的文件名可以一看就知道该文件在干嘛的，所以档名通常是很长很长。

3. Linux文件名的字符的限制：

由于`Linux`在文字接口下的一些指令操作关系，一般来说，你在设定`Linux`底下的文件名时，最好可以避免一些特殊字符比较好！例如底下这些：

`* ? > < ; & ! [] | \ ' " ` () { }`

因为这些符号在文字接口下，是有特殊意义的。另外，文件名的开头为小数点“.”时，代表这个文件为隐藏文件！同时，由于指令下达当中，常常会使用到 `-option` 之类的选项，所以你最好也避免将文件档名的开头以 `-` 或 `+` 来命名。

Linux文件属性详解

Linux 文件或目录的属性主要包括：文件或目录的节点、种类、权限模式、链接数量、所归属的用户和用户组、最近访问或修改的时间等内容。具体情况如下：

命令：

```
ls -lih
```

输出：

```
[root@localhost test]# ls -lih
总计 316K
2095120 lrwxrwxrwx 1 root root    11 11-22 06:58 linklog.log -> log2012.log
2095112 -rw-r--r-- 1 root root 296K 11-13 06:03 log2012.log
2095110 -rw-r--r-- 1 root root    61 11-13 06:03 log2013.log
2095107 -rw-r--r-- 1 root root     0 11-13 06:03 log2014.log
2095117 -rw-r--r-- 1 root root     0 11-13 06:06 log2015.log
2095118 -rw-r--r-- 1 root root     0 11-16 14:41 log2016.log
2095119 -rw-r--r-- 1 root root     0 11-16 14:43 log2017.log
2095113 drwxr-xr-x 6 root root 4.0K 10-27 01:58 scf
2095109 drwxrwxr-x 2 root root 4.0K 11-13 06:08 test3
2095131 drwxrwxr-x 2 root root 4.0K 11-13 05:50 test4
```

说明：

第一列：inode

第二列：文件种类和权限；

第三列：硬链接个数；

第四列：属主；

第五列：所归属的组；

第六列：文件或目录的大小；

第七列和第八列：最后访问或修改时间；

第九列：文件名或目录名

我们以log2012.log为例：

```
2095112 -rw-r--r-- 1 root root 296K 11-13 06:03 log2012.log
```

inode 的值是：2095112

文件类型：文件类型是-，表示这是一个普通文件； 关于文件的类型，请参考：[每天一个linux命令\(24\)：Linux文件类型与扩展名](#)

文件权限：文件权限是rw-r--r--，表示文件属主可读、可写、不可执行，文件所归属的用户组不可写，可读，不可执行，其它用户不可写，可读，不可执行；

硬链接个数： log2012.log这个文件没有硬链接；因为数值是1，就是他本身；

文件属主：也就是这个文件归哪于哪个用户，它归于root，也就是第一个root；

文件属组：也就是说，对于这个文件，它归属于哪个用户组，在这里是root用户组；

文件大小：文件大小是296k个字节；

访问可修改时间：这里的时间是最后访问的时间，最后访问和文件被修改或创建的时间，有时并不是一致的；
当然文档的属性不仅仅包括这些，这些是我们最常用的一些属性。

关于inode：

inode 译成中文就是索引节点。每个存储设备或存储设备的分区（存储设备是硬盘、软盘、U盘等等）被格式化为文件系统后，应该有两部份，一部份是inode，另一部份是Block，Block是用来存储数据用的。而inode呢，就是用来存储这些数据的信息，这些信息包括文件大小、属主、归属的用户组、读写权限等。inode为每个文件进行信息索引，所以就有了inode的数值。操作系统根据指令，能通过inode值最快的找到相对应的文件。

做个比喻，比如一本书，存储设备或分区就相当于这本书，Block相当于书中的每一页，inode 就相当于这本书前面的目录，一本书有很多的内容，如果想查找某部份的内容，我们可以先查目录，通过目录能最快的找到我们想要看的内容。虽然不太恰当，但还是比较形象。

当我们用ls 查看某个目录或文件时，如果加上-i 参数，就可以看到inode节点了；比如我们前面所说的例子：

```
[root@localhost test]# ls -li log2012.log  
2095112 -rw-r--r-- 1 root root 302108 11-13 06:03 log2012.log
```

log2012.log 的 inode 值是 2095112；查看一个文件或目录的 inode，要通过 ls 命令的的 -i 参数。

TOP 命令详解

```
top - 21:02:06 up 41 days, 6:29, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 24 total, 1 running, 23 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0
KiB Mem: 262144 total, 119000 used, 143144 free, 0 buffers
KiB Swap: 131072 total, 116824 used, 14248 free. 26412 cached Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3265 root 20 0 46880 5040 1548 S 0.3 1.9 5:06.00 ssserver
1 root 20 0 28472 1140 436 S 0.0 0.4 0:16.80 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd+
3 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khelper/+
66 systemd+ 20 0 26128 168 120 S 0.0 0.1 0:04.37 systemd--+
70 root 20 0 38860 8 4 S 0.0 0.0 0:00.00 systemd--+
72 root 20 0 28816 1308 1200 S 0.0 0.5 0:06.69 systemd--+
278 systemd+ 20 0 25692 8 4 S 0.0 0.0 0:00.00 systemd--+
279 root 20 0 55128 376 256 S 0.0 0.1 0:00.81 sshd
810 root 20 0 30772 192 128 S 0.0 0.1 0:00.58 pure-ftp
2697 root 20 0 31448 4 0 S 0.0 0.0 0:00.00 nginx
2700 www 20 0 52676 964 548 S 0.0 0.4 0:04.60 nginx
2718 root 20 0 4280 8 4 S 0.0 0.0 0:00.01 mysqld_s+
3201 mysql 20 0 779456 6896 196 S 0.0 2.6 6:43.86 mysqld
3241 root 20 0 168312 136 80 S 0.0 0.1 0:42.06 php-fpm
```

统计信息区：

前五行是当前系统情况整体的统计信息区。下面我们看每一行信息的具体意义。

第一行，任务队列信息，同 `uptime` 命令的执行结果，具体参数说明情况如下：

`21:02:41` – 当前系统时间

`up 41 days, 6:29` – 系统已经运行了 41 天 6 小时 29 分钟（在这期间系统没有重启过的吆！）

`2 users` – 当前有 2 个用户登录系统

`load average: 0.00, 0.00, 0.00` – `load average` 后面的三个数分别是 1 分钟、5 分钟、15 分钟的负载情况。

`load average` 数据是每隔 5 秒钟检查一次活跃的进程数，然后按特定算法计算出的数值。如果这个数除以逻辑 CPU 的数量，结果高于 5 的时候就表明系统在超负荷运转了。

第二行，`Tasks` – 任务（进程），具体信息说明如下：

系统现在共有 24 个进程，其中处于运行中的有 1 个，23 个在休眠（sleep），stoped 状态的有 0 个，zombie 状态（僵尸）的有 0 个。

第三行，`cpu` 状态信息，具体属性说明如下：

`0.0%us` – 用户空间占用 CPU 的百分比。

`0.0% sy` – 内核空间占用 CPU 的百分比。

`0.0% ni` – 改变过优先级的进程占用 CPU 的百分比

100.0% id – 空闲 CPU 百分比

0.0% wa – IO 等待占用 CPU 的百分比

0.0% hi – 硬中断 (Hardware IRQ) 占用 CPU 的百分比

0.0% si – 软中断 (Software Interrupts) 占用 CPU 的百分比

备注：在这里 CPU 的使用比率和 windows 概念不同，需要理解 linux 系统用户空间和内核空间的相关知识！

第四行，内存状态，具体信息如下：

262144k total – 物理内存总量 (256M)

119000k used – 使用中的内存总量 (116.2M)

143144k free – 空闲内存总量 (139.7M)

0k buffers – 缓存的内存量 (0K)

第五行，swap 交换分区信息，具体信息说明如下：

131072k total – 交换区总量 (128M)

116824k used – 使用的交换区总量 (114M)

14248k free – 空闲交换区总量 (14M)

26412k cached – 缓冲的交换区总量 (25.8M)

备注：

第四行中使用中的内存总量 (used) 指的是现在系统内核控制的内存数，空闲内存总量 (free) 是内核还未纳入其管控范围的数量。纳入内核管理的内存不见得都在使用中，还包括过去使用过的现在可以被重复利用的内存，内核并不把这些可被重新使用的内存交还到 free 中去，因此在 linux 上 free 内存会越来越少，但不用为此担心。

如果出于习惯去计算可用内存数，这里有个近似的计算公式：第四行的 free + 第四行的 buffers + 第五行的 cached，按这个公式此台服务器的可用内存：

18537836k +169884k +3612636k = 22GB 左右。

对于内存监控，在 top 里我们要时刻监控第五行 swap 交换分区的 used，如果这个数值在不断的变化，说明内核在不断进行内存和 swap 的数据交换，这是真正的内存不够用了。

第六行，空行。

第七行以下：各进程（任务）的状态监控，项目列信息说明如下：

PID – 进程 id

USER – 进程所有者

PR – 进程优先级

NI – nice 值。负值表示高优先级，正值表示低优先级

VIRT – 进程使用的虚拟内存总量，单位 kb。VIRT=SWAP+RES

RES – 进程使用的、未被换出的物理内存大小，单位 kb。RES=CODE+DATA

SHR – 共享内存大小，单位 kb

S – 进程状态。D=不可中断的睡眠状态 R=运行 S=睡眠 T=跟踪/停止 Z=僵尸进程

%CPU – 上次更新到现在的 CPU 时间占用百分比

%MEM – 进程使用的物理内存百分比

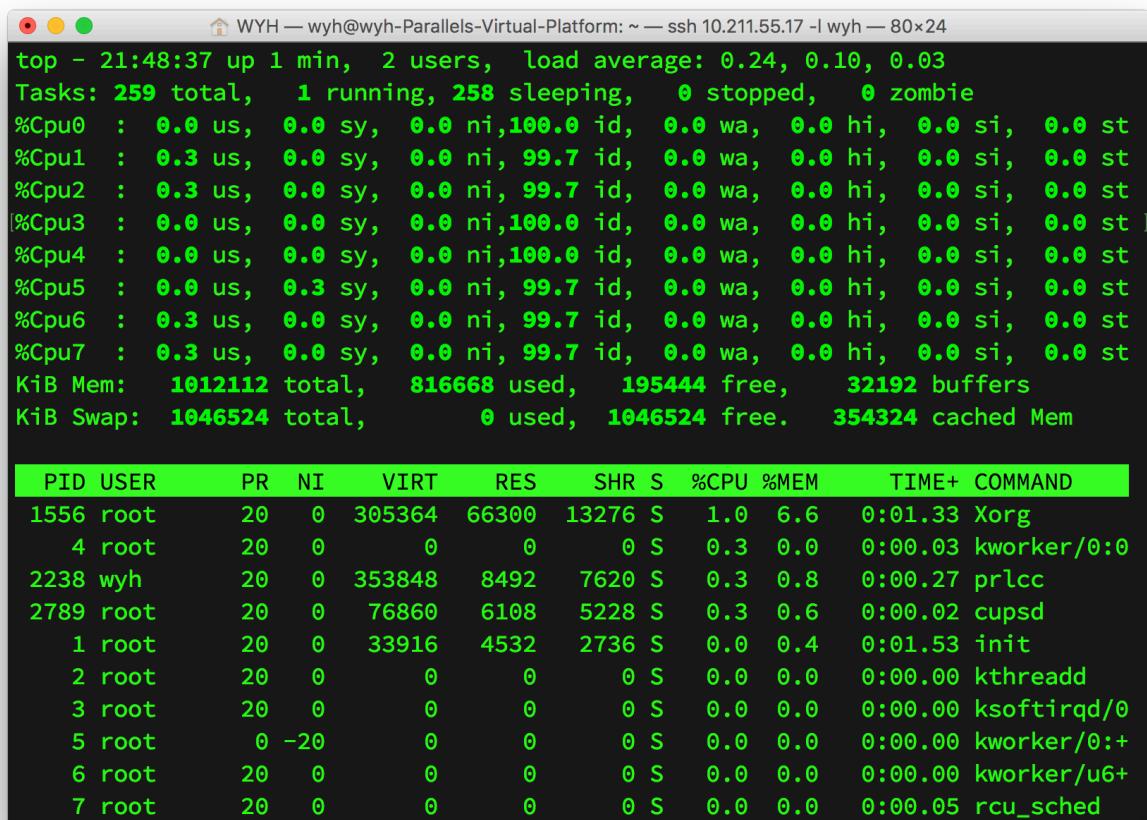
TIME+ – 进程使用的 CPU 时间总计，单位 1/100 秒

COMMAND – 进程名称（命令名/命令行）

其他使用技巧:

1. 多U多核CPU监控

在top基本视图中，按键盘数字“1”，可监控每个逻辑CPU的状况：



WYH — wyh@wyh-Parallels-Virtual-Platform: ~ — ssh 10.211.55.17 -l wyh — 80x24

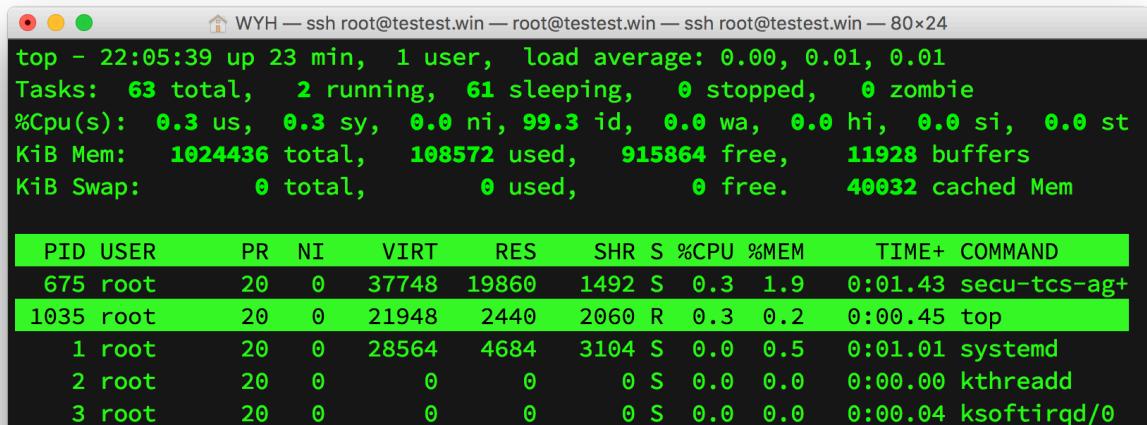
```
top - 21:48:37 up 1 min, 2 users, load average: 0.24, 0.10, 0.03
Tasks: 259 total, 1 running, 258 sleeping, 0 stopped, 0 zombie
%Cpu0 : 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
%Cpu1 : 0.3 us, 0.0 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
%Cpu2 : 0.3 us, 0.0 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
%Cpu3 : 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
%Cpu4 : 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
%Cpu5 : 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
%Cpu6 : 0.3 us, 0.0 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
%Cpu7 : 0.3 us, 0.0 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1012112 total, 816668 used, 195444 free, 32192 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free. 354324 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1556	root	20	0	305364	66300	13276	S	1.0	6.6	0:01.33	Xorg
4	root	20	0	0	0	0	S	0.3	0.0	0:00.03	kworker/0:0
2238	wyh	20	0	353848	8492	7620	S	0.3	0.8	0:00.27	prlcc
2789	root	20	0	76860	6108	5228	S	0.3	0.6	0:00.02	cupsd
1	root	20	0	33916	4532	2736	S	0.0	0.4	0:01.53	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:+
6	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kworker/u6+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.05	rcu_sched

观察上图，服务器有8个逻辑CPU。再按数字键1，就会返回到top基本视图界面。

2. 高亮显示当前运行进程

敲击键盘“b”（打开/关闭加亮效果），top的视图变化如下：



WYH — ssh root@testest.win — root@testest.win — ssh root@testest.win — 80x24

```
top - 22:05:39 up 23 min, 1 user, load average: 0.00, 0.01, 0.01
Tasks: 63 total, 2 running, 61 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.3 sy, 0.0 ni, 99.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1024436 total, 108572 used, 915864 free, 11928 buffers
KiB Swap: 0 total, 0 used, 0 free. 40032 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
675	root	20	0	37748	19860	1492	S	0.3	1.9	0:01.43	secu-tcs-ag+
1035	root	20	0	21948	2440	2060	R	0.3	0.2	0:00.45	top
1	root	20	0	28564	4684	3104	S	0.0	0.5	0:01.01	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.04	ksoftirqd/0

我们发现进程id为1035的“top”进程被加亮了，top进程就是视图第二行显示的唯一的运行态（running）的那个进程，可以通过敲击“y”键关闭或打开运行态进程的加亮效果。

3. 进程字段排序

默认进入top时，各进程是按照CPU的占用量来排序的，在下图中进程ID为1的/sbin/init进程排在第一（cpu占用0.0%），进程ID为2的kthreadd进程排在第二（cpu占用0.0%）。

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	28564	4684	3104	S	0.0	0.5	0:01.02	/sbin/init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kthreadd]
3	root	20	0	0	0	0	S	0.0	0.0	0:00.05	[ksoftirqd/+]
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	[kworker/0:+]
6	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kworker/u2+]

敲击键盘“x”（打开/关闭排序列的加亮效果），top的视图变化如下：

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	28564	4684	3104	S	0.3	0.5	0:01.02	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.04	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kworker/u2:0
7	root	20	0	0	0	0	R	0.0	0.0	0:00.08	rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0

可以看到，top默认的排序列是“%CPU”。

4. 通过“shift + >”或“shift + <”可以向右或左改变排序列

下图是按一次“shift + >”的效果图，视图现在已经按照%MEM来排序。

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
675	root	20	0	37748	19860	1492	S	0.0	1.9	0:02.00	secu-tcs-ag+
777	root	20	0	80644	5972	5108	S	0.0	0.6	0:00.23	sshd
339	root	20	0	55180	5288	4624	S	0.0	0.5	0:00.00	sshd
1	root	20	0	28564	4684	3104	S	0.0	0.5	0:01.02	systemd
779	root	20	0	21344	4420	2840	S	0.0	0.4	0:00.02	bash
406	root	20	0	258672	3536	2712	S	0.0	0.3	0:00.00	rsyslogd
345	message+	20	0	42124	3348	2980	S	0.0	0.3	0:00.01	dbus-daemon

5. top交互命令

在top 命令执行过程中可以使用的一些交互命令。这些命令都是单字母的，如果在命令行中使用了s 选项， 其中一些命令可能会被屏蔽。

h 显示帮助画面，给出一些简短的命令总结说明

k 终止一个进程。

i 忽略闲置和僵死进程。这是一个开关式命令。

q 退出程序

r 重新安排一个进程的优先级别

s 切换到累计模式

s 改变两次刷新之间的延迟时间（单位为s），如果有小数，就换算成m s。输入0值则系统将不断刷新，默认值是5 s

f或者F 从当前显示中添加或者删除项目

o或者O 改变显示项目的顺序

l 切换显示平均负载和启动时间信息
m 切换显示内存信息
t 切换显示进程和**CPU**状态信息
c 切换显示命令名称和完整命令行
M 根据驻留内存大小进行排序
P 根据**CPU**使用百分比大小进行排序
T 根据时间/累计时间进行排序
w 将当前设置写入`~/.toprc`文件中

grep 命令详解

Linux 系统中 grep 命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。grep 全称是 Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。

grep 的工作方式是这样的，它在一个或多个文件中搜索字符串模板。如果模板包括空格，则必须被引用，模板后的所有字符串被看作文件名。搜索的结果被送到标准输出，不影响原文件内容。

grep 可用于 shell 脚本，因为 grep 通过返回一个状态值来说明搜索的状态，如果模板搜索成功，则返回 0，如果搜索不成功，则返回 1，如果搜索的文件不存在，则返回 2。我们利用这些返回值就可进行一些自动化的文本处理工作。

1. 命令格式：

```
grep [option] pattern file
```

2. 命令功能：

用于过滤/搜索的特定字符。可使用正则表达式能多种命令配合使用，使用上十分灵活。

3. 命令参数：

-a --text #不要忽略二进制的数据。

-A<显示行数> --after-context=<显示行数> #除了显示符合范本样式的那一列之外，并显示该行之后的内容。

-b --byte-offset #在显示符合样式的那一行之前，标示出该行第一个字符的编号。

-B<显示行数> --before-context=<显示行数> #除了显示符合样式的那一行之外，并显示该行之前的内容。

-c --count #计算符合样式的列数。

-C<显示行数> --context=<显示行数>或-<显示行数> #除了显示符合样式的那一行之外，并显示该行之前后的内容。

-d <动作> --directories=<动作> #当指定要查找的是目录而非文件时，必须使用这项参数，否则 grep 指令将回报信息并停止动作。

-e<范本样式> --regexp=<范本样式> #指定字符串做为查找文件内容的样式。

-E --extended-regexp #将样式为延伸的普通表示法来使用。

-f<规则文件> --file=<规则文件> #指定规则文件，其内容含有一个或多个规则样式，让 grep 查找符合规则条件的文件内容，格式为每行一个规则样式。

-F --fixed-regexp #将样式视为固定字符串的列表。

-G --basic-regexp #将样式视为普通的表示法来使用。

-h --no-filename #在显示符合样式的那一行之前，不标示该行所属的文件名称。

-H --with-filename #在显示符合样式的那一行之前，表示该行所属的文件名称。

-i --ignore-case #忽略字符大小写的差别。

-l --file-with-matches #列出文件内容符合指定的样式的文件名称。

```
-L    --files-without-match  #列出文件内容不符合指定的样式的文件名  
称。  
-n    --line-number      #在显示符合样式的那一行之前，标示出该行的列数编  
号。  
-q    --quiet 或--silent   #不显示任何信息。  
-r    --recursive       #此参数的效果和指定“-d recurse”参数相同。  
-s    --no-messages     #不显示错误信息。  
-v    --revert-match    #显示不包含匹配文本的所有行。  
-V    --version         #显示版本信息。  
-w    --word-regexp     #只显示全字符串符合的列。  
-x    --line-regexp     #只显示全行字符串符合的列。  
-y    #此参数的效果和指定“-i”参数相同。
```

4. 规则表达式：

grep 的规则表达式：

```
^  #锚定行的开始 如: '^grep' 匹配所有以 grep 开头的行。  
$  #锚定行的结束 如: 'grep$' 匹配所有以 grep 结尾的行。  
. #匹配一个非换行符的字符 如: 'gr.p' 匹配 gr 后接一个任意字符，然后是  
p。  
* #匹配零个或多个先前字符 如: '*grep' 匹配所有一个或多个空格后紧跟  
grep 的行。  
. #一起用代表任意字符。  
[] #匹配一个指定范围内的字符，如'[Gg]rep' 匹配 Grep 和 grep。  
[^] #匹配一个不在指定范围内的字符，如: '[^A-FH-Z]rep' 匹配不包含 A-R  
和 T-Z 的一个字母开头，紧跟 rep 的行。  
\(..\) #标记匹配字符，如'\\(love\\)'，love 被标记为 1。  
\<      #锚定单词的开始，如:'\\<grep' 匹配包含以 grep 开头的单词的  
行。  
\>      #锚定单词的结束，如'grep\\>' 匹配包含以 grep 结尾的单词的  
行。  
x\{m\} #重复字符 x, m 次，如: '0\{5\}' 匹配包含 5 个 o 的行。  
x\{m,\} #重复字符 x, 至少 m 次，如: 'o\{5,\}' 匹配至少有 5 个 o 的  
行。  
x\{m,n\} #重复字符 x, 至少 m 次，不多于 n 次，如: 'o\{5,10\}' 匹配 5--  
10 个 o 的行。  
\w    #匹配文字和数字字符，也就是[A-Za-z0-9]，如: 'G\\w*p' 匹配以 G 后  
跟零个或多个文字或数字字符，然后是 p。  
\W    #\\w 的反置形式，匹配一个或多个非单词字符，如点号句号等。  
\b    #单词锁定符，如: '\\b' 只匹配 grep。
```

POSIX 字符：

为了在不同国家的字符编码中保持一致，
POSIX(The Portable Operating System Interface)增加了特殊的字符类，如
[:alnum:]是[A-Za-z0-9]的另一个写法。要把它们放到[]号内才能成为正则表达

式，如[A-Za-z0-9]或[:alnum:]。在linux下的grep除fgrep外，都支持POSIX的字符类。

```
[:alnum:]      #文字数字字符
[:alpha:]      #文字字符
[:digit:]     #数字字符
[:graph:]     #非空字符（非空格、控制字符）
[:lower:]      #小写字符
[:cntrl:]     #控制字符
[:print:]      #非空字符（包括空格）
[:punct:]      #标点符号
[:space:]      #所有空白字符（新行，空格，制表符）
[:upper:]      #大写字符
[:xdigit:]    #十六进制数字（0-9, a-f, A-F）
```

5. 使用实例：

实例 1：查找指定进程

命令：

```
ps -ef|grep svn
```

输出：

```
[root@localhost ~]# ps -ef|grep svn
root 4943 1 0 Dec05 ? 00:00:00 svnserve -d -
r /opt/svndata/grape/
root 16867 16838 0 19:53 pts/0 00:00:00 grep svn
[root@localhost ~]#
```

说明：

第一条记录是查找出的进程；第二条结果是grep进程本身，并非真正要找的进程。

实例 2：查找指定进程个数

命令：

```
ps -ef|grep svn -c
ps -ef|grep -c svn
```

输出：

```
[root@localhost ~]# ps -ef|grep svn -c
2
[root@localhost ~]# ps -ef|grep -c svn
2
[root@localhost ~]#
```

实例 3：从文件中读取关键词进行搜索

命令：

```
cat test.txt | grep -f test2.txt
```

输出：

```
[root@localhost test]# cat test.txt
hnlinux
```

```
peida.cnblogs.com
ubuntu
ubuntu linux
redhat
Redhat
linuxmint
[root@localhost test]# cat test2.txt
linux
Redhat
[root@localhost test]# cat test.txt | grep -f test2.txt
hnlinux
ubuntu linux
Redhat
linuxmint
[root@localhost test]#
```

说明：

输出 test.txt 文件中含有从 test2.txt 文件中读取出的关键词的内容行

实例 3：从文件中读取关键词进行搜索 且显示行号

命令：

```
cat test.txt | grep -nf test2.txt
```

输出：

```
[root@localhost test]# cat test.txt
hnlinux
peida.cnblogs.com
ubuntu
ubuntu linux
redhat
Redhat
linuxmint
[root@localhost test]# cat test2.txt
linux
Redhat
[root@localhost test]# cat test.txt | grep -nf test2.txt
1:hnlinux
4:ubuntu linux
6:Redhat
7:linuxmint
[root@localhost test]#
```

说明：

输出 test.txt 文件中含有从 test2.txt 文件中读取出的关键词的内容行，并显示每一行的行号

实例 5：从文件中查找关键词

命令：

```
grep 'linux' test.txt
```

输出：

```
[root@localhost test]# grep 'linux' test.txt
hnlinux
ubuntu linux
linuxmint
[root@localhost test]# grep -n 'linux' test.txt
1:hnlinux
4:ubuntu linux
7:linuxmint
[root@localhost test]#
```

实例 6：从多个文件中查找关键词

命令：

```
grep 'linux' test.txt test2.txt
```

输出：

```
[root@localhost test]# grep -n 'linux' test.txt test2.txt
test.txt:1:hnlinux
test.txt:4:ubuntu linux
test.txt:7:linuxmint
test2.txt:1:linux
[root@localhost test]# grep 'linux' test.txt test2.txt
test.txt:hnlinux
test.txt:ubuntu linux
test.txt:linuxmint
test2.txt:linux
[root@localhost test]#
```

说明：

多文件时，输出查询到的信息内容行时，会把文件的命名在行最前面输出并且加上 ":" 作为标示符

实例 7：grep 不显示本身进程

命令：

```
ps aux|grep \[s]sh
ps aux | grep ssh | grep -v "grep"
```

输出：

```
[root@localhost test]# ps aux|grep ssh
root    2720  0.0  0.0  62656  1212 ?        Ss    Nov02   0:00 /usr/sbi
n/sshd
root    16834  0.0  0.0  88088  3288 ?        Ss    19:53   0:00 sshd: ro
ot@pts/0
root   16901  0.0  0.0  61180    764 pts/0  S+   20:31   0:00 grep ssh
[root@localhost test]# ps aux|grep \[s]sh
```

```
[root@localhost test]# ps aux|grep \[s]sh
root 2720 0.0 0.0 62656 1212 ? Ss Nov02 0:00 /usr/sbi
n/sshd
root 16834 0.0 0.0 88088 3288 ? Ss 19:53 0:00 sshd: ro
ot@pts/0
[root@localhost test]# ps aux | grep ssh | grep -v "grep"
root 2720 0.0 0.0 62656 1212 ? Ss Nov02 0:00 /usr/sbi
n/sshd
root 16834 0.0 0.0 88088 3288 ? Ss 19:53 0:00 sshd: ro
ot@pts/0
```

实例 8：找出已 u 开头的行内容

命令：

```
cat test.txt |grep ^u
```

输出：

```
[root@localhost test]# cat test.txt |grep ^u
ubuntu
ubuntu linux
[root@localhost test]#
```

实例 9：输出非 u 开头的行内容

命令：

```
cat test.txt |grep ^[^u]
```

输出：

```
[root@localhost test]# cat test.txt |grep ^[^u]
hnlinux
peida.cnblogs.com
redhat
Redhat
linuxmint
[root@localhost test]#
```

实例 10：输出以 hat 结尾的行内容

命令：

```
cat test.txt |grep hat$
```

输出：

```
[root@localhost test]# cat test.txt |grep hat$
redhat
Redhat
[root@localhost test]#
```

实例 11：

输出：

实例 12：显示包含 ed 或者 at 字符的内容行

命令：

```
cat test.txt |grep -E "ed|at"
```

输出：

```
[root@localhost test]# cat test.txt |grep -E "peida|com"
peida.cnblogs.com
[root@localhost test]# cat test.txt |grep -E "ed|at"
redhat
Redhat
[root@localhost test]#
```

实例 13：显示当前目录下面以.txt 结尾的文件中的所有包含每个字符串至少有 7 个连续小写字符的字符串的行

命令：

```
grep '[a-z]\{7\}' *.txt
```

输出：

```
[root@localhost test]# grep '[a-z]\{7\}' *.txt
test.txt:hnlinux
test.txt:peida.cnblogs.com
test.txt:linuxmint
[root@localhost test]#
```