序言

 本篇文章主要讲解了一些linux常用命令，主要讲解模式是，命令介绍、命令参数格式、命令参数、命令常用参数示例。由于linux命令较多，我还特意选了一些日常使用较为频繁的命令进行讲解，但还是免不了文章很长，建议大家收藏起来，用到的时候不会了再来阅读。当然学习linux命令最好的方法是学会使用linux自带的man手册，所有linux命令规范和使用细则都会在该手册中讲解的很清楚，我在书写的过程中也是参考该手册和日常使用情况。

如果还有哪些是大家希望再补充的命令，可以留言给我，我会持续更新该文章，同时也作为我学习和工作的手册。

**Linux初级指令**

**ls** ——List

**ls 介绍**

 这是我学Linux的第一个命令，相信也是很多人学习Linux的第一个命令。ls全称list.

List information about the FILEs (the current directory by default).  Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

列出有关文件的信息(默认为当前目录)。如果没有指定-cftuvSUX或——Sort，则按字母顺序排序。

 官方的说的很清楚，默认列出当前目录，所以可以列出其他目录或者路径下的文件信息或者目录信息。

eg：

$ls /etc/hosts  
/etc/hosts

 ls还可以列出指定目录下的文件列表。

image-20191221161638964

**ls 参数格式**

ls [OPTION]... [FILE]...

**ls命令参数**

-a  列出指定目录下的所有文件，包括隐藏文件

-c 使用最后一次更改文件状态以进行排序(-t)或长时间打印(-l)的时间

-h 与-l选项一起使用时，请使用单位后缀:Byte、Kilobyte、mete、gb、tb和Petabyte，以便使用以2为基数的大小将数字减少到3或更少

-l 长格式列表。(见下文)。如果输出到终端，则所有文件大小的总和将输出到长清单前面的一行中

-n 以数字形式显示用户和组id，而不是在长(-l)输出中转换为用户或组名。这个选项默认打开-l选项

-o 以长格式列出，但省略组id

-s 显示每个文件实际使用的文件系统块的数量，以512字节为单位，其中部分单元四舍五入为下一个整数值

-t 在按照字典顺序对操作数排序之前，先按修改的时间排序(最近修改的是first)

-u 使用最后一次访问的时间，而不是最后一次修改文件进行排序

**ls 用法示例：**

$ ls  
test  tmp  
$ ls -a  
.  ..  .bash\_history  .bash\_logout  .bash\_profile  .bashrc  test  tmp  
#可以看到通过ls -a 可以查看当前目录影藏的文件，Linux下(.)开头的文件是隐藏文件。  
$ ls -l  
total 4  
-rw-r--r-- 1 test hero    0 Dec 21 19:54 test  
drwxr-xr-x 2 test hero 4096 Dec 21 19:54 tmp  
$ ls -lh  
total 4.0K  
-rw-r--r-- 1 test hero    0 Dec 21 19:54 test  
drwxr-xr-x 2 test hero 4.0K Dec 21 19:54 tmp  
$ ls -ll  
total 4  
-rw-r--r-- 1 test hero    0 Dec 21 19:54 test  
drwxr-xr-x 2 test hero 4096 Dec 21 19:54 tmp  
$ ls -alh  
total 28K  
drwx------   3 test hero 4.0K Dec 21 19:54 .  
drwxr-xr-x. 19 root  root 4.0K Aug  1 10:41 ..  
-rw-------   1 test hero  226 Dec 21 19:54 .bash\_history  
-rw-r--r--   1 test hero   18 Aug  3  2016 .bash\_logout  
-rw-r--r--   1 test hero  193 Aug  3  2016 .bash\_profile  
-rw-r--r--   1 test hero  231 Aug  3  2016 .bashrc  
-rw-r--r--   1 test hero    0 Dec 21 19:54 test  
drwxr-xr-x   2 test hero 4.0K Dec 21 19:54 tmp  
$ ls -o  
total 4  
-rw-r--r-- 1 test    0 Dec 21 19:54 test  
drwxr-xr-x 2 test 4096 Dec 21 19:54 tmp  
$ ls -oh  
total 4.0K  
-rw-r--r-- 1 test    0 Dec 21 19:54 test  
drwxr-xr-x 2 test 4.0K Dec 21 19:54 tmp

**pwd** —— Print Working Directory

**pwd介绍**

 打印当前工作目录的完整路径名。(print name of current/working directory)

**参数格式**

pwd [OPTION]...

**pwd 用法展示**

[test@Mfate171193 /home/test] 20:06  
$ pwd  
/home/test

**touch**  (change file timestamps)

**touch介绍**

Update the access and modification times of each FILE to the current time.

A FILE argument that does not exist is created empty, unless -c or -h is supplied.

 将每个文件的访问和修改时间更新为当前时间。除非提供-c或-h，否则将不存在的FILE参数创建为空。

**touch参数格式**

touch [OPTION]... FILE...

**touch命令参数**

-a  或--time=atime或--time=access或--time=use 只更改存取时间。

-c  或--no-create 不建立任何文档。

-d 使用指定的日期时间，而非现在的时间。

-f 此参数将忽略不予处理，仅负责解决BSD版本touch指令的兼容性问题。

-m  或--time=mtime或--time=modify 只更改变动时间。

-r 把指定文档或目录的日期时间，统统设成和参考文档或目录的日期时间相同。

-t 使用指定的日期时间，而非现在的时间。

**用法示例**

#创建三个文件  
$ touch test1 test2 test3  
#不创建文档  
$ touch -c test5    
$ ls  
test1  test2  test3  
#可以看到只创建了test1、test2、test3 ， -c不建立任何文件  
$ touch -t 201911110000 test1  
$ stat test\*  
#stat命令可以查看文件的详细变更时间，可以test1文件的最后修改时间为201911110000，说明touch -t可以修改文件最后访问时间。这个参数还是很有用的，你可以把你最近访问的时间修改为一个很早的时间。可以做一些有趣的事情，哈哈。  
  File: ‘test1’  
  Size: 0             Blocks: 0          IO Block: 4096   regular empty file  
Device: fd01h/64769d    Inode: 360736      Links: 1  
Access: (0644/-rw-r--r--)  Uid: (14060/   localhost)   Gid: ( 1001/    hero)  
Access: 2019-11-11 00:00:00.000000000 +0800  
Modify: 2019-11-11 00:00:00.000000000 +0800  
Change: 2019-12-21 20:58:11.290761038 +0800  
 Birth: -  
  File: ‘test2’  
  Size: 0             Blocks: 0          IO Block: 4096   regular empty file  
Device: fd01h/64769d    Inode: 360738      Links: 1  
Access: (0644/-rw-r--r--)  Uid: (14060/   localhost)   Gid: ( 1001/    hero)  
Access: 2019-12-21 20:56:34.523761038 +0800  
Modify: 2019-12-21 20:56:34.523761038 +0800  
Change: 2019-12-21 20:56:34.523761038 +0800  
 Birth: -  
  File: ‘test3’  
  Size: 0             Blocks: 0          IO Block: 4096   regular empty file  
Device: fd01h/64769d    Inode: 360740      Links: 1  
Access: (0644/-rw-r--r--)  Uid: (14060/   localhost)   Gid: ( 1001/    hero)  
Access: 2019-12-21 20:56:34.523761038 +0800  
Modify: 2019-12-21 20:56:34.523761038 +0800  
Change: 2019-12-21 20:56:34.523761038 +0800  
 Birth: -

**cat&tac** (Concatenate FILE(s), or standard input, to standard output.)

**cat介绍**

Concatenate FILE(s), or standard input, to standard output.

将FILE或标准输入连接到标准输出。

**cat 参数格式**

cat [OPTION]... [FILE]...

**cat命令参数**

-A, --show-all      等价于 -vET

-b, --number-nonblank  对非空输出行编号

-e            等价于 -vE

-E, --show-ends     在每行结束处显示

-n, --number   对输出的所有行编号,由1开始对所有输出的行数编号

-s, --squeeze-blank 有连续两行以上的空白行，就代换为一行的空白行

-t            与 -vT 等价

-T, --show-tabs     将跳格字符显示为 ^I

-u            (被忽略)

-v, --show-nonprinting  使用 ^ 和 M- 引用，除了 LFD 和 TAB 之外

**cat常用参数示例**

$ cat test  #展示文件内容  
-A, --show-all      等价于 -vET  
-b, --number-nonblank  对非空输出行编号  
-e            等价于 -vE  
  
$ cat -n test  #展示文件内容并且展示行号  
     1    -A, --show-all      等价于 -vET  
     2    -b, --number-nonblank  对非空输出行编号  
     3    -e            等价于 -vE

tac命令与cat命令展示内容相反，不能带行号输出。

$ tac test  
-e            等价于 -vE  
-b, --number-nonblank  对非空输出行编号  
-A, --show-all      等价于 -vET

**mkdir —— Make Directory**

**mkdir介绍**

Create the DIRECTORY(ies), if they do not already exist.

如果目录不存在，则创建目录。

**mkdir参数格式**

mkdir [OPTION]... DIRECTORY...

**mkdir命令参数**

-m, --mode=模式，设定权限<模式> (类似 chmod)，而不是 rwxrwxrwx 减 umask

-p, --parents 可以是一个路径名称。此时若路径中的某些目录尚不存在,加上此选项后,系统将自动建立好那些尚不存在的目录,即一次可以建立多个目录;

-v, --verbose 每次创建新目录都显示信息

--help  显示此帮助信息并退出

--version 输出版本信息并退出

**mkdir常用参数示例**

#创建目录文件test  
$ mkdir test  
#连续创建  
$ mkdir -p test1/tmp  
$ ls  
test  test1  
#创建时置顶目录权限  
#tmp目录拥有可执行权限，Linux文件权限问题后期文章会详细讲到，记得关注我  
$ mkdir -pm 777 test2/tmp  
$ ls -lh  
total 12K  
drwxr-xr-x 2 localhost hero 4.0K Dec 21 21:39 test  
drwxr-xr-x 3 localhost hero 4.0K Dec 21 21:40 test1  
drwxr-xr-x 3 localhost hero 4.0K Dec 21 21:40 test2  
$ ls  
test  test1  test2  
#-v 参数可确定文件是否已经存在，如果不存在则会创建，并显示如下信息  
$ mkdir -v test  
mkdir: cannot create directory ‘test’: File exists  
  
$ mkdir -v test7  
mkdir: created directory ‘test7’

**cd** —— Change Directory

**cd介绍**

 切换当前目录至指定目录

**常用参数示例**

#打印当前目录到标准输出  
$ pwd  
/Users/localhost  
#切换到目录/  
$cd /

 cd命令没什么参数就是切换目录到指定路径下，较为简单，但是使用频率极高。

rm&rmdir  —— Remove Directory

**rm介绍**

The rm utility attempts to remove the non-directory type files specified on the command line.  If the permissions of the file do not permit writing, and the standard input device is a terminal, the user is prompted (on the standard error output) for confirmation.

rm实用程序尝试删除命令行上指定的非目录类型文件。如果文件的权限不允许写入，并且标准输入设备是终端，则会提示用户（在标准错误输出上）进行确认。

 rm命令使用时还是需要注意的，他的删除恢复比较麻烦，有些系统会自带-i参数，输入命令之后还有一个确认步骤，有些是直接删掉了，是真删掉，从内存抹掉那种（其实底层是让该文件指针不指向该文件的内存块，内存上的内容原则上是存在的，但是恢复会比较复杂，需要扫描整块内存块才能拿到内容）。不要轻易删掉你写的重要代码，hh。

**rm参数格式**

rm [-dfiPRrvW] file ...

**rm命令参数**

-f, --force  忽略不存在的文件，从不给出提示。

-i, --interactive 进行交互式删除

-r, -R, --recursive  指示rm将参数中列出的全部目录和子目录均递归地删除。

-d, --dir 删除空目录

**rm常用参数示例**

# 创建三个文件  
$ touch tmp.cc tmp.java tmp.py tmp.go  
#创建目录文件  
$ mkdir -p linux/test  
#查看文件是否创建成功  
$ ls  
linux    tmp.cc   tmp.go   tmp.java tmp.py  
#删除文件，并进行提示  
$ rm -i tmp.cc  
remove tmp.cc? y  
#强制删除  
$ rm  -f tmp.go  
#删除目录  
$ rm -f linux  #删除目录失败  
rm: linux: is a directory  
#循环删除目录下所有文件  
$ rm -rf linux  #删除目录成功，  
$ ls  
tmp.java tmp.py

 rmdir==rm -d  删除空目录

mv —— Move

**mv介绍**

In its first form, the mv utility renames the file named by the source operand to the destination path named by the target operand.  This form is assumed when the last operand does not name an already existing directory.

In its second form, mv moves each file named by a source operand to a destination file in the existing directory named by the directory operand.  The destination path for each operand is the pathname produced by the concatenation of the last operand, a slash, and the final pathname component of the named file.

总结下，就是移动目录或者文件到置顶目录下，同时具有重命名的功能。

**mv参数格式**

mv [-f | -i | -n] [-v] source target mv [-f | -i | -n] [-v] source ... directory

**mv命令参数**

-b ：若需覆盖文件，则覆盖前先行备份。

-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

-n：不要覆盖现有文件。（-n选项将覆盖以前的任何-f或-i选项。）

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

**mv常用参数示例**

##修改文件名  
$ touch tmp.cc  
  
$ ls  
tmp.cc  
  
$ mv tmp.cc tmp.java  
  
$ ls  
tmp.java  
#移动文件或者目录  
$ pwd  
/Users/localhost/test  
#移动文件并重命名  
$ mv /Users/localhost/logs/tmp.txt ./tmp.log   
  
$ ls /Users/localhost/logs/  
discover-client metabase        tesla  
  
$ ls ./  
tmp.java tmp.log  
#移动目录并重命名  
$ mv /Users/localhost/logs/tesla  ./tesla.ba   
  
$ ls  
tesla.ba tmp.java tmp.log

 这个命令在写makefile文件的时候用起来很舒服，可以把编译的结果移到指定目录并重命名。

cp ——  Copy

**cp介绍**

In the first synopsis form, the cp utility copies the contents of the source\_file to the target\_file.  In the second synopsis form, the contents of each named source\_file is copied to the destination target\_directory.  The names of the files themselves are not changed.  If cp detects an attempt to copy a file to itself, the copy will fail.

cp实用程序将source\_file的内容复制到target\_file。在第二个大纲格式中，每个命名的source\_file的内容都复制到目标target\_directory。文件本身的名称不会更改。如果cp检测到尝试将文件复制到自身的尝试，则复制将失败。

**cp参数格式**

cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source\_file target\_file cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source\_file ... target\_directory

**命令参数**

-a, --archive  等于-dR --preserve=all

--backup[=CONTROL  为每个已存在的目标文件创建备份

-b        类似--backup 但不接受参数

--copy-contents    在递归处理是复制特殊文件内容

-d        等于--no-dereference --preserve=links

-f, --force    如果目标文件无法打开则将其移除并重试(当 -n 选项

存在时则不需再选此项)

-i, --interactive    覆盖前询问(使前面的 -n 选项失效)

-H        跟随源文件中的命令行符号链接

-l, --link      链接文件而不复制

-L, --dereference  总是跟随符号链接

-n, --no-clobber  不要覆盖已存在的文件(使前面的 -i 选项失效)

-P, --no-dereference  不跟随源文件中的符号链接

-p        等于--preserve=模式,所有权,时间戳

--preserve[=属性列表  保持指定的属性(默认：模式,所有权,时间戳)，如果

可能保持附加属性：环境、链接、xattr 等

-R, -r, --recursive 复制目录及目录内的所有项目

**常用参数示例**

$ cat tmp.cc  
change world  
  
#拷贝文件内容  
$ cp tmp.cc tmp.java  
  
$ cat tmp.java  
change world

echo

**echo介绍**

The echo utility writes any specified operands, separated by single blank (') characters and followed by a newline (\n') character, to the standard output.

echo实用程序将任何指定的操作数写入标准输出，这些操作数由单个空格（）字符分隔，后跟换行符（\ n'）字符。

这条命令较为简单，常用来打印变量、文本内容到，例如:

$ echo "change world"  
change world  
  
#s输出PWD环境变量的值  
$ echo $PWD  
/Users/localhost/test

$PWD 是取当前路径，然后echo到标准输出，一般echo \$name 用来查看某个环境变量的值

head&tail

**head介绍**

This filter displays the first count lines or bytes of each of the specified files, or of the standard input if no files are specified.  If count is omitted it defaults to 10.

此过滤器显示每个指定文件或标准输入（如果未指定文件）的前几行或字节。

If more than a single file is specified, each file is preceded by a header consisting of the string ==> XXX <=='' whereXXX'' is the name of the file.

如果省略count，则默认为10.如果指定了多个文件，则每个文件的头均由字符串==> XXX <==''组成，其中XXX''为文件名 文件。

**head参数格式**

head [-n count | -c bytes] [file ...]

**head常用参数示例**

-n 展示前n行

-c 展示前n个字符

**head常用参数示例**

$ cat -n test.txt #这就用上前面的cat命令的-n参数，要学会学以致用哦。  
     1    用  
     2    心  
     3    分  
     4    享，  
     5    共  
     6    同  
     7    成  
     8    长.  
     9  
    10    没  
    11    有  
    12    什  
    13    么  
    14    比  
    15    你  
    16    每  
    17    天  
    18    进  
    19    步  
    20    一  
    21    点  
    22    更  
    23    实  
    24    在  
    25    了  
    26    .  
  
$ head test.txt  #默认展示10行  
用  
心  
分  
享，  
共  
同  
成  
长.  
  
没  
  
$ head -n15 test.txt #展示15行  
用  
心  
分  
享，  
共  
同  
成  
长.  
  
没  
有  
什  
么  
比  
你  
  
$ head -c23 test.txt #展示前23个字符，中文一个汉字并非一个字符的(utf-8编码中文字符长度是可变的)  
用  
心  
分  
享，  
共

tail命令完全和他相反，是从尾部开始展示文本，但是他的参数用法更多。

-f 循环读取

-q 不显示处理信息

-v 显示详细的处理信息

-c<数目> 显示的字节数

-n<行数> 显示行数

--pid=PID 与-f合用,表示在进程ID,PID死掉之后结束.

-q, --quiet, --silent 从不输出给出文件名的首部

-s, --sleep-interval=S 与-f合用,表示在每次反复的间隔休眠S秒

tail和head命令经常用来查看日志，像我现在基本每天都会用，我的日志文件没办法用vim或者cat这样去看（因为我负责的业务日志量每天都是几十个G），要么用tail，要么用more&less(下面会讲)。

more&less

**more介绍**

Less  is  a  program similar to more (1), but which allows backward movement in the file as well as forward movement.  Also, less does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like vi (1).  Less uses termcap (or terminfo on some systems), so it can run on a variety of terminals.  There is even limited support for hardcopy terminals.  (On a hardcopy terminal, lines which should be printed  at the top of the screen are prefixed with a caret.)

more每次打开文件不是全部把文件读入内存而是流式读取，不会因为vi|vim某个大文件而造成系统oom。

more&less最重要的一点就是流式读取，支持翻页，像cat命令是全部读取输出到标准输出，如果文件太大会把屏幕刷满的，根本没办法看。

**more参数格式**

more [-dlfpcsu ] [-num ] [+/ pattern] [+ linenum] [file ... ]

**more命令参数**

+n   从笫n行开始显示

-n    定义屏幕大小为n行

+/pattern 在每个档案显示前搜寻该字串（pattern），然后从该字串前两行之后开始显示

-c    从顶部清屏，然后显示

-d    提示“Press space to continue，’q’ to quit（按空格键继续，按q键退出）”，禁用响铃功能

-l    忽略Ctrl+l（换页）字符

-p    通过清除窗口而不是滚屏来对文件进行换页，与-c选项相似

-s    把连续的多个空行显示为一行

-u    把文件内容中的下画线去掉

**less 与 more 类似，但使用 less 可以随意浏览文件，而 more 仅能向前移动，却不能向后移动，而且 less 在查看之前不会加载整个文件**

wc

**wc介绍**

The wc utility displays the number of lines, words, and bytes contained in each input file, or standard input (if no file is specified) to the standard output.  A line is defined as a string of characters delimited by acharac-ter.  Characters beyond the finalcharacter will not be included in the line count.

wc实用程序显示每个输入文件或标准输入（如果未指定文件）中每个输入文件中包含的行数，字数和字节数。一行定义为由字符分隔的字符串。最后一个字符之后的字符将不包括在行数中。

这条命令对我来说还是比较深刻的，我刚学习编程不久的时候，我感觉我写的代码很多了，那个时候我就很想知道我写了多少行代码了，一时兴起，说干就干，直接写了个程序去统计了一把，写完之后还感觉自己蛮厉害的，谁知道之后学习到这个wc，然后就觉得自己还是嫩了点，还是要多学习。

**wc参数格式**

wc [-clmw] [file ...]

**wc命令参数**

-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。

**常用参数示例**

$ ls  
test.txt  
  
$ wc test.txt  
      26      26     103 test.txt  
  
$ wc -l test.txt  #直接一把统计行数  
      26 test.txt  
  
$ wc -c test.txt  
     103 test.txt

date & cal

**date介绍**

When invoked without arguments, the date utility displays the current date and time.  Otherwise, depending on the options specified, date will set the date and time or print it in a user-defined way.

The date utility displays the date and time read from the kernel clock.  When used to set the date and time, both the kernel clock and the hardware clock are updated.

Only the superuser may set the date, and if the system securelevel (see securelevel(7)) is greater than 1, the time may not be changed by more than 1 second.

当不带参数调用时，date实用程序将显示当前日期和时间。否则，根据指定的选项，日期将设置日期和时间或以用户定义的方式打印日期和时间。

date实用程序显示从内核时钟读取的日期和时间。当用于设置日期和时间时，内核时钟和硬件时钟都将更新。

只有超级用户可以设置日期，并且如果系统安全级别（请参阅securelevel（7））大于1，则时间更改不得超过1秒。

**date参数格式**

date [-jRu] [-r seconds | filename] [-v [+|-]val[ymwdHMS]] ... [+output\_fmt] date [-jnu] [[[mm]dd]HH]MM[[cc]yy][.ss] date [-jnRu] -f input\_fmt new\_date [+output\_fmt] date [-d dst] [-t minutes\_west]

**date命令参数**

%H 小时(以00-23来表示)。

%I 小时(以01-12来表示)。

%K 小时(以0-23来表示)。

%l 小时(以0-12来表示)。

%M 分钟(以00-59来表示)。

%P AM或PM。

%r 时间(含时分秒，小时以12小时AM/PM来表示)。

%s 总秒数。起算时间为1970-01-01 00:00:00 UTC。

%S 秒(以本地的惯用法来表示)。

%T 时间(含时分秒，小时以24小时制来表示)。

%X 时间(以本地的惯用法来表示)。

%Z 市区。

%a 星期的缩写。

%A 星期的完整名称。

%b 月份英文名的缩写。

%B 月份的完整英文名称。

%c 日期与时间。只输入date指令也会显示同样的结果。

%d 日期(以01-31来表示)。

%D 日期(含年月日)。

%j 该年中的第几天。

%m 月份(以01-12来表示)。

%U 该年中的周数。

%w 该周的天数，0代表周日，1代表周一，异词类推。

%x 日期(以本地的惯用法来表示)。

%y 年份(以00-99来表示)。

%Y 年份(以四位数来表示)。

%n 在显示时，插入新的一行。

%t 在显示时，插入tab。

MM 月份(必要)

DD 日期(必要)

hh 小时(必要)

mm 分钟(必要)

ss 秒(选择性)

-d<字符串> 显示字符串所指的日期与时间。字符串前后必须加上双引号。

-s<字符串> 根据字符串来设置日期与时间。字符串前后必须加上双引号。

-u 显示GMT。

**常用参数示例**

$ date  
2019年12月22日 星期日 21时28分29秒 CST  
  
$ date '+%c'  
日 12/22 21:28:33 2019  
  
$  date '+%D'  
12/22/19  
  
$  date '+%T'  
21:28:50  
  
$ date '+%X'  
21时29分00秒

**cal介绍**

cal命令可以用来显示公历（阳历）日历。

**cal参数格式**

cal [-3hjy] [-A number] [-B number] [[month] year] cal [-3hj] [-A number] [-B number] -m month [year] ncal [-3hjJpwy] [-A number] [-B number] [-s country\_code] [[month] year] ncal [-3hJeo] [-A number] [-B number] [year] ncal [-CN] [-H yyyy-mm-dd] [-d yyyy-mm]

**cal命令参数**

-1 显示一个月的月历

-3 显示系统前一个月，当前月，下一个月的月历

-s  显示星期天为一个星期的第一天，默认的格式

-m 显示星期一为一个星期的第一天  
-j  显示在当年中的第几天（一年日期按天算，从1月1号算起，默认显示当前月在一年中的天数）  
-y  显示当前年份的日历

**cal常用参数示例**

$ cal  
      十二月 2019  
日 一 二 三 四 五 六  
 1  2  3  4  5  6  7  
 8  9 10 11 12 13 14  
15 16 17 18 19 20 21  
22 23 24 25 26 27 28  
29 30 31  
  
  
$ ncal  
    十二月 2019  
一      2  9 16 23 30  
二      3 10 17 24 31  
三      4 11 18 25  
四      5 12 19 26  
五      6 13 20 27  
六      7 14 21 28  
日   1  8 15 22 29

which

**which介绍**

The which utility takes a list of command names and searches the path for each executable file that would be run had these commands actually been invoked.

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。

**which参数格式**

-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。

**which常用参数示例**

$ which ls  
alias ls='ls --color=auto'  
    /bin/ls  
  
$ which which  
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'  
    /bin/alias  
    /usr/bin/which

whereis

**whereis介绍**

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。

**whereis参数格式**

whereis [program ...]

**whereis命令参数**

-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。

**whereis常用参数示例**

$ whereis ls  
/bin/ls  
  
$ whereis whereis  
/usr/bin/whereis

nl

**nl介绍**

nl命令在linux系统中用来计算文件中行号。nl 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， nl 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。

**nl参数格式**

nl [-p] [-b type] [-d delim] [-f type] [-h type] [-i incr] [-l num] [-n format] [-s sep] [-v startnum] [-w width] [file]

**nl命令参数**

-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。

**nl常用参数示例**

$ nl test.txt  
     1    用  
     2    心  
     3    分  
     4    享，  
     5    共  
     6    同  
     7    成  
     8    长.  
  
     9    没  
    10    有  
    11    什  
    12    么  
    13    比  
    14    你  
    15    每  
    16    天  
    17    进  
    18    步  
    19    一  
    20    点  
    21    更  
    22    实  
    23    在  
    24    了  
    25    .

ps

**ps介绍**

The ps utility displays a header line, followed by lines containing information about all of your processes that have controlling terminals.

ps实用程序显示标题行，其后是包含有关具有控制终端的所有进程的信息的行。

**ps参数格式**

ps [-AaCcEefhjlMmrSTvwXx] [-O fmt | -o fmt] [-G gid[,gid...]] [-g grp[,grp...]] [-u uid[,uid...]] [-p pid[,pid...]] [-t tty[,tty...]] [-U user[,user...]] ps [-L]

**ps命令参数**

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

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

**ps常用参数示例**

#查看所有进程  
$ps -a  
#查看进程的环境变量和程序间的关系  
$ps -ef

kill&killall

**kill介绍**

The  command kill sends the specified signal to the specified process or process group.  If no signal is specified, the TERM signal is sent.  The TERM signal will kill processes which do not catch this signal.  For other processes, it may be necessary to use the KILL (9) signal, since this signal cannot be caught.

命令kill将指定的信号发送到指定的进程或进程组。如果未指定信号，则发送TERM信号。TERM信号将杀死不捕获该信号的进程。对于其他过程，可能需要使用KILL（9）信号，因为无法捕获该信号。

**kill参数格式**

kill [-s signal|-p] [-q sigval] [-a] [--] pid...  
kill -l [signal]

**kill命令参数**

-l 信号，若果不加信号的编号参数，则使用“-l”参数会列出全部的信号名称

-a 当处理当前进程时，不限制命令名和进程号的对应关系

-p 指定kill 命令只打印相关进程的进程号，而不发送任何信号

-s 指定发送信号

-u 指定用户

**kill常用参数示例**

#查看当前系统信号  
$ kill -l  
 1) SIGHUP     2) SIGINT   3) SIGQUIT  4) SIGILL   5) SIGTRAP  
 6) SIGABRT     7) SIGBUS   8) SIGFPE   9) SIGKILL 10) SIGUSR1  
11) SIGSEGV    12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM  
16) SIGSTKFLT    17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP  
21) SIGTTIN    22) SIGTTOU 23) SIGURG  24) SIGXCPU 25) SIGXFSZ  
26) SIGVTALRM    27) SIGPROF 28) SIGWINCH    29) SIGIO   30) SIGPWR  
31) SIGSYS    34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  
38) SIGRTMIN+4    39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8  
43) SIGRTMIN+9    44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13  
48) SIGRTMIN+14    49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12  
53) SIGRTMAX-11    54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  
58) SIGRTMAX-6    59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2  
63) SIGRTMAX-1    64) SIGRTMAX

**解释**

HUP    1    终端断线  
INT     2    中断（同 Ctrl + C）  
QUIT    3    退出（同 Ctrl + \）  
TERM   15    终止  
KILL    9    强制终止  
CONT   18    继续（与STOP相反， fg/bg命令）  
STOP    19    暂停（同 Ctrl + Z）

kill -9 是我们使用的最多的信号，其实这种方式一点也不优雅，应该使用kill -15信号，大部分程序接收到SIGTERM信号后，会先释放自己的资源，然后再停止。但是也有程序可能接收信号后，做一些其他的事情（如果程序正在等待IO，可能就不会立马做出响应，等到io完成后在结束），也就是说，SIGTERM多半是会被阻塞的。

**讲到这里，linux基本指令差不多了够用了，差不多可以上你的服务器上随便搞搞了。但是也只是简单看看，更多的操作指令请看进阶部分。**

**Linux进阶指令**

find

**find介绍**

The find utility recursively descends the directory tree for each path listed, evaluating an expression (composed of the primaries'' andoperands'' listed below) in terms of each file in the tree.

find实用程序对列出的每个路径递归地遍历目录树，根据树中的每个文件计算表达式(由下面列出的“初选”和“操作数”组成)。

**这个命令使用频率极高，如果对这个命令了解很透彻，在日常工作中可以事半功倍。这个命令的参数较多，常用的参数我会在下面常用参数示例讲清楚**

**find参数格式**

find [-H | -L | -P] [-EXdsx] [-f path] path ... [expression] find [-H | -L | -P] [-EXdsx] -f path [path ...] [expression]

**find命令参数**

-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。

-name  按照文件名查找文件。

-perm  按照文件权限来查找文件。

-prune 使用这一选项可以使find命令不在当前指定的目录中查找，如果同时使用-depth选项，那么-prune将被find命令忽略。

-user  按照文件属主来查找文件。

-group 按照文件所属的组来查找文件。

-mtime -n +n 按照文件的更改时间来查找文件， - n表示文件更改时间距现在n天以内，+ n表示文件更改时间距现在n天以前。find命令还有-atime和-ctime 选项，但它们都和-m time选项。

-nogroup 查找无有效所属组的文件，即该文件所属的组在/etc/groups中不存在。

-nouser  查找无有效属主的文件，即该文件的属主在/etc/passwd中不存在。

-newer file1 ! file2 查找更改时间比文件file1新但比文件file2旧的文件。

-type 查找某一类型的文件，诸如：

* b - 块设备文件。
* d - 目录。
* c - 字符设备文件。
* p - 管道文件。
* l - 符号链接文件。
* f - 普通文件。

-size n：[c] 查找文件长度为n块的文件，带有c时表示文件长度以字节计。-depth：在查找文件时，首先查找当前目录中的文件，然后再在其子目录中查找。

-fstype：查找位于某一类型文件系统中的文件，这些文件系统类型通常可以在配置文件/etc/fstab中找到，该配置文件中包含了本系统中有关文件系统的信息。

-mount：在查找文件时不跨越文件系统mount点。

-follow：如果find命令遇到符号链接文件，就跟踪至链接所指向的文件。

-cpio：对匹配的文件使用cpio命令，将这些文件备份到磁带设备中。

另外,下面三个的区别:

-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常用参数示例**

**-name参数常用参数示例** 查找/user目录下所有以.log结尾的文件

image-20191225235202419

**-exec 参数后面跟的是command，它的终止是以;为结束标志的，所以这句命令后面的分号是不可缺少的，考虑到各个系统中分号会有不同的意义，所以前面加反斜杠。**

举一个我在工作中经常用到的例子，我有个日志目录，我系统的所有日志都会打到这个目录，目录的日志文件命名很随意，我没办法说根据名字删除，于是我想到用日期的方式删除，保存一个月的日志即可。

$find /home/midou/logs// -mtime +30 -name "\*.log.gz" -exec rm -rf {} \;  
# {} 这个是语法不能丢了 ，还有结尾的 ； 也不能丢了。

其实我把这个命令放在我的一个系统crontab文件里面，每天执行一次，这样我的日志目录就不用了手动清理。corntab使用详解在后面的命令中会讲到。

 -exec 后面可以接任何命令，你可以灵活运用，再结合到前面的-name参数，可以玩出花来。

grep

**grep介绍**

The grep utility searches any given input files, selecting lines that match one or more patterns.  By default, a pattern matches an input line if the regular expression (RE) in the pattern matches the input line without its trailing newline.  An empty expression matches every line.  Each input line that matches at least one of the patterns is written to the standard output

grep实用程序搜索任何给定的输入文件，选择与一个或多个模式匹配的行。默认情况下，如果模式中的正则表达式（RE）匹配输入行而没有尾随换行符，则该模式会匹配输入行。空表达式匹配每行。与至少一种模式匹配的每条输入线均写入标准输出

 这是个我每天都会用到的命令，我是做基础服务的，用我服务的人不免会遇到问题，这时候我就去要去看日志了，日志都是G级别的，当然不能用vim打开去搜索，会把系统挂掉，vim是全部文档加载到内存。这时候就需要使用grep命令去根据一些关键信息匹配查找了。（当然有些同学可能会说，既然经常查日志的话，就不能把日志接入到ElasticSearch这种可搜索的组建中，很好，用技术去解决实际问题。我们也是这样做的，但总免不了还是会去服务器上查一下日志，学会这个命令没错的）

**grep参数格式**

grep [-abcdDEFGHhIiJLlmnOopqRSsUVvwxZ] [-A num] [-B num] [-C[num]] [-e pattern] [-f file] [--binary-files=value] [--color[=when]] [--colour[=when]][--context[=num]] [--label] [--line-buffered] [--null] [pattern] [file ...]

**grep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  #只显示全字符合的列。

-x  --line-regexp  #只显示全列符合的列。

-y  此参数的效果和指定“-i”参数相同。

**grep常用参数示例**

$grep '20:[1-5][0-9]:' \*.log  #匹配当前目录下搜索log日志中，20点的日志  
$grep '20:[1-5][0-9]' 1.log 2.log 3.log  #指定在这三个文件中查找  
#grep规则是支持正则表达式的  
$ps -ef|grep java    #查找所有java进程  
$ps -ef|grep java    #-c可以统计查找的个数  
$grep '20:[1-5][0-9]:' \*.log | grep -v '20:[3-4][0-9]:'   # -v反向选择，相当于过滤  
$grep 'ab|bc' \*.log  #支持|语法，匹配含有ab或者bc的文本行

 掌握grep的常用参数，会让你查找日志或者内容非常轻松。特别是当你数据量很大的时候，没办法使用vi或者vim打开的情况下。

cut

**cut介绍**

 这个命令对我来说很有故事，当时我在学完C语言，简单了解了些linux上的基本命令，有次实习面试，面试官问我linux系统用过么，我斩钉截铁的说用过，面试官简单的用vim写了一行hello world。对我说，你怎样通过linux命令吧这个文本里面的hello world搞成十行，并且取出每一列的第七个字符。

**当时的我真的是心里一群草泥马跑过，这可难道我了，我沉思了片刻，说只要十行么？多点行么？。当然不行，只要十行，取每行的第七个字符续**沉思了片刻，拿起面试官的电脑就是一顿操作，于是有了我记忆深刻的下面这一行命令。

$ cat tmp.cc| >>tmp.cc|>>tmp.cc|>>tmp.cc|head -n10|>tmp.cc|cut -c7-7  
w  
w  
w  
w  
w  
w  
w  
w  
w  
w

 面试官一看，微微点了下头(肯定在心里默念，小伙子，不错啊)。所以这个命令我印象蛮深刻的。当然还要其他方式实现面试官的要求，比如cut第七个地府，写个循环重定向到管道，再到原文件。

The cut utility cuts out selected portions of each line (as specified by list) from each file and writes them to the standard output.  If no file arguments are specified, or a file argument is a single dash (`-'), cut reads from the standard input.  The items specified by list can be in terms of column position or in terms of fields delimited by a special character.  Column numbering starts from 1.

cut实用程序从每个文件中剪切出每行的选定部分（由列表指定），并将它们写入标准输出。如果未指定文件参数，或者文件参数为单破折号（-），则从标准输入中读取内容。列表指定的项目可以是列位置，也可以是由特殊字符分隔的字段。列编号从1开始。

**cut参数格式**

cut -b list [-n] [file ...] cut -c list [file ...] cut -f list [-d delim] [-s] [file ...]

**cut命令参数**

-b：仅显示行中指定直接范围的内容；

-c：仅显示行中指定范围的字符；

-d：指定字段的分隔符，默认的字段分隔符为“TAB”；

-f：显示指定字段的内容；

-n：与“-b”选项连用，不分割多字节字符；

--complement：补足被选择的字节、字符或字段；

--out-delimiter=<字段分隔符>：指定输出内容是的字段分割符；

**cut常用参数示例**

$cut -c-10 tmp.txt  #cut tmp.txt文件的前10列  
$cut -c3-5 tmp.txt  #cut tmp.txt文件的第3到5列  
$cut -c3- tmp.txt  #cut tmp.txt文件的第3到结尾列

diff

**diff介绍**

Compare files line by line.

比较两个文件的不同

**diff参数格式**

diff [OPTION]... FILES

**diff命令参数**

-b或--ignore-space-change 不检查空格字符的不同。

-B或--ignore-blank-lines 不检查空白行。

-c 显示全部内文，并标出不同之处。

-C或--context 与执行"-c-"指令相同。

-d或--minimal 使用不同的演算法，以较小的单位来做比较。

-D或ifdef 此参数的输出格式可用于前置处理器巨集。

-e或--ed 此参数的输出格式可用于ed的script文件。

-f或-forward-ed 输出的格式类似ed的script文件，但按照原来文件的顺序来显示不同处。

-H或--speed-large-files 比较大文件时，可加快速度。

-l或--ignore-matching-lines 若两个文件在某几行有所不同，而这几行同时都包含了选项中指定的字符或字符串，则不显示这两个文件的差异。

-i或--ignore-case 不检查大小写的不同。

-l或--paginate 将结果交由pr程序来分页。

-n或--rcs 将比较结果以RCS的格式来显示。

-N或--new-file 在比较目录时，若文件A仅出现在某个目录中，预设会显示：Only in目录：文件A若使用-N参数，则diff会将文件A与一个空白的文件比较。

-p 若比较的文件为C语言的程序码文件时，显示差异所在的函数名称。

-P或--unidirectional-new-file 与-N类似，但只有当第二个目录包含了一个第一个目录所没有的文件时，才会将这个文件与空白的文件做比较。

-q或--brief 仅显示有无差异，不显示详细的信息。

-r或--recursive 比较子目录中的文件。

-s或--report-identical-files 若没有发现任何差异，仍然显示信息。

-S或--starting-file 在比较目录时，从指定的文件开始比较。

-t或--expand-tabs 在输出时，将tab字符展开。

-T或--initial-tab 在每行前面加上tab字符以便对齐。

-u,-U或--unified= 以合并的方式来显示文件内容的不同。

-v或--version 显示版本信息。

-w或--ignore-all-space 忽略全部的空格字符。

-W或--width 在使用-y参数时，指定栏宽。

-x或--exclude 不比较选项中所指定的文件或目录。

-X或--exclude-from 您可以将文件或目录类型存成文本文件，然后在=中指定此文本文件。

-y或--side-by-side 以并列的方式显示文件的异同之处。

**diff常用参数示例**

$ diff testA.txt testB.txt

tar&gzip

**tar介绍**

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

**tar参数格式**

     tar [bundled-flags <args>] [<file> | <pattern> ...]  
     tar {-c} [options] [files | directories]  
     tar {-r | -u} -f archive-file [options] [files | directories]  
     tar {-t | -x} [options] [patterns]

**tar命令参数**

-A 新增压缩文件到已存在的压缩

-B 设置区块大小

-c 建立新的压缩文件

-d 记录文件的差别

-r 添加文件到已经压缩的文件

-u 添加改变了和现有的文件到已经存在的压缩文件

-x 从压缩的文件中提取文件

-t 显示压缩文件的内容

-z 支持gzip解压文件

-j 支持bzip2解压文件

-Z 支持compress解压文件

-v 显示操作过程

-l 文件系统边界设置

-k 保留原有文件不覆盖

-m 保留文件不被覆盖

-W 确认压缩文件的正确性

-b 设置区块数目

-C 切换到指定目录

-f 指定压缩文件

**tar常用参数示例**

#打包  tar -cvf 包名  文件名  
$tar -cvf test.tar test.txt   
#解包  tar -xvf 包名   
$tar -xvf test.tar  
#压缩  tar -czvf 包名 文件名  
$tar -czvf test.tgz test.txt  
#解压  tar -xzvf 包名  
$tar -xzvf test.tgz

du

**du介绍**

The du utility displays the file system block usage for each file argument and for each directory in the file hierarchy rooted in each directory argument.If no file is specified, the block usage of the hierarchy rooted in the current directory is displayed.

du实用程序显示每个文件自变量以及以每个目录自变量为根的文件层次结构中每个目录的文件系统块使用情况。如果未指定文件，则显示以当前目录为根的层次结构的块使用情况。

**du参数格式**

du [-H | -L | -P] [-a | -s | -d depth] [-c] [-h | -k | -m | -g] [-x] [-I mask] [file ...]

**du命令参数**

-a或-all 显示目录中个别文件的大小。

-b或-bytes 显示目录或文件大小时，以byte为单位。

-c或--total 除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和。

-k或--kilobytes 以KB(1024bytes)为单位输出。

-m或--megabytes 以MB为单位输出。

-s或--summarize 仅显示总计，只列出最后加总的值。

-h或--human-readable 以K，M，G为单位，提高信息的可读性。

-x或--one-file-xystem 以一开始处理时的文件系统为准，若遇上其它不同的文件系统目录则略过。

-L<符号链接>或--dereference<符号链接> 显示选项中所指定符号链接的源文件大小。

-S或--separate-dirs  显示个别目录的大小时，并不含其子目录的大小。

-X<文件>或--exclude-from=<文件> 在<文件>指定目录或文件。

--exclude=<目录或文件>     略过指定的目录或文件。

-D或--dereference-args  显示指定符号链接的源文件大小。

-H或--si 与-h参数相同，但是K，M，G是以1000为换算单位。

-l或--count-links  重复计算硬件链接的文件。

**du常用参数示例**

#查看指定文件大小  
$du -h filename  
#展示该目录下所有文件大小，大小以可读方式展示  
$du  -h /  
#展示当前目录大小  
$du -sh  
#展示当前目录下每个目录大小  
$du -sh ./  
#显示所有文件的大小，以可读方式展示  
$du -ah /

df

**df介绍**

The df utility displays statistics about the amount of free disk space on the specified filesystem or on the filesystem of which file is a part.  Values are displayed in 512-byte per block counts.  If neither a file or a filesystem operand is specified, statistics for all mounted filesystems are displayed (sub-ject to the -t option below).

df实用程序显示有关指定文件系统或其中一部分文件的文件系统上的可用磁盘空间量的统计信息。值以每块计数512字节的形式显示。如果未指定文件或文件系统操作数，则将显示所有已挂载文件系统的统计信息（受下面的-t选项约束）。

**df参数格式**

df [-b | -h | -H | -k | -m | -g | -P] [-ailn] [-t] [-T type] [file | filesystem ...]

**df命令参数**

-a 全部文件系统列表

-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

-k 区块为1024字节

-l 只显示本地文件系统

-m 区块为1048576字节

--no-sync 忽略 sync 命令

-P 输出格式为POSIX

--sync 在取得磁盘信息前，先执行sync命令

-T 文件系统类型

--block-size=<区块大小> 指定区块大小

-t<文件系统类型> 只显示选定文件系统的磁盘信息

-x<文件系统类型> 不显示选定文件系统的磁盘信息

**df常用参数示例**

#展示当前系统磁盘使用情况，以可读的方式展示  
$ df -h  
Filesystem      Size   Used  Avail Capacity iused               ifree %iused  Mounted on  
/dev/disk1s1   234Gi   78Gi  151Gi    35% 1376436 9223372036853399371    0%   /  
devfs          208Ki  208Ki    0Bi   100%     720                   0  100%   /dev  
/dev/disk1s4   234Gi  4.0Gi  151Gi     3%       4 9223372036854775803    0%   /private/var/vm  
map -hosts       0Bi    0Bi    0Bi   100%       0                   0  100%   /net  
map auto\_home    0Bi    0Bi    0Bi   100%       0                   0  100%   /home  
/dev/disk3s1   290Mi  271Mi   17Mi    94%    2156 9223372036854773651    0%   /Volumes/Sourcetrail\_2019\_4\_102

lsof

**lsof介绍**

lsof（list open files）是一个列出当前系统打开文件的工具。(在linux环境下，任何事物都以文件的形式存在)

lsof可以打开的文件包括：

1.普通文件

2.目录

3.网络文件系统的文件

4.字符或设备文件

5.(函数)共享库

6.管道，命名管道

7.符号链接

8.网络文件（例如：NFS file、网络socket，unix域名socket）

9.还有其它类型的文件，等等

**这个命令在我日常工作中使用场景很多，使用范围很广。**

**lsof参数格式**

lsof [ -?abChKlnNOPRtUvVX ] [ -A A ] [ -c c ] [ +c c ] [ +|-d d ] [ +|-D D ] [ +|-e s ] [ +|-E ] [ +|-f [cfgGn] ] [ -F [f] ] [ -g [s] ] [ -i [i] ] [ -k k ] [ +|-L [l] ] [ +|-m m ] [ +|-M ] [ -o [o] ] [ -p s ] [ +|-r [t[m<fmt>]] ] [ -s [p:s] ] [ -S [t] ] [ -T [t] ] [ -u s ] [ +|-w ] [ -x [fl] ] [ -z [z] ] [ -Z [Z] ] [ -- ] [names]

**lsof命令参数**

-a 列出打开文件存在的进程

-c<进程名> 列出指定进程所打开的文件

-g 列出GID号进程详情

-d<文件号> 列出占用该文件号的进程

+d<目录> 列出目录下被打开的文件

+D<目录> 递归列出目录下被打开的文件

-n<目录> 列出使用NFS的文件

-i<条件> 列出符合条件的进程。（4、6、协议、:端口、 @ip ）

-p<进程号> 列出指定进程号所打开的文件

-u 列出UID号进程详情

**lsof常用参数示例**

**lsof输出各列信息的意义如下：**

COMMAND：进程的名称

PID：进程标识符

PPID：父进程标识符（需要指定-R参数）

USER：进程所有者

PGID：进程所属组

FD：文件描述符，应用程序通过文件描述符识别该文件。如cwd、txt等

#显示当前系统打开的文件  
$lsof    
#查看某个文件的相关进程  lsof 文件名  
$ lsof /bin/bash  
COMMAND  PID  USER  FD   TYPE DEVICE SIZE/OFF   NODE NAME  
bash    9430 midou txt    REG  253,1   960392 140072 /usr/bin/bash  
#查看某个用户打开的文件信息  
$lsof -u username  
#列出某个程序进程所打开的文件信息  
$lsof -c java  
#列出除了某个用户外的被打开的文件信息  
$lsof -u ^midou  
#通过某个进程号显示该进行打开的文件  
$lsof -p pid  
#列出除了某个进程号，其他进程号所打开的文件信息  
$lsof -p ^pid  
#列出所有的网络连接  
$lsof -i  
#列出所有tcp 网络连接信息  
$lsof -i tcp  
#列出所有udp网络连接信息  
$lsof -i udp  
#列出谁在某个端口使用情况  
$lsof -i :port  
#特定的tcp端口  
$lsof -i tcp:port   
#特定的udp端口  
$lsof -i udp:port  
#列出某个用户的所有活跃的网络端口  
$lsof -a -u username -i  
#根据文件描述符范围列出文件信息  
$lsof -d 0-2

**ping**

**ping介绍**

send ICMP ECHO\_REQUEST packets to network hosts

将ICMP ECHO\_REQUEST数据包发送到网络主机

**ping参数格式**

 ping [-AaCDdfnoQqRrv] [-b boundif] [-c count] [-G sweepmaxsize] [-g sweepminsize] [-h sweepincrsize] [-i wait] [-k trafficclass] [-K netservicetype][-l preload] [-M mask | time] [-m ttl] [-P policy] [-p pattern] [-S src\_addr] [-s packetsize] [-t timeout] [-W waittime] [-z tos] [--apple-connect][--apple-time] host

**ping命令参数**

-d 使用Socket的SO\_DEBUG功能。

-f 极限检测。大量且快速地送网络封包给一台机器，看它的回应。

-n 只输出数值。

-q 不显示任何传送封包的信息，只显示最后的结果。

-r 忽略普通的Routing Table，直接将数据包送到远端主机上。通常是查看本机的网络接口是否有问题。

-R 记录路由过程。

-v 详细显示指令的执行过程。

-c 数目：在发送指定数目的包后停止。

-i 秒数：设定间隔几秒送一个网络封包给一台机器，预设值是一秒送一次。

-I 网络界面：使用指定的网络界面送出数据包。

-l 前置载入：设置在送出要求信息之前，先行发出的数据包。

-p 范本样式：设置填满数据包的范本样式。

-s 字节数：指定发送的数据字节数，预设值是56，加上8字节的ICMP头，一共是64ICMP数据字节。

-t 存活数值：设置存活数值TTL的大小。

**ping，在日常工作中都是简单的用来测试本机与其他机器之间的网络通信，当然如果了解这些参数的话，会有更多的用法。**

**ping命令参数**

#检测网络情况  
$ping host  
#ping网关  
$ping -b host  
#ping指定次数  
$ping -c 10 host  
#ping指定时间间隔和次数限制  
$ping -c 10 -i 0.5 host  
#通过域名ping公网上的站点

netstat

**netstat介绍**

The netstat command symbolically displays the contents of various network-related data structures.  There are a number of output formats, depending on the options for the information presented.  The first form of the command displays a list of active sockets for each protocol.  The second form presents the contents of one of the other network data structures according to the option selected. Using the third form, with a wait interval specified, netstat will continuously display the information regarding packet traffic on the configured network interfaces.  The fourth form displays statistics for the specified protocol or address family. If a wait interval is specified, the protocol information over the last interval seconds will be displayed.  The fifth form displays per-interface statistics for the specified protocol or address family.  The sixth form displays mbuf(9) statistics.  The seventh form displays routing table for the speci-fied address family.  The eighth form displays routing statistics.

netstat命令以符号形式显示各种与网络相关的数据结构的内容。有多种输出格式，具体取决于显示信息的选项。该命令的第一种形式显示每个协议的活动套接字列表。第二种形式根据选择的选项显示其他网络数据结构之一的内容。使用第三种形式，并指定等待间隔，netstat将在配置的网络接口上连续显示有关数据包流量的信息。第四种形式显示指定协议或地址族的统计信息。如果指定了等待间隔，将显示最近间隔秒的协议信息。第五种形式显示指定协议或地址族的每个接口的统计信息。第六种形式显示mbuf（9）统计信息。第七种形式显示指定地址系列的路由表。第八种形式显示路由统计信息。

**netstat参数格式**

     netstat [-AaLlnW] [-f address\_family | -p protocol]  
     netstat [-gilns] [-v] [-f address\_family] [-I interface]  
     netstat -i | -I interface [-w wait] [-c queue] [-abdgqRtS]  
     netstat -s [-s] [-f address\_family | -p protocol] [-w wait]  
     netstat -i | -I interface -s [-f address\_family | -p protocol]  
     netstat -m [-m]  
     netstat -r [-Aaln] [-f address\_family]  
     netstat -rs [-s]

**netstat命令参数**

-a或–all 显示所有连线中的Socket。

-A<网络类型>或–<网络类型> 列出该网络类型连线中的相关地址。

-c或–continuous 持续列出网络状态。

-C或–cache 显示路由器配置的快取信息。

-e或–extend 显示网络其他相关信息。

-F或–fib 显示FIB。

-g或–groups 显示多重广播功能群组组员名单。

-h或–help 在线帮助。

-i或–interfaces 显示网络界面信息表单。

-l或–listening 显示监控中的服务器的Socket。

-M或–masquerade 显示伪装的网络连线。

-n或–numeric 直接使用IP地址，而不通过域名服务器。

-N或–netlink或–symbolic 显示网络硬件外围设备的符号连接名称。

-o或–timers 显示计时器。

-p或–programs 显示正在使用Socket的程序识别码和程序名称。

-r或–route 显示Routing Table。

-s或–statistice 显示网络工作信息统计表。

-t或–tcp 显示TCP传输协议的连线状况。

-u或–udp 显示UDP传输协议的连线状况。

-v或–verbose 显示指令执行过程。

-V或–version 显示版本信息。

-w或–raw 显示RAW传输协议的连线状况。

-x或–unix 此参数的效果和指定”-A unix”参数相同。

–ip或–inet 此参数的效果和指定”-A inet”参数相同。

**netstat常用参数示例**

#列出所有端口使用情况  
$netstat -a  
#显示当前UDP连接状况  
$netstat -nu  
#显示UDP端口号的使用情况  
$netstat -apu  
#显示网卡列表  
$netstat -i  
#显示网络统计信息  
$netstat -s  
#显示监听的套接口  
$netstat -l  
#显示所有已建立的有效连接  
$netstat -n  
#显示关于路由表的信息  
$netstat -r  
#列出所有 tcp 端口  
$netstat -at  
#找出程序运行的端口  
$netstat -ap | grep ssh  
#在 netstat 输出中显示 PID 和进程名称  
$netstat -pt

ifconfig

**ifconfig介绍**

Ifconfig is used to configure the kernel-resident network interfaces.  It is used at boot time to set up interfaces as necessary.  After that, it is usually only needed when debugging or when system tuning is needed.

Ifconfig用于配置内核驻留的网络接口。它在引导时用于根据需要设置接口。之后，通常仅在调试或需要系统调整时才需要它。

**ifconfig参数格式**

 ifconfig [-v] [-a] [-s] [interface]  
 ifconfig [-v] interface [aftype] options | address ...

**ifconfig命令参数**

up 启动指定网络设备/网卡。

down 关闭指定网络设备/网卡。该参数可以有效地阻止通过指定接口的IP信息流，如果想永久地关闭一个接口，我们还需要从核心路由表中将该接口的路由信息全部删除。

arp 设置指定网卡是否支持ARP协议。

-promisc 设置是否支持网卡的promiscuous模式，如果选择此参数，网卡将接收网络中发给它所有的数据包

-allmulti 设置是否支持多播模式，如果选择此参数，网卡将接收网络中所有的多播数据包

-a 显示全部接口信息

-s 显示摘要信息（类似于 netstat -i）

add 给指定网卡配置IPv6地址

del 删除指定网卡的IPv6地址

<硬件地址> 配置网卡最大的传输单元

mtu<字节数> 设置网卡的最大传输单元 (bytes)

netmask<子网掩码> 设置网卡的子网掩码。掩码可以是有前缀0x的32位十六进制数，也可以是用点分开的4个十进制数。如果不打算将网络分成子网，可以不管这一选项；如果要使用子网，那么请记住，网络中每一个系统必须有相同子网掩码。

tunel 建立隧道

dstaddr 设定一个远端地址，建立点对点通信

-broadcast<地址> 为指定网卡设置广播协议

-pointtopoint<地址> 为网卡设置点对点通讯协议

multicast 为网卡设置组播标志

address 为网卡设置IPv4地址

txqueuelen<长度> 为网卡设置传输列队的长度

**ifconfig常用参数示例**

#显示网络设备信息  
$ifconfig  
#启动关闭指定网卡  
$ifconfig eth0 up  
$ifconfig eth0 down  
#配置IP地址  
$ifconfig eth0 ip  
#启用和关闭ARP协议  
$ifconfig eth0 arp  
$ifconfig eth0 -arp  
#设置最大传输单元  
$ifconfig eth0 mtu 1500

**解释**

* eth0 表示第一块网卡， 其中 HWaddr 表示网卡的物理地址
* inet addr 用来表示网卡的IP地址
* lo 是表示主机的回坏地址，这个一般是用来测试一个网络程序，但又不想让局域网或外网的用户能够查看，只能在此台主机上运行和查看所用的网络接口。

第一行：连接类型：Ethernet（以太网）HWaddr（硬件mac地址）

第二行：网卡的IP地址、子网、掩码

第三行：UP（代表网卡开启状态）RUNNING（代表网卡的网线被接上）MULTICAST（支持组播）MTU:1500（最大传输单元）：1500字节

第四、五行：接收、发送数据包情况统计

第七行：接收、发送数据字节数统计信息。

hostname

**hostname介绍**

Hostname is used to display the system's DNS name, and to display or set its hostname or NIS domain name.

主机名用于显示系统的DNS名称，并显示或设置其主机名或NIS域名。

**hostname参数格式**

hostname [-a|--alias] [-d|--domain] [-f|--fqdn|--long] [-A|--all-fqdns] [-i|--ip-address] [-I|--all-ip-addresses] [-s|--short] [-y|--yp|--nis]  
hostname [-b|--boot] [-F|--file filename] [hostname]  
hostname [-h|--help] [-V|--version]

**hostname命令参数**

-v：详细信息模式；  
-a：显示主机别名；  
-d：显示DNS域名；  
-f：显示FQDN名称；  
-i：显示主机的ip地址；  
-s：显示短主机名称，在第一个点处截断；  
-y：显示NIS域名。

**hostname常用参数示例**

#查看主机ip,这个命令我最推荐的一个用法就是查看主机ip，之前我一直用ifconfig  
$hostname -i

traceroute

**traceroute介绍**

traceroute  tracks  the  route  packets  taken  from an IP network on their way to a given host. It utilizes the IP protocol's time to live (TTL) field and attempts to elicit an ICMP TIME\_EXCEEDED response from each gateway along the path to the host.

traceroute跟踪从IP网络获取到给定主机的路由信息包。它利用IP协议的生存时间（TTL）字段并尝试从每个网关到主机的路径引发ICMP TIME\_EXCEEDED响应。

**traceroute参数格式**

 traceroute [-46dFITUnreAV] [-f first\_ttl] [-g gate,...]  
               [-i device] [-m max\_ttl] [-p port] [-s src\_addr]  
               [-q nqueries] [-N squeries] [-t tos]  
               [-l flow\_label] [-w waittime] [-z sendwait] [-UL] [-D]  
               [-P proto] [--sport=port] [-M method] [-O mod\_options]  
               [--mtu] [--back]  
               host [packet\_len]

**traceroute命令参数**

-d 使用Socket层级的排错功能。

-f 设置第一个检测数据包的存活数值TTL的大小。

-F 设置勿离断位。

-g 设置来源路由网关，最多可设置8个。

-i 使用指定的网络界面送出数据包。

-I 使用ICMP回应取代UDP资料信息。

-m 设置检测数据包的最大存活数值TTL的大小。

-n 直接使用IP地址而非主机名称。

-p 设置UDP传输协议的通信端口。

-r 忽略普通的Routing Table，直接将数据包送到远端主机上。

-s 设置本地主机送出数据包的IP地址。

-t 设置检测数据包的TOS数值。

-v 详细显示指令的执行过程。

-w 设置等待远端主机回报的时间。

-x 开启或关闭数据包的正确性检验。

**traceroute常用参数示例**

#traceroute 一下百度，看下数据包的路由途径  
$ traceroute www.baidu.com  
traceroute: Warning: www.baidu.com has multiple addresses; using 183.232.231.172  
traceroute to www.baidu.com (183.232.231.172), 64 hops max, 52 byte packets  
 1  192.168.0.1 (192.168.0.1)  6.059 ms  0.879 ms  0.843 ms  
 2  192.168.1.1 (192.168.1.1)  1.305 ms  2.232 ms  2.167 ms  
 3  10.104.0.1 (10.104.0.1)  5.085 ms  5.534 ms  4.466 ms  
 4  221.131.253.13 (221.131.253.13)  4.633 ms  11.736 ms  4.199 ms  
 5  117.148.181.1 (117.148.181.1)  4.544 ms \*  
    112.11.233.49 (112.11.233.49)  13.384 ms  
 6  221.183.47.165 (221.183.47.165)  6.591 ms  6.643 ms  
    221.183.47.161 (221.183.47.161)  5.591 ms  
 7  \* 221.183.40.225 (221.183.40.225)  27.242 ms  25.222 ms  
 8  221.183.59.154 (221.183.59.154)  27.937 ms  27.501 ms  26.869 ms  
 9  120.241.49.198 (120.241.49.198)  60.772 ms  
    120.241.49.30 (120.241.49.30)  33.451 ms  
    120.241.48.190 (120.241.48.190)  45.563 ms  
10  \* \* \*  
11  \* \* \*  
12  \* \* \*  
13  \* \* \*  
14  \* \* \*  
15  \* \* \*  
16  \* \* \*

**解释**

记录按序列号从1开始，每行纪录就是一跳 ，每跳表示一个网关，我们看到每行有三个时间，单位是 ms，其实就是-q的默认参数。探测数据包向每个网关发送三个数据包后，网关响应后返回的时间；如果您用 traceroute -q 10 www.baidu.com，表示向每个网关发送10个数据包。

有时我们traceroute 一台主机时，会看到有一些行是以星号表示的。出现这样的情况，可能是防火墙封掉了ICMP的返回信息，所以我们得不到什么相关的数据包返回数据。

route

**route介绍**

Route  manipulates the kernel's IP routing tables.  Its primary use is to set up static routes to specific hosts or networks via an interface after it has been config‐ured with the ifconfig(8) program.

Route操纵内核的IP路由表。它的主要用途是在使用ifconfig（8）程序对其进行配置后，通过接口设置到特定主机或网络的静态路由。

**route参数格式**

route [-CFvnNee] [-A family |-4|-6]  
  
route  [-v] [-A family |-4|-6] add [-net|-host] target [netmask Nm] [gw Gw] [metric N] [mss M] [window W] [irtt I] [reject] [mod] [dyn] [reinstate] [[dev] If]  
  
route  [-v] [-A family |-4|-6] del [-net|-host] target [gw Gw] [netmask Nm] [metric N] [[dev] If]

**route命令参数**

-c 显示更多信息

-n 不解析名字

-v 显示详细的处理信息

-F 显示发送信息

-C 显示路由缓存

-f 清除所有网关入口的路由表。

-p 与 add 命令一起使用时使路由具有永久性。

add:添加一条新路由。

del:删除一条路由。

-net:目标地址是一个网络。

-host:目标地址是一个主机。

**route常用参数示例**

#显示当前路由  
$route  
#屏蔽一条路由  
$route add -net 224.0.0.0 netmask 240.0.0.0 reject  
#删除路由记录  
$route del -net 224.0.0.0 netmask 240.0.0.0  
#删除和添加设置默认网关  
$route del default gw 192.168.0.100  
$route add default gw 192.168.0.100

wget

**wget介绍**

GNU Wget is a free utility for non-interactive download of files from the Web.  It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.

GNU Wget是一个免费实用程序，用于从Web非交互式下载文件。它支持HTTP，HTTPS和FTP协议，以及通过HTTP代理进行检索。

**wget参数格式**

wget [option]... [URL]...

**wget命令参数**

启动：  
  -V,  --version           显示 Wget 的版本信息并退出。  
  -h,  --help              打印此帮助。  
  -b,  --background        启动后转入后台。  
  -e,  --execute=COMMAND   运行一个“.wgetrc”风格的命令。  
  
日志和输入文件：  
  -o,  --output-file=FILE    将日志信息写入 FILE。  
  -a,  --append-output=FILE  将信息添加至 FILE。  
  -d,  --debug               打印大量调试信息。  
  -q,  --quiet               安静模式 (无信息输出)。  
  -v,  --verbose             详尽的输出 (此为默认值)。  
  -nv, --no-verbose          关闭详尽输出，但不进入安静模式。  
  -i,  --input-file=FILE     下载本地或外部 FILE 中的 URLs。  
  -F,  --force-html          把输入文件当成 HTML 文件。  
  -B,  --base=URL            解析与 URL 相关的  
                             HTML 输入文件 (由 -i -F 选项指定)。  
       --config=FILE         Specify config file to use.  
  
下载：  
  -t,  --tries=NUMBER            设置重试次数为 NUMBER (0 代表无限制)。  
       --retry-connrefused       即使拒绝连接也是重试。  
  -O,  --output-document=FILE    将文档写入 FILE。  
  -nc, --no-clobber              skip downloads that would download to  
                                 existing files (overwriting them).  
  -c,  --continue                断点续传下载文件。  
       --progress=TYPE           选择进度条类型。  
  -N,  --timestamping            只获取比本地文件新的文件。  
  --no-use-server-timestamps     不用服务器上的时间戳来设置本地文件。  
  -S,  --server-response         打印服务器响应。  
       --spider                  不下载任何文件。  
  -T,  --timeout=SECONDS         将所有超时设为 SECONDS 秒。  
       --dns-timeout=SECS        设置 DNS 查寻超时为 SECS 秒。  
       --connect-timeout=SECS    设置连接超时为 SECS 秒。  
       --read-timeout=SECS       设置读取超时为 SECS 秒。  
  -w,  --wait=SECONDS            等待间隔为 SECONDS 秒。  
       --waitretry=SECONDS       在获取文件的重试期间等待 1..SECONDS 秒。  
       --random-wait             获取多个文件时，每次随机等待间隔  
                                 0.5\*WAIT...1.5\*WAIT 秒。  
       --no-proxy                禁止使用代理。  
  -Q,  --quota=NUMBER            设置获取配额为 NUMBER 字节。  
       --bind-address=ADDRESS    绑定至本地主机上的 ADDRESS (主机名或是 IP)。  
       --limit-rate=RATE         限制下载速率为 RATE。  
       --no-dns-cache            关闭 DNS 查寻缓存。  
       --restrict-file-names=OS  限定文件名中的字符为 OS 允许的字符。  
       --ignore-case             匹配文件/目录时忽略大小写。  
  -4,  --inet4-only              仅连接至 IPv4 地址。  
  -6,  --inet6-only              仅连接至 IPv6 地址。  
       --prefer-family=FAMILY    首先连接至指定协议的地址  
                                 FAMILY 为 IPv6，IPv4 或是 none。  
       --user=USER               将 ftp 和 http 的用户名均设置为 USER。  
       --password=PASS           将 ftp 和 http 的密码均设置为 PASS。  
       --ask-password            提示输入密码。  
       --no-iri                  关闭 IRI 支持。  
       --local-encoding=ENC      IRI (国际化资源标识符) 使用 ENC 作为本地编码。  
       --remote-encoding=ENC     使用 ENC 作为默认远程编码。  
       --unlink                  remove file before clobber.  
  
目录：  
  -nd, --no-directories           不创建目录。  
  -x,  --force-directories        强制创建目录。  
  -nH, --no-host-directories      不要创建主目录。  
       --protocol-directories     在目录中使用协议名称。  
  -P,  --directory-prefix=PREFIX  以 PREFIX/... 保存文件  
       --cut-dirs=NUMBER          忽略远程目录中 NUMBER 个目录层。  
  
HTTP 选项：  
       --http-user=USER        设置 http 用户名为 USER。  
       --http-password=PASS    设置 http 密码为 PASS。  
       --no-cache              不在服务器上缓存数据。  
       --default-page=NAME     改变默认页  
                               (默认页通常是“index.html”)。  
  -E,  --adjust-extension      以合适的扩展名保存 HTML/CSS 文档。  
       --ignore-length         忽略头部的‘Content-Length’区域。  
       --header=STRING         在头部插入 STRING。  
       --max-redirect          每页所允许的最大重定向。  
       --proxy-user=USER       使用 USER 作为代理用户名。  
       --proxy-password=PASS   使用 PASS 作为代理密码。  
       --referer=URL           在 HTTP 请求头包含‘Referer: URL’。  
       --save-headers          将 HTTP 头保存至文件。  
  -U,  --user-agent=AGENT      标识为 AGENT 而不是 Wget/VERSION。  
       --no-http-keep-alive    禁用 HTTP keep-alive (永久连接)。  
       --no-cookies            不使用 cookies。  
       --load-cookies=FILE     会话开始前从 FILE 中载入 cookies。  
       --save-cookies=FILE     会话结束后保存 cookies 至 FILE。  
       --keep-session-cookies  载入并保存会话 (非永久) cookies。  
       --post-data=STRING      使用 POST 方式；把 STRING 作为数据发送。  
       --post-file=FILE        使用 POST 方式；发送 FILE 内容。  
       --content-disposition   当选中本地文件名时  
                               允许 Content-Disposition 头部 (尚在实验)。  
       --auth-no-challenge     发送不含服务器询问的首次等待  
                               的基本 HTTP 验证信息。  
  
HTTPS (SSL/TLS) 选项：  
       --secure-protocol=PR     选择安全协议，可以是 auto、SSLv2、  
                                SSLv3 或是 TLSv1 中的一个。  
       --no-check-certificate   不要验证服务器的证书。  
       --certificate=FILE       客户端证书文件。  
       --certificate-type=TYPE  客户端证书类型，PEM 或 DER。  
       --private-key=FILE       私钥文件。  
       --private-key-type=TYPE  私钥文件类型，PEM 或 DER。  
       --ca-certificate=FILE    带有一组 CA 认证的文件。  
       --ca-directory=DIR       保存 CA 认证的哈希列表的目录。  
       --random-file=FILE       带有生成 SSL PRNG 的随机数据的文件。  
       --egd-file=FILE          用于命名带有随机数据的 EGD 套接字的文件。  
  
FTP 选项：  
       --ftp-user=USER         设置 ftp 用户名为 USER。  
       --ftp-password=PASS     设置 ftp 密码为 PASS。  
       --no-remove-listing     不要删除‘.listing’文件。  
       --no-glob               不在 FTP 文件名中使用通配符展开。  
       --no-passive-ftp        禁用“passive”传输模式。  
       --retr-symlinks         递归目录时，获取链接的文件 (而非目录)。  
  
递归下载：  
  -r,  --recursive          指定递归下载。  
  -l,  --level=NUMBER       最大递归深度 (inf 或 0 代表无限制，即全部下载)。  
       --delete-after       下载完成后删除本地文件。  
  -k,  --convert-links      让下载得到的 HTML 或 CSS 中的链接指向本地文件。  
  -K,  --backup-converted   在转换文件 X 前先将它备份为 X.orig。  
  -m,  --mirror             -N -r -l inf --no-remove-listing 的缩写形式。  
  -p,  --page-requisites    下载所有用于显示 HTML 页面的图片之类的元素。  
       --strict-comments    用严格方式 (SGML) 处理 HTML 注释。  
  
递归接受/拒绝：  
  -A,  --accept=LIST               逗号分隔的可接受的扩展名列表。  
  -R,  --reject=LIST               逗号分隔的要拒绝的扩展名列表。  
  -D,  --domains=LIST              逗号分隔的可接受的域列表。  
       --exclude-domains=LIST      逗号分隔的要拒绝的域列表。  
       --follow-ftp                跟踪 HTML 文档中的 FTP 链接。  
       --follow-tags=LIST          逗号分隔的跟踪的 HTML 标识列表。  
       --ignore-tags=LIST          逗号分隔的忽略的 HTML 标识列表。  
  -H,  --span-hosts                递归时转向外部主机。  
  -L,  --relative                  只跟踪有关系的链接。  
  -I,  --include-directories=LIST  允许目录的列表。  
  --trust-server-names             use the name specified by the redirection  
                                   url last component.  
  -X,  --exclude-directories=LIST  排除目录的列表。  
  -np, --no-parent                 不追溯至父目录。

**参数有点多，没关系，常用的掌握，其他的用的时候再查**

**wget常用参数示例**

#下载某个文件，wget 文件的地址  
$wget https://blog.csdn.net/qq\_38646470

vmstat

**vmstat介绍**

vmstat reports information about processes, memory, paging, block IO, traps, disks and cpu activity.

vmstat报告有关进程，内存，页面调度，块IO，陷阱，磁盘和cpu活动的信息。

**vmstat参数格式**

vmstat [options] [delay [count]]

**vmstat命令参数**

-a：显示活跃和非活跃内存

-f：显示从系统启动至今的fork数量 。

-m：显示slabinfo

-n：只在开始时显示一次各字段名称。

-s：显示内存相关统计信息及多种系统活动数量。

delay：刷新时间间隔。如果不指定，只显示一条结果。

count：刷新次数。如果不指定刷新次数，但指定了刷新时间间隔，这时刷新次数为无穷。

-d：显示磁盘相关统计信息。

-p：显示指定磁盘分区统计信息

-S：使用指定单位显示。参数有 k 、K 、m 、M ，分别代表1000、1024、1000000、1048576字节（byte）。默认单位为K（1024 bytes）

**vmstat常用参数示例**

#显示虚拟内存情况  
$ vmstat  
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----  
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st  
 1  0      0 23764228 507816 36953948    0    0     3     5    0    0  1  0 98  0  0

**解释**

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：（现在的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

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 。以百分比表示。

wa: 等待IO时间

#表示在3秒时间内进行3次采样。将得到一个数据汇总他能够反映真正的系统情况。  
$vmstat 3 3  
#查看系统fork多少次  
$ vmstat -f  
    166484246 forks  
#查看内存使用的详细信息  
$vmstat -s  
#查看磁盘的读/写  
$vmstat -d  
#查看系统的slab信息  
$vmstat -m

free

**free介绍**

free  displays the total amount of free and used physical and swap memory in the system, as well as the buffers and caches used by the kernel.

free显示系统中可用和可用的物理内存和交换内存的总量，以及内核使用的缓冲区和高速缓存。

**free参数格式**

free [options]

**free命令参数**

-b 以Byte为单位显示内存使用情况。

-k 以KB为单位显示内存使用情况。

-m 以MB为单位显示内存使用情况。

-g  以GB为单位显示内存使用情况。

-o 不显示缓冲区调节列。

-s<间隔秒数> 持续观察内存使用状况。

-t 显示内存总和列。

**free常用参数示例**

#显示内存使用情况  
$ free  
              total        used        free      shared  buff/cache   available  
Mem:       65808884     4582700    23754736         684    37471448    60913052  
$ free -h  
              total        used        free      shared  buff/cache   available  
Mem:            62G        4.4G         22G        684K         35G         58G  
Swap:            0B          0B          0B

**解释**

total:总计物理内存的大小。

used:已使用多大。

free:可用有多少。

Shared:多个进程共享的内存总额。

Buffers/cached:磁盘缓存的大小。

第三行(-/+ buffers/cached):

used:已使用多大。

free:可用有多少。

#周期性的查询内存使用信息，5s执行一次  
$ free -s 5

top

**top介绍**

The  top  program provides a dynamic real-time view of a running system.  It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel.  The types of system summary information shown and the types, order and size of information displayed for  processes  are  all  user configurable and that configuration can be made persistent across restarts.  
      The  program  provides a limited interactive interface for process manipulation as well as a much more extensive interface for personal configuration  --  encompassing every aspect of its operation.  And while top is referred to throughout this document, you are free to name the program anything you wish.  That new name, possibly  an alias, will then be reflected on top's display and used when reading and writing a configuration file.

top程序提供正在运行的系统的动态实时视图。它可以显示系统摘要信息以及Linux内核当前正在管理的进程或线程的列表。所显示的系统摘要信息的类型以及为进程显示的信息的类型，顺序和大小都是用户可配置的，并且可以使配置在重新启动后保持不变。  
       该程序为流程操作提供了一个有限的交互式界面，并为个人配置提供了更为广泛的界面-涵盖了其操作的各个方面。尽管在本文档中始终引用top，但是您可以随意为程序命名。然后，该新名称（可能是别名）将反映在顶部的显示屏上，并在读写配置文件时使用。

**top参数格式**

top -hv|-bcHiOSs -d secs -n max -u|U user -p pid -o fld -w [cols]

**top命令参数**

-b 批处理

-c 显示完整的治命令

-I 忽略失效过程

-s 保密模式

-S 累积模式

-i<时间> 设置间隔时间

-u<用户名> 指定用户名

-p<进程号> 指定进程

-n<次数> 循环显示的次数

**top常用参数示例**

#top  
$ top  
top - 00:56:07 up 149 days, 14:40,  1 user,  load average: 0.00, 0.02, 0.05  
Tasks: 254 total,   1 running, 253 sleeping,   0 stopped,   0 zombie  
%Cpu(s):  1.4 us,  0.3 sy,  0.0 ni, 98.3 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st  
KiB Mem : 65808884 total, 23749772 free,  4586160 used, 37472952 buff/cache  
KiB Swap:        0 total,        0 free,        0 used. 60909608 avail Mem  
  
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND  
24397 dongshan  20   0 17.972g 688312  13728 S   6.2  1.0   7:09.11 java  
    1 root      20   0   42140   3684   1476 S   0.0  0.0  23:58.88 systemd  
    2 root      20   0       0      0      0 S   0.0  0.0   0:05.47 kthreadd  
    3 root      20   0       0      0      0 S   0.0  0.0   0:16.06 ksoftirqd/0  
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H  
    7 root      rt   0       0      0      0 S   0.0  0.0   1:27.00 migration/0  
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu\_bh  
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0

**解释**

**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：**

00:56:07 — 当前系统时间

up 149 days, 14:40 — 系统已经运行了149天14小时40分钟（在这期间系统没有重启过的）

1users — 当前有1个用户登录系统

load average: 0.00, 0.02, 0.05 — 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）**

系统现在共有254个进程，其中处于运行中的有1个，253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**

%Cpu(s):  1.4 us,  0.3 sy,  0.0 ni, 98.3 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st

1.4 us — 用户空间占用CPU的百分比。

0.3 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

98.3 id — 空闲CPU百分比

0.1 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

**第四行,内存状态**

65808884 total  物理内存总量

23749772 free  使用中的内存总量

4586160 used   空闲内存总量

37472952 buff/cache  缓存的内存量

**第五行，swap交换分区信息**

0 total   交换区总量

0 use   使用的交换区总量

0 free   空闲交换区总量

60909608 avail Mem  可用交换区总量

**第七行以下：各进程（任务）的状态监控**

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）

sar

**sar介绍**

sar（System Activity Reporter系统活动情况报告）是目前 Linux 上最为全面的系统性能分析工具之一，可以从多方面对系统的活动进行报告，包括：文件的读写情况、 系统调用的使用情况、磁盘I/O、CPU效率、内存使用状况、进程活动及IPC有关的活动等。

**sar参数格式**

sar [options] [-A] [-o file] t [n]

**sar命令参数**

-A：所有报告的总和

-u：输出CPU使用情况的统计信息

-v：输出inode、文件和其他内核表的统计信息

-d：输出每一个块设备的活动信息

-r：输出内存和交换空间的统计信息

-b：显示I/O和传送速率的统计信息

-a：文件读写情况

-c：输出进程统计信息，每秒创建的进程数

-R：输出内存页面的统计信息

-y：终端设备活动情况

-w：输出系统交换活动信息