

# Linux command summary

Command	description
• ll	查看当前目录下的所有文件的权限
• ls -al	显示所有（包括隐藏的文件）文件
• ls -ls	按照文件大小从大到小显示
• date	查看系统时间
• pwd	显示当前目录
• mkdir/rmdir	新建或删除一个新的/空目录
• Mkdir -p dir	新建多级目录
• cd -	返回上一次 cd 的目录
• cat -n filename	查看当前文件内容，顺便输出行号
• mv file1 file2	将文件名 file1 改为 file2
• ctrl+z	暂停目前的命令
• ctrl+c	终止目前的命令
• ctrl+s	暂停屏幕的输出
• ctrl+q	恢复屏幕的输出
• jobs	查看当前有多少 job 在执行
• bg	查看当前后台执行的 job 情况
• fg % id	将后台运行的某个进程放到前台执行
• ps	查看当前执行的进程数
• kill jobid	关闭当前的某个进程
• ssh -X	登陆服务器，加 option X 表示服务器可以可视化
• su user	切换当前账户到 user 这个账户
• who	查看当前服务器登录的用户情况
• who am i	显示当前 local 用户登录信息
• exit	退出当前执行的账户
• xeyes	查看当前服务器是否可视化，如果可以就有 eyes 弹窗显示
• chmod xxx filename	修改文件的 permission（readable，writeable，executable）
• touch filename	当前目录下新建空文件 filename，如果文件存在，可修改文件的时间戳
• rm -rf	强制执行删除一个文件
• double tab	查看当前用户可执行的文件数或者操作数
• vimdiff file1 file2	比较当前两个文件的差异
• echo str	当前目录下查找含有特定字符的文件
• alias command='str'	对可执行文件设置 alias 方便经常执行调用
• more/less	一页一页地显示文件内容，向下翻页查看文件内容

- `tail -f file`                      实时显示文件的状态，特别执行 `job` 时，查看 `log` 文件内容
- `man command`                    查看 `command` 的操作指南，特别是 `option` 的 `usage`

### About test

- `test -e demo.sh && echo "exist" || echo "not exist" #test the file demo.sh exist or not.`
- `test -z $filename && echo "you must input a filename"&& exit 0`
- `test ! -e $filename && echo "the filename does not exist" && exit 0`
- `test -f $filename && echo filetype="file is a filename"&& exit 0`
- `test -d $filename && echo filetype="file is a directory "&& exit 0`
- `test -r $filename && perm="readable"`
- `test -w $filename && perm="$perm writable"`
- `test -x $filename && perm="$perm executable" 1`

### About file types

- Regular file `-frwxrwxrwx`
- ASCII file
- Binary file
- Date file
- Directory `drwxrwxrwx`
- Link `lrwxrwxrwx`
- Device/block file `brwxrwxrwx` /character file `crwxrwxrwx` /sockets file `srwxrwxrwx`
- Sockets
- FIFO, pipe

### About vi/vim editors

#### ➤ Regular mode

- `h/j/k/l`                    turn left/down/up/right
- `n<space>`                turn right by n steps
- `G`                        turn to the bot
- `gg`                        turn to the top
- `/word`                    find the word downward
- `n`                        double find `/word`
- `N`                        double find `/word` reverse
- `?word`                    find the word upward
- `x, X`                    delete strings backward or forward
- `dd`                        delete the whole line

- `ndd`                delete n lines downward
- `yy`                 copy the whole line
- `p, P`               paste the copy context after/before this line.
- `u`                  rebuilt the last condition
- `O`                  turn to the head of line
- `$`                  turn to the end of line
- `fa`                 find the 1<sup>st</sup> string "a" forward
- `3fa`                find the 3<sup>rd</sup> string "a" forward
- `Ctrl+z`            将当前的 vi,vim editor 放置到后台执行，回来用 `fg`
- `Env`                查看当前终端有关的环境变量
- `tr 'a' 'b'`        将查找结果中的 a 全部替换成 b
- `Esc +u`            撤销上一条命令!!!
- `File filename`    打印出文件的类型信息 (ll filename 第一列的第一个字母代表)
- `File -b filename` 打印不包含文件名在内的文件的类型信息

#### ➤ Editor mode ( insert mode)

- `i`                  change from regular to editor mode
- `esc`                change from editor to regular mode
- `ctrl+p`            auto polishing (自动补齐功能)

#### ➤ Command mode

- `: wq`                save change and exit
- `: q!`                don't save change and exit
- `: set nu/nonu`     set row numbers auto

#### ➤ Multiple windows operate

- `:sp filename`     in command mode
- `Ctrl+w+(j/k)`    switch windows
- `Ctrl+w+q`        exit multiple windows operate

#### About retarget data

- `Com1 && com2` 如果 com1 正确执行，则开始执行 com2，如果没有，com2 不执行
- `Com1 || com2` 如果 com1 正确执行，则 com2 不执行，如果没有，com2 执行

#### About pipe

- `cut -d 'splits str' -f fields` 查找出用 splits str 分割的变量的第 fields 的区域。
- `cut -c fields` 排列整齐的信息。

- `grep [-acnv] --color=auto 'str' filename` 按行查找文件中的指定字符串。
- `xargs` 由于很多 `command` 并不支持 `pipe`，所以加上 `xargs` 在前面可以读取 `stdin`。
- `awk -F : '{print $1 $3 $5}'` 将得到的列表信息的第 1,3,5 列输出。
- `> filename` 将结果以文件 `file` 的形式保存，方便查看。

## Linux setting usage

### ➤ Set display row number as default

1. `cp /usr/share/vim/vimrc ~/.vimrc` (if Redhot system, change `usr` to `etc`)
2. `vi ~/.vimrc`
3. go into editor mode
4. syntax on
5. `set nu`

FYI, check a blog link: <http://www.cnblogs.com/yjmyzz/p/4019783.html>

### ➤ Set open vim editor as default when using vi editor

1. `which vi vim`(display the directory)
2. `rm /bin/vi` or `mv /bin/vi /bin/vi/old`
3. `cp /usr/bin/vim /bin/vi` or `alias vi ='/usr/bin/vim'`

### ➤ Change strings in files by one time (very useful)

1. `sed -i "s/old str/new str/g" `grep old str -rl dir``

### ➤ About close process

1. `jobs` and `kill % num`
2. `ps` and `kill jobid`

## Linux office usage

### ➤ Print in terminal

1.echo 自带换行符

Option: -n 标志可以忽略结尾换行符

-e 接受双引号字符串内的转义字符

2.Printf 没有自带换行符，可以通过加 “\n” 来添加换行

### ➤ get program's ID

Pgrep programName

### ➤ get environment variable

Cat /proc/\$PID/environ

### ➤ set variable value in shell

Var=value, should look out the diff between 'var=value' and 'var = value' (equal)

### ➤ set temporary environment variable

Export PATH

### ➤ set forever environment variable

Vi /etc/profile

Add "export "PATH""

Source /etc/profile or ./etc/profile

### ➤ get the length of a variable

Echo \${#var}

➤ 对 variable 的算术运算使用 let，但是需要注意的是 let 只能进行整数的操作，而 bc 可以用于浮点型的算术运算。

```
Var1=4;var2=5;let result=var1+var2
```

```
Echo "scale=2;var1*var2" | bc (将精度设置为 2 位小数)
```

```
Echo "obase=10;ibase=2; $result" | bc (将 result 这个变量的值由输入的 2 进制转换为输出的 10 进制)
```

➤ get some terminal info (tput and stty tools)

1.获取终端的行数和列数

```
Tput cols /tput lines
```

2.打印当前终端名

```
Tput longname
```

...

➤ 获取设置日期和延时

1.date "+%d %B %Y"

2.打印纪元时: date +%s

3.将日期串输出为纪元时: date --date "Jan 20 2001" +%A

4.检查一组命令花费的时间

```
Start=$(date +%s)
```

```
End=$(date +%s)
```

```
Diff=$(( end-start ))
```

➤ 调试脚本

```
Bash -x script.sh / sh -x script.sh
```

➤ 函数格式

```
Function fname(){
```

```
statements
```

```
}
```

或

```
Fname(){
```

```
statements
```

```
}
```

导出函数: `export -f fname`

```
Cmd;
```

`Echo $?;` #输出命令 `cmd` 的返回值（退出状态）。如果命令成功退出，退出状态应该是 0，否则是非 0。

#### ➤函数格式

`[ condition ] && action` #如果 `condition` 条件为真，执行 `action`

`[ condition ] || action` #如果 `condition` 条件为假，执行 `action`

`[ condition1 -a condition2 ]` #逻辑与

`[ condition1 -o condition2 ]` #逻辑或

比较符: `gt lt ge le eq ne`

`If [ condition ]` 可以写成 `test condition`

如果是字符串的比较，需比如: `[[ $str1 = $str2 ]]`

#### ➤命令乐趣-cat

`Cat file1 file2....:` 可以拼接多个文件一起显示

`Cat` 可以实现将输入文件的内容与标准输入拼接在一起

`Echo "hello" | cat - file1` “-”理解为标准输入文本的文件名。

`Cat` 可以实现将文本文件中的多余空行压缩到一行显示

`Cat -s file | tr -s '\n'` 可以将多个 `\n` 字符压缩成单个 `\n`

#### ➤录制与回放 terminal 会话 `script`、`scriptreplay`

```
Script -t 2> timing.log -a output.session
```

```
Type commands;
```

Exit

其中 `timing` 用于储存时序信息，描述每个 `command` 在何时运行，`output` 用于存储命令输出，`-t` 选项用于将时序数据导入 `stderr`。2>则将 `stderr` 重定向到 `timing` 中。借助这两个文件，可以利用下面命令回放命令执行过程：

`Scriptreplay timing.log output.session`

Script 实现多用户之间视频通话

1.建立通话连接

Terminal1（广播员）：`Mkfifo scriptfifo`

Terminal2（听众）：`cat scriptfifo`

2.开始通话

Terminal1：`script -f scriptfifo`

Commands

Exit

➤命令乐趣-find

- 1.find base\_path 查找当前目录以及子目录中所有文件和文件夹。
- 2.Find path -name “\*.sh” -print 查找当前目录以及子目录中包含 sh 后缀的文件。
- 3.Find path \{( -name “\*.sh” -o -name “\*.out” \} -print 查找多个类型文件。
- 4.Find path ! -name “\*.sh” -print 反向匹配查找出后缀不是 sh 的文件。
- 5.Find path -maxdepth num1 -mindepth num2 -print 设定搜索的范围。

文件类型	类型参数
普通文件	F
符号链接	l
目录	d



6.find  
文件类型搜索

字符设备	c
块设备	b
套接字	s
Fifo	p

path -type f-print 根据

7.基于访问时间、修改时间、变化时间的查找

-atime -mtime -ctime

-amin -mmin -cmin

Ex: find . -type f -amin -7 -print 打印出访问时间小于 7 分钟的所有文件。

8. 基于文件大小-size

find . -type -size +2k -print 打印出文件大小大于 2kb 的文件。

文件大

小

b	块
c	字节
w	字
k	千字节
M	兆
G	吉字节

9.删除匹配的文件 -delete

10.基于文件所有权的匹配 -perm

11.基于用户的文件的匹配 -user

Find . -type -f -user kevin - print 打印出用户 user 拥有的所有文件。

12.find 最强大的 option: -exec

-exec 可以与其他命令结合起来

Ex: # find . -type f -user root -exec chown kevin {} \;

{ } 是一个特殊的字符串，与-exec 结合使用，对于每一个匹配的文件，{ } 会被替换成相应的文件名。

#### ➤命令乐趣-xargs

管道可以将一个命令的 stdout 重定向到另一个命令的 stdin，但是有些命令只能以命令行参数的形式接受数据，而无法通过 stdin 接受数据流。Xargs 可以将标准输入数据转换成命令行参数，也可将单行或者多行文本输入转换成其他格式。

- 多行输入变成单行输入

Cat file | xargs

- 单行输入转换成多行输出

Cat file | xargs -n 3 (每行由 3 个参数组成划分多行)

- 用 xargs -d 设定数据划分的指定定界符（默认 xargs 采用内部字段分割符 IFS 作为输入定界符）

#### ➤命令乐趣-tr

- tr 用于替换、加密、解密

echo "hello" | tr 'a-z' 'A-Z' (大小写相互转换)

- tr 用于删除

Echo "hello 124 world 456" | tr -d "0-9" 将包含 0-9 的字符删除

Echo "hello 123 world 456" | tr -d -c '0-9' 将不包含 0-9 的字符删除

- tr 用于压缩

Echo 'hello 123 world 456' | tr -s ' ' 将包含多个空格的地方压缩成单个字符空格。

#### ➤校验和与核实

- 对于一个文件的上传与下载，可能会出现失真的情况，需要校验。

Md5sum file1 > output.md5 #采用的是 md5（一种算法机制）

Md5sum -c output.md5 应该 echo 一个 OK

#### ➤排序、单一、重复

`$#` 是传给脚本的参数个数

`$0` 是脚本本身的名字

`$1` 是传递给该 shell 脚本的第一个参数

`$2` 是传递给该 shell 脚本的第二个参数

`$@` 是传给脚本的所有参数的列表

`$$` 是程序的 PID

`$?` 是显示最后命令的退出状态，0 表示没有错误，其他表示有错误

`$*` 是以一个单字符串显示所有向脚本传递的参数，与位置变量不同，参数可超过 9 个

- `sort -nrk 1 file option` 中 `n` 表示按照数字排序，`r` 表示将 `file` 倒序，`k` 表示 `key`，代表的是第几列。

- `sort` 和 `uniq` 合用，可以打印出单一的行，如果一行重复出现多次，也只打印一次

`Sort file | uniq`

`Sort file | uniq -c` (统计行在文件中出现的次数)

`Sort file | uniq -d` (找出文件中重复的行)

#### ➤生成任意大小的文件

- `dd if=inputfile of=output bs=fileSize count=fileNumbers`

`if` 代表输入文件，`of` 代表输出文件，`bs` 代表输出文件的大小，`count` 代表输出文件的数目，如果 `if` 不指定，默认从 `stdin` 读取输入，如果 `of` 不指定，默认输出到 `stdout`。

#### ➤求文本文件的交集与差集

- `comm`: 必须使用排过序的文件作为输入

Ex: `comm a.txt b.txt` (输出的第一列包含了只在 `a.txt` 中出现的行，输出的第二列包含了只在 `b.txt` 中出现的行，输出的第三列包含了两个文件相同的行)，通过 `-1 -2 -3` 可以分别删除其中的第一第二第三列。

### ➤重命名文件

- `rename 's/ /_g' *` 将空格替换为字符 “\_”
- `rename 'y/A-Z/a-z/' *` 将文件名大写转换成小写。
- `find path -type f -name "*.txt" -exec mv { } target_dir \;` 将指定目录下文件后缀为.txt 的文件移动到 target\_dir 目录下。
- `find path -type f -exec rename 's/ /_g' {} \;` 将所有文件名中的空格替换为字符 “\_”

### ➤设置文件为不可修改

- `chattr +i filename` 设置文件为不可修改
- `chattr -i filename` 设置文件重新可写，移除不可修改性。

### ➤批量生成名字不同的文件

- `for name in {1..100}.txt`

`do touch $name done`

`Touch -a filename` 更改文件的访问时间（access time）

`Touch -m filename` 更改文件的修改时间（modify time）

`Touch -d “time format” filename` 将文件的时间戳指定为特定的时间和日期。

### ➤创建符号链接

- `ln -s target symbolic_link_name` 在 target 目录下创建 symbolic\_link\_name 文件。
- 打印出当前目录下的符号链接，打印出\$8 列为文件名

```
ls -l | grep "^l" | awk '{ print $8 }'
```

- 读取 link 文件的指向内容 `readlink ~/web`

### ➤查找文件差异并修补

- diff，在一体化 diff 输出中，以+开头的是新加入的行，以-开头的是删除的行。

```
Diff -u file1 file2 > diffoutfile
```

修补：当应用于 file1 时，就可以得到 file2，应用于 file2 时，就可以得到 file1

Patch -p1 file1/file2 <diffout

重复 patch 操作，可以撤销做出的修改。

#### ➤head 和 tail

Head 命令总是读取输入文件的头部

Tail 命令总是读取输入文件的尾部

- head -n filename 指定打印出文件的头 n 行内容
- head -n -number filename 指定打印出文件除最后 number 行的内容
- tail -n filename 指定打印出文件的尾 n 行内容
- tail -n +number filename 指定打印出文件除最开始 number 行的内容
- tail -f filename 指定打印出新添加到文件中的内容，一般用于显示 log 文件的信息，很有用！！

#### ➤列出当前目录下的目录文件

- ls -d \*/

#### ➤用 pushd 和 popd 快速定位路径

- pushd dir，将目录压入堆栈
- dirs 显示当前堆栈中的目录名，每条路径从 0 到 n 编号
- pushd +n 跳转到第 n 个目录
- popd 将最后一次压入堆栈的目录弹出
- popd +n 将第 n 个目录弹出堆栈
- 如果需要切换到刚才操作的目录，可以使用快捷命令 cd -

#### ➤统计文件的行数、单词数、字符数

统计 LOC(line of Code, 代码行数)是一件很重要的工作！

wc (word count)是一个用于统计的工具

- 统计行数

wc -l file or cat file | wc -l

- 统计单词数

`wc -w file` or `cat file | wc -w`

- 统计字符数

`wc -c file` or `cat file | wc -c`

- 统计行数，单词数，字符数

`wc file`

- 统计打印最长行的长度

`wc file -L`

#### ➤打印目录树

需要安装，一般 linux 发行版没有包含这个命令。

#### ➤用 cut 按列切分文件

• cut 是一个帮我们将文本按列进行切分的小工具，它也可以指定分隔每列的定界符。每列为一个字段，制表符是列或者字段的默认定界符。

- `cut -f 1,3 filename` : 提取出文件中的第 1 列和第 3 列。
- `cut -fs 1,3 filename` : 提取出文件的 1,3 列，避免打印出不包含定界符的行。
- `cut -f 3 --complement filename` : 提取除开第 3 列之外所有的列。
- `cut -f 1,2 -d “;” filename` : 按照 “;” 划定列，然后提取第 1,2 列。

#### ➤用 tar 归档

• `tar -rcf output.tar file1 file2 file3..` :-c 表示 创建文件，-f 表示指定文件名。-r 表示追加文件

- `tar -xf archive.tar` : -x 表示提取
- `tar -c directory address` : -c 表示提取到指定的目录下
- `tar -Af file1.tar file2.tar` : -A 表示将两个 tar 文件合并。
- `tar -f archive.tar --delete file1 file2 ..` :--delete 表示将 file1 file2 从 archive.tar 中去掉

➤sed 入门 sed 可以替换给定文本中的字符串，可以利用正则表达式进行匹配。

- sed -i ‘s/pattern/replace\_string/’ file option i 可以将替换结果应用于原文件。
- 后缀/g 表示替换每一处匹配。sed ‘s/pattern/replace\_string/g’ file 。
- 当需要从第 N 次匹配处开始匹配时，可以使用 /Ng, 如 sed ‘s/pattern/replace\_string/2g’ file,表示第一个匹配成功处不替换。
- /作为定界符，其他字符 ‘:’ ‘|’ 等都可以作为定界符。
- sed 可以用来移除空白行，sed ‘/^\$/d’ file ,/pattern/d 会移除匹配样式的行。  
^\$表示空白行。
- 已匹配字符串标记&，如： sed ‘s/\w\+/[&]/g’ file.正则表达式\w\+匹配每一个单词，然后用[&]替换它，&对应于之前所匹配的单词。
- 子串匹配标记\1 \2 …… 如： sed ‘s/digit \([0-9]\)/\1/’ file 表示匹配 pattern ‘digit \([0-9]\)’ 字符串中的子字符串\([pattern]\). 并且是第一个子字符串。 \2 表示是第二个子字符串，以此类推。

