

# Linux基础入门

讲师：王晓春

## Linux 基础入门

### 内容概述

#### 1 Linux 基础

##### 1.1 用户类型

##### 1.2 终端terminal

###### 1.2.1 终端类型

###### 1.2.2 查看当前的终端设备：tty

##### 1.3 交互式接口

###### 1.3.1 交互式接口类型

###### 1.3.2 什么是shell

###### 1.3.3 各种Shell

###### 1.3.4 bash shell

##### 1.4 设置主机名

##### 1.5 命令提示符

##### 1.6 执行命令

###### 1.6.1 执行命令过程

###### 1.6.2 shell中可执行的两类命令

###### 1.6.2.1 内部命令相关

###### 1.6.2.2 执行外部命令

###### 1.6.3 命令别名

###### 1.6.4 命令格式

##### 1.7 常见命令

###### 1.7.1 日期和时间

###### 1.7.2 关机和重启

###### 1.7.3 用户登录信息查看命令

###### 1.7.4 文本编辑

###### 1.7.5 会话管理

###### 1.7.5.1 screen

###### 1.7.5.2 tmux

###### 1.7.5 输出信息

##### 1.8 字符集和编码

###### 1.8.1 ASCII码

###### 1.8.2 Unicode

##### 1.9 命令行扩展和被括起来的集合

###### 1.9.1 命令行扩展：\$( )

###### 1.9.2 括号扩展：{ }

- 1.10 tab 键补全
  - 1.10.1 命令补全
  - 1.10.2 路径补全
  - 1.10.3 双击Tab键
- 1.11 命令行历史
- 1.12 调用命令行历史
- 1.13 bash的快捷键
- 2 获得帮助
  - 2.1 whatis
  - 2.2 查看命令的帮助
  - 2.3 --help或-h 选项
  - 2.4 man命令
  - 2.5 info
  - 2.6 Linux 安装提供的本地文档获取帮助
  - 2.7 命令自身提供的官方使用指南
  - 2.8 系统及第三方应用官方文档
    - 2.8.1通过在线文档获取帮助
    - 2.8.2 红帽知识库和官方在线文档
    - 2.8.3 红帽全球技术支持服务
  - 2.9 网站和搜索

# Linux 基础入门

---

## 内容概述

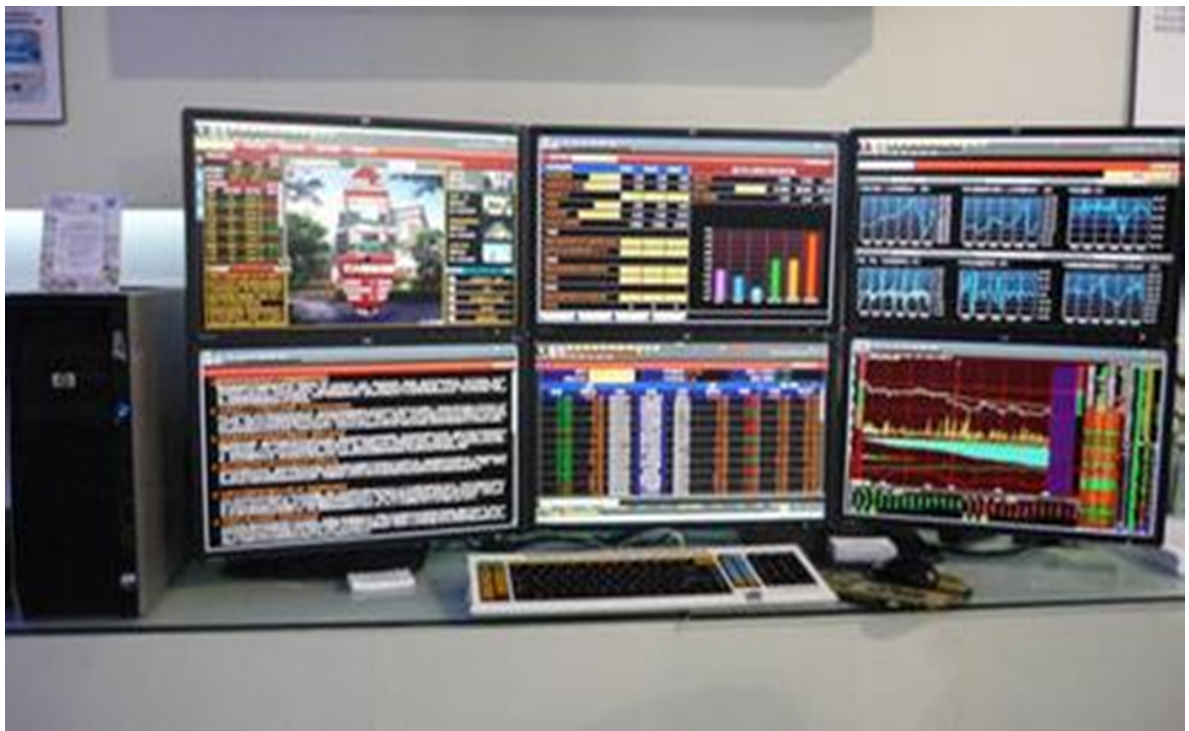
- 用户
- 终端
- Shell介绍
- 执行命令
- 简单命令
- Tab键补全
- 命令行历史
- bash快捷键
- 帮助用法

## 1 Linux 基础

### 1.1 用户类型

- root 用户 一个特殊的管理帐户 也被称为超级用户 root已接近完整的系统控制 对系统损害几乎有无限的能力 除非必要,不要登录为 root
- 普通（非特权）用户 权限有限 造成损害的能力比较有限

### 1.2 终端terminal



设备终端：键盘、鼠标、显示器

### 1.2.1 终端类型

- 控制台终端：/dev/console
- 串行终端：/dev/ttyS#
- 虚拟终端：tty：teletypewriters，/dev/tty#，tty 可有n个，Ctrl+Alt+F#
- 图形终端：startx, xwindows CentOS 6: Ctrl + Alt + F7 CentOS 7: 在哪个终端启动，即位于哪个虚拟终端
- 伪终端：pty：pseudo-tty，/dev/pts/# 如：SSH远程连接

### 1.2.2 查看当前的终端设备：tty

范例：

```
[root@centos8 ~]#tty  
/dev/pts/0
```

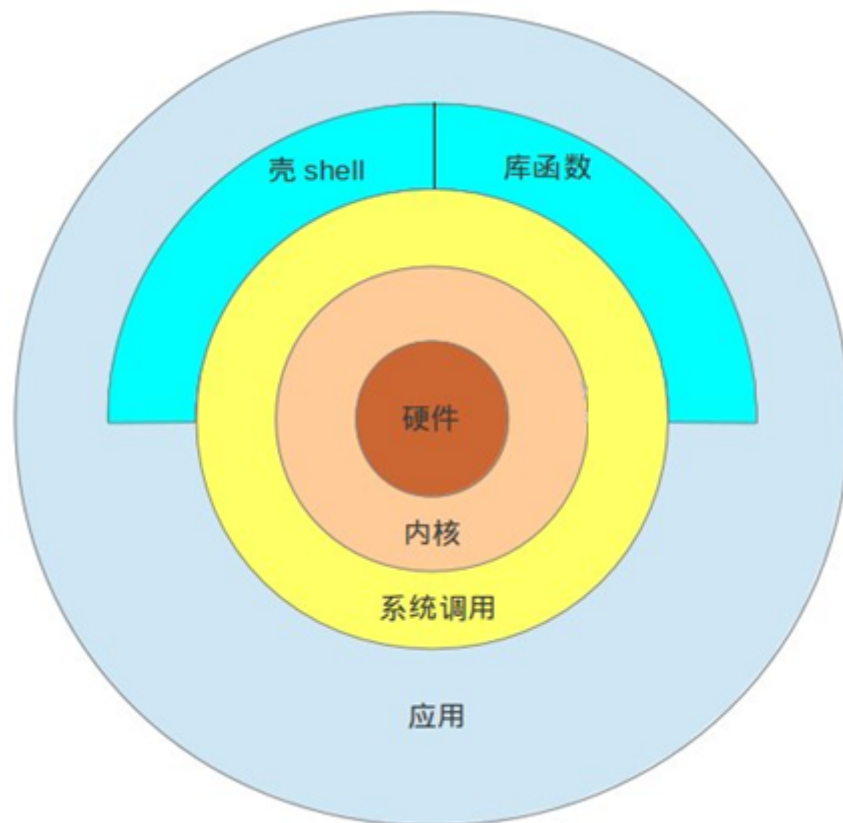
## 1.3 交互式接口

交互式接口：启动终端后，在终端设备附加一个交互式应用程序

### 1.3.1 交互式接口类型

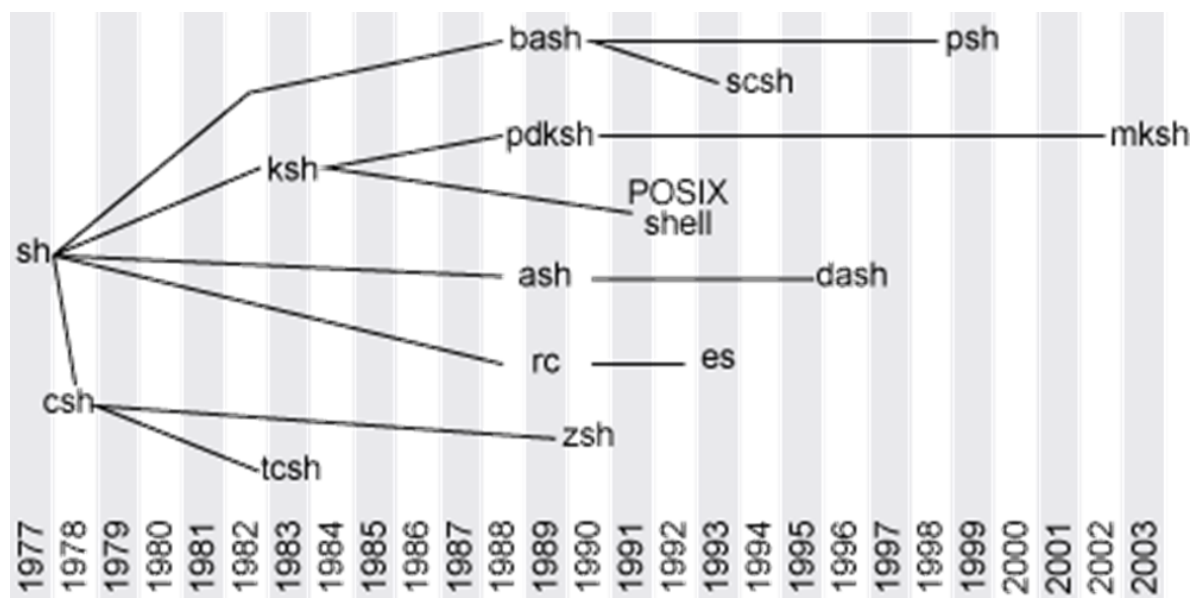
- GUI：Graphic User Interface X protocol, window manager, desktop Desktop: GNOME (C, 图形库gtk)，KDE (C++,图形库qt) XFCE (轻量级桌面)
- CLI：Command Line Interface shell程序

### 1.3.2 什么是shell



Shell 是Linux系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行 shell也被称为LINUX的命令解释器（command interpreter）shell是一种高级程序设计语言

### 1.3.3 各种Shell



- sh : Steve Bourne
- bash : Bourne-Again Shell , GPL , CentOS 和 Ubuntu 默认使用
- csh : c shell , C 语言风格
- tcsh
- ksh : Korn Shell, AIX 默认 shell
- zsh : MacOS默认shell

### 1.3.4 bash shell

GNU Bourne-Again Shell(bash)是GNU计划中重要的工具软件之一，目前也是 Linux标准的shell，与sh兼容

显示当前使用的shell

```
echo ${SHELL}
```

显示当前系统使用的所有shell

```
cat /etc/shells
```

## 1.4 设置主机名

```
hostname NAME
```

范例

```
[root@centos8 ~]#hostname      bj-yz-k8s-node1-100-10.magedu.com
```

注意：主机名不要使用下划线

## 1.5 命令提示符

命令提示符：prompt

范例：

```
[root@localhost ~]#
```

# 管理员

\$ 普通用户

显示提示符格式

```
[root@centos8 ~]#echo $PS1
\[ \e[1;35m\[ \u@\h \w]\$ \e[0m\]
```

修改提示符格式范例

```
PS1="\[ \e[1;5;41;33m\[ \u@\h \w]\$ \e[0m\]"
PS1=PS1="\[ \e[1;32m\[ \t \[ \e[1;33m\[ \u\[ \e[35m\]@\h\[ \e[1;31m\] \w\[ \e[1;32m\]\[ \e[0m\]\$"
```

提示符格式说明：

- \e 控制符\033
- \u 当前用户
- \h 主机名简称
- \H 主机名
- \w 当前工作目录
- \W 当前工作目录基名
- \t 24小时时间格式
- \T 12小时时间格式

- ! 命令历史数
- # 开机后命令历史数

范例：持久保存提示符格式相关变量 PS1

```
[root@centos8 ~]# echo 'PS1="\[\e[1;32m\][\t \[\e[1;33m\]\u\[\e[35m\]@h\[\e[1;31m\] \w\[\e[1;32m\]]\[\e[0m\]\\$"' > /etc/profile.d/env.sh
[root@centos8 ~]# cat /etc/profile.d/env.sh
PS1="\[\e[1;32m\][\t \[\e[1;33m\]\u\[\e[35m\]@h\[\e[1;31m\] \w\[\e[1;32m\]]\[\e[0m\]\\$"
[root@centos8 ~]# exit
logout

Connection closed by foreign host.

Disconnected from remote host(centos8) at 15:28:38.

Type `help' to learn how to use xshell prompt.
[c:\~]$
Reconnecting in 1 seconds. Press any key to exit local shell.
.

Connecting to 10.0.0.100:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+t+'.

Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Dec 11 15:28:34 2019 from 10.0.0.1
[15:28:40 root@centos8 ~]#
```

## 1.6 执行命令

### 1.6.1 执行命令过程

输入命令后回车，提请shell程序找到键入命令所对应的可执行程序或代码，并由其分析后提交给内核分配资源将其运行起来

### 1.6.2 shell中可执行的两类命令

- 内部命令：由shell自带的，而且通过某命令形式提供
- 外部命令：在文件系统路径下有对应的可执行程序文件

**区别指定的命令是内部或外部命令**

```
type COMMAND
```

范例: 查看是否存在对应内部和外部命令

```
[15:58:01 root@centos8 ~]# type -a echo
echo is a shell builtin
echo is /usr/bin/echo
```

### 1.6.2.1 内部命令相关

help 内部命令列表

enable 管理内部命令

- enable cmd 启用内部命令
- enable -n cmd 禁用内部命令
- enable -n 查看所有禁用的内部命令

### 1.6.2.2 执行外部命令

查看外部命令路径：

```
which -a | --skip-alias  
whereis
```

**Hash缓存表** 系统初始hash表为空，当外部命令执行时，默认会从PATH路径下寻找该命令，找到后会在这条命令的路径记录到hash表中，当再次使用该命令时，shell解释器首先会查看hash表，存在将执行之，如果不存在，将会去PATH路径下寻找，利用hash缓存表可大大提高命令的调用速率

hash 命令常见用法

- hash 显示hash缓存
- hash -l 显示hash缓存，可作为输入使用
- hash -p path name 将命令全路径path起别名为name
- hash -t name 打印缓存中name的路径
- hash -d name 清除name缓存
- hash -r 清除缓存

### 1.6.3 命令别名

对于经常执行的较长的命令，可以将其定义成较短的别名，以方便执行

显示当前shell进程所有可用的命令别名

```
alias
```

定义别名NAME，其相当于执行命令VALUE

```
alias NAME='VALUE'
```

范例:

```
[16:15:02 root@centos8 ~]#echo "alias free='free -h'" >> .bashrc
```

撤消别名：unalias

```
unalias [-a] name [name ...]  
unalias -a #取消所有别名
```

注意：在命令行中定义的别名，仅对当前shell进程有效

如果想永久有效，要定义在配置文件中

- 仅对当前用户：~/.bashrc

- 对所有用户有效：/etc/bashrc

编辑配置给出的新配置不会立即生效，bash进程重新读取配置文件

```
source /path/to/config_file
. /path/to/config_file
```

如果别名同原命令同名，如果要执行原命令，可使用

```
\ALIASNAME
"ALIASNAME"
'ALIASNAME'
command ALIASNAME
/path/command
```

## 1.6.4 命令格式

```
COMMAND [OPTIONS...] [ARGUMENTS...]
```

选项：用于启用或关闭命令的某个或某些功能

- 短选项：UNIX 风格选项，-c 例如：-l, -h
- 长选项：GNU风格选项，--word 例如：--all, --human
- BSD风格选项：一个字母，例如：a

参数：命令的作用对象，比如:文件名，用户名等

范例:

```
[16:28:27 root@centos8 ~]#id -u wang
1000
```

注意：

- 多个选项以及多参数和命令之间使用空白字符分隔
- 取消和结束命令执行：Ctrl+c，Ctrl+d
- 多个命令可以用 ";" 符号分开
- 一个命令可以用\分成多行

## 1.7 常见命令

### 1.7.1 日期和时间

Linux的两种时钟 系统时钟：由Linux内核通过CPU的工作频率进行的 硬件时钟：主板

相关命令

- date 显示和设置系统时间

范例：

```
date +%s
date -d @1509536033
```

- clock，hwclock: 显示硬件时钟 -s, --hctosys 以硬件时钟为准，校正系统时钟 -w, --systohc 以系统时钟为准，校正硬件时钟



时区：

```
/etc/localtime
```

范例:

```
[16:45:24 root@centos8 ~]#ll /etc/localtime
lrwxrwxrwx. 1 root root 35 Dec 11 11:19 /etc/localtime ->
../usr/share/zoneinfo/Asia/Shanghai
```

显示日历：

```
cal -y
```

范例:

```
[16:47:36 root@centos8 ~]#cal 9 1752
September 1752
Su Mo Tu We Th Fr Sa
          1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

### 1.7.2 关机和重启

关机：

- halt
- poweroff

重启：

reboot -f: 强制，不调用shutdown -p: 切断电源

关机或重启：shutdown

```
shutdown [OPTION]... [TIME] [MESSAGE]
```

-r: reboot -h: halt -c : cancel TIME : 无指定，默认相当于+1 (CentOS7) now: 立刻,相当于+0 +#: 相对时间表示法，几分钟之后；例如 +3 hh:mm: 绝对时间表示，指明具体时间

### 1.7.3 用户登录信息查看命令

- whoami: 显示当前登录有效用户
- who: 系统当前所有的登录会话
- w: 系统当前所有的登录会话及所做的操作

### 1.7.4 文本编辑

nano 工具可以实现文本的编辑，上手容易，适合初学者

### 1.7.5 会话管理

命令行的典型使用方式是，打开一个终端窗口（terminal window，以下简称“窗口”），在里面输入命令。用户与计算机的这种临时的交互，称为一次“会话”（session）会话的一个重要特点是，窗口与其中启动的进程是连在一起的。打开窗口，会话开始；关闭窗口，会话结束，会话内部的进程也会随之终止，不管有没有运行完一个典型的例子就是，SSH 登录远程计算机，打开一个远程窗口执行命令。这时，网络突然断线，再次登录的时候，是找不回上一次执行的命令的。因为上一次 SSH 会话已经终止了，里面的进程也随之消失了。为了解决这个问题，会话与窗口可以“解绑”：窗口关闭时，会话并不终止，而是继续运行，等到以后需要的时候，再让会话“绑定”其他窗口

终端复用器软件就是会话与窗口的“解绑”工具，将它们彻底分离。（1）它允许在单个窗口中，同时访问多个会话。这对于同时运行多个命令程序很有用。（2）它可以让新窗口“接入”已经存在的会话。（3）它允许每个会话有多个连接窗口，因此可以多人实时共享会话。（4）它还支持窗口任意的垂直和水平拆分。类似的终端复用器还有Screen，Tmux

### 1.7.5.1 screen

利用screen 可以实现会话管理,如：新建会话,共享会话等

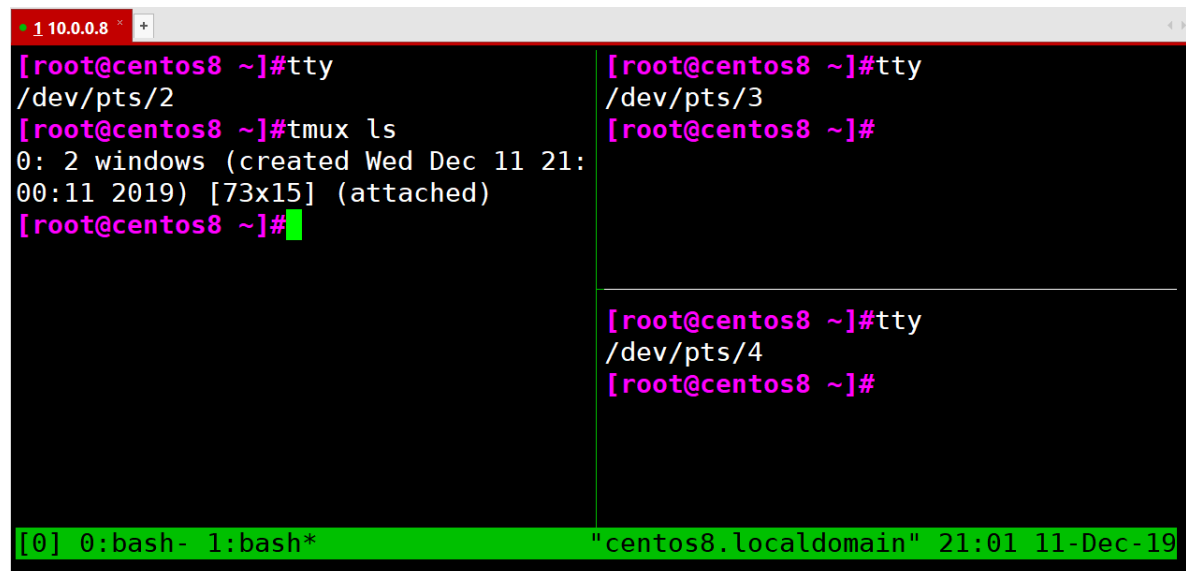
范例：安装 screen

```
[root@centos7 ~]#yum -y install screen
```

screen命令常见用法：

- 创建新screen会话 screen -S [SESSION]
- 加入screen会话 screen -x [SESSION]
- 退出并关闭screen会话 exit
- 剥离当前screen会话 Ctrl+a,d
- 显示所有已经打开的screen会话 screen -ls
- 恢复某screen会话 screen -r [SESSION]

### 1.7.5.2 tmux



```
[root@centos8 ~]#tty
/dev/pts/2
[root@centos8 ~]#tmux ls
0: 2 windows (created Wed Dec 11 21:00:11 2019) [73x15] (attached)
[root@centos8 ~]#

[root@centos8 ~]#tty
/dev/pts/3
[root@centos8 ~]#

[root@centos8 ~]#tty
/dev/pts/4
[root@centos8 ~]#

[0] 0: bash- 1: bash* "centos8.localdomain" 21:01 11-Dec-19
```

Tmux 是一个终端复用器（terminal multiplexer），类似 screen，但是更易用，也更强大

安装

```
yum install tmux
```

启动与退出

```
[root@centos8 ~]#tmux
[root@centos8 ~]#exit
logout
```

mux 窗口有大量的快捷键。所有快捷键都要通过前缀键唤起。默认的前缀键是 `Ctrl+b`，即先按下 `Ctrl+b`，快捷键才会生效。帮助命令的快捷键是 `Ctrl+b ?` 然后，按下 `q` 键，就可以退出帮助

新建会话 第一个启动的 Tmux 窗口，编号是0，第二个窗口的编号是1，以此类推。这些窗口对应的会话，就是 0 号会话、1 号会话。使用编号区分会话，不太直观，更好的方法是为会话起名。下面命令新建一个指定名称的会话。

```
tmux new -s <session-name>
```

tmux ls或Ctrl+b,s 可以查看当前所有的 Tmux 会话

```
tmux ls
tmux list-session
```

分离会话 在 Tmux 窗口中，按下Ctrl+b d或者输入tmux detach命令，就会将当前会话与窗口分离。

```
tmux detach
```

接入会话 tmux attach 命令用于重新接入某个已存在的会话。

```
tmux attach -t <session-name>
```

范例：

```
tmux attach -t 0
```

杀死会话 tmux kill-session命令用于杀死某个会话。

```
tmux kill-session -t <session-name>
```

切换会话 tmux switch命令用于切换会话

```
tmux switch -t <session-name>
```

可以将窗口分成多个窗格（pane），每个窗格运行不同的命令

上下分窗格

```
tmux split-window
ctrl+b,"
```

左右分窗格

```
tmux split-window -h
ctrl+b,%
```

窗格快捷键

Ctrl+b % : 划分左右两个窗格。 Ctrl+b " : 划分上下两个窗格。 Ctrl+b : 光标切换到其他窗格。是指向要切换到的窗格的方向键，比如切换到下方窗格，就按方向键↓。 Ctrl+b ; : 光标切换到上一个窗格。 Ctrl+b o : 光标切换到下一个窗格。 Ctrl+b { : 当前窗格左移。 Ctrl+b } : 当前窗格右移。 Ctrl+b Ctrl+o : 当前窗格上移。 Ctrl+b Alt+o : 当前窗格下移。 Ctrl+b x : 关闭当前窗格。 Ctrl+b ! : 将当前窗格拆分为一个独立窗口。 Ctrl+b z : 当前窗格全屏显示，再使用一次会变回原来大小。 Ctrl+b Ctrl+ : 按箭头方向调整窗格大小。 Ctrl+b q : 显示窗格编号

窗口管理 除了将一个窗口划分成多个窗格，Tmux 也允许新建多个窗口

新建窗口 tmux new-window命令用来创建新窗口。

```
tmux new-window
```

新建一个指定名称的窗口

```
tmux new-window -n <window-name>
```

切换窗口 tmux select-window命令用来切换窗口

切换到指定编号的窗口

```
tmux select-window -t <window-number>
```

切换到指定名称的窗口

```
tmux select-window -t <window-name>
```

窗口快捷键

Ctrl+b c : 创建一个新窗口，状态栏会显示多个窗口的信息。 Ctrl+b p : 切换到上一个窗口（按照状态栏上的顺序）。 Ctrl+b n : 切换到下一个窗口。 Ctrl+b : 切换到指定编号的窗口，其中的是状态栏上的窗口编号 Ctrl+b w : 从列表中选择窗口 Ctrl+b , : 窗口重命名

列出所有快捷键，及其对应的 Tmux 命令

```
tmux list-keys
```

列出所有 Tmux 命令及其参数

```
tmux list-commands
```

## 1.7.5 输出信息

echo 命令可以将后面跟的字符进行输出 功能：显示字符，echo会将输入的字符串送往标准输出。输出的字符串间以空白字符隔开，并在最后加上换行号 语法：

```
echo [-neE] [字符串]
```

选项：

- -E（默认）不支持\解释功能
- -n 不自动换行
- -e 启用\字符的解释功能

## 显示变量

```
echo "$VAR_NAME"    #变量会替换，弱引用
echo '$VAR_NAME'    #变量不会替换，强引用
```

启用命令选项-e，若字符串中出现以下字符，则特别加以处理，而不会将它当成一般文字输出

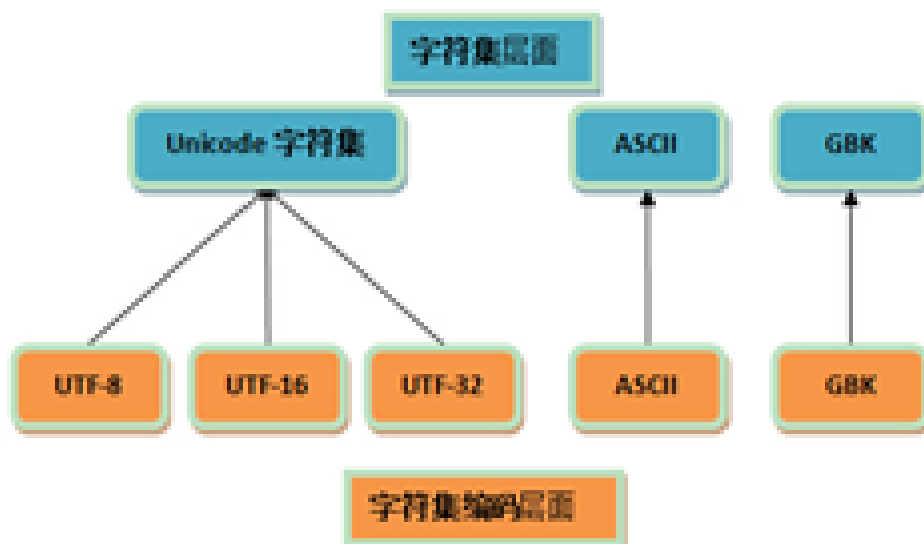
- \a 发出警告声
- \b 退格键
- \c 最后不加上换行符号
- \e escape，相当于\033
- \n 换行且光标移至行首
- \r 回车，即光标移至行首，但不换行
- \t 插入tab
- \ 插入\字符
- \0nnn 插入nnn（八进制）所代表的ASCII字符
- \xHH 插入HH（十六进制）所代表的ASCII数字（man 7 ascii）

范例：

```
echo -e '\033[43;31;5mmagedu\e[0m'
```

## 1.8 字符集和编码

许多场合下，字符集与编码这两个概念常被混为一谈，但两者是有差别的。字符集与字符集编码是两个不同层面的概念。charset是character set的简写，即字符集。encoding是charset encoding的简写，即字符集编码，简称编码



### 1.8.1 ASCII码

计算机内部，所有信息最终都是一个二进制值。上个世纪60年代，美国制定了一套字符编码，对英语字符与二进制位之间的关系，做了统一规定，即ASCII（American Standard Code for Information Interchange）码

ASCII 码一共规定了128个字符的编码，占用了一个字节的后面7位，最前面的一位统一规定为0

ASCII 字符代码表 一

高四位	ASCII非打印控制字符										ASCII 打印字符														
	0000					0001					0010	0011	0100	0101	0110	0111									
	0					1					2	3	4	5	6	7									
低四位	+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl	代码	字符解释	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	ctrl
0000	0	0	BLANK NULL	^@	NUL 空	16	▶	^P	DLE 数据链路转意	32		48	0	64	@	80	P	96	`	112	p				
0001	1	1	☺	^A	SOH 头标开始	17	◀	^Q	DC1 设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q				
0010	2	2	☹	^B	STX 正文开始	18	↕	^R	DC2 设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r				
0011	3	3	♥	^C	ETX 正文结束	19	!!	^S	DC3 设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s				
0100	4	4	♦	^D	EOF 传输结束	20	¶	^T	DC4 设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t				
0101	5	5	♣	^E	ENQ 查询	21	⌘	^U	NAK 反确认	37	%	53	5	69	E	85	U	101	e	117	u				
0110	6	6	♠	^F	ACK 确认	22	■	^V	SYN 同步空闲	38	&	54	6	70	F	86	V	102	f	118	v				
0111	7	7	●	^G	BEL 震铃	23	↕	^W	ETB 传输块结束	39	'	55	7	71	G	87	w	103	g	119	w				
1000	8	8	◻	^H	BS 退格	24	↑	^X	CAN 取消	40	(	56	8	72	H	88	X	104	h	120	x				
1001	9	9	○	^I	TAB 水平制表符	25	↓	^Y	EM 媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y				
1010	A	10	◻	^J	LF 换行/新行	26	→	^Z	SUB 替换	42	*	58	:	74	J	90	Z	106	j	122	z				
1011	B	11	♂	^K	VT 垂直制表符	27	←	^[	ESC 转意	43	+	59	;	75	K	91	[	107	k	123	{				
1100	C	12	♀	^L	FF 换页/新页	28	└	^[_	FS 文件分隔符	44	,	60	<	76	L	92	\	108	l	124					
1101	D	13	♂	^M	CR 回车	29	↔	^J	GS 组分隔符	45	-	61	=	77	M	93	]	109	m	125	}				
1110	E	14	♂	^N	SO 移出	30	▲	^G	RS 记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~				
1111	F	15	♂	^O	SI 移入	31	▼	^_	US 单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ				Back space

注：表中的ASCII字符可以用:ALT + “小键盘上的数字键”输入

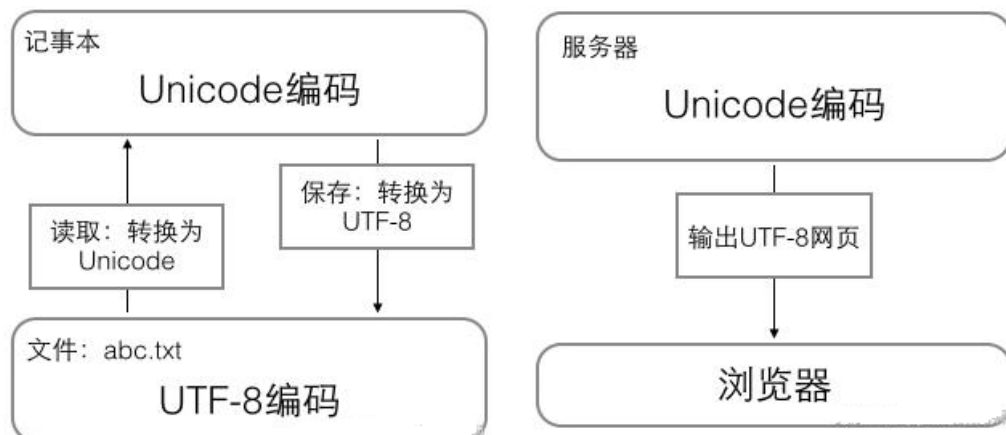
### 1.8.2 Unicode

由于计算机是美国人发明的，因此，最早只有127个字母被编码到计算机里，即ASCII编码，但是要处理中文显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。全世界有上百种语言，日本把日文编到Shift\_JIS里，韩国把韩文编到Euc-kr里，各国有各国的标准，就会不可避免地出现冲突，结果就是，在多语言混合的文本中，显示出来会有乱码

为了表示世界上所有语言中的所有字符。每一个符号都给予一个独一无二的编码数字，Unicode 是一个很大的集合，现在的规模可以容纳100多万个符号。Unicode 仅仅只是一个字符集，规定了每个字符对应的二进制代码，至于这个二进制代码如何存储则没有规定

#### Unicode编码方案：

- UTF-8：变长，1到4个字节
- UTF-16：变长，2或4个字节
- UTF-32：固定长度，4个字节



UTF-8 是目前互联网上使用最广泛的一种 Unicode 编码方式，可变长存储。使用 1 - 4 个字节表示一个字符，根据字符的不同变换长度。编码规则如下：对于单个字节的字符，第一位设为 0，后面的 7 位对应这个字符的 Unicode 码。因此，对于英文中的 0 - 127 号字符，与 ASCII 码完全相同。这意味着 ASCII 码的文档可用 UTF-8 编码打开 对于需要使用 N 个字节来表示的字符 (N > 1)，第一个字节的前

N 位都设为 1，第 N + 1 位设为 0，剩余的 N - 1 个字节的前两位都设位 10，剩下的二进制位则使用这个字符的 Unicode 码来填充

**编码转换和查询：** <http://www.chi2ko.com/tool/CJK.htm> <https://javawind.net/tools/native2ascii.jsp?action=transform> <http://tool.oschina.net/encode>

## Unicode和UTF-8

Unicode符号范围(十六进制)	UTF-8编码方式二进制 )
0000 0000-0000 007F	0xxxxxxx
0000 0080-0000 07FF	110xxxxx 10xxxxxx
0000 0800-0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000-0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

范例：

“汉”的 Unicode 码 0x6C49 ( 110 110001 001001 )，需要三个字节存储，格式为：1110xxxx 10xxxxxx 10xxxxxx，从后向前依次填充对应格式中的 x，多出的 x 用 0 补，得出 UTF-8 编码为 11100110 10110001 10001001 “马”的 Unicode 码 0x9A6C ( 1001 101001 101100 )，需要三个字节存储，格式为：1110xxxx 10xxxxxx 10xxxxxx，从后向前依次填充对应格式中的 x，多出的 x 用 0 补，得出 UTF-8 编码为 11101001 10101001 10101100

范例：修改LANG变量实现中文语言提示

```
[root@centos7 ~]#xxx
bash: xxx: command not found...
[root@centos7 ~]#echo $LANG
en_US.UTF-8
[root@centos7 ~]#xxx
bash: xxx: command not found...
[root@centos7 ~]#LANG=zh_CN.UTF-8
[root@centos7 ~]#echo $LANG
zh_CN.UTF-8
[root@centos7 ~]#xxx
bash: xxx: 未找到命令...
```

## 1.9 命令行扩展和被括起来的集合

### 1.9.1 命令行扩展：\$( )

把一个命令的输出打印给另一个命令的参数

```
$(CMD) 或 `CMD`
```

范例：

```
echo "This system's name is $(hostname) "
This system's name is server1.example.com
echo "i am `whoami` "
i am root
```

### 1.9.2 括号扩展：{ }

{ } 可以实现打印重复字符串的简化形式

范例：

```
echo file{1,3,5} 结果为: file1 file3 file5
rm -f file{1,3,5}
echo {1..10}
echo {a..z}
echo {000..20..2}
```

## 1.10 tab 键补全

tab键可以实现命令及路径等补全，提高输入效率，避免出错

### 1.10.1 命令补全

- 内部命令：
- 外部命令：bash根据PATH环境变量定义的路径，自左而右在每个路径搜寻以给定命令命名的文件，第一次找到的命令即为要执行的命令

注意：用户给定的字符串只有一条唯一对应的命令，直接补全，否则，再次Tab会给出列表

### 1.10.2 路径补全

把用户给出的字符串当做路径开头，并在其指定上级目录下搜索以指定的字符串开头的文件名 如果惟一：则直接补全 否则：再次Tab给出列表

### 1.10.3 双击Tab键

- command 2Tab 所有子命令或文件补全
- string 2Tab 以string开头命令
- /2Tab 显示所有根目录下一级目录，包括隐藏目录
- ./2Tab 当前目录下子目录，包括隐藏目录
- \*2Tab 当前目录下子目录，不包括隐藏目录
- ~2Tab 所有用户列表
- \$2Tab 所有变量
- @2Tab /etc/hosts记录（centos7不支持）
- =2Tab 相当于ls -A（centos7不支持）

## 1.11 命令行历史

保存你输入的命令历史。可以用它来重复执行命令 登录shell时，会读取命令历史文件中记录下的命令 ~/.bash\_history 登录进shell后新执行的命令只会记录在缓存中；这些命令会用户退出时“追加”至命令历史文件中

命令：history

```
history [-c] [-d offset] [n]
history -anrw [filename]
history -ps arg [arg...]
```

- -c: 清空命令历史
- -d offset: 删除历史中指定的第offset个命令
- n: 显示最近的n条历史
- -a: 追加本次会话新执行的命令历史列表至历史文件



- -r: 读历史文件附加到历史列表
- -w: 保存历史列表到指定的历史文件
- -n: 读历史文件中未读过的行到历史列表
- -p: 展开历史参数成多行, 但不存在历史列表中
- -s: 展开历史参数成一行, 附加在历史列表后

### 命令历史相关环境变量

- HISTSIZE : 命令历史记录的条数
- HISTFILE : 指定历史文件, 默认为 ~/.bash\_history
- HISTFILESIZE : 命令历史文件记录历史的条数
- HISTTIMEFORMAT="%F %T " 显示时间
- HISTIGNORE="str1:str2\*:..." 忽略str1命令, str2开头的历史
- HISTCONTROL : 控制命令历史的记录方式 ignoredups 是默认值, 可忽略重复的命令, 连续且相同为“重复” ignorespace 忽略所有以空白开头的命令 ignoreboth 相当于ignoredups, ignorespace 的组合 erasedups 删除重复命令

### 持久保存变量

- 以上变量可以 export 变量名="值" 形式存放在 /etc/profile 或 ~/.bash\_profile

范例：

```
[root@centos8 ~]#cat .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
export HISTCONTROL=ignoreboth
export HISTTIMEFORMAT="%F %T "

[root@centos8 ~]#history
 1 2019-12-13 08:39:05 ls /data
 2 2019-12-13 08:39:05 date
 3 2019-12-13 08:39:05 vie0
 4 2019-12-13 08:39:05 nano .bash_profile
 5 2019-12-13 08:39:05 exit
```

## 1.12 调用命令行历史

### #重复前一个命令方法

重复前一个命令使用上方向键, 并回车执行

按 !! 并回车执行

输入 !-1 并回车执行

按 Ctrl+p 并回车执行

!:0 执行前一条命令 (去除参数)

```

!n  执行history命令输出对应序号n的命令
!-n  执行history历史中倒数第n个命令
!string  重复前一个以“string”开头的命令
!?string  重复前一个包含string的命令
!string:p  仅打印命令历史，而不执行
!$打印输出:p 打印输出 !$ （上一条命令的最后一个参数）的内容
!*:p 打印输出 !*（上一条命令的所有参数）的内容
^string  删除上一条命令中的第一个string
^string1^string2 将上一条命令中的第一个string1替换为string2
!:gs/string1/string2 将上一条命令中所有的string1都替换为 string2
使用up（向上）和down（向下）键来上下浏览从前输入的命令
ctrl-r来在命令历史中搜索命令
（reverse-i-search）`:
Ctrl+g: 从历史搜索模式退出
#要重新调用前一个命令中最后一个参数
!$表示
Esc, .  点击Esc键后松开，然后点击 . 键
Alt+.  按住Alt键的同时点击 . 键
command !^  利用上一个命令的第一个参数做cmd的参数
command !$  利用上一个命令的最后一个参数做cmd的参数
command !*  利用上一个命令的全部参数做cmd的参数
command !:n  利用上一个命令的第n个参数做cmd的参数
command !n:^  调用第n条命令的第一个参数
command !n:$  调用第n条命令的最后一个参数
command !n:m  调用第n条命令的第m个参数
command !n:*  调用第n条命令的所有参数
command !string:^  从命令历史中搜索以 string 开头的命令，并获取它的第一个参数
command !string:$  从命令历史中搜索以 string 开头的命令,并获取它的最后一个参数
command !string:n  从命令历史中搜索以 string 开头的命令，并获取它的第n个参数
command !string:*  从命令历史中搜索以 string 开头的命令，并获取它的所有参数

```

## 1.13 bash的快捷键

Ctrl + l 清屏，相当于clear命令 Ctrl + o 执行当前命令，并重新显示本命令 Ctrl + s 阻止屏幕输出，锁定 Ctrl + q 允许屏幕输出 Ctrl + c 终止命令 Ctrl + z 挂起命令 Ctrl + a 光标移到命令行首，相当于Home Ctrl + e 光标移到命令行尾，相当于End Ctrl + f 光标向右移动一个字符 Ctrl + b 光标向左移动一个字符 Alt + f 光标向右移动一个单词尾 Alt + b 光标向左移动一个单词首 Ctrl + xx 光标在命令行首和光标之间移动 Ctrl + u 从光标处删除至命令行首 Ctrl + k 从光标处删除至命令行尾 Alt + r 删除当前整行 Ctrl + w 从光标处向左删除至单词首 Alt + d 从光标处向右删除至单词尾 Ctrl + d 删除光标处的一个字符 Ctrl + h 删除光标前的一个字符 Ctrl + y 将删除的字符粘贴至光标后 Alt + c 从光标处开始向右更改为首字母大写的单词 Alt + u 从光标处开始，将右边一个单词更改为大写 Alt + l 从光标处开始，将右边一个单词更改为小写 Ctrl + t 交换光标处和之前的字符位置 Alt + t 交换光标处和之前的单词位置 Alt + # 提示输入指定字符后，重复显示该字符#次 注意：Alt组合快捷键经常和其它软件冲突

## 2 获得帮助

获取帮助的能力决定了技术的能力！

多层次的帮助

- whatis
- command --help
- man and info
- /usr/share/doc/
- Red Hat documentation

- 其它网站和搜索

## 2.1 whatis

whatis 使用数据库来显示命令的简短描述 刚安装后不可立即使用，需要制作数据库

```
#CentOS 7 版本以后
mandb
#CentOS 6 版本之前
makewhatis
```

范例：

```
whatis cal
man -f cal
```

范例：

```
[root@centos8 ~]#whatis ls
ls: nothing appropriate.
#生成man相关数据库
[root@centos8 ~]#mandb
Processing manual pages under /usr/share/man...
Updating index cache for path `/usr/share/man/mann'. Wait...done.
Checking for stray cats under /usr/share/man...
...省略...
0 old database entries were purged.
[root@centos8 ~]#whatis ls
ls (1)                - list directory contents
```

## 2.2 查看命令的帮助

内部命令：

- help COMMAND
- man bash

外部命令和软件：

- COMMAND --help 或 COMMAND -h
- 使用手册(manual)  
man COMMAND
- 信息页 info COMMAND
- 程序自身的帮助文档 README INSTALL ChangeLog
- 程序官方文档 官方网站：Documentation
- 发行版的官方文档
- (7) Google

## 2.3 --help或-h 选项

显示用法总结和参数列表，使用的大多数，但并非所有的

范例：

```
[root@centos8 ~]#date --help
Usage: date [OPTION]... [+FORMAT]
  or:  date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
Display the current time in the given FORMAT, or set the system date.
```

格式说明：

- [] 表示可选项
- CAPS或 <> 表示变化的数据
- ... 表示一个列表
- x |y| z 的意思是“x 或 y 或 z”
- -abc的意思是-a -b -c
- {} 表示分组

范例：1、显示当前时间，格式：2016-06-18 10:20:30 2、显示前天是星期几 3、设置当前日期为2019-08-07 06:05:10

## 2.4 man命令

man 提供命令帮助的文件,手册页存放在/usr/share/man

几乎每个命令都有man的“页面”

中文man需安装包man-pages-zh-CN

man页面分组为不同的“章节”,统称为Linux手册，man 1 man

- 1：用户命令
- 2：系统调用
- 3：C库调用
- 4：设备文件及特殊文件
- 5：配置文件格式
- 6：游戏
- 7：杂项
- 8：管理类的命令
- 9：Linux 内核API

man命令的配置文件：

```
/etc/man.config
/etc/man_db.conf
```

格式：

```
MANPATH /PATH/TO/SOMEWHERE #指明man文件搜索位置
```

也可以指定位置下搜索COMMAND命令的手册页并显示

```
man -M /PATH/TO/SOMEWHERE COMMAND
```

查看man手册页

```
man [章节] keyword
```

## man 帮助段落说明

- NAME 名称及简要说明
- SYNOPSIS 用法格式说明
- [] 可选内容
- <> 必选内容
- a|b 二选一
- {} 分组
- ... 同一内容可出现多次
- DESCRIPTION 详细说明
- OPTIONS 选项说明
- EXAMPLES 示例
- FILES 相关文件
- AUTHOR 作者
- COPYRIGHT 版本信息
- REPORTING BUGS bug信息
- SEE ALSO 其它帮助参考

## 列出所有帮助

```
man -a keyword
```

## 搜索man手册

```
man -k keyword #列出所有匹配的页面，使用 whatis 数据库
```

## 相当于whatis

```
man -f keyword
```

## 打印man帮助文件的路径

```
man -w [章节] keyword
```

## man命令的操作方法：使用less命令实现

- space, ^v, ^f, ^F: 向文件尾翻页
- b, ^b: 向文件首部翻页
- d, ^d: 向文件尾部翻半屏
- u, ^u: 向文件首部翻半屏
- RETURN, ^N, e, ^E or j or ^J: 向文件尾部翻一行
- y or ^Y or ^P or k or ^K : 向文件首部翻一行
- q: 退出
- #: 跳转至第#行
- 1G: 回到文件首部
- G: 翻至文件尾部
- /KEYWORD 以KEYWORD指定的字符串为关键字，从当前位置向文件尾部搜索；不区分字符大小写 n: 下一个 N: 上一个
- ?KEYWORD 以KEYWORD指定的字符串为关键字，从当前位置向文件首部搜索；不区分字符大小写 n: 跟搜索命令同方向，下一个 N: 跟搜索命令反方向，上一个

范例：1、在本机字符终端登录时，除显示原有信息外，再显示当前登录终端号，主机名和当前时间  
2、今天18：30自动关机，并提示用户

## 2.5 info

man常用于命令参考，GNU工具info适合通用文档参考 没有参数,列出所有的页面 info 页面的结构就像一个网站 每一页分为“节点” 链接节点之前 \*

info 命令格式

```
info [ 命令 ]
```

导航info页

- 方向键，PgUp，PgDn 导航
- Tab键 移动到下一个链接
- d 显示主题目录
- Home 显示主题首部
- Enter进入 选定链接
- n/p/u/l 进入下/前/上一层/最后一个链接
- s 文字 文本搜索
- q 退出 info

## 2.6 Linux 安装提供的本地文档获取帮助

Applications -> documentation->help ( centos7 )

System->help ( centos6 )

## 2.7 命令自身提供的官方使用指南

/usr/share/doc目录

多数安装了的软件包的子目录,包括了这些软件的相关原理说明 常见文档：README INSTALL CHANGES 不适合其它地方的文档的位置 配置文件范例 HTML/PDF/PS 格式的文档 授权书详情

## 2.8 系统及第三方应用官方文档

### 2.8.1通过在线文档获取帮助

<http://httpd.apache.org> <http://www.nginx.org> <https://mariadb.com/kb/en> <https://dev.mysql.com/doc/> <http://tomcat.apache.org> <http://www.python.org>

### 2.8.2 红帽知识库和官方在线文档

通过发行版官方的文档光盘或网站可以获得安装指南、部署指南、虚拟化指南等

<http://kbase.redhat.com> <http://www.redhat.com/docs> <http://access.redhat.com> <https://help.ubuntu.com/lts/serverguide/index.html>

### 2.8.3 红帽全球技术支持服务

rhnsite.redhat.com或者本地卫星服务器/代理服务器 RHN账户为及其注册和基于网络管理的RHN用户 sosreport 收集所有系统上的日志信息的工具，并自动打成压缩包，方便技术支持人员和红帽全球支持提供分析问题依据

## 2.9 网站和搜索

<http://tldp.org> <http://www.slideshare.net> <http://www.google.com> Openstack filetype:pdf rhnsite:redhat.com/docs