Linux基础入门和帮助

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

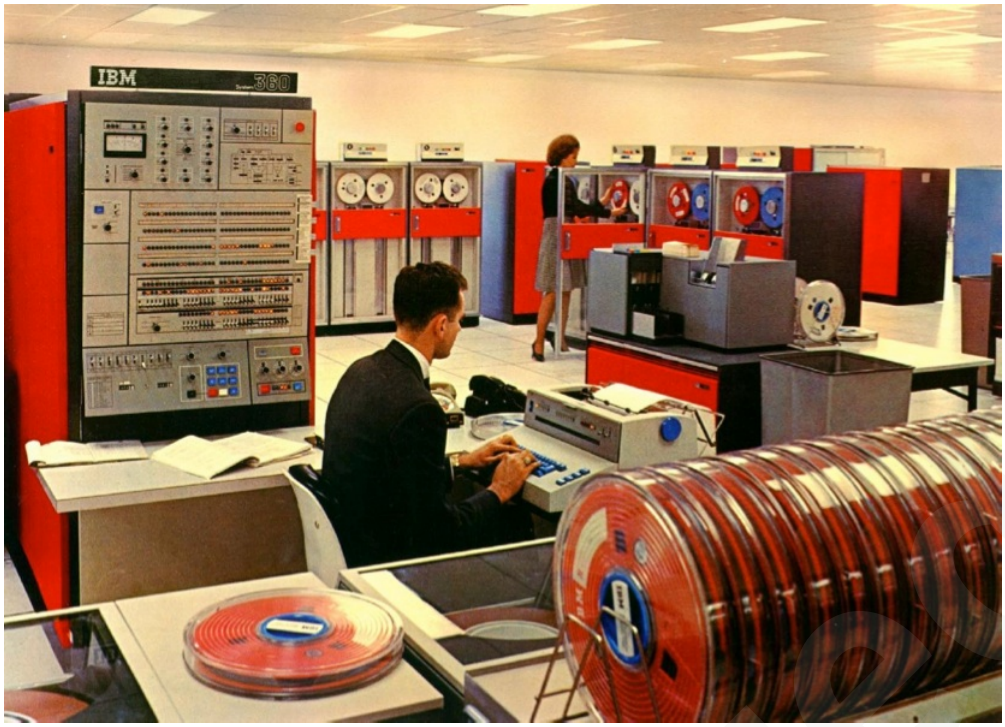
- 普通（非特权）用户

  权限有限

  造成损害的能力比较有限

1.2 终端 terminal

终端（英语：Computer terminal），是与计算机系统相连的一种输入输出设备，它用来显示主机运算的输出，并且接受主机要求的输入，通常离计算机较远。根据功能不同，可分若干类。典型的终端包括显示器键盘套件，打印机打字机套件等。


IBM
360

随着技术的发展，控制台，终端这些不再是单独的物理设备了，而是被键盘和显示器整合，替代。现在计算机的console控制台和tty终端都是虚拟出来的概念了，在概念上，键盘和显示器既是console，又是tty。至于什么时候是console，什么时候是tty，取决于那一刻在做什么。

这里的虚拟，指的是，Linux操作系统中，以一种设备文件的方式保留tty和console(一切皆文件)。当使用特定软件连接该主机时，就能看到当前连接所占用的终端设备文件，这样就表示该机器的一个终端被激活，正在使用中。

tty一词源于Teletypes，或者teletypewriters，原来指的是电传打字机，是通过串行线用打印机键盘通过阅读和发送信息的东西，后来这东西被键盘与显示器取代，所以现在叫终端比较合适

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#

伪终端：pty：pseudo-tty ， /dev/pts/# 如：SSH远程连接

图形终端：startx, xwindows

1.2.2 查看当前的终端设备

tty 命令可以查看当前所在终端

范例：

```
root@ubuntu2204:~# 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

  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

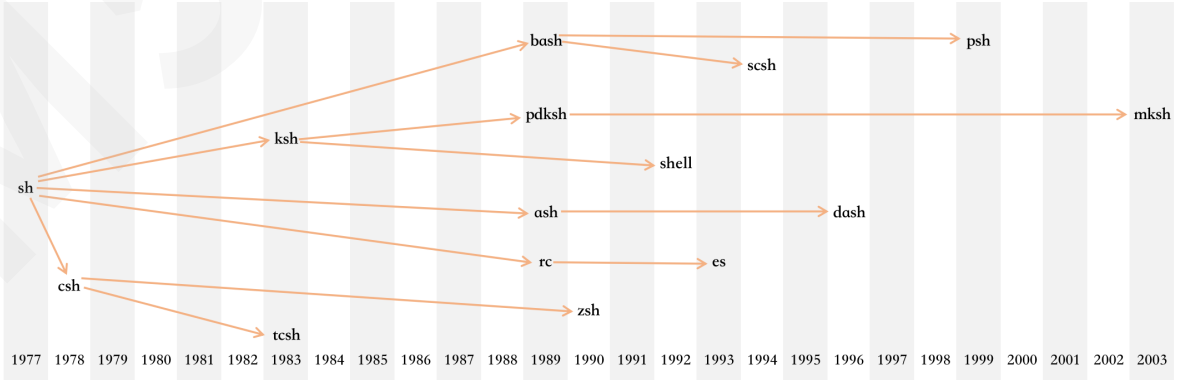
Shell 是Linux系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行 。

shell也被称为LINUX的命令解释器（command interpreter），Shell 本身是一个程序。将用户输入的命令行拆解为"命令名"与"参数"。接着，根据命令名找到对应要执行的程序，对被执行的程序进行初始化，然后将刚才解析出来的参数传给该程序并执行

shell是一种高级程序设计语言，提供了变量，函数，条件判断，循环等开发语言的功能。

由于Shell本身是个程序，所以它可以被任何用户自己开发的各种Shell所代替。

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

```
root@ubuntu2204:~# echo ${SHELL}
/bin/bash
```

显示当前系统使用的所有shell

```
root@ubuntu2204:~# cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/usr/bin/sh
/bin/dash
/usr/bin/dash
/usr/bin/tmux
/usr/bin/screen
```

```
[root@rocy8-1 ~]# cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
```

1.4 设置主机名

```
#临时生效
hostname NAME

#持久生效,支持CentOS7和Ubuntu18.04以上版本
hostnamectl set-hostname NAME
```

范例

```
root@ubuntu2204:~# hostname bj-yz-k8s-node1-100-10.magedu.org
```

注意:

- 修改hostname 需要root权限
- 主机名不支持使用下划线，但支持横线，可使用字母，横线或数字组合
- 有些软件对主机名有特殊要求
- 一般按照组织的要求设置主机名，通常有一定的意义的

范例：错误的主机名可能会导致某些服务无法启动

```
[root@rocky8 ~]# hostnamectl set-hostname centos8.3
[root@rocky8 ~]# systemctl restart postfix
Job for postfix.service failed because the control process exited with error
code.
See "systemctl status postfix.service" and "journalctl -xe" for details.
```

1.5 命令提示符 prompt

登录Linux后，默认的系统命令提示符毫无个性，无法明显辨别生产和测试环境，而导致误操作。 可以通过修改PS1变量实现个性的提示符格式，避免这种低级错误

范例：默认的提示符

```
#Rocky默认提示符
[root@rocky86 ~]#

#Ubuntu默认提示符
root@ubuntu22:~#
```

管理员

$ 普通用户

```
echo $PS1 #查看当前命令提示符
[\u@\h \W]\$



#如果以root用户登录主机，则默认提示符为，对应的就是 上面的 [\u@\h \W]\$
[root@rocky86 ~]#

#PS1中的值要单引号引用,如果是双引号，则某些替换符不会被解析
PS1='\e[31m[\u@\h \W]\$\e[0m'


#如果要永久保存，则要写文件
echo "PS1='\e[31;1m[\u@\h \W]\\$ \e[0m'">/etc/profile.d/env.sh
source /etc/profile.d/env.sh


#PS1变量中的常用选项
\d      #日期，格式为"星期 月 日"
\H      #完整的主机名。如默认主机名"localhost.localdomain"。
\h      #简写的主机名。如默认主机名"localhost"。
\t      #24小时制时间，格式为"HH:MM:SS"。
\T      #12小时制时间，格式为"HH:MM:SS"。
\A      #24小时制时间，格式为"HH:MM"。
\@      #12小时制时间，格式为"HH:MM am/pm"。
\u      #当前用户名。
\v      #Bash版本信息。
\w      #当前所在目录的完整名称。
\W      #当前所在目录的最后一个目录。
\#      #执行的第几条命令。
\$      #提示符。如果是 root 用户，则会显示提示符为"#"；如果是普通用户，则会显示提示符为"$"


#PS1中的颜色部份
\033[     #开始位
\033[0m   #结束位

#上面的\033 可以换成 \e
```

```
#颜色设置 \e[  颜色m    内容     \e[0m     颜色后面要用m结束，多个颜色，用；分割，只要一个m
\e[31m我是红色\e[0m


#各颜色表示方式
字体颜色    背景色
30          40          黑色
31          41          红色
32          42          绿色
33          43          黄色
34          44          蓝色
35          45          紫红色
36          46          青蓝色
37          47          白色


#字体颜色
\e[30m我是黑色\e[0m
\e[31m我是红色\e[0m
\e[32m我是绿色\e[0m
\e[33m我是黄色\e[0m
\e[34m我是蓝色\e[0m
\e[35m我是紫红色\e[0m
\e[36m我是青蓝色\e[0m
\e[37m我是白色\e[0m


#背景颜色
\e[40m我背景是黑色\e[0m
\e[41m我背景是红色\e[0m
\e[42m我背景是绿色\e[0m
\e[43m我背景是黄色\e[0m
\e[44m我背景是蓝色\e[0m
\e[45m我背景是紫红色\e[0m
\e[46m我背景是青蓝色\e[0m
\e[47m我背景是白色\e[0m


#字体颜色+背景颜色组合，字体颜色和背景颜色不分先后，因为值不一样
\e[30;41m红底黑字\e[0m
\e[41;30m红底黑字\e[0m


#特殊效果
0    #关闭效果
1    #高亮显示
3    #斜体
4    #下划线
5    #闪烁,闪烁效果与远程工具所在的环境有关
7    #反白显示
8    #隐藏
9    #删除线

#可组合使用，但如果效果有冲突时，以最后的为准，就是说，后面的效果，会覆盖前面的效果
```

```
\e[30;41;5m红底黑字闪烁\e[0m
\e[4;30;41;5m红底黑字下划线闪烁\e[0m

\e[0;4;30;41;5m红底黑字下划线闪烁\e[0m  #这里的0;4;5 都表示效果，但后面的4;5覆盖了前面的0
\e[4;30;41;5;0m红底黑字下划线闪烁\e[0m  #这样写，4;5的效果都被0去掉了
```

修改提示符格式范例

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


echo "PS1='\e[31;1m[\u@\h \w]\\$ \e[0m'" > /etc/profile.d/env.sh
```

范例：在CentOS系统实现持久保存提示符格式

```
[root@rocky8 ~]# 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@rocky8 ~]# 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@rocky8 ~]# exit


[root@rocky8 ~]# echo "PS1='\e[31;1m[\u@\h \w]\\$ \e[0m'" >
/etc/profile.d/env.sh
```

```
[21:55:45 root@rocky8 ~]#echo $PS1
\[\e[1;32m\][\t \[\e[1;33m\]\u\[\e[35m\]@\h\[\e[1;31m\] \W\[\e[1;32m\]]\[\e[0m\]\$
```

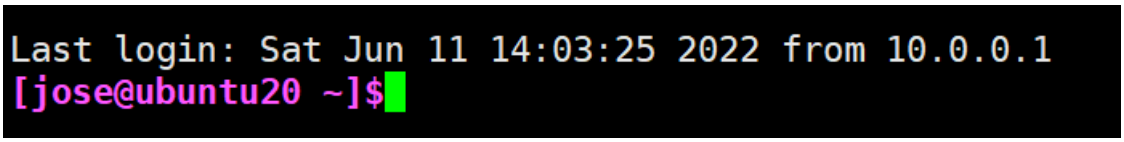
```
#ubuntu2204中PS1
root@ubuntu2204:~# echo $PS1
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\u@\h:\w\$


#       ${debian_chroot:+($debian_chroot)}  表示给变量赋值，如果存在
/etc/debian_chroot 文件，则该文件中内容会被赋值给变量$debian_chroot
```

```
#文件不存在
root@ubuntu2204:~# ls /etc/debian_chroot
ls: cannot access '/etc/debian_chroot': No such file or directory

#变量为空
root@ubuntu2204:~# echo $debian_chroot


#设置文件内容
root@ubuntu2204:~# echo "test" > /etc/debian_chroot

#文件存在
root@ubuntu2204:~# ls /etc/debian_chroot
/etc/debian_chroot

#查看文件内容
root@ubuntu2204:~# cat /etc/debian_chroot
test



#在另一个终端中登录，提示符多左边多了内容，变量也有了
(test)root@ubuntu2204:~# echo $debian_chroot
test
```

在 ubuntu2204 中，设置 PS1，如果要对所有普能用户生效，将 PS1 定义写在 /usr/share/bash-completion/bash_completion 的最下面，如里需要对每个用户单独定义，写在用户家目录的 .bashrc 文件的最下面。

范例：实现Ubuntu系统持久保存提示符格式

```
jose@ubuntu20:~$  echo "PS1='\[\e[1;35m\][\u@\h \w]\\$\[\e[0m\]'" >> .bashrc
jose@ubuntu20:~$  tail -1 .bashrc
PS1='\[\e[1;35m\][\u@\h \w]\$\[\e[0m\] '


mage@ubuntu:~$  echo "PS1='\e[31;1m[\u@\h:\w]\$\e[0m '" >> .bashrc
```



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，用户登录后自动加载并常驻内存中
- 外部命令：在文件系统路径下有对应的可执行程序文件，当执行命令时才从磁盘加载至内存中，执行完毕后从内存中删除

区别指定的命令是内部或外部命令

```
type COMMAND


type ls          #区分是内部命令还是外部命令
type -t echo     #简写，只给出类型，builtin | alias | file|keyword
type -a echo     #列出所有，有可能是内部命令，也同时会是外部命令
bash -c help     #查看bash中所有内容（不仅仅是内部命令）
help             #查看bash中所有内容（不仅仅是内部命令）
enable           #查看bash中所有内置命令
help echo        #查看内部命令帮助
```

范例: 查看是否存在对应内部和外部命令

```
root@ubuntu2204:~# type echo
echo is a shell builtin

root@ubuntu2204:~# type -a echo
echo is a shell builtin
echo is /usr/bin/echo
echo is /bin/echo
```

有内部命令，又有外部命令，因为不是所有主机都使用标准shell，所以常用内部命令会有一个外部命令的备份，防止内部命令执行失败。

在命令执行时，shell 先判断是否是内部命令，如果是，则执行内部命令，如果不是，则去特定目录下寻找外部命令。

bash shell 自身就是一个程序，里面有很多小工具，有用户通过终端连接主机，则该终端就有一个bash在后台运行着。

1.6.2.1 内部命令相关

```
help            #查看所有内部命令及帮助
enable          #查看所有启用的内部命令
enable cmd      #启用 cmd 命令
enable -n cmd   #禁用内部 cmd 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
```

范例

```
[root@rokcy8 ~]# which ls
alias ls='ls --color=auto'
    /usr/bin/ls


#ubuntu中的 which 不显示别名
root@ubuntu2204:~# which ls
/usr/bin/ls


#跳过别名
[root@rokcy8 ~]# which --skip-alias ls
/usr/bin/ls



# -a 表示在所有路径中查找匹配的命令，使用该命令时匹配第一个找到的
[root@rokcy8 ~]# which -a echo
/usr/bin/echo


#查看path
[root@rokcy8 ~]# echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin


#创建root下的 bin 目录
[root@rokcy8 ~]# mkdir bin

#拷贝 echo 命令到 /root/bin/ 目录下
[root@rokcy8 ~]# cp /usr/bin/echo /root/bin/


#再次查看

[root@rokcy8 ~]# which echo
/usr/bin/echo

[root@rokcy8 ~]# which -a echo
/usr/bin/echo
/root/bin/echo



#ubuntu中
root@ubuntu2204:~# which -a echo
/usr/bin/echo
/bin/echo
```

外部命令搜索路径

```
#root用户
[root@rokcy8 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

#普通用户
[root@rokcy8 ~]# su - mage
[mage@rokcy8 ~]$ echo $PATH
/home/mage/.local/bin:/home/mage/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/us
r/sbin
```

```
#root用户
root@ubuntu2204:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/snap/bin

#普通用户
root@ubuntu2204:~# su - mage
mage@ubuntu2204:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/snap/bin
```

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，找到后会将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 只对当前用户的当前终端进程有效，是一组临时数据；

切换用户后无效；

退出重新登录后会被重置；

hash 命令常见用法

```
hash                #显示当前终端进程中的hash 缓存
hash -l             #显示详创建此条hash 的命令，可作为输入使用
hash -p path name   #手动创建hash
hash -t name        #输出路径
hash -d name        #删除指定hash
hash -r             #清空所有hash
```

范例：

```
root@ubuntu2204:~# hash
hits    command
   1    /usr/bin/mesg
   2    /usr/bin/which
   1    /usr/bin/su
   3    /usr/bin/ls


root@ubuntu2204:~# hash -l
builtin hash -p /usr/bin/mesg mesg
builtin hash -p /usr/bin/which which
```

```
builtin hash -p /usr/bin/su su
builtin hash -p /usr/bin/ls ls


#手动创建hash
root@ubuntu2204:~# builtin hash -p /abc abc
root@ubuntu2204:~# hash
hits    command
   1    /usr/bin/mesg
   2    /usr/bin/which
   0    /abc
   1    /usr/bin/su
   3    /usr/bin/ls


#输出路径
root@ubuntu2204:~# hash -t su
/usr/bin/su

#删除指定hash
root@ubuntu2204:~# hash -d su

#清空所有hash
root@ubuntu2204:~# hash -r
```

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

```
alias                   #显示当前shell进程所有可用的命令别名
alias name              #查看指定别名
alias NAME='VALUE'      #定义别名NAME，其相当于执行命令VALUE
unalias                 #撤消别名
```

范例：

```
#列出所有别名
root@ubuntu2204:~# alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

```
#查看指定别名
root@ubuntu2204:~# alias ls
alias ls='ls --color=auto'


root@ubuntu2204:~# pwd
/root

#定义别名
root@ubuntu2204:~# alias cdnet='cd /etc/systemd/network'

#执行
root@ubuntu2204:~# cdnet

#查看当前目录
root@ubuntu2204:/etc/systemd/network# pwd
/etc/systemd/network



#撤销别名
root@ubuntu2204:/etc/systemd/network# unalias cdnet

root@ubuntu2204:/etc/systemd/network# cdnet
cdnet: command not found


#撤销所有别名
root@ubuntu2204:~# unalias -a
root@ubuntu2204:~# alias
root@ubuntu2204:~#
```

范例: 扫描新加的磁盘

```
[root@rocky8 ~]# alias scandisk='echo - - - >
/sys/class/scsi_host/host0/scan;echo - - - > /sys/class/scsi_host/host1/scan;echo
- - - > /sys/class/scsi_host/host2/scan'
```

在命令行中定义的别名，仅对当前shell进程有效

如里需要永久有效，则要写配置文件：

- 仅对当前用户：~/.bashrc
- 对所有用户有效：/etc/bashrc， /etc/bash.bashrc(ubuntu)

编辑配置文件新加的别名不会立即生效，要退出重新登录或在当前进程中重新读取配置文件

```
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/commmand    #只适用于外部命令
```

范例：

```
root@ubuntu2204:~# alias ls
alias ls='ls --color=auto'

#执行别名
root@ubuntu2204:~# ls

#执行原生命令
root@ubuntu2204:~# \ls

#执行原生命令
root@ubuntu2204:~# /usr/bin/ls
```

命令执行优先级

别名 -----> 内部命令 ------>hash--->外部命令

1.6.4 命令格式

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


COMMAND      #命令
OPTIONS      #选项，用于启用或关闭命令的某个或某些功能
ARGUMENTS    #参数，命令的作用对象，比如:文件名，用户名等
[]           #表示里面的内容是可选项，也就是说，一条命令，选项和参数是可以都没有的
...          #表示可以有多个值，也就是说，一条命令，可以有多个选项，或多个参数
```

选项有多种风格：

- 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，使用相对较少

范例:

```
root@ubuntu2204:~# id -u mage
1000
root@ubuntu2204:~# ls -a
root@ubuntu2204:~# ls --all
root@ubuntu2204:~# free -h
root@ubuntu2204:~# free --human
root@ubuntu2204:~# ps a
```

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 查看硬件信息

1.7.1.1 查看 cpu

lscpu 命令可以查看cpu信息

cat /proc/cpuinfo也可看查看到

范例：

```
[root@ubuntu2204 ~]# lscpu
Architecture:            x86_64
```

```
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         45 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  2
  On-line CPU(s) list:   0,1
Vendor ID:               GenuineIntel
  Model name:            11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz
    CPU family:          6
    Model:               140
    Thread(s) per core:  1                    #每个core 有几个线程
    Core(s) per socket:  1                    #每个槽位有1个core
    Socket(s):           2                    #服务器面板上有2个cpu 槽位
    Stepping:            1
    BogoMIPS:            5990.42
    Flags:               ...

Virtualization features:
  Hypervisor vendor:     VMware
  Virtualization type:   full
Caches (sum of all):
  L1d:                   96 KiB (2 instances)
  L1i:                   64 KiB (2 instances)
  L2:                    2.5 MiB (2 instances)
  L3:                    24 MiB (2 instances)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0,1


[root@ubuntu2204 ~]# cat /proc/cpuinfo
```

1.7.1.2 查看内存大小

```
[root@ubuntu2204 ~]# free
               total        used        free      shared  buff/cache   available
Mem:         1989528      353580      844332        1328      791616     1479600
Swap:        2097148           0     2097148


[root@ubuntu2204 ~]# cat /proc/meminfo
MemTotal:        1833232 kB
MemFree:          296844 kB
MemAvailable:     910248 kB
.....
.....
```

free 命令补充

```
#1秒刷新一次数据
[root@ubuntu2204 ~]# free -hs 1


#刷新2次数据后退出
[root@ubuntu2204 ~]# free -hc 2

total        #系统总的可用物理内存大小
used         #已被使用的物理内存大小
free         #还有多少物理内存可用
shared       #被共享使用的物理内存大小
buff/cache   #被 buffer 和 cache 使用的物理内存大小
available    #还可以被 应用程序 使用的物理内存大小


#free 是真正尚未被使用的物理内存数量。
#available 是应用程序认为可用内存数量，available = free + buffer + cache (大概的计算方法)
```

1.7.1.3 查看硬盘和分区情况

```
[root@centos8 ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda           8:0    0  200G  0 disk
├─sda1        8:1    0    1G  0 part /boot
└─sda2        8:2    0  199G  0 part
  ├─rl-root 253:0    0   70G  0 lvm  /
  ├─rl-swap 253:1    0    2G  0 lvm  [SWAP]
  └─rl-home 253:2    0  127G  0 lvm  /home
sr0          11:0    1 1024M  0 rom

[root@centos8 ~]# cat /proc/partitions
major minor  #blocks  name

   8        0  209715200 sda
   8        1    1048576 sda1
   8        2  208665600 sda2
  11        0    1048575 sr0
 253        0   73400320 dm-0
 253        1    2129920 dm-1
 253        2  133132288 dm-2


 #dm 是 lvm 设备
```

```
[root@ubuntu2204 ~]# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop1                       7:1    0  79.9M  1 loop /snap/lxd/22923
loop2                       7:2    0  61.9M  1 loop /snap/core20/1405
loop3                       7:3    0  53.2M  1 loop /snap/snapd/19122
loop4                       7:4    0  63.3M  1 loop /snap/core20/1879
loop5                       7:5    0 111.9M  1 loop /snap/lxd/24322
sda                         8:0    0   200G  0 disk
├─sda1                      8:1    0     1M  0 part
├─sda2                      8:2    0     2G  0 part /boot
└─sda3                      8:3    0   198G  0 part
  └─ubuntu--vg-ubuntu--lv 253:0    0    99G  0 lvm  /
sr0                        11:0    1   1.4G  0 rom



[root@ubuntu2204 ~]# cat /proc/partitions
major minor  #blocks  name

   7        1      81868 loop1
   7        2      63380 loop2
   7        3      54516 loop3
   7        4      64856 loop4
   7        5     114636 loop5
  11        0    1432338 sr0
   8        0  209715200 sda
   8        1       1024 sda1
   8        2    2097152 sda2
   8        3  207614976 sda3
 253        0  103804928 dm-0
```

1.7.2 查看系统版本信息

1.7.2.1 查看系统架构

```
[root@ubuntu2204 ~]# arch
x86_64
```

1.7.2.2 查看内核版本

```
[root@rocky8 ~]# uname -r
4.18.0-372.9.1.el8.x86_64

[root@ubuntu2204 ~]# uname -r
5.15.0-25-generic
```

1.7.2.3 查看操作系统发行版本

```
#CentOS8 查看发行版本
[root@centos8 ~]#cat /etc/redhat-release
CentOS Linux release 8.1.1911 (Core)

[root@centos8 ~]#cat /etc/os-release
NAME="CentOS Linux"
VERSION="8 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="8"
PLATFORM_ID="platform:el8"
PRETTY_NAME="CentOS Linux 8 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:8"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"
CENTOS_MANTISBT_PROJECT="CentOS-8"
CENTOS_MANTISBT_PROJECT_VERSION="8"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="8"

[root@centos8 ~]#lsb_release -a
LSB Version: :core-4.1-amd64:core-4.1-noarch
Distributor ID: CentOS
Description: CentOS Linux release 8.1.1911 (Core)
Release: 8.1.1911
Codename: Core



#ubuntu查看发行版本
[root@ubuntu2204 ~]# cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
```

```
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-
policy"
UBUNTU_CODENAME=jammy


[root@ubuntu2204 ~]# cat /etc/issue
Ubuntu 22.04 LTS \n \l


[root@ubuntu2204 ~]# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04 LTS
Release:    22.04
Codename:   jammy
```

范例: 查看 OS 版本

```
[root@centos8 ~]#lsb_release -is
CentOS
[root@centos8 ~]#lsb_release -cs
Core
[root@centos8 ~]#lsb_release -rs
8.2.2004
[root@centos7 ~]#lsb_release -is
CentOS
[root@centos7 ~]#lsb_release -cs
Core
[root@centos7 ~]#lsb_release -rs
7.9.2009
[root@centos6 ~]#lsb_release -is
CentOS
[root@centos6 ~]#lsb_release -cs
Final
[root@centos6 ~]#lsb_release -rs
6.10
root@ubuntu2004:~# lsb_release -is
Ubuntu
root@ubuntu2004:~# lsb_release -cs
focal
root@ubuntu2004:~# lsb_release -rs
20.04
[root@ubuntu1804 ~]#lsb_release -is
Ubuntu
[root@ubuntu1804 ~]#lsb_release -cs
bionic
[root@ubuntu1804 ~]#lsb_release -rs
18.04
```

1.7.3 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

date 显示和设置系统时间

```
Usage: date [OPTION]... [+FORMAT]
  or:  date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

范例：

```
[root@ubuntu2204 ~]# date
Mon May  8 02:21:33 AM UTC 2023

#显示时区信息
[root@ubuntu2204 ~]# date -R
Mon, 08 May 2023 02:21:40 +0000

#时间戳
[root@ubuntu2204 ~]# date +%s
1683512505


[root@ubuntu2204 ~]# date -d @`date +%s`
Mon May  8 02:21:51 AM UTC 2023

[root@ubuntu2204 ~]# date -d @1683512505
Mon May  8 02:21:45 AM UTC 2023

[root@ubuntu2204 ~]# date -d @1683512505 +%F_%T
2023-05-08_02:21:45

[root@ubuntu2204 ~]# date -d "2023-05-08" +%s
1683504000
```

clock，hwclock 显示硬件时钟

```
clock [function] [option...]

hwclock [function] [option...]


#常用选项
-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
```

```
[root@centos8 ~]# ll /usr/sbin/clock
lrwxrwxrwx. 1 root root 7 Oct 14  2021 /usr/sbin/clock -> hwclock

#对钟
[root@rocky86 ~]# clock -s


[root@ubuntu2204 ~]# ll /usr/sbin/clock
ls: cannot access '/usr/sbin/clock': No such file or directory


[root@ubuntu2204 ~]# ll /usr/sbin/hwclock
-rwxr-xr-x 1 root root 51704 Feb 21  2022 /usr/sbin/hwclock*
```

时区

```
/etc/localtime
```

范例: 设置时区

```
[root@ubuntu2204 ~]# timedatectl list-timezones

[root@ubuntu2204 ~]# timedatectl set-timezone Asia/Shanghai

[root@ubuntu2204 ~]# timedatectl status
               Local time: Mon 2023-05-08 10:33:48 CST
           Universal time: Mon 2023-05-08 02:33:48 UTC
                 RTC time: Mon 2023-05-08 02:33:48
                Time zone: Asia/Shanghai (CST, +0800)
```

```
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no


[root@ubuntu2204 ~]# ll /etc/localtime
lrwxrwxrwx 1 root root 33 May  8 10:33 /etc/localtime ->
/usr/share/zoneinfo/Asia/Shanghai

[root@ubuntu2204 ~]# cat /etc/timezone
Asia/Shanghai
```

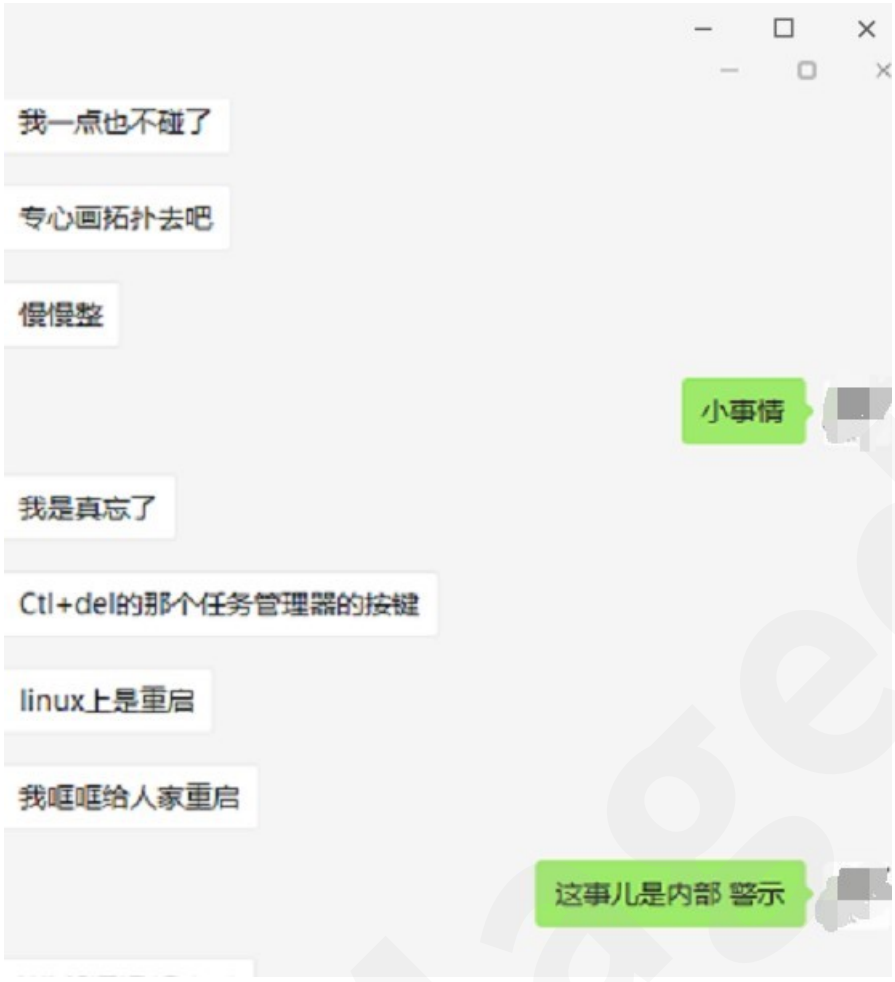
日历

```
Usage:
 cal [options] [[[day] month] year]
 cal [options] <timestamp|monthname>
```

范例:

```
[root@ubuntu2204 ~]# 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
```

1.7.4 关机和重启



关机：

```
halt
poweroff
init 0
shutdown -h now
```

重启：

```
reboot
init 6
shutdown -r now

#ctrl+alt+delete  三键
```

关机或重启：shutdown

shutdown 程序会广播已登录的用户，被看作是安全的关机命令

是一个计划关机任务，可撤销

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


shutdown                #一分钟后关机
shutdown +10            #10分钟后关机
shutdown 01:02          #1点过两分关机
shutdown -r|--reboot    #一分钟后重启
shutdown -r now         #现在重启
shutdown -H|--halt      #一分钟后调用halt 关机
shutdown -P|--poweroff  #一分钟后调用poweroff 关机
shutdown -c             #取消关机计划
```

```
#-r 表示一分钟后重启，如果想立即执行 shutdown -r now
[root@ubuntu2204 ~]# shutdown -r
Shutdown scheduled for Sun 2022-06-12 20:11:05 CST, use 'shutdown -c' to cancel.

#取消重启
[root@ubuntu2204 ~]# shutdown -c
```

1.7.5 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

whoami

```
#whoami
#显示当前用户的用户名


#who am i
#显示当前用户的用户名 终端 登录时间 来源IP
```

who

```
#who [选项]... [ 文件 | 参数1 参数2 ]
#显示当前已登录的用户信息。

who                 #列出在当前主机上所有登录用户
who -u | --users    #列出当前主机上所有用户的空闲时间 . 表示最近一分钟还是活跃状态  old 表示用户已经空闲超过24小时
who -s | --short    #列出在当前主机上所有登录用户，等同于who
who -q | --count    #登录用户统计
who -b | --boot     #上次系统启动时间
who -a | --all      #多选项组合
who -m              #who am i
```

w

```
#w [options]
#显示当前所有登录用户的具体信息

w
 19:19:03 up 28 min,  2 users,  load average: 0.04, 0.05, 0.11
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    192.168.31.236   18:40    2.00s  0.09s  0.00s w
jose     tty2     tty2             18:41   38:45  24.98s  0.04s
/usr/libexec/gsd-disk-utility-notify

#登录名 终端 来源IP 登录时间  空闲时间 当前终端中所有进程使用cpu的时间,不包括后台作业占用的时间  当前进程使用的cpu的时间
#load average 表示平均负载，最近一分钟，最近五分钟，最近15分钟


#查看特定用户
w root
```

1.7.6 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
- gedit 工具是图形工具

范例: 创建登录提示文件 /etc/motd

参考网站: https://www.bootschool.net/ascii-art


```
[root@ubuntu1804 ~]#cat /etc/motd
                       _oo0oo_
                      088888880
                      88"  . "88
                      (| -_- |)
                       0\  =  /0
                     ___/'---'\___
                   .' \\\\|     |//// '.
                  / \\\\|||:|||//// \
                 /_ |||||  -:-  ||||| _\
                |   |  \\\\\-/////  |   |
                | \_|    ''\---/''    |_/ |
                \  .-\__     '-'     __/-.  /
              ___'. .'   /--.--\   '. .'___
           ."" '<  '.___\_<|>_/___.' >'  "".
          | | : '-   \'.;'\  _  /';.'/ -  ' : | |
          \  \ '_.    \_ __\ /__ _/    .-' /  /
      ====='-.____'.___ \_____/___.-'____.-'=====

      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

              佛祖保佑     iii     永不死机
[root@ubuntu1804 ~]#exit
```

1.7.7 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，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行程序很有用。
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

1.7.7.1 screen

利用screen 可以实现会话管理,如：新建会话,共享会话等

注意：CentOS7 来自于base源，CentOS8 来自于epel源

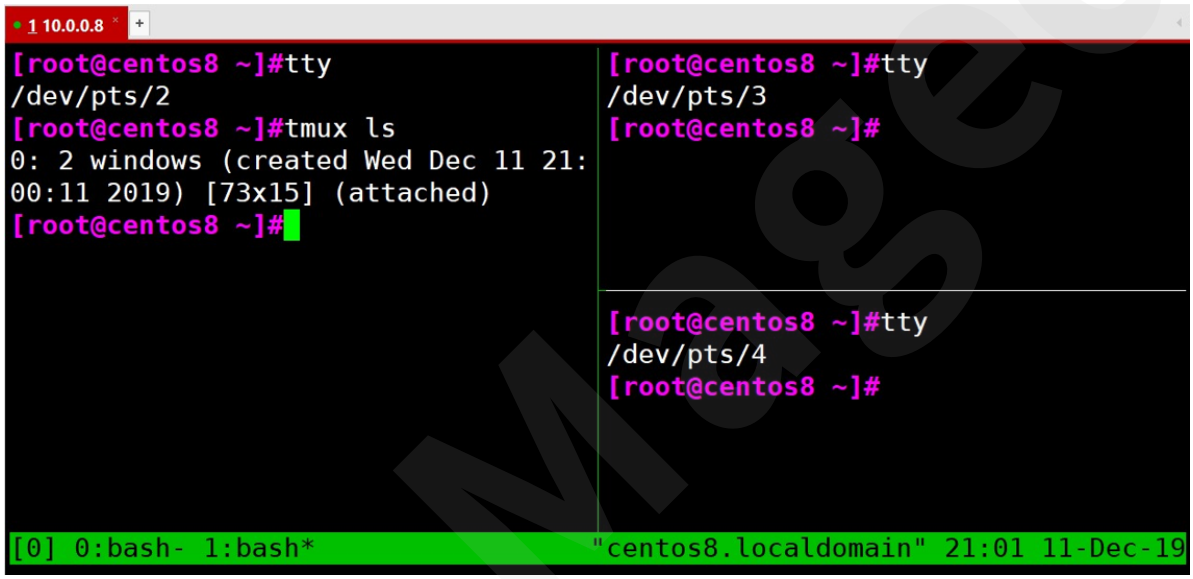
范例：安装 screen

```
#CentOS7 安装screen
[root@centos7 ~]# yum -y install screen

#CentOS8 安装screen
[root@centos8 ~]# dnf -y install epel-release
[root@centos8 ~]# dnf -y install screen


#ubuntu
[root@ubuntu ~]# apt install screen
```

screen命令常见用法：

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

1.7.7.2 tmux



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

Tmux 就是会话与窗口的"解绑"工具，将它们彻底分离，功能如下

- 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行程序很有用。
- 它可以让新窗口"接入"已经存在的会话。
- 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。
- 它还支持窗口任意的垂直和水平拆分

安装

```
[root@rocky8 ~]# yum install tmux


[root@ubuntu2204 ~]# apt update
root@ubuntu2204 ~]# apt install tmux
```

启动与退出

```
[root@ubuntu2204 ~]# tmux

[root@ubuntu2204 ~]# exit
[exited]
```

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 <arrow key>      #光标切换到其他窗格。<arrow key>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+<arrow key> #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 <number>     #切换到指定编号的窗口，其中的<number>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
```

1.7.8 输出信息 echo

1.7.8.1 echo 基本用法

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: echo [-neE] [字符串]


-n  #输出完成后不换行
-e  #转义特定字符串
-E  #不转义，原样输出，默认选项



#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）
```

显示变量

```
echo "$VAR_NAME”   #用变量值替换，弱引用
echo '$VAR_NAME’   #变量不会替换，强引用
```

范例:

```
#默认换行
[root@ubuntu2204 ~]#  echo hello
hello

#不换行
[root@ubuntu2204 ~]# echo -n hello
hello[root@ubuntu2204 ~]#

#-e 转义
[root@ubuntu2204 ~]# echo -e 'a\x0Ab'
a
b


[root@ubuntu2204 ~]# echo -e '\033[43;31;1;5mmagedu\e[0m'
magedu

[root@ubuntu2204 ~]# echo -e '\x57\x41\x4E\x47'
WANG

#原样输出
[root@ubuntu2204 ~]# echo \$PATH
$PATH

[root@ubuntu2204 ~]# echo \
>
[root@ubuntu2204 ~]# echo \\
\
[root@ubuntu2204 ~]# echo \\\
>

[root@ubuntu2204 ~]# echo \\\\
\\

#输出变量
[root@ubuntu2204 ~]# echo "$PATH"
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/snap/bin

#原样输出
[root@ubuntu2204 ~]# echo '$PATH'
$PATH
```

1.7.8.2 echo 高级用法

在终端中，ANSI定义了用于屏幕显示的Escape屏幕控制码

可以显示具有颜色的字符，其格式如下:

```
"\033[字符背景颜色;字体颜色;特效m字符串\033[0m"

"\033[41;32mtest\033[0m"


\033[30m -- \033[37m          #设置前景色
\033[40m -- \033[47m          #设置背景色
```

```
#字符背景颜色范围: 40--47
40:黑
41:红
42:绿
43:黄
44:蓝
45:紫
46:深绿
47:白色

#字体颜色: 30--37
30: 黑
31: 红
32: 绿
33: 黄
34: 蓝
35: 紫
36: 深绿
37: 白色
```

加颜色只是以下控制码中的一种，下面是常见的一些ANSI控制码:

```
\033[0m               #关闭所有属性
\033[1m               #设置高亮度
\033[4m               #下划线
\033[5m               #闪烁
\033[7m               #反显
\033[8m               #消隐
\033[nA               #光标上移n行
\033[nB               #光标下移n行
\033[nC               #光标右移n列
\033[nD               #光标左移n列
\033[x;yH             #设置光标位置x行y列
\033[2J               #清屏
\033[K                #清除从光标到行尾的内容
```

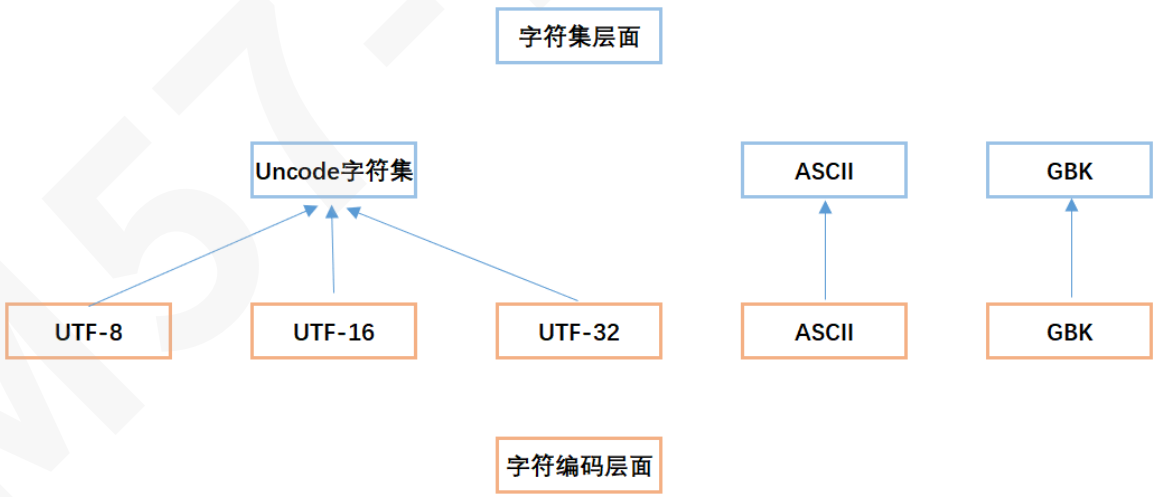
```
\033[s                  #保存光标位置
\033[u                  #恢复光标位置
\033[?25l               #隐藏光标
\033[?25h               #显示光标
\033[2J\033[0;0H        #清屏且将光标置顶
```

1.8 字符集和编码及语言环境

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，即二进制和字符的对应关系，不关注最终的存储形式

encoding是charset encoding的简写，即字符集编码，简称编码，实现如何将字符转化为实际的二进制进行存储，编码决定了空间的使用的大小

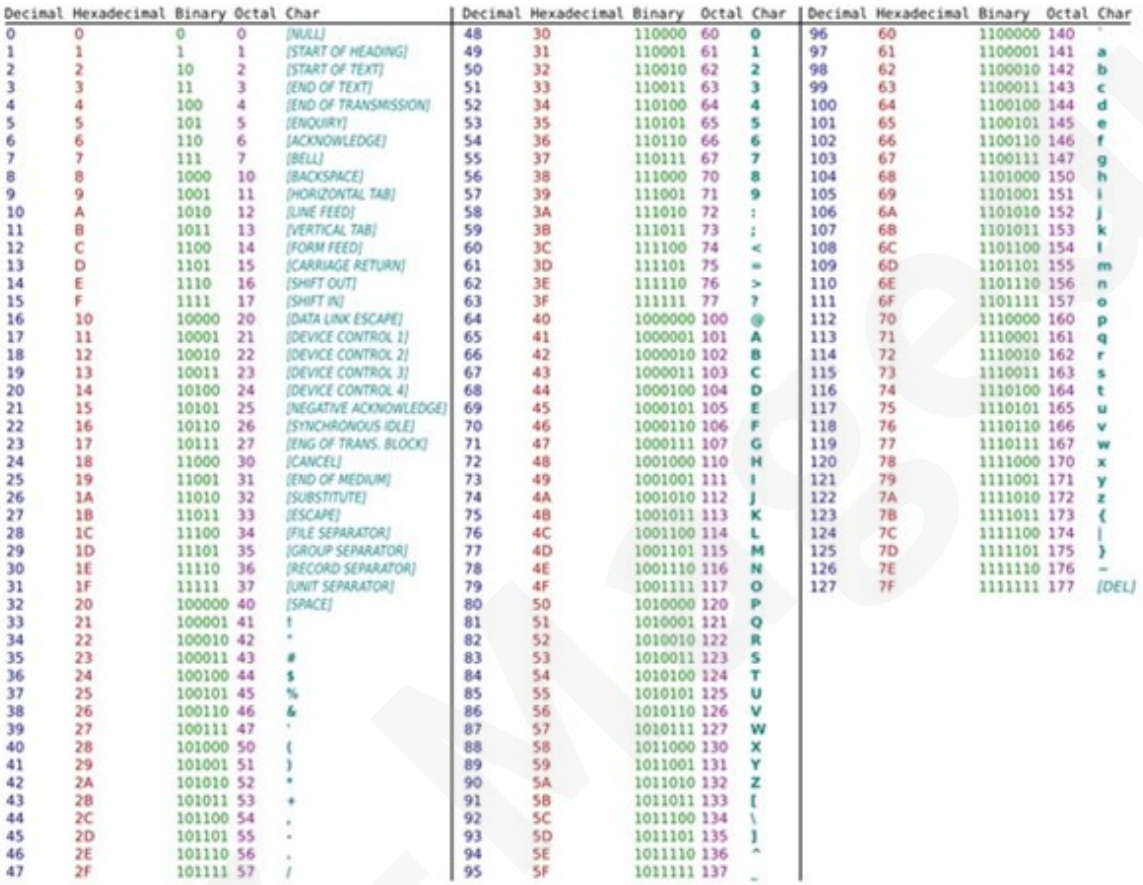


计算机 二进制，起源于美国

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



Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Oct 8进制

Dec 10进制 ascii 编码

Hex 16进制

Bin 2进制

Char 字符

范例：查看 ascii 表

```
[root@centos8 ~]#dnf -y install man-pages
[root@centos8 ~]#man ascii

[root@ubuntu2204 ~]# man ascii
```

Oct	Dec	Hex	Char	Oct	Dec	Hex	Char
000	0	00	NUL '\0' (null character)	100	64	40	@
001	1	01	SOH (start of heading)	101	65	41	A
002	2	02	STX (start of text)	102	66	42	B
003	3	03	ETX (end of text)	103	67	43	C
004	4	04	EOT (end of transmission)	104	68	44	D
005	5	05	ENQ (enquiry)	105	69	45	E
006	6	06	ACK (acknowledge)	106	70	46	F
007	7	07	BEL '\a' (bell)	107	71	47	G
010	8	08	BS '\b' (backspace)	110	72	48	H
011	9	09	HT '\t' (horizontal tab)	111	73	49	I
012	10	0A	LF '\n' (new line)	112	74	4A	J
013	11	0B	VT '\v' (vertical tab)	113	75	4B	K
014	12	0C	FF '\f' (form feed)	114	76	4C	L
015	13	0D	CR '\r' (carriage ret)	115	77	4D	M
016	14	0E	SO (shift out)	116	78	4E	N
017	15	0F	SI (shift in)	117	79	4F	O
020	16	10	DLE (data link escape)	120	80	50	P
021	17	11	DC1 (device control 1)	121	81	51	Q
022	18	12	DC2 (device control 2)	122	82	52	R
023	19	13	DC3 (device control 3)	123	83	53	S
024	20	14	DC4 (device control 4)	124	84	54	T
025	21	15	NAK (negative ack.)	125	85	55	U
026	22	16	SYN (synchronous idle)	126	86	56	V
027	23	17	ETB (end of trans. blk)	127	87	57	W
030	24	18	CAN (cancel)	130	88	58	X
031	25	19	EM (end of medium)	131	89	59	Y
032	26	1A	SUB (substitute)	132	90	5A	Z
033	27	1B	ESC (escape)	133	91	5B	[
034	28	1C	FS (file separator)	134	92	5C	\ '\\'
035	29	1D	GS (group separator)	135	93	5D	]
036	30	1E	RS (record separator)	136	94	5E	^
037	31	1F	US (unit separator)	137	95	5F	_
040	32	20	SPACE	140	96	60	`
041	33	21	!	141	97	61	a
042	34	22	"	142	98	62	b
043	35	23	#	143	99	63	c
044	36	24	$	144	100	64	d
045	37	25	%	145	101	65	e
046	38	26	&	146	102	66	f
047	39	27	'	147	103	67	g
050	40	28	(	150	104	68	h
051	41	29	)	151	105	69	i
052	42	2A	*	152	106	6A	j
053	43	2B	+	153	107	6B	k
054	44	2C	,	154	108	6C	l
055	45	2D	-	155	109	6D	m
056	46	2E	.	156	110	6E	n
057	47	2F	/	157	111	6F	o
060	48	30	0	160	112	70	p
061	49	31	1	161	113	71	q
062	50	32	2	162	114	72	r
063	51	33	3	163	115	73	s

064	52	34	4	164	116	74	t
065	53	35	5	165	117	75	u
066	54	36	6	166	118	76	v
067	55	37	7	167	119	77	w
070	56	38	8	170	120	78	x
071	57	39	9	171	121	79	y
072	58	3A	:	172	122	7A	z
073	59	3B	;	173	123	7B	{
074	60	3C	<	174	124	7C	|
075	61	3D	=	175	125	7D	}
076	62	3E	>	176	126	7E	~
077	63	3F	?	177	127	7F	DEL

1.8.2 Unicode

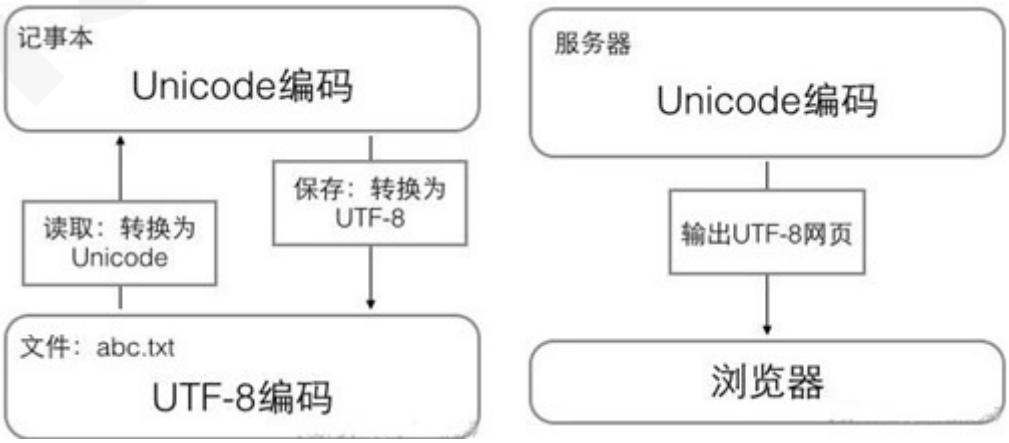
由于计算机是美国人发明的，因此，最早只有128个字符被编码到计算机里，即ASCII编码，但是要处理中文显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。

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为 10，剩下的二进制位则使用这个字符的 Unicode 码来填充

编码转换和查询参考链接：

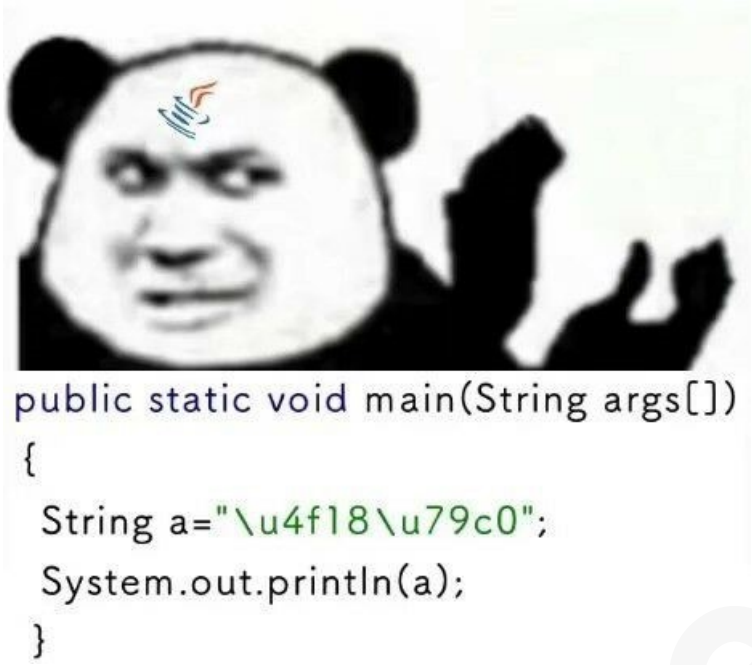
```
https://home.unicode.org/
https://unicode.yunser.com/unicode
http://www.chi2ko.com/tool/CJK.htm
https://www.bejson.com/convert/unicode_chinese/
https://javawind.net/tools/native2ascii.jsp?action=transform
http://tool.oschina.net/encode
http://web.chacuo.net/charsetescape
```

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：Unicode 转换为 UTF-8

```
“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
```



1.8.3 语言环境

默认系统为英文环境，可以修改为中文环境，从而查看帮助或提示可以变为中文

范例：临时修改LANG变量实现中文语言提示

```
[root@centos7 ~]#echo $LANG
en_US.UTF-8

[root@centos7 ~]#magedu
-bash: magedu: command not found

[root@centos7 ~]#LANG=zh_CN.UTF-8

[root@centos7 ~]#echo $LANG
zh_CN.UTF-8

[root@centos7 ~]#magedu
-bash: magedu: 未找到命令
```

```
[root@rocky8 ~]# localectl status
   System Locale: LANG=en_US.UTF-8
       VC Keymap: us
      X11 Layout: us

[root@rocky8 ~]# echo $LANG
```

```
en_US.UTF-8

[root@rocky8 ~]# localectl list-locales
C.utf8
en_AG
en_AU
en_AU.utf8
en_BW
en_BW.utf8
en_CA
en_CA.utf8
en_DK
en_DK.utf8
en_GB
en_GB.iso885915
en_GB.utf8
en_HK
en_HK.utf8
en_IE
en_IE.utf8
en_IE@euro
en_IL
en_IN
en_NG
en_NZ
en_NZ.utf8
en_PH
en_PH.utf8
en_SC.utf8
en_SG
en_SG.utf8
en_US
en_US.iso885915
en_US.utf8
en_ZA
en_ZA.utf8
en_ZM
en_ZW
en_ZW.utf8



[root@rocky8 ~]#yum list lang*
Last metadata expiration check: 1:00:42 ago on Sun 19 Sep 2021 10:34:15 AM CST.
Installed Packages
langpacks-en.noarch          1.0-12.el8
      @AppStream
Available Packages
......
langpacks-vi.noarch          1.0-12.el8       AppStream
langpacks-xh.noarch          1.0-12.el8       AppStream
langpacks-zh_CN.noarch       1.0-12.el8       AppStream
langpacks-zh_TW.noarch       1.0-12.el8       AppStream
langpacks-zu.noarch          1.0-12.el8       AppStream
langtable.noarch             0.0.51-4.el8     AppStream
```

```
[root@rocky8 ~]#yum -y install langpacks-zh_CN.noarch



[root@rocky8 ~]#localectl list-locales
C.utf8
en_AG
en_AU
.......
zh_CN
zh_CN.gb18030
zh_CN.gbk
zh_CN.utf8
zh_HK
zh_HK.utf8
zh_SG
......


#通用方法
[root@rocky8 ~]#localectl set-locale LANG=zh_CN.utf8


#或者下面方式,CentOS8支持,但ubuntu和Centos7不支持,不建议使用
[root@rocky8 ~]#localectl set-locale zh_CN.utf8

[root@rocky8 ~]#localectl status
   System Locale: LANG=zh_CN.utf8
       VC Keymap: us
      X11 Layout: us
[root@rocky8 ~]#echo $LANG
zh_CN.utf8

#重新登录后可以看到中文环境
[root@rocky8 ~]#exit
```

范例: Ubuntu 修改语言环境为中文

```
[root@ubuntu2204 ~]# localectl status
   System Locale: LANG=en_US.UTF-8
       VC Keymap: n/a
      X11 Layout: us
       X11 Model: pc105

[root@ubuntu2204 ~]# apt install language-pack-zh-hans -y

[root@ubuntu2204 ~]# localectl list-locales
C.UTF-8
en_US.UTF-8
zh_CN.UTF-8
```

```
zh_SG.UTF-8

[root@ubuntu2204 ~]# localectl set-locale LANG=zh_CN.utf8

[root@ubuntu2204 ~]# mage
找不到命令 “mage”，您的意思是：
  “marge” 命令来自 Snap 软件包 marge (0.4.1)
  “mace” 命令来自 Snap 软件包 mace (0.2.0)
  “sage” 命令来自 Debian 软件包 sagemath (9.5-4)
  “age” 命令来自 Debian 软件包 age (1.0.0-1)
  “mame” 命令来自 Debian 软件包 mame (0.242+dfsg.1-1)
  “mag” 命令来自 Debian 软件包 texlive-binaries (2021.20210626.59705-1build1)
  “make” 命令来自 Debian 软件包 make (4.3-4.1build1)
  “make” 命令来自 Debian 软件包 make-guile (4.3-4.1build1)
  “cage” 命令来自 Debian 软件包 cage (0.1.4-3)
  “page” 命令来自 Debian 软件包 tcllib (1.20+dfsg-1)
输入 “snap info <snapname>” 以查看更多版本。

[root@ubuntu2204 ~]# echo $LANG
zh_CN.UTF-8
```

1.9 命令行扩展和被括起来的集合

1.9.1 命令行扩展：`` 和 $()

把一个命令的输出打印给另一个命令的参数，放在``中的一定是有输出信息的命令

```
$(COMMAND)
`COMMAND`
```

变量

双引号，弱引用，可以解析内容

单引号，强引用，原样输出

命令

`cmd` 展开

$(cmd) 展开

范例：比较“”，‘’， ``三者区别

```
[root@ubuntu2204 ~]# echo "echo $HOSTNAME"
echo ubuntu2204

[root@ubuntu2204 ~]# echo 'echo $HOSTNAME'
echo $HOSTNAME

[root@ubuntu2204 ~]# echo `echo $HOSTNAME`
ubuntu2204


#结论：
单引号：强引用,六亲不认，变量和命令都不识别，都当成了普通的字符串,"最傻"
双引号：弱引用,不能识别命令，可以识别变量,"半傻不精"
反向单引号：里面的内容必须是能执行的命令并且有输出信息,变量和命令都识别，并且会将反向单引号的内容当成命令进行执行后，再交给调用反向单引号的命令继续,"最聪明"
```

范例：

```
[root@ubuntu2204 ~]# echo "This system's name is $(hostname)"
This system's name is ubuntu2204

[root@ubuntu2204 ~]# echo "I am `whoami`"
I am root

[root@ubuntu2204 ~]# touch $(date +%F).log

[root@ubuntu2204 ~]# ls
2023-05-08.log  snap


[root@ubuntu2204 ~]# touch `date +%F`.txt
[root@ubuntu2204 ~]# touch `date +%F_%H-%M-%S`.log

[root@ubuntu2204 ~]# ls -l
总计 4
-rw-r--r-- 1 root root    0  5月  8 11:12 2023-05-08_11-12-24.log
-rw-r--r-- 1 root root    0  5月  8 11:07 2023-05-08.log
-rw-r--r-- 1 root root    0  5月  8 11:08 2023-05-08.txt
drwx------ 3 root root 4096  5月  4 17:02 snap
```

范例：$()和 ``

```
[root@ubuntu2204 ~]# ll `echo `date +%F`.txt`
.txt: command not found
ls: cannot access 'date': No such file or directory
ls: cannot access '+%F': No such file or directory

[root@ubuntu2204 ~]# ll $(echo $(date +%F).txt)
-rw-r--r-- 1 root root 0 May  8 11:08 2023-05-08.txt

[root@ubuntu2204 ~]# ll `echo $(date +%F).txt`
-rw-r--r-- 1 root root 0 May  8 11:08 2023-05-08.txt

[root@ubuntu2204 ~]# ll $(echo `date +%F`.txt)
-rw-r--r-- 1 root root 0 May  8 11:08 2023-05-08.txt
```

1.9.2 括号扩展：{}

{} 可以实现打印重复字符串的简化形式

```
{元素1,元素2,元素3}
{元素1..元素2}
```

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 {1..10..2}
echo {000..20..2}
```

范例：

```
[root@ubuntu2204 ~]# echo {000..20..2}
000 002 004 006 008 010 012 014 016 018 020

[root@ubuntu2204 ~]# echo {a..z..2}
a c e g i k m o q s u w y

[root@ubuntu2204 ~]# echo {A..z}
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ ] ^ _ ` a b c d e f g h i
j k l m n o p q r s t u v w x y z

[root@ubuntu2204 ~]# echo {a..z} {A..Z}
a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
```

```
[root@ubuntu2204 ~]# echo {a..z}{1..3}
a1 a2 a3 b1 b2 b3 c1 c2 c3 d1 d2 d3 e1 e2 e3 f1 f2 f3 g1 g2 g3 h1 h2 h3 i1 i2 i3
j1 j2 j3 k1 k2 k3 l1 l2 l3 m1 m2 m3 n1 n2 n3 o1 o2 o3 p1 p2 p3 q1 q2 q3 r1 r2 r3
s1 s2 s3 t1 t2 t3 u1 u2 u3 v1 v2 v3 w1 w2 w3 x1 x2 x3 y1 y2 y3 z1 z2 z3
```

范例: 关闭和启用{}的扩展功能

```
[root@ubuntu2204 ~]# echo $-
himBHs

[root@ubuntu2204 ~]# echo {1..10}
1 2 3 4 5 6 7 8 9 10

[root@ubuntu2204 ~]# set +B
[root@ubuntu2204 ~]# echo $-
himHs

[root@ubuntu2204 ~]# echo {1..10}
{1..10}

[root@ubuntu2204 ~]# set -B
[root@ubuntu2204 ~]# echo $-
himBHs

[root@ubuntu2204 ~]# echo {1..10}
1 2 3 4 5 6 7 8 9 10
```

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名称的文件，第一次找到的命令即为要执行的命令
- 命令的子命令补全，需要安装 bash-completion

注意：用户给定的字符串只有一条惟一对应的命令，直接补全，否则，再次Tab会给出列表

范例：

```
[root@centos8 ~]#nmcli connection 2TAB
add     delete   edit     help     load     monitor show
clone   down     export   import   modify   reload   up
```

```
[root@ubuntu2204 ~]# apt 2TAB
autoclean     clean          full-upgrade  policy        search        upgrade
autopurge     depends        help          purge         show
autoremove    dist-upgrade   install       rdepends      showsrc
build-dep     download       list          reinstall     source
changelog     edit-sources   moo           remove        update
```

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
- string2Tab 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

当执行某个命令后，系统会在内存中记录下此命令，此时只记录在内存中，

当正常退出终端后，内存中记录下来的执行命令的历史，会保存到用户家目录下的 .bash_history 文件中

当用户下次登录，系统会自动将该文件中的内容加载到内存

被写入命令历史的，包括错误的命令

默认记录最近1000条，其配置在 /etc/profile 文件中的 HISTSIZE字段中，可以根据需要修改

利用命令历史，可以用它来重复执行命令，提高输入效率

命令：history

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



#常用选项
-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 `whoami` "                            #显示时间和用户
HISTIGNORE="str1:str2*:…"                                   #忽略str1命令，str2开头
的历史
HISTCONTROL=ignoredups|ignorespace|ignoreboth|erasedups     #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@ubuntu2204 ~]# cat .profile
# ~/.profile: executed by Bourne-compatible login shells.

if [ "$BASH" ]; then
  if [ -f ~/.bashrc ]; then
    . ~/.bashrc
  fi
fi

mesg n 2> /dev/null || true


PATH=$PATH:$HOME/bin
export PATH
export HISTCONTROL=ignoreboth
export HISTTIMEFORMAT="%F %T "


[root@ubuntu2204 ~]# history
    1  2023-05-08 11:14:58 shutdown -h now
    2  2023-05-08 11:14:58 passwd
    3  2023-05-08 11:14:58 vim /etc/ssh/sshd_config
    4  2023-05-08 11:14:58 systemctl restart sshd
    ......
```

1.12 调用命令行历史

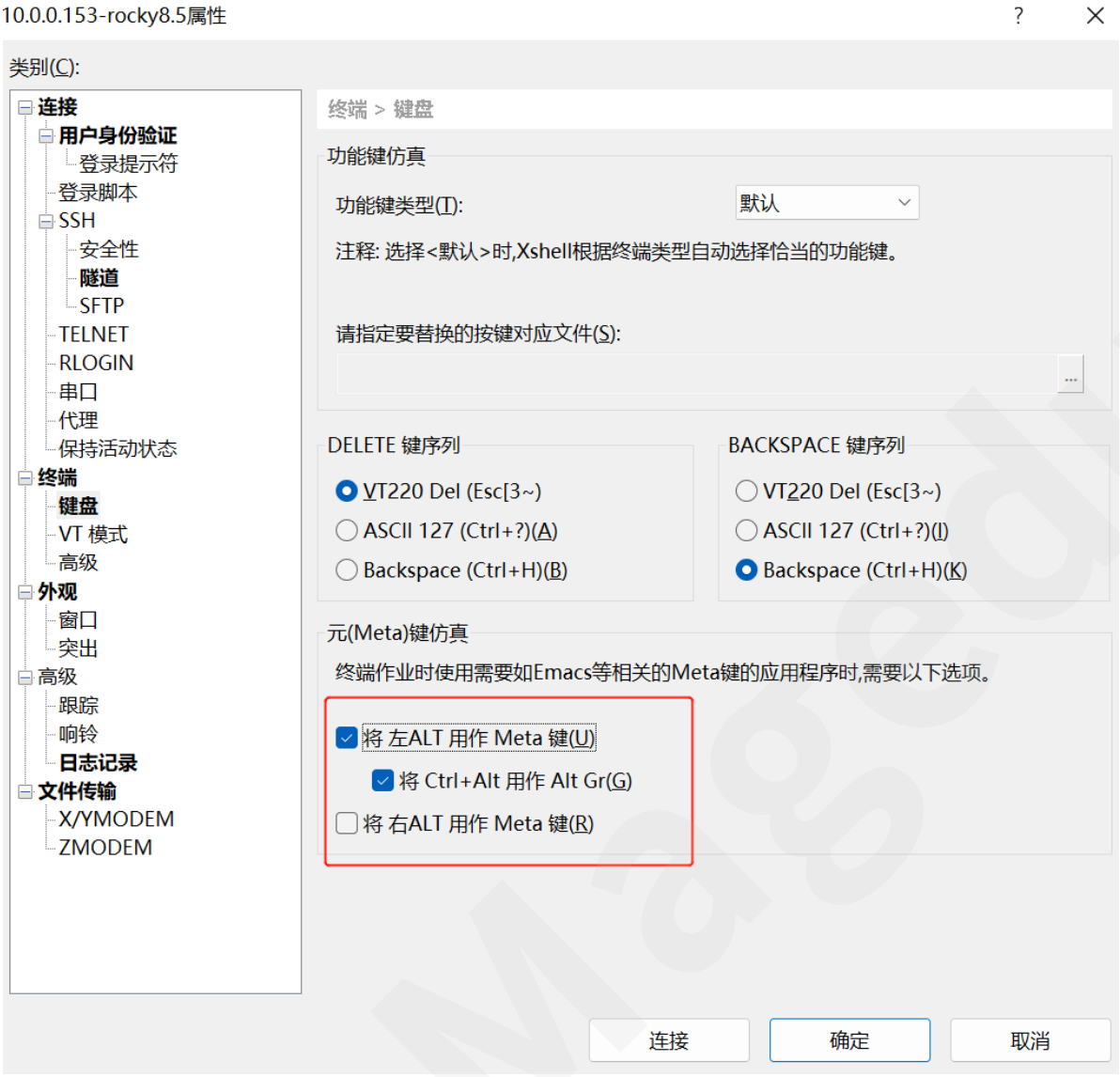
```
#重复前一个命令方法
重复前一个命令使用上方向键，并回车执行
按 !! 并回车执行
输入!-1 并回车执行
按 Ctrl+p 并回车执行


!: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
!$:p                        #打印输出 !$ （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
!$                          #表示前一个命令中最后一个参数
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command的参数
command !$                  #利用上一个命令的最后一个参数做command的参数
command !*                  #利用上一个命令的全部参数做command的参数
command !:n                 #利用上一个命令的第n个参数做command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，解锁
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
Ctrl + xx             #光标在命令行首和光标之间移动
ctrl+ >(方向键)       #光标向右移动一个单词尾，相当于 Alt + f
ctrl+ <(方向键)       #光标向左移动一个单词首，相当于 Alt + b
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
Alt + Backspace       #删除左边单词
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

范例：xshell中启动 alt 键



1.14 登录提示

1.14.1 登录前提示

在命令行模式下本地终端(tty1~tty6)登录界面，会有几行提示文字，

这些文字都保存在/etc/issue文件中，可以自行修改。

```
/etc/issue
/etc/issue.d/*.issue

#issue 支持转义字符，全部可用的转义字符可以通过 man agetty 查看，这里列出常用的
\d      #显示当前系统日期
\S      #显示操作系统名称
\m      #显示硬件体系结构，如i386、i686等
\n      #显示主机名
\o      #显示域名
\r      #显示内核版本
\t      #显示当前系统时间
\u      #显示当前登录用户的序列号
```

如果是远程终端ssh 登录，则其登录前信息，可以放在/etc/issue.net 中，但是该文件中的内容不支持转义

如果要使用远程终端ssh 登录前的提示信息，还需要修改sshd的相关配置文件

```
vim /etc/ssh/sshd_config

#Banner none  将此处的banner 指向对应的文件即可
Banner /etc/issue.net

#重启sshd 服务
service sshd restart
```

1.14.2 登录后提示

motd：message of the day

当用户从终端登录时，此文件的内容将会显示在终端上，如果shell工具支持中文，也可显示。

内容由使用者定制，经常用于通告信息，欢迎提示等。

但是，此文件只适用于命令行界面，如果是图形界面登录，将不显示此文件内容。

```
/etc/motd
/etc/motd.d/
```

```
#ubuntu2204中没有该文件，可自行创建
[root@ubuntu2204 ~]# ls /etc/motd
ls: cannot access '/etc/motd': No such file or directory


#登录后的提示来自于此目录下不同文件，如果不需要默认提示，可以将该目录清空
[root@ubuntu2204 ~]# ls /etc/update-motd.d/
00-header              85-fwupd                  91-release-upgrade      98-
fsck-at-reboot
10-help-text           88-esm-announce           92-unattended-upgrades  98-
reboot-required
50-landscape-sysinfo   90-updates-available      95-hwe-eol
50-motd-news           91-contract-ua-esm-status 97-overlayroot
```

2 获得帮助

获取帮助的能力决定了技术的能力!

- whatis & whereis 命令
- command --help 选项
- man 手册
- 程序自带文档 /usr/share/doc
- 项目官网文档
- 行业网站
- 搜索引擎

2.1 whatis & whereis

whatis 使用数据库来显示命令的简短描述，以及对应的man手册的章节

刚装完系统此命令不可用，其数据要进行初始化，

如果要立即使用，则可手动初始化

```
#CentOS 7 版本以后
mandb

#CentOS 6 版本之前
makewhatis
```

范例：

```
[root@rocky8 ~]# whatis ls
ls (1)               - list directory contents
ls (1p)              - list directory contents


[root@ubuntu2204 ~]# whatis ls
ls (1)               - list directory contents


[root@ubuntu2204 ~]# mandb
Purging old database entries in /usr/share/man...
Processing manual pages under /usr/share/man...
Purging old database entries in /usr/share/man/pt...
......
```

whereis 可以列出命令或系统文件路径，以及其对应的man 手册中的文档路径

```
[root@ubuntu2204 ~]# whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz

[root@ubuntu2204 ~]# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man5/passwd.5.gz
/usr/share/man/man1/passwd.1.gz /usr/share/man/man1/passwd.1ssl.gz
```

2.2 查看命令的帮助

2.2.1 内部命令帮助

```
help COMMAND

#内部命令都在bash中，包括一些全局变量，可以在man手册中查看全部bash信息
man bash
```

范例：

```
#直接help，查看所有内部命令帮助
[root@ubuntu2204 ~]# help

[root@ubuntu2204 ~]# type history
history is a shell builtin

[root@ubuntu2204 ~]# help history
history: history [-c] [-d offset] [n] or history -anrw [filename] or history -ps
arg [arg...]
    ..........
```

2.2.2 外部命令及软件帮助

```
COMMAND --help|-h

man COMMAND

info COMMAND

#程序自身的帮助文档：README、INSTALL、ChangeLog 等

#项目官网文档

#行业网站

#搜索引擎
```

2.3 外部命令的 --help 或 -h 选项

显示用法总结和参数列表，大多数命令使用，但并非所有的

范例：

```
[root@ubuntu2204 ~]# ls --help
Usage: ls [OPTION]... [FILE]...
......
......


[root@ubuntu2204 ~]# date --help
Usage: date [OPTION]... [+FORMAT]
......
......


[root@ubuntu2204 ~]# cal -h
Usage: cal [general options] [-jy] [[month] year]
......
......


[root@ubuntu2204 ~]# date -h
date: invalid option -- 'h'
Try 'date --help' for more information.


[root@ubuntu2204 ~]# ls -h
```

格式说明：

常见用法	含义
[]	表示是可选项
CAPS或 <>	表示变化的数据
...	表示是一个列表，即可以有多个值
x|y|z	x 或 y 或 z
-abc	-a -b -c，多选项写一起
{}	表示分组

2.4 man 命令

man 是单词 manual 的简写，是Linux系统中的帮助手册和文档

man 是一个外部命令，基本质就是读取特定文件，进行输出，其读取的文件一般位于/usr/share/man/目录下

新安装一个软件或程序后，对应的man手册也会放到/usr/share/man/目录下

几乎每个命令都有man的 “页面”

在线手册

```
https://man7.org/linux/man-pages/index.html
https://man7.org/linux/man-pages/dir_all_alphabetic.html
```

安装

```
[root@rokcy8 ~]# yum install man-pages
```

man 页面分组

不同类型的帮助称为不同的“章节”，统称为Linux手册，man 1 man

```
[root@ubuntu2204 ~]# man man
......
......
```

标准man 手册一般有以下几个章节:

1. 可执行程序或 shell 命令
2. 系统调用(内核提供的函数)
3. 库调用(程序库中的函数)
4. 特殊文件(通常位于 /dev)
5. 文件格式和规范，如 /etc/passwd
6. 游戏
7. 杂项
8. 系统管理命令(通常只针对 root 用户)
9. 内核API

man 命令的配置文件:

```
#CentOS 6 之前版 man 的配置文件
/etc/man.config

#CentOS 7 之后版 man 的配置文件
/etc/man_db.conf

#ubuntu man 的配置文件
/etc/manpath.config
```

man 命令常见用法:

```
#man[选项...] [章节] 手册页...

man passwd                          #默认打开第一章帮助
man 5 passwd                        #指定第五章帮助
man 9 passwd                        #总共9个章节，中间的数字不能超过9，如果要找的章节里面
没有对应的内容，则会提示没有
man 10 passwd                       #总共9个章节，中间的数字不能超过9，如果是数字大于9，
则直接定位到最小的章

man -a passwd                       #打开所有帮助
man -f passwd                       #显示passwd 相关的章节，相当于 whatis passwd
man -k passwd                       #在man 手册中搜索所有与passwd 相关的内容，使用
whatis 数据库
```

```
man -w ls                                   #显示ls的man 文件路径
man -aw passwd                              #显示所有跟passwd有关的man文件路径
man -w 5 passwd                             #显示第5章的passwd的man文件路径
man -M /usr/local/share/man/zh_CN           #显示中文man 的内容



#常用快捷键
e|ctrl+E|j|ctrl+J|enter丨往下方向键  往下一行
y|ctrl+Y|k|ctrl+K|ctrl+P丨往上方向键 往上一行
z|f|ctrl+F|ctrl+V丨space 往下一屏
w|b|ctrl+B|esc+v 往上一屏
d|ctrl+D 往下半屏
u|ctrl+U 往上半屏
1G 回到首行
G 跳转至结尾
/abc  向下搜索abc  按n向下跳转，按N向上跳转
?abc  向上搜索abc  按n向下跳转，按N向上跳转
q 退出



#常用关键字及格式
[] 可选项
<> 必选项
a|b 二选一
... 同一内容可出现多次
{ } 分组



#段落说明
NAME                    #名称及说明
DESCRIPTION             #详细说明
SYNOPSIS                #使用格式
FILES                   #相关文件
OPTIONS                 #选项
AUTHOR                  #作者
REPORTING BUGS          #相关bug信息
COPYRIGHT               #版权及开源协议信息
EXAMPLES                #使用示例
SEE ALSO                #可参考其它部份



#其它命令

#显示man手册文件路径
[root@ubuntu2204 ~]# manpath
/usr/local/man:/usr/local/share/man:/usr/share/man

#指定man手册文件路径
[root@ubuntu2204 ~]# manpath  /path/dir
```

```
#在指定位置下搜索,自定义安装的软件，用man查看帮助时，可以指定man文件地址
[root@ubuntu2204 ~]# man -M /path/dir/  COMMAND
```

范例: 直接打开man的帮助文件

```
[root@ubuntu2204 ~]# man -w ls
/usr/share/man/man1/ls.1.gz

[root@ubuntu2204 ~]# file /usr/share/man/man1/ls.1.gz
/usr/share/man/man1/ls.1.gz: gzip compressed data, max compression, from Unix,
original size modulo 2^32 8049

#直接打开man的gz压缩格式文档
[root@ubuntu2204 ~]# man /usr/share/man/man1/ls.1.gz
......


#直接打开man格式文档
[root@ubuntu2204 ~]# file /root/nginx-1.22.1/man/nginx.8
/root/nginx-1.22.1/man/nginx.8: troff or preprocessor input, ASCII text

[root@ubuntu2204 ~]# man -l /root/nginx-1.22.1/man/nginx.8
```

范例: CentOS7 显示中文帮助

```
[root@centos7 ~]#yum -y install man-pages-zh-CN.noarch
[root@centos7 ~]#rpm -ql man-pages-zh-CN.noarch

#直接查看中文帮助
[root@centos7 ~]#man -l /usr/share/man/zh_CN/man8/iptables.8.gz

#默认为英文帮助
[root@centos7 ~]#man iptables

#修改语言环境
[root@centos7 ~]#echo $LANG
[root@centos7 ~]#LANG=zh_CN.UTF-8
[root@centos7 ~]#localectl set-locale LANG=zh_CN.utf8

#中文帮助
[root@centos7 ~]#man iptables
```

练习:

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 命令

info 是自由软件基金会的GNU项目，是GNU的超文本帮助系统，

整个结构类似于一个网站，有导航，支持链接跳转

不带参数，默认进入的是首页

```
#info [OPTION]... [MENU-ITEM...]

info          #进入整个info文档
info ls       #在info 中查看ls的信息


#常用快捷键
向上方向键     #上移一行
向下方向键     #下移一行
PgUp          #向上一屏
PgDn          #向下一屏
Tab           #在链接间滚动
Enter         #进入链接查看具体内容
s|/           #搜索
n/p/u/l       #进入下/前/上一层/最后一个链接
q             #退出
```

2.6 命令自身提供的官方使用指南

```
/usr/share/doc
```

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

范例

```
[root@ubuntu2204 ~]# ls /usr/share/doc/nano/
AUTHORS              copyright  faq.html         nano.html  README.gz  TODO
changelog.Debian.gz  examples   IMPROVEMENTS.gz  NEWS.gz    THANKS.gz

[root@ubuntu2204 ~]# ls /usr/share/doc/nano/nano.html
/usr/share/doc/nano/nano.html
```

2.7 系统及第三方应用官方文档

2.7.1 Linux官方在线文档和知识库

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://www.redhat.com/docs
http://kbase.redhat.com
http://access.redhat.com
https://help.ubuntu.com/lts/serverguide/index.html
http://tldp.org
```

2.7.2 通过在线文档获取帮助

```
http://www.github.com
https://www.kernel.org/doc/html/latest/
http://httpd.apache.org
http://www.nginx.org
https://mariadb.com/kb/en
https://dev.mysql.com/doc/
http://tomcat.apache.org
https://jenkins.io/zh/doc/
https://kubernetes.io/docs/home/
https://docs.openstack.org/train/
http://www.python.org
http://php.net
```

2.8 TLDR 命令

当我们在使用一个不熟悉的命令时，可以使用 -h 或 --help 选项来查看帮助，或者使用 man 手册还查看更详细的文档，但这两种方式，会列出所有选项，而有些选项很少使用，根据二八原则，只有一小部份选项才是最常用的，如是基于此，有人开发了一个开源的查看命令工具，此工具只列出命令的常用选项的帮助。

TLDR: Too Long；Didn't Read(太长不看)，也可以叫作 "偷懒的人"

项目主页

```
https://github.com/tldr-pages/tldr
```

```
[root@ubuntu2204 ~]# apt install python3-pip

[root@ubuntu2204 ~]# pip3 install tldr


[root@ubuntu2204 ~]# vim /etc/hosts

185.199.109.133  raw.githubusercontent.com


#查看 ls 命令帮助
[root@ubuntu2204 ~]# tldr ls

  ls

  List directory contents.
  More information: https://www.gnu.org/software/coreutils/ls.

  - List files one per line:
    ls -1

  - List all files, including hidden files:
    ls -a
```

```
  - List all files, with trailing `/` added to directory names:
    ls -F

  - Long format list (permissions, ownership, size, and modification date) of
all files:
    ls -la

  - Long format list with size displayed using human-readable units (KiB, MiB,
GiB):
    ls -lh

  - Long format list sorted by size (descending):
    ls -lS

  - Long format list of all files, sorted by modification date (oldest first):
    ls -ltr

  - Only list directories:
    ls -d */
```

2.9 相关网站和搜索

```
http://www.google.com
http://bing.com
http://www.baidu.com
http://www.slideshare.net
```