

ifconfig

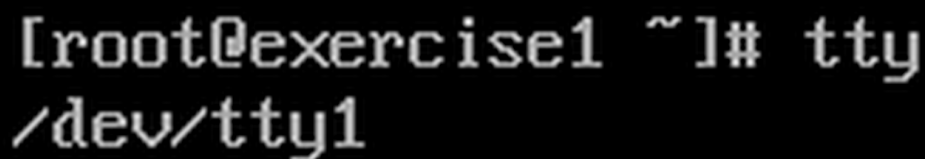
```
1 用于查看IP地址，用于显示或设置网络设备
2 记录ip地址方便使用远程连接工具(xshell、MobaXterm等)
3
4 也是我们第一条要熟悉的命令
5
6 [root@exercise1 ~]# ifconfig
7 ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
   1500
8         inet 192.168.119.141  netmask 255.255.255.0
   broadcast 192.168.119.255
9         inet6 fe80::d223:f0d0:c686:786f  prefixlen 64
   scopeid 0x20<link>
10        ether 00:0c:29:51:bc:aa  txqueuelen 1000
   (Ethernet)
11        RX packets 1967  bytes 173443 (169.3 KiB)
12        RX errors 0  dropped 0  overruns 0  frame 0
13        TX packets 1467  bytes 145537 (142.1 KiB)
14        TX errors 0  dropped 0 overruns 0  carrier 0
   collisions 0
15
16 lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
17        inet 127.0.0.1  netmask 255.0.0.0
18        inet6 ::1  prefixlen 128  scopeid 0x10<host>
19        loop txqueuelen 1  (Local Loopback)
20        RX packets 0  bytes 0 (0.0 B)
21        RX errors 0  dropped 0  overruns 0  frame 0
22        TX packets 0  bytes 0 (0.0 B)
23        TX errors 0  dropped 0 overruns 0  carrier 0
   collisions 0
```

ifconfig == ip a 其中ens33是网卡，lo是回环地址

```
1 [root@exercise1 ~]# ip a
2 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
   state UNKNOWN qlen 1
3     link/loopback 00:00:00:00:00:00 brd
   00:00:00:00:00:00
4     inet 127.0.0.1/8 scope host lo
5         valid_lft forever preferred_lft forever
6     inet6 ::1/128 scope host
7         valid_lft forever preferred_lft forever
8 2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
   qdisc pfifo_fast state UP qlen 1000
9     link/ether 00:0c:29:51:bc:aa brd ff:ff:ff:ff:ff:ff
10    inet 192.168.119.141/24 brd 192.168.119.255 scope
   global dynamic ens33
11        valid_lft 1577sec preferred_lft 1577sec
12    inet6 fe80::d223:f0d0:c686:786f/64 scope link
13        valid_lft forever preferred_lft forever
```

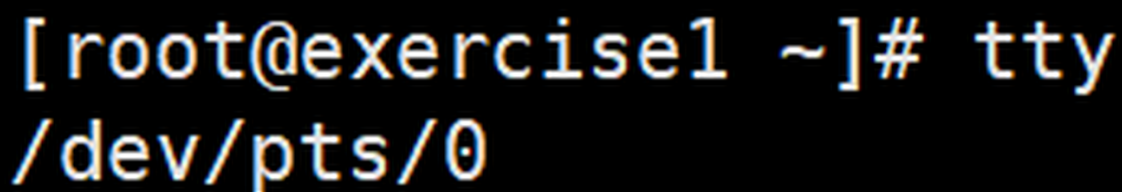
tty

- 1 通过 **tty** 命令看到当前所属的虚拟终端！
- 2 下图是本机打开的命令行终端



```
[root@exercise1 ~]# tty
/dev/tty1
```

- 1 这张图是本地图形界面下打开的终端或者是远程连接时打开的终端



```
[root@exercise1 ~]# tty
/dev/pts/0
```

- 1 总结可以得出：**tty**命令执行结果显示的是/**dev/tty**数字，则是本机打开的命令行终端，
- 2 若是/**dev/pts**/数字，则是本地图形界面下打开的终端或是远程连接打开的终端。
- 3 **alt +** 左右箭头，在多个本地**tty**命令行终端之间进行切换。新建**tty**终端。

用户身份

- 1 **#**与**\$**的区别：
- 2 **#**代表的是超级用户(**root**)
- 3 **\$**代表的是普通用户

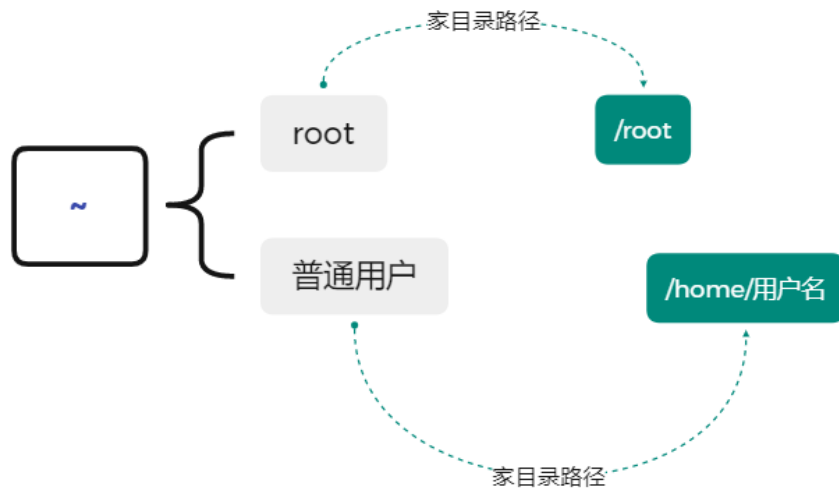
[root@exercise1 ~]# ###表示是 root 用户登录，管理员账号登陆

[root@exercise1 ~]# su - abc ###切换到 abc 普通用户

[abc@exercise1 ~]\$ ###表示普通用户登录

```
[root@exercise1 ~]#  
[root@exercise1 ~]# su - abc  
[abc@exercise1 ~]$
```

- 1 其中[]里面各项内容的意思
- 2 [root @ exercise1 ~]
 #
- 3 [用户名 @ 主机名 当前所在的目录(~代表的是当前用户的家目录)] 代表用户身份(管理员/普通用户)
- 4
- 5 /root是root的家目录
- 6 /home是普通用户的家目录



认识 SHELL

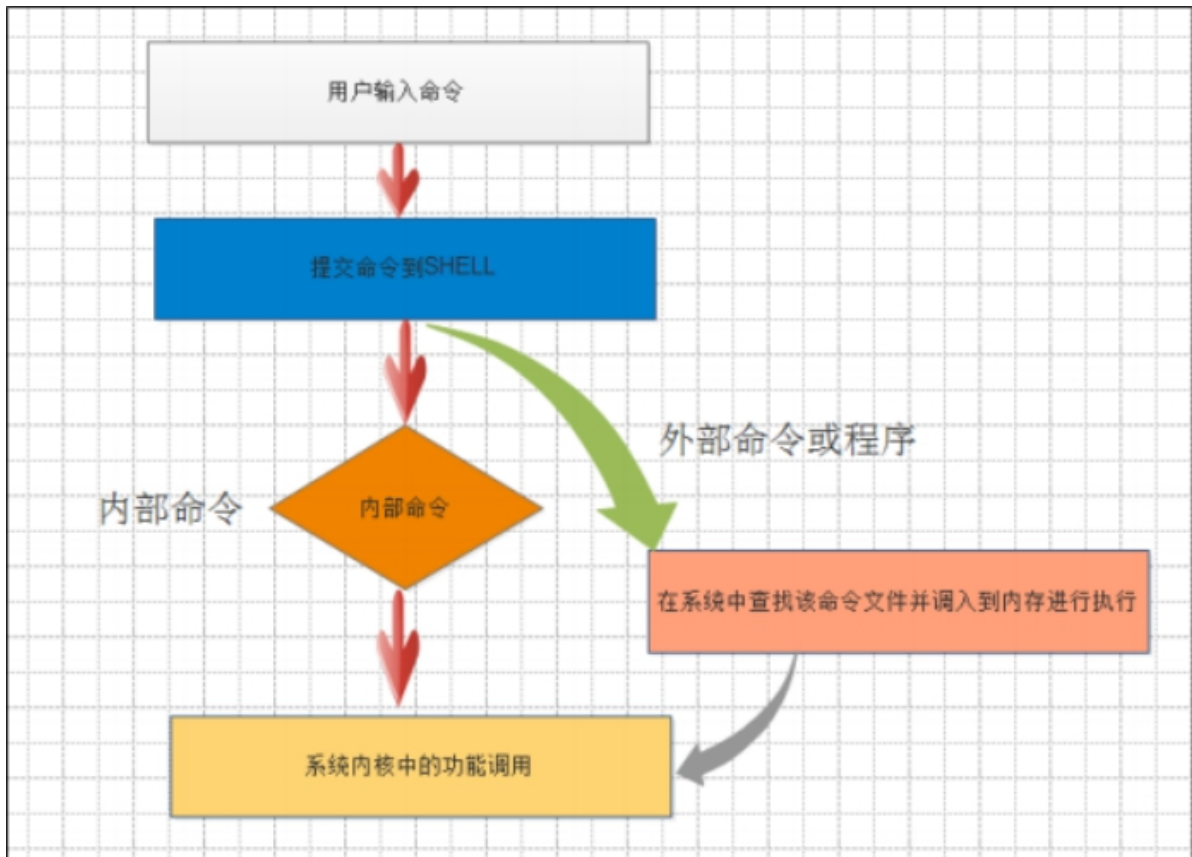
- 1 **Shell** 俗称壳，它提供了用户与内核进行交互操作的一种接口，
- 2 它接收用户输入的命令并把它送入内核去执行，**shell** 实际上是一个命令解释器，它通过解释用户输入的命令
- 3 并把它传输给系统内核去执行。
- 4 **Shell** 有自己的编程语言用于对命令的编辑，它允许用户编写由**shell** 命令组成的程序。
- 5 **Shell** 编程语言具有普通编程语言的很多特点，比如它也有循环结构和分支控制结构等，
- 6 用这种编程语言编写的 **shell** 程序不其他应用程序具有同样的效果。

type命令区分内外部命令

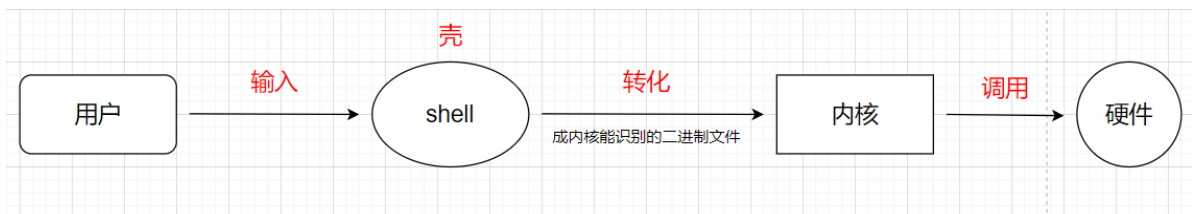
- 1 内部命令：在系统启动时就调入内存，是常驻内存的，所以执行效率高
- 2 外部命令：是系统软件的功能，用户需要时才从硬盘中读入内存 如何区分内外部命令？
- 3
- 4 使用 **type** 命令
- 5 语法：**type** 要检测的命令
- 6
- 7 命令：
- 8 `[root@exercise1 ~]# type cd`
- 9 `cd 是 shell 内嵌`
- 10 `[root@exercise1 ~]# type vi`
- 11 `vi 是 /usr/bin/vi`

```
[root@exercise1 ~]# type cd
cd 是 shell 内嵌
[root@exercise1 ~]# type vi
vi 是 /usr/bin/vi
```

图一



图二



经典shell报错

```
1 [root@exercise1 ~]# ll
2 -bash: ll: 未找到命令
3 [root@exercise1 ~]# cd /opt1
4 -bash: cd: /opt1: 没有那个文件或目录
```

```
[root@exercise1 ~]# ll  
-bash: ll: 未找到命令  
[root@exercise1 ~]# cd /opt1  
-bash: cd: /opt1: 没有那个文件或目录
```

总结

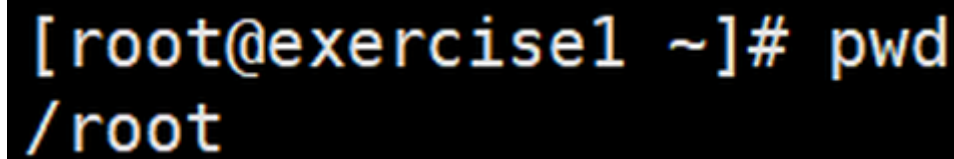
- 1 通过SHELL，我们可以对 LINUX 实现哪些操作或管理： 例如：
- 2 1、对文件的管理（创建、删除、复制、修改）
- 3 2、对用户的管理（添加、删除）
- 4 3、相关权限的管理（授权用户对相关文件的管理，比如增删改查）
- 5 4、对磁盘的管理（分区、raid、lvm）
- 6 5、对软件的管理
- 7 6、对网络的管理
- 8 我们要活用Tab键（用好tab键，操作效率提升 70-80%） 智能查找
- 9
- 10 补全命令，补全 操作 的 文件和目录 的名称。
- 11 两次 tab，列出 相似的命令或者 文件目录。

路径

- 1 路径：分为绝对路径与相对路径
- 2
- 3 绝对路径
- 4 以正斜线“/”开头
- 5 描述道文件位置的完整说明
- 6 任何时候你想指定文件名的时候都可以使用
- 7 cd /opt/aa
- 8
- 9 相对路径
- 10 不以正斜线开头
- 11 制定相对于你的当前工作目录而言的位置
- 12 可以使用做制定文件名的快捷方式
- 13 相对路径：cd ../opt

pwd

```
1 查看当前所在的工作目录
2
3 [root@exercise1 ~]# pwd
4 /root
```



```
[root@exercise1 ~]# pwd
/root
```

命令格式

```
1 命令 [选项] [参数]
```

cd

```
1 作用：切换目录（change directory）
2 语法：cd 目录
3 说明：直接输入 cd 表示回到当前用户的宿主（家）目录
4
5 例子1：
6 [root@exercise1 ~]# cd /etc/sysconfig/network-scripts/
7 [root@exercise1 network-scripts]# cd
8 [root@exercise1 ~]# cd ~
```

cd .. 表示返回到上级目录位置，也就是父目录

cd . 表示进入到当前用户所在的目录

例子2：

```
[root@exercise1 ~]# cd /etc/sysconfig/network-scripts/
```

```
[root@exercise1 network-scripts]# cd ../ #返回上一级目录，
```

即/etc/sysconfig

```
[root@exercise1 sysconfig]# cd ./network-scripts/ #进去当前目录下的network-scripts目录
```

```
[root@exercise1 network-scripts]#
```

例子3：

```
[root@exercise1 network-scripts]# cd - #表示返回切换前的目录
```

/etc/sysconfig

[root@exercise1 sysconfig]#

ls

```
1 作用：查看当前目录下有哪些文件（list）
2 语法：ls 目录/文件，如果什么也不加，那么查看的是当前目录下的内容
3 常用选项：命令后面不加任何选项、-l、-a、-S、-h、-t
4
5 例子1：
6 -l 列出文件的详细信息，如创建者，创建时间，文件的读写权限列表等等，
   长列表
7 [root@exercise1 ~]# ls -l
8 总用量 4
9 -rw-----. 1 root root 1390 1月 9 09:31 anaconda-
   ks.cfg
```

例子2：

-a 列出目录下所有的文件，包括以“.”开头的隐藏文件

(linux下隐藏文件是以.开头的，如果存在2个点代表存在着父目录,1个点表示当前目录)

```
[root@exercise1 ~]# ls -a
```

```
. .. anaconda-ks.cfg .bash_history .bash_logout .bash_profile
.bashrc .cshrc .tcshrc
```

例子3：

-S 以文件的大小进行排序

```
[root@exercise1 ~]# ls -lS
```

总用量 4

```
-rw-----. 1 root root 1390 1月 9 09:31 anaconda-ks.cfg
drwxr-xr-x. 2 root root 6 1月 9 13:05 a
```

例子5：

-h 以可读性较高容量显示

```
[root@exercise1 etc]# ls -lh
```

总用量 1.1M

```
-rw-r--r--. 1 root root 16 1月 9 09:31 adjtime
-rw-r--r--. 1 root root 1.5K 6月 7 2013 aliases
```



```
-rw-r--r--. 1 root root 12K 1月 9 09:43 aliases.db
drwxr-xr-x. 2 root root 236 1月 9 09:29 alternatives
-rw-----. 1 root root 541 8月 3 2017 anacrontab
-rw-r--r--. 1 root root 55 3月 1 2017 asound.conf
drwxr-x---. 3 root root 43 1月 9 09:29 audisp
```

例子6:

-t 以时间反向顺序进行排序

```
[root@exercise1 etc]# ls -lt
```

总用量 1040

```
-----. 1 root root 356 1月 9 11:20 gshadow
-rw-r--r--. 1 root root 449 1月 9 11:20 group
-----. 1 root root 585 1月 9 11:20 shadow
-rw-r--r--. 1 root root 846 1月 9 11:20 passwd
-----. 1 root root 707 1月 9 10:30 shadow-
-rw-r--r--. 1 root root 461 1月 9 10:30 group-
```

```
ls -l == ll
```

```
[root@exercise1 ~]# ls -l
```

总用量 4

```
-rw-----. 1 root root 1390 1月 9 09:31 anaconda-ks.cfg
```

```
[root@exercise1 ~]# ll
```

总用量 4

```
-rw-----. 1 root root 1390 1月 9 09:31 anaconda-ks.cfg
```

```
[root@exercise1 ~]# type ll
```

ll 是 `ls -l --color=auto' 的别名

别名

- 1 有时候在Linux下执行一条命令过长，挺麻烦的，
- 2 我们可以在**.bashrc**文件下设置**alias**，对命令设置简短的别名，相当于缩短命令，方便操作。
- 3 分临时别名和永久别名，临时别名只要关机或者重启就失效了；所以如果有临时和永久设置最好
- 4 两个都设置，做两手准备。

临时别名例子：

[root@exercise1 ~]# alias ens33='cat /etc/sysconfig/network-scripts/ifcfg-ens33' #设置别名

[root@exercise1 ~]# en #设置成功后使用Tab键能显示补全

enable ens33 env envsubst

[root@exercise1 ~]# ens33 #成功执行本质命令cat 文件

TYPE="Ethernet"

PROXY_METHOD="none"

BROWSER_ONLY="no"

BOOTPROTO="dhcp"

DEFROUTE="yes"

IPV4_FAILURE_FATAL="no"

IPV6INIT="yes"

IPV6_AUTOCONF="yes"

IPV6_DEFROUTE="yes"

IPV6_FAILURE_FATAL="no"

IPV6_ADDR_GEN_MODE="stable-privacy"

NAME="ens33"

UUID="e385b4ab-ff26-429b-82ec-25e3a9929163"

DEVICE="ens33"

ONBOOT="yes"

卸载临时别名：

[root@exercise1 ~]# unalias ens33 #卸载别名 格式：unalias 别名名称

[root@exercise1 ~]# en #此时Tab键已经不会显示ens33证明卸载成功

enable env envsubst

临时取消别名：

[root@exercise1 ~]# type ll

ll 是 `ls -l --color=auto' 的别名

[root@exercise1 ~]# \ll #使用\临时取消别名

-bash: ll: 未找到命令

[root@exercise1 ~]#

设置永久生效

当前用户使用：

[root@exercise1 ~]# vi ~/.bashrc #.bashrc文件是在家目录下面的，所以

用~既可以代表root用户，也可以代表普通用户

文件最后插入alias ens33='cat /etc/sysconfig/network-scripts/ifcfg-ens33'

[root@exercise1 ~]# source ~/.bashrc #刷新文件，

不必重启机子即可生效(重启机子是高危操作)

[root@exercise1 ~]# ens33 #测试

TYPE="Ethernet"

PROXY_METHOD="none"

BROWSER_ONLY="no"

BOOTPROTO="dhcp"

DEFROUTE="yes"

IPV4_FAILURE_FATAL="no"

IPV6INIT="yes"

IPV6_AUTOCONF="yes"

IPV6_DEFROUTE="yes"

IPV6_FAILURE_FATAL="no"

IPV6_ADDR_GEN_MODE="stable-privacy"

NAME="ens33"

UUID="e385b4ab-ff26-429b-82ec-25e3a9929163"

DEVICE="ens33"

ONBOOT="yes"

全局使用：

[root@exercise1 ~]# vi /etc/bashrc

文件最后插入alias opt='cd /opt'

[root@exercise1 ~]# su - abc

[abc@exercise1 ~]opt[abc@exercise1opt]

touch

- 1 作用：常用来创建空文件，如果文件存在，则修改这个文件的时间
- 2 语法：touch 文件名

```
3
4 例子1:
5 创建单个文件
6 [root@exercise1 ~]# cd /opt/
7 [root@exercise1 opt]# ll
8 总用量 0
9 [root@exercise1 opt]# touch a.txt
10 [root@exercise1 opt]# ll
11 总用量 0
12 -rw-r--r--. 1 root root 0 1月  9 13:36 a.txt
13
14 例子2:
15 创建多个文件
16     方法一:
17     [root@exercise1 opt]# ll
18     总用量 0
19     -rw-r--r--. 1 root root 0 1月  9 13:36 a.txt
20     [root@exercise1 opt]# touch b.txt c.txt
21     [root@exercise1 opt]# ll
22     总用量 0
23     -rw-r--r--. 1 root root 0 1月  9 13:36 a.txt
24     -rw-r--r--. 1 root root 0 1月  9 13:37 b.txt
25     -rw-r--r--. 1 root root 0 1月  9 13:37 c.txt
26
27     方法二:
28     [root@exercise1 opt]# ll
29     总用量 0
30     -rw-r--r--. 1 root root 0 1月  9 13:36 a.txt
31     -rw-r--r--. 1 root root 0 1月  9 13:37 b.txt
32     -rw-r--r--. 1 root root 0 1月  9 13:37 c.txt
33     [root@exercise1 opt]# touch {1..5}
34     [root@exercise1 opt]# ll
35     总用量 0
36     -rw-r--r--. 1 root root 0 1月  9 13:38 1
37     -rw-r--r--. 1 root root 0 1月  9 13:38 2
38     -rw-r--r--. 1 root root 0 1月  9 13:38 3
39     -rw-r--r--. 1 root root 0 1月  9 13:38 4
40     -rw-r--r--. 1 root root 0 1月  9 13:38 5
41     -rw-r--r--. 1 root root 0 1月  9 13:36 a.txt
42     -rw-r--r--. 1 root root 0 1月  9 13:37 b.txt
```

```

43 -rw-r--r--. 1 root root 0 1月 9 13:37 c.txt
44 [root@exercise1 opt]# touch {d..f}.txt
45 [root@exercise1 opt]# ll
46 总用量 0
47 -rw-r--r--. 1 root root 0 1月 9 13:38 1
48 -rw-r--r--. 1 root root 0 1月 9 13:38 2
49 -rw-r--r--. 1 root root 0 1月 9 13:38 3
50 -rw-r--r--. 1 root root 0 1月 9 13:38 4
51 -rw-r--r--. 1 root root 0 1月 9 13:38 5
52 -rw-r--r--. 1 root root 0 1月 9 13:36 a.txt
53 -rw-r--r--. 1 root root 0 1月 9 13:37 b.txt
54 -rw-r--r--. 1 root root 0 1月 9 13:37 c.txt
55 -rw-r--r--. 1 root root 0 1月 9 13:39 d.txt
56 -rw-r--r--. 1 root root 0 1月 9 13:39 e.txt
57 -rw-r--r--. 1 root root 0 1月 9 13:39 f.txt
58 [root@exercise1 opt]# ll
59 总用量 0
60 -rw-r--r--. 1 root root 0 1月 9 13:38 1
61 -rw-r--r--. 1 root root 0 1月 9 13:38 2
62 -rw-r--r--. 1 root root 0 1月 9 13:38 3
63 -rw-r--r--. 1 root root 0 1月 9 13:38 4
64 -rw-r--r--. 1 root root 0 1月 9 13:38 5
65 -rw-r--r--. 1 root root 0 1月 9 13:36 a.txt
66 -rw-r--r--. 1 root root 0 1月 9 13:37 b.txt
67 -rw-r--r--. 1 root root 0 1月 9 13:37 c.txt
68 -rw-r--r--. 1 root root 0 1月 9 13:39 d.txt
69 -rw-r--r--. 1 root root 0 1月 9 13:39 e.txt
70 -rw-r--r--. 1 root root 0 1月 9 13:39 f.txt
71 [root@exercise1 opt]# touch 1{d..f}
72 [root@exercise1 opt]# ll
73 总用量 0
74 -rw-r--r--. 1 root root 0 1月 9 13:38 1
75 -rw-r--r--. 1 root root 0 1月 9 13:41 1d
76 -rw-r--r--. 1 root root 0 1月 9 13:41 1e
77 -rw-r--r--. 1 root root 0 1月 9 13:41 1f
78 -rw-r--r--. 1 root root 0 1月 9 13:38 2
79 -rw-r--r--. 1 root root 0 1月 9 13:38 3
80 -rw-r--r--. 1 root root 0 1月 9 13:38 4
81 -rw-r--r--. 1 root root 0 1月 9 13:38 5
82 -rw-r--r--. 1 root root 0 1月 9 13:36 a.txt

```

```

83 -rw-r--r--. 1 root root 0 1月 9 13:37 b.txt
84 -rw-r--r--. 1 root root 0 1月 9 13:37 c.txt
85 -rw-r--r--. 1 root root 0 1月 9 13:39 d.txt
86 -rw-r--r--. 1 root root 0 1月 9 13:39 e.txt
87 -rw-r--r--. 1 root root 0 1月 9 13:39 f.txt
88 [root@exercise1 opt]#
89
90 例子3:
91 指定路径创建
92 [root@exercise1 opt]# cd
93 [root@exercise1 ~]# touch /opt/11.txt
94 [root@exercise1 ~]# ls /opt/
95 1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt
   d.txt e.txt f.txt

```

创建文件方法：touch、vi/vim、重定向、tee、cat、软连接

mkdir

```

1 作用：创建目录
2 语法：mkdir (选项) 文件名
3 在创建一个目录的时候，如果这个目录的上一级不存在的话，要加参数-p
4
5 例子1:
6 创建单个目录
7 [root@exercise1 opt]# 11
8 总用量 0
9 -rw-r--r--. 1 root root 0 1月 9 13:38 1
10 -rw-r--r--. 1 root root 0 1月 9 13:46 11.txt
11 -rw-r--r--. 1 root root 0 1月 9 13:41 1d
12 -rw-r--r--. 1 root root 0 1月 9 13:41 1e
13 -rw-r--r--. 1 root root 0 1月 9 13:41 1f
14 -rw-r--r--. 1 root root 0 1月 9 13:38 2
15 -rw-r--r--. 1 root root 0 1月 9 13:38 3
16 -rw-r--r--. 1 root root 0 1月 9 13:38 4
17 -rw-r--r--. 1 root root 0 1月 9 13:38 5
18 -rw-r--r--. 1 root root 0 1月 9 13:36 a.txt
19 -rw-r--r--. 1 root root 0 1月 9 13:37 b.txt
20 -rw-r--r--. 1 root root 0 1月 9 13:37 c.txt
21 -rw-r--r--. 1 root root 0 1月 9 13:39 d.txt

```

```
22 -rw-r--r--. 1 root root 0 1月 9 13:39 e.txt
23 -rw-r--r--. 1 root root 0 1月 9 13:39 f.txt
24 [root@exercise1 opt]# mkdir g
25 [root@exercise1 opt]# ls
26 1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt
   d.txt e.txt f.txt g
```

例子2:

创建多个目录

```
[root@exercise1 opt]# ls
1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt d.txt e.txt f.txt g
[root@exercise1 opt]# mkdir h i
[root@exercise1 opt]# ls
1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt d.txt e.txt f.txt g h i
```

例子3:

指定路径创建单个目录

```
[root@exercise1 opt]# ls
1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt d.txt e.txt f.txt g h i
[root@exercise1 opt]# cd
[root@exercise1 ~]# mkdir /opt/j
[root@exercise1 ~]# ls /opt/
1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt d.txt e.txt f.txt g h i j
```

例子4:

在创建一个目录的时候，如果这个目录的上一级不存在的话，要加参数-p

```
[root@exercise1 opt]# mkdir z/1.txt
mkdir: 无法创建目录"z/1.txt": 没有那个文件或目录
[root@exercise1 opt]# mkdir -p z/1.txt
[root@exercise1 opt]# ls
1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt d.txt e.txt f.txt g h i j
z
[root@exercise1 opt]# ls z
1.txt
```

rm

- 1 作用：可以删除一个目录中的一个或多个文件或目录，对于链接文件，只是删除整个链接文件，而原文件保持不变的
- 2 语法：rm (选项) 处理对象
- 3 选项：

-f 强制删除，没有提示

-r 删除目录，递归

例子1：

单删文件或文件夹

```
[root@exercise1 opt]# ls
```

```
1 11.txt 1d 1e 1f 2 3 4 5 a.txt b.txt c.txt d.txt e.txt f.txt g h  
i j z
```

```
[root@exercise1 opt]# rm -f a.txt
```

```
[root@exercise1 opt]# ls
```

```
1 11.txt 1d 1e 1f 2 3 4 5 b.txt c.txt d.txt e.txt f.txt g h i j z
```

1 例子2：

2 文件和文件夹混合删

```
3 [root@exercise1 opt]# rm -f b.txt g
```

```
4 [root@exercise1 opt]# ls
```

```
5 1 11.txt 1d 1e 1f 2 3 4 5 c.txt d.txt  
e.txt f.txt h i j z
```

6

7 例子3：

8 递归删除

```
9 [root@exercise1 opt]# rm -rf z
```

```
10 [root@exercise1 opt]# ls
```

```
11 1 11.txt 1d 1e 1f 2 3 4 5 c.txt d.txt  
e.txt f.txt h i j
```

12

13 例子4：

14 模糊删除

```
15 [root@exercise1 opt]# ls
```

```
16 1 1d 1e 1f 2 3 4 5 c.txt d.txt e.txt  
f.txt i.bak j log log.bak messages
```

```
17 [root@exercise1 opt]# rm -rf *.txt*
```



```
18 [root@exercise1 opt]# ls
19 1 1d 1e 1f 2 3 4 5 i.bak j log log.bak
   messages
20
21 注意：用rm删除东西的时候，老林建议去到对应的目录下用相对路径去删除，防止误删
```

cp

```
1 语法：cp 源文件/目录 目录文件/目录
2 选项：
```

-R/r：递归处理，将指定目录下的所有文件与子目录一并处理
-p：保持源文件的属性在拷贝的过程中,不发生变化;

例子1：

复制文件

```
[root@exercise1 opt]# cp /var/log/messages /opt/
```

```
[root@exercise1 opt]# ls
```

```
1 11.txt 1d 1e 1f 2 3 4 5 c.txt d.txt e.txt f.txt h i j messages
```

```
1 例子2：
```

```
2 复制目录
```

```
3 [root@exercise1 opt]# cp -r /var/log /opt/
```

```
4 [root@exercise1 opt]# ls
```

```
5 1 11.txt 1d 1e 1f 2 3 4 5 c.txt d.txt e.txt
   f.txt h i j log messages
```

```
6
```

```
7 例子3：
```

```
8 复制并改名字
```

```
9 [root@exercise1 opt]# cp -r /var/log /opt/log.bak
```

```
10 [root@exercise1 opt]# ls
```

```
11 1 11.txt 1d 1e 1f 2 3 4 5 c.txt d.txt e.txt
   f.txt h i j log log.bak
```

```
12
```

```
13 例子4:
14 一次拷贝多个文件:/etc/hostname /etc/fstab /var /tmp /root
   /home-->/backup
15 [root@exercise1 opt]# \cp -r /etc/hostname /etc/fstab
   /var/ /home/ /tmp/ /root/ /backup/
16 # 前提: /backup目录必须存在;
17 # 最后一个目录,一定是目标;
```

mv

```
1 语法: mv    源文件/目录    目录文件/目录
2
3 例子1:
4 移动目录
5 [root@exercise1 opt]# ls
6 1  11.txt  1d  1e  1f  2  3  4  5  c.txt  d.txt  e.txt
   f.txt  h  i  j  log  log.bak  messages
7 [root@exercise1 opt]# mv h i
8 [root@exercise1 opt]# ls
9 1  11.txt  1d  1e  1f  2  3  4  5  c.txt  d.txt  e.txt
   f.txt  i  j  log  log.bak  messages
10 [root@exercise1 opt]# ls i
11 h
12 [root@exercise1 opt]#
13
14 例子2:
15 移动文件
16 [root@exercise1 opt]# mv 11.txt i
17 [root@exercise1 opt]# ls
18 1  1d  1e  1f  2  3  4  5  c.txt  d.txt  e.txt  f.txt
   i  j  log  log.bak  messages
19 [root@exercise1 opt]# ls i
20 11.txt  h
21 [root@exercise1 opt]#
22
23 例子3:
24 移动并改名字
25 [root@exercise1 opt]# ls
```

```
26 1 1d 1e 1f 2 3 4 5 c.txt d.txt e.txt f.txt
    i j log log.bak messages
27 [root@exercise1 opt]# mv i i.bak
28 [root@exercise1 opt]# ls
29 1 1d 1e 1f 2 3 4 5 c.txt d.txt e.txt f.txt
    i.bak j log log.bak messages
30
31 例子4:
32 一次移动多个文件
33 [root@exercise1 opt]# mkdir test
34 [root@exercise1 opt]# mv c.txt d.txt e.txt test
35 #目录test必须在最后面, 而且它前面不能再出现其他目录
```

cat

```
1 选项:
2
3     -n    显示行号
4
5 语法: cat 文件名
6 作用: 查看文件内容, 一次显示整个文件的内容
7
8 例子1:
9 [root@exercise1 opt]# cat /var/log/messages
10
11 例子2:
12 [root@exercise1 opt]# cat /var/log/messages
    /opt/messages > 1
13 #文件/var/log/messages, /opt/messages合并到1
```

more

```
1 作用: 以分页形式显示文件内容
2 语法: more + 文件名
3 说明: 按下回车刷新一行, 按下空格刷新一屏, 按b返回上一屏, 输入 q 键退出
```

less

- 1 作用:和 `more` 功能一样
- 2 语法:`less +文件名`
- 3 说明: 按下回车刷新一行, 按下空格刷新一屏, 按**b**返回上一屏, 输入 `q` 键退出

more 和 less的区别:

- 1 1. `less`可以按键盘上下方向键显示上下内容,`more`不能通过上下方向键控制显示,但是可以通过`ctrl+B`返回上一页。
- 2 2. `less`不必读整个文件,加载速度会比`more`更快(适用于超过1G以上文件)
- 3 3. `less`退出后`shell`不会留下刚显示的内容,而`more`退出后会在`shell`上留下刚显示的内容, `ctrl+n`可以一行行删除

head

- 1 作用: 用于显示文件的开头的内容。在默认情况下, `head` 命令显示文件的头 10 行内容
- 2 语法:`head(选项)文件名`
- 3 参数:
- 4 `-n` 显示从文件头开始的行数
- 5 `-c<数目>` 显示的字节数
- 6
- 7 例子1:
- 8 `[root@exercise1 opt]# head /var/log/messages`
- 9 `Jan 9 09:43:19 exercise1 journal: Runtime journal is`
`using 6.1M (max allowed 48.8M, trying to leave 73.2M`
`free of 482.0M available → current limit 48.8M).`
- 10 `Jan 9 09:43:19 exercise1 kernel: Initializing cgroup`
`subsys cpuset`
- 11 `Jan 9 09:43:19 exercise1 kernel: Initializing cgroup`
`subsys cpu`

```
12 Jan  9 09:43:19 exercise1 kernel: Initializing cgroup
    subsys cpuacct
13 Jan  9 09:43:19 exercise1 kernel: Linux version 3.10.0-
    693.el7.x86_64 (builder@kbuilder.dev.centos.org) (gcc
    version 4.8.5 20150623 (Red Hat 4.8.5-16) (GCC) ) #1
    SMP Tue Aug 22 21:09:27 UTC 2017
14 Jan  9 09:43:19 exercise1 kernel: Command line:
    BOOT_IMAGE=/vmlinuz-3.10.0-693.el7.x86_64
    root=UUID=c14ecb64-e445-457e-a838-04314faa45bf ro
    crashkernel=auto rhgb quiet LANG=zh_CN.UTF-8
15 Jan  9 09:43:19 exercise1 kernel: Disabled fast string
    operations
16 Jan  9 09:43:19 exercise1 kernel: e820: BIOS-provided
    physical RAM map:
17 Jan  9 09:43:19 exercise1 kernel: BIOS-e820: [mem
    0x0000000000000000-0x0000000000009ebff] usable
18 Jan  9 09:43:19 exercise1 kernel: BIOS-e820: [mem
    0x0000000000009ec00-0x0000000000009ffff] reserved
19 [root@exercise1 opt]#
```

```
1 例子2:
2 [root@exercise1 opt]# head -n 3 /var/log/messages    #显
    示前 3 行
3 Jan  9 09:43:19 exercise1 journal: Runtime journal is
    using 6.1M (max allowed 48.8M, trying to leave 73.2M
    free of 482.0M available → current limit 48.8M).
4 Jan  9 09:43:19 exercise1 kernel: Initializing cgroup
    subsys cpuset
5 Jan  9 09:43:19 exercise1 kernel: Initializing cgroup
    subsys cpu
6 [root@exercise1 opt]#
```

例子3:

```
[root@exercise1 opt]# head -3 /var/log/messages    #显示前 3 行
```

例子4: 显示文件前2个字节

```
[root@exercise1 ~]# head -c2 /opt/c.txt
```

a:

```
[root@exercise1 ~]#
```

tail

- 1 作用：用于显示文件中的尾部内容。默认在屏幕上显示指定文件的末尾 10 行
- 2 语法：`tail` (选项) 文件名
- 3 参数：

-n 显示文件尾部多少行的内容(n 为数字) -f 动态显示数据 (不关闭),常用来查看日志

例子1:

默认查看

```
[root@exercise1 opt]# tail /var/log/messages
Jan  9 14:50:20 exercise1 systemd: Starting Network Manager Script
Dispatcher Service...
Jan  9 14:50:20 exercise1 dbus-daemon: dbus[511]: [system]
Activating via systemd: service name='org.freedesktop.nm_dispatcher'
unit='dbus-org.freedesktop.nm-dispatcher.service'
Jan  9 14:50:20 exercise1 dhclient[676]: bound to 192.168.119.141 --
renewal in 835 seconds.
Jan  9 14:50:20 exercise1 dbus[511]: [system] Successfully activated
service 'org.freedesktop.nm_dispatcher'
Jan  9 14:50:20 exercise1 dbus-daemon: dbus[511]: [system]
Successfully activated service 'org.freedesktop.nm_dispatcher'
Jan  9 14:50:20 exercise1 systemd: Started Network Manager Script
Dispatcher Service.
Jan  9 14:50:20 exercise1 nm-dispatcher: req:1 'dhcp4-change'
[ens33]: new request (3 scripts)
Jan  9 14:50:20 exercise1 nm-dispatcher: req:1 'dhcp4-change'
[ens33]: start running ordered scripts...
Jan  9 15:01:01 exercise1 systemd: Started Session 3 of user root.
Jan  9 15:01:01 exercise1 systemd: Starting Session 3 of user root.
```

例子2:

```
[root@exercise1 opt]# tail -n 3 /var/log/messages #查看最后 3 行记录
Jan  9 15:04:15 exercise1 systemd: Started Network Manager Script
```

Dispatcher Service.

```
Jan 9 15:04:15 exercise1 nm-dispatcher: req:1 'dhcp4-change'
[ens33]: new request (3 scripts)
Jan 9 15:04:15 exercise1 nm-dispatcher: req:1 'dhcp4-change'
[ens33]: start running ordered scripts...
[root@exercise1 opt]#
```

例子3:

tail -f /var/log/messages #动态显示

多开同机子远程终端一台使用命令tail -f /var/log/messages 另一台输入echo aaa >> /var/log/messages 在那台tail机子就会看到效果

history

```
1 | 历史命令
2 | 4 个快速查找 Linux 历史命令的技巧:
3 | 方法 1: 光标上下键
4 | 方法 2: ctrl+r -》输入某条命令的关键字-》找出来对应的命令, 按下
   | 光标键
5 | 方法 3: !数字 //执行历史命令中第 N 条命令
6 | history --查看命令历史
7 | history -c --清空命令历史
8 | history -w --把命令历史写入默认文件中, 并覆盖原文件
   | ($HOME/.bash_history ==> ~/.bash_history)
```

!!和!\$

```
1 | !! 代表上一个命令
2 | !$ 引用上一个命令的最后一个参数
```

!\$

例子:

###相当于执行:cat /opt/id补全命令使用tab键,Tab只能补全命令和文件

```
[root@exercise1 opt]# vi /opt/id
```

```
[root@exercise1 opt]# cat !$
```

```
cat /opt/id
```

```
jjsjssj
```

```
sjkdhkajsd
dhjkashdkas
asjhdgasjkdg
[root@exercise1 opt]#
```

！！例子：

```
[root@exercise1 opt]# ls
1 1d 1e 1f 2 3 4 5 i.bak id j log log.bak messages
[root@exercise1 opt]# !!
ls
1 1d 1e 1f 2 3 4 5 i.bak id j log log.bak messages
[root@exercise1 opt]#
```

Linux 下快捷键

- 1 都是用Ctrl+下面的单词， ^表示 Ctrl
- 2 ^C
- 3 终止前台运行的程序，如：ping g.cn 后，想停止按下Ctrl+C
- 4 ^D
- 5 退出 等价exit
- 6 ^L
- 7 清屏与 clear 功能一样
- 8 ^R
- 9 搜索历史命令，可以利用好关键词
- 10 ^K
- 11 删除当前光标到后面的所有内容
- 12 ^u
- 13 删除当前光标到前面的所有内容
- 14 home
- 15 快速回到行首
- 16 end
- 17 快速回到行尾
- 18 ^v ==> j ==> shift ==> i ==> # ==> esc
- 19 多行注释(vi/vim)
- 20 d ==> shift ==> G
- 21 全删、部删(vi/vim)
- 22 ^s

- 23 输入不显示，但其实已输入(冻结窗口)
- 24 ^q
- 25 取消冻结

系统时间管理

- 1 在 **Linux** 中有硬件时钟与系统时钟等两种时钟。
- 2 硬件时钟是指主机板上的时钟设备，也就是通常可在 **BIOS** 画面设定的时钟；
- 3 系统时钟则是指 **kernel**中的时钟；所有 **Linux** 相关指令与函数都是读取系统时钟的设定
- 4 当 **Linux** 启动时，系统时钟会去读取硬件时钟的设定，之后系统时钟再独立运作

查看硬件时间：

```
[root@exercise1 opt]# hwclock
```

2022年01月09日 星期日 15时25分53秒 -0.425155 秒

查看系统时间：

```
[root@exercise1 opt]# date
```

2022年 01月 09日 星期日 15:26:25 CST

UTC (Universal Time Coordinated) : 世界标准时间

GMT (Greenwich Mean Time) : 格林尼治时间

CST (China standard Time) : 中国标准时间

date

```
1 date 命令相关参数:
2 date --help ==> date帮助命令
3
4     -s, --set=STRING      把时间设为字符串所描述的时间
5
6 例子:
7 [root@exercise1 opt]# date -s "2023-12-30 14:15:00"
8 2023年 12月 30日 星期六 14:15:00 CST
```

%F 完整日期格式, 等价于 %Y-%m-%d

例子:

```
[root@exercise1 opt]# date "+%F"
```

2023-12-30

%y 年份最后两位数位 (00-99)

%Y 年份

%m month (01..12)

%d 按月计的日期(例如: 01)

%M minute (00..59)

%H 小时(00-23)

%S 秒(00-60)

例子:

```
[root@exercise1 opt]# date "+%Y%m%d"
```

20231230

```
[root@exercise1 opt]# date "+%Y-%m-%d %H:%M:%S" #在年月日之前可以添加自己想要的符号
```

2023-12-30 14:20:00

```
[root@exercise1 opt]# date "+%Y/%m/%d %H:%M:%S"
```

2023/12/30 14:21:06

```
[root@exercise1 opt]# date "+%Y%m%d %H:%M:%S"
```

20231230 14:21:19

-d, --date=STRING #显示由字符串描述的时间, 而不是“当前时间”

例子:

```
[root@exercise1 opt]# date
```

2023年 12月 30日 星期六 14:26:02 CST

```
[root@exercise1 opt]# date -d "+1 months" +%F
```

2024-01-30

```
[root@exercise1 opt]# date -d "+1 minutes" +%S
```

11

```
[root@exercise1 opt]# date -d "+1 days" +%F
```

2023-12-31

帮助命令使用

```
1 遇到命令不知道添加哪个参数，可以使用命令帮助查看相关介绍，常用的查看帮助信息命令有如下几个
2 man 命令 (manual) : 查看手册页或命令描述
3 [root@exercise1 opt]# man ls
4 man 命令查看帮助时，支持它支持上翻下翻，搜索(直接输入斜线)，退出用 q
5
6 查看命令拥有哪个级别的帮助
7 man -f 命令
8 1 用户命令
9 2 内核系统调用(从用户空间到进入点内核的)
10 3 库函数
11 4 特殊文件和设备
12 5 配置文件格式和规范
13 6 游戏
14 7 规范、标准和其他页面
15 8 管理员用的命令帮助手册
16 9 linux、内核API(内核调用)
17
18 安装中文版man
19 yum install man-pages-zh-CN
20 使用-h 或--help 查看命令选项
21 [root@exercise1 opt]# find -h #不可以执行
22 [root@exercise1 opt]# find --help #帮
```

常用的几个关机，重启命令

```
1 shutdown init reboot poweroff
2
```

```

3  关机命令之--shutdown
4  作用：关机，重启，定时关机
5  语法：shutdown    [选项]
6  参数：
7  -r      => 重新启动计算机
8  -h      => 关机
9  -h  时间  =>定时关机
10 -c      =>取消
11 -k  'xxxx' 可以指定提示文字
12
13 例如：
14 关机动作
15 [root@exercise1 opt]# shutdown -h +10                #10 分
   钟之后关机
16 [root@exercise1 opt]# shutdown -h 23:30              #指定具体
   的时间点进行关机
17 [root@exercise1 opt]# shutdown -h 15:05 -k '15:05关机'
   #指定具体的时间点进行关机，并有提示语
18 [root@exercise1 opt]# shutdown -h now                #立即关
   机
19 [root@exercise1 opt]# poweroff                        #
   关机
20 [root@exercise1 opt]# systemctl poweroff              #
   关机
21
22 重启动作
23 [root@exercise1 opt]# shutdown -r 22: 22             #22:22
   以后重启
24 [root@exercise1 opt]# reboot
   #立即重启
25 [root@exercise1 opt]# poweroff --reboot               #
   立即重启

```

centos 下 Linux 运行init 级别 0-6 的各自含义

0： 关机模式

1： 单用户模式 ， 用于破解 root 密码（救援模式）

2： 无网络， 支持的多用户模式

3： 有网络支持的多用户模式（一般叫字符界面，工作中最长使用的模式）

- 4：保留，未使用
- 5：有网络支持，支持图形界面，支持的多用户模式（图形界面）
- 6：重新引导系统，及重启

总结

- 1 如果在创建、移动、复制、删除的时候不指定路径就默认是当前路径下操作

服务命令

服务	centOS6	centOS7
启动	service name start	systemctl start name.service
停止	service name stop	systemctl stop name.service
重启	service name restart	systemctl restart name.service
状态	service name status	systemctl status name.service
重载或重启服务(先加载，再启动)	-	systemctl reload-or-restart name.service

根目录下的文件夹

文件夹绝对路径	英文全称	文件夹作用
/bin	Binaries	存放系统命令的目录，所有用户都可以执行。
/sbin	Super User Binaries	保存和系统环境设置相关的命令，只有超级用户可以使用这些命令，有些命令可以允许普通用户查看
/usr	Unix Shared Resources	Unix共享资源目录，存放所有命令、库、手册页等
/usr/bin	Unix Shared Resources Binaries	存放系统命令的目录，所有用户可以执行。这些命令和系统启动无关，单用户模式下不能执行
/usr/sbin	Superuser Binaries	存放根文件系统不必要的系统管理命令，超级用户可执行
/dev	Devices	存放设备文件
/etc	Editable Text Configuration Chest	存放配置文件的地方,配置文件的目录
/opt	Optional Application Software Packages	可选应用软件包，第三方安装的软件保存位置
/lib	Library	存放系统程序运行所需的共享库
/proc	Processes	虚拟文件系统，数据保存在内存中，存放当前进程信息
/root	Root	存放root用户的相关文件,root用户的家目录。宿主目录 超级用户
/tmp	Temporary	存放临时文件
/var	Variable	是储存各种变化的文件，比如log等等

文件夹绝对路径	英文全称	文件夹作用
/home	Home	普通用户主目录
/lost+found	Lost And Found	存放一些系统出错的检查结果 (centos6中出现)
/srv	Server	服务数据目录
/mnt	Mount	挂载目录。临时文件系统的安装点，默认挂载光驱和软驱的目录
/media	Media	挂载目录。 挂载媒体设备，如软盘和光盘
/run	Run	里面的东西是系统运行时需要的, 不能随便删除. 但是重启的时候应该抛弃. 下次系统运行时重新生成