

linux常用命令总结

1.重启和关机

重启和关机需要系统管理员用户权限。

重启

```
init 6或者 reboot
```

关机

```
init 0或者 shutdown 或者 halt
```

如果没有执行关机命令，强制断电或关闭本地虚拟机的窗口，会导致linux操作系统文件的损坏，严重的可能导致操作系统无法正常启动。

2.清屏

清除当前屏幕上显示的内容。

```
clear
```

3.查看服务器的地址

```
ip addr (ifconfig也可以进行查看)
```

```
ll@ubuntu:~$ ip addr
: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
  inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 00:0c:29:49:5f:f3 brd ff:ff:ff:ff:ff:ff
  alt name enp2s1
  inet 192.168.234.128/24 brd 192.168.234.255 scope global dynamic noprefixroute ens33
    valid_lft 1501sec preferred_lft 1581sec
  inet6 fe80::948a:cd33:5e77:cd23/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
  link/ether 02:42:ab:7a:fd:38 brd ff:ff:ff:ff:ff:ff
  inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
    valid_lft forever preferred_lft forever
  inet6 fe80::42:abff:fe7a:fd38/64 scope link
    valid_lft forever preferred_lft forever
ll@ubuntu:~$
```

4.事件操作

当前时间

```
date
```

```
l1l@ubuntu:~$ date
Sun 28 Aug 2022 09:28:01 AM CST
l1l@ubuntu:~$
```

设置时区为中国上海时间

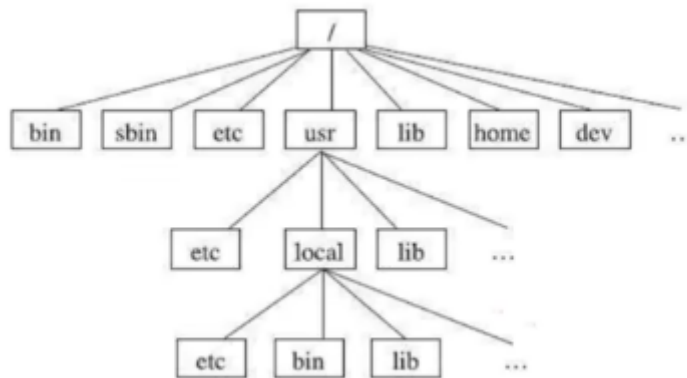
```
cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

设置时间

```
date -s "yyyy-mm-dd hh:ii:ss"
#例如: date -s "2022-8-28 09:30:30"
```

5. 目录和文件

文件系统就像一棵树，树干是/（根）目录，树枝是子目录，树枝后面还有树枝（子目录中还有子目录），树枝最好是树叶，目录的最后是文件。



严谨的说，文件名是由**目录+文件名**组成的。

- 从根目录开始，包含完整的目录名和文件名的是绝对路径。
- 登录linux后，一定处在目录树的某个目录中，这个目录称之为当前工作目录，简称**当前目录**。
- 文件的**相对路径**就是相对于当前目录所在的路径。
- 一个圆点.表示当前工作目录。
- 两个圆点..表示当前工作目录的上一级目录。

查看当前目录

```
pwd
```

切换目录

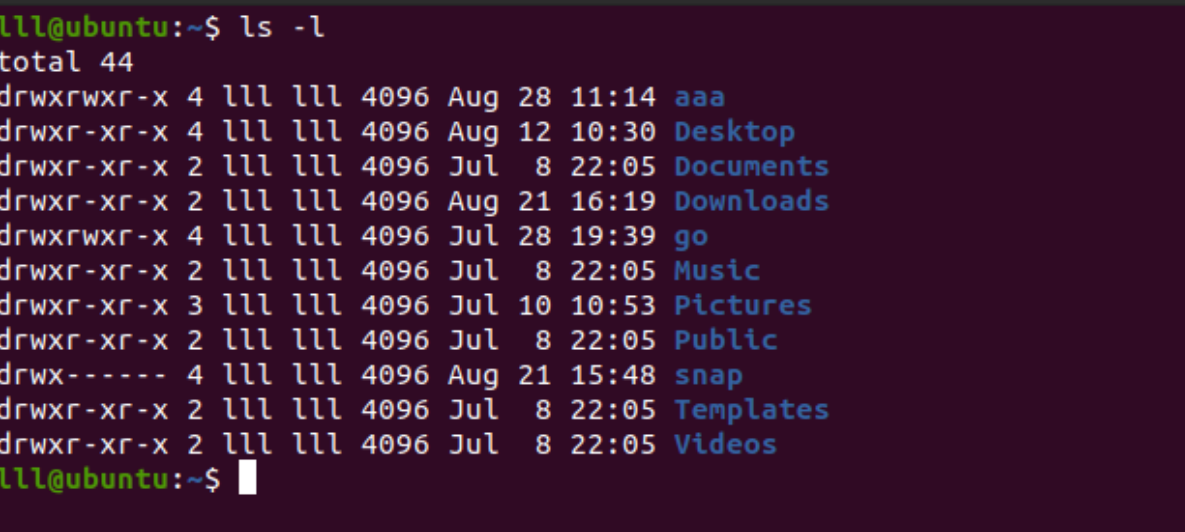
```
cd 目录
#例如：
进入/etc目录：cd /etc
切换到上一级目录：cd ..
```

列出目录和文件信息

```
ls

#列出目录的详细信息：ls -l

#列出/etc目录下的内容：ls /etc
```



文件的权限

number	permission type	symbol
0	no permission	---
1	excute	--X
2	write	-W-
3	excute(1)+write(2)	-WX
4	read	r--
5	read(4)+execute(1)	r-X
6	read(4)+write(2)	rw-
7	read(4)+write(2)+excute(1)	rwX

对于通过ls -l查到的权限如下：

```
drwxrwxr-x 4 lll lll 4096 Aug 28 11:14 aaa
```

#以drwxrwxr-x为例：

第一个位置的值表示当前是文件还是目录，-表示一个文件，d表示为一个目录

接着后三个位置的值表示的是创建者对该文件的权限，rwx即可读可写可执行

之后接着三个位置的值表示的是组对该文件的权限，rwx即可读可写可执行

最后的三个未知的值表示的是其他对该文件的权限，r-x即可读可执行不可写

修改文件的权限

方式一：chmod permission filename

```
chmod ugo filename
```

u:user 对应于user的权限

g:group 对应于用户所在group的权限

o:other 对应于other的权限

例如：对于文件a.txt，让user可读可写写可执行，让group和other都没有权限

```
chmod 700 a.txt
```

方式二：通过符号修改

operator	description
+	增加一个权限
-	删除一个权限
=	修改一个权限

```
#例如，对于a.txt文件
```

#删除user在它上面的写权限

```
chmod u-w a.txt
```

#增加other在它上面的执行和写权限

```
chmod o+wx a.txt
```

#修改user的权限为可读可写可执行

```
chmod u=rwx a.txt
```

#修改所有的权限为可读可写可执行

```
chmod a=rwx a.txt
```

正则表达式

正则表达式又称规则表达式，通配符，目录和文件名都支持正则表达式，正则表达式的规则比较多，比较常用的两种是：星号“*”和问号“?”。

星号“*”：匹配任意数量的字符。

问号“?”匹配任意的一个字符。

```
#列出/tmp目录下以systemd开头的文件或目录
ls /tmp/systemd*
```

创建目录

```
mkdir 目录名

#创建一个aaa目录：mkdir aaa
```

创建文件

```
touch 文件名

当前路径下创建一个a.txt文件：touch a.txt
绝对路径下创建一个a.txt文件：touch /home/a.txt
```

删除文件和目录

```
rm 文件名
参数：
-f 强制删除
-r 删除目录

#删除/tmp目录下的所有内容
rm -rf /tmp

#同时也可以结合正则表达式一起使用
删除/tmp目录下以.tar结尾的文件：rm -rf /tmp/*.tar
```

移动文件和目录

```
mv 旧目录或文件名 新目录或文件名

如果第二个参数是已经存在的目录，则把第一个参数（旧目录或文件）移动到该目录中。
```

1. 使用mv实现文件重命名的效果, 将当前目录下的a.txt文件重命名为b.txt

```
mv a.txt b.txt
```

2. 如果bbb目录存在，以下命令将把当前工作目录下的a.txt文件移动到bbb目录中

```
mv a.txt bbb
```

3. bbb/cc目录不存在，以下命令将把当前工作目录下的a.txt文件改名为/bbb/cc

```
mv a.txt bbb/cc
```

复制文件和目录

`cp [-r]` 旧目录或者文件名 新目录或者文件名
选项-r是复制目录，如果没有选项-r就只复制文件。

1. 复制文件a.txt的内容到b.txt（没有b.txt文件就会新建）

```
cp a.txt b.txt
```

2. 复制目录aaa到bbb/cc

```
cp -r aaa bbb/cc  
#如果bbb/cc目录不存在，就会把将aaa目录下的所有东西复制到bbb/aaa目录下  
  
#如果bbb/cc目录存在，就会复制到bbb/cc/aaa下
```

6.打包和压缩

打包

`tar -zcvf` 压缩包名 需要打包的文件1/目录1 需要打包的文件2/目录2 ...需要打包的文件n/目录n

参数含义：

- z：通过gzip的支持进行压缩/解压缩，此时文件最好为*.tar.gz
- c：创建压缩文件
- v：显示细节
- f：要操作的文件名

#例如：将目录aaa和bbb以及文件a.txt都压缩到test.tar.gz压缩包中

```
tar -zcvf test.tar.gz aaa bbb a.txt
```

解压

```
tar -zxvf 压缩包名 解压到的目录
```

参数含义：

-z：通过gzip的支持进行压缩/解压，此时文件最好为*.tag.gz

-x：解压缩

-v：显示细节

-f：要操作的文件名

#例如：将test.tar.gz解压到bbb目录下

```
tar -zxvf test.tar.gz bbb
```

7.判断网络是否连通

```
ping ip地址/域名
```

例如：检查是否可以连接上百度

```
ping www.baidu.com
```

8.显示文本文件的内容

显示整个文件的内容

```
cat 文件名1 文件名2 ... 文件名n
```

分页显示文件的内容

```
more 文件名1 文件名2 ... 文件名n
```

显示文件尾部的内容

tail 命令可用于查看文件的内容，有一个常用的参数 -f 常用于查阅正在改变的日志文件。

tail -f filename 会把 filename 文件里的最尾部的内容显示在屏幕上，并且不断刷新，只要 filename 更新就可以看到最新的文件内容。

```
tail 文件名
```

9.输出命令

```
echo "内容" | $变量 [> 输出文件路径]
```

#直接在终端输出内容：

```
echo "hello"
```

#在终端中输出变量的值：

```
echo $path
```

#将内容输出到a.txt文件

```
echo "hello" > a.txt
```

10.搜索文件中的内容

```
grep [-c] "内容" 文件名
```

#如果内容中没有空格等特殊字符，可以不用双引号括起来

#例如：在views.py文件中搜索request

```
grep "request" views.py
```

#如果使用-c则显示的是行数

```
l1l1@ubuntu:~/Desktop/python/blog/MyBlog/app$ grep "request" views.py
def isLoginStatus(request):
    user_id = request.get_signed_cookie('user_id', salt=L1L1) # 从cookie中获取用户的id
def login(request):
    if request.method == 'GET':
        return render(request, 'login.html')
    if request.method == 'POST':
        account = request.POST.get('account')
        password = request.POST.get('password')
        response = render(request, 'login.html', context=locals())
def register(request):
    if request.method == 'GET':
        return render(request, 'register.html')
    if request.method == 'POST':
        telValue = request.POST.get('tel') # 用户的手机号码
        yzmValue = request.POST.get('yzm') # 验证码
        pwdValue = request.POST.get('pwd') # 密码
        forgetPwd = request.POST.get('forgetPwd')
def sendCode(request):
    if request.method == 'POST':
        telValue = request.POST.get('tel') # 用户的手机号码
        forgetPwd = request.POST.get('forgetPwd')
def error(request):
    return render(request, '404.html')
def index(request):
    if request.method == 'GET':
        LoginStatus, user_id = isLoginStatus(request) # 如果是登录状态则显示个人中心
        page_num = request.GET.get('page', '1') # 获取页码，如果没有获取到则默认为1
        return render(request, 'index.html', context=context)
def personalCenter(request, user_id):
    LoginStatus, user_id = isLoginStatus(request) # 如果是登录状态则显示个人中心
    if request.method == 'GET':
        return render(request, 'personalCenter.html', context=locals())
def returnCollectList(request):
    LoginStatus, user_id = isLoginStatus(request) # 如果是登录状态则显示个人中心
    if request.method == 'POST':
        nowId = request.POST.get('nowId')
def logout(request, user_id):
def modifyAvatar(request):
    LoginStatus, user_id = isLoginStatus(request) # 如果是登录状态则显示个人中心
    if request.method == 'POST':
        user_id = request.POST.get('userId')
        imgData = request.POST.get('imgData')
def modifyDesc(request):
    LoginStatus, user_id = isLoginStatus(request) # 如果是登录状态则显示个人中心
    if request.method == 'POST':
```

11.统计文本文件的行数，单词数和字节数

WC 文件名

```
990      2631      38369 total
lll@ubuntu:~/Desktop/python/blog/MyBlog/app$ wc *.py
 58      185      2330 admin.py
  6         12       106 apps.py
 14        26       397 forms.py
  0         0         0 __init__.py
149      342     5287 models.py
 17        42       432 tests.py
 36       139     2185 urls.py
710     1885    7632 views.py
990     2631    38369 total
lll@ubuntu:~/Desktop/python/blog/MyBlog/app$
```

文件名

行数 单词数 字节数 (文件大小)

12.查看系统磁盘空间使用情况

df [-h] [-T]

参数含义：

-h：以方便阅读的方式显示

-T：列出文件系统类型

```
lll@ubuntu:~/Desktop/python/blog/MyBlog/app$ df -h -T
Filesystem      Type      Size  Used Avail Use% Mounted on
udev            devtmpfs  2.7G   0    2.7G   0% /dev
tmpfs           tmpfs     556M   2.0M 554M   1% /run
/dev/sda5       ext4      49G   25G   22G   54% /
tmpfs           tmpfs     2.8G  341M  2.4G  13% /dev/shm
tmpfs           tmpfs     5.0M   4.0K 5.0M   1% /run/lock
tmpfs           tmpfs     2.8G   0    2.8G   0% /sys/fs/cgroup
/dev/loop0      squashfs  128K  128K   0 100% /snap/bare/5
/dev/loop1      squashfs   56M   56M   0 100% /snap/core18/2409
/dev/loop2      squashfs   56M   56M   0 100% /snap/core18/2538
/dev/loop3      squashfs   62M   62M   0 100% /snap/core20/1587
/dev/loop5      squashfs  219M  219M   0 100% /snap/gnome-3-34-1804/72
/dev/loop7      squashfs   92M   92M   0 100% /snap/gtk-common-themes/1535
/dev/loop6      squashfs   66M   66M   0 100% /snap/gtk-common-themes/1515
/dev/loop4      squashfs   62M   62M   0 100% /snap/core20/1611
/dev/loop8      squashfs  401M  401M   0 100% /snap/gnome-3-38-2004/112
/dev/loop9      squashfs  219M  219M   0 100% /snap/gnome-3-34-1804/77
/dev/loop11     squashfs   47M   47M   0 100% /snap/snapd/16292
/dev/loop10     squashfs   51M   51M   0 100% /snap/snap-store/547
/dev/loop12     squashfs   55M   55M   0 100% /snap/snap-store/558
/dev/loop13     squashfs   98M   98M   0 100% /snap/typora/57
/dev/sda1       vfat      511M   4.0K 511M   1% /boot/efi
tmpfs           tmpfs     556M   36K 556M   1% /run/user/1000
lll@ubuntu:~/Desktop/python/blog/MyBlog/app$
```

