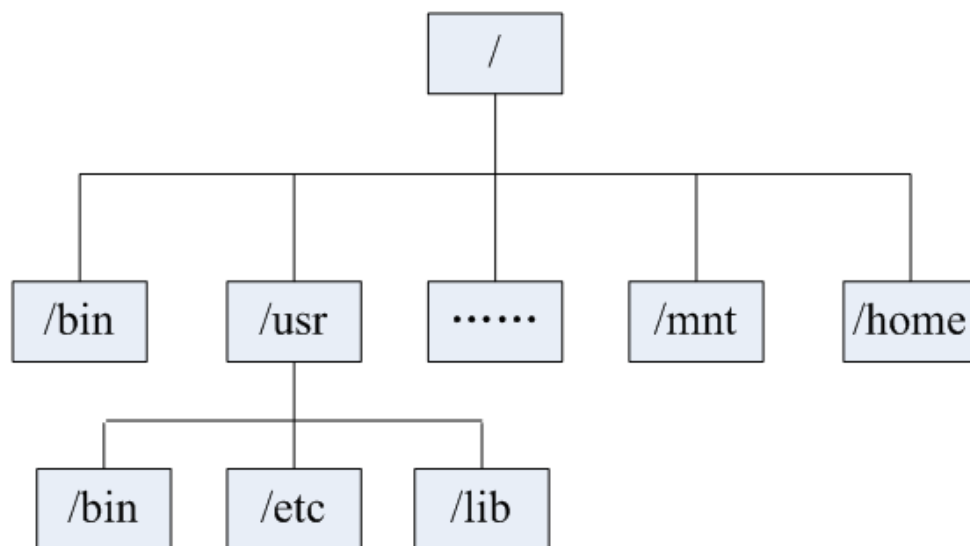


04 文件与目录管理

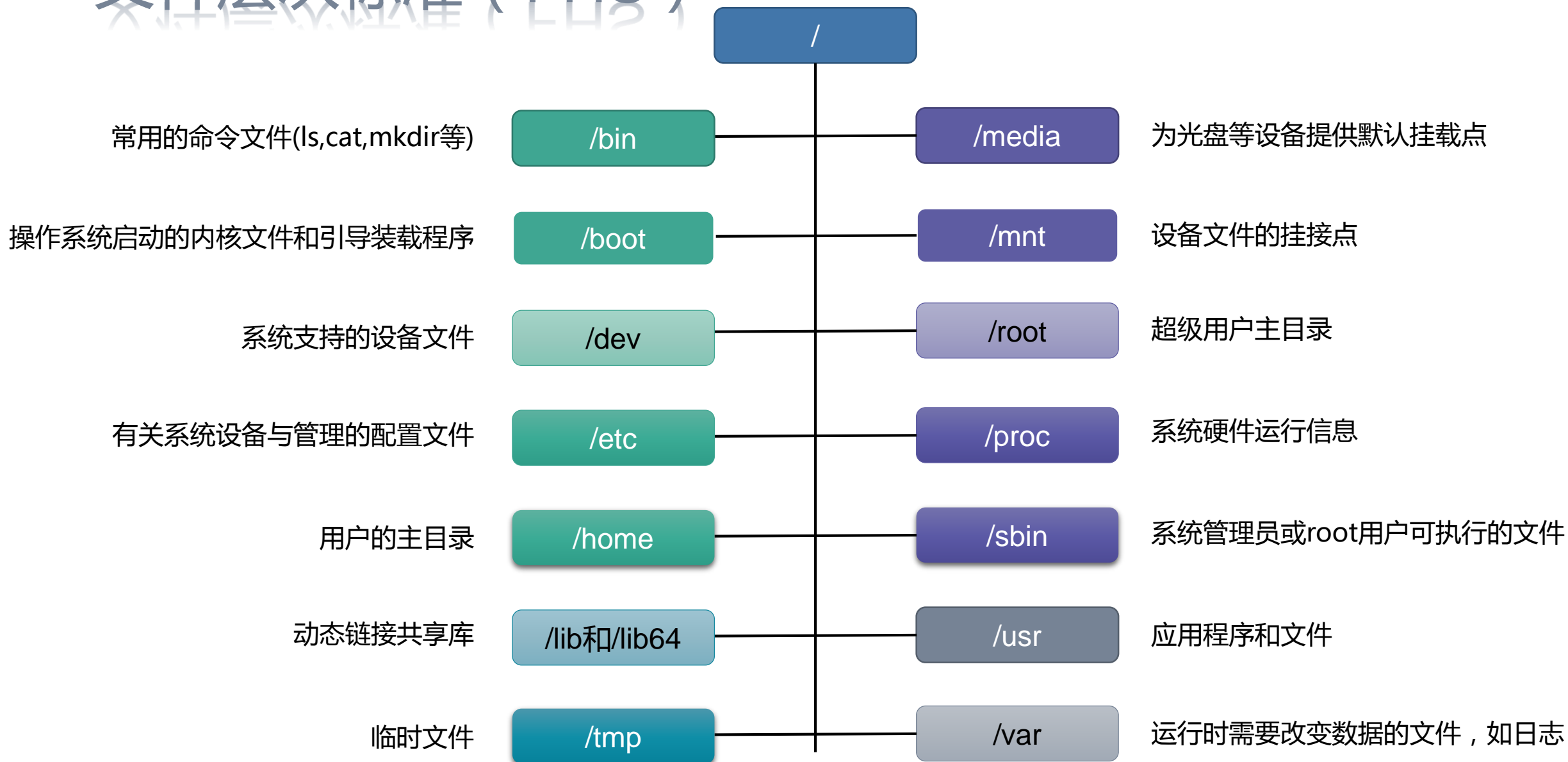
LINUX目录结构

与Windows相类似，Linux也采用了树状结构的文件系统，它由目录和目录下的文件一起构成。但Linux文件系统不使用盘符这个概念，而是使用单一的根目录结构，所有的分区都挂载到单一的“/”目录上。“/”目录也称为根目录，位于Linux文件系统目录结构的顶层。



文件系统层次标准（FHS）规范在根目录（/）下面各个主要目录以及子目录（/usr，/var）应该放置的文件。

文件层次标准 (FHS)



路径

- 路径：从树形中的某个目录层次到某个文件的一条路线。此路径的主要构成是目录名称，中间用/分开。
- 用户在对文件进行访问时，要给出文件所在的路径。
- 绝对路径：从/开始的路径，又称完全路径。
- 相对路径：从用户工作的目录开始的路径，有4种符号表示方法
 - . 代表当前目录 (current directory)
 - .. 代表上一层目录
 - ~ 代表用户主目录
 - 代表上一个使用目录
- 应注意到的是，在树型目录结构中某一确定文件的绝对路径只有一条。绝对路径是确定不变的，而相对路径则随着用户工作目录的变化而变化。

LINUX文件结构

硬盘格式化的时候，操作系统将硬盘分成两个区域。一个是数据区，存放文件数据；还有一个是索引节点区（inode table），存放索引节点所包括的信息。

Linux中任何文件都由两部分构成：数据和索引节点。

- 数据：文件的实际内容，有自己的结构。
- 索引节点存储文件的元信息。

元信息包括文件的创建者、文件的创建日期、文件的大小、文件的权限等等。

Linux文件系统以块为单位存储信息，为了找到某一个文件所在存储空间的位置，用索引节点对每个文件进行索引。

INODE索引节点内容

inode包括文件的元信息：

- * 文件的字节数
- * 文件拥有者的UID
- * 文件的GID
- * 文件的读、写、运行权限
- * 文件的时间戳：Modify记录文件内容修改时间（元数据）、Change 记录文件属性修改时间、

Access 读取文件时间

- * 链接数，即有多少文件名称指向这个inode
- * 文件数据block的大小

STAT命令

查看文件inode命令：stat filename

```
lei@lei-VirtualBox:~$ stat code.c
  File: code.c
  Size: 74          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d Inode: 313119       Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/   lei)   Gid: ( 1000/   lei)
Access: 2019-10-13 09:08:51.654805045 -0400
Modify: 2019-10-13 10:27:06.255982892 -0400
Change: 2019-10-13 10:27:06.259982892 -0400
 Birth: -
```

访问时间(Access) : atime	查看内容，例cat a.txt
修改时间(Modify) : mtime	修改内容，例vim a.txt
改变时间(Change) : ctime	修改文件属性，例如chmod +x a.sh

文件类型

Linux系统的文件类型有：

- 普通文件
- 目录文件
- 设备文件
- 链接文件

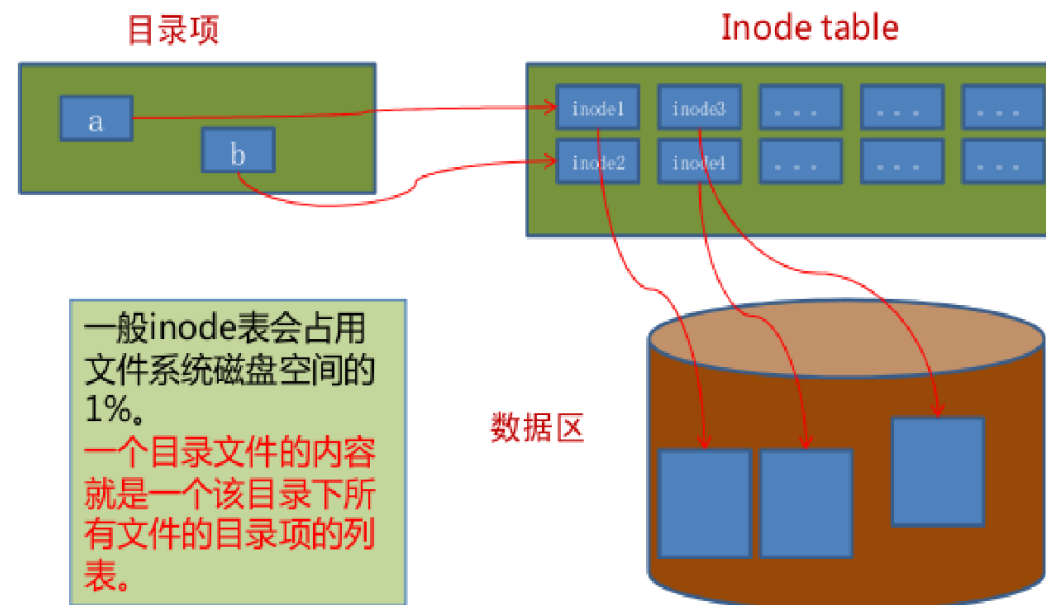
普通文件

普通文件也成为常规文件，包含各种长度的字符串。这些字符串不会被内核结构化，只是作为有序的字节序列提交给应用程序。应用程序自己组织和解释这些数据，普通文件包括的类型有：

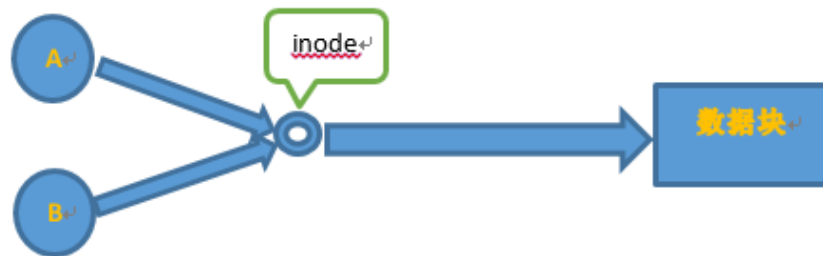
- 文本文件，由ASCII字符组成。
- 数据文件，由数字型和文本型数据构成。
- 二进制文件，经过编译的，计算机可以直接执行的机器代码文件。

目录文件

目录文件是一类特殊的文件，主要用来组织和管理文件和其它目录，利用它可以构成文件系统的分层树型结构。目录文件包含的是结构化的数据，是由成对的“索引节点号/文件名”构成的列表。通过文件名进行操作时，系统会通过对应路径目录的索引节点号找到目录的索引节点表数据，进而通过数据指针获取目录的数据，匹配文件名，得到文件的索引节点号；再到索引节点表中找到此文件的表项，由文件的数据指针获取真正的文件数据。



链接文件



链接文件分为符号链接和硬链接两种。

在Linux中，允许多个文件名指向同一个索引节点。这意味着可以使用多个不同的文件名访问同一个文件，同时对文件内容进行修改，会影响到所有的文件名，但是，删除其中的一个文件名，不会对其它文件名产生影响，其它文件名可以正常访问文件。这种情况就被成为硬链接。

创建硬链接的方法： `ln filename [linkname]`

```
lei@lei-VirtualBox:~$ ln code.c code.c.link
lei@lei-VirtualBox:~$ stat code.c
  File: code.c
  Size: 74          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d Inode: 313119       Links: 2
Access: (0664/-rw-rw-r--) Uid: ( 1000/ lei)  Gid: ( 1000/ lei)
Access: 2019-10-13 09:08:51.654805045 -0400
Modify: 2019-10-13 10:27:06.255982892 -0400
Change: 2019-10-13 11:01:09.543982892 -0400
 Birth: -
lei@lei-VirtualBox:~$ stat code.c.link
  File: code.c.link
  Size: 74          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d Inode: 313119       Links: 2
Access: (0664/-rw-rw-r--) Uid: ( 1000/ lei)  Gid: ( 1000/ lei)
Access: 2019-10-13 09:08:51.654805045 -0400
Modify: 2019-10-13 10:27:06.255982892 -0400
Change: 2019-10-13 11:01:09.543982892 -0400
 Birth: -
```

链接文件

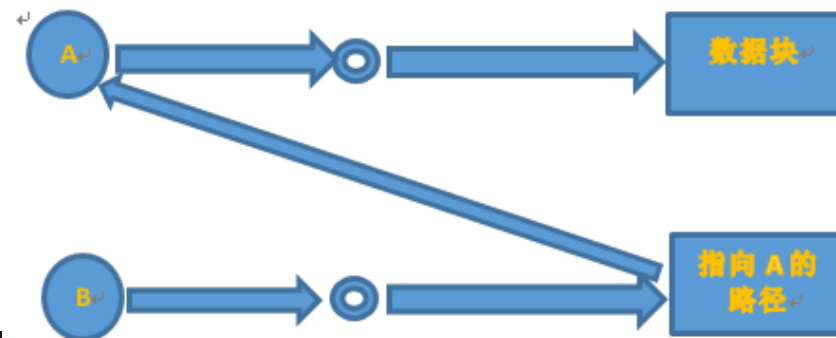
软链接也叫符号链接，它相当于windows中的快捷方式。当我们需要在不同目录用到相同的文件时，我们不需要在每个目录下都放置一个相同的文件，而只需要在某个固定的目录下放置该文件，然后创建软链接，软链接指向这个文件，软链接的数据是它指向的这个文件的路径，从而避免了重复占用磁盘空间。

创建软链接的方法：`ln -s filename [linkname]`

软链接有自己的 inode 号以及用户数据块。

软链接可以对目录进行链接，硬链接中只能对文件进行链接。

```
lei@lei-VirtualBox:~$ stat code.c.sl
  File: code.c.sl -> code.c
  Size: 6          Blocks: 0          IO Block: 4096   symbolic link
Device: 801h/2049d Inode: 264453       Links: 1
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/   lei)   Gid: ( 1000/   lei)
Access: 2019-10-13 11:04:48.975982892 -0400
Modify: 2019-10-13 11:04:48.975982892 -0400
Change: 2019-10-13 11:04:48.975982892 -0400
 Birth: -
```



设备文件

在Linux系统中，所有设备都作为一类特别文件对待，用户像使用普通文件那样对设备进行操作。但是设备文件除了存放在文件索引节点中的信息外，它们不包含任何数据。系统利用他们来标识各个设备驱动器，内核使用它们与硬件设备通信。通常分为两类：字符设备、块设备。

Linux 将设备文件放在/dev目录下，系统中每个设备在该目录下有一个对应的设备文件。

例如，

- 串口COM1的文件名为/dev/ttyS0，
- 第一个SCSI硬盘文件名为/dev/sda，
- /dev/sda5对应第一个SCSI硬盘第一个逻辑分区，
- /dev/cdrom表示光驱。

查看文件类型

使用 `ls -l` 命令以长格式列出目录时，每一行第一个字符代表文件类型。

- - 表示普通文件
- d 表示目录文件
- c 表示字符设备文件
- b 表示块设备文件
- l 表示符号链接文件

```
lei@lei-VirtualBox:~$ ls -l
total 84
-rw-r--r-- 1 lei lei 0 Oct 3 08:33 abc
-rwxr-xr-x 1 lei lei 8304 Oct 5 00:17 a.out
-rw-rw-r-- 2 lei lei 74 Oct 13 10:27 code.c
-rw-rw-r-- 1 lei lei 53 Oct 5 00:11 code.c~
-rw-rw-r-- 2 lei lei 74 Oct 13 10:27 code.c.link
lrwxrwxrwx 1 lei lei 6 Oct 13 11:04 code.c.sl -> code.c
drwxr-xr-x 2 lei lei 4096 Sep 22 07:48 Desktop
drwxr-xr-x 2 lei lei 4096 Sep 22 07:48 Documents
drwxr-xr-x 2 lei lei 4096 Sep 22 07:48 Downloads
```

```
lei@lei-VirtualBox:/dev$ ls -l
total 0
crw-r--r-- 1 root root 10, 235 Oct 13 08:33 autofs
drwxr-xr-x 2 root root 440 Oct 13 08:33 block
drwxr-xr-x 2 root root 100 Oct 13 08:33 bsg
crw----- 1 root root 10, 234 Oct 13 08:33 btrfs-control
drwxr-xr-x 3 root root 60 Oct 13 08:33 bus
lrwxrwxrwx 1 root root 3 Oct 13 08:33 cdrom -> sr0
drwxr-xr-x 2 root root 3720 Oct 13 08:33 char
crw----- 1 root root 5, 1 Oct 13 08:34 console
lrwxrwxrwx 1 root root 11 Oct 13 08:33 core -> /proc/kcore
```

查看文件类型

file命令可以查看文件的具体类型

```
lei@lei-VirtualBox:~$ file Pictures/  
Pictures/: directory  
lei@lei-VirtualBox:~$ file code.c  
code.c: C source, ASCII text  
lei@lei-VirtualBox:~$ file code.c.sl  
code.c.sl: symbolic link to code.c  
lei@lei-VirtualBox:~$ file hello.sh  
hello.sh: Bourne-Again shell script, ASCII text executable
```

目录操作命令

- 创建目录：mkdir
- 删除目录：rmdir
- 改变工作目录：cd
- 显示当前目录：pwd
- 显示目录内容：ls

MKDIR / RMDIR 命令

mkdir / rmdir: 创建和删除目录

用法: `mkdir` 目录名
 `rmdir` 目录名

◆ 举例:

```
mkdir data
```

在当前目录下创建一个名为 `data` 的目录。

```
rmdir olddata
```

删除当前目录下的 `olddata` 目录，若该目录非空，则报错

`rmdir` 只能删除空目录，非空目录可用 `rm -r` 删除

CD / PWD 命令

cd: 改变工作目录

用法: `cd 目录`

- ◆ 进入指定的目录，即将工作目录改为指定的目录

pwd: 显示当前目录

用法: `pwd`

- ◆ 告诉用户当前的工作目录，即当前在哪个目录下

LS 命令

ls: 列出指定目录内容

◆ 常用选项:

- i 在输出的第一列显示文件的索引节点号
- a 列出目录下的所有文件，包括以 . 开头的隐含文件。
- l 列出文件的详细信息，通常称为“长格式”。
- R 递归地显示目录的各个子目录中的文件。

```
lei@lei-VirtualBox:~$ ls -li
266393 abc                264453 code.c.sl        313103 hello.sh         395620 regexp
315640 a.out              526395 Desktop          288258 hello.sh~         526398 Templates
313119 code.c             526400 Documents          526401 Music           288836 test.sh
313083 code.c~            526397 Downloads        135199 Pictures        135200 Videos
313119 code.c.link       270439 examples.desktop  526399 Public
```

文件操作命令

CAT命令

cat: 连接文件并打印到标准输出, 查看文件内容

用法: `cat` [选项] 文件列表

- ◆ 功能之一 是用来显示文件内容。它依次读取指定文件的内容并在标准输出中输出。
- ◆ 功能之二 是用来将两个或多个文件连接起来。
- ◆ 常用选项:

-n 在文件的每行前面显示行号。

CAT命令

```
cat -n file1
```

将文件 file1 的内容在屏幕上输出，并显示行号。

```
cat file1 file2 > file3
```

将文件 file1 和文件 file2 的内容合并起来，放入文件 file3 中，此时在屏幕上并不能直接看到该命令执行后的结果。若想看到连接后的文件内容，可以输入命令 `cat file3`

注：当文件内容过长时，就带来一个问题，因为文本在屏幕上迅速地闪过，用户来不及看清其内容。因此，当文件内容较长时，一般可用 `more` 或 `less` 等命令分屏显示。

MORE / LESS命令

more / less: 查看文件内容

用法: more 文件名
less 文件名

- 该命令一次显示一屏文本，显示满之后，停下来，并在终端底部打印出 “-- More --”，若要继续显示，按回车（显示下一行）或空格键（显示下一屏）即可。若要退出，按 **q**
- 显示文件内容时，可以向前翻，也可以向后翻。向前翻时，除了用回车和空格外，还可以用字母 **f**；翻看后面的内容，可以用字母 **b**。

HEAD 命令

head: 在屏幕上显示文件的开头若干行或若干字节

用法: `head` [选项] 文件名

◆ 常用选项:

-n(n为行数值) 指定显示文件开头的行数（默认10行）

```
head -5 /etc/passwd
```

-c 后跟参数指定显示文件开头的字节数, 可以使用单位**b**为512字节, **kB**1000字节, **K**为1024字节。

TAIL 命令

`tail`: 在屏幕上显示文件的末尾若干行或若干字节

用法: `tail` [选项] 文件名

◆ 常用选项:

-n(n为行数值) 指定显示文件末尾的行数（默认10行）

```
tail -5 /etc/passwd
```

-c 后跟参数指定显示文件末尾的字节数, 可以使用单位**b**为512字节, **kB**1000字节, **K**为1024字节。

文件内容查找

`grep` 用来在文本文件中查找指定模式的单词或短语，并在标准输出上显示包括给定字符串模式的所有行。

用法: `grep` [选项] 模式 文件名

◆ 常用选项:

- i : 表示忽略大小写
- x : 强制整行匹配
- w : 强制关键字完全匹配
- e : 用于定义正则表达式
- n : 指定输出的同时打印行号
- H : 为每一匹配项打印文件名
- r : 在指定目录中递归查询
- v : 反转查找

```
grep -i 'home' /etc/passwd
```

文件内容比较

comm对已经排序的文件进行逐行比较。

用法: comm [-123] 文件1 文件2

◆ 选项:

- 1 : 不显示仅在文件1存在的行
- 2 : 不显示仅在文件2存在的行
- 3 : 不显示两个文件都存在的行

文件内容比较

`diff` 逐行比较两个文件，列出它们的不同之处，并且提示为使两个文件一致需要修改哪些行。

用法: `diff` [选项] 文件1 文件2

不要求事先对文件进行排序。

```
lei@lei-VirtualBox:~$ cat code.c
#include <stdio.h>

int main()
{
    printf("first c code");
    return 1;
}

lei@lei-VirtualBox:~$ cat code.c.bk
#include <stdio.h>

int main()
{
    printf("first c code");
    return 0;
}
```

```
lei@lei-VirtualBox:~$ diff code.c code.c.bk
6c6
<  return 1;
---
>  return 0;
```

文件内容排序

`sort`用于对文本文件的各行进行排序。

用法: `sort` [选项] 文件列表

◆ 选项:

- u 对排序后认为相同的行只保留其中一行。
- r 按逆序输出排序结果

逐行对文本文件中的所有行进行排序，并将结果显示在标准输出上。

UNIQ命令

uniq命令可以将文件内的重复行数据从输出文件中删除，只留下每条记录的唯一样本。

用法: `uniq` [选项] 文件列表

◆ 选项:

- d 只显示重复行
- u 只显示不重复的行

文件内容统计

wc用于统计指定文件的字节数，字数，行数，并输出结果。

用法：wc [选项] 文件列表

输出格式：

行数 字数 字节数 文件名

◆ 常用选项：

- c：表示统计字节数
- l：表示统计行数
- w：表示统计字数

文件查找

`find`用于在目录中搜索满足查询条件的文件并执行指定操作。

用法: `find 路径 匹配表达式 -exec 外部命令 {} \;`

◆ 常用选项:

-name : 按文件名查找

-user : 按文件所有者查找

-type : 按文件类型查找 (b是块设备文件, d为目录, c为字符设备文件, l为链接文件, f为普通文件)

查找主目录下后缀为txt的文件

```
find ~ -name "*.txt"
```

首先查找所有文件名为passwd*的文件, 然后执行grep查看这些文件中是否有一个名为“wang”的用户

```
sudo find /etc -name "passwd*" -exec grep "wang" {} \;
```


文件查找

`locate`用于查找文件，比`find`搜索速度快，但需要一个数据库，数据库每天由例行工作程序自动建立和维护。

用法： `locate` [选项] [模式]

◆ 常用选项：

- d：指定所使用的数据库，以代替默认的数据库：`/var/lib/mlocate/mlocate.db`。
- c：只列出查到的条目数量，
- A：列出匹配的所有条目，
- w：匹配整个路径，

`locate chgpasswd`

WHICH 命令

which: 查找指定命令所在的位置

用法: `which command`

◆ 显示指定命令的全路径名。

```
which ls
```

查找命令 ls 的全路径名

LN 命令

ln: 创建链接 (link)

用法: `ln` [选项] 目标 [链接名]

- ◆ 创建链接，实际上是给系统中已有的某个文件指定另外一个可用于访问它的名称。如果链接指向目录，用户就可以利用该链接直接进入被链接的目录而不用打一大堆的路径名。删除这个链接，不会破坏原来的文件和目录。
- ◆ 链接有两种，一种被称为**硬链接**（Hard Link），另一种被称为**符号链接**（Symbolic Link），默认是建立硬链接。
- ◆ 若没有指定链接名，则链接名与原文件名相同。

LN 命令举例

◆ 常用选项：

-s 建立符号链接。

◆ 举例：

```
ln -s file1.txt /home/lei/
```

在 `/home/lei/` 目录下建立一个指向 `file1.txt` 的符号链接，链接名为 `file1.txt`，若 `/home/lei/` 目录中已存在名为 `file1.txt` 的文件，则系统会报错。

```
ln -s /home/lei/bin /home/student/
```

在 `/home/student/` 目录下建立一个指向目录 `/home/lei/bin` 的符号链接，链接名为 `bin`，若 `/home/student/` 目录中已存在名为 `bin` 的文件或目录，则报错。

文件压缩与解压

GZIP / GUNZIP 命令

gzip/gunzip: 文件压缩与解压缩

用法: `gzip` [选项] 文件列表
`gunzip` [选项] 文件列表

- ◆ Linux下常用的压缩和解压缩命令。
- ◆ 压缩后 gzip 会在每个文件的后面添加扩展名 .gz。
- ◆ 压缩后原文件会被自动删除。
- ◆ 在 windows 下可以用 winzip 或 winrar 解压。

ZIP / UNZIP 命令

zip / unzip: 压缩与解压缩 zip 文件

用法: `zip` [选项] 压缩文件名 文件
`unzip` `zip` 压缩文件

◆ 作用与windows下的 `winzip` 类似，压缩文件或目录

ZIP / UNZIP 命令举例

```
zip file1.zip file1
```

压缩文件 `file1`，压缩后的文件名为 `file1.zip`，原文件保留

```
zip -r data1.zip data1
```

将子目录 `data1/` 下的所有文件压缩到一个文件 `data1.zip`

```
unzip data1.zip
```

释放压缩文件 `data1.zip` 中的所有文件

TAR命令

tar 用于对文件和目录进行打包

用法: `tar` [选项] 打包文件名 文件或目录名

◆ 常用选项:

- c : 创建tar包。
- f : 指定tar包的文件名。
- v : 显示处理文件信息的速度。
- z : 用gzip来压缩/解压缩文件。
- r : 把增加的文件追加到备份文件中。
- u : 更新文件。
- x : 从备份文件中释放文件。

TAR命令举例

```
tar -czvf /tmp/etc.tar.gz /etc
```

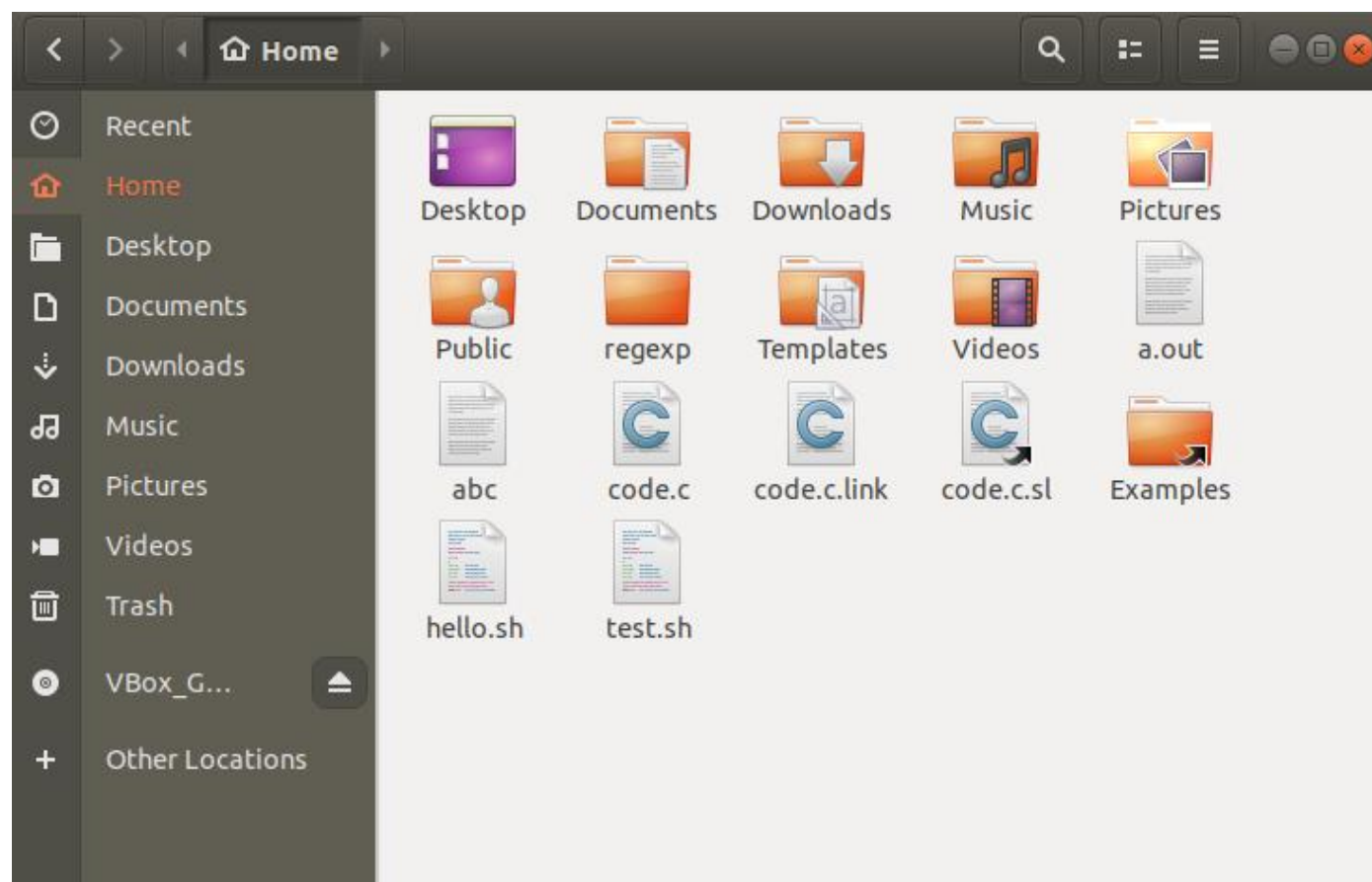
将/etc压缩并打包。

```
tar -xzvf /tmp/etc.tar.gz
```

将打包文件解开。

使用文件管理器操作目录文件

Ubuntu桌面环境使用的文件管理器是Nautilus.



文件权限

文件权限是指对文件的访问控制，决定哪些用户和哪些组对某文件或目录具有哪些访问权限。

Linux文件的访问者身份分为三个类别：

- 所有者：即创建文件的人。所有者对文件具有所有权。
- 所属组：文件所属的组，默认情况下，文件的创建者的主要组为文件的所属组。
- 其他用户：所有者，所属组以及root之外的所有用户。

Linux把文件访问权限也分为3类：


- 可读（read）：读取文件或查看目录，用**r**表示。
- 可写（write）：修改文件或创建、删除文件，用**w**表示。
- 可执行（execute）：执行文件或允许使用cd命令进入目录，用**x**表示。

文件访问权限

- 字母表示法

Linux中每个文件的访问权限可用9个字符表示。

字符	-	r	w	x	r	w	x	r	w	x
位数	1	2	3	4	5	6	7	8	9	10



文件类型 所有者权限 所属组权限 其他用户权限

文件类型：b是块设备文件，d为目录，c为字符设备文件，l为链接文件，-为普通文件

```
lei@lei-VirtualBox:~$ ls -l test.py
-rw-r--r-- 1 lei lei 22 Oct  2 18:48 test.py
```

文件访问权限

- 文件权限的数字表示法

字母表示形式	数字表示	权限
---	0	无任何权限
--X	1	可执行
-W-	2	可写
-WX	3	可写可执行
r--	4	可读
r-X	5	可读可执行
rW-	6	可读可写
rWX	7	可读可写可执行

变更文件访问者身份

CHOWN 命令

chown: 改变文件的所有者和所属组

用法: `chown` [选项] 新用户或组 文件列表

◆ 将指定文件的所有者改为指定的用户或组。用户可以是用户名或UID，组可以是组名或GID。

◆ 常用选项:

-R 递归式地改变指定目录及其下的所有子目录和文件的拥有者。

注: 该命令通常只有超级用户使用

CHOWN 命令举例

```
chown liu file1
```

将文件 file1 的所有者改为 liu

```
chown -R zhang:users /home/lei/data/
```

将目录 /home/lei/data/ 及其下面的所有子目录和文件的属主改成 users 组中的成员 zhang

变更所属组

chgrp 变更文件的所属组

用法: `chgrp` [选项] 新的所属组 文件列表

-R连同子目录中的文件一起变更所属组



chgrp命令也需要root权限

设置文件访问权限

chmod 修改文件权限

用法: `chmod` [选项] 模式[, 模式] 文件名

文件权限用字符表示

+表示增加某种权限，-表示撤销某种权限，=表示指定某种权限。

用户类型：所有者，所属组和其他用户分别用字符u，g，o表示。

权限类型：读，写，执行分别用r，w，x表示。

给所属组用户添加写权限，给其他用户增加读权限

```
chmod g+w,o+r /home/lei/myfile
```

设置文件访问权限

chmod 修改文件权限

用法: `chmod` [选项] 模式[,模式] 文件名

文件权限用数字表示

使文件所有者拥有读写权限，所属组用户和其他用户只能读取

`chmod 644 filename`

等同于

`chmod u=rw-,go=r-- filename`

设置默认文件访问权限

默认情况下，管理员创建的普通文件的权限设置为rw-r--r--，数字表示为644。新创建的目录权限为rwxr-xr-x，数字表示为755。

默认权限是通过umask（掩码）实现的。默认掩码值为022。

默认目录权限： $777-022=755$

默认文件权限： $666-022=644$

查看umask

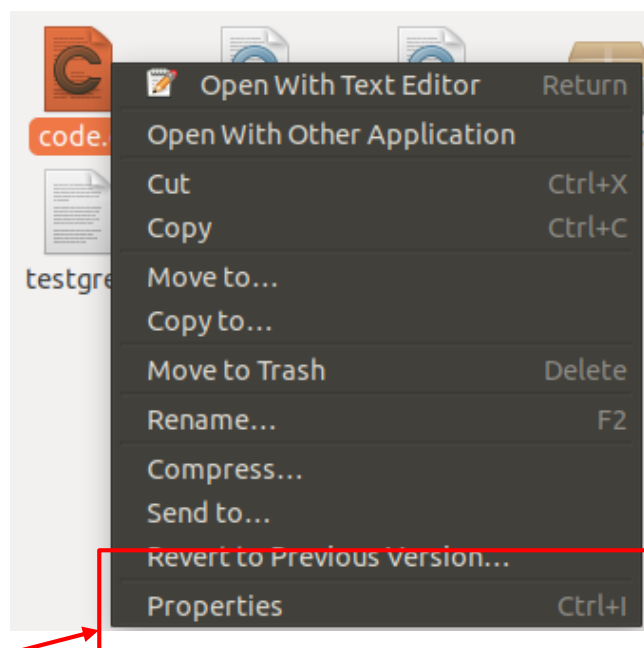
```
umask
```

修改umask

```
umask 002
```

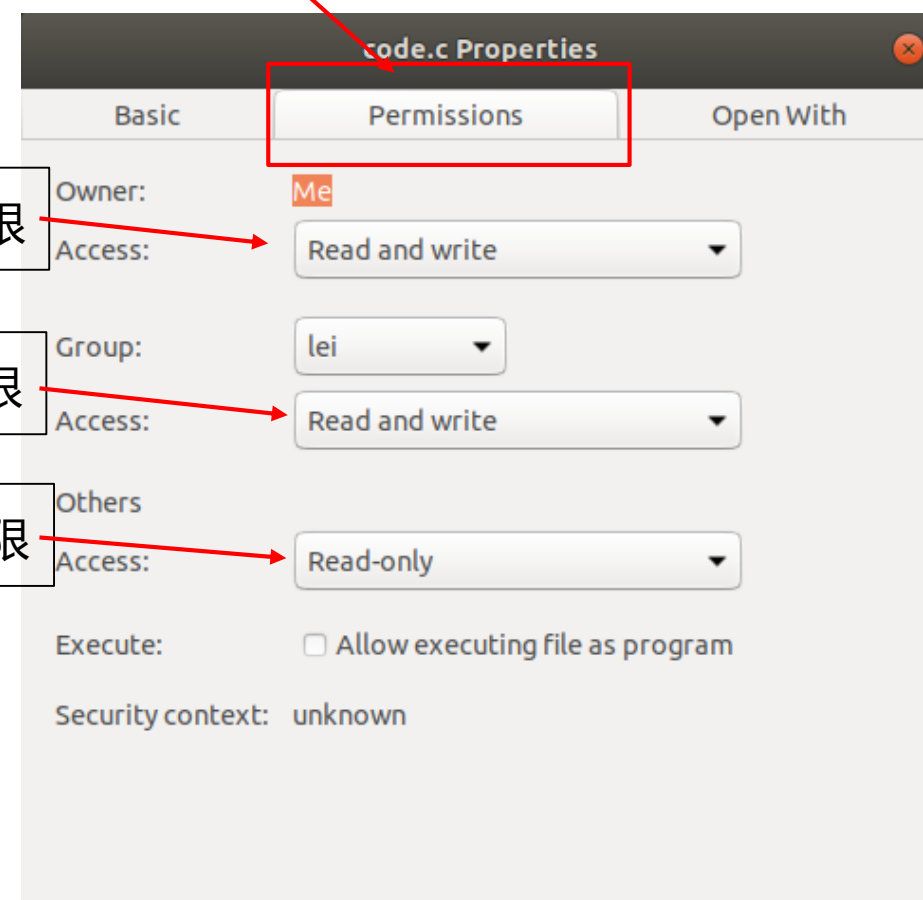
图形界面中管理文件访问权限

1. 在文件上点击右键



2. 选择属性

1. 打开权限标签



2. 所有者权限

3. 所属组权限

4. 其他用户权限

图形界面中管理文件夹访问权限

1. 打开权限标签

2. 所有者权限

3. 所属组权限

4. 其他用户权限

同时改变文件夹中的文件和文件夹权限

The screenshot shows the 'data Properties' dialog box with the 'Permissions' tab selected. The 'Permissions' tab is highlighted with a red box. The 'Owner' is 'Me' with 'Access' set to 'Create and delete files'. The 'Group' is 'lei' with 'Access' set to 'List files only'. The 'Others' have 'Access' set to 'Access files'. A red box highlights the 'Change Permissions for Enclosed Files...' button at the bottom. Red arrows point from the numbered labels to the 'Permissions' tab, the 'Owner' section, the 'Group' section, the 'Others' section, and the 'Change Permissions for Enclosed Files...' button.

通配符

* 匹配 0 或多个字符

a*b a与b之间可以有任意长度的任意字符, 也可以一个也没有, 如aabc**b**, axyz**b**, a012**b**, a**b**。

例如显示当前文件夹下所有python文件 `ls *.py`

? 匹配任意一个字符

a?b a与b之间必须也只能有一个字符, 可以是任意字符, 如aab, abb, acb, a0b。

通配符

[list] 匹配 list 中的任意单一字符

a[xyz]b a与b之间必须也只能有一个字符, 但只能是 x 或 y 或 z,
如: axb, ayb, azb。

[!list] 匹配 除list 中的任意单一字符

a[!0-9]b a与b之间必须也只能有一个字符, 但不能是阿拉伯数字,
如axb, aab, a-b。

[c1-c2] 匹配 c1-c2 中的任意单一字符 如 : [0-9] [a-z]

a[0-9]b 0与9之间必须也只能有一个字符 如a0b, a1b... a9b。

GREP正则表达式元字符集（基本集）

文件通配符通常用于查找文件，而正则表达式通常用于查找文件内容。

- ^ : 锚定行的开始 如：'^grep'匹配所有以grep开头的行。
- \$: 锚定行的结束 如：'grep\$'匹配所有以grep结尾的行。
- . : 匹配一个非换行符的字符 如：'gr.p'匹配gr后接一个任意字符，然后是p。
- * : 匹配零个或多个先前字符 如： '*grep'匹配所有一个或多个空格后紧跟grep的行。
.*一起用代表任意多个字符。
- [] : 匹配一个指定范围内的字符，如'[Gg]rep'匹配Grep和grep。
- [^] : 匹配一个不在指定范围内的字符，如： '[^A-RT-Z]rep'匹配不包含A-R和T-Z的一个字母开头，紧跟rep的行。

GREP正则表达式元字符集（基本集）

- \< : 锚定单词的开始，如:\<grep'匹配包含以grep开头的单词的行。
- \> : 锚定单词的结束，如'grep\>'匹配包含以grep结尾的单词的行。
- \? : 匹配其前面的字符0或1次
- \+ : 匹配其前面的字符至少1次
- x\{m\} : 重复字符x，m次，如：'o\{5\}'匹配包含5个o的行。
- x\{m,\} : 重复字符x,至少m次，如：'o\{5,\}'匹配至少有5个o的行。
- x\{m,n\} : 重复字符x，至少m次，不多于n次，如：'o\{5,10\}'匹配5-10个o的行。
- \w : 匹配文字和数字字符，也就是[A-Za-z0-9]，如： 'G\w*p'匹配以G后跟零个或多个文字或数字 字符，然后是p。
- \W : \w的反置形式，匹配一个或多个非单词字符，如点号，句号等。
- \b : 单词锁定符，如: '\bgrepb\'只匹配grep。

特殊匹配模式

匹配模式	含义
[[:alnum:]]	字母与数字字符,如grep [[:alnum:]] words.txt
[[:alpha:]]	字母
[[:ascii:]]	ASCII字符
[[:blank:]]	空格或制表符
[[:cntrl:]]	ASCII控制字符
[[:digit:]]	数字
[[:graph:]]	非控制、非空格字符
[[:lower:]]	小写字母
[[:print:]]	可打印字符
[[:punct:]]	标点符号字符
[[:space:]]	空白字符，包括垂直制表符

GREP -E的元字符扩展集

扩展正则表达式与标准正则表达式的区别在省略了转义字符\

`+` : 匹配一个或多个先前的字符。如： `'[a-z]+able'` , 匹配一个或多个小写字母后跟able的串，如loveable,enable,disable等。

`?` : 匹配零个或多个先前的字符。如： `'gr?p'`匹配gr后跟一个或没有字符，然后是p的行。

`a|b|c` : 匹配a或b或c。如： `grep|sed`匹配grep或sed

`()` : 分组符号，如： `love(able|rs)ov+`匹配loveable或lovers，匹配一个或多个ov。

`x{m},x{m,},x{m,n}` : 作用同`x\{m\},x\{m,\},x\{m,n\}`

GREP命令选项

-G或--basic-regexp 将pattern视为普通的表示法来使用。

```
grep -G 'g\w\{1,5\}s' testgrep  
grepis a good tool.
```

```
lei@lei-VirtualBox:~$ grep -G 'g\w\{1,5\}s' testgrep  
grepis a good tool.
```

-E或--extended-regexp 将pattern为扩展的表示法来使用。

```
grep -G 'g\w*s' testgrep  
grepis a good tool.
```

```
lei@lei-VirtualBox:~$ grep -G 'g\w*s' testgrep  
grepis a good tool.
```

-F或--fixed-regexp 将pattern视为固定字符串的列表。

```
grep -F '^gr.p' testgrep  
^gr.p$
```

```
lei@lei-VirtualBox:~$ grep -F '^gr.p' testgrep  
^gr.p$
```

grep is a good tool.

grepis a good tool.

^gr.p\$

westaaaest

loveable或loversovov

2019-10-16

2019-9-1

2019年10月22号

2019年10月22日

GREP命令选项

```
grep -G 'w\(es\)t.*\1' testgrep  
westaaaest
```

```
lei@lei-VirtualBox:~$ grep -G 'w\(es\)t.*\1' testgrep  
westaaaest
```

```
grep -E 'w(es)t.*\1' testgrep  
westaaaest
```

```
lei@lei-VirtualBox:~$ grep -E 'w(es)t.*\1' testgrep  
westaaaest
```

grep is a good tool.

grepis a good tool.

^gr.p\$

westaaaest

loveable或loversovov

2019-10-16

2019-9-1

2019年10月22号

2019年10月22日

使用实例

```
grep -E '[0-9]{4}-[0-9]{1,2}-[0-9]{1,2}' testgrep  
2019-10-16  
2019-9-1
```

```
lei@lei-VirtualBox:~$ grep -E '[0-9]{4}-[0-9]{1,2}-[0-9]{1,2}' testgrep  
2019-10-16  
2019-9-1
```

```
grep -E 'love(able|rs)ov+' testgrep  
loveable或loversovov
```

```
lei@lei-VirtualBox:~$ grep -E 'love(able|rs)ov+' testgrep  
loveable或loversovov
```

```
grep -E '[0-9]{4}年[0-9]{1,2}月[0-9]{1,2}(号|日)' testgrep
```

```
lei@lei-VirtualBox:~$ grep -E '[0-9]{4}年[0-9]{1,2}月[0-9]{1,2}(号|日)' testgrep  
2019年10月22号  
2019年10月22日
```

grep is a good tool.

grep is a good tool.

^gr.p\$

westaaaest

loveable或loversovov

2019-10-16

2019-9-1

2019年10月22号

2019年10月22日

FIND

-regextype "type" , 可选的类型有 'findutils-default' , 'awk' , 'egrep' , 'ed' , 'emacs' , 'gnu-awk' , 'grep' , 'posix-awk' , 'posix-basic' , 'posix-egrep' , 'posix-extended' , 'posix-minimal-basic' , 'sed' .

-regex选项不是匹配文件名，而是匹配完整的文件名（包括路径，当前路径以./开头）。

例如，匹配当前文件夹下文件名全是字母的文件

```
find ./ -regextype posix-extended -regex "\./[a-z]+"
```

```
lei@lei-VirtualBox:~$ find ./ -regextype posix-extended -regex "\./[a-z]+"\n./data\n./testgrep\n./abc\n./regexp\n./linkdir
```