
Table of Contents

Linux基础	1.1
Linux云计算运维工程师	1.1.1
计算机基础	1.1.2
cpu	1.1.2.1
内存	1.1.2.2
显卡	1.1.2.3
磁盘	1.1.2.4
主板	1.1.2.5
电源	1.1.2.6
网卡	1.1.2.7
服务器基本知识	1.1.2.8
软件基本知识	1.1.2.9
LINUX 系统介绍与环境搭建准备	1.2
vmware安装Linux	1.2.1
vmware网络管理与基本配置	1.2.2
Linux远程连接	1.2.3
Linux系统命令基础	1.3
Linux系统文件与启动流程	1.3.1
Linux文件目录管理命令	1.4
Linux文件属性与管理	1.4.1
Linux用户管理	1.4.2
Linux文件权限	1.4.3
Linux文件管理练习题	1.4.4
Linux通配符	1.5
Linux正则表达式与grep	1.5.1
Linux三剑客sed	1.5.2
Linux三剑客awk基础	1.5.3
awk分隔符	1.5.3.1
awk变量	1.5.3.2
awk格式化	1.5.3.3
awk模式pattern	1.5.3.4
awk动作	1.5.3.5
awk数组	1.5.3.6

awk内置函数	1.5.3.7
三剑客练习题	1.5.4
Shell编程	1.6
Shell变量	1.6.1
Linux定时任务	1.7
Linux磁盘管理与文件系统	1.8
Linux软硬链接	1.8.1
Linux文件系统管理	1.8.2
Linux文件系统挂载	1.8.3
RAID技术	1.8.4
LVM逻辑卷	1.8.5
Linux系统优化	1.9
Linux软件包管理	1.9.1
Linux网络基础	1.10
Linux网络管理命令	1.10.1
Linux进程管理	1.11
互联网服务基础	1.12
dns	1.12.1
sshd	1.12.2
web基础	1.12.3
apache	1.12.3.1
黄金架构LAMP	1.12.4
vsftpd	1.12.5
nfs	1.12.6
samba	1.12.7
iptables	1.12.8
企业级负载均衡实战	1.13
Nginx	1.13.1
Nginx静态资源站点	1.13.2
LNMP黄金架构	1.13.3
Introduction	1.14

[TOC]

Linux系统命令基础



前面咱们已经成功安装了Linux系统--centos7，那么现在跟着超哥奔向Linux命令行的世界。

Linux命令格式

命令		条件/参数		对象/文件/目录
结婚	空格	-有车有房有存款	空格	白富美
结婚	空格	-没有车有房有存款	空格	是个女的就行
rm	空格	-f	空格	/tmp/olddbey.txt

1.一般情况下，【参数】是可选的，一些情况下【文件或路径】也是可选的

2.参数 > 同一个命令，跟上不同的参数执行不同的功能

执行linux命令，添加参数的目的是让命令更加贴切实际工作的需要！

linux命令，参数之间，普遍应该用一个或多个空格分割！

Linux命令行



命令提示符

[py@pylinux ~]\$ 普通用户py，登陆后

[root@pylinux ~]# 超级用户root，登录后

root代表当前登录的用户

@ 分隔符

pylinux 主机名

~ 当前的登录的位置，此时是家目录

超级用户身份提示符

\$ 普通用户身份提示符

操作系统目录分隔符

Windows平台命令行目录分隔符



Linux平台命令行目录分隔符




Linux与Windows的目录结构比较

Linux首先是建立一个根"/"文件系统，所有的目录也都是由根目录衍生出来。

登录系统后，在当前命令窗口输入命令：

```
ls /
```

查看结果如下图：



```
[root@pylinux ~]#  
[root@pylinux ~]#  
[root@pylinux ~]#  
[root@pylinux ~]# ls /  
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var  
boot  etc  lib  media  opt  root  sbin  sys  usr  
[root@pylinux ~]#  
[root@pylinux ~]#  
[root@pylinux ~]#  
[root@pylinux ~]#  
[root@pylinux ~]#
```

在Linux底下，所有的文件与目录都是由根目录开始，是目录与文件的源头，然后一个个的分支下来，如同树枝状，因此称为这种目录配置为：**目录树**。

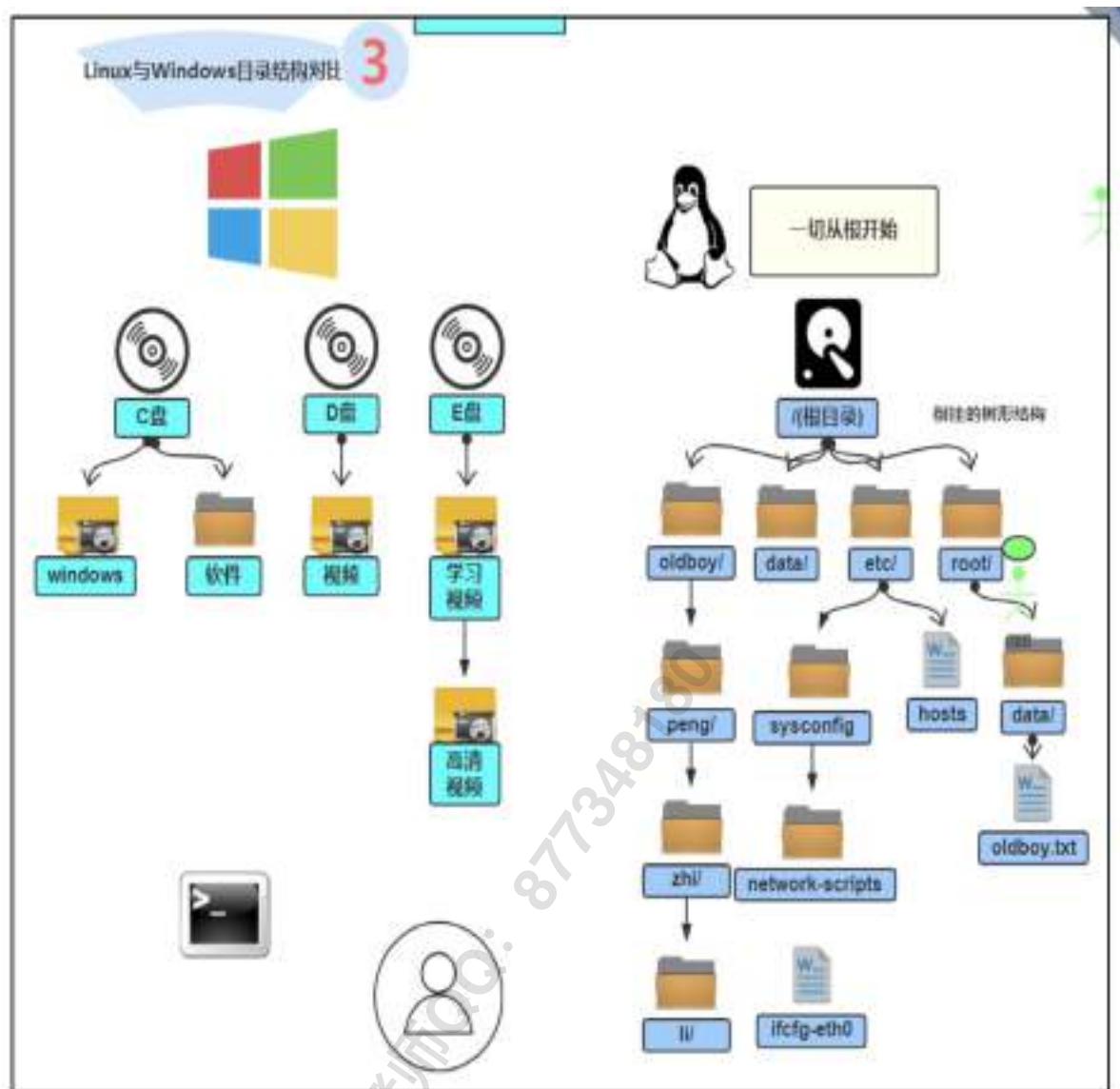
目录树的特点是什么呢？

- 目录树的起始点是根目录(/,root);
- 每一个目录不止能使用本地的文件系统，也可以使用网络上的文件系统，可以利用NFS服务器挂载特定目录。
- 每一个文件在此目录树中的文件名，包含完整路径都是独一无二的。

图解linux与Windows目录

Linux与windows区别

- windows特点:E:\学习视频\高清视频\
- Linux目录特点:/etc/hosts /root/data/oldboy.txt



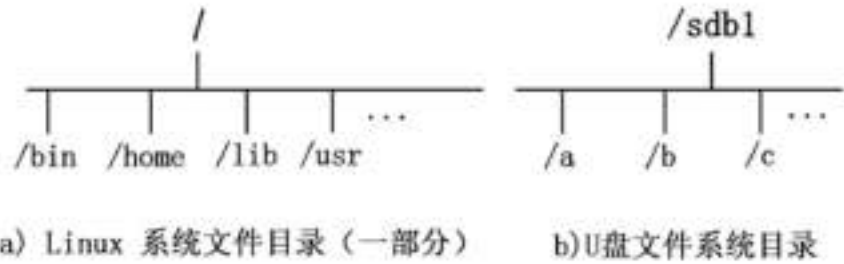
Linux 系统目录结构基本特点：

- 1.Linux下一切从 根 开始
- 2.Linux下面的目录是一个有层次的目录结构
- 3.在linux中每个目录可以挂载到不同的设备(磁盘)上
- 4.Linux 下设备不挂载不能使用，不挂载的设备相当于没门没窗户的监狱(进不去出不来)，挂载相当于给设备创造了一个入口(挂载点，一般为目录)

Linux目录挂载

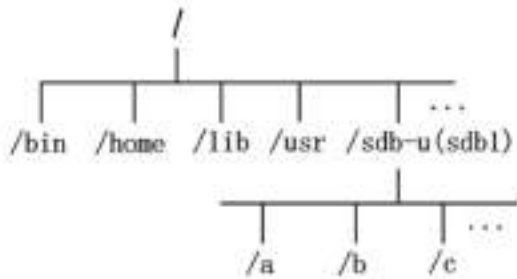
挂载通常是将一个 存储设备 挂接到一个已经存在的 目录 上，访问这个 目录 就是访问该存储设备的内容。

对于Linux系统来说，一切接文件，所有文件都放在以 根目录 为起点的树形目录结构中，任何硬件设备也都是文件形式



如图所示，是U盘存储设备和Linux系统自己的文件系统结构，此时Linux想要使用U盘的硬件设备，必须将Linux 本身的目录 和硬件设备的文件目录合二为一，此过程就称之为 挂载 。

挂载操作会隐藏原本Linux目录中的文件，因此选择Linux本身的目录，最好是新建空目录用于挂载
挂载之后，这个目录被称为挂载点



此时U盘文件系统已经是Linux文件系统的一部分，访问/sdb-u文件夹，即是访问访问U盘系统中的文件夹。



- **/bin**: bin是Binary的缩写, 这个目录存放着最经常使用的命令。
- **/boot**: 这里存放的是启动Linux时使用的一些核心文件，包括一些连接文件以及镜像文件。
- **/dev**: dev是Device(设备)的缩写, 该目录下存放的是Linux的外部设备，在Linux中访问设备的方式和访问文件的方式是相同的。
- **/etc**: 这个目录用来存放所有的系统管理所需要的配置文件和子目录。

- **/home**: 用户的主目录, 在Linux中, 每个用户都有一个自己的目录, 一般该目录名是以用户的账号命名的。
- **/lib**: 这个目录里存放着系统最基本的动态连接共享库, 其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。
- **/lost+found**: 这个目录一般情况下是空的, 当系统非法关机后, 这里就存放了一些文件。
- **/media**: linux系统会自动识别一些设备, 例如U盘、光驱等等, 当识别后, linux会把识别的设备挂载到这个目录下。
- **/mnt**: 系统提供该目录是为了让用户临时挂载别的文件系统的, 我们可以将光驱挂载在/mnt/上, 然后进入该目录就可以查看光驱里的内容了。
- **/opt**: 这是给主机额外安装软件所摆放的目录。比如你安装一个ORACLE数据库则就可以放到这个目录下。默认是空的。
- **/proc**: 这个目录是一个虚拟的目录, 它是系统内存的映射, 我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里, 我们也可以直接修改里面的某些文件, 比如可以通过下面的命令来屏蔽主机的ping命令, 使别人无法ping你的机器:

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

- **/root**: 该目录为系统管理员, 也称作超级权限者的用户主目录。
- **/sbin**: s就是Super User的意思, 这里存放的是系统管理员使用的系统管理程序。
- **/selinux**: 这个目录是Redhat/CentOS所特有的目录, Selinux是一个安全机制, 类似于windows的防火墙, 但是这套机制比较复杂, 这个目录就是存放selinux相关的文件的。
- **/srv**: 该目录存放一些服务启动之后需要提取的数据。
- **/sys**: 这是linux2.6内核的一个很大的变化。该目录下安装了2.6内核中新出现的一个文件系统sysfs。

sysfs文件系统集成了下面3种文件系统的信息: 针对进程信息的proc文件系统、针对设备的devfs文件系统以及针对伪终端的devpts文件系统。该文件系统是内核设备树的一个直观反映。当一个内核对象被创建的时候, 对应的文件和目录也在内核对象子系统中被创建。

- **/tmp**: 这个目录是用来存放一些临时文件的。
- **/usr**: 这是一个非常重要的目录, 用户的很多应用程序和文件都放在这个目录下, 类似于windows下的program files目录。
- **/usr/bin**: 系统用户使用的应用程序。
- **/usr/sbin**: 超级用户使用的比较高级的管理程序和系统守护程序。
- **/usr/src**: 内核源代码默认的放置目录。
- **/var**: 这个目录中存放着在不断扩充着的东西, 我们习惯将那些经常被修改的目录放在这个目录下。包括各种日志文件。

在linux系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。

/etc: 上边也提到了，这个是系统中的配置文件，如果你更改了该目录下的某个文件可能会导致系统不能启动。

/bin, /sbin, /usr/bin, /usr/sbin: 这是系统预设的执行文件的放置目录，比如 **ls** 就是在 **/bin/ls** 目录下的。

值得提出的是，**/bin, /usr/bin** 是给系统用户使用的指令（除root外的通用户），而 **/sbin, /usr/sbin** 则是给root使用的指令。

/var: 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在 **/var/log** 目录下，另外mail的预设放置也是在这里。

为什么要学Linux命令

- Linux从诞生就是黑屏界面，所有操作倚靠命令完成，如磁盘读写、文件操作、网络管理等
- 企业中，服务器的维护工作都是 **ssh客户端** 完成，没有图形界面
- 程序员想要管理linux服务器，必须学习常用命令

Linux命令学习方法

- 熟能生巧，多敲打，多练习即可
- 不可能一下子掌握所有命令用法，学会使用搜索引擎查阅命令资料

当年超哥在一家美资企业，一位台湾老程序员送我的一本书。。。

可能是看我骨骼惊奇吧！！



Linux文件及目录管理命令

命令	对应英文	作用
ls	list	查看文件夹内容

pwd	print work directory	查看当前所在目录
cd 目录名	Change directory	切换文件夹
touch 文件名	touch	如果文件不存在，则创建
mkdir 目录名	Make directory	创建目录
rm 文件名	Remove	删除指定文件

我们知道切换目录的指令是cd，那么首先得知道如何切换目录，这个得用心记呀！

.	当前目录
..	上一层目录
-	前一个工作目录
~	当前【用户】所在的家目录
/	顶级根目录

cd命令，变换目录

cd是change directory的缩写，这是用来变换工作目录的命令，注意命令和目录之间有一个空格。

```
[root@localhost ~]# cd ~      进入用户家目录
[root@localhost ~]# pwd      打印当前目录
/root
[root@localhost ~]# cd      没有加上路径，也代表进入家目录
[root@localhost ~]# cd ..    去往上一级目录
[root@localhost /]# cd -     返回刚才的目录
/root
[root@localhost ~]# cd /home/  进入/home目录
[root@localhost home]# cd ../  进入指定的上一层目录
[root@localhost /]# _
```

需要注意的是，在所有目录底下都存在两个目录，分别是【.】和【..】，分别代表当前目录，上层目录！那么如何证明它的存在呢？

```
命令： ls -la /
查看命令解释： man ls （Linux下的帮助指令）
结论： ls - list directory contents （列出目录内容）
ls -la / 以竖状格式化显示列出/目录所有内容
```

```

root@localhost ~# ls -la /
total 16
dr-xr-xr-x. 17 root root 224 Jul 16 00:17 .
dr-xr-xr-x. 17 root root 224 Jul 16 00:17 ..
lrwxrwxrwx. 1 root root 7 Jul 16 00:13 bin -> usr/bin
dr-xr-xr-x. 5 root root 4096 Jul 16 00:18 boot
lrwxr-xr-x. 21 root root 3200 Jul 16 00:19 dev
lrwxr-xr-x. 75 root root 8192 Jul 16 00:19 etc
lrwxr-xr-x. 2 root root 6 Apr 11 12:59 home
lrwxrwxrwx. 1 root root 7 Jul 16 00:13 lib -> usr/lib
lrwxrwxrwx. 1 root root 9 Jul 16 00:13 lib64 -> usr/lib64
lrwxr-xr-x. 2 root root 6 Apr 11 12:59 media
lrwxr-xr-x. 2 root root 6 Apr 11 12:59 mnt
lrwxr-xr-x. 2 root root 6 Apr 11 12:59 opt
dr-xr-xr-x. 189 root root 8 Jul 16 00:19 proc
dr-xr-xr-x. 2 root root 114 Jul 16 00:10 root
lrwxr-xr-x. 23 root root 680 Jul 16 12:51 run
lrwxrwxrwx. 1 root root 8 Jul 16 00:13 sbin -> usr/sbin
lrwxr-xr-x. 2 root root 6 Apr 11 12:59 srv
dr-xr-xr-x. 13 root root 8 Jul 16 00:19 sys
lrwxrwxrwt. 8 root root 151 Jul 16 12:51 tmp
lrwxr-xr-x. 13 root root 155 Jul 16 00:13 usr
lrwxr-xr-x. 19 root root 267 Jul 16 00:19 var

```

命令

两个目录
· 当前
.. 上一级

tree命令

以树形结构显示目录下内容



tree命令可能要单独安装:

yum install tree -y

tree命令语法:

tree常用参数

- C 在文件和目录清单加上色彩，便于区分各种类型。
- d 显示目录名称而非内容。
- D 列出文件或目录的更改时间。
- f 在每个文件或目录之前，显示完整的相对路径名称。
- F 在条目后加上文件类型的指示符号(*, /, =, @, |, 其中的一个) 目录/

ls命令

显示目录下内容及属性信息的命令

- a 显示指定目录下所有子目录与文件，包括以.开头的隐藏文件
- l 以列表方式显示文件的详细信息 `ls -l` 等于 `ll` 用法
- h, --human-readable 与 -l 一起，以易于阅读的格式输出文件大小
(例如 1K 234M 2G)
- t 根据最后修改时间排序，默认是以文件名排序，通常与 -l 连用
- F 在条目后加上文件类型的指示符号(*, /, =, @, |, 其中的一个)
注:可以标识文件类型
加上 * 代表可执行的普通文件
加上 = 表示套接字
加上 | 表示FIFOs(队列系统)
加上 @表示符号链接
加上 / 表示文件夹
- d 显示目录本身的信息 而不是显示目录的内容
- r, --reverse 逆序排列
- S 根据文件大小排序, 从大到小排序
- i 显示索引节点信息(索引节点相当于身份证号)
- full-time 以完整的时间格式输出(也就是按照中国的时间日期显示)

案例

```
ls -lt 按照时间进行排序
ls -lrt 找出最新的文件
ls -d */ 列出当前所有目录
ll -hS ./* 显示当前目录下所有内容详细，且以kb,mb,gb单位从大到小排序
```

```
[root@pylinux bin]# ls -F
2to3*          f2py3.7*      imagemagick_remove_bin*  python3.7m-config*  tango_monitor*      taurusimage*
2to3-3.7*      fardango*     pasteurize*             python3-config@     tango_property*     tauruspanel*
CopyCatDS*     floak*        pip3*                   pyvenv@             tango_servers*      taurusplot*
csv2tango*     folder05*     pip3.7*                 sardasact*          taurusconfigbrowser*  taurustestsuite*
ctds*          folder-gui*   __pycache__/*           skivi*               tauruscure*          taurustrend*
django-admin*  futurize*     pydoc3@                 sqlformat*           taurustrendd*        taurustrend2d*
django-admin.py*  glance*       python3@                tango2csv*           taurusdevicepanel*   virtualenv*
dynamicDS*      heiheihei 白色普通文件  python3.7*           tango2json*          taurusform*          WorkerDS*
easy_install-3.7*  idle3@       python3.7-config@       tango_cleanup*       taurussgal*          taurusconcatlog*
f2py*          imagemagick_download_bin*  python3.7m*           tango_host*
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
[root@pylinux bin]#
```

深蓝色以 / 结尾是文件夹 浅蓝色且结尾是 @ 代表是链接文件

绿色且结尾是 * 代表可执行文件

mkdir命令

创建文件夹

用法: `mkdir [选项]... 目录...`

若指定目录不存在则创建目录。

`-m, --mode=模式` 设置权限模式(类似`chmod`), 而不是`rw-rw-rw-` 减`umask`
`-p, --parents` 需要时创建目标目录的上层目录, 但即使这些目录已存在也不当作错误处理
`mkdir {1..3}`加花括号创建连续的目录, 用`..` 隔开 花括号内可以是连续的数字、连续的字母`mkdir {a..e}`

案例

`mkdir {alex,pyyu,mjj}` 创建三个文件夹, 逗号隔开

`mkdir alex{1..5}` 创建连续的目录

`mkdir cunzhang longting` 创建少量连续目录

```
[root@pylinux tmp]# mkdir alex{1..5}
[root@pylinux tmp]# ls
alex1 alex2 alex3 alex4 alex5      创建大量连续的文件夹
[root@pylinux tmp]#
[root@pylinux tmp]# mkdir {alex,pyyu,mjj} 创建少量文件夹
[root@pylinux tmp]# ls
alex alex1 alex2 alex3 alex4 alex5 mjj pyyu
[root@pylinux tmp]#
[root@pylinux tmp]# mkdir cunzhang longting 创建少量文件夹
[root@pylinux tmp]# ls
alex alex1 alex2 alex3 alex4 alex5 cunzhang longting mjj pyyu
[root@pylinux tmp]#
[root@pylinux tmp]# mkdir -p ./boy/{mjj,cunzhang} ./girl/{longting} 递归创建文件夹
[root@pylinux tmp]# ls
alex alex1 alex2 alex3 alex4 alex5 boy cunzhang girl longting mjj pyyu
[root@pylinux tmp]# tree
--
|-- alex
|-- alex1
|-- alex2
|-- alex3
|-- alex4
|-- alex5
|-- boy
|   |-- cunzhang  boy底下有cunzhang. mjj
|   |-- mjj
|-- cunzhang
|-- girl
|   |-- {longting}  girl底下有longting
|-- longting
|-- mjj
|-- pyyu
15 directories, 0 files
```

touch命令

创建文件或修改文件时间戳

用法: `touch [选项]... 文件...`

将每个文件的访问时间和修改时间改为当前时间。

不存在的文件将会被创建为空文件, 除非使用`-c` 或`-h` 选项。

`touch {连续数字或字母}` 创建多个文件序列

`touch {1..10}`

```
touch {a..z}
```

- c, --no-create 不创建任何文件
- t STAMP 使用[[CC]YY]MMDDhhmm[.ss] 格式的时间替代当前时间
- r, --reference=文件 使用指定文件的时间属性替代当前文件时间

案例

```
[root@pylinux tmp]# touch {1..5}
[root@pylinux tmp]#
[root@pylinux tmp]# touch alex{1..5}
[root@pylinux tmp]#
[root@pylinux tmp]# touch {001..5}
[root@pylinux tmp]# ls
001 002 003 004 005 1 2 3 4 5 alex1 alex2 alex3 alex4 alex5
[root@pylinux tmp]#
```

修改文件时间

```
touch -t 06010808 alex1    #修改alex1文件的时间是 6月1号8点8分
touch -r alex1 alex2      #把alex2的时间改成alex1一样
```

cp复制

复制命令

windows复制

可以说是相当简单了

ctrl + c 复制

ctrl + v 黏贴



linux复制

用法: cp [选项]... [-T] 源文件 目标文件
或: cp [选项]... 源文件... 目录

或: `cp [选项]... -t 目录 源文件...`

将源文件复制至目标文件, 或将多个源文件复制至目标目录。

`-r` 递归式复制目录, 即复制目录下的所有层级的子目录及文件 `-p` 复制的时候 保持属性不变

`-d` 复制的时候保持软连接(快捷方式)

`-a` 等于`-pdr`

`-p` 等于`--preserve=模式`, 所有权, 时间戳, 复制文件时保持源文件的权限、时间属性

`-i, --interactive` 覆盖前询问提示

案例

```
复制 > copy > cp
#移动xxx.py到/tmp目录下
cp xxx.py /tmp/
#移动xxx.py顺便改名为chaoge.py
cp xxx.py /tmp/chaoge.py
```

Linux下面很多命令, 一般没有办法直接处理文件夹, 因此需要加上 (参数)

`cp -r` 递归, 复制目录以及目录的子孙后代

`cp -p` 复制文件, 同时保持文件属性不变 可以用`stat`

`cp -a` 相当于`-pdr`

#递归复制test文件夹, 为test2

```
cp -r test test2
```

`cp`是个好命令, 操作文件前, 先备份

```
cp main.py main.py.bak
```

移动多个文件, 放入文件夹c中

```
cp -r 文件1 文件2 文件夹a 文件夹c
```

案例2

```
[root@pylinux opt]# cp luffy_boy.zip luffy_boy.zip.bak2
```

```
cp: 是否覆盖"luffy_boy.zip.bak2"? y
```

```
[root@pylinux opt]# cp luffy_boy.zip luffy_boy.zip.bak2 -i
```

```
cp: 是否覆盖"luffy_boy.zip.bak2"? y
```

`cp`确认是否覆盖是`-i`参数作用, 默认`alias`因为添加了别名

```
[root@pylinux opt]# alias
```

```
alias cp='cp -i'
```

```
[root@pylinux opt]# cp luffyCity/ luffyCity2 #必须添加-r参数才可以复制递归目录
```

```
cp: omitting directory 'luffyCity/'
```

```
[root@pylinux opt]#
```

```
[root@pylinux opt]#
```

```
[root@pylinux opt]#
```

```
[root@pylinux opt]# cp -r luffyCity/ luffyCity2
```

```
[root@pylinux opt]#
```



```
[root@pylinux opt]#
[root@pylinux opt]# ls
luffyCity luffyCity2
```

取消cp别名的方式

- 使用命令绝对路径
- 命令开头用反斜线 \
- 取消cp命令别名
- 写入环境变量配置文件

```
1.
[root@pylinux opt]# which cp
alias cp='cp -i'
/usr/bin/cp
[root@pylinux opt]# /usr/bin/cp luffy_boy.zip luffy_boy.zip.bak

2.
[root@pylinux opt]# \cp luffy_boy.zip luffy_boy.zip.bak

3.
[root@pylinux opt]# unalias cp
[root@pylinux opt]#
[root@pylinux opt]# cp luffy_boy.zip luffy_boy.zip.bak

4.
[root@pylinux opt]# vim ~/.bashrc #可以注释掉如下配置
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
#alias cp='cp -i'
alias mv='mv -i'
```

快速备份配置文件



```
[root@pylinux opt]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0.bak
[root@pylinux opt]#
[root@pylinux opt]# cp /etc/sysconfig/network-scripts/ifcfg-eth0{,.origin}
[root@pylinux opt]#
[root@pylinux opt]# ls /etc/sysconfig/network-scripts/ifcfg-eth0
ifcfg-eth0 ifcfg-eth0.bak ifcfg-eth0.origin
[root@pylinux opt]# ls /etc/sysconfig/network-scripts/ifcfg-eth0
```

源文件 路径重复了，可以省略写法 复制备份文件

此为bash语法，大括号拆开如上的路径，再复制

简写方式备份了网卡配置文件

mv命令



什么？移动命令？

mv命令就是move的缩写，作用是移动或是重命名文件

用法：mv [选项]... [-T] 源文件 目标文件

或：mv [选项]... 源文件... 目录

或：mv [选项]... -t 目录 源文件...

将源文件重命名为目标文件，或将源文件移动至指定目录。

-f, --force 覆盖前不询问

-i, --interactive 覆盖前询问

-n, --no-clobber 不覆盖已存在文件如果您指定了-i、-f、-n 中的多个，仅最后一个生效。

-t, --target-directory=DIRECTORY 将所有参数指定的源文件或目录移动至 指定目录

-u, --update 只在源文件文件比目标文件新，或目标文件不存在时才进行移动

mv移动|重命名

```

[root@pylinux tmp]# mkdir -p alex/dsb
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# tree
└── alex
    └── dsb

```

当前/tmp目录下有个alex文件夹，里面有个dsb文件夹

```

2 directories, 0 files
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# mkdir lufficity 创建lufficity文件夹
[root@pylinux tmp]#
[root@pylinux tmp]# mv alex/ lufficity/
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# ls
lufficity
[root@pylinux tmp]# ls lufficity/
alex

```

将alex文件夹，放入lufficity内
目标文件夹存在，则是移动

```

[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# mkdir -p alex/dsb
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# ls
alex lufficity
[root@pylinux tmp]# mv alex/ oldboy_alex/
[root@pylinux tmp]# ls
lufficity oldboy_alex
[root@pylinux tmp]# ls oldboy_alex/
dsb

```

如果目标文件夹是不存在的
mv命令作用就是 重命名

mv案例

移动（搬家）命令 > move > mv

1. 给文件重命名

```
mv abc abc.py
```

2. 如果目标文件存在，-i参数则提示是否覆盖

```
mv test1.txt test2.txt
```

3. 使用反斜杠命令屏蔽别名

```
\mv kunkun wuyifan
```

4. 取消别名

5. 移动单个文件

```
mv file1.txt dir/
```

6. 移动多个文件

```
mv file1.txt file2.txt dir/
```

7. 通配符移动多个文件

```
mv dir/file* ../
```

```

[root@pylinux tmp]# touch caixukun
[root@pylinux tmp]#
[root@pylinux tmp]# mv caixukun kunkun
[root@pylinux tmp]#
[root@pylinux tmp]# ls
kunkun
[root@pylinux tmp]#
[root@pylinux tmp]# touch wuyifan
[root@pylinux tmp]#
[root@pylinux tmp]# touch kuanmian
[root@pylinux tmp]#
[root@pylinux tmp]# mv wuyifan kuanmian
mv: 是否覆盖 "kuanmian"? n

```

目标文件 kunkun 不存在
mv 是 改名作用

目标文件 kuanmian 已存在
mv 通过 -i 参数 提示是否覆盖

rm命令

Linux在使用rm（删除）、cp（覆盖）、mv（搬家）等命令的时候，必须非常小心，因为这些命令都是“炸弹”，想必大家都听过“删库到跑路”，一言不合“rm -rf /”，假如你真的这么做了，那么。。。上帝保佑你



用法：rm [选项]... 文件...

删除（unlink）文件。

rm命令就是remove的含义，删除一个或者多个文件，这是Linux系统重要命令

-f, --force	强制删除。忽略不存在的文件，不提示确认
-i	在删除前需要确认
-I	在删除超过三个文件或者递归删除前要求确认。
-d, --dir	删除空目录
-r, -R, --recursive	递归删除目录及其内容
-v, --verbose	详细显示进行的步骤
--help	显示此帮助信息并退出
--version	显示版本信息并退出

案例

1. 删除普通文件, 需要确认提示, 默认添加了-i参数

```
rm file1.txt
```

2. 强制删除文件, 不提示

```
rm -f file2.txt
```

3. 递归删除文件夹

```
[root@pylinux tmp]# rm -r heh/
rm: 是否进入目录"heh/"? y
rm: 是否删除普通空文件 "heh/kuanmian2"? y
rm: 是否删除普通空文件 "heh/kuanmian"? y
rm: 是否删除目录 "heh/"? y
```

炸弹命令

务必看清楚敲打的命令, 是否正确、不得有空格

务必看清楚敲打的命令, 是否正确、不得有空格

务必看清楚敲打的命令, 是否正确、不得有空格

1. 强制删除且不让用户确认

```
rm -rf 文件夹
```

2. 强制删除且显示过程

```
[root@pylinux tmp]# rm -rfv ./*
已删除"./456.txt"
已删除目录: "./q/w/e/r/t/yt"
已删除目录: "./q/w/e/r/t"
已删除目录: "./q/w/e/r"
已删除目录: "./q/w/e"
已删除目录: "./q/w"
已删除目录: "./q"
```

注意文件恢复

rm命令删除文件后可以通过如ext3grep工具恢复数据, 若是想要粉碎文件, 还有其他方式

Linux帮助命令

*man*帮助命令

当你不知道linux命令如何使用的时候, 使用man命令帮助你

语法

```
man 命令
```

如:

```
man ls
```

进入man帮助文档后，按下q退出



使用--help参数

语法：

命令 --help

帮助命令的精简版

如 ls --help

help命令获取帮助

语法：

help 命令

只针对bash内置命令

info命令获取帮助

语法：

info 命令

从互联网中获取

互联网有很多在线linux中文文档网站

Linux开关机命令

shutdown 重启或者关机

```
[root@pylinux ~]#shutdown --help
shutdown [OPTIONS...] [TIME] [WALL...]

Shut down the system.
```

重启

```
语法：
shutdown -r参数    -r --reboot    Reboot the machine

shutdown -r 10      #十分钟后重启
shutdown -r 0        #立刻重启
shutdown -r now      #立刻重启
```

关机

```
语法：
shutdown -h        --halt    停止的含义

shutdown -h 10      #十分钟后关机
shutdown -h 0
shutdown -h now      #立即关机
```

halt, *poweroff*, *reboot*命令关机与重启

reboot 重启

poweroff
halt 关机

关机、重启、注销命令列表

命令	说明
shutdown -h now	立刻关机，企业用法

shutdown -h 1	1分钟后关机，也可以写时间如 11:30
halt	立刻关闭系统，需手工切断电源
init 0	切换运行级别为0，0表示关机
poweroff	立刻关闭系统，且关闭电源
重启	
reboot	立刻重启机器，企业用法
Shutdown -r now	立刻重启，企业用法
shutdown -r 1	一分钟后重启
Init 6	切换运行级别为6，此级别是重启
注销命令	
logout	注销退出当前用户
exit	注销退出当前用户，快捷键ctrl + d

Linux命令行常用快捷键

```
ctrl + c    cancel取消当前操作
ctrl + l    清空屏幕内容
ctrl + d    退出当前用户
ctrl + a    光标移到行首
ctrl + e    光标移到行尾
ctrl + u    删除光标到行首的内容
```

Linux的环境变量

同学们应该都会配置windows下的环境变量（PATH），都知道系统会按照PATH的设定，去每个PATH定义的目录下搜索可执行文件。

那么如何查看Linux下的PATH环境变量呢？

```
执行命令：
echo $PATH
echo命令是有打印的意思
$符号后面跟上PATH,表示输出PATH的变量
```

PATH(一定是大写的)这个变量是由一堆目录组成，分隔符是":"号，而不同于windows的";"号。

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

绝对路径与相对路径

Linux中非常重要的概念--路径，路径用来定位如何找到某个文件。

这里超哥先讲个例子，到底什么是相对路径，绝对路径

比如一个老外，要来老男孩教育学习python，但是他找不到地点，因此向你问路，你可以告诉他：

1.先坐飞机来中国北京，从北京机场坐地铁到沙河地铁站，然后走路800米到沙河汇德商厦，上四楼，找到超哥，结束寻路。

2. 超哥就在汇德商厦403办公室，武佩奇后面坐着呢！！

Linux下特别注意文件名/路径的写法，可以将所谓的路径(path)定义为绝对路径(absolute)和相对路径(relative)。这两种文件名/路径的写法依据是这样的：

- 绝对路径：由根目录(/)为开始写起的文件名或者目录名称，如/home/oldboy/test.py;
- 相对路径：相对于目前路径的文件名写法。例如./home/oldboy/exam.py
或../home/oldboy/exam.py，简单来说只要开头不是/，就是属于相对路径

因此你必须了解，相对路径是：以你当前所在路径的相对路径来表示的。



例如你现在在/home 这个目录下，如要进入/var/log这个路径，如何写呢？

1. cd /var/log (绝对路径)
2. cd ../var/log(相对路径)

结果如图：

因为你在/home底下，因此你要回到上一层(..)之后，才能继续前往/var，特别注意：

- ..:代表当前的目录，也可以用./ 来表示
- ...:代表上一层的目录，也可以用../来表示

```
[[root@python_linux home]# pwd          打印当前位置
/home
[[root@python_linux home]# cd ../var/log  相对路径
[[root@python_linux log]# pwd            进入/var/log目录
/var/log
[[root@python_linux log]#
```



```
[root@python_linux home]# pwd
/home
[root@python_linux home]# cd /var/log
[root@python_linux log]# pwd
/var/log
```

使用绝对路径，切换目录

老师QQ: 877348180

Linux系统文件与启动流程

/etc初始化系统重要文件

- /etc/sysconfig/network-scripts/ifcfg-eth0:网卡配置文件
- /etc/resolv.conf:Linux系统DNS客户端配置文件
- /etc/hostname (CentOS7) /etc/sysconfig/network:(CentOS 6)主机名配置文件
- /etc/hosts:系统本地的DNS解析文件
- /etc/fstab:配置开机设备自动挂载的文件
- /etc/rc.local:存放开机自启动程序命令的文件
- /etc/inittab:系统启动设定运行级别等配置的文件
- /etc/profile及/etc/bashrc:配置系统的环境变量/别名等的文件
- /etc/profile.d:用户登录后执行的脚本所在的目录
- /etc/issue和/etc/issue.net:配置在用户登录终端前显示信息的文件
- /etc/init.d:软件启动程序所在的目录(centos 6)
- /usr/lib/systemd/system/ 软件启动程序所在的目录(centos 7)
- /etc/motd:配置用户登录系统之后显示提示内容的文件
- /etc/redhat-release:声明RedHat版本号和名称信息的文件
- /etc/sysctl.conf:Linux内核参数设置文件

/proc重要路径

/proc/meminfo:系统内存信息

/proc/cpuinfo:关于处理器的信息，如类型，厂家，型号，性能等



/proc/loadavg:系统负载信息，uptime 的结果

/proc/mounts:已加载的文件系统的列表

/var目录下文件

/var/log:记录系统及软件运行信息文件所在的目录

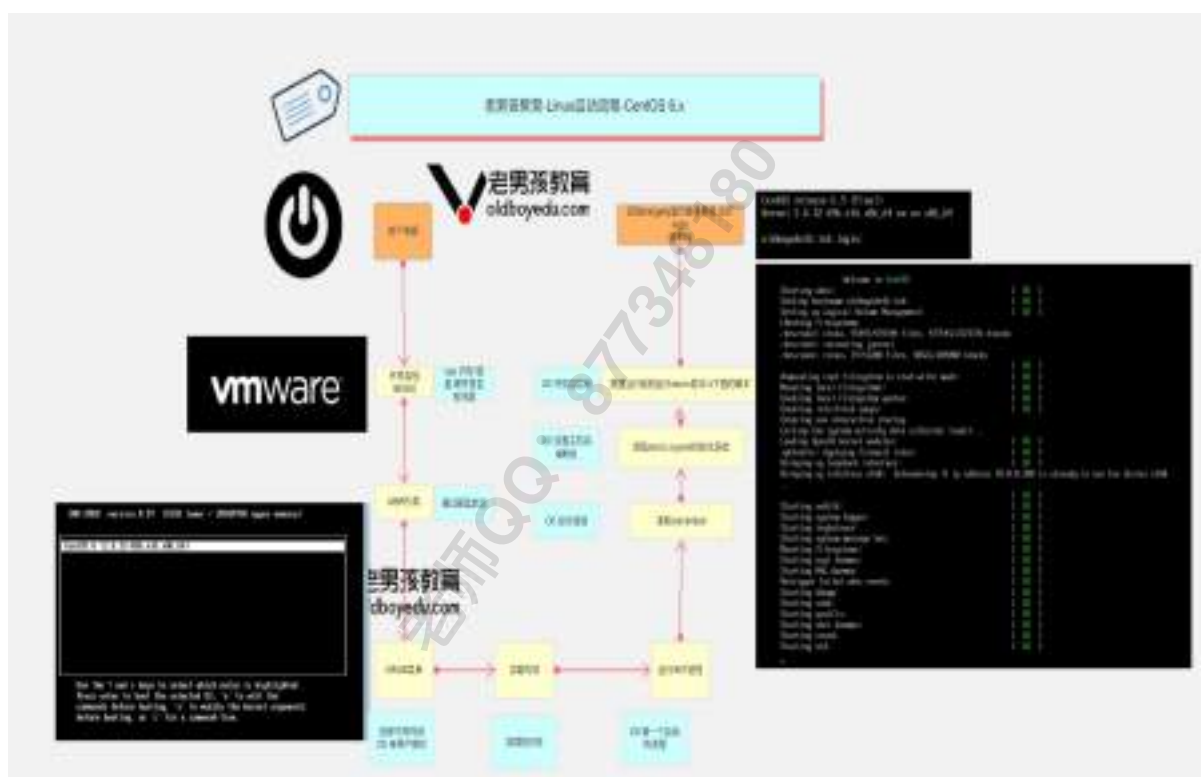
/var/log/messages:系统级别日志文件

/var/log/secure:用户登录信息日志文件

/var/log/dmesg:记录硬件信息加载情况的日志文件

Linux开机启动流程

作为一个运维人，必须得保障服务器正常工作，机器宕机了，也得明确是什么问题，从何查起，那么了解启动流程就能够对症下药，排查问题。



- BIOS自检

检查硬件是否健康。如 cpu 风扇是否正常，内存是否正常，时钟是否正常，这个过程是读取 ROM 上的指令执行的。

- 微控制器

系统想要启动必须先加载 BIOS，按下电源键时，给微控制器下达一条复位指令，各寄存器复位，最后下达一条跳转指令，跳转到 BIOS 的 ROM，使得硬件去读取主板上的 BIOS 程序，在这之前都是由硬件来完成，之后硬件就会把控制权交给 BIOS。

- BIOS->POST

随后 BIOS 程序加载 CMOS(可读写的 RAM 芯片, 保存 BIOS 设置硬件参数的数据)的信息, 借 CMOS 取得主机的各项硬件配置。取得硬件配置的信息之后, BIOS 进行加电自检(Power-on self Test, POST)过程,检测计算机各种硬件信息, 如果发现硬件错误则会报错(发出声音警告)。之后 BIOS 对硬件进行初始化。BIOS 将自己复制到物理内存中继续执行, 开始按顺序搜寻可引导存储设备, 决定存储设备的顺序(即定义第一个可引导的磁盘, 当然是在有两个磁盘的前提), 接下来就会读取磁盘的内容, 但是要读取磁盘文件必须要有文件系统, 这对 BIOS 挂载文件系统来说是不可能, 因此需要一个不依赖文件系统的方法使得 BIOS 读取磁盘内容, 这种方法就是引入 MBR。最后 BIOS 通过 INT13 硬件中断功能读取第一个可引导的存储设备的 MBR(0 磁道 0 扇区)中的 boot loader。将 MBR 加载到物理内存中执行。MBR 载入内存后, BIOS 将控制权转交给 MBR(准确的说应该是 MBR 中的 boot loader), 然后 MBR 接管任务开始执行。

- MBR引导

载入了第一个可引导的存储设备的 MBR 后, MBR 中的 boot loader 就要读取所在磁盘的操作系统核心文件(即后面所说的内核)了。但是不同操作系统的文件系统格式不同, 还有一个磁盘可以安装多个操作系统, 如何让 boot loader 做到引导的就是用户想要的操作系统, 这么多不同的功能单靠一个 446 字节的 boot loader 是远远不够的。必须有一个相对应的程序来处理各自对应的操作系统核心文件, 这个程序就是操作系统的 loader(注意不是 MBR 中的 boot loader), 这样一来 boot loader 只需要将控制权交给对应操作系统的 loader, 让它负责去启动操作系统就行了。一个硬盘的每个分区的第一个扇区叫做 boot sector, 这个扇区存放的就是操作系统的 loader, 所以常说一个分区只能安装一个操作系统。MBR 的 boot loader 有三个功能:提供选单, 读取内核文件, 转交给其它 loader。提供选单就是给用户提供一个选项单, 让用户选择进入哪个操作系统;读取内核文件的意思是, 系统会有一个默认启动的操作系统, 这个操作系统的 loader 在所在分区的 boot sector 有一份, 除此之外, 也会将这个默认启动的操作系统的 loader 复制一份到 MBR 的 boot loader 中, 这样一来 MBR 就会直接读取 boot loader 中的 loader 了, 然后就是启动默认的操作系统的 loader, 当用户选择其它操作系统启动的时候, boot loader 会将控制权转交给对应的 loader, 让它负责操作系统的启动。

- GRUB引导

grub 是 boot loader 中的一种, 就 grub 来说, 为了打破在 MBR 中只有 446Bytes 用于存放 boot loader这一限制, 所以这一步的实现是这样的:grub 是通过分成三个阶段来实现加载内核这一功能的, 这三个阶段分别是:stage1, stage1.5 以及 stage2。stage1:存放于 MBR 的前 446Bytes, 用于加载 stage1.5 阶段, 目的是为了识别并驱动 stage2(或者 /boot)所在分区的文件系统。stage1.5:存放于 MBR 之后的扇区, 加载 stage2 所在分区的文件系统驱动, 让 stage1 中的 boot loader 能识别 stage2 所在分区的文件系统。stage2:存放于磁盘分区之上, 具体存放于/boot/grub 目录之下, 主要用于加载内核文件(vmlinuz- VERSION-RELEASE)以及 ramdisk 这个临时根文件系统(initrd-VERSION-RELEASE.img 或 initramfs- VERSION-RELEASE.img)。概述:假如要启动的是硬盘设备, 首先硬件平台主板 BIOS 必须能够识别硬盘, 然后 BIOS 才能加载硬盘中的 boot loader, 而 boot loader 自身加载后就能够直接识别当前主机上的硬盘设备了;不过, 能够识别硬盘设备不代表能够识别硬盘设备中的文件系统, 因为文件系统是额外附加的一层软件组织的文件结构, 所以要对接一种文件系统, 就必须要有对应的能够识别和理解这种文件系统的驱动, 这种驱动就称为文件系统驱动。而 stage1.5 就是向 grub 提供文件系统驱动的, 这样 stage1 就能访问 stage2 及内核所在的分区(/boot)了。

- 加载内核

内核(Kernel)在得到系统控制权之后, 首先要进行自身初始化, 而初始化的主要作用是: 探测可识别到的所有硬件设备; 加载硬件驱动程序, 即加载真正的根文件系统所在设备的驱动程序(有可能会借助于 ramdisk 加载 驱动); 以只读方式挂载根文件系统(如果有借助于 ramdisk 这个临时文件系统(虚根), 则在这一步之后 会执行根切换;否则不执行根切换); 运行用户空间的第一个应用程序:/sbin/init。到这里内核空间的启动流程就结束了, 而接下来是用户空间完成后续的系统启动流程。注意:ramdisk 和内核是由 boot loader 一同加载到内存当中的, ramdisk 是用于实现系统初始化的、基于内存的磁盘设备, 即加载至内存(的某一段空间)后把内存当磁盘使用, 并在内存中作为临时 根文件系统提供给内核使用, 帮助内核挂载真正的根文件系统。而之所以能够帮助内核挂载根文件 系统是因为在 ramdisk 这个临时文件系统的/lib/modules 目录下有真正的根文件系统所在设备的驱动 程序;除此之外, 这个临时文件系统也遵循 FHS, 例如有这些固定目录结构:/bin, /sbin, /lib, /lib64, /etc, /mnt, /media, ... 因为 Linux 内核有一个特性就是通过使用缓冲/缓存来达到加速对磁盘上文件的访问的目的, 而 ramdisk 是加载到内存并模拟成磁盘来使用的, 所以 Linux 就会为内存中的“磁盘”再使用一层缓冲/缓存, 但是 ramdisk 本来就是内存, 它只不过被当成硬盘来使用罢了, 这就造成双缓冲/缓存了, 而且不会起到提速效果, 甚至影响了访问性能;CentOS 5 系列以及之前版本的 ramdisk 文件为 initrd- VERSION-RELEASE.img, 就会出现上述所说到的问题;而为了解决一问题, CentOS 6/7 系列版本就将其改为 initramfs-VERSION-RELEASE.img, 使用文件系统的方式就可以避免双缓冲/缓存了, 可以说这是一种提速机制。

- 启动init进程

grub 中默认指定 init=/sbin/init 程序, 可以在 grub.conf 中 kernel 行自定义执行程序 init=/bin/bash,此时 可以绕过下面步骤直接进入 bash 界面。内核源代码文件中显示 996 行左右, 规定了 init 启动的顺序, /sbin/init->/etc/init->/bin/init->/bin/sh。

- 读取/etc/inittab 文件

inittab 文件里面定义了系统默认运行级别, 这一步做了一些工作如下: 初始运行级别(RUN LEVEL); 系统初始化脚本; 对应运行级别的脚本目录; 定义 UPS 电源终端/恢复脚本; 在虚拟控制台生成 getty,以生成终端; 在运行级别 5 初始化 X。

- 执行/etc/rc.d/rc.sysinit 程序

系统初始化一些脚本, 主要完成以下工作。设置主机名; 设置欢迎信息;

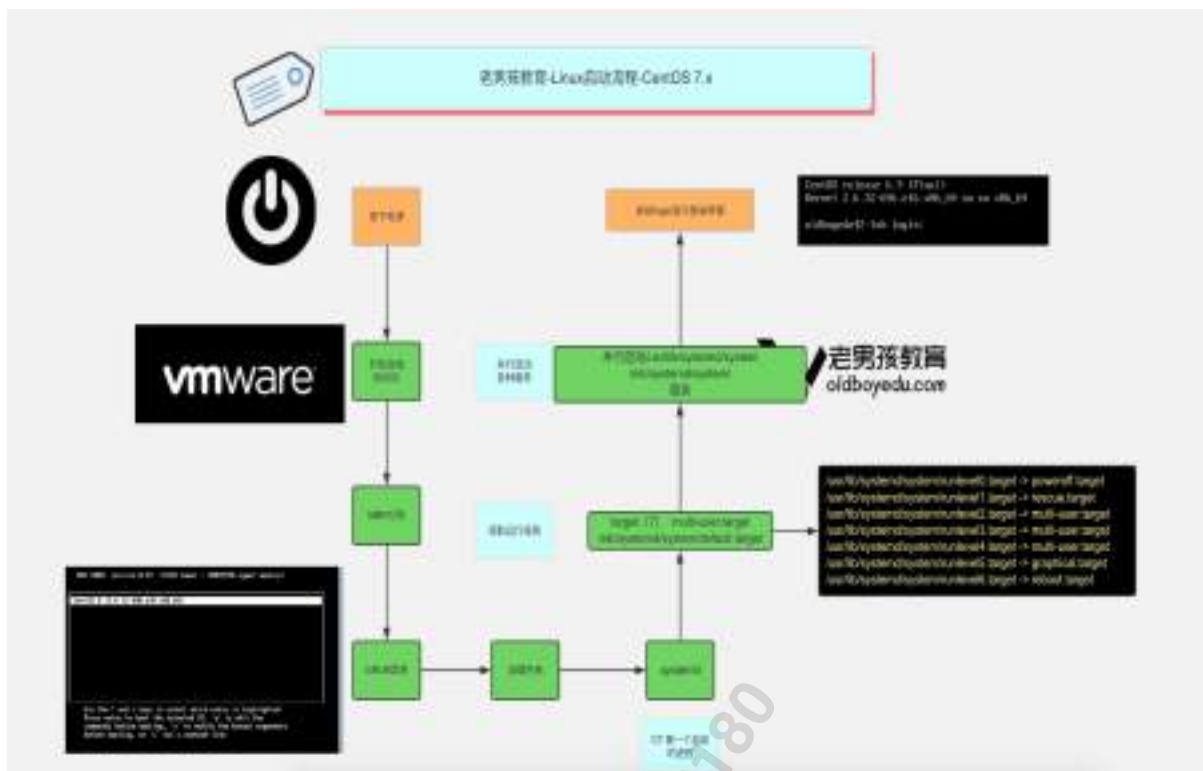
激活 udev 和 selinux 可以在 grub.conf 中,kernel 行添加 selinux=0 以关闭 selinux; 挂载/etc/fstab 文件中定义的文件系统; 检测根文件系统, 并以读写方式重新挂载根文件系统; 设置系统时钟;

激活 swap 设备; 根据/etc/sysctl.conf 文件设置内核参数; 激活 lvm 及 software raid 设备; 加载额外设备的驱动程序; 清理操作。/etc/rc*.d/文件(各种服务) 里面定义的是各种服务的启动脚本, 可以 ls 查看, S 开头代表开机启动的服务, K 开头的是关机要 执行的任务。#代表数字, 一个数字代表一个运行级别, 共 7 个运行级别。/etc/rc.d/rc.local 文件 这里面可以自定义开机启动的命令。

- 执行/bin/login

执行/bin/login 程序, 等待用户登录。

centos7启动流程



CentOS7 和 CentOS6 启动流程差不多，只不过到 init 程序时候，改为了 systemd，因此详细解释一下 systemd 后的启动流程。

- uefi或BIOS初始化，开始post开机自检;
- 加载mbr到内存
- 加载内核和initramfs模块
- 内核开始初始化，使用systemd代替centos6的init程序

1.执行initrd.target，包括挂载/etc/fstab文件中的系统，此时挂载后，就可以切换到根目录了

2.从initramfs根文件系统切换到磁盘根目录

3.systemd执行默认target配置

CentOS7 系表面是有“运行级别”这个概念，实际上是为了兼容以前的系统，每个所谓的“运行级别”都有对应的软连接指向，默认的启动级别是/etc/systemd/system/default.target，根据它的指向可以找到系统要进入哪个模式。

centos7的7个启动模式是：

- 0 ==> runlevel0.target, poweroff.target
- 1 ==> runlevel1.target, rescue.target
- 2 ==> runlevel2.target, multi-user.target
- 3 ==> runlevel3.target, multi-user.target
- 4 ==> runlevel4.target, multi-user.target
- 5 ==> runlevel5.target, graphical.target
- 6 ==> runlevel6.target, reboot.target
- systemd执行sysinit.target;
- systemd启动multi-user.target下的本机与服务器服务;

- systemd执行multi-user.target下的/etc/rc.d/rc.local。
- Systemd 执行 multi-user.target 下的 getty.target 及登录服务;
- systemd 执行 graphical 需要的服务。

老师QQ: 877348180

Linux文件目录管理命令

[TOC]

vim与程序员

所有的 Unix Like 系统都会内建 vi 文书编辑器，其他的文书编辑器则不一定会存在。

但是目前我们使用比较多的是 vim 编辑器。

vim 具有程序编辑的能力，可以主动的以字体颜色辨别语法的正确性，方便程序设计。

什么是vim

Vim是从 vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。

简单的来说，vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。vim 则可以说是程序开发者的一项很好用的工具。

vi/vim 的使用

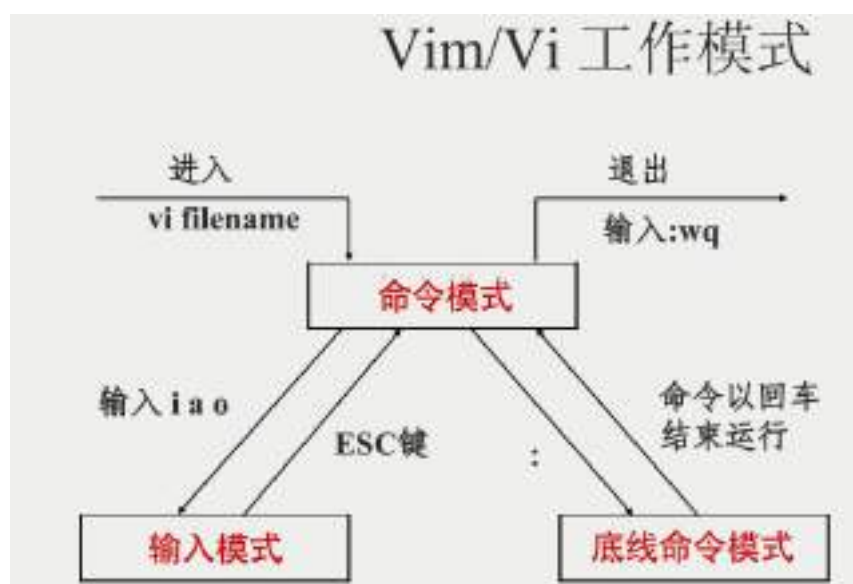
基本上 vi/vim 共分为三种模式，分别是**命令模式 (Command mode)**，**输入模式 (Insert mode)** 和 **底线命令模式 (Last line mode)**。这三种模式的作用分别是：

vim工作模式

命令模式:进入 vim 默认的模式

编辑模式:按 i 进入的 a o 也可以进入

底行模式: 按下:(冒号)之后进入到的模式



vim基础用法

- vi oldboy.txt #打开文件
- 打开后无法直接编辑，需要按 i 进入编辑模式
- 修改这个文件内容吧
- 修改完后，按 esc 退出编辑模式:wq
- 保存退出 #注意":"必须是英文符号

```
:wq 保存并退出
:q! 强制退出不保存
:wq! 强制保存退出
```

命令模式

用户刚刚启动 vi/vim，便进入了命令模式。

此状态下敲击键盘动作会被Vim识别为命令，而非输入字符。比如我们此时按下i，并不会输入一个字符，i被当作了一个命令



移动光标

w(e)	移动光标到下一个单词
b	移动到光标上一个单词
数字0	移动到本行开头
\$	移动光标到本行结尾
H	移动光标到屏幕首行
M	移动到光标到屏幕的中间一行
L	移动光标到屏幕的尾行
gg	移动光标到文档的首行
G	移动光标到文档尾行
ctrl + f	下一页
ctrl + b	上一页

` . 移动光标到上一次的修改行

查找

/chaoge	在整篇文档中搜索chaoge字符串, 向下查找
?chaoge	在整篇文档中搜索chaoge字符串, 向上查找
*	查找整个文档, 匹配光标所在的所有单词, 按下n查找下一处, N上一处
#	查找整个文档, 匹配光标所在的所有单词, 按下n查找下一处, N上一处
gd	找到光标所在单词匹配的单词, 并停留在非注释的第一个匹配上
%	找到括号的另一半!!

复制、删除、粘贴

yy	拷贝光标所在行
dd	删除光标所在行
D	删除当前光标到行尾的内容
dG	删除当前行到文档尾部的内容
p	粘贴yy所复制的内容
x	向后删除字符
X	向前删除字符
u	撤销上一步的操作
.	重复前一个执行过的动作

数字与命令

3yy	拷贝光标所在的3行
5dd	删除光标所在5行

快捷操作

删除光标所在位置到行尾的内容并进入编辑模式 C(大写字母)
在命令模式下按下字母i, 即可进入输入模式, 可以编写代码啦。。。
在当前行下面插入一行并进入编辑模式 o(小写字母)
在当前行上面插入一行并进入编辑模式 O(大写字母)
快速到达行尾并进入编辑模式 A
快速保存并退出 ZZ

批量快捷操作

批量删除:
进入批量编辑模式(可视块)
ctrl+v
选择 上下左右

删除 d

批量增加:进入批量编辑模式(可视块)ctrl+v

选择区域

输入大写的 I 进入编辑模式 编辑

按下ESC键

批量去掉注释

1. 进入命令行模式, 按ctrl + v进入 visual block模式, 按字母l横向选中列的个数, 例如 // 需要选中2列
2. 按字母j, 或者k选中注释符号
3. 按d键就可全部取消注释



vim批量缩进

```
:set tabstop=4  设定tab宽度为4个字符
:set shiftwidth=4  设定自动缩进为4个字符
:set expandtab  用space替代tab的输入
:set noexpandtab  不用space替代tab的输入
```

1. 命令模式, 按下v, 进入可视模式
2. 光标移动选择行, 输入 > 大于号, 缩进, 输入< 缩进

输入行号缩进:

1. 显示行号

```
:set nu      #显示
```

```
:set nonu    #关闭
```

2. 行号缩进

```
:10,20 >    #10到20行, 缩进一次
```

底线命令模式

在命令模式下输入冒号 (英文的:), 就进入了底线命令模式, 在底线命令模式下可以输入单个或多个字符的命令, 常用命令有:

```
:q!      强制退出
:wq!     强制写入退出
:set nu  显示行号
:数字    调到数字那行
:set nonu 取消显示行号
```

随时按下esc可以退出底线命令模式

vim执行流程与常见故障解析

swp文件



重定向符号

符号	解释
重定向的意思是，”将数据传到其他地方“	
< 或者<<	标准输入stdin， 代码为0
>或>>	标准输出stdout， 代码为1
2>或2>>	标准错误输出stderr， 代码为2

特殊符号

符号	解释

*	匹配任意个字符				
?	匹配一个字符				
\		管道符			
&	后台进程符				
&&	逻辑与符号，命令1 && 命令2，当命令1执行成功继续执行命令2				
\	\		逻辑或符号，命令1 \	\	命令2，当命令1执行失败才会执行命令2
#	注释符				
""	双引号表示字符串，能够识别，``反引号，\$符，\转义符				
''	单引号表示普通字符串，无特殊含义				
\$	变量符 如 \$name				
\	转义字符				

cat命令

cat命令用于查看纯文本文件（常用于内容较少的），可以理解为是 猫，瞄一眼文件内容
其单词是 concatenate，指的是可以连接多个文件且打印到屏幕，或是重定向到文件中



cat功能

功能	说明
查看文件内容	cat file.txt
多个文件合并	cat file.txt file2.txt > file3.tx
非交互式编辑或追加内容	cat >> file.txt << EOF 欢迎来到路飞学城 EOF
清空文件内容	cat /dev/null > file.txt 【/dev/null是linux系统的黑洞文件】

参数

用法: cat [选项] [文件]...

将[文件]或标准输入组合输出到标准输出。

清空文件内容, 慎用

> 文件名

-A, --show-all	等价于 -vET
-b, --number-nonblank	对非空输出行编号
-e	等价于 -vE
-E, --show-ends	在每行结束处显示 \$
-n, --number	对输出的所有行编号
-s, --squeeze-blank	不输出多行空行
-t	与 -vT 等价
-T, --show-tabs	将跳格字符显示为 ^I
-u	(被忽略)
-v, --show-nonprinting	使用 ^ 和 M- 引用, 除了 LFD 和 TAB 之外
--help	显示此帮助信息并退出
--version	输出版本信息并退出

如果[文件]缺省, 或者[文件]为 -, 则读取标准输入。

案例

#查看文件, 显示行号

```
cat -n xxx.py
```

#猫, 查看文件

```
cat xxx.py
```

#在每一行的结尾加上\$符

```
[root@master tmp]# cat -E 1.txt
```

#追加文字到文件

```
cat >>/tmp/oldboy.txt << EOF
```

唧唧复唧唧

木兰开飞机

开的什么机

波音747

EOF

```
[root@luffycity tmp]# cat -b file.txt
```

#输出非空行的编号

1 欢迎来到路飞学城

2 学生还有五分钟到达战场

3 全军出击

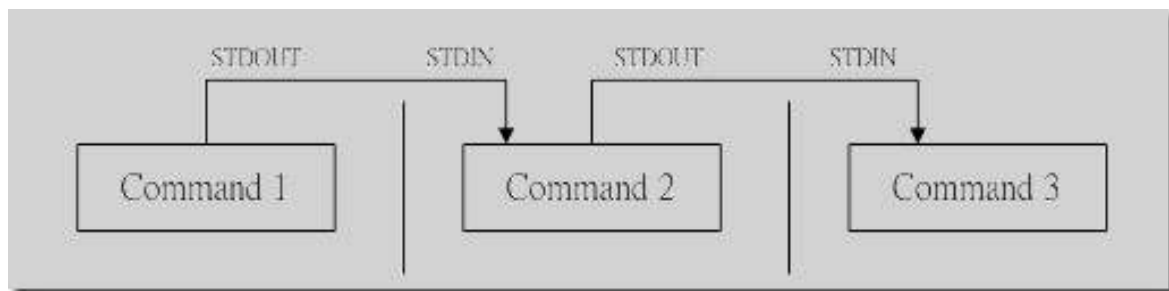
```
[root@luffycity tmp]#  
[root@luffycity tmp]#  
[root@luffycity tmp]#  
[root@luffycity tmp]# cat -n file.txt          #输出所有行的编号  
  1      欢迎来到路飞学城  
  2      学生还有五分钟到达战场  
  3  
  4  
  5      全军出击  
  
[root@luffycity tmp]# cat -E file.txt          #显示出每行的结束符，$符号  
欢迎来到路飞学城$  
学生还有五分钟到达战场$  
$  
$  
全军出击$  
  
[root@luffycity tmp]# cat -s file.txt          # -s参数把多个空行，换成一个，可以让文件更精  
炼阅读  
欢迎来到路飞学城  
学生还有五分钟到达战场  
  
全军出击  
  
[root@luffycity tmp]# cat > alex.txt          #cat写入内容，用ctrl+c 结束，一般不用  
我是金角老妖怪alex  
^C
```

tac命令

与cat命令作用相反，反向读取文件内容

```
[root@luffycity tmp]# cat alex.txt  
我是金角老妖怪alex  
我是老妖怪alex  
[root@luffycity tmp]#  
[root@luffycity tmp]# tac alex.txt  
我是老妖怪alex  
我是金角老妖怪alex
```

管道符



Linux提供的管道符“|”将两条命令隔开，管道符左边命令的输出会作为管道符右边命令的输入。

常见用法：

#检查python程序是否启动

```
ps -ef|grep "python"
```

#找到/tmp目录下所有txt文件

```
ls /tmp|grep '.txt'
```

#检查nginx的端口是否存活

```
netstat -tunlp |grep nginx
```

more命令

More是一个过滤器, 用于分页显示 (一次一屏) 文本, 以当前屏幕窗口尺寸为准

语法

```
more 参数 文件
```

-num 指定屏幕显示大小为num行

+num 从num行开始显示

交互式more的命令:

空格 向下滚动一屏

Enter 向下显示一行

= 显示当前行号

q 退出

案例

#显示5行内容

```
[root@luffycity tmp]# more -5 alex.txt
```

```
root:x:0:0:root:/root:/bin/bash
```

```
root:x:0:0:root:/root:/bin/bash
```

```
root:x:0:0:root:/root:/bin/bash
```

```
root:x:0:0:root:/root:/bin/bash
```

```
root:x:0:0:root:/root:/bin/bash
```

```
--More-- (0%)
```

#从6行开始输出内容到屏幕


```
more +6 alex.txt
```

#将显示结果分页输出，需控制窗口大小

```
[root@luffycity tmp]# netstat -tunlp |more -3
```

less命令

less命令是more的反义词

语法：

less 参数 文件

-N 显示每行编号

-e 到文件结尾自动退出，否则得手动输入q退出

子命令

整个的翻页

b 向前一页

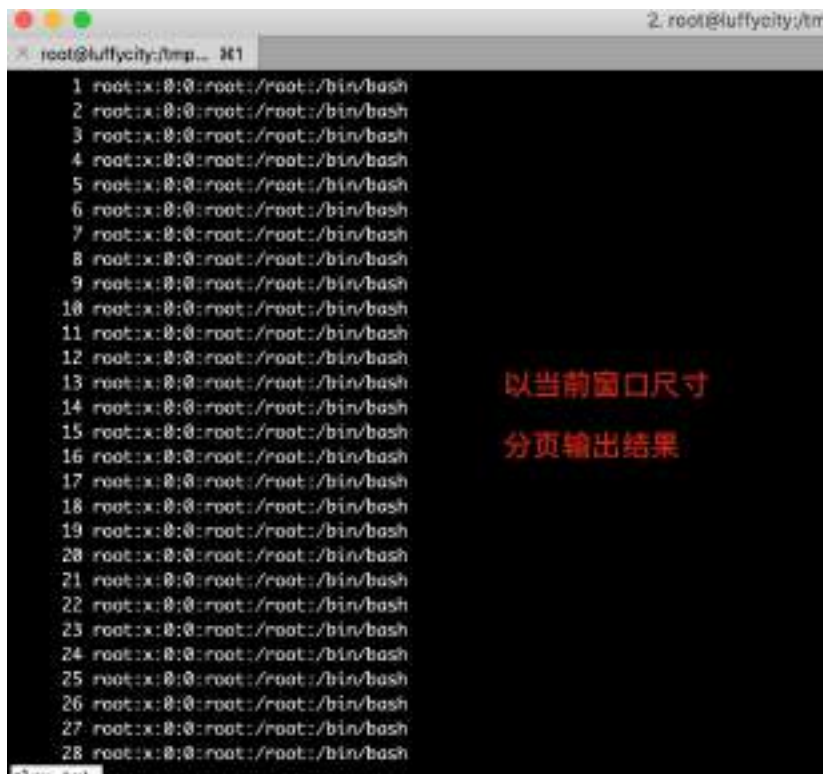
f 向后一页

空格 查看下一行，等于 ↓

y 查看上一行，等于 ↑

q退出

案例



```

1 root:x:0:0:root:/root:/bin/bash
2 root:x:0:0:root:/root:/bin/bash
3 root:x:0:0:root:/root:/bin/bash
4 root:x:0:0:root:/root:/bin/bash
5 root:x:0:0:root:/root:/bin/bash
6 root:x:0:0:root:/root:/bin/bash
7 root:x:0:0:root:/root:/bin/bash
8 root:x:0:0:root:/root:/bin/bash
9 root:x:0:0:root:/root:/bin/bash
10 root:x:0:0:root:/root:/bin/bash
11 root:x:0:0:root:/root:/bin/bash
12 root:x:0:0:root:/root:/bin/bash
13 root:x:0:0:root:/root:/bin/bash
14 root:x:0:0:root:/root:/bin/bash
15 root:x:0:0:root:/root:/bin/bash
16 root:x:0:0:root:/root:/bin/bash
17 root:x:0:0:root:/root:/bin/bash
18 root:x:0:0:root:/root:/bin/bash
19 root:x:0:0:root:/root:/bin/bash
20 root:x:0:0:root:/root:/bin/bash
21 root:x:0:0:root:/root:/bin/bash
22 root:x:0:0:root:/root:/bin/bash
23 root:x:0:0:root:/root:/bin/bash
24 root:x:0:0:root:/root:/bin/bash
25 root:x:0:0:root:/root:/bin/bash
26 root:x:0:0:root:/root:/bin/bash
27 root:x:0:0:root:/root:/bin/bash
28 root:x:0:0:root:/root:/bin/bash

```

以当前窗口尺寸
分页输出结果

head命令

用于显示文件内容头部，默认显示开头10行

用法：head [选项]... [文件]...

将每个指定文件的头10 行显示到标准输出。

如果指定了多于一个文件，在每一段输出前会给出文件名作为文件头。

如果不指定文件，或者文件为"-"，则从标准输入读取数据。

- | | |
|-----------------------|---|
| -c, --bytes=[-]K | 显示每个文件的前K 字节内容；
如果附加"-"参数，则除了每个文件的最后K字节数据外
显示剩余全部内容 |
| -n, --lines=[-]K | 显示每个文件的前K 行内容；
如果附加"-"参数，则除了每个文件的最后K 行外显示
剩余全部内容 |
| -q, --quiet, --silent | 不显示包含给定文件名的文件头 |
| -v, --verbose | 总是显示包含给定文件名的文件头 |
| --help | 显示此帮助信息并退出 |
| --version | 显示版本信息并退出 |

案例

```

[root@luffycity ~]# head -5 /etc/passwd          #显示前五
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin

```

```
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
[root@luffycity ~]# head -c 6 /etc/passwd          #显示文件前6个字节
root:x[root@luffycity ~]#
```

#显示多个文件

```
[root@luffycity tmp]# echo 你就是金角大王吧 > alex.txt
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# echo 你就是银角大王吧 > peiqi.txt
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# head alex.txt  peiqi.txt
==> alex.txt <==
你就是金角大王吧

==> peiqi.txt <==
你就是银角大王吧
```

tail命令

显示文件内容的末尾，默认输出后10行

-c 数字	指定显示的字节数
-n 行数	显示指定的行数
-f	实时刷新文件变化
-F 等于 -f --retry	不断打开文件，与-f合用
--pid=进程号	进程结束后自动退出tail命令
-s 秒数	检测文件变化的间隔秒数

案例

```
#显示文件后10行
tail alex.txt
```

```
#显示文件后5行
tail -5 alex.txt
```

```
#从文件第3行开始显示文件
tail -n +3 alex.txt
```

```
#检测文件变化
tail -f alex.txt
```

-F与-f参数的用法

```
[root@luffycity tmp]# tail -f alex.txt          # -f文件不存在，直接报错，退出
```

```
tail: cannot open 'alex.txt' for reading: No such file or directory
tail: no files remaining
```

```
[root@luffycity tmp]# tail -F alex.txt          # -F    文件不存在报错，等待文件生成
tail: cannot open 'alex.txt' for reading: No such file or directory
```

```
tail: 'alex.txt' has appeared; following end of new file
qwe
```



cut命令

cut - 在文件的每一行中提取片断

在每个文件FILE的各行中，把提取的片断显示在标准输出。

语法

cut 参数 文件

-b	以字节为单位分割
-n	取消分割多字节字符，与-b一起用
-c	以字符为单位
-d	自定义分隔符，默认以tab为分隔符
-f	与-d一起使用，指定显示哪个区域
N	第 N 个 字节，字符 或 字段，从 1 计数 起
N-	从 第 N 个 字节，字符 或 字段 直至 行尾
N-M	从 第 N 到 第 M (并包括 第M) 个 字节，字符 或 字段
-M	从 第 1 到 第 M (并包括 第M) 个 字节，字符 或 字段

案例

以字节作为分隔符 -b参数

```
[root@luffycity tmp]# cat alex.txt
My name is alex and i like da xi gua

#切割出第四个字符
[root@luffycity tmp]# cut -b 4 alex.txt
n
```

```
[root@luffycity tmp]# cut -b 4-6 alex.txt      #输出第4到6的字符
nam
[root@luffycity tmp]# cut -b 4,6 alex.txt      #输出第4和6的字符
nm

[root@luffycity tmp]# cut -b -5 alex.txt       #输出第一个到第五个的字符
My na

[root@luffycity tmp]# cut -b 5- alex.txt       #输出第五个字符到结尾的字符
ame is alex and i like da xi gua
```

以字符作为分隔符 `-c` 参数，区别在于中英文

```
[root@luffycity tmp]# cut -b 3-7 alex.txt      #字节切割
name
[root@luffycity tmp]#
[root@luffycity tmp]# cut -c 3-7 alex.txt       #字符切割，没有区别，因为1个英文字母是1
个字节存储
name
[root@luffycity tmp]# cat alex.txt
My name is alex and i like da xi gua

#注意点如下
[root@pylinux tmp]# cat peiqi.txt
我是小猪佩奇
i am xiao zhu pei qi

[root@pylinux tmp]# cut -c 6- peiqi.txt         #从第六个字符开始切割到结尾
奇                                             #由于当前机器是utf-8编码，一个汉字等于一个字符
xiao zhu pei qi                             #英文字符正常
[root@pylinux tmp]# cut -b 6- peiqi.txt        #从第六个字节开始切割到结尾
小猪佩奇                                     #乱码，-b以二进制字节计算
xiao zhu pei qi

#加上-n参数，取消切割多字节的字符
[root@pylinux tmp]# cut -nc 6- peiqi.txt
奇
xiao zhu pei qi
[root@pylinux tmp]# cut -nb 6- peiqi.txt
奇
```

自定义分隔符

```
cut -f7 -d : /etc/passwd | head -5           #以冒号切割，显示第七区域信息

[root@pylinux tmp]# cut -f6-7 -d : /etc/passwd | head -5      #以冒号切割，显示第6-7
的区域信息
/root:/bin/bash
/bin:/sbin/nologin
```

```
/sbin:/sbin/nologin
/var/adm:/sbin/nologin
/var/spool/lpd:/sbin/nologin
```

```
[root@pylinux tmp]# head -5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[root@pylinux tmp]#
[root@pylinux tmp]# cut -d : -f 7 /etc/passwd | head -5
/bin/bash
/sbin/nologin
/sbin/nologin
/sbin/nologin
/sbin/nologin
```

以冒号切割
取出第7个区域的结果
显示前五内容

sort命令

sort命令将输入的文件内容按照规则排序，然后输出结果

用法: sort [选项]... [文件]...
或: sort [选项]... --files0-from=F
串联排序所有指定文件并将结果写到标准输出。

-b, --ignore-leading-blanks	忽略前导的空白区域
-n, --numeric-sort	根据字符串数值比较
-r, --reverse	逆序输出排序结果
-u, --unique	配合-c, 严格校验排序; 不配合-c, 则只输出一次排序结果
-t, --field-separator=分隔符	使用指定的分隔符代替非空格到空格的转换
-k, --key=位置1[,位置2]	在位置1 开始一个key, 在位置2 终止(默认为行尾)

案例

#sort 是默认以第一个数据来排序，而且默认是以字符串形式来排序，所以由字母 a 开始升序排序

```
[root@luffycity tmp]# cat /etc/passwd | sort
```

```
[root@luffycity tmp]# sort -n sort.txt          #按照数字从大到小排序
```

```
[root@luffycity tmp]# sort -nr sort.txt         #降序排序
```

```
[root@luffycity tmp]# sort -u sort.txt          #去重排序
```

```
[root@luffycity tmp]# sort -t " " -k 2 sort.txt      #指定分隔符，指定序列
```

```
10.0.0.15 a
```

```
10.0.0.12 e
```

```
10.0.0.22 e
10.0.0.54 f
10.0.0.34 q
10.0.0.63 q
10.0.0.3 r
10.0.0.34 r
10.0.0.4 v
10.0.0.44 w
10.0.0.5 x
```

```
[root@luffycity tmp]# cat /etc/passwd | sort -t ":" -k 3      #以分号分割，对第三列排序，
以第一位数字排序
```

#以分号分割，对第一个区域的第2到3个字符排序

```
[root@luffycity tmp]# cat /etc/passwd | sort -t ":" -k 1.2,1.3
```

uniq命令

uniq命令可以输出或者忽略文件中的重复行，常与sort排序结合使用

用法: `uniq [选项]... [文件]`

从输入文件或者标准输入中筛选相邻的匹配行并写入到输出文件或标准输出。

不附加任何选项时匹配行将在首次出现处被合并。

<code>-c, --count</code>	在每行前加上表示相应行目出现次数的前缀编号
<code>-d, --repeated</code>	只输出重复的行
<code>-u, --unique</code>	只显示出现过一次的行,注意了, uniq的只出现过一次,是针对-c统计之后的结果

案例

```
#测试数据文件
[root@luffycity tmp]# cat luffy.txt
10.0.0.1
10.0.0.1
10.0.0.51
10.0.0.51
10.0.0.1
10.0.0.1
10.0.0.51
10.0.0.31
10.0.0.21
10.0.0.2
10.0.0.12
10.0.0.2
10.0.0.5
10.0.0.5
10.0.0.5
```

```
10.0.0.5
```

```
[root@luffycity tmp]# uniq luffy.txt
```

#仅仅在首次出现的时候合并，最好是排序后去重

```
10.0.0.1
10.0.0.51
10.0.0.1
10.0.0.51
10.0.0.31
10.0.0.21
10.0.0.2
10.0.0.12
10.0.0.2
10.0.0.5
```

```
[root@luffycity tmp]# sort luffy.txt |uniq -c
```

#排序后去重且显示重复次数

```
4 10.0.0.1
1 10.0.0.12
2 10.0.0.2
1 10.0.0.21
1 10.0.0.31
4 10.0.0.5
3 10.0.0.51
```

```
[root@luffycity tmp]# sort luffy.txt |uniq -c -d
```

#找出重复的行，且计算重复次数

```
4 10.0.0.1
2 10.0.0.2
4 10.0.0.5
3 10.0.0.51
```

```
[root@luffycity tmp]# sort luffy.txt |uniq -c -u
```

#找到只出现一次的行

```
1 10.0.0.12
1 10.0.0.21
1 10.0.0.31
```

wc命令

wc命令用于统计文件的行数、单词、字节数

```
-c, --bytes 打印字节数
-m, --chars 打印字符数
-l, --lines 打印行数
-L, --max-line-length 打印最长行的长度
-w, --words 打印单词数
```


案例

```
[root@luffycity tmp]# wc -l luffy.txt           #统计文本有多少行，如同cat -n 看到的行数
21 luffy.txt

#统计单词数量，以空格区分
[root@luffycity tmp]# echo "alex peiqi yuchao mjj cunzhang" | wc -w
5

[root@luffycity tmp]# echo "alex" |wc -m        #统计字符数，由于结尾有个$
5

[root@luffycity tmp]# echo "alex" |cat -E       #证明结尾有个$
alex$

[root@luffycity tmp]# wc -L alex.qq            #统计最长的行，字符数
9 alex.qq

[root@luffycity tmp]# who|wc -l                #当前机器有几个登录客户端
```

tr命令

tr命令从标准输入中替换、缩减或删除字符，将结果写入到标准输出

用法：tr [选项]... SET1 [SET2]

从标准输入中替换、缩减和/或删除字符，并将结果写到标准输出。

字符集1：指定要转换或删除的原字符集。

当执行转换操作时，必须使用参数“字符集2”指定转换的目标字符集。

但执行删除操作时，不需要参数“字符集2”；

字符集2：指定要转换成的目标字符集。

-c或—complement：取代所有不属于第一字符集的字符；

-d或—delete：删除所有属于第一字符集的字符；

-s或--squeeze-repeats：把连续重复的字符以单独一个字符表示；

-t或--truncate-set1：先删除第一字符集较第二字符集多出的字符。

案例

#将输入字符由小写换为大写：

```
[root@luffycity ~]# echo "My name is alex" | tr 'a-z' 'A-Z'
MY NAME IS ALEX
```

#tr删除字符或数字，只要匹配上属于第一个字符串的字符，都被删掉

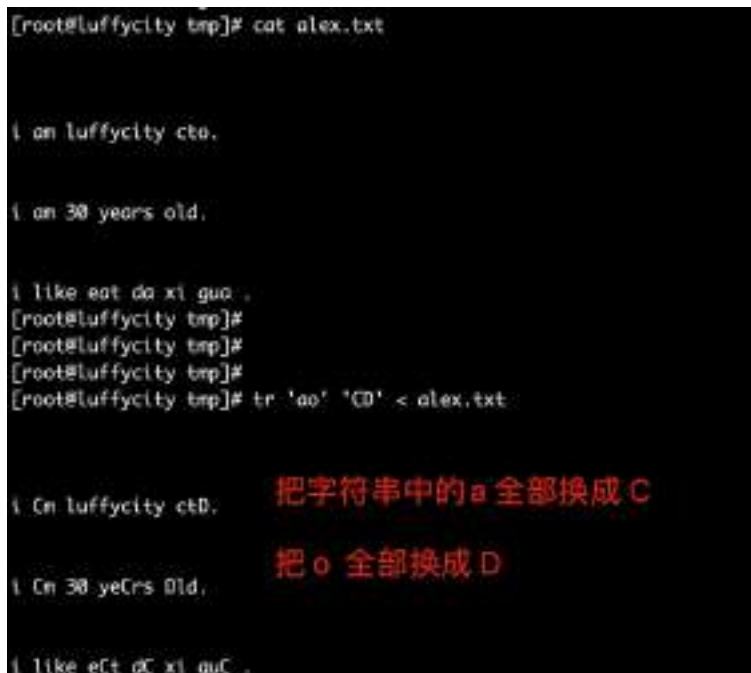
```
[root@luffycity ~]# echo "My name is alex and i am 30 years old." | tr -d "0-9"
```

My name is alex and i am 33456 years old.

```
[root@luffycity ~]# echo "My name is alex and i am 33456 years old." | tr -d "1234"
My name is alex and i am 56 years old.
```

#删除字符，所有的数字，以及小写字符

```
[root@luffycity ~]# echo "My name is alex and i am 33456 years old." | tr -d "0-9",
"a-z"
```



```
[root@luffycity tmp]# tr "[a-z]" "[A-Z]" < alex.txt
I AM LUFFYCITY CTO.
I AM 30 YEARS OLD.
I LIKE EAT DA XI GUA .
```

#全部换成大写

#删除文中出现的换行符、制表符（tab键）

```
tr -d "\n\t" < alex.txt
```

#去重连续的字符，tr是挨个匹配"ia" 每一个字符，包括空格去重

```
[root@luffycity tmp]# echo "iiii am aaaaalex,iiii like hot girl" | tr -s "ia"
i am alex,i like hot girl
```

#-c取反结果，将所有除了'a'以外的全部替换为'A'

```
[root@luffycity tmp]# echo 'i am alex' | tr -c 'a' 'A'
AAaAAaAAAA
```

stat命令

stat命令用于显示文件的状态信息。**stat命令**的输出信息比**ls命令**的输出信息要更详细。

语法

```
stat(选项)(参数)
```

选项

-L, --dereference	跟随链接
-f, --file-system	显示文件系统状态而非文件状态
-c --format=格式	使用指定输出格式代替默认值，每用一次指定格式换一新行
--printf=格式	类似 --format，但是会解释反斜杠转义符，不使用换行作输出结尾。如果您仍希望使用换行，可以在格式中加入"\n"
-t, --terse	使用简洁格式输出
--help	显示此帮助信息并退出
--version	显示版本信息并退出

有效的文件格式序列(不使用 --file-system):

%a 八进制权限

参数

文件：指定要显示信息的普通文件或者文件系统对应的设备文件名。

```
[root@www ~]#stat abc.ph
 文件: "abc.ph"
 大小: 0                块: 0                IO 块: 4096    普通空文件
设备: 801h/2049d        Inode: 1200314        硬链接: 1
权限: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
最近访问: 2013-05-14 13:24:30.830729223 +0800
最近更改: 2013-05-14 13:24:30.830729223 +0800
最近改动: 2013-05-14 13:24:30.830729223 +0800
创建时间: -

[root@luffycity tmp]# stat test.txt
File: 'test.txt'
Size: 16                Blocks: 8                IO Block: 4096    regular file
Device: fd00h/64768d    Inode: 17540200        Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2019-10-18 14:58:59.465647961 +0800
Modify: 2019-10-18 14:58:57.799636638 +0800
Change: 2019-10-18 14:58:57.799636638 +0800
Birth: -
```

#显示文件权限

```
[root@pylinux test_find]# stat -c %a alex.txt
644
```

stat的时间戳

```
Access: 2019-10-18 14:58:59.465647961 +0800
Modify: 2019-10-18 14:58:57.799636638 +0800
Change: 2019-10-18 14:58:57.799636638 +0800
```

access、最近访问，文件每次被cat之后，时间变化，由于操作系统特性，做了优化，频繁访问，时间不变
modify、最近更改，更改文件内容，vim等
change、最近改动，文件元数据改变，如文件名

find命令

find命令用来在指定目录下查找文件。任何位于参数之前的字符串都将被视为欲查找的目录名。

如果使用该命令时，不设置任何参数，则find命令将在当前目录下查找子目录与文件。

并且将查找到的子目录和文件全部进行显示。

```
[root@luffycity tmp]# find --help
Usage: find [-H] [-L] [-P] [-Olevel] [-D help|tree|search|stat|rates|opt|exec] [path...] [expression]

find 处理符号链接 要查找的路径 参数 限定条件 执行动作
find -H -L -P PATH options tests actions
```

语法

find 查找目录和文件，语法：

find 路径 -命令参数 [输出形式]

参数说明：

路径：告诉find在哪儿去找你要的东西，

参数	解释
pathname	要查找的路径
options选项	
-maxdepth	<目录层级>：设置最大目录层级；
-mindepth	<目录层级>：设置最小目录层级；
tests模块	
-atime	按照文件访问access的时间查找，单位是天
	按照文件的改变change状态来查找文件，单位是天

	按照文件的改变change状态来查找文件，单位是天
-mtime	根据文件修改modify时间查找文件 【最常用】
-name	按照文件名字查找，支持* ? [] 通配符
-group	按照文件的所属组查找
-perm	按照文件的权限查找
-size n[cwbkMG]	按照文件的大小 为 n 个由后缀决定的数据块。 其中后缀为： b: 代表 512 位元组的区块（如果用户没有指定后缀，则默认为 b） c: 表示字节数 k: 表示 kilo bytes （1024字节） w: 字 （2字节） M:兆字节（1048576字节） G: 千兆字节 （1073741824字节）
-type 查找某一类型的文件	b - 块设备文件。 d - 目录。 c - 字符设备文件。 p - 管道文件。 l - 符号链接文件。 f - 普通文件。 s - socket文件
-user	按照文件属主来查找文件。
-path	配合-prune参数排除指定目录
Actions模块	
-prune	使find命令不在指定的目录寻找
-delete	删除找出的文件
-exec 或-ok	对匹配的文件执行相应shell命令
-print	将匹配的结果标准输出
OPERATORS	
!	取反
-a -o	取交集、并集，作用类似&&和\

案例

根据名字查找

```
[root@luffycity tmp]# ls
alex.txt
[root@luffycity tmp]# find . -name "alex.txt" -delete      #找出名为alex.txt且删除
[root@luffycity tmp]# ls      #已经找不到

[root@luffycity tmp]# touch python{1..10}.pid
[root@luffycity tmp]# ls
python1.pid python10.pid python2.pid python3.pid python4.pid python5.pid pyth
```

```

on6.pid python7.pid python8.pid python9.pid
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# find . -name "*.pid"          #找出所有的pid
./python1.pid
./python2.pid
....

[root@luffycity tmp]# find . -name "[0-9]*.pid"      #找到所以以数字开头的pid文件
./123a.pid
./123b.pid
.....

```

UNIX/Linux文件系统每个文件都有三种时间戳：

- 访问时间 (-atime/天, -amin/分钟)：用户最近一次访问时间（文件修改了，还未被读取过，则不变）。
- 修改时间 (-mtime/天, -mmin/分钟)：文件最后一次修改时间（数据变动）。
- 变化时间 (-ctime/天, -cmin/分钟)：文件数据元（例如权限等）最后一次修改时间。

```

[root@luffycity tmp]# stat 123p.pid
File: '123p.pid'
Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d Inode: 17540175 Links: 1
Access: (0044/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2019-10-16 10:55:08.051567087 +0800
Modify: 2019-10-16 10:55:08.051567087 +0800
Change: 2019-10-16 10:55:08.051567087 +0800
Birth: -

```

访问时间
修改时间
变化时间

- 文件任何数据改变，change变化，无论是元数据变动，或是对文件mv，cp等
- 文件内容被修改时，modify和change更新
- 当change更新后，第一次访问该文件（cat，less等），access time首次会更新，之后则不会

```

touch -a : 仅更新Access time (同时更新Change为current time)
touch -m : 仅更新Modify time (同时更新Change为current time)
touch -c : 不创建新文件
touch -t : 使用指定的时间更新时间戳 (仅更改Access time与Modify time, Change time更新为current time)

```

find根据修改时间查找文件

```

#一天以内，被访问access过的文件
find . -atime -1

#一天以内，内容变化的文件
find . -mtime -1

#恰好在7天内被访问过的文件
[root@pylinux home]# find / -maxdepth 3 -type f -atime 7

```

时间说明

- -atime -2 搜索在2天内被访问过的文件
- -atime 2 搜索恰好在2天前被访问过的文件
- -atime +2 超过2天内被访问的文件

find反向查找

```
[root@pylinux opt]# find . -maxdepth 1 -type d      #在opt目录下 查找最大目录深度为1 文  
件夹类型的数据
```

```
[root@pylinux opt]# find . -maxdepth 1 ! -type d    # 加上感叹号，后面接条件，代表取除了  
文件夹以外类型
```

根据权限查找

```
[root@pylinux opt]# find . -maxdepth 2 -perm 755 -type f  #寻找权限类型是755的文件
```

按照文件大小查

```
[root@pylinux opt]# du -h `find . -maxdepth 2 -size +10M`      #找出超过10M大小的文  
件  
14M    ./Python-3.7.3/python  
24M    ./Python-3.7.3/libpython3.7m.a  
322M   ./s21-centos-vim.tar.gz
```

查找文件时忽略目录

```
[root@pylinux s18tngx]# tree
.
├── conf
│   ├── fastcgi.conf
│   ├── fastcgi.conf.default
│   ├── fastcgi_params
│   ├── fastcgi_params.default
│   ├── koi-utf
│   ├── koi-win
│   ├── mime.types
│   ├── mime.types.default
│   ├── nginx.conf
│   ├── nginx.conf.default
│   ├── scgi_params
│   ├── scgi_params.default
│   ├── uwsgi_params
│   ├── uwsgi_params.default
│   └── win-utf
├── conf.d
│   ├── nginx.conf
│   └── nginx.conf.default
├── html
│   ├── 50x.html
│   └── index.html
├── logs
├── sbin
│   └── nginx
└──
```

5 directories, 20 files

```
[root@pylinux s18tngx]# find . -path "./conf.d" -prune -o -name "*.conf" -print
./conf/fastcgi.conf
./conf/nginx.conf
```

跳过这个文件夹 输出名字叫做*.conf的文件

```
[root@pylinux s18tngx]# find . -path "./conf.d" -prune -o -name "*.conf" -print
```

根据用户组匹配

```
[root@pylinux home]# find / -maxdepth 3 -group yu          #全局搜索深度为3，用户组是yu的文件
/home/yu
/home/yu/.bashrc
/home/yu/.bash_profile
/home/yu/.bash_history
/home/yu/.cache
/home/yu/.bash_logout
/home/yu/.config
```

使用-exec或是-ok再次处理

-ok比-exec更安全，存在用户提示确认

```
#找出以.txt结尾的文件后执行删除动作且确认
[root@pylinux opt]# find /opt/luffy_boy -type f -name "*.txt" -ok rm {} \;
```

备注

-exec 跟着shell命令，结尾必须以;分号结束，考虑系统差异，加上转义符\;
{ }作用是替代find查阅到的结果

{ }前后得有空格

```
#找到目录中所有的.txt文件，且将查询结果写入到all.txt文件中
[root@pylinux opt]# find ./mydj2/ -type f -name "*.txt" -exec cat {} \; > all.txt
```

```
#把30天以前的日志，移动到old文件夹中
find . -type f -mtime +30 -name "*.log" -exec cp {} old \;
```

xargs命令

xargs 又称管道命令，构造参数等。

是给命令传递参数的一个过滤器,也是组合多个命令的一个工具它把一个数据流分割为一些足够小的块,以方便过滤器和命令进行处理。

简单的说就是 把其他命令的给它的数据，传递给它后面的命令作为参数

```
-d 为输入指定一个定制的分割符，默认分隔符是空格
-i 用 {} 代替 传递的数据
-I string 用string来代替传递的数据 -n[数字] 设置每次传递几行数据
-n 选项限制单个命令行的参数个数
-t 显示执行详情
-p 交互模式
-P n 允许的最大线程数量为n
-s[大小] 设置传递参数的最大字节数(小于131072字节)
-x 大于 -s 设置的最大长度结束 xargs命令执行
-0, --null项用null分隔，而不是空白，禁用引号和反斜杠处理
```

案例

多行输入变单行

```
[root@luffycity tmp]# cat mjj.txt
1 2 3 4
5 6 7 8
9 10
[root@luffycity tmp]# xargs < mjj.txt
1 2 3 4 5 6 7 8 9 10
```

-n参数限制每行输出个数

```
[root@luffycity tmp]# xargs -n 3 < mjj.txt          #每行最多输出3个
1 2 3
4 5 6
7 8 9
10
```

自定义分隔符-d参数

```
[root@luffycity tmp]# echo "alex,alex,alex,alex,alex," |xargs -d ","
alex alex alex alex alex

#定义分隔符后，限制每行参数个数
[root@luffycity tmp]# echo "alex,alex,alex,alex,alex," |xargs -d "," -n 2
alex alex
alex alex
alex
```

-i参数的用法，用{}替换传递的数据

-I 参数用法，用string代替数据

```
#找到当前目录所有的.txt文件，然后拷贝到其他目录下
[root@luffycity tmp]# find . -name "*.txt" |xargs -i cp {} heihei/

[root@luffycity tmp]# find . -name "*.txt" |xargs -I data cp data heihei/

#找到当前目录下所有txt文件，然后删除
[root@luffycity tmp]# find . -name "*.txt" |xargs -i rm -rf {}
```

重点

xargs识别字符串的标识是空格或是换行符，因此如果遇见文件名有空格或是换行符，xargs就会识别为两个字符串，就会报错

- -print0在find中表示每一个结果之后加一个NULL字符，而不是换行符（find默认在结果后加上\n，因此结果是换行输出的）
- Xargs -0 表示xargs用NULL作为分隔符

```
[root@luffycity tmp]# touch hello\ luffycity.txt
[root@luffycity tmp]# ls
data data2 heihei hello luffycity.txt
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# ls -l
total 0
drwxr-xr-x. 2 root root 6 Oct 16 16:19 data
drwxr-xr-x. 2 root root 6 Oct 16 16:19 data2
drwxr-xr-x. 2 root root 6 Oct 16 16:19 heihei
-rw-r--r--. 1 root root 0 Oct 16 16:29 hello luffycity.txt
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# find . -name "*.txt"
./hello luffycity.txt
[root@luffycity tmp]# find . -name "*.txt" | xargs rm
rm: cannot remove './hello': No such file or directory
rm: cannot remove 'luffycity.txt': No such file or directory
```

利用反斜线转义空格，创建携带空格的文件

能够正常查询出

xargs 误认文件中间是空格，报错文件不存在

```
#修改find的输出结果，-print0可以改结尾为null
[root@luffycity tmp]# find . -name "*.txt" -print0
```

```
./hello luffycity.txt  
[root@luffycity tmp]# find . -name "*.txt" -print0  
./hello luffycity.txt[root@luffycity tmp]#
```

#修改xargs, 理解默认分隔符是NULL

```
find . -name "*.txt" -print0 |xargs -0 rm
```

老师QQ: 877348180

[TOC]

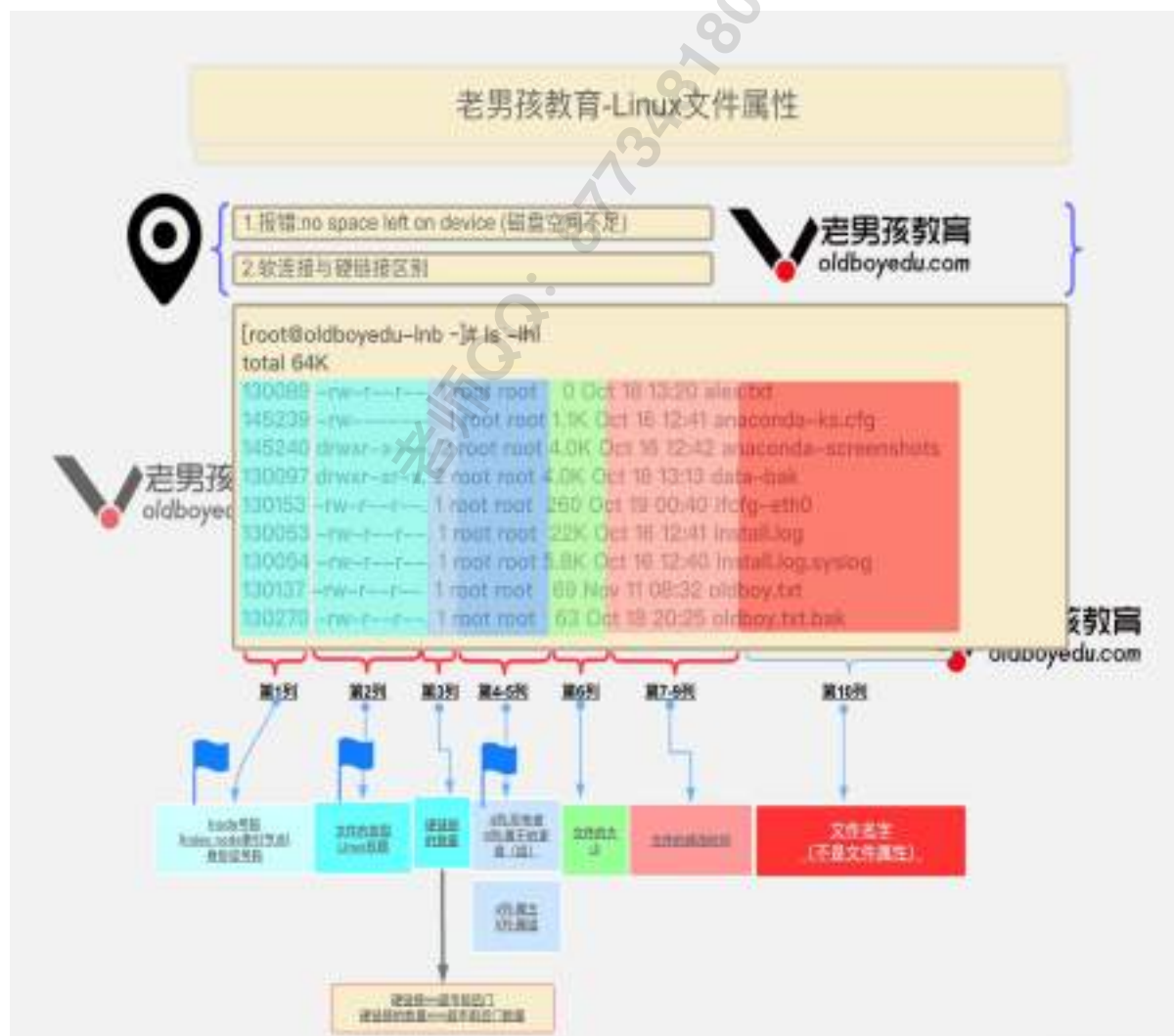
Linux文件属性与管理

文件或目录属性主要包括：

- 索引节点，inode
- 文件类型
- 文件权限
- 硬链接个数
- 归属的用户和用户组
- 最新修改时间

查看命令

```
ls -lhi /opt
```



图解：

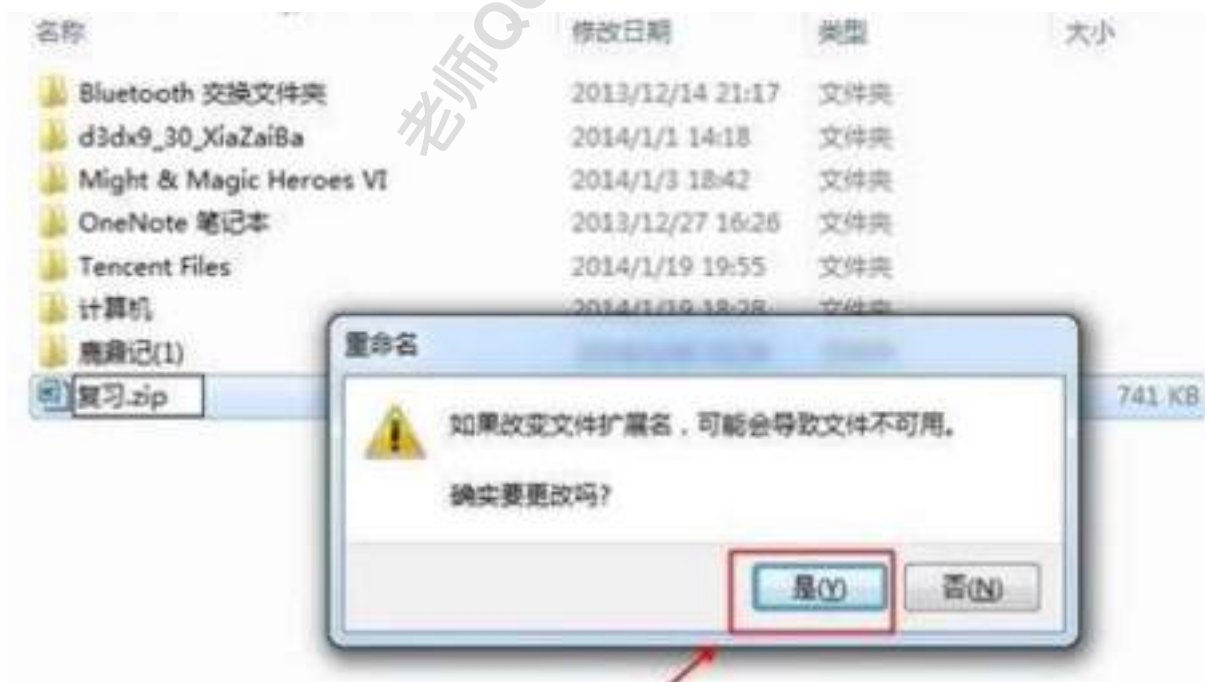
1. Inode索引节点号，（人的身份证，家庭地址等信息，唯一），系统寻找文件名 > Inode号 > 文件内容
2. 文件权限，第一个字符是文件类型，随后9个字符是文件权限，最后一个. 有关selinux
3. 文件硬链接数量，与ln命令配合
4. 文件所属用户
5. 文件所属用户组
6. 文件目录大小
7. 文件修改时间
8. 文件名

文件扩展名

windows下的文件扩展名

- docx
- pptx
- pdf
- jpg
- avi
- mp4
- gif
- rar
- Zip

对于windows系统，文件名后缀有问题则会影响使用



扩展名

Linux文件的扩展名只是方便阅读，对文件类型不影响

Linux通过文件属性区分文件类型

- .txt文本类型
- .conf .cfg .configure 配置文件
- .sh .bash 脚本后缀
- .py 脚本后缀
- .rpm 红帽系统二进制软件包名
- .tar .gz .zip 压缩后缀

文件类型

可以通过ls -F 给文件结尾加上特殊标识

格式	类型
ls -l看第一个字符	
-	普通文件regular file, (二进制, 图片, 日志, txt等)
d	文件夹directory
b	块设备文件, /dev/sda1, 硬盘, 光驱
c	设备文件, 终端/dev/tty1,网络串口文件
s	套接字文件, 进程间通信 (socket) 文件
p	管道文件pipe
l	链接文件,link类型, 快捷方式

普通文件

通过如下命令生成都是普通文件(windows中各种扩展名的文件, 放入linux也是普通文件类型)

- echo
- touch
- cp
- cat
- 重定向符号 >

普通文件特征就是文件类型是, "-"开头, 以内容区分一般分为

- 纯文本, 可以用cat命令读取内容, 如字符、数字、特殊符号等
- 二进制文件 (binary), Linux中命令属于这种格式, 例如ls、cat等命令

文件夹

文件权限开头带有d字符的文件表示文件夹, 是一种特殊的Linux文件

- mkdir
- cp拷贝文件夹

链接类型

- ln命令创建

类似windows的快捷方式

file命令

显示文件的类型

```
[root@luffycity tmp]# file /usr/bin/python2.7          #二进制解释器类型
/usr/bin/python2.7: ELF 64-bit LSB executable

[root@luffycity tmp]# file /usr/bin/yum                #yum是python的脚本文件
/usr/bin/yum: Python script, ASCII text executable

[root@luffycity tmp]# file /usr/bin/cd                 #shell脚本，内置命令
/usr/bin/cd: POSIX shell script, ASCII text executable

[root@luffycity tmp]# file hehe.txt                    #text类型
hehe.txt: ASCII text

[root@luffycity tmp]# file heihei                      #文件夹
heihei: directory

[root@luffycity tmp]# file /usr/bin/python2            #软链接类型
/usr/bin/python2: symbolic link to `python2.7'
```

which

查找PATH环境变量中的文件，linux内置命令不在path中

```
[root@luffycity tmp]# which python
/usr/bin/python
```

whereis命令

whereis命令用来定位指令的二进制程序、源代码文件和man手册页等相关文件的路径。

```
[root@luffycity tmp]# whereis python
python: /usr/bin/python /usr/bin/python2.7 /usr/lib/python2.7 /usr/lib64/python2.7
/etc/python /usr/include/python2.7 /usr/share/man/man1/python.1.gz
```

tar命令

tar命令在linux系统里，可以实现对多个文件进行，压缩、打包、解包



打包

将一大堆文件或目录汇总成一个整体。

压缩

将大文件压缩成小文件，节省磁盘空间。



语法：

`tar (选项) (参数)`

- A或--catenate：新增文件到已存在的备份文件；
- B：设置区块大小；
- c或--create：建立新的备份文件；
- C <目录>：这个选项用在解压缩，若要在特定目录解压缩，可以使用这个选项。
- d：记录文件的差别；
- x或--extract或--get：从备份文件中还原文件；
- t或--list：列出备份文件的内容；
- z或--gzip或--ungzip：通过gzip指令处理备份文件；
- Z或--compress或--uncompress：通过compress指令处理备份文件；
- f<备份文件>或--file=<备份文件>：指定备份文件；
- v或--verbose：显示指令执行过程；
- r：添加文件到已经压缩的文件；
- u：添加改变了和现有的文件到已经存在的压缩文件；
- j：支持bzip2解压文件；
- v：显示操作过程；
- l：文件系统边界设置；
- k：保留原有文件不覆盖；
- m：保留文件不被覆盖；


```
-w: 确认压缩文件的正确性;  
-p或--same-permissions: 用原来的文件权限还原文件;  
-P或--absolute-names: 文件名使用绝对名称, 不移除文件名称前的"/"号; 不建议使用  
-N <日期格式> 或 --newer=<日期时间>: 只将较指定日期更新的文件保存到备份文件里;  
--exclude=<范本样式>: 排除符合范本样式的文件。  
-h, --dereference跟踪符号链接; 将它们所指向的文件归档并输出
```

案例

仅打包, 不压缩

```
#tar 参数 包裹文件名 需要打包的文件  
[alex@luffycity tmp]$ tar -cvf alltmp.tar ./*
```

打包后且用gzip命令压缩, 节省磁盘空间

```
[alex@luffycity tmp]$ tar -zcvf alltmp.tar ./*
```

注意

- f参数必须写在最后, 后面紧跟压缩文件名
- tar命令仅打包, 习惯用.tar作为后缀
- tar命令加上z参数, 文件以.tar.gz或.tgz表示

列出tar包内的文件

```
#根据tar包文件后缀, 决定是否添加z参数, 调用gzip  
[alex@luffycity tmp]$ tar -ztvf alltmp2.tar.gz
```

拆开tar包

```
[root@luffycity tmp]# tar -xf alltmp.tar
```

拆开tar的压缩包

```
tar -zxvf ../alltmp2.tar.gz ./
```

拆除tar包中部分文件

```
#正常解压命令, 单独加上你要拆除的文件名, 指定路径  
#先看下tar包中有什么内容, 再指定文件解压  
  
[root@luffycity tmp]# tar -ztvf ../alltmp2.tar.gz  
  
[root@luffycity tmp]# tar -zxvf ../alltmp2.tar.gz ./alltmp.tar  
./alltmp.tar
```

指定目录解tar包

```
[root@luffycity tmp]# tar -xf alltmp.tar -C /opt/data/
```

排除文件解包

#注意--exclude 跟着文件名或是文件夹，不得加斜杠，排除多个文件，就写多个--exclude

```
[root@luffycity tmp]# tar -zxvf ../alltmp2.tar.gz --exclude data
```

打包链接文件

-h参数能够保证，打包的不仅仅是个快捷方式，而是找到源文件

```
[root@luffycity tmp]# ll
total 116108
-rw-rw-r--. 1 alex alex      0 Oct 16 18:36 123
-rw-rw-r--. 1 alex alex 11888884 Oct 17 09:39 alex.txt
drwxr-xr-x. 2 root root      6 Oct 16 16:19 data
drwxr-xr-x. 2 root root      6 Oct 16 16:19 data2
lrwxrwxrwx. 1 root root      8 Oct 16 17:26 h.txt -> hehe.txt 软链接
-rw-r--r--. 1 root root    16 Oct 16 16:36 hehe.txt
drwxr-xr-x. 2 root root      6 Oct 16 16:19 hehei

[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# tar -zcf test_link.tgz ./h.txt 压缩且打包
[root@luffycity tmp]# h.txt 软链接
[root@luffycity tmp]#
[root@luffycity tmp]# tar -tzvf test_link.tgz 查看压缩包中内容
lrwxrwxrwx root/root      0 2019-10-16 17:26 ./h.txt -> hehe.txt
[root@luffycity tmp]# 发现是个快捷方式
[root@luffycity tmp]#
[root@luffycity tmp]# tar -hczf test_link2.tgz ./h.txt
[root@luffycity tmp]# 再次压缩，加上-h参数
[root@luffycity tmp]# tar -tzvf test_link2.tgz
-rw-r--r-- root/root    16 2019-10-16 16:36 ./h.txt 发现已然是源文件
```

打包/etc下所有普通文件

```
[root@luffycity tmp]# tar -zcvf etc.tgz `find /etc -type f`
[root@luffycity tmp]# tar -tzvf etc.tgz
```

gzip命令

要说tar命令是个纸箱子用于打包，gzip命令就是压缩机器

gzip通过压缩算法`lempeI-ziv` 算法(`lz77`) 将文件压缩为较小文件, 节省60%以上的存储空间, 以及网络传输速率

gzip(选项)(参数)

- a或—ascii: 使用ASCII文字模式;
- c或--stdout或--to-stdout 把解压后的文件输出到标准输出设备。
- d或--decompress或---uncompress: 解开压缩文件;
- f或—force: 强行压缩文件。不理睬文件名称或硬连接是否存在以及该文件是否为符号连接;
- h或—help: 在线帮助;
- l或—list: 列出压缩文件的相关信息;
- L或—license: 显示版本与版权信息;
- n或--no-name: 压缩文件时, 不保存原来的文件名称及时间戳记;
- N或—name: 压缩文件时, 保存原来的文件名称及时间戳记;
- q或—quiet: 不显示警告信息;
- r或—recursive: 递归处理, 将指定目录下的所有文件及子目录一并处理;
- S或<压缩字尾字符串>或---suffix<压缩字尾字符串>: 更改压缩字尾字符串;
- t或—test: 测试压缩文件是否正确无误;
- v或—verbose: 显示指令执行过程;
- V或—version: 显示版本信息;
- <压缩效率>: 压缩效率是一个介于1-9的数值, 预设值为“6”, 指定愈大的数值, 压缩效率就会愈高;
- best: 此参数的效果和指定“-9”参数相同;
- fast: 此参数的效果和指定“-1”参数相同。

案例

```
#压缩目录中每一个html文件为.gz, 文件夹无法压缩, 必须先tar打包
gzip *.html          #gzip压缩, 解压都会删除源文件
```

列出压缩文件中信息

```
[root@luffycity tmp]# gzip -l *.gz          #不解压显示压缩文件内信息, 以及压缩率
```

compressed	uncompressed	ratio	uncompressed_name
28	0	0.0%	10.html
24	0	0.0%	123
27	0	0.0%	1.html
27	0	0.0%	2.html
27	0	0.0%	3.html
27	0	0.0%	4.html
27	0	0.0%	5.html
27	0	0.0%	6.html
27	0	0.0%	7.html
27	0	0.0%	8.html
27	0	0.0%	9.html
23581672	118888884	80.2%	alex.txt
23582535	118896640	80.2%	alltmp.tar
289	470	44.9%	glances.log
45	16	-12.5%	hehe.txt
47164836	237786010	80.2%	(totals)

解压缩且显示过程

```
[root@luffycity tmp]# gzip -dv *.gz
10.html.gz:      0.0% -- replaced with 10.html
123.gz:          0.0% -- replaced with 123
1.html.gz:       0.0% -- replaced with 1.html
2.html.gz:       0.0% -- replaced with 2.html
3.html.gz:       0.0% -- replaced with 3.html
4.html.gz:       0.0% -- replaced with 4.html
5.html.gz:       0.0% -- replaced with 5.html
6.html.gz:       0.0% -- replaced with 6.html
7.html.gz:       0.0% -- replaced with 7.html
8.html.gz:       0.0% -- replaced with 8.html
9.html.gz:       0.0% -- replaced with 9.html
alex.txt.gz:     80.2% -- replaced with alex.txt
alltmp.tar.gz:   80.2% -- replaced with alltmp.tar
glances.log.gz:  44.9% -- replaced with glances.log
hehe.txt.gz:    -12.5% -- replaced with hehe.txt
```

压缩保留源文件

```
#-c参数
[root@luffycity tmp]# gzip -c alltmp.tar > alltmp.tar.gz
```

gzip套件提供了许多方便的工具命令，可以直接操作压缩文件内容

- zcat, 直接读取压缩文件内容 `zcat hehe.txt.gz`
- zgrep
- zless
- zdiff

zip命令

zip 命令：是一个应用广泛的跨平台的压缩工具，压缩文件的后缀为 zip文件，还可以压缩文件夹

语法：

`zip 压缩文件名 要压缩的内容`

- A 自动解压文件
- c 给压缩文件加注释
- d 删除文件
- F 修复损坏文件
- k 兼容 DOS
- m 压缩完毕后，删除源文件
- q 运行时不显示信息处理信息
- r 处理指定目录和指定目录下的使用子目录

```
-v 显示信息的处理信息
-x “文件列表” 压缩时排除文件列表中指定的文件
-y 保留符号链接
-b<目录> 指定压缩到的目录
-i<格式> 匹配格式进行压缩
-L 显示版权信息
-t<日期> 指定压缩文件的日期
-<压缩率> 指定压缩率
最后更新 2018-03-08 19:33:4
```

案例

```
#压缩当前目录下所有内容为alltmp.zip文件
[root@luffycity tmp]# zip alltmp.zip ./*

#压缩多个文件夹
[root@luffycity tmp]# zip -r data.zip ./data ./data2
```

unzip命令用于解压

参数

```
-l: 显示压缩文件内所包含的文件；
-d<目录> 指定文件解压后所要存储的目录。
```

案例

```
#查看压缩文件内容
[root@luffycity tmp]# unzip -l data.zip

#解压缩zip文件
[root@luffycity tmp]# unzip data.zip
```

date命令

date命令用于显示当前系统时间，或者修改系统时间

语法

```
date 参数 时间格式
```

参数

```
-d, --date=STRING
    显示由 STRING 指定的时间，而不是当前时间
```

```
-s, --set=STRING
    根据 STRING 设置时间

-u, --utc, --universal
    显示或设置全球时间(格林威治时间)
```

时间格式

```
%%
    文本的 %

%a
    当前区域的星期几的简写 (Sun..Sat)

%A
    当前区域的星期几的全称 (不同长度) (Sunday..Saturday)

%b
    当前区域的月份的简写 (Jan..Dec)

%B
    当前区域的月份的全称(变长) (January..December)

%C
    当前区域的日期和时间 (Sat Nov 04 12:02:33 EST 1989)

%d
    (月份中的)几号(用两位表示) (01..31)

%D
    日期(按照 月/日期/年 格式显示) (mm/dd/yy)

%e
    (月份中的)几号(去零表示) ( 1..31)

%h
    同 %b

%H
    小时(按 24 小时制显示, 用两位表示) (00..23)

%I
    小时(按 12 小时制显示, 用两位表示) (01..12)

%j
    (一年中的)第几天(用三位表示) (001..366)

%k
    小时(按 24 小时制显示, 去零显示) ( 0..23)

%l
    小时(按 12 小时制显示, 去零表示) ( 1..12)

%m
    月份(用两位表示) (01..12)

%M
    分钟数(用两位表示) (00..59)

%n
    换行

%p
    当前时间是上午 AM 还是下午 PM

%r
    时间, 按 12 小时制显示 (hh:mm:ss [A/P]M)

%S
    从 1970年1月1日0点0分0秒到现在历经的秒数 (GNU扩充)
```

%S
秒数(用两位表示)(00..60)

%t
水平方向的 tab 制表符

%T
时间,按 24 小时制显示(hh:mm:ss)

%U
(一年中的)第几个星期,以星期天作为一周的开始(用两位表示)(00..53)

%V
(一年中的)第几个星期,以星期一作为一周的开始(用两位表示)(01..52)

%w
用数字表示星期几(0..6);0 代表星期天

%W
(一年中的)第几个星期,以星期一作为一周的开始(用两位表示)(00..53)

%x
按照(mm/dd/yy)格式显示当前日期

%X
按照(%H:%M:%S)格式显示当前时间

%y
年的后两位数字(00..99)

%Y
年(用4位表示)(1970...)

%Z
按照RFC-822中指定的数字时区显示(如,-0500)(为非标准扩充)

%Z
时区(例如,EDT(美国东部时区)),如果不能决定是哪个时区则为空

默认情况下,用0填充数据的空缺部分.GNU的date命令能分辨在`%'和数字指示之间的以下修改.

`-` (连接号) 不进行填充
`_` (下划线) 用空格进行填充

案例

显示当前系统部分时间

- 1.显示短年份
date +%y
- 2.显示长年份
date +%Y
- 3.显示月份
date +%m
- 4.显示几号
date +%d
- 5.显示几时
date +%H

6. 显示几分

```
date +%M
```

7. 显示整秒

```
date +%S
```

8. 显示时间如, 年-月-日

```
date +%F
```

9. 显示时间如, 时: 分: 秒

```
date +%T
```

-d参数指定时间显示, 仅仅是显示

1. 显示昨天

```
date +%F -d "-1day"
```

2. 显示昨天

```
date +%F -d "yesterday"
```

3. 显示前天

```
date +%F -d "-2day"
```

4. 显示明天日期

```
date +%F -d "+1day"
```

5. 显示明天, 英文表示

```
date +%F -d "tomorrow"
```

6. 显示一个月之前, 之后

```
[root@pylinux /]# date +%F -d "1month"
```

```
2019-12-01
```

```
[root@pylinux /]# date +%F -d "-1month"
```

```
2019-10-01
```

7. 显示一年后

```
date +%F -d "1year"
```

8. 显示60分钟后

```
date +%T -d "60min"
```

+表示未来

-表示过去

day表示日

month表示月份

year表示年

min表示分钟

-s设置时间

设置时间较少，一般配置ntp时间服务器

1. 设置时间

```
[root@pylinux /]# date -s "20170808"
2017年 08月 08日 星期二 00:00:00 CST
[root@pylinux /]#
[root@pylinux /]# date
2017年 08月 08日 星期二 00:00:00 CST
```

2. 修改分钟

```
[root@pylinux /]# date -s "05:06:33"
2017年 08月 08日 星期二 05:06:33 CST
[root@pylinux /]# date
2017年 08月 08日 星期二 05:06:33 CST
```

3. 修改日期和分钟

```
[root@pylinux /]# date -s "20180606 05:30:30"
2018年 06月 06日 星期三 05:30:30 CST
[root@pylinux /]# date
2018年 06月 06日 星期三 05:30:31 CST
```

4. 可设置不同格式的时间

```
date -s "2018-06-06 05:30:30"
date -s "2018/07/07 05:30:30"
```

shred命令



用法：shred [选项]... 文件...

多次覆盖文件，使得即使是昂贵的硬件探测仪器也难以将数据复原。

```
-u, --remove 覆盖后截断并删除文件  
shred heihei.txt 随机覆盖文件内容，不删除源文件
```

案例

彻底粉碎且删除文件

```
[root@pylinux tmp]# ls -lh  
总用量 25M  
-rw-r--r-- 1 root root 25M 10月 14 15:02 heihei.txt  
[root@pylinux tmp]#  
[root@pylinux tmp]# shred -u heihei.txt
```

老师QQ: 877348180

[TOC]

Linux用户管理

用户管理篇



Root用户登录系统后可以做很多事

- 写文档
- 听音乐
- 聊QQ
- 聊微信
- 写代码
- 上班五分钟，闲聊俩小时
- 打卡下班
- ...

当然了，完全可以坐在办公室，远程连接服务器工作

多用户多任务



多个用户使用同一个操作系统，每个人做自己的事。

每个人都有自己的账号密码，权限也不一样，好比老板权限最大，员工权限较低

多用户大多都是远程登录去控制服务器

Linux的用户管理

Linux系统不同用户权限不一样，好比小张想用我的服务器，我为了保护隐私与资料安全，开通普通用户(useradd xiaozhang)，普通用户权限较低，随便他折腾了。

还有计算机程序默认创建的用户，如ftp, nobody等等

用户信息存放在/etc/passwd文件中

用户角色划分



- root
- 普通用户
- 虚拟用户

现代操作系统一般属于多用户的操作系统，也就是说，同一台机器可以为多个用户建立账户，一般这些用户都是为普通用户，这些普通用户能同时登录这台计算机，计算机对这些用户分配一定的资源。

普通用户在所分配到的资源内进行各自的操作，相互之间不受影响。但是这些普通用户的权限是有限制的，且用户太多的话，管理就不便，从而引入root用户。

此用户是唯一的，且拥有系统的所有权限。root用户所在的组称为root组。

“组”是具有相似权限的多个用户的集合。

root的权利

Linux系统的特性就是可以满足多个用户，同时工作，因此Linux系统必须具备很好的安全性。

在安装RHEL7时设置的root管理员密码，这个root管理员就是所有UNIX系统中的超级用户，它拥有最高的系统所有权，能够管理系统的各项功能，如添加/删除用户，启动/关闭进程，开启/禁用硬件设备等等。

因此“能力越大，责任越大”，root权限必须很好的掌握，否则一个错误的命令可能会摧毁整个系统。

root为什么叫root?

- UID, user Identify, 好比身份证号
- GID, group Identify, 好比户口本的家庭编号
- 在Linux系统中，用户也有自己的UID身份账号且唯一

- 在Linux中UID为0，就是超级用户，如要设置管理员用户，可以改UID为0，建议用sudo
- 系统用户UID为1~999 Linux安装的服务程序都会 创建独有的用户 负责运行。
- 普通用户UID从1000开始：由管理员创建（centos7），最大值1000~60000范围
- centos6创建普通用户是500开始

UID 用户id号，身份证号

GID 用户组id号，部门编号

root 用户、组、id都为0，属于老板

用户组group

为了方便管理属于同一组的用户，Linux 系统中还引入了用户组的概念。通过使用用户组号码(GID, Group Identification)，我们可以把多个用户加入到同一个组中，从而方便为组中的用户统一规划权限或指定任务。

假设有一个公司中有多个部门，每个部门中又有很多员工。

如果只想让员工访问本部门内的资源，则可以针对部门而非具体的员工来设置权限。

例如，可以通过对技术部门设置权限，使得只有技术部门的员工可以访问公司的数据库信息等。

Linux管理员在创建用户时，将自动创建一个与其同名的用户组，这个用户组只有该用户一个人，



windows下的管理员



Linux/unix是一个多用户、多任务的操作系统。

root: 默认在Unix/linux操作系统中拥有最高的管理权限。可以理解为qq群的群主↓↓↓



root QQ群主 权利最大，随心所欲
普通用户 QQ群主有权拉进群，踢掉

普通用户: 是管理员或者具备管理权限的用户所创建的，只能读、看，不能增、删、改。

用户和组的关系

- 一对一，一个用户可以存在一个组里，组里就一个成员
- 一对多，一个用户呆在多个组里面
- 多对一，多个用户在一个组里，这些用户和组有相同的权限
- 多对多，多个用户存在多个组里

常用命令解释器

```
/bin/sh 默认
/bin/bash 默认
/sbin/nologin 虚拟用户
/dash ubuntu
csh unix
tsh unix
```

用户信息配置文件

/etc/passwd文件内容

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:999:User for polkitd:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
abrt:x:173:173:/:etc/abrt:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
postfix:x:89:89:/:var/spool/postfix:/sbin/nologin
alex:x:1000:1000:/:home/alex:/bin/bash
yu:x:1001:1001:/:home/yu:/bin/bash
```

冒号分割

用户名 密码 UID GID 用户注释 用户家目录 用户使用的解释器

- /etc/passwd 用户信息
- /etc/shadow 用户密码信息
- /etc/group 用户组信息
- /etc/gshadow 用户组密码信息
- /etc/skel

/etc/passwd字段信息解释

字段名	解释
用户名	对应UID，是用户登录系统的名字，系统中唯一不得重复
用户密码	存放在/etc/shadow文件中进行保护密码
用户UID	用户ID号，由一个整数表示
用户组GID	范围、最大值和UID一样，默认创建用户会创建用户组
用户说明	对此用户描述
用户家目录	用户登录后默认进去的家目录，一般是【/home/用户名】
shell解释器	当前登录用户使用的解释器。centos/redhat系统中，默认的都是bash。若是禁止此用户登录机器，改为/sbin/nologin即可

用户/密码文件权限

```
#用户信息文件，权限是644，所有人可读，有一定安全隐患
[root@pylinux ~]# ll /etc/passwd
-rw-r--r-- 1 root root 1698 10月 13 2019 /etc/passwd

#用户密码文件，除了root用户，其他用户默认是没有任何权限，
[root@pylinux ~]# ll /etc/shadow
----- 1 root root 892 10月 20 2019 /etc/shadow

#用户密码文件
[root@pylinux ~]# tail -5 /etc/shadow
mysql:!!:17980::::::
yu:$1$Kx9cz6sK$GE3jiHtjJikn9Ai4ECINn/:18031:0:99999:7:::
epmd:!!:18074::::::
rabbitmq:!!:18074::::::
py:!!:18182:0:99999:7:::
```

用户组/组密码文件

```
#用户组信息文件
/etc/group

#用户组密码文件
/etc/gshadow
```

对于大型服务器，用户和用户组数量较多，需要定制复杂的权限控制，会用到组密码

用户管理的命令

命令	作用
useradd	创建用户
usermod	修改用户信息
userdel	删除用户及配置文件
passwd	更改用户密码
chpasswd	批量更新用户密码
chage	修改用户密码属性
id	查看用户UID、GID、组信息
su	切换用户
sudo	用root身份执行命令

visudo

编辑sudoers配置文件

用户创建家目录过程

1. 命令创建

```
useradd chaoge
```

2. 系统把/etc/skel目录下的内容复制到创建的用户家目录

```
[root@luffycity skel]# pwd
```

```
/etc/skel
```

```
[root@luffycity skel]# ls -la
```

```
total 24
```

```
drwxr-xr-x. 3 root root 78 Oct 17 13:59 .
```

```
drwxr-xr-x. 85 root root 8192 Oct 17 14:03 ..
```

```
-rw-r--r--. 1 root root 18 Apr 11 2018 .bash_logout
```

```
-rw-r--r--. 1 root root 193 Apr 11 2018 .bash_profile
```

```
-rw-r--r--. 1 root root 231 Apr 11 2018 .bashrc
```

```
drwxr-xr-x. 4 root root 39 Oct 13 17:42 .mozilla
```

3. 修改权限拥有者

```
[root@luffycity skel]# ls /home/yu/ -la
```

```
total 16
```

```
drwx-----. 3 yu yu 99 Oct 16 18:36 .
```

```
drwxr-xr-x. 4 root root 28 Oct 16 18:35 ..
```

```
-rw-----. 1 yu yu 77 Oct 16 18:36 .bash_history
```

```
-rw-r--r--. 1 yu yu 18 Apr 11 2018 .bash_logout
```

```
-rw-r--r--. 1 yu yu 193 Apr 11 2018 .bash_profile
```

```
-rw-r--r--. 1 yu yu 231 Apr 11 2018 .bashrc
```

```
drwxr-xr-x. 4 yu yu 39 Oct 13 17:42 .mozilla
```

useradd命令

useradd命令用于Linux中创建的新的系统用户。

useradd可用来建立用户帐号。帐号建好之后，再用passwd设定帐号的密码，而可用userdel删除帐号。

使用useradd指令所建立的帐号，实际上是保存在/etc/passwd文本文件中。

-c<备注>：加上备注文字。备注文字会保存在passwd的备注栏位中；

-d<登入目录>：指定用户登入时的启始目录；

-D：变更预设值；

-e<有效期限>：指定帐号的有效期限；

-f<缓冲天数>：指定在密码过期后多少天即关闭该帐号；

-g<群组>：指定用户所属的群组；

-G<群组>：指定用户所属的附加群组；

-m：自动建立用户的登入目录；

-M：不要自动建立用户的登入目录；

-n：取消建立以用户名称为名的群组；

-r：建立系统帐号；

```
-s<shell>: 指定用户登入后所使用的shell;
-u<uid>: 指定用户id。
```

创建用户流程

- 1.useradd chaoge
- 2.系统读取/etc/login.defs（用户定义文件），和/etc/default/useradd（用户默认配置文件）俩文件中定义的规则创建新用户
- 3.向/etc/passwd和/etc/group文件中添加用户和组信息，向/etc/shadow和/etc/gshadow中添加密码信息
- 4.根据/etc/default/useradd文件中配置的信息创建用户家目录
- 5.把/etc/skel中所有的文件复制到新用户家目录中

文件/etc/login.defs

```
[root@pylinux ~]# grep -v "^#" /etc/login.defs | grep -v "^$"
MAIL_DIR      /var/spool/mail      #用户的邮件存放位置
PASS_MAX_DAYS  99999          #密码最长使用天数
PASS_MIN_DAYS  0              #更换密码最短时间
PASS_MIN_LEN   8              #密码最小长度
PASS_WARN_AGE  7              #密码失效前几天开始报警
UID_MIN        1000         #UID开始位置
UID_MAX        60000        #UID结束位置
SYS_UID_MIN    201
SYS_UID_MAX    999
GID_MIN        1000
GID_MAX        60000
SYS_GID_MIN    201
SYS_GID_MAX    999
CREATE_HOME    yes          #是否创建家目录
UMASK          077          #家目录的umask值
USERGROUPS_ENAB yes
ENCRYPT_METHOD  MD5          #密码加密算法
MD5_CRYPT_ENAB yes
```

文件/etc/default/useradd

```
[root@pylinux ~]# grep -v "^#" /etc/default/useradd | grep -v "^$"
GROUP=100
HOME=/home          #在/home目录下创建家目录
INACTIVE=-1         #开启用户过期
EXPIRE=             #用户终止日期
SHELL=/bin/bash     #新用户默认解释器
SKEL=/etc/skel      #用户环境变量文件存放目录
CREATE_MAIL_SPOOL=yes
```

创建用户有关的目录/etc/skel

此目录存放新用户需要的基础环境变量文件，添加新用户的时候，这个目录下所有文件自动被复制到新家目录下，且默认是隐藏文件，以点开头。

```
[root@pylinux ~]# ls -la /etc/skel/
总用量 28
drwxr-xr-x.  2 root root  4096 4月  11 12:59 .
drwxr-xr-x. 97 root root 12288 7月  11 05:14 ..
-rw-r--r--.  1 root root   18 4月  11 08:53 .bash_logout      #用户环境变量文件，退出加载
-rw-r--r--.  1 root root  193 4月  11 08:53 .bash_profile      #环境变量文件，登录加载
-rw-r--r--.  1 root root  231 4月  11 08:53 .bashrc            #环境变量文件，登录加载
```

面试题，如何恢复用户命令提示符

1. 正常切换用户

```
su - yu
```

```
[root@pylinux ~]# su - yu
```

```
[yu@pylinux ~]$
```

2. 删除用户家目录下环境变量文件

```
[yu@pylinux ~]$ rm -rf .bash*
```

3. 退出登录，再次登录，发现命令提示符故障

```
[yu@pylinux ~]$ logout
```

```
[root@pylinux ~]#
```

```
[root@pylinux ~]#
```

```
[root@pylinux ~]# su - yu
```

```
上一次登录: 三 7月 11 06:19:08 CST 2018pts/1 上
```

```
-bash-4.2$
```

4. 恢复方法

```
-bash-4.2$ cp /etc/skel/.bash* ~/
```

```
-bash-4.2$ logout
```

```
[root@pylinux ~]# su - yu
```

```
上一次登录: 三 7月 11 06:20:10 CST 2018pts/1 上
```

```
[yu@pylinux ~]$
```

用户管理命令案例

useradd命令

-c<备注>: 加上备注文字。备注文字会保存在passwd的备注栏位中;

-d<登入目录>: 指定用户登入时的启始目录;

- D: 变更预设值;
- e<有效期限>: 指定帐号的有效期限;
- f<缓冲天数>: 指定在密码过期后多少天即关闭该帐号;
- g<群组>: 指定用户所属的群组;
- G<群组>: 指定用户所属的附加群组;
- m: 自动建立用户的登入目录;
- M: 不要自动建立用户的登入目录;
- n: 取消建立以用户名称为名的群组;
- r: 建立系统帐号;
- s<shell>: 指定用户登入后所使用的shell;
- u<uid>: 指定用户id。

普通创建用户

```
[root@luffycity ~]# useradd chaoge

[root@luffycity ~]# ls -l /home/
total 0
drwx----- . 3 chaoge chaoge 78 Oct 17 14:42 chaoge

[root@luffycity ~]# grep -w chaoge /etc/passwd      # grep过滤 -w必须全字符串匹配
chaoge:x:1002:1002::/home/chaoge:/bin/bash

[root@luffycity ~]# grep -w chaoge /etc/shadow      #默认没密码, 通过passwd设置密码
chaoge:!!:18186:0:99999:7:::

[root@luffycity ~]# grep -w chaoge /etc/group
chaoge:x:1002:

[root@luffycity ~]# id chaoge      #查看用户信息
uid=1002(chaoge) gid=1002(chaoge) groups=1002(chaoge)
```

用户且指定uid和属组

```
1. 创建用户组
groupadd -g 801 old

2. 创建新用户
useradd -g old -u 888 oldchao

3. 检查用户信息
[root@luffycity ~]# id oldchao
uid=888(oldchao) gid=801(old) groups=801(old)
```

-M -s参数用法

```
#创建用户禁止登陆, 且不创建家目录
[root@luffycity ~]# useradd -M -s /sbin/nologin oldyu
```

```
[root@luffycity ~]# grep -w oldyu /etc/passwd
oldyu:x:1003:1003:~/home/oldyu:/sbin/nologin
[root@luffycity ~]# ls /home/      #没有oldyu
alex  chaoge  oldchao  yu
```

多个参数用法，指定用户信息

```
[root@luffycity ~]# useradd -u 789 -s /bin/sh -c learn_linux -G root,old -e "2019/10/18" -f 2 -d /tmp/luffychao luffychao

[root@luffycity ~]# id luffychao      #检查用户信息
uid=789(luffychao) gid=1004(luffychao) groups=1004(luffychao),0(root),801(old)
```

```
[root@luffycity ~]#
[root@luffycity ~]#
[root@luffycity ~]#
[root@luffycity ~]#
[root@luffycity ~]#
[root@luffycity ~]# useradd -u 789 -s /bin/sh -c learn_linux -G root,old -e "2019/10/18" -f 2 -d /tmp/luffychao luffychao
```

-D参数用来修改配置文件/etc/default/useradd文件的默认值

```
useradd -D 参数选项

-e default_expire_date 用户停止日期
-s default_shell 用户登录后使用的解释器
```

-D使用案例

```
[root@pylinux ~]# grep -i "shell" /etc/default/useradd      #检查默认创建用户的登录解
释器
SHELL=/bin/bash
[root@pylinux ~]#
[root@pylinux ~]#
[root@pylinux ~]# useradd -D -s /sbin/nologin                #修改默认解释器为禁止登录
[root@pylinux ~]# grep -i "shell" /etc/default/useradd      #检查配置文件参数
SHELL=/sbin/nologin
[root@pylinux ~]#
[root@pylinux ~]# useradd testyu                              #此时创建新用户
[root@pylinux ~]# grep "testyu" /etc/passwd                 #检查信息，发现是禁止登录
testyu:x:1002:1002:~/home/testyu:/sbin/nologin
[root@pylinux ~]# useradd -D -s /bin/bash                    #最好还是改回去，防止后面出错
[root@pylinux ~]# grep -i "shell" /etc/default/useradd      #验证是否修改回来
SHELL=/bin/bash
```

usermode命令

usermod命令用于修改系统已存在的用户信息，只能修改未使用中的用户

语法

usermod(选项)(参数)

选项

- c<备注>: 修改用户帐号的备注文字;
- d<登入目录>: 修改用户登入时的目录;
- e<有效期限>: 修改帐号的有效期限;
- f<缓冲天数>: 修改在密码过期后多少天即关闭该帐号;
- g<群组>: 修改用户所属的群组;
- G<群组>; 修改用户所属的附加群组;
- l<帐号名称>: 修改用户帐号名称;
- L: 锁定用户密码, 使密码无效;
- s<shell>: 修改用户登入后所使用的shell;
- u<uid>: 修改用户ID;
- U: 解除密码锁定。

案例

```
[root@luffycity ~]# usermod -u 788 -s /sbin/nologin -c changeUser -G old -e "2020/10/10" -f 10 -d /home/luffychao luffychao
[root@luffycity ~]#
[root@luffycity ~]#
[root@luffycity ~]# id luffychao
uid=788(luffychao) gid=1004(luffychao) groups=1004(luffychao),801(old)
[root@luffycity ~]#
[root@luffycity ~]# grep -w luffychao /etc/passwd
luffychao:x:788:1004:changeUser:/home/luffychao:/sbin/nologin
```

userdel命令

删除用户与相关文件

- 建议注释/etc/passwd用户信息而非直接删除用户

语法

userdel(选项)(参数)

选项

- f: 强制删除用户, 即使用户当前已登录;
- r: 删除用户的同时, 删除与用户相关的所有文件。

案例

```
[root@luffycity ~]# userdel oldyu          #保留家目录
[root@luffycity ~]# userdel -rf oldchao    #强制删除用户与家目录
```

groupadd命令

groupadd命令用于创建一个新的工作组，新工作组的信息将被添加到系统文件中。

```
语法
groupadd - 建立新群组
groupadd [ -ggid [ -o ]] [ -r ] [ -f ] group [[ ]]
```

选项

- g: 指定新建工作组的id;
- r: 创建系统工作组，系统工作组的组ID小于500;
- K: 覆盖配置文件"/ect/login.defs";
- o: 允许添加组ID号不唯一的工作组。

案例

```
group -g 801 old
```

groupdel命令

删除用户组

```
groupdel 组名
```

passwd命令

passwd命令修改用户密码和过期时间等，root可以改普通用户，反之不可以

```
语法
passwd(选项)(参数)
选项
-d: 删除密码，仅有系统管理者才能使用；
-f: 强制执行；
-k: 设置只有在密码过期失效后，方能更新；
-l: 锁住密码；
-s: 列出密码的相关信息，仅有系统管理者才能使用；
-u: 解开已上锁的帐号。
-i: 密码过期多少天后禁用账户
-x: 设置x天后可以修改密码
-n: 设置n天内不得改密码
-e: 密码立即过期，强制用户修改密码
-w: 用户在密码过期前收到警告信息的天数
```

案例

改自己的密码

```
[root@luffycity ~]# passwd          #对当前用户改密
```



```
Changing password for user root.  
New password:  
BAD PASSWORD: The password is shorter than 8 characters  
Retype new password:  
passwd: all authentication tokens updated successfully.      #即使密码太简单，也能改
```

修改普通用户密码

```
[root@luffycity ~]# passwd luffychao  
Changing password for user luffychao.  
New password:  
BAD PASSWORD: The password is shorter than 8 characters  
Retype new password:  
passwd: all authentication tokens updated successfully.
```

列出用户密码信息

```
[root@luffycity ~]# passwd -S luffychao  
luffychao PS 2019-10-17 0 99999 7 10 (Password set, SHA512 crypt.)
```

一条命令设置密码，企业常用

```
[root@luffycity ~]# echo "123123" |passwd --stdin luffychao      #--stdin从标准输入  
中获取123123  
Changing password for user luffychao.  
passwd: all authentication tokens updated successfully.
```

passwd实际场景

7天内用户不得改密码，60天后可以修改，过期前10天通知用户，过期30天后禁止用户登录

```
[root@luffycity ~]# passwd -n 7 -x 60 -w 10 -i 30 luffychao  
Adjusting aging data for user luffychao.  
passwd: Success  
  
[root@luffycity ~]# passwd -S luffychao  
luffychao PS 2019-10-17 7 60 10 30 (Password set, SHA512 crypt.)
```

批量更新密码命令

1. 查看当前机器的用户信息
`tail /etc/passwd`
2. 批量改密码，ctrl+d结束输入
`[root@pylinux ~]# chpasswd`

```
yu:123
py:456
testyu:789
```

用户查询相关命令

id命令

id命令用于检查用户和组以及对应的UID，GID等信息

```
[root@pylinux ~]# id yu
uid=1000(yu) gid=1000(yu) 组=1000(yu)

[root@pylinux ~]# id -u yu      #显示用户id
1000
[root@pylinux ~]# id -g yu      #显示组id
1000
[root@pylinux ~]# id -un yu     #显示用户名
yu
[root@pylinux ~]# id -gn yu     #显示组名
yu
```

whoami、who、w、last、lastlog

whoami显示可用于查看当前登录的用户，我是谁

```
[root@pylinux ~]# whoami
root
```

w命令显示当前以登录的用户

```
[root@pylinux ~]# w
04:15:01 up 15 days, 18:03,  1 user,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
root      pts/0    122.71.x5.xx  04:05   5.00s  0.07s  0.00s w
```

1. 显示当前系统时间、系统从启动到运行的时间、系统运行中的用户数量和平均负载（1、5、15分钟平均负载）
2. 第二行信息

user: 用户名

tty: 用户使用的终端号

from: 表示用户从哪来，远程主机的ip信息

login: 用户登录的时间和日期

IDLE: 显示终端空闲时间

JCPU: 该终端所有进程以及子进程使用系统的总时间

PCPU: 活动进程使用的系统时间

WHAT: 用户执行的进程名称

who

```
[root@pylinux ~]# who
root      pts/0      2018-07-12 04:05 (122.71.x5.xx)
```

名称	用户终端	用户登录的系统时间	从哪来的机器ip
root	pts/0	2018-07-12 04:05	(122.71.x5.xx)

last、lastlog命令查看用户详细的登录信息

案例

#last命令显示已登录的用户列表和登录时间

```
[root@pylinux ~]# last
```

root	pts/0	122.71.x5.xx	Thu Jul 12 04:05	still logged in
root	pts/0	122.71.x5.xx	Thu Jul 12 04:02 - 04:05	(00:02)
root	pts/1	122.71.x5.xx	Wed Jul 11 16:56 - 16:57	(00:00)

wtmp begins Sun Jul 8 06:23:25 2018

lastlog命令显示当前机器所有用户最近的登录信息

```
[root@pylinux ~]# lastlog
```

用户名	端口	来自	最后登陆时间
root	pts/0	122.71.65.73	四 7月 12 04:05:09 +0800 2018
bin			**从未登录过**
yu	pts/0		四 7月 12 04:05:51 +0800 2018
epmd			**从未登录过**
rabbitmq			日 9月 29 03:42:01 +0800 2019
py	pts/0		四 7月 12 04:06:02 +0800 2018
testyu			**从未登录过**

Linux用户身份切换命令

su命令



su命令用于切换到指定用户

语法

su(选项)(参数)

选项

- c<指令>或--command=<指令>: 执行完指定的指令后, 即恢复原来的身份;
- f或--fast: 适用于csh与tsch, 使shell不用去读取启动文件;
- l或--login: 改变身份时, 也同时变更工作目录, 以及HOME, SHELL, USER, logname。此外, 也会变更PATH变量;
- m, -p或--preserve-environment: 变更身份时, 不要变更环境变量;
- s<shell>或--shell=<shell>: 指定要执行的shell;
- help: 显示帮助;
- version: 显示版本信息。

案例

root切换普通用户, 无须密码

第一种 (不推荐)

su 用户

```
[root@luffycity ~]# su chaoge          #直接切换, 没有完全切换环境变量

[chaoge@luffycity root]$ env|egrep "USER|MAIL|PWD|LOGNAME"
USER=chaoge
MAIL=/var/spool/mail/root
PWD=/root
LOGNAME=chaoge
```

第二种 (标准切换法)

su - 用户

```
#用户切换
[root@luffycity ~]# su - chaoge
[chaoge@luffycity ~]$ pwd
/home/chaoge

[chaoge@luffycity ~]$ env|egrep "USER|MAIL|PWD|LOGNAME"
USER=chaoge
MAIL=/var/spool/mail/chaoge
PWD=/home/chaoge
LOGNAME=chaoge
```

普通用户切换其他用户, 需要输入用户密码

```
[chaoge@luffycity root]$ su - root
Password:
```

```
Last login: Thu Oct 17 09:39:35 CST 2019 from 192.168.178.1 on pts/1
[root@luffycity ~]#
```

visudo命令

visudo用于编辑/etc/sudoers文件，且提供语法检测，用于配置sudo命令

给chaoge用户配置sudo使用权

```
1.直接输入visudo命令，相当于打开vim /etc/sudoers
找到如下行
89 ## The COMMANDS section may have other options added to it.
90 ##
91 ## Allow root to run any commands anywhere
92 root    ALL=(ALL)        ALL

2.添加你想让执行sudo命令的用户
89 ## The COMMANDS section may have other options added to it.
90 ##
91 ## Allow root to run any commands anywhere
92 root    ALL=(ALL)        ALL
93 chaoge  ALL=(ALL)        ALL

3.保存退出，使用vim/vi的模式，此时已经可以用chaoge用户，使用sudo命令了
```

sudo配置文件

用户或组	机器= (角色)	允许执行命令
User	machine=	Commands
oldboy	ALL=(ALL)	/usr/sbin/useradd、/usr/sbin/userdel

配置sudo目的在于即能让运维方便干活（权限不足问题），又不威胁系统安全（权限把控）

sudo命令



sudo命令用来以其他身份来执行命令，预设的身份为root。在 /etc/sudoers 中设置了可执行sudo指令的用户。

普通用户不需要root密码即可用root权限执行命令。

语法
sudo(选项)(参数)
选项
-b: 在后台执行指令;
-h: 显示帮助;
-H: 将HOME环境变量设为新身份的HOME环境变量;
-k: 结束密码的有效期限, 也就是下次再执行sudo时便需要输入密码;。
-l: 列出目前用户可执行与无法执行的指令;
-p: 改变询问密码的提示符号;
-s<shell>: 执行指定的shell;
-u<用户>: 以指定的用户作为新的身份。若不加此参数, 则预设以root作为新的身份;
-v: 延长密码有效期限5分钟;
-V : 显示版本信息。

案例

使用sudo命令

```
[root@luffycity ~]# su - chaoge
Last login: Thu Oct 17 16:55:26 CST 2019 on pts/0

[chaoge@luffycity ~]$ ls /root/
ls: cannot open directory /root/: Permission denied

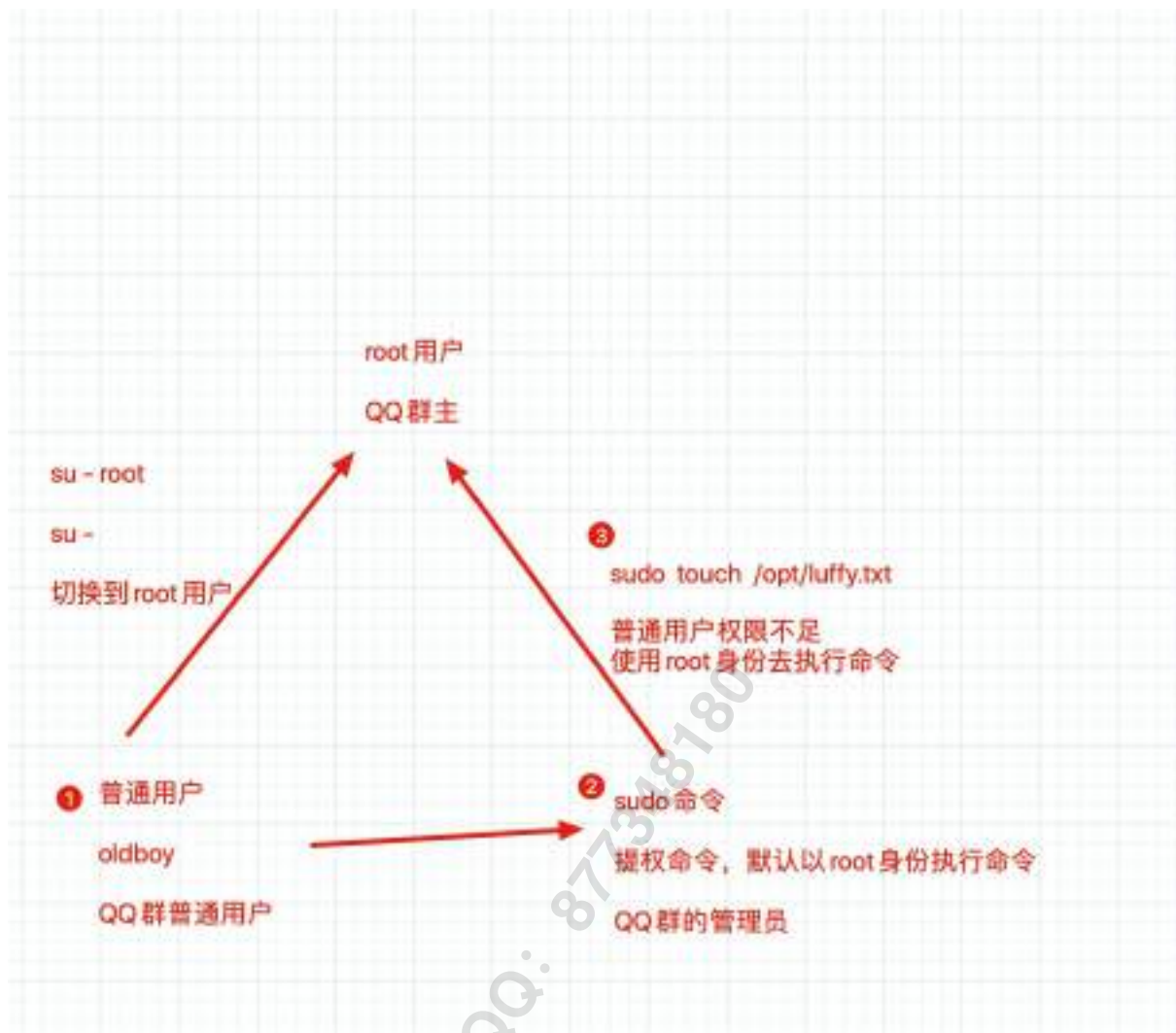
[chaoge@luffycity ~]$ sudo ls /root

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for chaoge:
123 123456 456 anaconda-ks.cfg
```

- 配置了/etc/sudoers文件后, 可以对用户命令提权, sudo 命令
- 想要切换root执行操作, 可以sudo su - , 需要输入当前用户密码



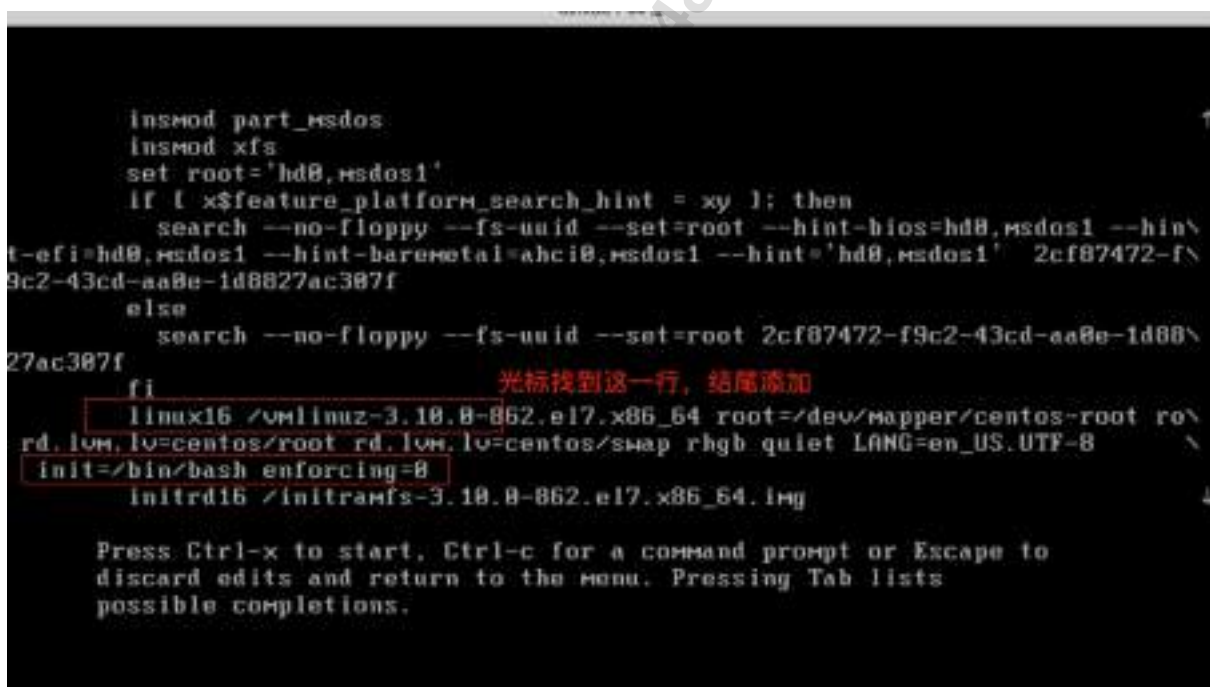
Centos7忘记root密码怎么办

第一，重启linux，进入系统的GRUB菜单界面，按下小写字母e进入编辑界面



第二，按下方向键，找到以字符串Linux16开头的行，光标移动到结尾，然后输入`init=/bin/bash enforcing=0`

代表登录系统后，加载bash解释器，然后关闭selinux



第三，按下`ctrl+x`组合键以单用户模式启动Linux

第四，输入如下命令，重新挂载根目录，进入可写状态，因为默认单用户模式是只读的

```
mount -o rw,remount / #重新挂载
passwd root          #修改密码
exec /sbin/init      #重启
```

第五，更改完毕密码后,重启系统

如果在第二步，没有添加enforcing参数的话，则需要额外的再添加命令

```
touch /.autorelabel
```

告诉系统下次启动重新标记系统所有相关文件，因为selinux在开启时，修改root密码引发安全报错

如果加了enforcing=0，则代表关闭了selinux，不需要再创建 /.autorelabel文件了

第六，重启机器，验证新的密码

```
yumac:~ yuchao$ ssh root@192.168.178.180
```

```
root@192.168.178.180's password:
```

Last login: Tue Nov 5 11:33:32 2019

/ **

* ——神兽出没—— *

[illegible]

老师QQ: 877348180

Linux文件权限



文件权限

```

r    read可读，可以用cat等命令查看
w    write写入，可以编辑或者删除这个文件
x    executable    可以执行
-    没有权限

```

文件夹权限

权限这里测试不要用root实验！！！！root太牛逼了

请用普通用户执行！！！！测试文件、文件夹权限操作，请用普通用户！

```

r    可以对此目录执行ls列出所有文件
w    可以在这个目录创建文件
x    可以cd进入这个目录，或者查看详细信息

```

如何知道我和文件之间的权限

- 你是什么身份，皇帝还是老百姓
- 文件属于你吗，还是属于你们家
- 你和这个文件具体的关系

文件权限与数字转化

权限分配	文件所有者			文件所属组			其他用户		
权限项	读	写	执行	读	写	执行	读	写	执行
字符表示	r	w	x	r	w	x	r	w	x
数字表示	4	2	1	4	2	1	4	2	1

```

rwx权限表示
r read 读取 4
w write 写 2
x execute 执行 1

```

-	无权限	0
---	-----	---

常见权限对应

444	r--r--r--
600	rw-----
644	rw-r--r--
666	rw-rw-rw-
700	rwX-----
744	rwXr--r--
755	rwXr-Xr-X
777	rwXrwxrwx

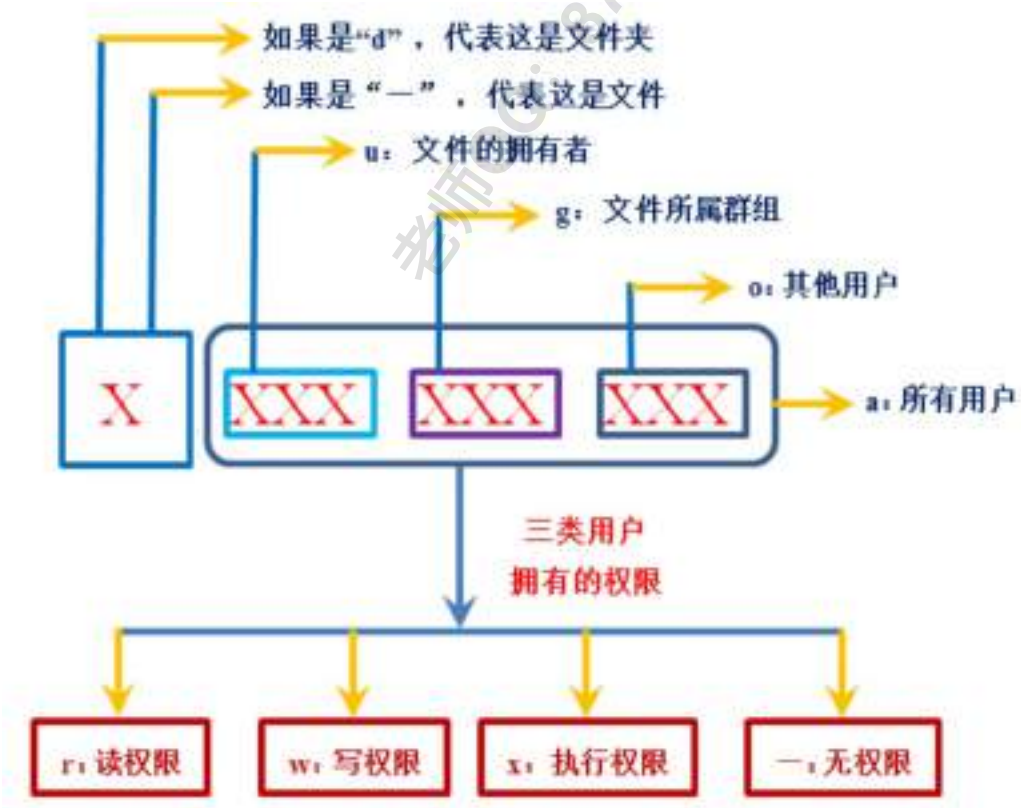
chmod命令

chmod命令用来变更文件或目录的权限。

在UNIX系统家族里，文件或目录权限的控制分别以读取、写入、执行3种一般权限来区分，另有3种特殊权限可供运用。

用户可以使用chmod指令去变更文件与目录的权限，设置方式采用文字或数字代号皆可。

符号连接的权限无法变更，如果用户对符号连接修改权限，其改变会作用在被连接的原始文件。



权限范围:
操作对象

u 文件属主权限
g 同组用户权限
o 其它用户权限
a 所有用户（包括以上三种）

权限设定

+ 增加权限
- 取消权限
= 唯一设定权限

权限类别

r 读权限
w 写权限
x 执行权限
X 表示只有当该档案是个子目录或者该档案已经被设定过为可执行。
s 文件属主和组id
i 给文件加锁，使其它用户无法访问

r-->4

w-->2

x-->1

案例

设置文件所有人可读

```
[root@luffycity tmp]# chmod ugo+r data.zip    #第一写法  
[root@luffycity tmp]# chmod a+r data.zip      #第二种
```

文件属主和属组内成员可写，其他人不可写

```
[root@luffycity tmp]# chmod ug+w,o-w data.zip
```

文件属主可以执行

```
[root@luffycity tmp]# chmod u+x data.zip
```

设置data目录所有资料，任何人可读取

```
[root@luffycity tmp]# chmod -R a+r data/
```

所有人都可以读、写、执行

```
chmo 777 filename.txt
```

使用数字权限转化

```
chmod 755 alex.txt
chmod 644 alex.txt
chmod 777 alex.txt
chmod 0 alex.txt #所有人无权限
chmod 65 alex.txt #属主是0权限，group权限6，其他人权限5
chmod 4 alex.txt #其实是004，只有其他人可读
```

实际效果，rwx对于文件的操作

权限	效果
r	可以读取、阅读文件内容
w	可以新增、修改文件内容，删除文件的权限由父目录权限控制，删除权限又父目录的w控制
x	1.可读取文件内容并执行 如 . luffy.sh source luffy.sh sh luffy.sh 2.普通用户必须对文件有r权限才可执行 3.root只需要任意u,g,o存在x权限即可执行

rwx对于目录的权限

权限	效果
r	1.可以浏览目录下内容，如ls 2.没有x无法进入目录 3.可以看到目录下文件名，无法访问文件信息（内容，权限）
w	【有x权限前提下】文件夹有增加、删除、修改文件夹中文件名的权限
x	1.进入文件夹，如cd 2.没有r无法查看列表 3.没有w无法创建文件

chown命令

修改文件属主、数组信息

```
语法：

chown alex test.txt #文件属于alex
chown :组 test.txt #修改文件属组
chown 用户:组 #修改

参数
```

-R, --recursive	递归处理所有的文件及子目录
-v, --verbose	为处理的所有文件显示诊断信息

案例

```
chown alex:alex nginx.conf      #将nginx.conf所属用户和组改为alex,alex
chown -R alex:alex data         #将data目录下所有内容属主和属组改为alex

chown .alex test.txt           #test.txt的属组改为alex
```

chgrp命令

chgrp命令用来改变文件或目录所属的用户组。

该命令用来改变指定文件所属的用户组。

其中，组名可以是用户组的id，也可以是用户组的组名。

案例

```
-c或—changes: 效果类似“-v”参数，但仅回报更改的部分；
-f或--quiet或—silent: 不显示错误信息；
-h或--no-dereference: 只对符号连接的文件作修改，而不是该其他任何相关文件；
-R或—recursive: 递归处理，将指令目录下的所有文件及子目录一并处理；
-v或—verbose: 显示指令执行过程；
--reference=<参考文件或目录>: 把指定文件或目录的所属群组全部设成和参考文件或目录的所属群组相同；
```

```
chgrp -R alex /data    #把/data目录下所有文件的属组改为alex
```

umask命令

umask命令用来限制新文件权限的掩码。

也称之为遮罩码，防止文件、文件夹创建的时候，权限过大

当新文件被创建时，其最初的权限由文件创建掩码决定。

用户每次注册进入系统时，**umask命令**都被执行，并自动设置掩码改变默认值，新的权限将会把旧的覆盖。

- umask默认配置在/etc/profile 61-64行

umask值就是指“Linux文件的默认属性需要减掉的权限”。

比如Linux普通文件的最大默认属性是666，目录文件的最大属性是777。但是我们不想要用户在新建立文件时，文件的属性是666或777，那么我们就需要设置umask值。

Linux系统预置的umask值是022，那么用户在新建立普通文件时，普通文件的属性就是666-022=644，新建立目录文件时，目录文件的属性就是777-022=755。

语法参数

- S: 以字符的形势显示当前的掩码。
- p: 带umask开头以数字的形势显示当前掩码

计算umask文件权限

系统默认umask数值对应的权限

记住公式

默认的文件、文件夹权限，减去umaks的值等于最终的权限值

文件最大777 文件夹最大777

默认umask值,root和普通用户不一样

```
#root用户umask
[root@localhost ~]# umask          #查看当前用户预设权限
0022

[root@localhost ~]# umask -S        #以字母的形势显示权限，7-0 7-2 7-2
u=rwx,g=rx,o=rx

[root@localhost ~]# umask -p
umask 0022

#普通用户umask
[chaoge@luffycity tmp]$ umask
0002
[chaoge@luffycity tmp]$ umask -S
u=rwx,g=rwx,o=rx
[chaoge@luffycity tmp]$ umask -p
umask 0002
```

案例

root登录系统默认创建文件，文件夹的权限

```
[root@luffycity tmp]# touch file.txt          #文件是644
[root@luffycity tmp]# mkdir luffy             #文件夹是755
[root@luffycity tmp]# ll
total 0
-rw-r--r--. 1 root root 0 Oct 18 11:19 file.txt
drwxr-xr-x. 2 root root 6 Oct 18 11:19 luffy
```

普通用户


```
[chaoge@luffycity tmp]$ touch chao.txt #权限664
[chaoge@luffycity tmp]$ mkdir chao #权限775
[chaoge@luffycity tmp]$ ll
total 0
drwxrwxr-x. 2 chaoge chaoge 6 Oct 18 11:27 chao
-rw-rw-r--. 1 chaoge chaoge 0 Oct 18 11:27 chao.txt
```

- 操作系统创建文件，默认最大权限是666(-rw-rw-rw-)
- 创建普通文件权限是

```
#文件默认起始权限666，减去umask默认值022，等于创建文件的权限
666
022
-----
644 #系统创建文件权限默认是644 - rw- r-- r--
```

- umask值不得大于6
- umask计算出的文件权限不得拥有执行权限x，如果计算结果中有执行权限，则需要将其加一

```
#文件默认权限666，假设umask值设置为045
666
045
-----
621 #存在执行权限，必须+1，默认文件不得执行
----
622 #最终文件权限
```

演示

```
#临时修改默认umask变量值
[root@luffycity tmp]# umask 045
[root@luffycity tmp]# umask
0045

#创建文件，查看权限，此时默认已经是644的权限
[root@luffycity tmp]# touch chao.txt
[root@luffycity tmp]# ll
total 0
-rw--w--w-. 1 root root 0 Oct 18 11:43 chao.txt
```

计算目录umask权限

- 系统创建目录默认权限最大777

```
#目录最大权限777，减去umask值，得到目录创建后的权限
777
```

```
022
-----
755
```

案例

```
[root@luffycity tmp]# umask
022

[root@luffycity tmp]# mkdir chao
[root@luffycity tmp]# ll
total 0
drwxr-xr-x. 2 root root 6 Oct 18 11:54 chao
```

Linux默认权限

- 文件644 rw- r-- r--
- 目录655rwx r-x r-x



chattr命令

chattr命令用于更改文件的扩展属性，比chmod更改的rwx权限更底层

参数

a: 只能向文件中添加数据，不得删除
-R: 递归更改目录属性
-V: 显示命令执行过程

模式

+ 增加参数
- 移除参数
= 更新为指定参数
A 不让系统修改文件最后访问时间
a 只能追加文件数据，不得删除
i 文件不能被删除、改名、修改内容

案例

```
[root@luffycity tmp]# chattr +i chao.txt

[root@luffycity tmp]# lsattr chao.txt
----i----- chao.txt

[root@luffycity tmp]# rm -rf chao.txt
rm: cannot remove 'chao.txt': Operation not permitted

[root@luffycity tmp]# echo qqqq > chao.txt

[root@luffycity tmp]# chattr -i chao.txt

[root@luffycity tmp]# rm -rf chao.txt
```

-a参数，只能追加内容

```
[root@luffycity tmp]# chattr +a test.txt
[root@luffycity tmp]# lsattr test.txt
-----a----- test.txt
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# echo "qwe" > test.txt
-bash: test.txt: Operation not permitted
[root@luffycity tmp]# echo "qwe" >> test.txt
[root@luffycity tmp]#
[root@luffycity tmp]#
[root@luffycity tmp]# cat test.txt
happy
qweqe
qwe
```

lsattr命令

lsattr命令用于查看文件的第二扩展文件系统属性，结合**chattr**一起用

```
-R      递归地列出目录以及其下内容的属性。
-V      显示程序版本。
-a      列出目录中的所有文件, 包括以`.`开头的文件的属性。
-d      以列出其它文件的方式那样列出目录的属性, 而不列出其下的内容。
-v      显示文件版本。
```

案例

```
[root@localhost ~]# chattr +i nginx.conf          #设置该文件不能进行任何形
势的修改

[root@localhost ~]# mv nginx.conf nginx.conf_bak
mv: 无法将"nginx.conf" 移动至"nginx.conf_bak": 不允许的操作

[root@localhost ~]# lsattr nginx.conf              #查看chattr设置的权限
----i-----e- nginx.conf

[root@localhost ~]# chattr -i nginx.conf           #取消-i的权限设置

[root@localhost ~]# lsattr nginx.conf              #查看chattr设置的权限
-----e- nginx.conf
```

[TOC]

Linux通配符

通配符

就是键盘上的一些特殊字符，可以实现特殊的功能，例如模糊搜索一些文件。

文件名	通配符	模糊匹配
luffy	*	luffyalex
		luffychao
		luffycunzhang
oldboy	?	oldboy1
		oldboy2
		oldboyz
		oldboyx

利用通配符可以更轻松的处理字符信息

常见通配符

符号	作用
*	匹配任意，0或多个字符，字符串
?	匹配任意1个字符，有且只有一个字符
符号集合	匹配一堆字符或文本
[abcd]	匹配abcd中任意一个字符，abcd也可以是不连续任意字符
[a-z]	匹配a到z之间任意一个字符，要求连续字符，也可以连续数字，匹配[1-9]
[!abcd]	不匹配括号中任意一个字符，也可以书写[!a-d]，同于写法
[^abcd]	同上，！可以换成 ^

特殊通配符

符号	作用
[[:upper:]]	所有大写字母
[[:lower:]]	所有小写字母
[[:alpha:]]	所有字母
[[:digit:]]	所有数字

<code>[:alnum:]</code>	所有的字母和数字
<code>[:space:]</code>	所有的空白字符
<code>[:punct:]</code>	所有标点符号

案例

1. 找出根目录下最大文件夹深度是3，且所有以l开头的文件，且以小写字母结尾，中间出现任意一个字符的文本文件，

```
[root@chaogelinux luffy_data]# find / -maxdepth 3 -type f -name "l?[:lower:]"
/usr/bin/ldd
/usr/bin/lua
/usr/sbin/lvm
/usr/sbin/lid
```

2. 找出/tmp下以任意一位数字开头，且以非数字结尾的文件

```
[root@chaogelinux luffy_data]# find /tmp -type f -name '[0-9][^0-9]'
```

3. 显示/tmp下以非字母开头，后面跟着一个字母以及其他任意长度的字符的文件

```
find /tmp -type f -name "[^a-zA-Z][a-z]*"
```

4. mv/tmp/下，所有以非字母开头的文件，复制到/tmp/allNum/目录下

```
mv /tmp/[^a-zA-Z]* /tmp/allNum/
```

5. 复制/tmp目录下所有的.txt文件结尾的文件，且以y、t开头的文件，放入/data目录

```
[root@chaogelinux tmp]# cp -r /tmp/[y,t]*.txt /data/
```

*号

准备数据如下

```
[root@pylinux tmp]# ls
chaoge.sh  chaoge.txt  oldboy.txt  oldgirl.txt  oldluffy.txt  yu.sh
```

1. 匹配所有的txt文本

```
[root@pylinux tmp]# ls
chaoge.sh  chaoge.txt  oldboy.txt  oldgirl.txt  oldluffy.txt  yu.sh
```

2. 匹配所有的sh脚本

```
[root@pylinux tmp]# ls *.sh
chaoge.sh  yu.sh
```

3. 查看所有的old开头的文件

```
[root@pylinux tmp]# ls old*
```

```
oldboy.txt  oldgirl.txt  oldluffy.txt
```

? 号

1.匹配一个任意字符

```
[root@pylinux tmp]# ls *.s?  
chaoge.sh  yu.sh
```

2.匹配两个，三个任意字符

```
[root@pylinux tmp]# ls old????.txt  
oldboy.txt  
[root@pylinux tmp]# ls old?????.txt  
oldgirl.txt
```

[abc]

1.匹配[abc]中的一个字符

```
[root@pylinux tmp]# ls  
a.txt  b.txt  chaoge.sh  chaoge.txt  l.txt  oldboy.txt  oldgirl.txt  oldluffy.txt  
yu.sh  
  
[root@pylinux tmp]# ls [abc].txt  
a.txt  b.txt
```

2.匹配如下类型文件

olda.txt oldb.txt oldc.txt*

```
[root@pylinux tmp]# ls old[abcd]*.txt  
oldboy.txt
```

[a-z]作用

[]中括号里面的a-z，表示从a到z之间的任意一个字符，a-z要连续，也可以用连续的数字，如[1-9]

```
[root@pylinux tmp]# ls
a.txt b.txt chaoge.sh chaoge.txt chao.txt l.txt oldboy.txt oldgirl.txt oldluffy.txt yu.sh yu.txt
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# ls [a-z].txt 匹配任意1个字符
a.txt b.txt l.txt
[root@pylinux tmp]# ls [a-z]?.txt 匹配任意2个字符
yu.txt
[root@pylinux tmp]# ls [a-z]???.txt
ls: 无法访问[a-z]???.txt: 没有那个文件或目录 匹配任意3个字符, 没有这个文件
[root@pylinux tmp]# ls [a-z]????.txt 匹配任意4个字符
chao.txt
```

[!abcd]

```
[^abcd]
[^a-d]
效果同上
```

除了abcd四个字符以外的任意一个字符

```
[root@pylinux tmp]# ls *.txt
ab.txt b.txt chao.txt d.txt l.txt oldgirl.txt yu.txt
a.txt chaoge.txt c.txt e.txt oldboy.txt oldluffy.txt
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# ls [!abcd]*.txt 找出除了以abcd字符开头的txt文件
e.txt l.txt oldboy.txt oldgirl.txt oldluffy.txt yu.txt
```

```
[root@pylinux tmp]# ls *.txt
1.txt 35.txt a.txt chaoge.txt c.txt e.txt oldboy.txt oldluffy.txt
2.txt ab.txt b.txt chao.txt d.txt l.txt oldgirl.txt yu.txt
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# ls [!a-d].txt 找出除了a.txt b.txt c.txt d.txt以外的 任意一个字符.txt
1.txt 2.txt e.txt l.txt
[root@pylinux tmp]#
[root@pylinux tmp]# ls [^1-9].txt 找出除了1到9之间, 单个字符的.txt
a.txt b.txt c.txt d.txt e.txt l.txt
```

结合find命令使用

```
[root@pylinux tmp]# find /tmp -type f -name "[a-z].txt" #找出a到z之间单个字符
的文件
/tmp/b.txt
/tmp/e.txt
/tmp/a.txt
/tmp/l.txt
/tmp/d.txt
/tmp/c.txt

[root@pylinux tmp]# find /tmp -type f -name "[!a-d].txt" #找出除了a到d之间单
个字符的文件
```



```

/tmp/e.txt
/tmp/2.txt
/tmp/1.txt
/tmp/1.txt

[root@pylinux tmp]# find /tmp -type f -name "?*.txt"           #找出所有单个字符的文件
/tmp/b.txt
/tmp/e.txt
/tmp/2.txt
/tmp/a.txt
/tmp/1.txt
/tmp/1.txt
/tmp/d.txt
/tmp/c.txt

[root@pylinux tmp]# find /tmp -type f -name "*.txt"           #找出所有的txt文本

```

Linux特殊符号

路径相关

符号	作用
~	当前登录用户的家目录
-	上一次工作路径
.	当前工作路径，或隐藏文件 .chaogetxt
..	上一级目录

波浪线案例

```

[root@pylinux tmp]# cd ~
[root@pylinux ~]# pwd
/root

[yu@pylinux ~]$
[yu@pylinux ~]$ pwd
/home/yu

```

短横杠案例

```

[root@pylinux opt]# cd /tmp
[root@pylinux tmp]# cd -
/opt
[root@pylinux opt]# cd -
/tmp

```

```
[root@pylinux tmp]# echo $OLDPWD
/opt
```

点案例

```
[root@pylinux tmp]# find . -name "*.sh"
./yu.sh
./chaoge.sh
```

点点案例

```
[root@pylinux tmp]# mkdir ../opt/超哥nb
[root@pylinux tmp]#
[root@pylinux tmp]# ls /opt/
超哥nb

[root@pylinux home]# ls -a
.  ..  py  testyu  yu
```

特殊引号

在linux系统中，单引号、双引号可以用于表示字符串

名称	解释
单引号 "	所见即所得，强引用，单引号中内容会原样输出
双引号 "	弱引用，能够识别各种特殊符号、变量、转义符等，解析后再输出结果
没有引号	一般连续字符串、数字、路径可以省略双引号，遇见特殊字符，空格、变量等，必须加上双引号
反引号 `	常用于引用命令结果，同于\$(命令)

反引号案例

用反引号进行命令解析

```
[root@pylinux tmp]# date +%F
2019-11-05
[root@pylinux tmp]# touch `date +%F`.txt    #创建文件，名字是当前时间格式
[root@pylinux tmp]# ls
2019-11-05.txt

[root@pylinux tmp]# echo date
date
[root@pylinux tmp]# echo `date`           #反引号中命令会被执行
```

2019年 11月 05日 星期二 16:29:28 CST

```
[root@pylinux tmp]# which cat
/usr/bin/cat
[root@pylinux tmp]#
[root@pylinux tmp]# ls -l `which cat`          #反引号中命令被执行
-rwxr-xr-x. 1 root root 54080 4月 11 2018 /usr/bin/cat
```

双引号案例

当输出双引号内所有内容时，内容中有命令需要用反引号标记

```
[root@pylinux tmp]# echo "date"
date
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# echo "`date`"
2019年 11月 05日 星期二 16:30:42 CST

[root@pylinux tmp]# echo "今天是星期 `date +%w`"
今天是星期 2

[root@pylinux tmp]# echo "今年是$(date +%Y)年"
今年是2019年
```

单引号案例

单引号中内容是强引用，保持原样输出

```
[root@pylinux tmp]# echo "今天日期是 `date +%F`"          #双引号可以
今天日期是 2019-11-05

[root@pylinux tmp]# echo '今天日期是 `date +%F`'          #单引号不可以
今天日期是 `date +%F`
```

无引用案例

没有引号，很难确定字符串的边界，且linux命令是以空格区分的

建议用双引号代替不加引号

```
[root@pylinux tmp]# echo "今天是 `date +%Y`年"
今天是 2019年

[root@pylinux tmp]# echo 今天是 `date +%Y`年
今天是 2019年
```

```
[root@pylinux tmp]# ls "/tmp"
2019-11-05.txt
[root@pylinux tmp]#
[root@pylinux tmp]# ls /tmp
2019-11-05.txt
```

输出重定向特殊符号

输入设备

- 键盘输入数据
- 文件数据导入

输出设备

- 显示器、屏幕终端
- 文件

数据流



程序的数据流：

- 输入流：<---标准输入（stdin），键盘
- 输出流：-->标准输出（stdout），显示器，终端
- 错误输出流：-->错误输出（stderr）

文件描述符

在Linux系统中，一切设备都看作文件。

而每打开一个文件，就有一个代表该打开文件的文件描述符。

程序启动时默认打开三个I/O设备文件：

- 标准输入文件stdin，文件描述符0
- 标准输出文件stdout，文件描述符1
- 标准错误输出文件stderr，文件描述符2

符号	特殊符号	简介
标准输入stdin	代码为0，配合< 或<<	数据流从右向左

标准输出stdout	代码1, 配合>或>>	数据从左向右
标准错误stderr	代码2, 配合>或>>	数据从左向右
重定向符号		数据流是箭头方向
标准输入重定向	0< 或 <	数据一般从文件流向处理命令
追加输入重定向	0<<或<<<	数据一般从文件流向处理命令
标准输出重定向	1>或>	正常输出重定向给文件, 默认覆盖
标准输出追加重定向	1>>或>>>	内容追加重定向到文件底部, 追加
标准错误输出重定向	2>	讲标准错误内容重定向到文件, 默认覆盖
标准错误输出追加重定向	2>>	标准错误内容追加到文件底部

错误输出

```
[root@chaogelinux tmp]# ls yyy
ls: 无法访问yyy: 没有那个文件或目录
[root@chaogelinux tmp]#
[root@chaogelinux tmp]# ls yyy > cuowu.txt
ls: 无法访问yyy: 没有那个文件或目录
[root@chaogelinux tmp]# ls yyy >> cuowu.txt
ls: 无法访问yyy: 没有那个文件或目录
[root@chaogelinux tmp]# ls yyy 2> cuowu.txt
[root@chaogelinux tmp]#
[root@chaogelinux tmp]#
[root@chaogelinux tmp]# cat cuowu.txt
ls: 无法访问yyy: 没有那个文件或目录
```

特殊重定向, 合并重定向

- 2>&1 把标准错误, 重定向到标准输出

把命令的执行结果写入文件, 标准错误当做标准输出处理, 也写入文件

- Command > /path/file 2>&1

```
echo "I am oldboy" 1>>oldboy.txt 2>>oldboy.txt

echo "I am oldboy" >> /dev/null 2>&1 # 命令已经将结果重定向到/dev/null, 2
>&1符号也会将标准错误输出到/dev/null, /dev/null是一个黑洞, 只写文件
```

输入重定向

数据流输入

```
[root@chaogelinux tmp]# cat < yu2.txt
```

我是 yu2，你是谁，想偷看我？

#mysql数据导入

```
mysql -uroot -p < db.sql
```

```
[root@chaogelinux ~]# cat chaoge.txt
```

```
a b c d e f g
```

```
[root@chaogelinux ~]# tr -d 'a-c' < chaoge.txt
```

```
d e f g
```

```
[root@chaogelinux ~]# wc -l < chaoge.txt
```

```
1
```

其他特殊符号

符号	解释
;	分号，命令分隔符或是结束符
#	1.文件中注释的内容 2.root身份提示符
	管道符，传递命令结果给下一个命令
\$	1.\$变量，取出变量的值 2.普通用户身份提示符
\	转义符，将特殊含义的字符还原成普通字符
{ }	1.生成序列 2.引用变量作为变量与普通字符的分割

； 号

- 表示命令的结束
- 命令间的分隔符
- 配置文件的注释符

```
[root@pylinux tmp]# pwd;ls          #执行两条命令
/tmp
2019-11-05.txt  oldboy.txt  txt
```

#号

- 注释行

```
# nginx.conf
```

```
#user  nobody;
```

```
worker_processes  1;
```

```
#error_log  logs/error.log;
```

```
#error_log  logs/error.log  notice;
```

```
#error_log logs/error.log info;

#pid logs/nginx.pid;
```

|号

比如生活中的管道，能够传输物质

Linux管道符 | 用于传输数据，对linux命令的处理结果再次处理，直到得到最终结果

```
[root@pylinux ~]# ifconfig |grep inet
    inet 10.141.32.137 netmask 255.255.192.0 broadcast 10.141.63.255
    inet 127.0.0.1 netmask 255.0.0.0

[root@pylinux tmp]# ls | grep .txt
2019-11-05.txt
oldboy.txt

[root@pylinux tmp]# netstat -tunlp|grep 22
tcp        0      0 0.0.0.0:22          0.0.0.0:*           LISTEN      140
1/sshd

[root@pylinux tmp]# netstat -tunlp|grep 3306

[root@pylinux tmp]# netstat -tunlp|grep 80

[root@pylinux tmp]# ps aux|grep python

[root@pylinux tmp]# ps aux|grep mariadb
```

能一次出结果的命令，尽量不要二次过滤，效率并不高

\$符

Linux系统命令行中，字符串前加\$符，代表字符串变量的值

```
[root@pylinux tmp]# echo OLDPWD
OLDPWD
[root@pylinux tmp]# echo $OLDPWD
/root
[root@pylinux tmp]# echo PATH
PATH
[root@pylinux tmp]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/opt/python37/bin:/root/bin
[root@pylinux tmp]# name="超哥带你学linux"
[root@pylinux tmp]# echo name
name
```

```
[root@pylinux tmp]# echo $name
超哥带你学linux
```

{}符

1.生成序列，一连串的文本

```
[root@pylinux tmp]# echo {1..100}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 5
9 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
87 88 89 90 91 92 93 94 95 96 97 98 99 100

[root@pylinux tmp]# echo {a..i}
a b c d e f g h i

[root@pylinux tmp]# echo {i..a}
i h g f e d c b a

[root@pylinux tmp]# echo {1..9}
1 2 3 4 5 6 7 8 9

[root@pylinux tmp]# echo {o,l,d}
o l d
```

2.利用{}快速备份文件

```
[root@pylinux tmp]# cp /etc/sysconfig/network-scripts/ifcfg-eth0{,.ori}
```

3.将变量括起来作为变量的分隔符

```
[root@pylinux tmp]# echo $week
3
[root@pylinux tmp]# echo "$week_oldboy"           #输出为空，系统人为week_oldboy是整个变量

[root@pylinux tmp]# echo "${week}_oldboy"          #花括号中才会识别是变量，作了分割
3_oldboy
```

逻辑操作符

逻辑符既可以在linux系统中直接用，也可以在Bash脚本中用

命令	解释	
&&	前一个命令成功，再执行下一个命令	
		前一个命令失败了，再执

		行下一个命令
!	1.在bash中取反 2.在vim中强制性 3.历史命令中 !ls找出最近一次以ls开头的命令	

1.&&案例

```
[root@pylinux tmp]# ls && cd /opt && pwd          #执行成功会继续下一个命令
2019-11-05.txt  oldboy.txt  txt
/opt
```

```
[root@pylinux opt]# ls /tmpp && cd /tmp          #执行失败就结束
ls: 无法访问/tmpp: 没有那个文件或目录
```

1. ||案例

```
[root@pylinux opt]# ls /tmpp || cd /tmp          #执行失败才会继续下一个命令
ls: 无法访问/tmpp: 没有那个文件或目录
[root@pylinux tmp]#
```

```
[root@pylinux tmp]# cd /opt || cd /root          #执行成功则不会继续下一个命令
[root@pylinux opt]#
```

1. 感叹号

2. 取反

```
[root@pylinux tmp]# ls [a-f]*
a b c d e f
[root@pylinux tmp]# ls [!a-f]*          #取反的意思
2019-11-05.txt  g h i j k l m n o oldboy.txt  p q r s t txt u v w
x y z
```

- 感叹号的vim强制退出
- 找出历史命令

[TOC]

bash是什么



- bash是一个命令处理器，运行在文本窗口中，并能执行用户直接输入的命令
- bash还能从文件中读取linux命令，称之为脚本
- bash支持通配符、管道、命令替换、条件判断等逻辑控制语句

bash的特性

- 命令行展开

```
[root@chaogelinux ~]# echo {tom,bob,chaoge,jerry}
tom bob chaoge jerry
```

```
[root@chaogelinux ~]# echo chaoge{666,888}
chaoge666 chaoge888
```

```
[root@chaogelinux ~]# echo chaoge{1..5}
chaoge1 chaoge2 chaoge3 chaoge4 chaoge5
```

```
[root@chaogelinux ~]# echo chaoge{1..10..2}
chaoge1 chaoge3 chaoge5 chaoge7 chaoge9
```

```
[root@chaogelinux ~]# echo chaoge{01..10..2}
chaoge01 chaoge03 chaoge05 chaoge07 chaoge09
```

- 命令别名

```
alias,unalias
```

- 命令历史

```
history
!行号
!! 上一次的命令
```

- 快捷键

```
ctrl + a  移动到行首
ctrl + e  移动到行尾
ctrl + u  删除光标之前的字符
ctrl + k  删除光标之后的字符
ctrl + l  清空屏幕终端内容，同于clear
```

- 命令补全

```
tab键
补全
$PATH中存在的命令
```

- 文件路径补全

```
/opt/chaoge/linux_study
```

Linux正则表达式



正则表达式：Regual Expression, REGEXP

由一类特殊字符及文本字符所编写的模式，其中有些字符不表示其字面意义，而是用于表示控制或通配的功能；

分两类：

基本正则表达式：BRE

扩展正则表达式：ERE

正则表达式的意义

- 处理大量的字符串
- 处理文本

通过特殊符号的辅助，可以让linux管理员快速过滤、替换、处理所需要的字符串、文本，让工作高效。

通常Linux运维工作，都是面临大量带有字符串的内容，如

- 配置文件
- 程序代码
- 命令输出结果
- 日志文件

且此类字符串内容，我们常会有特定的需要，查找出符合工作需要的特定的字符串，因此正则表达式就出现了

- 正则表达式是一套规则和方法
- 正则工作时以单位进行，一次处理一行
- 正则表达式化繁为简，提高工作效率
- linux仅受三剑客（sed、awk、grep）支持，其他命令无法使用

正则表达式应用非常广泛，应用在如Python、Java、Perl等，Linux下普通命令无法使用正则表达式的，只能使用三剑客。

通配符是大部分普通命令都支持的，用于查找文件或目录，而正则表达式是通过三剑客命令在文件（数据流）中过滤内容的

Linux三剑客

文本处理工具，均支持正则表达式引擎

- grep：文本过滤工具，（模式：pattern）工具
- sed：stream editor，流编辑器；文本编辑工具
- awk：Linux的文本报告生成器（格式化文本），Linux上是gawk

正则表达式的分类

Linux三剑客主要分两类

- 基本正则表达式（BRE、basic regular expression）

BRE对应元字符有 `^$.[]*`

- 扩展正则表达式 (ERE、extended regular expression)

ERE在BRE基础上，增加上 `(){}?+|` 等字符

基本正则表达式BRE集合

- 匹配字符
- 匹配次数
- 位置锚定

符号	作用
<code>^</code>	尖角号，用于模式的最左侧，如 <code>"^oldboy"</code> ，匹配以oldboy单词开头的行
<code>\$</code>	美元符，用于模式的最右侧，如 <code>"oldboy\$"</code> ，表示以oldboy单词结尾的行
<code>^\$</code>	组合符，表示空行
<code>.</code>	匹配任意一个且只有一个字符，不能匹配空行
<code>\</code>	转义字符，让特殊含义的字符，现出原形，还原本意，例如 <code>\.</code> 代表小数点
<code>*</code>	匹配前一个字符（连续出现）0次或1次以上，重复0次代表空，即匹配所有内容
<code>.*</code>	组合符，匹配任意长度的任意字符
<code>^.*</code>	组合符，匹配任意多个字符开头的内容
<code>.*\$</code>	组合符，匹配以任意多个字符结尾的内容
<code>[abc]</code>	匹配[]集合内的任意一个字符，a或b或c，可以写 [a-c]
<code>[^abc]</code>	匹配除了^后面的任意字符，a或b或c，^表示对 [abc] 的取反
<code><pattern></code>	匹配完整的内容
<code><或></code>	定位单词的左侧，和右侧，如 <code><chao></code> 可以找出"The chao ge"，缺找不出"yuchao"

扩展正则表达式ERE集合

扩展正则必须用 `grep -E` 才能生效

字符	作用
<code>+</code>	匹配前一个字符1次或多次，前面字符至少出现1次
<code>[:/]+</code>	匹配括号内的 <code>":"</code> 或者 <code>"/"</code> 字符1次或多次
<code>?</code>	匹配前一个字符0次或1次，前面字符可有可无
竖线	表示或者，同时过滤多个字符串
<code>()</code>	分组过滤，被括起来的内容表示一个整体
<code>a{n,m}</code>	匹配前一个字符最少n次，最多m次

a{n,}	匹配前一个字符最少n次
a{n}	匹配前一个字符正好n次
a{,m}	匹配前一个字符最多m次

Tip:

```
grep命令需要使用参数 -E即可支持正则表达式
egrep不推荐使用，使用grep -E替代
grep不加参数，得在特殊字符前面加\"反斜杠，识别为正则
```

grep

全拼：Global search REgular expression and Print out the line.

作用：文本搜索工具，根据用户指定的“模式（过滤条件）”对目标文本逐行进行匹配检查，打印匹配到的行

模式：由正则表达式的 元字符 及 文本字符 所编写出的 过滤条件 ；

```
语法：
grep [options] [pattern] file
命令 参数 匹配模式 文件数据
      -i: ignorecase, 忽略字符的大小写；
      -o: 仅显示匹配到的字符串本身；
      -v, --invert-match: 显示不能被模式匹配到的行；
      -E: 支持使用扩展的正则表达式元字符；
      -q, --quiet, --silent: 静默模式，即不输出任何信息；
```

grep命令是Linux系统中最重要命令之一，功能是从 文本文件 或 管道数据流 中筛选匹配的 行 和 数据 ，如果再配合 正则表达式 ，功能十分强大，是Linux运维人员必备命令

grep命令里的 匹配模式 就是你想要找的东西，可以是 普通的文字符号 ，也可以是正则表达式

参数选项	解释说明
-v	排除匹配结果
-n	显示匹配行与行号
-i	不区分大小写
-c	只统计匹配的行数
-E	使用egrep命令
--color=auto	为grep过滤结果添加颜色
-w	只匹配过滤的单词
-o	只输出匹配的内容

案例

```

cat /etc/passwd > /tmp/test_grep.txt

grep "login" /tmp/test_grep.txt -n                #找出login有关行
grep "login" /tmp/test_grep.txt -n -v            #找出没有login的行
grep "ROOT" /tmp/test_grep.txt -i                #忽略大小写, 找出root有关行
grep -E "root|sync" /tmp/test_grep.txt --color=auto #同时过滤出root和sync有关行
grep "login" /tmp/test_grep.txt -c                #统计匹配结果的行数
grep "login" /tmp/test_grep.txt -n -o            #只输出匹配出的内容

grep "oldboy" /tmp/test_grep.txt -w                #完整匹配, 字符串精确匹配, 整个单词
grep -E "^#|^$" /tmp/test_grep.txt                #过滤掉空白和注释行

```

正则表达式grep实践

准备测试文件

```

[root@pylinux data]# cat -n luffy.txt
 1 I am oldboy teacher
 2 I teach linux.
 3 I like python.
 4
 5 My qq is 877348180.
 6
 7 My name is chaoge.
 8
 9 Our school website is http://oldboyedu.com
10
11
12

```

^符号

1. 输出所有以m开头的行

```

[root@pylinux data]# grep -i -n "^m" luffy.txt    # -i忽略大小写 -n 显示仪行号
5:My qq is 877348180.
7:My name is chaoge.

```

```
[root@pylinux data]# grep -n -i "^M" luffy.txt
5:My qq is 877348180.
7:My name is chaoge.
```

2. 输出所有以i开头的行

```
[root@pylinux data]# grep -i -n "^i" luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
```

```
[root@pylinux data]# grep -i -n "^i" luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
```

\$符

1. 输出所有以r结尾的行

```
[root@pylinux data]# grep -i -n "r$" luffy.txt
1:I am oldboy teacher
```

```
[root@pylinux data]# grep -i -n "r$" luffy.txt
1:I am oldboy teacher
```

2. 输出所以以m开头的行

```
[root@pylinux data]# grep -i -n "m$" luffy.txt
9:Our school website is http://oldboyedu.com
```

```
[root@pylinux data]# grep -i -n "m$" luffy.txt
9:Our school website is http://oldboyedu.com
```

TIP

注意在Linux平台下，所有文件的结尾都有一个\$符
可以用cat -A 查看文件

3. 输出所有以"."结尾的行，注意用转义符

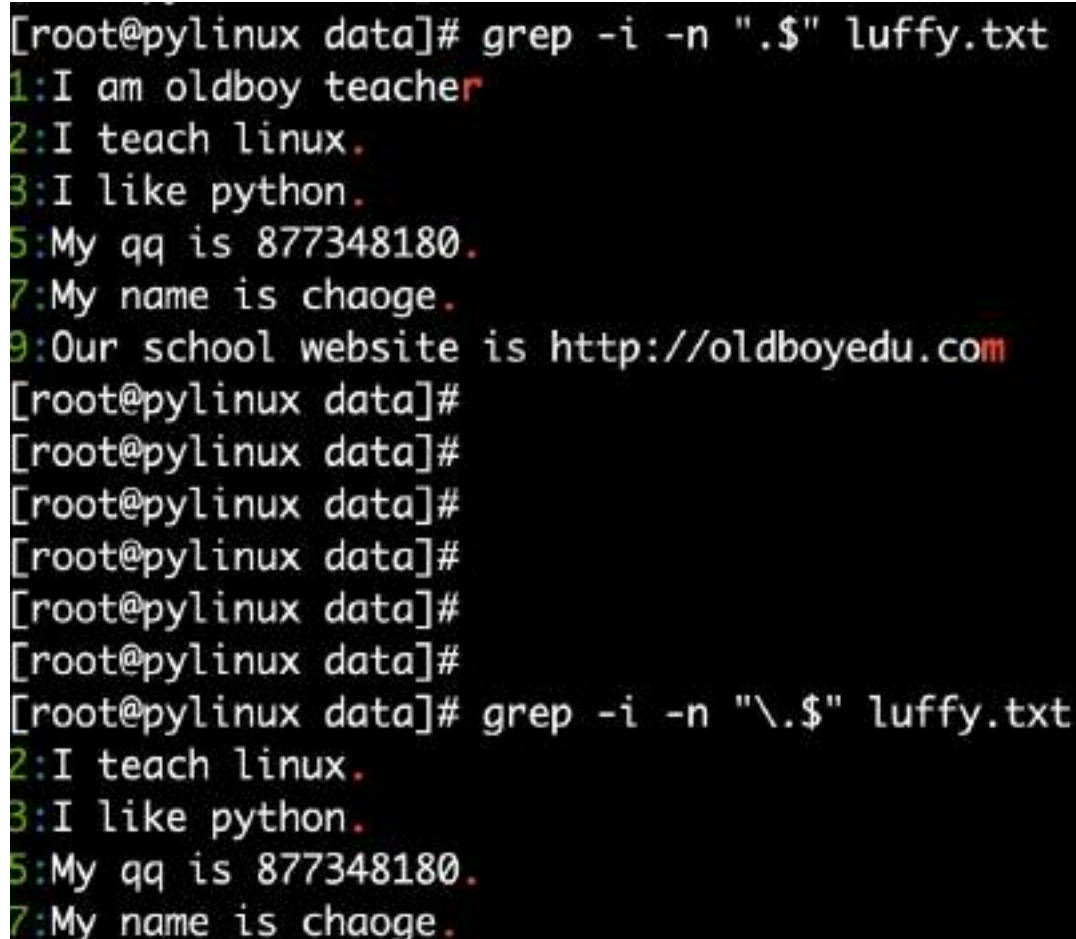
1. 注意不加转义符的结果，正则里"."是匹配任意1个字符，grep把.当做正则处理了，因此把有数据的行找出来

了,

```
[root@pylinux data]# grep -i -n ".$" luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
5:My qq is 877348180.
7:My name is chaoge.
9:Our school website is http://oldboyedu.com
```

2. 加上转义符, 当做普通的小数点过滤

```
[root@pylinux data]# grep -i -n "\.$" luffy.txt
2:I teach linux.
3:I like python.
5:My qq is 877348180.
7:My name is chaoge.
```



```
[root@pylinux data]# grep -i -n ".$" luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
5:My qq is 877348180.
7:My name is chaoge.
9:Our school website is http://oldboyedu.com
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]# grep -i -n "\.$" luffy.txt
2:I teach linux.
3:I like python.
5:My qq is 877348180.
7:My name is chaoge.
```

^\$组合符

1. 找出文件的空行, 以及行号

```
[root@pylinux data]# grep "^$" luffy.txt -n
```

```
4:
6:
8:
10:
11:
12:
```

```
[root@pylinux data]# grep "^$" luffy.txt -n
4:
6:
8:
10:
11:
12:
```

.点符号

"."点表示任意一个字符，有且只有一个，不包含空行

```
[root@pylinux data]# grep -i -n "." luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
5:My qq is 877348180.
7:My name is chaoge.
9:Our school website is http://oldboyedu.com
```

```
[root@pylinux data]# grep -i -n "." luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
5:My qq is 877348180.
7:My name is chaoge.
9:Our school website is http://oldboyedu.com
```

匹配出".ac"，找出任意一个三位字符，包含ac

```
[root@pylinux data]# grep -i -n ".ac" luffy.txt
1:I am oldboy teacher
2:I teach linux.
```

```
[root@pylinux data]# grep -i -n ".ac" luffy.txt
1:I am oldboy teacher
2:I teach linux.
```

\转义符

1.找出文中所有的点"."

```
[root@pylinux data]# grep "\." luffy.txt
I teach linux.
I like python.
My qq is 877348180.
My name is chaoge.
Our school website is http://oldboyedu.com
```

```
[root@pylinux data]# grep "\." luffy.txt
I teach linux.
I like python.
My qq is 877348180.
My name is chaoge.
Our school website is http://oldboyedu.com
```

*符

1.找出前一个字符0次或多次，找出文中出现"i"的0次或多次

```
[root@pylinux data]# grep -n "i*" luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
4:
5:My qq is 877348180.
6:
7:My name is chaoge.
8:
9:Our school website is http://oldboyedu.com
10:
11:
12:
```

```
[root@pylinux data]# grep -n "i*" luffy.txt
1:I am oldboy teacher
2:I teach linux.
3:I like python.
4:
5:My qq is 877348180.
6:
7:My name is chaoge.
8:
9:Our school website is http://oldboyedu.com
10:
11:
12:
```

. * 组合符

.表示任意一个字符，*表示匹配前一个字符0次或多次，因此放一起，代表匹配所有内容，以及空格

```
[root@pylinux data]# grep '.*' luffy.txt
I am oldboy teacher
I teach linux.
I like python.

My qq is 877348180.

My name is chaoge.

Our school website is http://oldboyedu.com
```

```
[root@pylinux data]# grep '.*' luffy.txt
I am oldboy teacher
I teach linux.
I like python.

My qq is 877348180.

My name is chaoge.

Our school website is http://oldboyedu.com

[root@pylinux data]#
```

^.*o符

^以某字符为开头

.任意0或多个字符

.*代表匹配所有内容

o普通字符，一直到字母o结束

这种匹配相同字符到最后一个字符的特点，称之为贪婪匹配

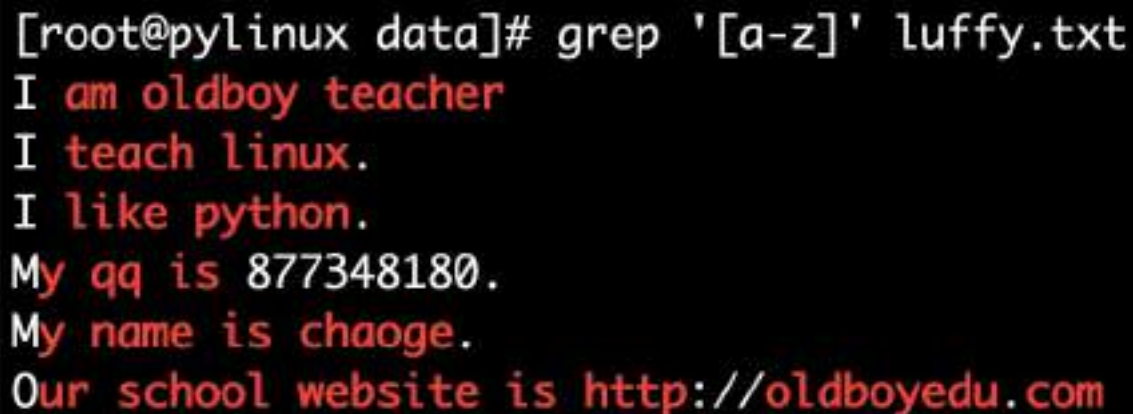
```
[root@chaogelinux data]# grep "I.*o" luffy.txt
I am oldboy teacher
I like python.
```

[abc]中括号

中括号表达式，[abc]表示匹配中括号中任意一个字符，a或b或c，常见形式如下

- [a-z]匹配所有小写单个字母
- [A-Z]匹配所有单个大写字母
- [a-zA-Z]匹配所有的单个大小写字母
- [0-9]匹配所有单个数字
- [a-zA-Z0-9]匹配所有数字和字母

```
[root@pylinux data]# grep '[a-z]' luffy.txt
I am oldboy teacher
I teach linux.
I like python.
My qq is 877348180.
My name is chaoge.
Our school website is http://oldboyedu.com
```



```
[root@pylinux data]# grep '[a-z]' luffy.txt
I am oldboy teacher
I teach linux.
I like python.
My qq is 877348180.
My name is chaoge.
Our school website is http://oldboyedu.com
```

```
[root@pylinux data]# grep '[abcd]' luffy.txt
I am oldboy teacher
I teach linux.
My name is chaoge.
Our school website is http://oldboyedu.com
```



```
[root@pylinux data]# grep '[abcd]' luffy.txt
I am oldboy teacher
I teach linux.
My name is chaoge.
Our school website is http://oldboyedu.com
```

grep参数-o

使用"-o"选项，可以只显示被匹配到的关键字，而不是讲整行的内容都输出。

显示文件中有多少个字符a

```
[root@pylinux data]# grep -o 'a' luffy.txt |wc -l
5
```

[^abc]中括号中取反

[^abc] 或 [^a-c] 这样的命令，"^"符号在中括号中第一位表示排除，就是排除字母a或b或c

出现在中括号里的尖角号表示取反

1.找出除了小写字母以外的字符

```
[root@pylinux data]# grep '[^a-z]' luffy.txt
I am oldboy teacher
I teach linux.
I like python.
My qq is 877348180.
My name is chaoge.
Our school website is http://oldboyedu.com
```



```
[root@pylinux data]# grep '[^a-z]' luffy.txt
I am oldboy teacher
I teach linux.
I like python.
My qq is 877348180.
My name is chaoge.
Our school website is http://oldboyedu.com
```

扩展正则表达式实践

此处使用grep -E进行实践扩展正则，egrep官网已经弃用

+号

+号表示匹配前一个字符1次或多次，必须使用grep -E 扩展正则

```
[root@pylinux data]# grep -E 'l+' luffy.txt
I am oldboy teacher
I teach linux.
I like python.
Our school website is http://oldboyedu.com
```



```
[root@pylinux data]# grep 'l+' luffy.txt
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]# grep -E 'l+' luffy.txt
I am oldboy teacher
I teach linux.
I like python.
Our school website is http://oldboyedu.com
```

?符

匹配前一个字符0次或1次

1. 找出文件中包含gd或god的行

```
[root@pylinux data]# grep -E 'go?d' luffycity.txt
god          #字母o出现了一次
gd           #字母o出现了0次
```

|符

竖线|在正则中是或者的意思

1. 找出系统中的txt文件，且名字里包含a或b的字符

```
[root@pylinux data]# find / -maxdepth 3 -name "*.txt" |grep -i -E "a|b"
/data/luffycity.txt
/data/luffy.txt
/test_find/chaoge.txt
/test_find/alex.txt
/opt/all.txt
/opt/_book/123.txt
/opt/Python-3.7.3/pybuilddir.txt
/opt/alltxt.txt
/opt/s15oldboy/qiong.txt
/opt/IIS/keystorePass.txt
```

()小括号

将一个或多个字符捆绑在一起，当作一个整体进行处理；

- 小括号功能之一是 分组过滤被括起来的内容，括号内的内容表示一个整体
- 括号()内的内容可以被后面的"\n"正则引用，n为数字，表示引用第几个括号的内容

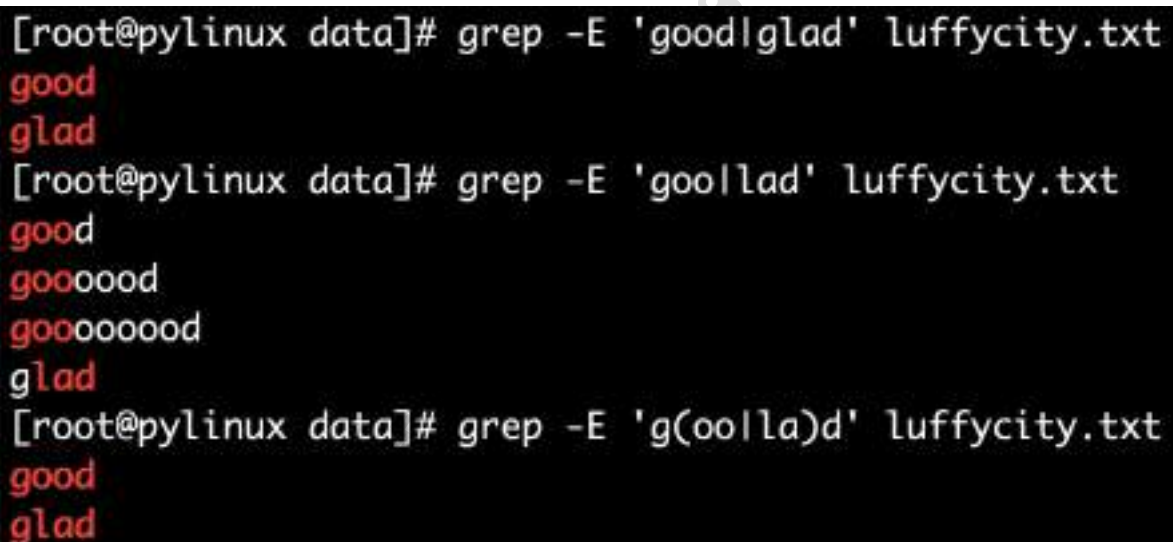
- `\1` : 表示从左侧起, 第一个括号中的模式所匹配到的字符
- `\2` : 从左侧起, 第二个括号中的模式所匹配到的字符

1. 找出包含good和glad的行

```
[root@pylinux data]# grep -E 'goo|lad' luffycity.txt      #结果不是我们想要的
good
gooood
gooooooood
glad

[root@pylinux data]# grep -E 'good|glad' luffycity.txt    #我们希望能够实现这样的匹配
good
glad

[root@pylinux data]# grep -E 'g(oo|la)d' luffycity.txt
good
glad
```



```
[root@pylinux data]# grep -E 'good|glad' luffycity.txt
good
glad
[root@pylinux data]# grep -E 'goo|lad' luffycity.txt
good
gooood
gooooooood
glad
[root@pylinux data]# grep -E 'g(oo|la)d' luffycity.txt
good
glad
```

分组之后向引用

```
[root@chaogelinux data]# cat lovers.txt
I like my lover.
I love my lover.
He likes his lovers.
He love his lovers.

[root@chaogelinux data]# grep -E '(l..e).*\1' lovers.txt
I love my lover.
He love his lovers.

[root@chaogelinux data]# grep -E '(r..t).*\1' /etc/passwd    #案例2
root:x:0:0:root:/root:/bin/bash
```


[TOC]

Linux三剑客sed

注意sed和awk使用单引号，双引号有特殊解释

sed是Stream Editor（字符流编辑器）的缩写，简称流编辑器。

sed是操作、过滤和转换文本内容的强大工具。

常用功能包括结合正则表达式对文件实现快速增删改查，其中查询的功能中最常用的两大功能是过滤（过滤指定字符串）、取行（取出指定行）。

老师QQ: 877348180



语法：

```
sed [选项] [sed内置命令字符] [输入文件]
```

选项：

参数选项	解释
-n	取消默认sed的输出，常与sed内置命令p一起用
-i	直接将修改结果写入文件，不用-i，sed修改的是内存数据
-e	多次编辑,不需要管道符了
-r	支持正则扩展

sed的 内置命令字符 用于对文件进行不同的操作功能，如对文件增删改查

sed常用 内置命令字符：

--	--

sed的内置命令字符	解释
a	append, 对文本追加, 在指定行后面添加一行/多行文本
d	Delete, 删除匹配行
i	insert, 表示插入文本, 在指定行前添加一行/多行文本
p	Print, 打印匹配行的内容, 通常p与-n一起用
s/正则/替换内容/g	匹配正则内容, 然后替换内容 (支持正则), 结尾g代表全局匹配

sed匹配范围

范围	解释
空地址	全文处理
单地址	指定文件某一行
/pattern/	被模式匹配到的每一行
范围区间	10,20 十到二十行, 10,+5第10行向下5行, /pattern1/,/pattern2/
步长	1~2, 表示1、3、5、7、9行, 2~2两个步长, 表示2、4、6、8、10、偶数行

sed案例

准备测试数据

```
[root@pylinux data]# cat -n luffycity.txt
1    My name is chaoge.
2    I teach linux.
3    I like play computer game.
4    My qq is 877348180.
5    My website is http://pythonav.cn.
```

1.输出文件第2, 3行的内容

```
[root@pylinux data]# sed -n '2,3p' luffycity.txt
I teach linux.
I like play computer game.
```

```
[root@pylinux data]# sed "2,3p" luffycity.txt
My name is chaoge.
I teach linux.
I teach linux.
I like play computer game.
I like play computer game.
My qq is 877348180.
My website is http://pythonav.cn.
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]# sed -n "2,3p" luffycity.txt
I teach linux.
I like play computer game.
[root@pylinux data]#
```

sed 默认输出所有结果

添加-n 参数不显示默认输出

2.过滤出含有linux的字符串

#sed可以实现grep的过滤效果，必须吧要过滤的内容放在双斜杠中

```
[root@pylinux data]# sed -n '/linux/p' luffycity.txt
I teach linux.
```

3.删除含有game的行

注意sed想要修改文件内容，还得用-i参数

```
[root@pylinux data]# sed '/game/d' luffycity.txt
My name is chaoge.
I teach linux.
My qq is 877348180.
My website is http://pythonav.cn.
```

```
[root@pylinux data]# cat -n luffycity.txt
 1 My name is chaoge.
 2 I teach linux.
 3 I like play computer game.
 4 My qq is 877348180.
 5 My website is http://pythonav.cn.
[root@pylinux data]#
[root@pylinux data]#
[root@pylinux data]# sed '/game/d' luffycity.txt
My name is chaoge.
I teach linux.
My qq is 877348180.
My website is http://pythonav.cn.
```

有关game的行删了

想要将修改结果写入到文件，还得这么敲

```
[root@pylinux data]# sed -i '/game/d' luffycity.txt #不会输出结果，直接写入文件
```

删掉2, 3两行

```
[root@pylinux data]# sed '2,3d' luffycity.txt
```

删除第5行到结尾

```
[root@pylinux data]# sed '5,$d' luffycity.txt
My name is chaoge.
-----
I teach linux.
-----
```

4.将文件中的My全部替换为His

- s内置符配合g，代表全局替换，中间的"/"可以替换为"#@"等


```
[root@pylinux data]# sed 's/My/His/g' luffycity.txt
His name is chaoge.
I teach linux.
I like play computer game.
His qq is 877348180.
His website is http://pythonav.cn.
```

5.替换所有My为His，同时换掉QQ号为8888888

```
[root@pylinux data]# sed -e 's/My/His/g' -e 's/877348180/88888/g' luffycity.txt
His name is chaoge.
I teach linux.
I like play computer game.
His qq is 88888.
His website is http://pythonav.cn.
```

6.在文件第二行追加内容 a字符功能，写入到文件，还得添加 -i

```
[root@pylinux data]# sed -i '2a I am useing sed command' luffycity.txt
My name is chaoge.
I teach linux.
I am useing sed command
I like play computer game.
My qq is 877348180.
My website is http://pythonav.cn.
```



```
[root@pylinux data]# sed -i '2a I am useing sed command' luffycity.txt
[red text: 在第二行下面，添加了内容]
[red arrow points to line 3]

[root@pylinux data]#
[root@pylinux data]# cat luffycity.txt -n
 1 My name is chaoge.
 2 I teach linux.
 3 I am useing sed command
 4 I like play computer game.
 5 My qq is 877348180.
 6 My website is http://pythonav.cn.
```

添加多行信息，用换行符"\n"

```
[root@pylinux data]# sed -i "3a i like linux very much.\nand you?" luffycity.txt
[root@pylinux data]#
[root@pylinux data]# cat -n luffycity.txt
 1 My name is chaoge.
 2 I teach linux.
 3 I am useing sed command
 4 i like linux very much.
 5 and you?
 6 I like play computer game.
 7 My qq is 877348180.
 8 My website is http://pythonav.cn.
```



```
[root@pylinux data]# sed -i "3a i like linux very much.\nand you?" luffycity.txt
[root@pylinux data]#
[root@pylinux data]# cat -n luffycity.txt
 1 My name is chaoge.
 2 I teach linux.
 3 I am useing sed command
 4 i like linux very much.
 5 and you?
 6 I like play computer game.
 7 My qq is 877348180.
 8 My website is http://pythonav.cn.
```

在每一行下面插入新内容

```
[root@pylinux data]# sed "a -----" luffycity.txt
My name is chaoge.
-----
I teach linux.
-----
I am useing sed command
-----
i like linux very much.
-----
and you?
-----
I like play computer game.
-----
My qq is 877348180.
-----
My website is http://pythonav.cn.
-----
```

7.在第二行上面插入内容

```
[root@pylinux data]# sed '2i i am 27' luffycity.txt
My name is chaoge.
i am 27
-----
I teach linux.
-----
I am useing sed command
-----
i like linux very much.
-----
and you?
-----
I like play computer game.
-----
My qq is 877348180.
```

```
-----
My website is http://pythonav.cn.
-----
```

sed配合正则表达式企业案例

上一节是用grep -E 扩展正则表达式，这一节是用sed配合正则表达式使用

取出linux的IP地址

1.删除网卡信息

```
[root@pylinux ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.141.32.137 netmask 255.255.192.0 broadcast 10.141.63.255
    ether 52:54:00:12:35:79 txqueuelen 1000 (Ethernet)
    RX packets 3958953 bytes 447491542 (426.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5001880 bytes 605578926 (577.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

去头去尾法

交给管道符，一次去头，一次去尾

思路：

1. 首先取出第二行

```
[root@pylinux ~]# ifconfig | sed -n '2p'
    inet 10.141.32.137 netmask 255.255.192.0 broadcast 10.141.63.255
```

2. 找到第二行后，去掉ip之前的内容

```
[root@pylinux ~]# ifconfig eth0|sed -n '2s#^.*inet##gp'
10.141.32.137 netmask 255.255.192.0 broadcast 10.141.63.255
```

解释：

-n是取消默认输出

2s是处理第二行内容

#^.*inet## 是匹配inet前所有的内容

gp代表全局替换且打印替换结果

3. 再次处理，去掉ip后面的内容

```
[root@pylinux tmp]# sed -n '2s/^.*inet//gp' ip.txt | sed -n 's/net.*$//gp'
```

```
10.141.32.137
```

解释:

`net.*$` 匹配net到结尾的内容

`s/net.*$//gp` #把匹配到的内容替换为空

-e参数多次编辑

```
[root@pylinux tmp]# ifconfig eth0 | sed -ne '2s/^.*inet//g' -e '2s/net.*$//gp'  
10.141.32.137
```

```
[root@pylinux tmp]# ifconfig eth0 | sed -ne '2s/^.*inet//g' -e '2s/net.*$//gp'  
10.141.32.137  
[root@pylinux tmp]#  
[root@pylinux tmp]#  
[root@pylinux tmp]#  
[root@pylinux tmp]#  
[root@pylinux tmp]# ifconfig eth0  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.141.32.137 netmask 255.255.192.0 broadcast 10.141.32.191  
    ether 52:54:00:4f:00:00 txqueuelen 1000 (Ethernet)  
    RX packets 3966141 bytes 448278800 (427.5 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 5008162 bytes 606448585 (578.3 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

第一次处理
去掉第二行ip之前的信息
不打印

对文件第二次编辑
去掉第二行ip之后的信息
且打印最终结果

[TOC]

Linux三剑客awk

awk是一个强大的linux命令，有强大的文本格式化的能力，好比将一些文本数据格式化成专业的excel表的样式

awk早期在Unix上实现，我们用的awk是gawk，是GUN awk的意思

```
[root@devlinux ~]# ll /usr/bin/awk  
lrwxrwxrwx. 1 root root 4 KB 6月28日 22:51 /usr/bin/awk -> gawk
```

awk更是一门编程语言，支持条件判断、数组、循环等功能

再谈三剑客

- grep，擅长单纯的查找或匹配文本内容
- awk，更适合编辑、处理匹配到的文本内容
- sed，更适合格式化文本内容，对文本进行复杂处理

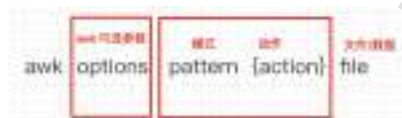
三个命令称之为Linux的三剑客

awk基础

awk语法

```
awk [option] 'pattern[action]' file ...
```

awk 参数 '条件动作' 文件



- Action指的是动作，awk擅长文本格式化，且输出格式化后的结果，因此最常用的动作就是 `print` 和 `printf`

awk场景

动作场景



```
[root@pylinux tmp]# cat alex.txt
alex1 alex2 alex3 alex4 alex5
alex6 alex7 alex8 alex9 alex10
alex11 alex12 alex13 alex14 alex15
alex16 alex17 alex18 alex19 alex20
alex21 alex22 alex23 alex24 alex25
alex26 alex27 alex28 alex29 alex30
alex31 alex32 alex33 alex34 alex35
alex36 alex37 alex38 alex39 alex40
alex41 alex42 alex43 alex44 alex45
alex46 alex47 alex48 alex49 alex50
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# cat alex.txt |awk '{print $2}'
alex2
alex7
alex12
alex17
alex22
alex27
alex32
alex37
alex42
alex47
```

我们执行的命令是 `awk '{print $2}'`，没有使用参数和模式，`$2` 表示输出文本的 第二列 信息

awk默认以空格为分隔符，且多个空格也识别为一个空格，作为分隔符

awk是按行处理文件，一行处理完毕，处理下一行，根据用户指定的分割符去工作，没有指定则默认空格

指定了分隔符后，awk把每一行切割后的数据对应到内置变量



- `$0`表示整行

- \$NF表示当前分割后的最后一列
- 倒数第二列可以写成\$(NF-1)

awk内置变量

内置变量	解释
\$n	指定分隔符后，当前记录的第n个字段
\$0	完整的输入记录
FS	字段分隔符，默认是空格
NF(Number of fields)	分割后，当前行一共有多少个字段
NR(Number of records)	当前记录数，行数
更多内置变量可以man手册查看	man awk

一次性输出多列

```
[root@pylinux tmp]# awk '{print $1,$2}' alex.txt
alex1 alex2
alex6 alex7
alex11 alex12
alex16 alex17
alex21 alex22
alex26 alex27
alex31 alex32
alex36 alex37
alex41 alex42
alex46 alex47
```

```
[root@pylinux tm]# cat -n a1a.txt
1 a1a1 a1a2 a1a3 a1a4 a1a5
2 a1a6 a1a7 a1a8 a1a9 a1a10
3 a1a11 a1a12 a1a13 a1a14 a1a15
4 a1a16 a1a17 a1a18 a1a19 a1a20
5 a1a21 a1a22 a1a23 a1a24 a1a25
6 a1a26 a1a27 a1a28 a1a29 a1a30
7 a1a31 a1a32 a1a33 a1a34 a1a35
8 a1a36 a1a37 a1a38 a1a39 a1a40
9 a1a41 a1a42 a1a43 a1a44 a1a45
10 a1a46 a1a47 a1a48 a1a49 a1a50
[root@pylinux tm]# awk '{print $6,$1}' a1a.txt
a1a4 a1a1
a1a5 a1a6
a1a6 a1a11
a1a7 a1a16
a1a8 a1a21
a1a9 a1a26
a1a10 a1a31
a1a11 a1a36
a1a12 a1a41
a1a13 a1a46
```

自定义输出内容

awk, 必须 外层单引号 , 内层双引号

内置变量 \$1、\$2 都不得添加双引号, 否则会识别为文本, 尽量别加引号

老师QQ: 877348180



```
[root@pylinux tmp]# awk '{print "第一列",$1,"第二列",$2,"第三列",$3}' alex.txt
第一列 alex1 第二列 alex2 第三列 alex3
第一列 alex6 第二列 alex7 第三列 alex8
第一列 alex11 第二列 alex12 第三列 alex13
第一列 alex16 第二列 alex17 第三列 alex18
第一列 alex21 第二列 alex22 第三列 alex23
第一列 alex26 第二列 alex27 第三列 alex28
第一列 alex31 第二列 alex32 第三列 alex33
第一列 alex36 第二列 alex37 第三列 alex38
第一列 alex41 第二列 alex42 第三列 alex43
第一列 alex46 第二列 alex47 第三列 alex48
```

输出整行信息

```
[root@pylinux tmp]# awk '{print}' alex.txt #两种写法都可以
[root@pylinux tmp]# awk '{print $0}' alex.txt
```

awk参数

参数	解释
-F	指定分割字段符
-v	定义或修改一个awk内部的变量
-f	从脚本文件中读取awk命令

awk案例

测试文件内容

```
[root@pylinux tmp]# cat pwd.txt -n
1 sync:x:5:0:sync:/sbin:/bin/sync
2 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
3 halt:x:7:0:halt:/sbin:/sbin/halt
4 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
5 operator:x:11:0:operator:/root:/sbin/nologin
6 games:x:12:100:games:/usr/games:/sbin/nologin
7 ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
8 nobody:x:99:99:Nobody:/:/sbin/nologin
9 systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
```



```

10    dbus:x:81:81:System message bus:/:/sbin/nologin
11    polkitd:x:999:998:User for polkitd:/:/sbin/nologin
12    libstoragemgmt:x:998:997:daemon account for libstoragemgmt:/var/run/lsm:/
sbin/nologin
13    rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
14    ntp:x:38:38::/etc/ntp:/sbin/nologin

```

显示文件第五行

```

#NR在awk中表示行号，NR==5表示行号是5的那一行
#注意一个等于号，是修改变量值的意思，两个等于号是关系运算符，是"等于"的意思
[root@pylinux tmp]# awk 'NR==5' pwd.txt
operator:x:11:0:operator:/root:/sbin/nologin

```

显示文件2-5行

设置模式（条件）

```

#告诉awk，我要看行号2到5的内容
[root@pylinux tmp]# awk 'NR==2,NR==5' pwd.txt
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin

```

给每一行的内容添加行号

添加变量，NR等于行号，\$0表示一整行的内容

{print}是awk的动作

```

[root@pylinux tmp]# awk '{print NR,$0}' alex.txt
1 alex1 alex2 alex3 alex4 alex5
2 alex6 alex7 alex8 alex9 alex10
3 alex11 alex12 alex13 alex14 alex15
4 alex16 alex17 alex18 alex19 alex20
5 alex21 alex22 alex23 alex24 alex25
6 alex26 alex27 alex28 alex29 alex30
7 alex31 alex32 alex33 alex34 alex35
8 alex36 alex37 alex38 alex39 alex40
9 alex41 alex42 alex43 alex44 alex45
10 alex46 alex47 alex48 alex49 alex50 alex51

```

显示文件3-5行且输出行号

```
[root@pylinux tmp]# awk 'NR==3,NR==5 {print NR,$0}' alex.txt
3 alex11 alex12 alex13 alex14 alex15
4 alex16 alex17 alex18 alex19 alex20
5 alex21 alex22 alex23 alex24 alex25
```

显示pwd.txt文件的第一列，倒数第二和最后一列

```
[root@pylinux tmp]# cat pwd.txt -n
 1 sync:x:5:0:sync:/sbin:/bin/sync
 2 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
 3 halt:x:7:0:halt:/sbin:/sbin/halt
 4 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
 5 operator:x:11:0:operator:/root:/sbin/nologin
 6 games:x:12:100:games:/usr/games:/sbin/nologin
 7 ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
 8 nobody:x:99:99:Nobody:/:/sbin/nologin
 9 systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
10 dbus:x:81:81:System message bus:/:/sbin/nologin
11 polkitd:x:999:998:User for polkitd:/:/sbin/nologin
12 libstoragemgmt:x:998:997:daemon account for libstoragemgmt:/var/run/lsm:/
sbin/nologin
13 rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
14 ntp:x:38:38:/:etc/ntp:/sbin/nologin
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk -F ':' '{print $1,$(NF-1),$NF}' pwd.txt
sync /sbin /bin/sync
shutdown /sbin /sbin/shutdown
halt /sbin /sbin/halt
mail /var/spool/mail /sbin/nologin
operator /root /sbin/nologin
games /usr/games /sbin/nologin
ftp /var/ftp /sbin/nologin
nobody / /sbin/nologin
systemd-network / /sbin/nologin
dbus / /sbin/nologin
polkitd / /sbin/nologin
libstoragemgmt /var/run/lsm /sbin/nologin
rpc /var/lib/rpcbind /sbin/nologin
ntp /etc/ntp /sbin/nologin
```



老师QQ: 877348180

Linux三剑客awk

老师QQ: 877348180



awk的分隔符有两种

- 输入分隔符，awk默认是空格，空白字符，英文是field separator，变量名是FS
- 输出分隔符，output field separator，简称OFS

FS输入分隔符

awk逐行处理文本的时候，以输入分割符为准，把文本切成多个片段，默认符号是空格

当我们处理特殊文件，没有空格的时候，可以自由指定分隔符特点

```
[root@pylinux tmp]# awk -F '#' '{print $1}' chaoge.txt
```

```
超哥c
超哥f
超哥i
超哥l
超哥o
超哥r
超哥u
超哥x
```



- 除了使用-F选项，还可以使用变量的形式，指定分隔符，使用-v选项搭配，修改FS变量

```
[root@pylinux tmp]# awk -v FS='#' '{print $1}' chaoge.txt
```

```
超哥c
超哥f
超哥i
超哥l
超哥o
超哥r
超哥u
超哥x
```

OFS输出分割符

awk执行完命令，默认用空格隔开每一列，这个空格就是awk的默认输出符，例如

```
[root@pylinux tmp]# cat chaoge.txt
超哥c#超哥d#超哥e
超哥f#超哥g#超哥h
超哥i#超哥j#超哥k
超哥l#超哥m#超哥n
超哥o#超哥p#超哥q
超哥r#超哥s#超哥t
超哥u#超哥v#超哥w
超哥x#超哥y#超哥z
[root@pylinux tmp]# awk -v FS='#' '{print $1,$3}' chaoge.txt
超哥c 超哥e
超哥f 超哥h
超哥i 超哥k
超哥l 超哥n
超哥o 超哥q
超哥r 超哥t
超哥u 超哥w
超哥x 超哥z
```



```
[root@pylinux tmp]# cat chaoge.txt
超哥c#超哥d#超哥e
超哥f#超哥g#超哥h
超哥i#超哥j#超哥k
超哥l#超哥m#超哥n
超哥o#超哥p#超哥q
超哥r#超哥s#超哥t
超哥u#超哥v#超哥w
超哥x#超哥y#超哥z
[root@pylinux tmp]# awk -v FS='#' '{print $1,$3}' chaoge.txt
超哥c 超哥e
超哥f 超哥h
超哥i 超哥k
超哥l 超哥n
超哥o 超哥q
超哥r 超哥t
超哥u 超哥w
超哥x 超哥z
```

通过OFS设置输出分割符，记住修改变量必须搭配选项 -v

```
[root@pylinux tmp]# cat chaoge.txt
超哥c#超哥d#超哥e
超哥f#超哥g#超哥h
超哥i#超哥j#超哥k
超哥l#超哥m#超哥n
超哥o#超哥p#超哥q
超哥r#超哥s#超哥t
超哥u#超哥v#超哥w
超哥x#超哥y#超哥z
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk -v FS='#' -v OFS='---' '{print $1,$3 }' chaoge.txt
超哥c---超哥e
```

```
超哥f---超哥h
超哥i---超哥k
超哥l---超哥n
超哥o---超哥q
超哥r---超哥t
超哥u---超哥w
超哥x---超哥z
[root@pylinux tmp]#
```



比对这个文本内容，没有空格
中间以#分割

```
[root@pylinux tmp]# cat chaoge.txt
超哥c超哥e
超哥f超哥h
超哥i超哥k
超哥l超哥n
超哥o超哥q
超哥r超哥t
超哥u超哥w
超哥x超哥z
[root@pylinux tmp]#
[root@pylinux tmp]# 删去输入分隔符
[root@pylinux tmp]#
[root@pylinux tmp]# awk -v FS='#' -v OFS=' ' '{print $1,$3}' chaoge.txt
超哥c 超哥e
超哥f 超哥h
超哥i 超哥k
超哥l 超哥n
超哥o 超哥q
超哥r 超哥t
超哥u 超哥w
超哥x 超哥z
[root@pylinux tmp]#
```

指定输出分隔符

因此针对#分割后，打印第一列，第三列结果

中间的输出分隔符，被改成了，---

输出分隔符与逗号

awk是否存在输出分隔符，特点在于 '{print \$1,\$3 }' 逗号 的区别

- 添加逗号，默认是空格分隔符

```
[root@pylinux tmp]# awk -v FS='#' '{print $1,$3 }' chaoge.txt
超哥c 超哥e
超哥f 超哥h
超哥i 超哥k
超哥l 超哥n
超哥o 超哥q
超哥r 超哥t
超哥u 超哥w
超哥x 超哥z
```

- 不加逗号

```
[root@pylinux tmp]# awk -v FS='#' '{print $1$3 }' chaoge.txt
超哥c超哥e
超哥f超哥h
超哥i超哥k
超哥l超哥n
超哥o超哥q
超哥r超哥t
超哥u超哥w
超哥x超哥z
```

- 修改分割符，改为\t(制表符，四个空格)或者任意字符

```
[root@pylinux tmp]# awk -v FS='#' -v OFS='\t\t' '{print $1,$3 }' chaoge.txt
超哥c      超哥e
超哥f      超哥h
超哥i      超哥k
超哥l      超哥n
超哥o      超哥q
超哥r      超哥t
超哥u      超哥w
超哥x      超哥z
```

老师QQ: 877348180

[TOC]

awk变量

awk参数

参数	解释
-F	指定分割字段符
-v	定义或修改一个awk内部的变量
-f	从脚本文件中读取awk命令

对于awk而言，变量分为

- 内置变量
- 自定义变量

内置变量	解释
FS	输入字段分隔符，默认为空白字符
OFS	输出字段分隔符，默认为空白字符
RS	输入记录分隔符(输入换行符)，指定输入时的换行符
ORS	输出记录分隔符（输出换行符），输出时用指定符号代替换行符
NF	NF： number of Field，当前行的字段的个数(即当前行被分割成了几列)，字段数量
NR	NR： 行号，当前处理的文本行的行号。
FNR	FNR： 各文件分别计数的行号
FILENAME	FILENAME： 当前文件名
ARGC	ARGC： 命令行参数的个数
ARGV	ARGV： 数组，保存的是命令行所给定的各参数

内置变量

NR、NF、FNR

- awk的内置变量NR、NF是不用添加\$符号的
- 而 \$0 \$1 \$2 \$3 ... 是需要添加\$符号的

输出每行行号，以及字段总个数

```
[root@pylinux tmp]# cat -n alex.txt
1    alex1 alex2 alex3 alex4 alex5
2    alex6 alex7 alex8 alex9 alex10
```

```
3   alex11 alex12 alex13 alex14 alex15
4   alex16 alex17 alex18 alex19 alex20
5   alex21 alex22 alex23 alex24 alex25
6   alex26 alex27 alex28 alex29 alex30
7   alex31 alex32 alex33 alex34 alex35
8   alex36 alex37 alex38 alex39 alex40
9   alex41 alex42 alex43 alex44 alex45
10  alex46 alex47 alex48 alex49 alex50 alex51
[root@pylinux tmp]#
[root@pylinux tmp]# awk '{print NR,NF}' alex.txt
1 5
2 5
3 5
4 5
5 5
6 5
7 5
8 5
9 5
10 6
```

输出每行行号，以及指定的列

```
[root@pylinux tmp]# awk '{print NR,$1,$5}' alex.txt
1 alex1 alex5
2 alex6 alex10
3 alex11 alex15
4 alex16 alex20
5 alex21 alex25
6 alex26 alex30
7 alex31 alex35
8 alex36 alex40
9 alex41 alex45
10 alex46 alex50
```

处理多个文件显示行号

```
# 普通的NR变量，会将多个文件按照顺序排序
[root@pylinux tmp]# awk '{print NR,$0}' alex.txt pwd.txt
```

```
#使用FNR变量，可以分别对文件行数计数
[root@pylinux tmp]# awk '{print FNR,$0}' alex.txt pwd.txt
```

```
### 内置变量RS
```

RS变量作用是 输入分隔符 ，默认是 回车符 ，也就是 回车(Enter键)换行符

我们也可以自定义 空格 作为 行分隔符 ，每遇见一个空格，就换行处理

老师QQ: 877348180



```
[root@pylinux tmp]# awk -v RS=' ' '{print NR,$0}' chaoge.txt
```

内置变量ORS

ORS是输出分隔符的意思，awk默认认为，每一行结束了，就得添加 回车换行符

ORS变量可以更改输出符

```
awk -v ORS='@@@' '{print NR,$0}' chaoge.txt
```



内置变量FILENAME

显示awk正在处理文件的名字

```
[root@pylinux tmp]# awk '{print FILENAME,FNR,$0}' chaoge.txt alex.txt
chaoge.txt 1 超哥a 超哥b
chaoge.txt 2 超哥c 超哥d 超哥e
chaoge.txt 3 超哥f 超哥g 超哥h
chaoge.txt 4 超哥i 超哥j 超哥k
chaoge.txt 5 超哥l 超哥m 超哥n
chaoge.txt 6 超哥o 超哥p 超哥q
chaoge.txt 7 超哥r 超哥s 超哥t
chaoge.txt 8 超哥u 超哥v 超哥w
chaoge.txt 9 超哥x 超哥y 超哥z
alex.txt 1 alex1 alex2 alex3 alex4 alex5
alex.txt 2 alex6 alex7 alex8 alex9 alex10
alex.txt 3 alex11 alex12 alex13 alex14 alex15
alex.txt 4 alex16 alex17 alex18 alex19 alex20
```

变量ARGC、ARGV

ARGV表示的是一个数组，数组中保存的是命令行所给的 参数

数组是一种数据类型，如同一个盒子

盒子有它的名字，且内部有N个小格子，标号从0开始

给一个盒子起名字叫做months，月份是1~12，那就如图所示

month
1
"January"
2
"February"
3
"March"
4
"April"
5
"May"
6
"June"
7
"July"
8
"August"
9
"September"
10
"October"
11
"November"
12
"December"

```
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢"}' chaoge.txt
超哥教你学内置awk变量呢
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢", ARGV[0]}' chaoge.txt
超哥教你学内置awk变量呢 awk
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢", ARGV[0], ARGV[1]}' chaoge.txt
超哥教你学内置awk变量呢 awk chaoge.txt
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢", ARGV[0], ARGV[1], ARGV[2]}' chaoge.txt alex.txt
超哥教你学内置awk变量呢 awk chaoge.txt alex.txt
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
```

给awk设置一个模式（条件）

其实这个命令我们给awk传入了几个参数

在终端执行后，先打印“超哥教你学内置awk变量呢”

参数存入了数组变量中ARGV

索引	ARGV数组
索引0	awk
索引1	chaoge.txt
索引2	alex.txt

打印了索引0的值

打印了索引0和索引1的值

打印了索引0、1、2的值

```
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢"}' chaoge.txt
超哥教你学内置awk变量呢
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢", ARGV[0]}' chaoge.txt
超哥教你学内置awk变量呢 awk
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢", ARGV[0], ARGV[1]}' chaoge.txt
超哥教你学内置awk变量呢 awk chaoge.txt
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学内置awk变量呢", ARGV[0], ARGV[1], ARGV[2]}' chaoge.txt alex.txt
超哥教你学内置awk变量呢 awk chaoge.txt alex.txt
```

自定义变量

顾名思义，是我们自己定义变量

- 方法一， `-v varName=value`
- 方法二，在程序中直接定义

方法一：

```
[root@pylinux tmp]# awk -v luffyVarName="超哥nb, awk讲的好啊" 'BEGIN{print luffyVarName}' choage.txt
超哥nb, awk讲的好啊
```

```
[root@pylinux tmp]# awk -v luffyVarName="超哥nb, awk讲的好啊" 'BEGIN{print luffyVarName}' choage.txt
超哥nb, awk讲的好啊
[red]白定义变量与赋值[red]
[red]定义模式 (条件)[red]
[red]打印变量[red]
[red]打印变量值[red]
[red]且处理文件内容[red]
[red]超哥a 超哥b[red]
[red]超哥c 超哥d 超哥e[red]
[red]超哥f 超哥g 超哥h[red]
[red]超哥i 超哥j 超哥k[red]
[red]超哥l 超哥m 超哥n[red]
[red]超哥o 超哥p 超哥q[red]
[red]超哥r 超哥s 超哥t[red]
[red]超哥u 超哥v 超哥w[red]
[red]超哥x 超哥y 超哥z[red]
```

方法二：

```
[root@pylinux tmp]# awk 'BEGIN{chaogeVar="超哥带你学linux, 还怕学不会咋的";chaogeVar2="学的会, 必须学得会" ;print chaogeVar,chaogeVar2}'
超哥带你学linux, 还怕学不会咋的 学的会, 必须学得会
```

```
[root@pylinux tmp]# awk 'BEGIN{chaogeVar="超哥带你学linux, 还怕学不会咋的";chaogeVar2="学的会, 必须学得会" ;print chaogeVar,chaogeVar2}'
超哥带你学linux, 还怕学不会咋的 学的会, 必须学得会
[red]定义变量1[red]
[red]定义变量2[red]
[red]打印变量值[red]
```

方法三：间接引用shell变量

```
[root@pylinux tmp]# studyLinux="超哥讲的linux是真滴好, 嘿嘿"
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk -v myVar=$studyLinux 'BEGIN{print myVar}' # -v是给awk定义变量
超哥讲的linux是真滴好, 嘿嘿
```

awk格式化

前面我们接触到的awk的输出功能，是{print}的功能，只能对文本简单的输出，并不能美化或者修改格式

printf格式化输出

如果你学过C语言或是go语言，一定见识过printf()函数，能够对文本格式化输出

printf和print的区别

format的使用

要点：

- 1、其与print命令的最大不同是，printf需要指定format；
- 2、format用于指定后面的每个item的输出格式；
- 3、printf语句不会自动打印换行符；\\n

format格式的指示符都以%开头，后跟一个字符；如下：

%c：显示字符的ASCII码；
%d，%i：十进制整数；
%e，%E：科学计数法显示数值；
%f：显示浮点数；
%g，%G：以科学计数法的格式或浮点数的格式显示数值；
%s：显示字符串；
%u：无符号整数；
%%：显示%自身；

printf修饰符：

-：左对齐；默认右对齐，
+：显示数值符号； printf "%+d"

- printf动作默认不会添加换行符
- print默认添加空格换行符

```
[root@pylinux tmp]# awk '{print $1}' 超哥nb.txt
超哥nb1
超哥nb4
超哥nb7
超哥nb10
```

```
[root@pylinux tmp]# awk '{@print $1}' 超哥nb.txt
超哥nb1
超哥nb4
超哥nb7
```



```
超哥nb10
[root@pylinux tmp]#
[root@pylinux tmp]# awk '{printf $1}' 超哥nb.txt
超哥nb1超哥nb4超哥nb7超哥nb10[root@pylinux tmp]#
```

给printf添加格式

- 格式化字符串 %s 代表字符串的意思

```
[root@pylinux tmp]# awk '{printf "%s\n",$1}' 超哥nb.txt
超哥nb1
超哥nb4
超哥nb7
超哥nb10
```



超哥nb10--
超哥nb7--
超哥nb10--
[root@pylinux tmp]# awk '{printf "%s\n",\$1}' 超哥nb.txt
超哥nb1
超哥nb4
超哥nb7
超哥nb10

printf 用 "%s\n" 对数据进行格式化
%s 显示字符串
\\n 换行符

对多个变量进行格式化

当我们使用linux命令printf时，是这样的，一个 %s格式替换符，可以对多个参数进行重复格式化

```
[root@pylinux tmp]# printf "%s\n" a b c d
a
b
c
d
```

然而awk的 格式替换符 想要修改多个变量，必须传入多个

```
[root@pylinux tmp]# awk 'BEGIN{printf "%d\n%d\n%d\n%d\n%d\n",1,2,3,4,5}'
1
2
3
4
5
```



```
[root@pylinux tmp]# awk 'BEGIN{printf "%d\n%d\n%d\n%d\n",1,2,3,4,5}'
1
2
3
4
```

注意awk不跟文件数据，必须添加BEGIN
%d 代表的是十进制数字

- printf对输出的文本不会换行，必须添加对应的 格式替换符 和 \n
- 使用printf动作， '{printf "%s\n",\$1}'，替换的格式和变量之间得有逗号，
- 使用printf动作， %s %d 等格式化替换符 必须 和 被格式化的数据 一一对应

printf案例

```
[root@pylinux tmp]# cat 超哥nb.txt
```

```
超哥nb1 超哥nb2 超哥nb3
```

```
超哥nb4 超哥nb5 超哥nb6
```

```
超哥nb7 超哥nb8 超哥nb9
```

```
超哥nb10
```

```
[root@pylinux tmp]# awk '{printf "第一列: %s 第二列: %s 第三列: %s\n", $1, $2, $3}' 超哥nb.txt
```

```
第一列: 超哥nb1 第二列: 超哥nb2 第三列: 超哥nb3
```

```
第一列: 超哥nb4 第二列: 超哥nb5 第三列: 超哥nb6
```

```
第一列: 超哥nb7 第二列: 超哥nb8 第三列: 超哥nb9
```

```
第一列: 超哥nb10 第二列: 第三列:
```



- awk通过空格切割文档
- printf动作对数据格式化

对pwd.txt文件格式化



```
awk -F ":" 'BEGIN{printf "%-25s\t %-25s\t %-25s\t %-25s\t %-25s\t %-25s\t %-25s\n",
"用户名", "密码", "UID", "GID", "用户注释", "用户家目录", "用户使用的解释器"} {printf "%-25s\t %
-25s\t %-25s\t %-25s\t %-25s\t %-25s\t %-25s\n", $1, $2, $3, $4, $5, $6, $7}' pwd.txt
```

参数解释

'BEGIN{printf "格式替换符 格式替换符2","变量1","变量2"}' 执行BEGIN模式

%s是格式替换符，替换字符串

%s\t 格式化字符串后，添加制表符，四个空格

%-25s 已然是格式化字符串，-代表左对齐，25个字符长度

老师QQ: 877348180

[TOC]

awk模式pattern

再来回顾下awk的语法

```
awk [option] 'pattern[action]' file ...
```

awk是按行处理文本，刚才讲解了 `print` 动作，现在讲解特殊的 `pattern`：`BEGIN` 和 `END`

- `BEGIN`模式是处理文本之前需要执行的操作
- `END`模式是处理完所有行之后执行的操作

```
[root@pylinux tmp]# awk 'BEGIN{print "超哥教你学awk"}'
```

超哥教你学awk

#上述操作没有指定任何文件作为数据源，而是awk首选会执行BEGIN模式指定的print操作，打印出如上结果，然后发现没有任何文件需要操作，就结束了

```
[root@pylinux tmp]# awk 'BEGIN{print "超哥带你学awk"}{print $1}' alex.txt
```

超哥带你学awk

alex1

alex6

alex11

alex16

alex21

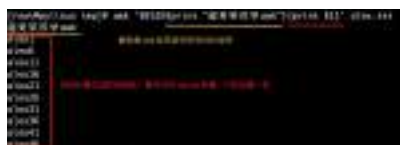
alex26

alex31

alex36

alex41

alex46



- 再次总结，`BEGIN`就是处理文本前，先执行BEGIN模式指定的动作
- 再来看`END`的作用（awk处理完所有指定的文本后，需要执行的动作）



awk结合BEGIN和END模式

```
[root@pylinux tmp]# awk 'BEGIN{print "来路飞学城听超哥讲linux"}{print $1,$2}END{print "超哥nb"}' alex.txt
```



awk模式pattern讲解

再来看一下awk的语法， 模式 也可以理解是 条件

```
awk [option] 'pattern[action]' file ...
```

刚才我们学了两个模式(条件)

- BEGIN
- END

awk默认是按行处理文本，如果不指定任何模式（条件），awk默认一行行处理

如果指定了模式，只有符合模式的才会被处理

模式（条件）案例

```
[root@pylinux tmp]# cat -n alex.txt
1    alex1 alex2 alex3 alex4 alex5
2    alex6 alex7 alex8 alex9 alex10
3    alex11 alex12 alex13 alex14 alex15
4    alex16 alex17 alex18 alex19 alex20
5    alex21 alex22 alex23 alex24 alex25
6    alex26 alex27 alex28 alex29 alex30
7    alex31 alex32 alex33 alex34 alex35
8    alex36 alex37 alex38 alex39 alex40
9    alex41 alex42 alex43 alex44 alex45
10   alex46 alex47 alex48 alex49 alex50 alex51
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# awk 'NF==5 {print $1}' alex.txt
alex1
alex6
alex11
alex16
alex21
alex26
alex31
alex36
alex41
[root@pylinux tmp]# awk 'NF==6 {print $1}' alex.txt
```

alex46

老师QQ: 877348180



awk的模式

关系运算符	解释	示例
<	小于	x<y
<=	小于等于	x<=y
==	等于	x==y
!=	不等于	x!=y
>=	大于等于	x>=y
>	大于	x>y
~	匹配正则	x~/正则/
!~	不匹配正则	x!~/正则/

案例

```
[root@pylinux tmp]# awk 'NR>3{print $0}' alex.txt
alex16 alex17 alex18 alex19 alex20
alex21 alex22 alex23 alex24 alex25
alex26 alex27 alex28 alex29 alex30
alex31 alex32 alex33 alex34 alex35
alex36 alex37 alex38 alex39 alex40
alex41 alex42 alex43 alex44 alex45
alex46 alex47 alex48 alex49 alex50 alex51
[root@pylinux tmp]#
[root@pylinux tmp]# awk '$1=="alex36"{print $0}' alex.txt
alex36 alex37 alex38 alex39 alex40
```



awk基础总结

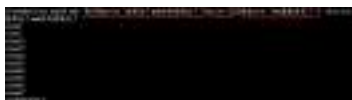
- 空模式，没有指定任何的模式（条件），因此每一行都执行了对应的动作，空模式会匹配文档的每一行，每一行都满足了（空模式）

```
[root@pylinux tmp]# awk '{print $1}' alex.txt
alex1
alex6
alex11
alex16
alex21
alex26
alex31
alex36
alex41
alex46
```

- 关系运算符模式，awk默认执行打印输出动作

```
[root@pylinux tmp]# awk 'NR==2,NR==5' alex.txt
alex6 alex7 alex8 alex9 alex10
alex11 alex12 alex13 alex14 alex15
alex16 alex17 alex18 alex19 alex20
alex21 alex22 alex23 alex24 alex25
```

- BEGIN/END模式（条件设置）



awk与正则表达式

正则表达式主要与awk的 pattern模式（条件） 结合使用

- 不指定模式，awk每一行都会执行对应的动作
- 指定了模式，只有被模式匹配到的、符合条件的行才会执行动作

找出pwd.txt中有以games开头的行

1.用grep过滤

```
[root@pylinux tmp]# cat -n pwd.txt
 1  sync:x:5:0:sync:/sbin:/bin/sync
 2  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
 3  halt:x:7:0:halt:/sbin:/sbin/halt
 4  mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
 5  operator:x:11:0:operator:/root:/sbin/nologin
 6  games:x:12:100:games:/usr/games:/sbin/nologin
 7  ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
 8  nobody:x:99:99:Nobody:/:/sbin/nologin
 9  systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
10  dbus:x:81:81:System message bus:/:/sbin/nologin
11  polkitd:x:999:998:User for polkitd:/:/sbin/nologin
12  libstoragemgmt:x:998:997:daemon account for libstoragemgmt:/var/run/lsm:/
   sbin/nologin
13  rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
14  ntp:x:38:38::/etc/ntp:/sbin/nologin

[root@pylinux tmp]# grep '^games' pwd.txt
games:x:12:100:games:/usr/games:/sbin/nologin
```

2.awk怎么办?

```
[root@pylinux tmp]# awk '/^games/{print $0}' pwd.txt
games:x:12:100:games:/usr/games:/sbin/nologin

#省略写法
[root@pylinux tmp]# awk '/^games/' pwd.txt
games:x:12:100:games:/usr/games:/sbin/nologin
```

awk使用正则语法

grep '正则表达式' pwd.txt

awk '/正则表达式/动作' /etc/passwd

awk命令使用正则表达式，必须把正则放入 "/" 双斜杠中，匹配到结果后执行动作{print \$0}，打印整行信息

grep可以过滤，那我还用你awk干啥？

awk强大的格式化文本

```
[root@pylinux tmp]# cat pwd.txt
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:system message bus:/:/sbin/nologin
polkitd:x:999:999:User for polkitd:/:/sbin/nologin
libstoragemgmt:x:988:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
nawerwertp:x:38:38:/:etc/ntp:/sbin/nologin
nqwerqstp:x:38:38:/:etc/ntp:/sbin/nologin
nqweqsdtp:x:38:38:/:etc/ntp:/sbin/nologin
nqwetp:x:38:38:/:etc/ntp:/sbin/nologin

[root@pylinux tmp]#
[root@pylinux tmp]# awk -F ":" 'BEGIN{printf "%-10s\t%-10s\n", "用户名", "用户id"} /\n/ {printf "%-10s\t%-10s\n", $1, $3}' pwd.txt
用户名      用户id
nobody       99
ntp           38
nawerwertp   38
nqwerqstp    38
nqweqsdtp    38
nqwetp       38
```

```
[root@pylinux tmp]# awk -F ":" 'BEGIN{printf "%-10s\t%-10s\n", "用户名", "用户id"} /\n/ {printf "%-10s\t%-10s\n", $1, $3}' pwd.txt
用户名      用户id
nobody       99
ntp           38
nawerwertp   38
nqwerqstp    38
nqweqsdtp    38
nqwetp       38
```

awk命令执行流程

解读需求：从pwd.txt文件中，寻找我们想要的信息，按照以下顺序执行

```
awk 'BEGIN{ commands } pattern{ commands } END{ commands }'
```

1. 优先执行 BEGIN{} 模式中的语句
2. 从pwd.txt文件中读取第一行，然后执行 pattern{commands} 进行正则匹配 /\n/ 寻找n开头的行，找到了执行 {print} 进行打印
3. 当awk读取到文件数据流的结尾时，会执行 END{commands}

找出pwd.txt文件中禁止登录的用户（ /sbin/nologin ）

正则表达式中如果出现了 "/" 则需要进行转义

找出pwd.txt文件中禁止登录的用户（ /sbin/nologin ）

1.用grep找出

```
grep '/sbin/nologin$' pwd.txt
```

老师QQ: 877348180



2. awk用正则得用双斜杠 /正则表达式/

```
[root@pylinux tmp]# awk '/\sbin\nologin${print $0}' pwd.txt
```



找出文件的区间内容

1. 找出mail用户到nobody用户之间的内容

正则模式

```
awk '/正则表达式/{动作}' file.txt
```

行范围模式

```
awk '/正则 1/,/正则 2/{动作}' file.txt
```

```
[root@pylinux tmp]# awk '/^mail/,/^nobody/ {print $0}' pwd.txt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```

2. 关系表达式模式

```
[root@pylinux tmp]# awk 'NR>=4 && NR<=8 {print $0}' pwd.txt
```

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```

awk企业实战nginx日志

Access.log

```
39.96.187.239 - - [11/Nov/2019:10:08:01 +0800] "GET / HTTP/1.1" 302 0 "-" "Zabbix"
211.162.238.91 - - [11/Nov/2019:10:08:02 +0800] "GET /api/v1/course_sub/category/list/?belong=1 HTTP/1.1" 200 363 "https://www.luffycity.com/free" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36"
211.162.238.91 - - [11/Nov/2019:10:08:02 +0800] "GET /api/v1/degree_course/ HTTP/1.1" 200 370 "https://www.luffycity.com/free" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36"
```

统计日志的访客ip数量

```
#sort -n 数字从大到小排序
#wc -l 统计行数，也就是ip的条目数
[root@pylinux tmp]# awk '{print $1}' 500access.log |sort -n|uniq|wc -l
75
```

查看访问最频繁的前10个ip

```
#uniq -c 去重显示次数
#sort -n 从大到小排序
```

1. 先找出所有ip排序，排序，然后去重统计出现次数

```
awk '{print $1}' 500access.log |sort -n |uniq -c
```

2. 再次从大到小排序，且显示前100个ip

```
[root@pylinux tmp]# awk '{print $1}' 500access.log |sort -n |uniq -c |sort -nr |head -10
```

```
32 113.225.0.211
22 119.123.30.32
21 116.30.195.155
20 122.71.65.73
18 163.142.211.160
16 39.96.187.239
16 124.200.147.165
16 101.249.53.64
14 120.228.193.218
```

```
14 113.68.155.221
```



老师QQ: 877348180

awk动作

老师QQ: 877348180

awk数组

老师QQ: 877348180

awk内置函数

老师QQ: 877348180

三剑客练习题

grep练习题

a.找出有关`root`的行

```
[root@chaogelinux init.d]# grep 'root' /etc/passwd
```

b.找出`root`开头的行

```
[root@chaogelinux init.d]# grep '^root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

c.匹配以`root`开头或者以`yu`开头的行，注意定位锚点

```
[root@chaogelinux init.d]# grep -E "^(root|yu)\>" /etc/passwd
```

d.过滤出`bin`开头的行，切显示行号

```
[root@chaogelinux init.d]# grep '^bin' /etc/passwd -n
2:bin:x:1:1:bin:/bin:/sbin/nologin
```

e.过滤出除了`root`开头的行

```
[root@chaogelinux init.d]# grep -v '^root' /etc/passwd -n
```

f.统计`yu`用户出现的次数

```
[root@chaogelinux init.d]# grep -c '^yu' /etc/passwd
3
```

g.匹配`yu`用户，最多2次

```
[root@chaogelinux init.d]# grep -m 2 '^yu' /etc/passwd
```

h.匹配多文件，列出存在信息的文件名字

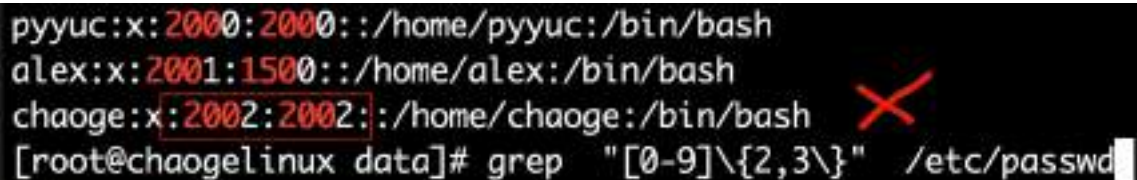
```
[root@chaogelinux data]# grep -l "root" pwd.txt  pwd2.txt  test1.sh
pwd.txt
pwd2.txt
```

1.显示/etc/passwd文件中不以/bin/bash结尾的行*

```
grep -v "/bin/bash$" /etc/passwd
```

2.找出/etc/passwd文件中的两位数或三位数

```
grep "[0-9]\{2,3\}" /etc/passwd #注意这个，是找出包含了2或3个数字的行，不严谨，有误
```



```

puyuc:x:2000:2000:./home/puyuc:/bin/bash
alex:x:2001:1500:./home/alex:/bin/bash
chaoge:x:2002:2002:./home/chaoge:/bin/bash
[root@chaogelinux data]# grep "[0-9]\{2,3\}" /etc/passwd

```

正确思路，匹配完整的单词，只找到2-3个数字

```
[root@chaogelinux data]# grep "\<[0-9]\{2,3\}\>" /etc/passwd
```



```

[root@chaogelinux data]# grep "\<[0-9]\{2,3\}\>" /etc/passwd
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:./sbin/nologin
systemd-network:x:192:192:systemd Network Management:./sbin/nologin
dbus:x:81:81:System message bus:./sbin/nologin
polkitd:x:999:998:User for polkitd:./sbin/nologin
libstoragemgmt:x:998:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
ntp:x:38:38:./etc/ntp:/sbin/nologin
abrt:x:173:173:./etc/abrt:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
postfix:x:89:89:./var/spool/postfix:/sbin/nologin
chrony:x:997:995:./var/lib/chrony:/sbin/nologin

```

3.找出文件中，以至少一个空白字符开头，后面是非空字符的行

```
[root@chaogelinux data]# cat lovers.txt
```

```

I like my lover.
I love my lover.
He likes his lovers.
He loves his lovers.
#she loves her cat

```

```

[root@chaogelinux data]# grep -E "^[[:space:]]+[^[:space:]]" lovers.txt
I like my lover.
He loves his lovers.

```

4. 找出 *lovers.txt* 文件中，所有大小写 *i* 开头的行

```
[root@chaogelinux data]# cat lovers.txt
I like my lover.
I love my lover.
He likes his lovers.
He loves his lovers.
#she loves her cat
i want ride my bike

[root@chaogelinux data]# grep -i "^i" lovers.txt
I love my lover.
i want ride my bike

[root@chaogelinux data]# grep "^[iI]" lovers.txt
I love my lover.
i want ride my bike

[root@chaogelinux data]# grep -E "(i|I)" lovers.txt
I love my lover.
i want ride my bike
```

5. 找出系统上 *root*、*yu*、*nobody* 用户的信息

注意，机器上可能存在多个近似用户，精确搜索得加上 <>

```
[root@chaogelinux data]# grep -E "^\\<(root|yu|nobody)\\>" /etc/passwd
root:x:0:0:root:/root:/bin/bash
nobody:x:99:99:Nobody:/:/sbin/nologin
yu:x:1000:1000:./home/yu:/bin/bash
```

6. 找出 */etc/init.d/functions* 文件中的所有函数名

提示：找出这样的结果

```
checkpid()
checkpids()
kill()
run()
pidof()
daemon()
killproc()
```

```
[root@chaogelinux init.d]# grep -E "[a-zA-Z]+\(\)" /etc/init.d/functions -o
[root@chaogelinux init.d]# grep -E "[[:alnum:]]+\(\)" /etc/init.d/functions -o
```

7. 找出用户名和 *shell* 相同的用户

```
[root@chaogelinux init.d]# grep -E "^([:]+>).*\1$" /etc/passwd
```

```
[root@chaogelinux init.d]# grep -E "^([:]+>).*\1$" /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt 以非冒号开头的单词，中间是任意，结尾取出分组的值
[root@chaogelinux init.d]#
```

sed练习题

提示，sed命令加上-i参数将结果写入到文件

准备文件pwd2.txt

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
ba:x:1002:1002::/home/zhangy:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/false
mail:x:8:12:mail:/var/spool/mail:/bin/false
ftp:x:14:11:ftp:/home/ftp:/bin/false
&nobody:$:99:99:nobody:./bin/false
http:x:33:33:./srv/http:/bin/false
dbus:x:81:81:System message bus:./bin/false
hal:x:82:82:HAL daemon:./bin/false
mysql:x:89:89:./var/lib/mysql:/bin/false
aaa:x:1001:1001:./home/aaa:/bin/bash
test:x:1003:1003:./home/test:/bin/bash
```

a. 替换文件的root为chaoge，只替换一次，与替换所有

```
sed 's/root/chaoge/' pwd2.txt |grep chaoge
sed 's/root/chaoge/g' pwd2.txt |grep chaoge
```

b. 替换文件所有的root为chaoge，且仅仅打印替换的结果

```
[root@chaogelinux data]# sed 's/root/chaoge/gp' pwd2.txt -n
chaoge:x:0:0:chaoge:/chaoge:/bin/bash
```

c. 替换前10行b开头的用户，改为C，且仅仅显示替换的结果

```
[root@chaogelinux data]# sed -n "1,10s/^b/C/gp" pwd2.txt
```

d. 替换前10行b开头的用户，改为C，且将m开头的行，改为M，且仅仅显示替换的结果

```
[root@chaogelinux data]# sed -n -e "1,10s/^b/C/pg" -e "1,10s/^m/M/gp" pwd2.txt
```

e.删除4行后面所有

```
[root@chaogelinux data]# sed '4,$d' pwd2.txt
```

f.删除从root开始，到ftp之间的行

```
[root@chaogelinux data]# sed '/^root/,/^ftp/d' pwd2.txt
```

准备文件2

```
[root@chaogelinux data]# cat lovers.txt
I like my lover.
I love my lover.
He likes his lovers.
  He loves his lovers.
#she loves her cat
```

1.将文件中空白字符开头的行，添加注释符

```
[root@chaogelinux data]# sed -e 's/^[[:space:]]/#/g' -e 's/^$/#/g' lovers.txt
#I like my lover.
I love my lover.
He likes his lovers.
#He loves his lovers.
#she loves her cat
#
#
#
```

2.删除文件的空白和注释行

```
[root@chaogelinux data]# sed '/^$/d;/^#/d' lovers.txt
I like my lover.
I love my lover.
He likes his lovers.
  He loves his lovers.
```

3.给文件前三行，添加@符号

```
[root@chaogelinux data]# sed '1,3s/^(^\.)/#\1/g' lovers.txt
# I like my lover.
#I love my lover.
#He likes his lovers.
  He loves his lovers.
```

```
#she loves her cat
```

```
[root@chaogelinux data]# sed '1,3s/^(^.\)/#\1/g' lovers.txt
# I like my lover.
#I love my lover.
#He likes his lovers.
He loves his lovers.
#she loves her cat
```

1,3s 替换1到3行内容

\(^.\) 转义符 () 分组 ^匹配第一个字符 放入第一个组中, 后面用\1引用分组的结果

结论: 找出前3行的内容, 替换第一个字符为"#字符" 即, 在前3行开头, 添加注释符

4.sed取出ip地址

#多次管道符编辑

```
[root@chaogelinux data]# ifconfig eth0 | sed -n '2p' | sed 's/^.*.inet//' | sed 's/netmask.*//'
```

10.141.32.137

#利用分组功能, 引用ip地址

```
[root@chaogelinux data]# ifconfig eth0 | sed -n '2p' | sed -r 's/^.*.inet(.*)netmask.*\1/'
```

10.141.32.137

#sed支持扩展正则 -r参数

```
[root@chaogelinux data]# ifconfig eth0 | sed -r -n '2s/^.*.inet (.*)netmask.*\1/p'
```

10.141.32.137

5.找出系统版本

```
[root@chaogelinux data]# cat /etc/centos-release
```

CentOS Linux release 7.7.1908 (Core)

```
[root@chaogelinux data]# sed -r -n 's/.*release[[:space:]]*([^.]+).*/\1/p' /etc/centos-release
```

7

```
[root@chaogelinux data]# cat /etc/centos-release
CentOS Linux release 7.7.1908 (Core)
```

```
[root@chaogelinux data]# sed -r -n 's/.*release[[:space:]]*([^.]+).*/\1/p' /etc/centos-release
```

7

release[[:space:]] 匹配到 CentOS Linux release

([^.]+) 分组匹配, 找到了小数点以外的内容, 一次或者多次, 匹配到 7

* 代表结尾的任意内容

awk练习题

1.在当前系统中打印出所有普通用户的用户和家目录 (/etc/passwd)

以passwd文件中root一行为例介绍各个字段作用

1	2	3	4	5	6	7
root	!x	:0	:0	:root	:/root	/bin/bash
用户名称	: 用户密码	: 用户UID	: 用户组GIUD	: 用户说明	: 用户家目录	: shell解释

```
[root@chaogelinux ~]# cat /etc/passwd | awk -F ":" '$3>=1000{print $1,"\t\t",$NF}'
```

2.给/tmp/chaoge.txt文件的前五行，添加#号

```
[root@chaogelinux tmp]# cat chaoge.txt
爱的魔力转圈圈1 爱的魔力转圈圈2 爱的魔力转圈圈3
爱的魔力转圈圈4 爱的魔力转圈圈5 爱的魔力转圈圈6
爱的魔力转圈圈7 爱的魔力转圈圈8 爱的魔力转圈圈9
爱的魔力转圈圈10 爱的魔力转圈圈11 爱的魔力转圈圈12
爱的魔力转圈圈13 爱的魔力转圈圈14 爱的魔力转圈圈15
爱的魔力转圈圈16 爱的魔力转圈圈17 爱的魔力转圈圈18
爱的魔力转圈圈19 爱的魔力转圈圈20

[root@chaogelinux tmp]# awk 'NR<6{print "#"$0}' chaoge.txt
```

3.统计文本信息

姓名 区号 电话 三个月捐款数量

```
Mike Harrington:[510] 548-1278:250:100:175

Christian Dobbins:[408] 538-2358:155:90:201

Susan Dalsass:[206] 654-6279:250:60:50

Archie McNichol:[206] 548-1348:250:100:175

Jody Savage:[206] 548-1278:15:188:150

Guy Quigley:[916] 343-6410:250:100:175

Dan Savage:[406] 298-7744:450:300:275

Nancy McNeil:[206] 548-1278:250:80:75
```



```
John Goldenrod:[916] 348-4278:250:100:175

Chet Main:[510] 548-5258:50:95:135

Tom Savage:[408] 926-3456:250:168:200

Elizabeth Stachelin:[916] 440-1763:175:75:300
```

显示所有电话号码

提示:

```
awk -F "[:]" '{print $1,$2}' tel.txt #见到冒号就切一刀
```

```
awk -F "[ ]" '{print $1,$2}' tel.txt #见到空格就切一刀
```

```
awk -F "[ :]" '{print $1,$2,$3,$4}' tel.txt #见到空格或冒号，都切一刀
```

答案:

```
awk -F "[ :]" '!/^$/{print $4}' tel.txt #排除空行，取出电话
```

显示Tom的电话

```
[root@chaogelinux tmp]# awk -F "[ :]+" '/^Tom/{print $4}' tel.txt
926-3456
```

显示Nancy的姓名、区号、电话

```
[root@chaogelinux tmp]# awk -F "[ :]" '/^Nancy/{print $1,$2,$4}' tel.txt
Nancy McNeil 548-1278
```

显示所有D开头的姓

```
[root@chaogelinux tmp]# awk -F "[ :]" '/^D/{print $2}' tel.txt
Savage
[root@chaogelinux tmp]# awk -F "[ :]" '$2~/^D/{print $2}' tel.txt
Dobbins
Dalsass
```

匹配D开头的行，不对

针对第二列进行正则匹配

```
[root@chaogelinux tmp]# awk -F "[ :]" '/^D/{print $2}' tel.txt
Savage
[root@chaogelinux tmp]# awk -F "[ :]" '$2~/^D/{print $2}' tel.txt
Dobbins
Dalsass
```

显示所有区号是916的人名

```
#针对第三列匹配正则，打印第一列
[root@chaogelinux tmp]# awk -F "[:]" '$3~/\[916\]/{print $1}' tel.txt
Guy
John
Elizabeth
```

显示Mike的捐款信息，在每一款前加上美元符

```
[root@chaogelinux tmp]# awk -F "[:]" '/^Mike/{print "$"$(NF-2), "$"$(NF-1), "$"$(NF)}' tel.txt
$250 $100 $175
```

显示所有人的`姓+逗号+名`

```
[root@chaogelinux tmp]# awk -v FS="[:]" -v OFS="," '!/^$/{print $2,$1}' tel.txt
Harrington, Mike
Dobbins, Christian
Dalsass, Susan
McNichol, Archie
Savage, Jody
Quigley, Guy
Savage, Dan
McNeil, Nancy
Goldenrod, John
Main, Chet
Savage, Tom
Stachelin, Elizabeth
```

删除文件的空白行

```
[root@chaogelinux tmp]# awk '!/^$/{print $0}' tel.txt
[root@chaogelinux tmp]# awk '!/^$/' tel.txt
```

Linux定时任务



你每天是怎么起床的？有的人有女朋友，，或是男朋友，，而我是被穷醒的，，，

什么是计划任务： 后台运行，到了预定的时间就会自动执行的任务，前提是：事先手动将计划任务设定好。

- 周期性任务执行
- 清空/tmp目录下的内容
- mysql数据库备份
- redis数据备份

这就用到了crond服务。

检查crond服务相关的软件包

```
[root@MiWiFi-R3-srv ~]# rpm -qa |grep cron
cronie-anacron-1.4.11-14.el7.x86_64
crontabs-1.11-6.20121102git.el7.noarch
cronie-1.4.11-14.el7.x86_64      #定时任务主程序包，提供crond守护进程等工具

rpm -ivh  安装rpm软件
rpm -qa  查看软件是否安装
rpm -ql  查看软件详细信息s
rpm -qf  查看命令属于的安装包
rpm -e   卸载软件
```

检查crond服务是否运行

```
systemctl status crond.service  #centos7
service crond status           #centos6
```

crond定时任务服务应用

cron 定时任务的看字

crond 定时任务进程名

crontab 管理定时任务命令

Cron是Linux系统中以后台进程模式周期性执行命令或指定程序任务的服务软件名。

Linux系统启动后，cron软件便会启动，对应的进程名字叫做crond，默认是定期（每分钟检查一次）检查系统中是否有需要执行的任务计划，如果有，则按计划进行，好比我们平时用的闹钟。

- crond定时任务默认最快的频率是每分钟执行
- 若是需要以秒为单位的计划任务，则编写shell脚本更格式，crond不适用了

```
#秒级shell脚本
[root@pylinux tmp]# cat test_cron.sh
#!/bin/bash
while true
do
echo "超哥还是强呀"
sleep 1
done
```

```
[root@pylinux tmp]# cat test_cron.sh
#!/bin/bash
while true
do
echo "超哥还是强呀"
sleep 1
done
[root@pylinux tmp]#
[root@pylinux tmp]#
[root@pylinux tmp]# sh test_cron.sh
超哥还是强呀
超哥还是强呀
超哥还是强呀
超哥还是强呀
超哥还是强呀
```

为什么需要crond定时任务

- 夜间数据库定时备份
- 夜间网站数据（用户上传、文件、图片、程序）备份
- 备份等待时间过长
- 任务重复性高

利用Linux的定时任务cron工具可以解决重复性、周期性的、自动备份等运维工作

linux下定时任务软件

- at 定时任务工具，依赖于 atd 服务，适用于执行一次就结束的调度任务

例如突发任务，某天夜里3点需要临时性备份数据，可以使用at软件

```
语法
HH:MM
```

```

YYYY-mm-dd
noon      正午中午12点
midnight  午夜晚12点
teatime   下午茶时间, 下午四点
tomorrow  明天
now+1min  #一分钟之后
now+1minutes/hours/days/weeks

```

一分钟之后运行ls /opt

```
at now+1min
```

```

[root@chaogelinux ~]# at now+1min      #ctrl+d提交任务
at> ls /data
at> <EOT>
job 2 at Thu Nov 21 10:38:00 2019

```

运行之后, 通过邮件检查

```

[root@chaogelinux ~]#
您在 /var/spool/mail/root 中有新邮件
[root@chaogelinux ~]# mail #通过mail, 检查at的任务结果

```

#检查定时任务

```

at -l #列出等待中的作业
#通过文件交互式读取任务, 不用交互式输入
[root@chaogelinux data]# cat mytasks.at
echo "chaoge 666"
[root@chaogelinux data]# at -f ./mytasks.at now+3min
job 5 at Thu Nov 21 10:51:00 2019

```

#删除任务

```

at -d 6
atrm 6 #效果一样

```

- cron 定时任务依赖于 crond 服务, 启动 crond服务后 通过linux命令 crontab 可以配置周期性定时任务, 是Linux运维最常用的工具

定时任务与邮件服务

任务计划触发执行后, 会通过邮件发送给用户, (并非互联网上的邮件, 而是系统内部的邮件服务)

```

1. 检查服务器端口, 25号邮件端口是否打开, centos5是sendmail, centos6、7是postfix服务
ss -tnl |grep 25
netstat -tnl |grep 25

```

```

2. 发现未启动25端口的话, 则需要启动postfix服务, 用于发送邮件
首先更改postfix配置文件

```

```
vim /etc/postfix/main.cf  
修改如下参数  
inet_interfaces = all  
inet_protocols = all
```

3.启动postfix服务

```
systemctl start postfix
```

本地电子邮件服务

网易邮箱邮件协议解释

```
smtp: simple mail transmission protocol  
pop3: Post Office Protocol  
imap4: Internet Mail Access Protocol
```

mailx命令

了解三个概念：

MTA: Mail Transport Agent , 邮件传送代理, 也就是 postfix 服务

MUA: Mail User Agent , 收发邮件的客户端, 可以是 foxmail , 也可以是 其他客户端

Centos7通过命令 mailx 发送邮件, 通过 mail 命令是收邮件

```
[root@chaogelinux ~]# mailx -s "hello chaoge" chaoge      # 给chaoge系统用户发送邮件, -s  
    添加主题  
hi chaoge,how are you?      #文章内容  
.  
    #输入点, 退出邮件  
EOT      #结束符号, end out
```

mail命令

```

[root@chaogelinux ~]# su - chaoge          切换为 chaoge 系统用户
[chaoge@chaogelinux ~]$
[chaoge@chaogelinux ~]$ mail              输入mail命令检查邮件
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/chaoge": 1 message 1 new
>N 1 root      提示有一封邮件      Thu Nov 21 09:42  18/645  "hello chaoge"
& 1 这里输入1, 代表查看第一封邮件
Message 1: 这里是信息
From root@chaogelinux.localdomain Thu Nov 21 09:42:19 2019 发信人是谁
Return-Path: <root@chaogelinux.localdomain>
X-Original-To: chaoge 收信的目标人, 同时也可以抄送给其他人
Delivered-To: chaoge@chaogelinux.localdomain 收信人的地址
Date: Thu, 21 Nov 2019 09:42:19 +0800 发送时间
To: chaoge@chaogelinux.localdomain 发给谁的
Subject: hello chaoge 邮件标题
User-Agent: Heirloom mailx 12.5 7/5/10 发信人, 用的什么工具, 是mailx
Content-Type: text/plain; charset=us-ascii 邮件编码格式
From: root@chaogelinux.localdomain (root)
Status: R

hi chaoge,how are you? 邮件正文
&

```

按下q退出

```

& q
Held 1 message in /var/spool/mail/chaoge
You have mail in /var/spool/mail/chaoge

```

非交互式发邮件

用chaoge用户给root回一封邮件, 从文本中读取数据

```

[chaoge@chaogelinux ~]$ echo "I fine,thank you root,and you?" > fine.txt
[chaoge@chaogelinux ~]$
[chaoge@chaogelinux ~]$ mail -s "hello root" root < fine.txt
[chaoge@chaogelinux ~]$ logout
您在 /var/spool/mail/root 中有邮件
[root@chaogelinux ~]# mail

```

定时任务cron实践

向crond进程提交任务的方式与at不同, crond需要读取配置文件, 且有固定的文件格式, 通过crontab命令管理文件

cron任务分为两类

• 系统定时任务

crond服务除了会在工作时查看 `/var/spool/cron` 文件夹下的定时任务文件以外，还会查看 `/etc/cron.d` 目录以及 `/etc/anacrontab` 下面的文件内容，里面存放 每天、每周、每月需要执行的系统任务

```
[root@pylinux ~]# ls -l /etc/|grep cron*
-rw-----. 1 root root 541 4月 11 2018 anacrontab
drwxr-xr-x. 2 root root 4096 8月 30 11:08 cron.d          #系统定时任务
drwxr-xr-x. 2 root root 4096 8月 8 2018 cron.daily        #每天的任务
-rw-----. 1 root root 0 4月 11 2018 cron.deny
drwxr-xr-x. 2 root root 4096 8月 8 2018 cron.hourly       #每小时执行的任务
drwxr-xr-x. 2 root root 4096 6月 10 2014 cron.monthly     #每月的定时任务
-rw-r--r-- 1 root root 507 5月 10 2019 crontab
drwxr-xr-x. 2 root root 4096 6月 10 2014 cron.weekly      #每周的定时任务
```

系统定时任务配置文件 `/etc/crontab`

```
[root@chaogelinux data]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin          #路径信息很少，因此定时任务用绝对路径
MAILTO=root                                #执行结果发送邮件给用户

# For details see man 4 crontabs

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,
# | | | | | sat
# | | | | |
# * * * * * user-name command to be executed

#每一行，就是一条周期性任务
user-name 是以某一个用户身份运行任务
command to be executed 任务是什么
```

• 用户定时任务计划

当系统管理员（root）或是普通用户(chaoge)创建了需要定期执行的任务，可以使用 `crontab` 命令配置，

crond服务在启动时，会每分钟查看 `/var/spool/cron` 路径下以 系统用户名 命名的 定时任务文件，以确定是否有需要执行的任务。

```
#root用户有一个定时任务文件
[root@pylinux ~]# ls -l /var/spool/cron/
```



```
总用量 4
-rw----- 1 root root 141 10月  9 14:42 root

#查看此root定时任务文件的内容
[root@pylinux ~]# cat /var/spool/cron/root
*/1 * * * * /usr/local/qcloud/stargate/admin/start.sh > /dev/null 2>&1 &
0 0 * * * /usr/local/qcloud/YunJing/YDCrontab.sh > /dev/null 2>&1 &

#等同于如下命令
[root@pylinux ~]# crontab -l
*/1 * * * * /usr/local/qcloud/stargate/admin/start.sh > /dev/null 2>&1 &
0 0 * * * /usr/local/qcloud/YunJing/YDCrontab.sh > /dev/null 2>&1 &
```

crontab命令

crontab命令被用来提交和管理用户的需要周期性执行的任务，与windows下的计划任务类似

参数	解释	使用示例
-l	list查看定时任务	crontab -l
-e	edit编辑定时任务，建议手动编辑	crontab -e
-i	删除定时任务，提示用户确认删除，避免出错	crontab -i
-r	删除定时任务，移除/var/spool/cron/username文件，全没了	crontab -r
-u user	指定用户执行任务，root可以管理普通用户计划任务	crontab -u chaoge -l

crontab命令就是在修改 /var/spool/cron 中的定时任务文件

用户查看定时任务

```
crontab -l #列出用户设置的定时任务，等于cat var/spool/cron/root
crontab -e #编辑用户的定时任务，等于如上命令编辑的是 vi /var/spool/cron/root文件
```

检查crond服务是否运行

```
[root@pylinux ~]# systemctl is-active crond
active

[root@pylinux ~]# ps -ef|grep crond
root      711      1  0 10月20 ?        00:00:01 /usr/sbin/crond -n
```

定时任务相关的文件

```
/var/spool/cron  定时任务的配置文件所在目录
/var/log/cron    定时任务日志文件
/etc/cron.deny   定时任务黑名单
```



定时任务语法格式

口诀：什么时候做什么事

查看定时任务配置文件

```
[root@luffycity ~]# cat /etc/crontab
```



案例

```
每天上午8点30，去上学
30 08 * * * go to school

每天晚上12点回家回家睡觉
00 00 * * * go home
```

定时任务符号

```
crontab任务配置基本格式:
* * * * * command
分钟(0-59) 小时(0-23) 日期(1-31) 月份(1-12) 星期(0-6, 0代表星期天) 命令
```

第1列表示分钟1~59 每分钟用*或者 */1表示
第2列表示小时1~23 (0表示0点)
第3列表示日期1~31
第4列表示月份1~12
第5列标识号星期0~6 (0表示星期天)
第6列要运行的命令

(注意: day of month和day of week一般不同时使用)
(注意: day of month和day of week一般不同时使用)
(注意: day of month和day of week一般不同时使用)

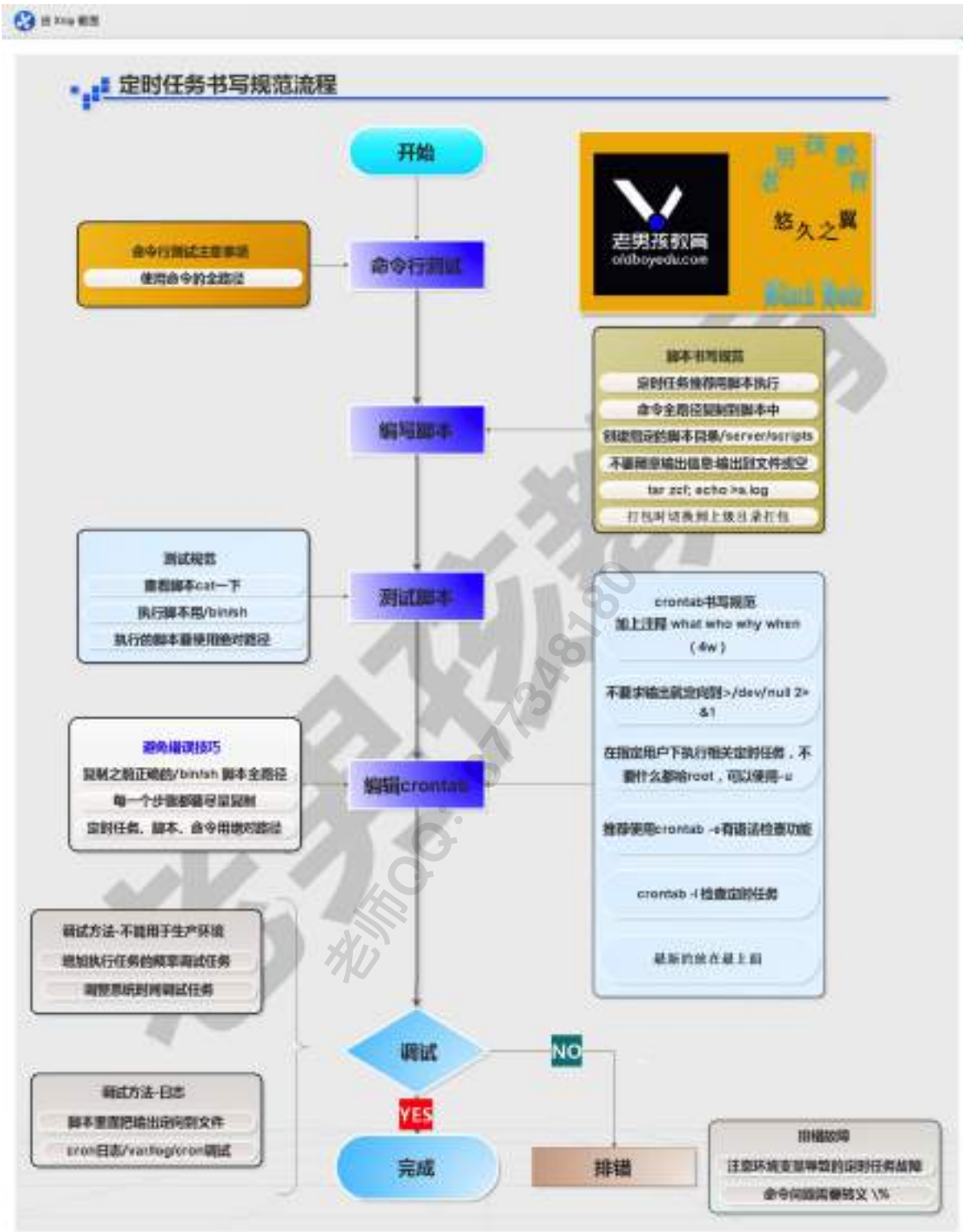
时间表示法:

- 特定值, 时间点有效取值范围内的值
- 通配符, 某时间点有效范围内的所有值, 表示"每"的意思

特殊符号	含义
*	号, 表示"每"的意思, 如 00 23 cmd表示每月每周每日的23:00整点执行命令
-	减号表示时间范围分隔符, 如17-19, 代表每天的17、18、19点
,	逗号, 表示分隔时段, 如30 17,18,19 * cmd 表示每天的17、18、19的半点执行命令
/n	n表示可以整除的数字, 每隔n的单位时间, 如每隔10分钟表示 /10 * cmd

示例

0 * * * * 每小时执行, 每小时的整点执行
1 2 * * 4 每周执行, 每周四的凌晨2点1分执行
1 2 3 * * 每月执行, 每月的3号的凌晨2点1分执行
1 2 3 4 * 每年执行, 每年的4月份3号的凌晨2点1分执行
1 2 * * 3,5 每周3和周五的2点1分执行
* 13,14 * * 6,0 每周六、周日的下午1点和2点的每一分钟都执行
0 9-18 * * 1-5 周一到周五的每天早上9点一直到下午6点的每一个整点(工作日的每个小时整点)
*/10 * * * * 每隔10分钟执行一次任务
*7 * * * * 如果没法整除, 定时任务则没意义, 可以自定制脚本控制频率
定时任务最小单位是分钟, 想完成秒级任务, 只能通过其他方式(编程语言)



案例1

```
*/* * * * * /bin/sh /scripts/data.sh #每分钟执行命令

30 3,12 * * * /bin/sh /scripts/data.sh #每天的凌晨3点半, 和12点半执行脚本

30 */6 * * * /bin/sh /scripts/data.sh #每隔6小时, 相当于6、12、18、24点的半点时刻, 执行脚本
```

```

30 8-18/2 * * * /bin/sh /scripts/data.sh # 30代表半点, 8-18/2表示早上8点到下午18点之间每隔两小时也就是8、10、12、14、16、18的半点时刻执行脚本

30 21 * * * /opt/nginx/sbin/nginx -s reload #每天晚上9点30重启nginx

45 4 1,10 * * /bin/sh /scripts/data.sh #每月的1、10号凌晨4点45执行脚本

10 1 * 6,0 /bin/sh /scripts/data.sh #每周六、周日的凌晨1点10分执行命令

0,30 18-23 * * * #每天的18点到23点之间, 每隔30分钟执行一次

00 */1 * * * /bin/sh /scripts/data.sh #每隔一小时执行一次

00 11 * 4 1-3 /bin/sh /scripts/data.sh #4月份的周一到周三的上午11点执行脚本

```

案例2

```

每天早上7点到上午11点, 每2小时运行cmd命令
00 07-11/2 * * * CMD

0 6 * * * /var/www/test.sh #每天6点执行脚本
0 4 * * 6 /var/www/test.sh #每周六凌晨4:00执行
5 4 * * 6 /var/www/test.sh #每周六凌晨4:05执行
40 8 * * * /var/www/test.sh #每天8:40执行
31 10-23/2 * * * /var/www/test.sh #在每天的10:31开始, 每隔2小时重复一次
0 2 * * 1-5 /var/www/test.sh #每周一到周五2:00
0 8,9 * * 1-5 /var/www/test.sh #每周一到周五8:00, 每周一到周五9:00
0 10,16 * * * /var/www/test.sh #每天10:00、16:00执行

```

生产环境用户配置定时任务流程

需求: 每分钟向 `/testcron/hellochaoge.txt` 文件中写入一句话“超哥带你学linux”

第一步: 新手要注意确保定时任务的正确执行, 切莫编写了定时任务就不管不问

```

[root@pylinux ~]# echo "超哥带你学linux" >> /testcron/hellochaoge.txt
-bash: /testcron/hellochaoge.txt: 没有那个文件或目录
[root@pylinux ~]#
[root@pylinux ~]# mkdir /testcron
[root@pylinux ~]# echo "超哥带你学linux" >> /testcron/hellochaoge.txt
[root@pylinux ~]#
[root@pylinux ~]# cat /testcron/hellochaoge.txt
超哥带你学linux

```

第二步: 编辑定时任务文件, 写入需要定时执行的任务

```

crontab -e
写入

```

```
* * * * * /usr/bin/echo "超哥带你学linux" >> /testcron/hellochaoge.txt
保存后
[root@pylinux ~]# crontab -e
crontab: installing new crontab
```

第三步：检查定时任务

```
[root@pylinux ~]# crontab -l
* * * * * /usr/bin/echo "超哥带你学linux" >> /testcron/hellochaoge.txt
```

第四步：可以检测文件内容

```
tail -f /testcron/hellochaoge.txt
```

每5分钟让服务器进行时间同步

```
crontab -e
*/5 * * * * /usr/sbin/ntpdate ntp1.aliyun.com &> /dev/null
```

每晚0点整，把站点目录/var/www/html下的内容打包备份到/data目录下

- 提醒，tar命令不建议使用绝对路径打包，特殊情况可以使用-P参数

```
1. 检查文件夹是否存在，不存在则创建
[root@pylinux ~]# ls -d /var/www/html /data
ls: 无法访问/var/www/html: 没有那个文件或目录
ls: 无法访问/data: 没有那个文件或目录

2. 创建文件夹
[root@pylinux ~]# mkdir -p /var/www/html /data
[root@pylinux ~]# ls -d /var/www/html /data
/data /var/www/html

3. 创建测试文件
[root@pylinux ~]# touch /var/www/html/chaoge{1..10}.txt
[root@pylinux ~]# ls /var/www/html/
chaoge10.txt chaoge1.txt chaoge2.txt chaoge3.txt chaoge4.txt chaoge5.txt chaoge6.txt chaoge7.txt chaoge8.txt chaoge9.txt

4. 打包压缩命令
[root@pylinux www]# tar -zcvf /data/bak_$(date +%F).tar.gz ./html/
```



编写shell脚本，丢给定时任务定期执行

```
[root@pylinux scripts]# cat bak.sh
#!/bin/bash
cd /var/www && \
/bin/tar -zcf /data/bak_$(date +%F).tar.gz ./html
```

创建定时任务

```
crontab -e
写入
00 00 * * * /bin/sh /server/scripts/bak.sh > /dev/null 2>&1
```

#解释 >/dev/null 2>&1 代表把所有输出信息重定向到黑洞文件

> 是重定向符号

/dev/null是黑洞文件

2>&1 代表让标准错误和标准输出一样

此命令表示将脚本执行的正常或者错误日志都重定向到/dev/null，也就是什么都不输出

>/dev/null 2>&1 等价于 1>/dev/null 2>/dev/null 等价于 &> /dev/null

取消定时任务发邮件功能

1. 定时任务的命令 > /dev/null #命令的执行正确结果输出到黑洞，标准错误还是报错

2. 定时任务的命令 &> /dev/null #组合符 &> 正确和错误的输出，都写入黑洞，危险命令，有风险，慎用

补充anacron

如果由于机器故障关机，定时任务未执行，下次开机也不会执行任务

使用anacron下次开机会扫描定时任务，将未执行的，全部执行，服务器很少关机重启，所以忽略。