

导航

博客园
新随笔
联系
已订阅 
管理

| | | | | | | |
|-------------|----|----|----|----|----|----|
| < 2020年7月 > | | | | | | |
| 日 | 一 | 二 | 三 | 四 | 五 | 六 |
| 28 | 29 | 30 | 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 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

公告

昵称: R_e
园龄: 4年2个月
粉丝: 770
关注: 1
-取消关注

搜索

找找看

谷歌搜索

最新随笔

- 1.nginx中的server_name s_hash_max_size和serve r_names_hash_bucket_si ze
- 2.想要资源的就加群吧，懒得每次发链接了
- 3.Docker 009 容器的资源限制
- 4.Docker 008 自建Registr Y
- 5.Docker 007 镜像的发布与删除
- 6.Docker 006 Dockerfile 指令
- 7.docker 004 镜像和仓库
- 8.Docker 005 构建镜像
- 9.docker 003 基本操作
- 10.docker 002 安装docke r-ce

我的标签

python(4)
re(1)
编码(1)
翻译(1)
官方文档(1)
简介(1)
迁移(1)
入门(1)
算法(1)
正则表达式(1)
更多

随笔分类 (69)

Django(2)
Django2.0.1 文档翻译(13)
docker(15)
GO语言(2)
Linux(1)
openstack(4)
Python(14)
zabbix(1)
随笔(12)
学习方法(1)
运维(4)

随笔档案 (73)

2020年7月(2)

Ubuntu学习——第一篇

一、Ubuntu简介

Ubuntu（乌班图）是一个基于Debian的以桌面应用为主的Linux操作系统，据说其名称来自非洲南部祖鲁语或科萨语的“ubuntu”一词，意思是“人性”、“我的存在是因为大家的存在”，是非洲传统的一种价值观。

Ubuntu的目标在于为一般用户提供一个最新同时又相当稳定，主要以自由软件建构而成的操作系统。Ubuntu目前具有庞大的社区力量支持，用户可以方便地从社区获得帮助。

二. 安装

ubuntu官方网站：<http://www.ubuntu.com/> 对应 中文地址为 http://www.ubuntu.org.cn/index_kylin

桌面版下载地址：<http://www.ubuntu.com/download/desktop>

目前最新版本是： Ubuntu 16.04.1 LTS，建议下载： Ubuntu 16.04.1 Desktop (64-bit)

虚拟机软件： vmware /VirtualBox，mac下还可以使用：parallels，其中VirtualBox是开源免费的。

.....

三、安装过程中的知识点：

虚拟机的网络类型的简单理解：

虚拟机是在我们的操作系统里使用软件模拟出来的，相当于虚拟机是寄宿在我们的真实的物理机的操作系统里的，虚拟机和物理机之间的关系是 寄宿与被寄宿的关系，真实的物理机被称为宿主机。

1. bridged（桥接模式）： 我们的电脑在上网的时候都需要有一个网络地址（IP地址），通过这个地址可以确定我们的电脑在网络上的位置，桥接模式就是我们将我们虚拟机中的网卡的网络地址 放在我们真实的物理机的网卡上。这样的话，我们的虚拟机就好像跟我们的宿主机所在的局域网中的一台机器一样。桥接模式适合有路由器的情况，和真实的物理环境一样。

2. NAT（网络地址转换模式）： 在宿主机上制作一个虚拟网卡，通过这个网卡，给虚拟机分配IP。宿主机在这里的角色相当于局域网中的路由器。NAT模式适合于没有路由器的情况，虚拟机通过宿主机去上网。

3.Host-Only（模式）： 和NAT模式很像，唯一的区别是，没有地址转换服务，所以该模式下虚拟机只能访问到主机。无法访问外网。

- 2020年4月(1)
- 2020年3月(5)
- 2020年1月(6)
- 2019年12月(1)
- 2019年10月(1)
- 2019年3月(2)
- 2018年12月(2)
- 2018年11月(2)
- 2018年9月(4)
- 2018年8月(3)
- 2018年7月(1)
- 2018年6月(1)
- 2018年5月(3)
- 2018年4月(1)
- 2018年2月(4)
- 2018年1月(9)
- 2017年9月(1)
- 2017年8月(1)
- 2017年4月(4)
- 2017年1月(3)
- 2016年12月(1)
- 2016年10月(1)
- 2016年9月(2)
- 2016年8月(8)
- 2016年6月(1)
- 2016年5月(3)

文章分类 (4)

- PYQT(1)
- SaltStack(3)

文章档案 (23)

- 2017年10月(4)
- 2017年7月(2)
- 2017年6月(1)
- 2017年4月(9)
- 2016年12月(1)
- 2016年11月(4)
- 2016年8月(2)

Links

- 银角大王
- 金角大王
- 咆哮金刚猪
- Eva-J
- 帅
- 索大爷
- GO大神-李文周的博客

最新评论

- 1. Re:电脑简史
666666

--来自新疆吐鲁番的小伙子
- 2. Re:电脑结构和CPU、内存、硬盘三者之间的关系
@泽西 机械硬盘安全性比固态硬盘高太多了，追求速度的运算服务器节点直接都使用的大量内存 而且企业级机械的寿命及稳定性也不是固态硬盘可比的...

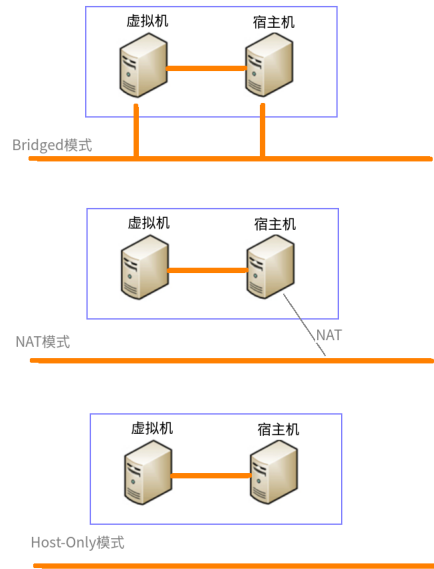
-- | 墨月 |
- 3. Re:北京市图书馆免费入口
又看不了

--程康华
- 4. Re:电脑结构和CPU、内存、硬盘三者之间的关系
还是不错的，非常形象。这句话应该说反了。好点的企业用机械硬盘：一般的固态硬盘：固态硬盘相比机械硬盘性能上领先机械很多的....

--泽西
- 5. Re:elasticsearch-dump 迁移es数据 (elasticdump)
不错不错 学习了

--minseo
- 6. Re:大独裁者最后演讲台词
@ HoneyCY对啊...

--R_e

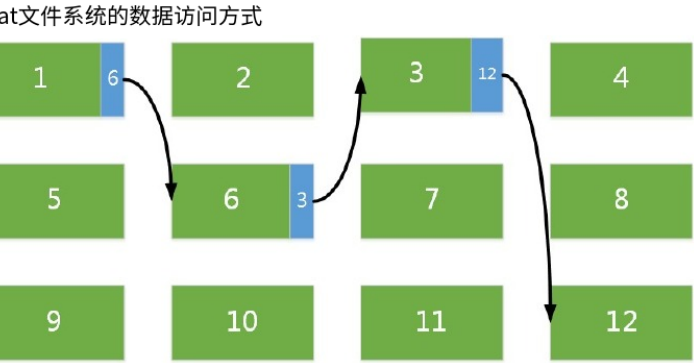
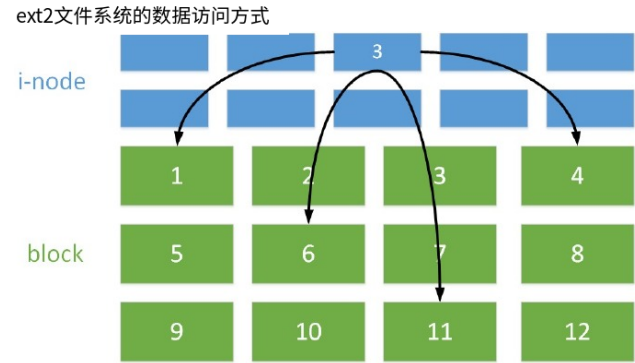


分区：

文件系统类型：默认为 ext4， 文件系统分很多种， ext2、ext3、ext4、fat、ntfs等等

什么是文件系统： 文件系统是操作系统用于明确磁盘或分区上的文件的方法和数据结构；即在磁盘上组织文件的方法。

两种文件系统的对比：



Linux目录结构：

- /： 所有目录都在
- /boot： boot 配置文件、内核和其它启动 时所需的文件
- /etc： 存放系统配置有关的文件
- /home： 存放普通用户目录
- /mnt： 硬盘上手动 挂载的文件系统
- /media： 自动挂载（加载）的硬盘分区以及类似CD、数码相机等可移动介质。
- /cdrom： 挂载光盘？
- /opt： 存放一些可选程序,如某个程序测试版本,安装到该目录的程序的所有数据,库文件都存在同个目录下
- /root： 系统管理员的目录，对于系统来说，系统管理员好比上帝，他可以对系统做任何操作，比如删除你的文件，一般情况下不要使用root用户。

7. Re:面向对象的弊端是什么 (转)
看完了 实话说 不知道接口是什么
--HoneyCY
8. Re:大独裁者最后演讲台词
卓别林 演的大独裁者?
--HoneyCY
9. Re:Ubuntu学习——第一篇
laoge wen
--pp凉
10. Re:Ubuntu学习——第一篇
666
--Man、pp

阅读排行榜

1. Ubuntu学习——第一篇(32361)
2. 电脑结构和CPU、内存、硬盘三者之间的关系(25613)
3. elasticsearch-dump 迁移es数据 (elasticsearchdump) (20708)
4. 给自己的 MAC 添加一个桌面日历(19243)
5. Openstack入坑指南(16754)
6. python中的进程、线程 (threading、multiprocessing、Queue、subprocess) (15482)
7. 计算机中的进制和编码(8653)
8. python3.5 正则表达式(5486)
9. 开始使用Python(5244)
10. openstack cloudinit 遇坑记(4384)

评论排行榜

1. 电脑结构和CPU、内存、硬盘三者之间的关系(5)
2. Ubuntu学习——第一篇(5)
3. 开始使用Python(4)
4. 操作系统简史(3)
5. zabbix问题处理(2)
6. 给自己的 MAC 添加一个桌面日历(2)
7. 大独裁者最后演讲台词(2)
8. elasticsearch-dump 迁移es数据 (elasticsearchdump) (1)
9. 北京市图书馆免费入口(1)
10. 面向对象的弊端是什么 (转) (1)

推荐排行榜

1. Ubuntu学习——第一篇(31)
2. 电脑结构和CPU、内存、硬盘三者之间的关系(7)
3. 电脑简史(6)
4. Openstack入坑指南(4)
5. 开始使用Python(3)

/bin : 存放常用的程序文件 (命令文件)。

/sbin : 系统管理命令, 这里存放的是系统管理员使用的管理程序

/tmp : 临时目录, 存放临时文件, 系统会定期清理该目录下的文件。

/usr : 在这个目录下, 你可以找到那些不适合放在/bin或/etc目录下的额外的工具。比如游戏、打印工具等。/usr目录包含了许多子目录: /usr/bin目录用于存放程序;/usr/share用于存放一些共享的数据, 比如音乐文件或者图标等等;/usr/lib目录用于存放那些不能直接运行的, 但却是许多程序运行所必需的一些函数库文件。/usr/local : 这个目录一般是用来存放用户自编编译安装软件的存放目录; 一般是通过源码包安装的软件, 如果没有特别指定安装目录的话, 一般是安装在这个目录中。

/usr/bin/ 非必要可执行文件 (在单用户模式中不需要); 面向所有用户。

/usr/include/ 标准包含文件。

/usr/lib/ /usr/bin/和/usr/sbin/中二进制文件的库。

/usr/sbin/ 非必要的系统二进制文件, 例如: 大量网络服务的守护进程。

/usr/share/ 体系结构无关 (共享) 数据。

/usr/src/ 源代码,例如:内核源代码及其头文件。

/usr/X11R6/ X Window系统 版本 11, Release 6。

/usr/local/ 本地数据的第三层次, 具体到本台主机。通常而言有进一步的子目录, 例如: bin/、lib/、share/。

/var : 该目录存放那些经常被修改的文件, 包括各种日志、数据文件;

/var/cache/ 应用程序缓存数据。这些数据是在本地生成的一个耗时的I/O或计算结果。应用程序必须能够再生或恢复数据。缓存的文件可以被删除而不导致数据丢失。

/var/lib/ 状态信息。由程序在运行时维护的持久性数据。例如: 数据库、包装的系统元数据等。

/var/lock/ 锁文件, 一类跟踪当前使用中资源的文件。

/var/log/ 日志文件, 包含大量日志文件。

/var/mail/ 用户的电子邮箱。

/var/run/ 自最后一次启动以来运行中的系统的信息, 例如: 当前登录的用户和运行中的守护进程。现已经被/run代替 [13]。

/var/spool/ 等待处理的任务的脱机文件, 例如: 打印队列和未读的邮件。

/var/spool/mail/ 用户的邮箱(不鼓励的存储位置)

/var/tmp/ 在系统重启过程中可以保留的临时文件。

/lib : 目录是根文件系统上的程序所需的共享库, 存放了根文件系统程序运行所需的共享文件。这些文件包含了可被许多程序共享的代码, 以避免每个程序都包含有相同的子程序的副本, 故可以使得可执行文件变得更小, 节省空间。

/lib32 : 同上

/lib64 : 同上

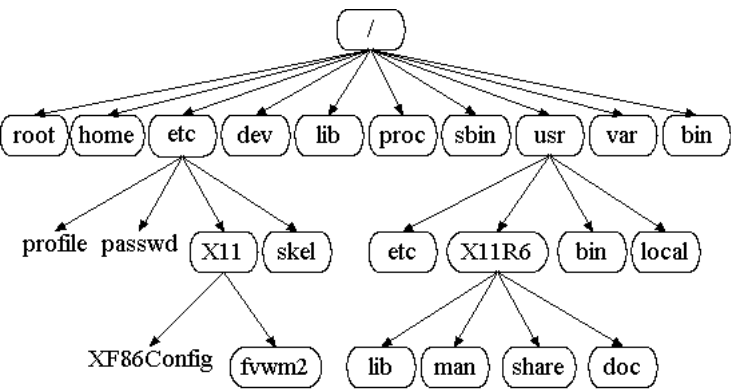
/lost+found : 该目录在大多数情况下都是空的。但当突然停电、或者非正常关机后, 有些文件就临时存放在;

/dev : 存放设备文件

/run : 代替/var/run目录,

/proc : 虚拟文件系统, 可以在该目录下获取系统信息, 这些信息是在内存中由系统自己产生的, 该目录的内容不在硬盘上而在内存里;

/sys : 和proc一样, 虚拟文件系统, 可以在该目录下获取系统信息, 这些信息是在内存中由系统自己产生的, 该目录的内容不在硬盘上而在内存里;



SWAP分区的作用:

当系统的物理内存不够用的时候, 就需要将物理内存中的一部分空间释放出来, 以供当前运行的程序使用。那些被释放的空间可能来自一些很长时间没有什么操作的程序, 这些被释放的空间被临时保存到Swap空间中, 等到那些程序要运行时, 再从Swap中恢复保存的数据到内存中。这样, 系统总是在物理内存不够时, 才进行Swap交换。

sudo cat /proc/sys/vm/swappiness

```
该值默认值是60。

swappiness=0的时候表示最大限度使用物理内存，然后才是 swap空间，

swappiness = 100的时候表示积极的使用swap分区，并且把内存上的数据及时的搬运到swap空间里面。

--临时性修改：

[root@rhce ~]# sysctl vm.swappiness=10

vm.swappiness = 10

[root@rhce ~]# cat /proc/sys/vm/swappiness

10

这里我们的修改已经生效，但是如果我们重启了系统，又会变成60。

--永久修改：

在/etc/sysctl.conf 文件里添加如下参数：

vm.swappiness=10
```

语言环境

查看是否安装了中文支持

```
locale -a
```

如果有 zh_CN.utf8 则表示系统已经安装了中文locale，如果没有则需要安装相应的软件包。安装方式如下：


```
sudo apt-get install language-pack-zh-hans language-pack-zh-hans-base
```

.....

软件管理 apt (Advanced Packaging Tool)，他可以自动下载、配置、安装软件包；简化了Linux系统上的。Debian 及衍生版中都包含了apt，RedHat系列的linux的则使用yum来进行管理，其中Fedora22中Centos7中开始使用dnf来替代yum。



```
apt-cache search package 搜索包
apt-cache show package 获取包的相关信息，如说明、大小、版本等
sudo apt-get install package 安装包
sudo apt-get install package -reinstall 重新安装包
sudo apt-get -f install 强制安装
sudo apt-get remove package 删除包
sudo apt-get remove package -purge 删除包，包括删除配置文件等
sudo apt-get autoremove 自动删除不需要的包
sudo apt-get update 更新源
sudo apt-get upgrade 更新已安装的包
sudo apt-get dist-upgrade 升级系统
sudo apt-get dselect-upgrade 使用 dselect 升级
apt-cache depends package 了解使用依赖
apt-cache rdepends package 了解某个具体的依赖
sudo apt-get build-dep package 安装相关的编译环境
apt-get source package 下载该包的源代码
sudo apt-get clean && sudo apt-get autoclean 清理下载文件的存档
sudo apt-get check 检查是否有损坏的依赖
```



apt的配置文件



```
/etc/apt/sources.list 设置软件包的获取来源
/etc/apt/apt.conf apt配置文件
/etc/apt/apt.conf.d apt的零碎配置文件
/etc/apt/preferences 版本参数
/var/cache/apt/archives/partial 存放正在下载的软件包
/var/cache/apt/archives 存放已经下载的软件包
/var/lib/apt/lists 存放已经下载的软件包详细信息
/var/lib/apt/lists/partial 存放正在下载的软件包详细信息
```



软件源配置文件格式：



```
deb http://security.ubuntu.com/ubuntu xenial-security main restricted
# deb-src http://security.ubuntu.com/ubuntu xenial-security main restricted
deb http://security.ubuntu.com/ubuntu xenial-security universe
# deb-src http://security.ubuntu.com/ubuntu xenial-security universe
deb http://security.ubuntu.com/ubuntu xenial-security multiverse
# deb-src http://security.ubuntu.com/ubuntu xenial-security multiverse
```



Ubuntu 软件仓库被分为四个部分:main (主要的), restricted (受限的), universe (广泛的), multiverse (多元的), 这主要根据我们对软件的支持能力, 以及软件的目的是否符合我们的 自由软件哲学。

先看了一下配置文件的一段内容：

第一个deb表示软件包的格式, 可以是 deb 或 deb-src, 前者表示所指向的存放 binary 格式(已编译), 后者为 sources 格式(源代码)。

第二个URI, 即 Universal Resource Identifier, 通用资源标识符, 可以是以: file(系统)、cdrom(光驱)、http、ftp、copy、rsh、ssh 等几个参数开头的软件包所在位置。

第三个Distribution 指发行版本号, 可以是: stable, testing, unstable, sarge, etch, sid 等, 具体可参考Debian 文档。

后面的几个component表示具体的软件包分类：

- main: 完全遵循 [Debian 自由软件准则](#) 即DFSG的软件包;
- contrib: 软件包均遵循DFSG自由使用原则, 但是其使用了某些不符合DFSG的第三方库;
- non-free: 不符合DFSG的软件包。

dpkg是Debian软件包管理器的基础, 被用于安装、卸载和供给和.deb软件包相关的信息。dpkg本身是一个底层的工具, 本身并不能从远程包仓库下载包以及处理包的依赖的关系, 需要将包从远程下载后再安装。DPKG常用命令：



```
dpkg -i package.deb 安装包
dpkg -r package 删除包
dpkg -P package 删除包 (包括配置文件)
dpkg -L package 列出与该包关联的文件
dpkg -l package 显示该包的版本
dpkg -unpack package.deb 解开 deb 包的内容
dpkg -S keyword 搜索所属的包内容
dpkg -l 列出当前已安装的包
dpkg -c package.deb 列出 deb 包的内容
dpkg -configure package 配置包
```



date : 用来显示或设定系统的日期和与时间



```
date //显示当前日期
# 日期格式化
#      %Y      year
#      %m      month (01..12)
#      %d      day of month (e.g., 01)
#      %H      hour (00..23)
#      %I      hour (01..12)
#      %M      minute (00..59)
#      %S      second (00..60)
date +"%Y%m%d %H%M%S"
20160824 223856
```

```
date +"%Y-%m-%d %H:%M:%S"
2016-08-24 22:39:07

date -s //设置当前时间, 只有root权限才能设置, 其他只能查看。
date -s 20061010 //设置成20061010, 这样会把具体时间设置成空00:00:00
date -s 12:23:23 //设置具体时间, 不会对日期做更改
date -s "12:12:23 2006-10-10" //这样可以设置全部时间

# 注意: 重新设置时间后需要将时间捅不到硬件时钟。方式如下:
hwclock -w
```



cal : 显示一个日历

```
cal # 现实当前月份的日历
cal -y # 显示当年的日历
cal 2016 # # 显示指定年份的日历
```

设置时区

```
tzselect

# 或者

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

修改密码:

```
# 修改密码的命令
passwd # 默认修改当前用户的密码

passwd username # 修改指定用户的密码, 需要管理员权限
```

忘记密码

开始时长按shift键, 进入grub菜单--> 按字母e 进入编辑模式 --> 编辑内容--> 启动 进入但用户模式, 重新设置用户密码, --> 按照F10重启 --> 使用新密码进入系统



```
if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; \
fi
insmod part_msdos
insmod ext2
set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1\
--hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 ca274ec6-9623-42c4\
-8528-5cf0604fc4cc
else
search --no-floppy --fs-uuid --set=root ca274ec6-9623-42c4-852\
8-5cf0604fc4cc
fi
linux /boot/vmlinuz-4.4.0-21-generic root=UUID=ca274ec6-9\
623-42c4-8528-5cf0604fc4cc ro quiet splash $vt_handoff
initrd /boot/initrd.img-4.4.0-21-generic
```


```
if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; \
fi
insmod part_msdos
insmod ext2
set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
  search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1\
--hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 ca274ec6-9623-42c4\
-8528-5cf0604fc4cc
else
  search --no-floppy --fs-uuid --set=root ca274ec6-9623-42c4-852\
8-5cf0604fc4cc
fi
linux /boot/vmlinuz-4.4.0-21-generic root=UUID=ca274ec6-9\
623-42c4-8528-5cf0604fc4cc rw single init=/bin/bash
initrd /boot/initrd.img-4.4.0-21-generic
```

注销/重启/关机

logout # 注销


reboot # 重启系统：需要管理员全新啊

shutdown # 关机：需要管理员权限




```
shutdown -r now # 现在立即重启
shutdown -r +5 # 三分钟后重启
shutdown -r 12:12 #在12:12时将重启计算机

shutdown -h now # 现在立即关机
shutdown -h +5 "The System will shutdown after 3 minutes" # 提示使用者将在三分钟后关机
shutdown -h +5 # 5分钟后关机
shutdown -h 12:00 # 12点钟关机
shutdown -c # 取消关机操作
```



cd : 切换目录



```
cd # 回到当前用户的家目录
# ~ 可用于表示用户家目录
cd /etc # 切换到/etc目录

cd -
# 切换到上一次的目录
```



pwd : 查看当前的工作路径

创建目录:

```
# mkdir 目录名
mkdir my_dir

# -p 参数 : 递归创建目录, 用于同时创建多级目录
mkdir a/b/c/d
```

获取帮助

-h --help info man



```
man man # 查看man命令的手册
man cd
man pwd
```

```
man 5 passwd
man -k passwd # 模糊查找
man -f passwd # 精确查找
```



创建文件

touch : 改变文件或目录的时间, 文件不存在时会创建一个空文件。



```
touch file1 # file1 不存在时被创建
touch -c file1 # 不创建文件
touch -r ref_file file1 更新file1.txt的时间戳和ref+file相同
touch -t 201210120505.25 file1

# -t time 使用指定的时间值 time 作为指定文件相应时间戳记的新值。此处的 # time规定为如下形式的十进制数:
# [[CC]YY]MMDDhhmm[.SS]
# 这里, CC为年数中的前两位, 即“世纪数”; YY为年数的后两位, 即某世纪中的年数。如果不给出CC的值, 则touch 将把年数C
```



注意: 如果文件以 “.” 开头, 则表示文件是隐藏文件。

删除:

rm : 删除命令

```
rm -f file1 # 强制删除文件
rm -r a/b/file1 # 删除指定目录及其下的所有文件和目录
rm -rf a/b/file1 # 强制删除指定目录及其下的所有文件和目录

# rm 命令太危险, 不建议使用
```

mv : 移动或重命名文件或目录

```
mv SOURCE DEST #

mv test.log test.txt # 文件改名
mv test1.txt dir1/ #移动文件
mv test1.txt test2.tx test3.tx dir1/ #移动多个文件
```

cp : 复制



```
cp SOURCE DEST # 复制文件

cp -i SOURCE DEST # 如果遇到需要覆盖的情况, 则提示
cp -r dir1 dir2 # 若给出的源文件是一目录文件, 此时cp将递归复制该目录下所有的子目录和文件。此时目标文件必须为一个
cp -p file1 file2 # 此时cp除复制源文件的内容外, 还将把其修改时间和访问权限也复制到新文件中。
cp -rp dir1 dir2
```



stat : 查看文件相信信息

```
stat filename

# Access time(ctime):是指取用文件的时间, 所谓取用, 常见的操作有: 使用编辑器查看文件内容, 使用cat命令显示文件内容,
# Modify time(mtime): 是指修改文件内容的时间, 只要文件内容有改动 (如使用转向输出或转向附加的方式) 或存盘的操作, 就
# Change time(ctime):是指文件属性或文件位置改动的时间, 如使用chmod, chown, mv指令集使用ln做文件的硬连接, 就会改
```


cat : 链接文件后输出文件内容到屏幕上, 其实就是查看文件内容

tac : 反转行的输出

```
cat file1 #显示 file1的文件内容
cat file1 file2 # 显示file1和file2的文件内容
cat -n file1 # 由1开始对所有输出的行数编号
cat -s file # 当遇到连续2行以上的空白行, 只保留一行空白行
```

wc : 统计指定文件中的字节数、字数、行数, 并将统计结果显示输出

```
-c 统计字节数。
-l 统计行数。
-m 统计字符数。这个标志不能与 -c 标志一起使用。
-w 统计字数。一个字被定义为由空白、跳格或换行字符分隔的字符串
```

sort : 排序



```
sort [-fbMrtuk] [file or stdin]
选项与参数:
-f : 忽略大小写的差异, 例如 A 与 a 视为编码相同;
-b : 忽略最前面的空格符部分;
-n : 使用『纯数字』进行排序 (默认是以文字型态来排序的);
-r : 反向排序;
-u : 就是 uniq , 相同的数据中, 仅出现一行代表;
-t : 分隔符, 默认是用 [tab] 键来分隔;
-k : 以那个区间 (field) 来进行排序的意思
```



uniq : 忽略或报告重复行

```
uniq [-icu]
选项与参数:
-i : 忽略大小写字符的不同;
-c : 进行计数
-u : 只显示唯一的行
```

cut命令可以从一个文本文件或者文本流中提取文本列。

```
选项与参数:
-d : 后面接分隔字符。与 -f 一起使用;
-f : 依据 -d 的分隔字符将一段信息分割成为数段, 用 -f 取出第几段的意思;
-c : 以字符 (characters) 的单位取出固定字符区间;
```

tee : 读取标准输入的数据, 并将其内容输出成文件。

```
cat sec.log | tee file1 # 读取sec.log , 并生成file1文件
cat sec.log | tee - a file1 # 读取sec.log , 并追加,
cat sec.log |tee file1 file2
```

history : 查看执行过的命令。

```
history # 显示最近1000条历史命令
history 5 # 显示最后5条命令
!number# number为history之后命令前的序号: 执行该条命令
!cat # 执行最后一条以cat开头的命令
```

more : 查看文件内容

less : 查看文件内容

head : 输出文件的开始的部分, 可以指定行数, 默认显示10行

```
head -n 5 file
```

tail ： 查看文件尾部的内容。默认显示最后10行

```
tail file1
tail -n 5 file1
tail -f file1 # 动态监控文件
```

which # 查找其他命令的位置

```
which ls
```

ls ： 列出目标目录中所有的子目录和文件

格式: ls [选项] [目录名]

- a 用于显示所有文件和子目录(保罗点文件)。
 - l 除了文件名之外，还将文件的权限、所有者、文件大小等信息详细列出来。
 - r 将目录的内容清单以英文字母顺序的逆序显示。
 - t 按文件修改时间进行排序，而不是按文件名进行排序。
 - A 同-a，但不列出“.”(表示当前目录)和“..”(表示当前目录的父目录)。
 - F 在列出的文件名和目录名后添加标志。例如，在可执行文件后添加“*”，在目录名后添加“/”以区分不同的类型。
 - R 如果目标目录及其子目录中有文件，就列出所有的文件。
- . 和..
- . 表示当前目录
- .. 表示父目录



```
ls # 列出当前目录下的文件和目录
ls . # 列出当前目录下的文件和目录
ls .. # 列出当前目录的父目录下的文件和目录
ls /etc # 列出/etc目录下的文件和目录

ls -l # 以长格式显示文件信息
总用量 76
-rwxrwxrwx 1 will will 78 5月 13 18:11 ss_start.sh
```



文件类型

- 普通文件
- d 目录文件
- b 块设备文件
- c 字符设备文件
- l 链接文件
- p 管道文件
- s socket文件

```
ls -l /dev # 可以查看字符设备文件和块设备文件
ls -l /run # 可以找到socket文件
ls -l /run/systemd/inhibit/ # 可以查看到管道文件
```

文件权限

rwxrwxr-- ： 三组rwx 分别表示 所有者、所有组、其他人 的权限。

r ： 表示可读, 可以用数字 4 来表示

w ： 标识可写 , 可以用数字 2 来表示

x : 表示可执行 , 可以用数字 1 来表示
- : 表示没有相应权限 可以用数字 0 来表示

修改权限的方法:



```
chmod o+w file1
chmod g-w file1
chmod go-w file1
chmod u=rwx file1
```

```
chmod 755 file1 # -rwxr-xr-x (755) 只有所有者才有读, 写, 执行的权限, 组群和其他人只有读和执行的权限
chmod 644 # -rw-r--r-- (644) 只有所有者才有读和写的权限, 组群和其他人只有读的权限
```

```
# 其中:
# u 代表所有者 (user)
# g 代表所有者所在的组群 (group)
# o 代表其他人, 但不是u和g (other)
# a 代表全部的人, 也就是包括u, g和o
```



目录上的权限:

r : 表示是否可以读取目录下的文件名
w : 表示是否可以在目录下创建修改文件
x : 表示目录是否可以被搜索

有x权限后, 就可以使用 ./a.py 的方式执行文件。

chown : 更改文件的所有者和所有组

```
chown root:root file
chown root file
chown :root file
```

特殊权限

SUID: 让一般用户在执行某些程序的时候, 能够暂时具有该程序拥有者的权限, SUID对目录是无效的

SGID: 文件: 如果SGID设置在二进制文件上, 则不论用户是谁, 在执行该程序的时候, 它的有效用户组 (effective group) 将会变成该程序的用户组所有者 (group id); 目录: 如果SGID是设置在某目录上, 则在该目录内所建立的文件或目录的用户组, 将会是该目录的用户组。 SGID多用在特定的多人团队的项目开发上, 在系统中用得较少

STICKY: 只针对目录有效, 在具有SBit的目录下, 用户若在该目录下具有w及x权限, 则当用户在该目录下建立文件或目录时, 只有文件拥有者与root才有权力删除。

rwsrw-r-- 表明有suid标识,

rwxrws--- 表明有sgid标识,

rwxrw-rwt 表明有stick标识,

当设置了特别权限位时, 如果原来这个位上有x, 那么这个特殊标识就显示为小写字母s,s,t, 否则就显示为大写S,S,T, 此时他们不生效。

用户和用户组

linux使用文件保存用户信息:

文件

```
# /etc/passwd 用户账户信息。
# /etc/shadow 安全用户账户信息。
```

```
# /etc/group 组账户信息。
# /etc/gshadow 安全组账户信息。
# /etc/default/useradd 账户创建的默认值。
# /etc/skel/ 包含默认文件的目录。
# /etc/login.defs Shadow 密码套件配置。
```

useradd：添加用户



```
# -c 备注 加上备注。并将此备注文字加在/etc/passwd中的第5项字段中
# -d 用户主文件夹。指定用户登录所进入的目录，并赋予用户对该目录的完全控制权
# -e 有效期限。指定帐号的有效期限。格式为YYYY-MM-DD，将存储在/etc/shadow
# -f 缓冲天数。限定密码过期后多少天，将该用户帐号停用
# -g 主要组。设置用户所属的主要组 www.cit.cn
# -G 次要组。设置用户所属的次要组，可设置多组
# -M 强制不创建用户主文件夹
# -m 强制建立用户主文件夹，并将/etc/skel/当中的文件复制到用户的根目录下
# -p 密码。输入该帐号的密码
# -s shell。用户登录所使用的shell
# -u uid。指定帐号的标志符user id，简称uid

useradd user1 # 添加用户 user1
useradd -d /home/userTT user2
```



userdel：删除用户



```
userdel user1 #
userdel -r user1

# -r, --remove 用户主目录中的文件将随用户主目录和用户邮箱一起删除。在其它文件系统中的文件必须手动搜索并删除。
# -f, --force 此选项强制删除用户账户，甚至用户仍然在登录状态。它也强制删除用户的主目录和邮箱，即使其它用户也依
```



usermod：修改用户信息



```
# -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。

usermod -G staff user2 # 将 newuser2 添加到组 staff 中
usermod -l newuser1 newuser # 修改 newuser 的用户名为 newuser1
usermod -L newuser1 # 锁定账号 newuser1
usermod -U newuser1 # 解除对 newuser1 的锁定
```



groupadd：添加组

```
groupadd group1
groupadd -g 1000 group1 # 指定gid
```

groupdel：删除组

```
groupdel group1 # 删除组
```

su与 sudo

su : 切换用户, 没有参数时, 默认切换为root用户;

```
su    # 切换为root

## 推荐
su -  # 切换为root 并加载user1的环境配置
su - user1 # 切换为用户1 并加载user1的环境配置
```

sudo : 让当前用户暂时以管理员的身份root来执行命令。

Ubuntu 默认没有启用root用户, 普通用户执行一些特殊的操作时, 使用sudo就可以让普通用户以root用户的身份执行命令

sudo有一个配置文件: /etc/sudoers ; 通过修改配置文件可以让指定用户使用sudo命令



man sudoers # 查看man手册

看下面几行:

Host alias specification # 配置Host_Alias: 就是主机的列表

Host_Alias HOST_FLAG = hostname1, hostname2, hostname3

User alias specification # 配置User_Alias: 就是具有sudo权限的用户的列表

User_Alias USER_FLAG = user1, user2, user3

Cmnd alias specification # 配置Cmnd_Alias: 就是允许执行的命令的列表, 命令前加上!表示不能执行此命令. 命令一定要

Cmnd_Alias COMMAND_FLAG = command1, command2, command3, !command4

配置Runas_Alias: 就是用户以什么身份执行 (例如root, 或者oracle) 的列表

Runas_Alias RUNAS_FLAG = operator1, operator2, operator3

User privilege specification

配置权限的格式如下:

USER_FLAG HOST_FLAG=(RUNAS_FLAG) COMMAND_FLAG

root ALL=(ALL:ALL) ALL

如果不需要密码验证的话, 则按照这样的格式来配置

USER_FLAG HOST_FLAG=(RUNAS_FLAG) NOPASSWD: COMMAND_FLAG

格式为: 用户名(用户别名) 主机名(主机别名)=[(运行用户或是Runas_Alias)可选] [tag可选] 可以执行的命令(或Cmnd_Alias)

user1 host1 = /bin/kill # user1 可以在host1上使用命令/bin/kill

user1 host1 = NOPASSWD: /bin/kill # user1 可以在host1上使用命令/bin/kill 同时可以不必输入密码(这里就是使用

user1 host1 = NOPASSWD: /bin/kill, PASSWORD: /bin/ls # user1 可以在host1上使用命令/bin/kill无需输入密

user1 host1 = (operator) /bin/kill # user1 可以在host1上使用命令/bin/kill但是必须是以operator用户运行这

user1 host1 = (:group_name) /bin/kill # user1 可以在host1上使用命令/bin/kill,且必须以group_name这个用户

%group_name host1 = /bin/kill # 所有group_name里面的用户都可以在host1上执行/bin/kill(Linux中一般代表整个用

再举个实际例子, 我之前对sudo su这个命令不理解, 为什么我可以直接就su到root用户了呢, 连密码都不需要? 查看了一下sudoer

xxx ALL=NOPASSWD: /bin/su



alias : 给命令起别名

```
alias ll='ls -aF'
alias la='ls -A'
alias l='ls -CF'
```

如果需要别名永久生效, 需要保存到 .bashrc 文件

我们用到的终端默认使用的shell 是bash 其他的shell 有dash 、csh 、tcsh、zsh等等

Shell本身是一个用C语言编写的程序，它是用户使用Unix/Linux的桥梁，用户的大部分工作都是通过Shell完成的。Shell既是一种命令语言，又是一种程序设计语言。作为命令语言，它交互式地解释和执行用户输入的命令；作为程序设计语言，它定义了各种变量和参数，并提供了许多在高级语言中才具有的控制结构，包括循环和分支。

自定义账户的个性化环境的三个重要文件

`.bash_history` `.bash_logout` `.bashrc`

刚登录Linux时，首先启动 `/etc/profile` 文件，`~/.bash_profile`、`~/.bash_login`、`~/.profile`。如果 `~/.bash_profile`文件存在的话，一般还会执行 `~/.bashrc`文件。

关于各个文件的作用域，在网上找到了以下说明：

- (1) **/etc/profile**: 此文件为系统的每个用户设置环境信息,当用户第一次登录时,该文件被执行. 并从`/etc/profile.d`目录的配置文件搜集shell的设置。
- (2) **/etc/bashrc**: 为每一个运行bash shell的用户执行此文件.当bash shell被打开时,该文件被读取（即每次新开一个终端，都会执行bashrc）。
- (3) **~/.bash_profile**: 每个用户都可使用该文件输入专用于自己使用的shell信息,当用户登录时,该文件仅仅执行一次。默认情况下,设置一些环境变量,执行用户的`.bashrc`文件。
- (4) **~/.bashrc**: 该文件包含专用于你的bash shell的bash信息,当登录时以及每次打开新的shell时,该文件被读取。
- (5) **~/.bash_logout**: 当每次退出系统(退出bash shell)时,执行该文件. 另外,`/etc/profile`中设定的变量(全局)的可以作用于任何用户,而`~/.bashrc`等中设定的变量(局部)只能继承 `/etc/profile`中的变量,他们是"父子"关系。
- (6) **~/.bash_profile**: 也可能是 `.profile` 是交互式、login 方式进入 bash 运行的`~/.bashrc` 是交互式 non-login 方式进入 bash 运行的通常二者设置大致相同，所以通常前者会调用后者。

PATH变量的设置

`env` : 查看当前环境变量

`export` : 设置或显示环境变量。

`source` : 在当前bash环境下读取并执行FileName中的命令。该filename文件可以无"执行权限"

```
env
export name = "SN"
source /etc/profile
```

`echo` `echo`会将输入的字符串送往标准输出。输出的字符串间以空白字符隔开并在最后加上换行号。

`-n` 不要在最后自动换行

`-e` 若字符串中出现以下字符，则特别加以处理，而不会将它当成一般

文字输出：

`\a` 发出警告声；

`\b` 删除前一个字符；

`\c` 最后不加上换行符号；

`\f` 换行但光标仍旧停留在原来的位置；

`\n` 换行且光标移至行首；

`\r` 光标移至行首，但不换行；

`\t` 插入tab；

`\v` 与`\f`相同；

管道符

管道符 就是 `|` : 他的作用是 将前一个命令的结果 交给后一个命令使用

重定向

> 重定向，如果的文件存在，则覆盖文件内容，文件不存在时创建文件

>> 重定向，如果的文件存在，则向文件追加内容，文件不存在时创建文件

1> 标准正确输出，同上

1>> 标准正确输出，同上

2> 标准错误输出, 同上

2>> 标准错误输出, 同上

&> 标准正确输出和标准错误输出, 同上

locate # 查找文件

```
locate /etc/sh # 搜索etc目录下所有以sh开头的文件。
locate ~/a # 搜索用户主目录下, 所有以a开头的文件。
locate -i ~/a # 搜索用户主目录下, 所有以a开头的文件, 并且忽略大小写。
```

find



使用方法:

```
find path -option [-print] [-exec -ok command] {} \;
```

根据文件名查找

find / -name filename 再根目录里面搜索文件名为filename的文件

find /home -name "*.txt"

find /home -iname "*.txt" # 忽略大小写

根据文件类型查找

find . -type 类型参数

f 普通文件

l 符号连接

d 目录

c 字符设备

b 块设备

s 套接字

p Fifo

根据目录深度查找

find . -maxdepth 3 -type f # 最大深度为3

find . -mindepth 2 -type f # 最小深度为2

根据文件的权限或者大小名字类型进行查找

find . -type f -size (+|-)文件大小 # +表示大于 -表示小于

b — 块 (512字节)

c — 字节

w — 字 (2字节)

k — 千字节

M — 兆字节

G — 吉字节

按照时间查找

-atime (+|-) n # 此选项代表查找出n天以前被读取过的文件。

-mtime (+|-) n # 此选项代表查找出n天以前文件内容发生变化的文件。

-ctime (+|-) n # 此选项代表查找出n天以前的文件的属性发生变化的文件。

-newer file # 此选项代表查找出所有比file新的文件。

-newer file1 ! -newer file2 # 此选项代表查找比file1文件时间新但是没有file2时间新的文件。

注意:

n为数字, 如果前面没有+或者-号, 代表的是查找出n天以前的, 但是只是一天之内的范围内发生变化的文件。

如果n前面有+号, 则代表查找距离n天之前的发生变化的文件。如果是减号, 则代表查找距离n天之内的所有发生变化的文件。

-newer file1 ! -newer file2中的!是逻辑非运算符

按照用户/权限查找

-user 用户名: 根据文件的属主名查找文件。

-group 组名: 根据文件的属组名查找文件。

-uid n: 根据文件属主的UID进行查找文件。

-gid n: 根据文件属组的GID进行查找文件。

-nouser: 查询文件属主在/etc/passwd文件中不存在的文件。

-nogroup: 查询文件属组在/etc/group文件中不存在的文件

-perm 777: 查询权限为777的文件

来自: <http://man.linuxde.net/find>

查找时指定多个条件

-o: 逻辑或, 两个条件只要满足一个即可。

-a: 逻辑与, 两个条件必须同时满足。

```
find /etc -size +2M -a -size -10M
```

对查找结果进行处理

```
-exec shell命令 {} \;
```

```
-ok shell命令 {} \;
```

其中-exec就是代表要执行shell命令, 后面加的是shell指令, 再后面的“{}”表示的是要对前面查询到的结果进行查询, 最后的“\”;

```
find /home -name "*.txt" -ok ls -l {} \;
```

```
find /home -name "*.txt" -ok rm {} \;
```



df

-T : 显示文件系统类型

-h : 以能显示的最大单位显示

```
df -Th
```

du



-s : 如果后面是目录, 只显示一层

-h : 以能显示的最大单位显示

```
du dirname # 显示dirname下所有目录及其子目录的大小
```

```
du -sh dirname 显示dirname的大小
```



mount / umount 3 挂载和卸载设备



```
mount # 查询挂在设备及属性
```

挂载光盘

```
mount -t iso9660 /dev/cdrom /mnt
```

```
mount /dev/sr0 /mnt
```

重新挂载设备

```
mount -o remount,rw /mnt # 重新挂载设备并设置rw属性
```

挂载iso文件

```
mount a.iso -o loop /mnt
```

```
umount /mnt # 卸载设备
```

```
umount -l /mnt # 强制卸载
```



crontab




```

* * * * * command to be executed
- - - - -
| | | | |
| | | | | --- 预执行的命令
| | | | | ----- 表示星期0~7 (其中星期天可以用0或7表示)
| | | ----- 表示月份1~12
| | ----- 表示日期1~31
| ----- 表示小时1~23 (0表示0点)
----- 表示分钟1~59 每分钟用*或者 */1表示

-u user: 用来设定某个用户的crontab服务;
-e: 编辑某个用户的crontab文件内容。如果不指定用户, 则表示编辑当前用户的crontab文件。
-l: 显示某个用户的crontab文件内容, 如果不指定用户, 则表示显示当前用户的crontab文件内容。
-r: 从/var/spool/cron目录中删除某个用户的crontab文件, 如果不指定用户, 则默认删除当前用户的crontab文件。
-i: 在删除用户的crontab文件时给确认提示

```



tar



```

-c : 建立一个压缩文件的参数指令(create 的意思);
-x : 解开一个压缩文件的参数指令!
-t : 查看 tarfile 里面的文件!
特别注意 c/x/t 同时仅能存在一个, 因为不可能同时压缩与解压缩。
-z : 是否同时具有 gzip 的属性? 亦即是否需要用 gzip 压缩?
-j : 是否同时具有 bzip2 的属性? 亦即是否需要用 bzip2 压缩?
-v : 压缩的过程中显示文件! 这个常用, 但不建议用在背景执行过程!
-f : 使用档名, 请留意, 在 f 之后要立即接文件名
-p : 使用原文件的原来属性(属性不会依据使用者而变)
-P : 可以使用绝对路径来压缩!
-N : 比后面接的日期(yyyy/mm/dd)还要新的才会被打包进新建的文件中!

```

```

# 将当前目录下所有.txt文件打包并压缩归档到文件this.tar.gz
tar czvf this.tar.gz ./*.txt
# 将当前目录下的this.tar.gz中的文件解压到当前目录
tar xzvf this.tar.gz ./

# 将整个 /etc 目录下的文件全部打包成为 /tmp/etc.tar
tar -cvf /tmp/etc.tar /etc # 仅打包, 不压缩!
tar -zcvf /tmp/etc.tar.gz /etc # 打包后, 以 gzip 压缩
tar -jcvf /tmp/etc.tar.bz2 /etc # 打包后, 以 bzip2 压缩

# 解压文件
tar -xf a.tar.gz #
tar -xf a.tar.gz -C /tmp # 指定解包路径

```



grep



```

格式:
grep [OPTIONS] PATTERN [FILE...]
grep [OPTIONS] [-e PATTERN] [FILE...]

参数:
-c --count #计算符合样式的列数
-l --file-with-matches #列出文件内容符合指定的样式的文件名称。
-v --invert-match #显示不包含匹配文本的所有行。
-i --ignore-case #忽略字符大小写的差别。
-o # 只显示匹配到的关键字
-n # 现实行号

-E 使用正则表达式

```



初识正则表达式

^ : 匹配开头

\$: 匹配结尾

[] : 范围匹配

[a-z] : 匹配有小写字母

[A-Z] : 匹配所有大写字母

[0-9] : 匹配所有数字

. : 匹配单个字符

* : 表示*前面的内容出现0次或多次

+ : 表示+前面的内容出现1次或多次

? : 表示? 前面的内容出现0次或1次

```
cat a.txt | grep hat$ # 匹配以hat结尾的行
cat a.txt | grep ^hat # 匹配以hat开头的行
cat a.txt | grep -E "[0-9]*" # 匹配有0到多个数字的行
cat a.txt | grep -E "[0-9]+" # 匹配有至少有1个数字的行
cat a.txt | grep -E "[0-9]? " # 匹配有0到1个数字的行
```

sed : 流编辑器，一次处理一行内容



```
sed [-nefr] [动作] [文件]
```

选项与参数:

-n : 使用安静(silent)模式。在一般 sed 的用法中，所有来自 STDIN 的数据一般都会被列出到终端上。但如果加上 -n 参数后，
-e : 直接在命令列模式上进行 sed 的动作编辑
-f : 直接将 sed 的动作写在一个文件内，-f filename 则可以运行 filename 内的 sed 动作
-r : sed 的动作支持的是延伸型正规表示法的语法。(默认是基础正规表示法语法)
-i : 直接修改读取的文件内容，而不是输出到终端。

动作说明: [n1[,n2]] 动作:

n1, n2 : 不一定存在，一般代表选择进行动作的行数，比如，如果我的动作是需要要在 10 到 20 行之间进行的，则10,20[动作行为

动作:

#a : 新增，a 的后面可以接字符串，而这些字符串会在新的一行出现(目前的下一行)

#c : 取代，c 的后面可以接字符串，这些字符串可以取代 n1,n2 之间的行!

#d : 删除，因为是删除啊，所以 d 后面通常不接任何咚咚;

```
sed "3d" file # 删除第三行
sed "1,3d" # 删除前三行
sed "1d;3d;5d" # 删除1、3、5行
sed "/^$/d" #删除空行
sed "/abc/d" #删除所有含有abc的行
sed "/abc/, /def/d" #删除abc 和 def 之间的行，包括其自身
sed "1, /def/d" #删除第一行到 def 之间的行，包括其自身
sed "/abc/, +3d " # 删除含有abc的行之后，在删除3行
sed "/abc/, ~3d" #从含有abc的行开始，共删除3行
sed "1~2d" # 从第1行开始，每2行删除一行， 删除奇数行
sed "2~2d" # 从第2行开始，每2行删除一行， 删除奇数行
sed "$d" # 删除最后一行
sed "/dd\|cc/d" 删除有dd或者cc的行
```

#i : 插入，i 的后面可以接字符串，而这些字符串会在新的一行出现(目前的上一行);

#p : 列印，亦即将某个选择的数据印出。通常 p 会与参数 sed -n 一起运行

```
sed -n "3p" file # 显示第三行
sed -n "1,3p" # 显示前三行
sed -n "2,+3p" # 显示第二行，及后面的三行
sed -n "$p" # 显示最后一行
sed -n "1p;3p;5p" # 只显示文件1、3、5行
sed -n "$=" # 显示文件行数
```

#s : 替换，可以直接进行取代的工作。通常这个 s 的动作可以搭配正规表示法，例如 1,20s/old/new/g
's/old/new/g'

```
sed "s/(all)/bb/"
sed -r "s/(all)/bb/"
```



awk：一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。



```
# 命令行调用方式
awk [-F field-separator] 'commands' input-file(s)

# commands 是真正awk命令，[-F域分隔符]是可选的。 input-file(s) 是待处理的文件。 在awk中，文件的每一行中，由

# awk工作流程：
# 读入有'\n'换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0则表示所有域，$1表示第一个域，$n表示第n个域

cat /etc/passwd |awk -F ':' '{print $1}'
cat /etc/passwd |awk -F ':' '{print $1"\t"$7}'

awk 常用内置变量
ARGC 命令行参数个数
ARGV 命令行参数排列， ARGV[0] ARGV[1]
ENVIRON 支持队列中系统环境变量的使用
FILENAME awk浏览的文件名
FNR 浏览文件的记录数
FS 设置输入域分隔符，等价于命令行 -F选项
NF 浏览记录的域的个数
NR 已读的记录数
OFS 输出域分隔符
ORS 输出记录分隔符
RS 控制记录分隔符

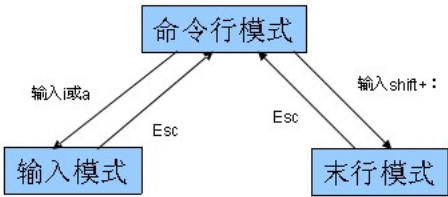
# 统计/etc/passwd:文件名，每行的行号，每行的列数，对应的完整行内容：
#awk -F ':' '{print "filename:" FILENAME ",linenumber:" NR ",columns:" NF ",linecontent:"$0}' /etc/passwd

# 使用printf替代print,可以让代码更加简洁，易读
awk -F ':' '{printf("filename:%10s,linenumber:%s,columns:%s,linecontent:%s\n",FILENAME,NR,NF,$0)}
```





vi/vim：强大的编辑器



Vim 命令小抄

original card by Laurent Gregoires
redesign by brohan

基本移动

| | |
|----------------|----------------------------|
| h l j k | 左/右 移动一个字符; 上/下 移动一行 |
| b w | 向 左/右 移动一个单词 |
| ge e | 向 左/右 移动一个单词 (光标在词尾) |
| { } | 移动到 前一个/后一个 段落开头 |
| O gm | 移动到 行首/行正中 |
| 0 S | 移动到行 第一个非空白字符/行尾 |
| nG ngg | 移动到第 n 行, 默认到 尾行/首行 |
| n% | 移动到文件的 n 百分比处 |
| n | 移动到当前行的第 n 列 |
| nH nL | 移动到窗口 顶端/底部 第 n 行 |
| % | 匹配下一个项目 (括号、方括号等.....) |
| M | 移动到窗口的中间行 |

编辑修改

| | |
|--------------------|--------------------------|
| i a | 在光标 前/后 插入文本 |
| I A | 在 行首/行尾 插入文本 |
| gl | 在第一列插入文本 |
| o O | 在光标的 下方/上方 插入新行 |
| ro | 把光标下的字符换成 c |
| gr | 类似 r , 但替换的是虚拟字符 |
| gR | 进入替换模式 |
| gR | 类似 r , 但替换的是虚拟字符 |
| cm | 更改动作 m 跨越的文本 |
| CC or S | 更改当前行的内容 |
| c | 更改光标到行尾的文本 |
| S | 更改一个字符并进入插入模式 |
| ~ | 切换当前字符大小写, 同时光标前移 |
| ~m | 切换动作 m 跨越文本的大小写 |
| gum gUm | 小写/大写 动作 m 跨越的文本 |
| <m >m | 左移/右移 动作 m 跨越的行 |
| n< n> | 将 n 行 左移/右移 一个缩进位 |

删除

| | |
|-------------|---------------------------------|
| x X | 删除光标 后/前 的字符 |
| dm | 删除动作 m 跨越的文本 |
| dd D | 删除 当前行/到行尾 |
| J gJ | 连接两行 删除缩进、插入空格/直接连接 |
| rd | 删除范围 r 包括的行 |
| rdx | 删除范围 r 包括的行到寄存器 x |

插入模式

| | |
|-----------------------|----------------------------------|
| ^Vc ^Vn | 插入字符 c 的 本义/十进制值 n |
| ^A | 插入最近插入的文本 |
| ^@ | 插入最近插入的文本, 并退出插入模式 |
| ^R x | 原样插入寄存器 x 的内容 |
| ^N ^P | 补全关键字并作 正向/反向 查找 |
| ^W | 删除光标前的单词 |
| ^U | 删除当前行光标前所有输入的字符 |
| ^D ^T | 左移/右移 一个缩进位 |
| ^K C1C2 | 输入 \C1,C2 代表的 二合字母 |
| ^O c | 在临时命令模式执行命令 c |
| ^X^E | 向上滚动一行 |
| ^X^Y | 向下滚动一行 |
| <esc> ^[| 结束插入模式, 回到命令模式 |

复制

| | |
|----------------|----------------------------|
| "x | 指定下次删除、抽出和放置使用寄存器 x |
| reg | 显示所有寄存器的内容 |
| reg x | 显示寄存器 x 的内容 |
| ym | 抽出动作 m 跨越的文本 |
| yy or Y | 抽出整 行 |
| p P | 放置寄存器内容到光标 之后/之前 |
| lp [P | 类似 "p" "P", 但调整当前行的缩进 |
| gp GP | 类似 "p" "P", 光标停留在新文本之后 |

复杂改动

| | |
|----------------|-------------------------------------|
| g?m | 对动作 m 跨越的文本做 rot13 编码 |
| n^A n^X | 光标之上或之后的数值或者字母 +n/-n |
| gqm | 格式化动作 m 跨越的所有行 |
| rce w | 将范围 r 中的行依据宽度 w 居中对齐 |
| rri i | 将范围 r 中的行靠左对齐, 缩进 i 列 |
| rri w | 将范围 r 中的行依据宽度 w 靠右对齐 |
| lmc | 将动作 m 跨越的行用命令 c 过滤 |
| nlc | 将 n 行用命令 c 过滤 |
| rlc | 将范围 r 中的行用命令 c 过滤 |

可视模式

| | |
|-----------------|-------------------------|
| v V | 以 字符/行 方式开始 (结束) 高亮 |
| ^V | 以 列表方式开始 (结束) 高亮 |
| o | 交换高亮区域的开始处和光标位置 |
| gv | 使用上一次的可视区域开始高亮 |
| aw as ap | 选择 "一个单词"/"一个句子"/"一个段落" |
| ab aB | 选择 "一个块"/"一个大块" () |

撤消, 重做

| | |
|-------------|---------------------|
| u U | 撤销最近的改动 / 恢复最近被改动的行 |
| . ^R | 重复最近的改动 / 重做最近撤销的改动 |
| n. | 重复最近的改动 n 次 |

寄存器

| | |
|----------------|--|
| q c qC | 记录键入的字符, 存入/添加到 寄存器 c |
| q | 停止记录 |
| @c | 执行寄存器 c 的内容 |
| @@ | 重复上次的 @c 操作 |
| @c | 将寄存器 c 的内容当作 Ex 命令来执行 |
| r/g/p/c | 在范围 r 中找到匹配内容 p 时执行 Ex 命令 c |

复合移动

| | |
|-----------------|---|
| - + | 上移/下移 一行, 至第一个非空白字符 |
| B W | 向 前/后 移动一个单词 |
| gE E | 向 前/后 移动一个单词 (光标在末尾) |
| n. | 下移 n-1 行, 至第一个非空白字符 |
| g0 | 移动到屏幕行 第一个非空白字符 |
| g^ g\$ | 移动到屏幕行 第一个非空白字符/最后一个字符 |
| gk gj | 上移/下移 一个屏幕行 |
| tc Tc | 移动到 后一个/前一个 字符 c 处 |
| to To | 移动到 后一个/前一个 字符 c 之前 |
| f , | 正向/反向 重复上次 " f ", " F ", " t " 或 " T " 命令 |
| [] | 向后/向前 一小节, 置于小节的开始 |
| [{ }] | 向后/向前 一小节, 置于小节的末尾 |
| [(]] | 向后/向前 至未闭合的 '[' / ']' |
| [{ }{ }] | 向后/向前 至未闭合的 '[' / ']' |
| [m]m | 向后/向前 至 java method 的开始 |
| [#]# | 向后/向前 至未闭合的 #if , #else , #endif |
| [']' | 向后/向前 至注释的 开始/结束 /" " / |

查找替换

| | |
|------------------|---|
| /s ?s | 向前/向后 查找 s |
| /s/o ?s/o | 向前/向后 查找 s , 光标偏移量 o |
| n or ? | 向前重复上次查找 |
| N or ? | 向后重复上次查找 |
| # * | 向后/向前 查找光标下的标识符 |
| g# g* | 同 " # " / " * ", 但也查找部分匹配 |
| gd gD | 跳转到光标下标识符的 局部/全局 声明 |
| r/s/f/t/x | 将范围 r 中的 s 替换成 t |
| | 选项 u : c 所有匹配项, c 确认替换 |
| r/s x | 在新的范围 r 中以选项 x 重复上一替换 |

进入vi的命令

vi filename :打开或新建文件, 并将光标置于第一行首

vi +n filename : 打开文件, 并将光标置于第n行首

vi + filename : 打开文件, 并将光标置于最后一行首

vi +/pattern filename: 打开文件, 并将光标置于第一个与pattern匹配的串处

vi -r filename : 在上次正用vi编辑时发生系统崩溃, 恢复filename

vi filename....filename : 打开多个文件, 依次进行编辑

移动光标类命令

h : 光标左移一个字符

l : 光标右移一个字符

space: 光标右移一个字符

Backspace: 光标左移一个字符

k或Ctrl+p: 光标上移一行

j或Ctrl+n : 光标下移一行

Enter : 光标下移一行

w或W : 光标右移一个字至字首

b或B : 光标左移一个字至字首

e或E : 光标右移一个字至字尾

) : 光标移至句尾

(: 光标移至句首

} : 光标移至段落开头

{ : 光标移至段落结尾

nG: 光标移至第n行首

n+: 光标下移n行

n-: 光标上移n行

n

: 光标移至第n行尾 **H** : 光标移至屏幕顶行 **M** : 光标移至屏幕中间行 **L** : 光标移至屏幕最后一行 **0** : (注意是数字零) 光标移至当前行首

: 光标移至当前行尾

屏幕翻滚类命令

Ctrl+u: 向文件首翻半屏

Ctrl+d: 向文件尾翻半屏

Ctrl+f: 向文件尾翻一屏

Ctrl+b; 向文件首翻一屏

nz: 将第n行滚至屏幕顶部, 不指定n时将当前行滚至屏幕顶部。

插入文本类命令

i : 在光标前

I : 在当前行首
a: 光标后
A: 在当前行尾
o: 在当前行之下新开一行
O: 在当前行之上新开一行
r: 替换当前字符
R: 替换当前字符及其后的字符, 直至按ESC键
s: 从当前光标位置处开始, 以输入的文本替代指定数目的字符
S: 删除指定数目的行, 并以所输入文本代替之
ncw或nCw: 修改指定数目的字
nCC: 修改指定数目的行

删除命令

ndw或ndW: 删除光标处开始及其后的n-1个字
do: 删至行首
d
: 删至行尾 *ndd*: 删除当前行及其后 $n - 1$ 行 *x* 或 *X*: 删除一个字符, *x* 删除光标后的, 而 *X* 删除光标前的 *Ctrl + u*: 删除输入方式
搜索 *pattern?pattern*: 从光标开始处向文件首搜索 *patternn*: 在同一方向重复上一次搜索命令 *N*: 在反方向上重复上一次搜索
 $n2s/p1/p2/g$: 将第 $n1$ 至 $n2$ 行中所有 $p1$ 均用 $p2$ 替代: $g/p1/s//p2/g$: 将文件中所有 $p1$ 均用 $p2$ 替换选项设置 *all*: 列出所有选
在搜索中忽略大小写 *list*: 显示制表位 (*Ctrl + I*) 和行尾标志 (
)
number: 显示行号
report: 显示由面向行的命令修改过的数目
terse: 显示简短的警告信息
warn: 在转到别的文件时若没保存当前文件则显示NO write信息
nomagic: 允许在搜索模式中, 使用前面不带“\”的特殊字符
nowrapscan: 禁止vi在搜索到达文件两端时, 又从另一端开始
mesg: 允许vi显示其他用户用write写到自己终端上的信息

最后行方式命令

: n1,n2 co n3: 将n1行到n2行之间的内容拷贝到第n3行下
: n1,n2 m n3: 将n1行到n2行之间的内容移至到第n3行下
: n1,n2 d : 将n1行到n2行之间的内容删除
: w : 保存当前文件
: e filename: 打开文件filename进行编辑
: x: 保存当前文件并退出
: q: 退出vi
: q!: 不保存文件并退出vi
: !command: 执行shell命令command
: n1,n2 w!command: 将文件中n1行至n2行的内容作为command的输入并执行之, 若不指定n1, n2, 则表示将整个文件内容作为command的输入
: r!command: 将命令command的输出结果放到当前行

寄存器操作

"?nyy: 将当前行及其下n行的内容保存到寄存器? 中, 其中?为一个字母, n为一个数字
"?nyw: 将当前行及其下n个字保存到寄存器? 中, 其中?为一个字母, n为一个数字
"?nyl: 将当前行及其下n个字符保存到寄存器? 中, 其中?为一个字母, n为一个数字
"?p: 取出寄存器? 中的内容并将其放到光标位置处。这里? 可以是一个字母, 也可以是一个数字
ndd: 将当前行及其下共n行文本删除, 并将所删内容放到1号删除寄存器中。

一、插入文本

i 在当前字符前插入文本
I 在行首插入文本
a 在当前字符后添加文本
A 在行末添加文本
o 在当前行后面插入一空行
O 在当前行前面插入一空行
R 以改写方式输入文本

二、移动光标

j或下箭头 向下移动一行
 k或上箭头 向上移动一行
 h或左箭头 左移一个字符
 l或右箭头 右移一个字符
 w 右移一个词
 W 右移一个以空格分隔的词
 b 左移一个词
 B 左移一个以空格分隔的词
 O 移到行首
 Ctrl-F 向前翻页
 Ctrl-B 向后翻页
 nG 到第n行
 G 到最后一行

三、替换文本

\$ 到行尾
 (到句子的开头
) 到句子的末尾
 { 到段落的开头
 } 到段落的末尾

四、删除文本

r 替换一个字符
 c 修改文本直到按下Esc键
 cw 修改下一个词
 cnw 修改接下来的n个词

五、文本编辑

yy 将一行文本移到缺省缓冲区中
 yn 将下一个词移到缺省缓冲区中
 ynw 将后面的n个词移到缺省缓冲区中
 p 如果缺省缓冲区中包含一行文本，则在当前行后面插入一个空行并将缺省缓冲区中的内容粘贴到这一行中；如果缺省缓冲区中包含多个词，把这些词粘贴到光标的右边。
 P 如果缺省缓冲区中包含一行文本，则正当前行前面插入一个空行并将缺省缓冲区中的内容粘贴到这一行中；如果缺省缓冲区中包含多个词，把这些词粘贴到光标的左边

六、保存退出

zz 保存并退出
 :w filename 写入文件
 :W 写入文件
 :x 保存(如果当前文件修改过)并退出
 :q! 不保存文件，直接退出
 :q 退出vi

vi编辑器的启动与退出

直接进入编辑环境

\$ vi

进入编辑环境并打开（新建）文件

\$ vi myfile

退出vi编辑环境

输入末行命令放弃对文件的修改，并退出编辑器

:q!

保存文件

保存对vi编辑器中已打开文件的修改

:w

另存为文件

将vi编辑器中的内容另存为指定文件名

:w myfile

退出vi编辑器的多种方法

未修改退出

没有对vi编辑器中打开的文件进行修改，或已对修改进行了保存，直接退出vi编辑器

:q

对vi编辑器中的文件进行保存并退出vi编辑器

:wq

不保存退出

放弃对文件内容的修改，并退出vi编辑器

:q!

光标的移动和翻页操作

h向左移动光标

l向右移动光标

k向上移动光标

j向下移动光标

翻页Ctrl + f向前翻整页

Ctrl + b向后翻整页

Ctrl + u向前翻半页

Ctrl + d向后翻半页

行内快速跳转

^将光标快速跳转到本行的行首字符

\$将光标快速跳转到本行的行尾字符

w将光标快速跳转到当前光标所在位置的后一个单词的首字母

b将光标快速跳转到当前光标所在位置的前一个单词的首字母

e将光标快速跳转到当前光标所在位置的后一个单词的尾字母

文件内行间快速跳转

命令功能

:set nu 在编辑器中显示行号

:set nonu 取消编辑器中的行号显示

1G跳转到文件的首行

G跳转到文件的末尾行

#G跳转到文件中的第#行

进入输入模式

i在当前光标处进入插入状态

a在当前光标后进入插入状态

A将光标移动到当前行的行末，并进入插入状态

o在当前行的下面插入新行，光标移动到新行的行首，进入插入状态

O在当前行的上面插入新行，光标移动到新行的行首，进入插入状态

cw删除当前光标到所在单词尾部的字符，并进入插入状态

c\$删除当前光标到行尾的字符，并进入插入状态

c^命令删除当前光标之前（不包括光标上的字符）到行首的字符，并进入插入状态

输入模式的编辑键操作

方向键进行上下左右方向的光标移动

Home快速定位光标到行首

End快速定位光标到行尾

PageUp进行文本的向上翻页

PageDown进行文本的向下翻页

Backspace删除光标左侧的字符

Del删除光标位置的字符

删除操作

x删除光标处的单个字符

dd删除光标所在行

dw删除当前字符到单词尾（包括空格）的所有字符

de删除当前字符到单词尾（不包括单词尾部的空格）的所有字符

d\$删除当前字符到行尾的所有字符

d^删除当前字符到行首的所有字符

J删除光标所在行行尾的换行符，相当于合并当前行和下一行的内容

替换操作

:s/old/new 将当前行中查找到的第一个字符“old” 串替换为“new”

:s/old/new/g 将当前行中查找到的所有字符串“old” 替换为“new”

:#,#s/old/new/g 在行号“#,#”范围内替换所有的字符串“old”为“new”

:%s/old/new/g 在整个文件范围内替换所有的字符串“old”为“new”

:s/old/new/c 在替换命令末尾加入c命令，将对每个替换动作提示用户进行确认

撤消操作

u取消最近一次的操作，并恢复操作结果

可以多次使用u命令恢复已进行的多步操作

U取消对当前行进行的所有操作

Ctrl + r对使用u命令撤销的操作进行恢复

复制与粘贴操作

yy复制当前行整行的内容到vi缓冲区

yw复制当前光标到单词尾字符的内容到vi缓冲区

y\$复制当前光标到行尾的内容到vi缓冲区

y^复制当前光标到行首的内容到vi缓冲区

p读取vi缓冲区中的内容，并粘贴到光标当前的位置（不覆盖文件已有的内容）

字符串查找操作

/word从上而下在文件中查找字符串“word”

?word 从下而上在文件中查找字符串“word”

n定位下一个匹配的被查找字符串

N定位上一个匹配的被查找字符串

快捷键

ctrl-a : 把光标移动到命令行最开始的地方。
ctrl-e : 把光标移动到命令行末尾。
ctrl-u : 清除命令行中光标所处位置之前的所有字符。
ctrl-k : 清除从提示符所在位置到行末尾之间的字符
ctrl-w : 清除左边的字段
ctrl-y : 将会贴上被ctrl-u 或者 ctrl-k 或者 ctrl-w清除的部分。
ctrl-r : 将自动在命令历史缓存中增量搜索后面入的字符。
tab : 命令行自动补全 - 自动补全当前的命令行。如果启用自动补全脚本命令参数和选项也可以自动补齐。

ctrl-l : 清屏

分类: [Python](#)

好文要顶

已关注

收藏该文

R_e

关注 - 1

粉丝 - 770

310

我在关注他

取消关注

« 上一篇: [开始使用Python](#)

» 下一篇: [学习方法--提问](#)

posted on 2016-08-29 08:51 R_e 阅读(32361) 评论(5) 编辑 收藏

评论

#1楼 回复 引用

bucuo

支持(1) 反对(0)

2017-08-16 16:06 | 唐木槿

#2楼 回复 引用

老哥厉害

支持(1) 反对(0)

2017-12-09 14:26 | 西园公子zwj

#3楼 回复 引用

牛逼

支持(0) 反对(0)

2018-02-01 17:54 | yyystation

#4楼 回复 引用

666

支持(0) 反对(0)

2018-05-21 20:46 | Monomania\ pp

#5楼 回复 引用

laoge wen

支持(0) 反对(0)

2018-07-04 08:27 | pp凉

发表评论

编辑

预览

B

支持 Markdown

提交评论

退出

订阅评论

[Ctrl+Enter快捷键提交]

- 【推荐】了不起的开发者，势不可挡的华为，园子里的品牌专区
- 【推荐】有道智云周年庆，API服务大放送，注册即送100元体验金！
- 【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】精品问答：Java 技术 1000 问