工作中你应该掌握的 linux 命令大全

news 2023/5/21 1:32:17

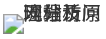
作为码农，工作中你一定离不开使用 linux 命令，linux 命令繁多，学习起来确实很累，所以在工作的两年多的时间里，我把自己平时遇到的一些命令都零零碎碎的记录下来，正好这几天有时间对原先的记录做一个整体总结，并对这些基础的 linux 命令分了几个大类整理如下。

希望这篇文章可以对大家有所帮助，如果哪里有问题欢迎指出，文章会持续更新！！！！

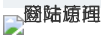
- 用户登录相关命令:
  - sudo命令
  - su 命令
  - w命令
  - who 命令
  - whoami命令
  - who am i命令
  - last命令
  - tty命令
  - passwd命令
  - write命令
  - mesg命令
  - wall命令
- 查看系统运行状态和进程相关命令:
  - 查看cpu内核数目
  - uname指令
  - w 命令
  - uptime命令
  - pidof
  - systemctl
  - top命令
  - ps命令
  - &
  - ctrl + z
  - jobs
  - fg命令
  - bg命令
  - kill命令
  - nohup命令
  - pstree命令
  - killall命令
  - nice命令/renice命令
  - free命令
  - ulimit命令
- 网络相关命令：
  - ping 命令
  - curl命令
  - wget命令
  - telnet命令
  - netstat命令
  - traceroute命令
  - host命令
  - nslookup命令
  - dig命令
  - ssh命令
- 命令流操作相关：
  - 重定向操作(>/>>/<</<)
  - 管道操作(|)
  - xargs命令
  - tee命令
- 软连接和硬连接：
  - ln 命令
- 命令文档相关：
  - man 查看命令的详细描述

最新文章

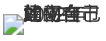

爬虫基础：HTTP基本原理
2023/5/21 1:32:17


网站访问流程及原理分析
2023/5/21 1:32:16

暂无图片
网站运行原理及开发流程
2023/5/21 1:32:14


网站访问登陆原理
2023/5/21 1:32:07


【VR】详解 Facebook Oculus团队的手势追踪...
2023/5/21 1:31:54


如何在2022年建立自己的NFT市场平台
2023/5/21 1:31:53


技术人员如何创业：打造超强执行力团队
2023/5/21 1:31:52


第六届团队程序设计天梯赛全题目解析讲解
2023/5/21 1:31:48

- which 显示命令的可执行文件的位置
- type 查看命令的类型
- whatis 显示非常简洁的命令说明
- alias 给命令起别名
- unalias 删除别名命令
- apropos命令
- 系统命令可用性检查

- 文件/文件夹/路径相关操作指令：
- pwd 打印当前目录路径
- ls 列出当前路径下的文件和文件夹
- cd 切换目录
- file 查看文件或者文件夹的类型
- cat 将文档在控制台输出 ， -n输出行号，-s禁止输出空白行, -A将空白字符输出
- less 打开文档，可以上下页切换
- stat 命令
- rm 删除文件或者文件夹
- cp 文件及文件夹的复制
- mkdir 创建文件目录
- touch 创建文档，如果文档存在，可以修改文档的更新，修改时间
- mv 剪切文件并复制到其它位置
- basename命令
- dirname命令
- 几个查找文件的命令

- shell 环境相关配置和使用：
- shell配置简介
- bash的环境配置文件详解
- shell状态查看和切换
- history命令
- clear 命令
- !number 命令
- printenv命令
- set命令
- env命令
- export命令
- source命令
- command命令
- linux中常用的环境变量
- shell常用快捷键
- sh命令

- 文本操作相关命令：
- 几个文件查阅的命令
- cut命令
- fmt命令
- echo命令
- sort命令
- uniq 命令
- grep 命令
- wc命令
- head命令
- tail命令
- od命令
- xxd命令
- awk命令
- sed命令
- read命令
- tr命令
- col命令
- join命令
- paste命令
- split命令
- diff命令
- cmp命令
- patch命令
- pr命令
- test命令
- seq命令

- 用户/角色/权限相关命令：
- id 命令

- groups 命令
- usermod 命令
- gpasswd 命令
- newgrp 命令
- useradd 命令
- passwd 命令
- chage 命令
- userdel 命令
- chsh 命令
- groupadd 命令
- groupmod 命令
- groupdel 命令
- ls -l filename
- chmod
- chown 更改文件的所有者和所有组
- chgrp 改变文件所属用户组
- umask命令
- lsattr和chattr命令

- 打包和解压相关命令：
- tar 命令
- gzip命令
- zcat／zmore／zless命令
- bzip2命令
- bzcat/bzmore/bzless命令
- xz命令
- xzcat/xzless/xzmore命令

- 常用工具命令：
- cal命令
- date命令
- md5命令和md5sum命令
- bc命令
- mail命令
- 开关机命令
- sync命令

- shell脚本相关知识：
- shell中特殊字符的含义
- declare命令
- 双引号和单引号
- 变量部分替换命令
- shebang in linux(!#)
- 命令的顺序执行
- printf命令
- "$"开头的含义
- linux中的数组
- for循环

- 磁盘和文件系统相关命令：
- df命令
- du命令
- fuser命令
- lsof命令
- 磁盘分区相关命令
- xfsdump命令用于备份文件系统
- xfsrestore命令用于还原文件系统

- 定时任务相关命令：
- at命令
- batch命令
- crontab命令
- anacron命令

- 文件的编译和运行:
- gcc命令
- make命令

用户登录相关命令:

sudo命令

以另一个用户身份执行shell，权限下放，用户自己的密码就可以

1. sudo -l 可以查看当前用户可以以sudo身份执行的命令
2. sudo -v 使sudo密码时限延长
3. sudo -u 以指定的用户作为新的身份,如果没有指定，则以root账号
4. sudo -i 以指定的用户登录

5. sudo 命令的配置文件在/etc/sudoers中配置

```
# 在 /etc/sudoers 中定义别名
Host_Alias hostalias = hostname1, hostname2, ...
User_Alias useralias = user1, user2, ...
Cmnd_Alias cmndalias = command1, command2, ...# user  host = commands 让某些用户或者用户
Cmnd_Alias YOUDATA = /etc/init.d/, /usr/bin/apt-get, /sbin/iptables, /bin/ipset
youdata ALL = NOPASSWD:YOUDATA# 容许youdata_group组在本机登录时拥有/usr/bin/emerge权限
%youdata_group  localhost = /usr/bin/emerge# 非root用户身份切换: users  hosts = (run-as
Cmnd_Alias KILL = /bin/kill, /usr/bin/pkill
# 容许其他用户像apache，gorg用户一样执行/bin/kill, /usr/bin/pkill命令
swift   ALL = (apache, gorg) KILL
```

4. 在/etc/sudoers.d/下配置自己的命令权限
   系统文档推荐的做法，不是直接修改/etc/sudoers文件，而是将修改写在/etc/sudoers.d/目录下的文件中，任何在/etc/sudoers.d/目录下，不以~号结尾的文件和不包含.号的文件，都会被解析成/etc/sudoers的内容。如果使用这种方式修改sudoers，需要在/etc/sudoers文件的最后行，加上#includedir /etc/sudoers.d语句(默认已有):

```
#includedir /etc/sudoers.d
```

   注意了，这里的指令#includedir是一个整体, 前面的#号不能丢，也不能在#号后有空格
   su 命令

- su命令的主要作用是让你可以在已登录的会话中切换到另外一个用户
- 一般 su 命令需要输入目标用户的密码
- “su -” 切换用户以后保持切换后用户的环境，而"su" 切换用户以后保持原有用户的环境
- “su -c command” 切换用户后指定执行的命令,但是执行完以后，切换回原先账号，不需要exit

   w命令
   用于显示目前登入系统的用户信息，它不但可以显示有谁登录到系统，还可以显示出这些用户当前正在进行的工作，并且统计数据相对who命令来说更加详细和科学

```
$ w
=> 第一行显示的内容以此是：当前时间，系统启动到现在的时间，登录时间，登录用户的数目，系
=> 从第二行开始显示的是每个用户的各项数据
12:00  up 4 days,  2:58, 2 users, load averages: 1.57 1.57 1.53
USER          TTY        FROM             LOGIN@  IDLE WHAT
zhangdianpeng console    -                三09    4days -
zhangdianpeng s001       -                11:54       - w
```

   who 命令
   该命令主要用于查看当前在线上的用户情况

```
$ who
zhangdianpeng console  Nov  8 09:02
zhangdianpeng ttys001  Nov 12 11:54$ who -m
=> 自己在系统中的用户名，登录终端，登录时间
zhangdianpeng ttys001  Nov 12 11:54$ who -q
=> 显示登录账号和登录用户的数量
zhangdianpeng zhangdianpeng
# users = 2
```

   whoami命令
   只显示出自己在系统中的用户名
   who am i命令
   等同于 who -m
   last命令
   显示所有用户的历史登录时间
   tty命令
   可以使用tty来报告当前登录用户所连接的设备或终端
   passwd命令
   修改用户密码，passwd作为普通用户和超级权限用户都可以运行，但作为普通用户只能更改自己的用户密码

```
password       # 修改自己的密码
password user  # 修改其它用户的密码
```

```
$ tty
/dev/ttys001
```

write命令

write命令可以直接发信息给正在在线的其他用户

```
write hzzhangdianpeng pst/7    # 给用户hzzhangdianpeng的pst/7的终端发信息
```

mesg命令

```
mesg n          # 可以拒绝其他用户的write信息
mesg y          # 开放接受所有用户的write信息
```

wall命令

可以对所有在线用户下达命令

查看系统运行状态和进程相关命令:

查看cpu内核数目

```
//mac系统
sysctl -a | grep core_count
=> machdep.cpu.core_count: 2//linux系统
grep -c ^processor /proc/cpuinfo
=> 4lscpu
=> CPU(s):   2//Solaris系统
/usr/sbin/psrinfo -vp
=> 4
```

uname指令

uname命令用于打印当前系统相关信息（内核版本号、硬件架构、主机名称和操作系统类型等）

- uname -a 显示操作系统的全部信息
- uname -m 查看操作系统的位版本
- uname -r 查看操作系统的核心版本
- uname -n 显示主机名称
- uname -v 显示操作系统版本

w 命令

w命令（上文已有详解）

uptime命令

返回系统的运行状态：当前时间，系统启动到现在的时间，登录时间，登录用户的数目，系统在最近1秒、5秒和15秒的平均负载

等同于w命令返回的第一行

pidof

列出某个正在执行的程序的PID

```
pidof systemd rsyslogd
```

systemctl

systemctl用于管理单一的服务的开机启动，停止，状态能信息

```
systemctl  start               # 启动
systemctl  stop                # 停止
systemctl  restart             # 重启
systemctl  reload              # 重新载入配置文件
systemctl  enable              # 下次开机自动启动
systemctl  disable             # 下次开机不启动
sysetmctl  status              # 查看状态（开机是否立即启动，有没有正在运行等状态）
systemctl  is-active           # 目前有没有正在运行
systemctl  is-enable           # 开机是否立即运行
systemctl  mask                # 注销服务
systemctl  unmask              # 取消注销服务
systemctl  list-units          # 列出目前启动的所有服务，加参数--all，包括没有启动的服
systemctl  list-dependencies   # 列出某个服务依赖其他服务的情况
systemctl  list-dependencies --reverse   # 列出依赖于该服务的其他服务
```

top命令

返回结果解释如下：

- 第一行进程和线程运行情况：共有多少个进程，几个正在运行的进程等情况。一共有多少个线程
- 第二行cpu的使用情况

- 第三行内存试用情况
- 第四行swap交换分区信息
- 第五行网络使用情况
- 第六行是各进程的状态监控：PID， COMMAND，CPU，MEM，USER等

```
# 在top的执行过程中可以使用以下按键进行显示P 以CPU的使用资源排序
M 以Memory的使用资源排序
N 以PID来排序
T 以CPU时间累积排序# 输出结果：
top - 23:05:53 up 498 days,  7:43,  1 user,  load average: 0.01, 0.12, 0.10
Tasks: 148 total,   1 running, 145 sleeping,   0 stopped,   2 zombie
%Cpu(s):  0.3 us,  0.6 sy,  0.0 ni, 99.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  16471292 total, 11863976 used,  4607316 free,  1047120 buffers
KiB Swap:        0 total,        0 used,        0 free,  7640176 cachedPR表示优先级
30179 ds        20   0  563m 192m 7728 S   2.7  1.2  18120:15 java13 root      20   0
15980 root      20   0     0    0    0 S   0.7  0.0   1:23.34 kworker/0:218 root      2
```

ps命令
显示瞬间进程 (process) 的动态

```
$ ps -aUSER       PID    %CPU      %MEM      VSZ        RSS           TTY    STAT  STAR
# 用户       进程Id  CPU占用  内存在用   虚存的大小 实际内存大小  终端   状态  启动时间
#显示指定进程的信息$ ps -o pcpu,rssize,time,start
# 自定格式的输出进程相关信息$ ps -ef
# 显示当前系统中运行的所有进程，相当于ps -A$ ps -a
# 显示当前终端所有进程ps aux # 观看系统的所有程序数据ps -l   # 只查看自己相关bash的程序
# 而S字段表示程序状态，S=R表示正在运行中，S=S表示睡眠状态，可以被唤醒，S=D表示不可以被唤
```

&
执行命令后面跟上&可以将该程序放在后台执行，如果用户离线了，那么在后台执行的程序将会被暂停
ctrl + z
将当前程序丢到后台并暂停
jobs
查看后台运行的程序情况

```
jobs -l       # 列出job number及pid号码
jobs -r       # 列出正则run的工作
jobs -s       # 列出被stop掉的工作
```

fg命令
将后台的工作拿到前台来执行

```
fg %3     # 将job number = 3的job拿到前台来执行
```

bg命令
将后台stop状态的job变为run状态

```
bg %3     # 将job number = 3的job在后台变为run状态
```

kill命令
结束后台运行的程序，后接pid

```
kill -9     # 立刻强制删除一个程序
kill -15    # 以正常的程序方式终止一个程序
```

nohup命令

- nohup命令可以将程序以忽略挂起信号的方式运行起来，被运行的程序的输出信息将不会显示到终端，无论是否将 nohup 命令的输出重定向到终端，输出都将附加到当前目录的 nohup.out 文件中
- 该命令可以在你退出帐户/关闭终端之后继续运行相应的进程
- 和&不同，nohup在你离开系统后，仍然会执行

pstree命令
显示每个进程的树状结构
killall命令
可以删除某个服务，可以将系统以某个指令名称启动的程序全部删掉

```
killall -9 httpd     # 终止所有以httpd启动的服务
killall -i -9 bash   # -i参数询问每个bash是否要终止
```

nice命令/renice命令
调整程序的cpu调度的优先级命令
free命令
观察内存使用情况
ulimit命令
通过ulimit改善系统性能
linux 系统中的 ulimit 指令，对资源限制和系统性能优化提供了一条便捷的途径。从用户的shell启动脚本，应用程序启动脚本，以及直接在控制台，都可以通过该指令限制系统资源的使用，包括所创建的内核文件的大小、进程数据块的大小、Shell进程创建文件的大小、内存锁住的大小、常驻内存集的大小、打开文件描述符的数量、分配堆栈的最大大小、CPU时间单个用户的最大线程数、Shell进程所能使用的最大虚拟内存等等方面
ulimit 限制的是当前 shell 进程以及其派生的子进程

```
-a   显示目前全部限制情况
-c   容许 core dump文件的最大值，单位为区块
-d   <数据节区大小> 程序数据节区的最大值，单位为KB
-f   <文件大小> shell 所能建立的最大文件，单位为区块
-H   设定资源的硬性限制，也就是管理员所设下的限制
-m   <内存大小> 指定可使用内存的上限，单位为 KB
-n   <文件描述符数目> 指定同一时间最多可开启的 fd 数
-p   <缓冲区大小> 指定管道缓冲区的大小，单位512字节
-s   <堆叠大小> 指定堆叠的上限，单位为 KB
-S   设定资源的弹性限制
-t   指定CPU使用时间的上限，单位为秒
-u   <进程数目> 用户最多可开启的进程数目
-v   <虚拟内存大小> 指定可使用的虚拟内存上限，单位为 KB
```

网络相关命令：
ping 命令
ping命令用于测试到达目的主机的网络是否连接，可以接收域名或者ip，不能检查端口，工作在 OSI 参考模型的第三层-网络层

```
//直接ping ip
$ ping 10.165.124.134
PING 10.165.124.134 (10.165.124.134): 56 data bytes
64 bytes from 10.165.124.134: icmp_seq=0 ttl=63 time=3.923 ms
64 bytes from 10.165.124.134: icmp_seq=1 ttl=63 time=4.103 ms//ping域名，可以获取到指定
$ ping dev.youdata.com
PING dev.youdata.com (106.2.44.33): 56 data bytes
64 bytes from 106.2.44.33: icmp_seq=0 ttl=58 time=3.477 ms
64 bytes from 106.2.44.33: icmp_seq=1 ttl=58 time=3.904 ms
64 bytes from 106.2.44.33: icmp_seq=2 ttl=58 time=3.878 ms// -c参数指定发送包的个数
$ ping -c 2 dev.youdata.com// -W参数指定ping命令的等待时间
$ ping -W 2 dev.youdata.com
```

curl命令
curl是强大的URL传输工具，支持FILE, FTP, HTTP, HTTPS, IMAP, LDAP, POP3,RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET以及TFTP等协议。我们使用这个命令最常用的功能就是通过命令行发送HTTP请求以及下载文件，它几乎能够模拟所有浏览器的行为请求，比如模拟refer(从哪个页面跳转过来的）、cookie、agent（使用什么浏览器）等等，同时还能够模拟表单数据。

```
$ curl www.sina.com
=>
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://www.sina.com.cn/">here</a>.</p>
</body></html>$ curl -o sina.html www.sina.com
=> 用于将请求数据保存在文件里$ curl -I www.sina.com
=> -I 可以获取请求的头信息HTTP/1.1 200 OK
Server: nginx
Date: Thu, 16 Nov 2017 02:24:48 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 16325
Connection: keep-alive
```

```
Vary: Accept-Encoding
set-cookie: A_SESSION_KEY=s%3A3CbnLf8n-qCV-pdGayjxemXrZeIhzGkY.if4UQOlDOOP3vk7IRMqoNadz
Strict-Transport-Security: max-age=15768000$ curl -X POST --data "data=xxx" example.com
=> -X 参数指定请求方式$ curl --header "Content-Type:application/json" http://example.co
=> --header 参数用于添加请求头信息$ curl --cookie "name=xxx" www.example.com
=> 使用`--cookie`参数，可以让curl发送cookie$ curl -L -o redis1.msi https://github.com/M
=> -L参数支持跳转到新的网页，以上通过跳转实现下载redis安装包，和wget功能一样
```

wget命令

wget是一个强大的非交互网络下载工具，虽然curl也支持文件下载，不过wget更强大，比如支持断点下载等

```
$ wget https://github.com/MicrosoftArchive/redis/releases/download/win-3.2.100/Redis-x6
=> 自动下载并保存在当前路径下$ wget -i download.txt
=> 通过-i参数实现批量下载，将所有下载地址写在download.txt文件里$ wget -o Music/redis.ms
=> -o 参数可以指定下载路径及文件名$ wget -b https://www.baidu.com
=> 使用-b可以实现后台下载$ wget -o Music/baidu.txt -r -l 3 https://www.baidu.com
=> -r参数可以递归下载，-l参数可以指定递归下载的层级，可以实现爬虫$ wget -A.jpg -r -l 2
=> -A<后缀名>: 指定要下载文件的后缀名，多个后缀名之间使用逗号进行分隔; $ wget --limit-r
=> --limit-rate参数可以指定下载的最大速度$ wget -c http://www.linuxde.net/testfile.zip
=> 使用wget -c重新启动下载中断的文件,可以实现断点下载
```

telnet命令

- telnet命令用于登录远程主机，telnet因为采用明文传送报文，安全性不好，很多Linux服务器都不开放telnet服务，而改用更安全的ssh方式了,Telnet是位于OSI模型的第7层—应用层上的一种协议
- telnet ip port 用于测试远程主机的某个端口是否开放

netstat命令

netstat 命令用来打印 Linux 中网络系统的状态信息，可让你得知整个 Linux 系统的网络情况，这个命令暂且还不是特别懂，需要加强学习

traceroute命令

traceroute 命令用于追踪数据包在网络上的传输时的全部路径，它默认发送的数据包大小是40字节，记录按序列号从1开始，每个纪录就是一跳 ，每跳表示一个网关

host命令

host命令是常用的分析域名查询工具，可以用来测试域名系统工作是否正常

```
host www.linuxde.net                        # 查看域名对应的ip
=>
# www.linuxde.net is an alias for host.1.linuxde.net.
# host.1.linuxde.net has address 100.42.212.8host -a www.linuxde.net
```

nslookup命令

nslookup命令是常用域名查询工具，就是查DNS信息用的命令,有两种工作模式，即“交互模式”和“非交互模式”

```
nslookup www.linuxde.net
# Server:               192.168.0.1   dns服务器
# Address:      192.168.0.1#53# Non-authoritative answer:
# Name: www.linuxde.net
# Address: 47.104.99.244   ip地址
```

dig命令

dig命令是常用的域名查询工具，可以用来测试域名系统工作是否正常

```
dig  www.linuxde.net; <<>> DiG 9.8.3-P1 <<>> www.linuxde.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31330
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0;; QUESTION SECTION
;www.linuxde.net.               IN      A;; ANSWER SECTION:                # ip地址
www.linuxde.net.        310     IN      A       47.104.99.244;; AUTHORITY SECTION:
linuxde.net.            2716    IN      NS      f1g1ns2.dnspod.net.
linuxde.net.            2716    IN      NS      f1g1ns1.dnspod.net.;; Query time: 59 ms
;; SERVER: 192.168.0.1#53(192.168.0.1)      # 域名服务器
;; WHEN: Sat May 12 12:05:05 2018
;; MSG SIZE  rcvd: 100
```

ssh命令

- -N 不执行登录远程的命令
- -f 后台执行
- -R 远程端口转发
- -L 本地端口转发

```
# 将本机的3000端口转发到远程主机dev1的80端口
ssh -Nf -L 3000:localhost:80 dev1# 不登录远程主机，而在远程主机执行某个命令
ssh dev1 'pwd'# 将远程主机的8008端口转发到本机的7000端口
ssh -Nf -R 8008:127.0.0.1:7000 dev2
```

命令流操作相关：
重定向操作(>/>>/<</<)
重定向的操作时，0表示标准输入，1表示标准输出，2表示错误输出

1. 基本重定向命令

命令	介绍
command >filename	把标准输出重定向到新文件中
command 1>filename	同上
command >>filename	把标准输出追加到文件中
command 1>>filename	同上
command 2>filename	把标准错误重定向到新文件中
command 2>>filename	把标准错误追加到新文件中
command < filename	以filename文件作为标准输入
command 0< filename	同上
command << delimiter	从标准输入中读入，直到遇到delimiter分隔符

2. 重定向绑定

- command 2>&1：错误输出将和标准输出同用一个文件描述符
- command >/dev/null 2>&1：该条shell命令将不会输出任何信息到控制台，也不会有任何信息输出到文件中

管道操作(|)

- 它仅能处理经由前面一个指令传出的正确输出信息，也就是standard output的信息
- 对于 stdandard error 信息没有直接处理能力
- 将上一个命令的正确输出传递给下一个命令，作为标准的输入standard input

xargs命令
该指令用于产生某些指令的输入，主要是因为有些指令不支持管道命令，可以借助xargs命令来实现

```
# 使用方式
somecommand | xargs command
# 因为id指令不支持管道，以下是获取/etc/password中前三个用户名，然后每次一个参数一个参数
cut -d ':' -f 1 /etc/passwd | head -n 3 |  xargs -n 1 id
```

tee命令
tee命令可以把数据重定向到给定文件和屏幕上，就是说不仅可以放到文件里，同时还可以提供一份重定向数据的副本作为后续命令的stdin,存在缓存机制，每1024个字节将输出一次。若从管道接收输入数据，应该是缓冲区满，才将数据转存到指定的文件中。
如果你既想把输出放到文件，同时又想把输出通过管道方式给下一个命令执行，可以使用 tee 命令。

```
ls -l | tee -a test.js    # 将ls -l命令的结果放到文件test.js中，并打印到屏幕上
```

软连接和硬连接：
ln 命令
硬连接和软连接的区别：

- 硬链接原文件和链接文件共用同一个inode，只是文件名存在不同的block而已
- 软链接的源文件和链接文件各自有自己的inode，通过文件名进行寻找源文件
- 硬连接不能跨文件系统，也不能link目录
- 软链接和硬链接的文件内容都不占用实际空间，文件都保持同步变化；
- 当删除源文件时，因为软链接可以链接不存在的文件，所以这个时候只是找不到文件，而硬连接中的文件仍然存在，并保存了源文件中内容；
- 硬链接的作用是防止用户误操作删除，如果删除了源文件，链接文件将独立成一份文件，实现备份；

```
ln -s A B   # 创建软链接
ln A B      # 创建硬链接
-d          # 允许超级用户制作目录的硬链接
-b          # 覆盖以前建立的链接
```

命令文档相关：
man 查看命令的详细描述

```
man ls  => 查看ls指令的详细使用方式man -k keyword => 根据keyword找到相关指令
```

man指令可以查看类型说明：

代号	说明
1	在shell环境中可以操作的指令或者可执行文件
2	系统核心可以调用的函数或者工具
3	一些常用的函数或者函数库，大部分为c的函数库
4	设备文件的说明
5	配置文件或者某些文件的格式
6	游戏
7	协定，比如文件系统，网络协定的说明
8	系统管理员可用的命令
9	跟kernel有关的文件

which 显示命令的可执行文件的位置
type 查看命令的类型
whatis 显示非常简洁的命令说明
alias 给命令起别名

- 别名定义只影响当前 shell 环境和任何子shell程序的执行环境，别名定义不影响当前shell的父进程或shell调用的任何实用程序环境
- 如果想让别名在所有shell中生效，可以修改相关rc文件，如果时zsh形式的shell，可以修改当前用户目录下的.zshrc或者/etc/zshrc文件；如果是bash形式的shell，可以修改当前用户目录下的.bashrc或者/etc/bashrc文件

unalias 删除别名命令
apropos命令
根据某个关键字匹配适当的命令，相当于man -k, 用法如下：

```
man -k zip
apropos zip
```

系统命令可用性检查
linux系统中有以下几种方式检查系统命令是否可用：

- where 命令
  使用where命令来检测当前目录或 %PATH% 下，有没有我们所需要的命令，如果有返回0,否则返回1，根据返回结果判断命令是否存在

```
where ls
echo $?   # 返回0
```

- which 命令
  which是一个可以返回可执行命令完整路径的命令。当相应命令存在时，会打印其在系统上的位置

```
$ which bash
/usr/bin/bash
$ echo $?    # 返回0
$ which nonexistent
$ echo $?    # 返回1
```

- type 命令
  type是shell内置的命令，可以用于显示命令的类型

```
$ type bash
bash is /usr/bin/bash
$ echo $? # 返回0
$ type nonexistent
```

```
nonexistent: not found
$ echo $? # 返回1
```

- hash 命令
  hash 是 shell 内置的可以用于缓存命令在磁盘上位置的命令，成功执行时没有任何输出，返回 0；而执行失败时，则会在 STDERR 上打印相应的提示信息，并返回一个大于 0 的值
  使用 hash 命令来检测命令可用性有一额外的副作用就是会将被检测的命令缓存在 shell 的 hash 表中以减少后续定位该命令所需的时间


```
$ hash bash
$ echo $?   # 返回0
$ hash nonexistent
sh: 1: hash: nonexistent: not found
$ echo $?   # 返回1
```

- command命令
  command 同样也是 shell 内置的命令，可以用于执行命令，或者使用 -v 选项以显示关于命令的有关信息，当能够找到相关匹配的命令时，command 返回 0，否则会返回大于一个 0 的值

```
$ command -v bash
/usr/bin/bash
$ echo $?   # 返回0
$ command -v nonexistent
$ echo $?   # 返回1
```

- 脚本的方式判断

```
CMD=cmd
if which "$CMD" >/dev/null 2>&1; thenecho 命令 $CMD 存在
elseecho 命令 $CMD 不存在
fi
```

文件/文件夹/路径相关操作指令：
pwd 打印当前目录路径
ls 列出当前路径下的文件和文件夹

```
ls  -A   => 列出除了. 及 ..以外的任何文件
ls -a    => 列出包括.及..在内的所有文件
ls       => 列出除了.及..以及以.开头以外的所有文件
ls -p    => 如果是文件夹，后加“／”
ls -l    => 文件的详细描述
ls -S    => 从大到小排序
ls -Sr   => 从小到大排序
ls --full-time => 显示完整文件的修改时间
ls -l --time=atime 显示文件最近被读取的时间 mtime表示表示数据变更的时间，ctime表示状态变
=>drw-r--r--     1 zhangdianpeng  staff     455  3 19 16:33 Dockerfile
```

cd 切换目录

```
cd ～ => 进入当前用户的根目录
cd -  => 返回到进入当前目录之前的目录
cd    => 等于cd ~
```

file 查看文件或者文件夹的类型
cat 将文档在控制台输出 ， -n输出行号，-s禁止输出空白行, -A将空白字符输出
less 打开文档，可以上下页切换
stat 命令

显示文件的详细信息，是ls的加强版
rm 删除文件或者文件夹

```
-r => 删除当前目录及子目录下的所有文件和文件夹
-f => 删除前无需询问，直接删除
-i => 删除前需要询问后再删除
```

cp 文件及文件夹的复制
mkdir 创建文件目录

```
mkdir -p dir1/dir2/dir3  # 可以是一个路径名称。此时若路径中的某些目录尚不存在,加上此-p选
```

touch 创建文档，如果文档存在，可以修改文档的更新，修改时间
mv 剪切文件并复制到其它位置
basename命令
获取最后的文件名
dirname命令
获取目录路径
几个查找文件的命令

- whereis 从一些特定的目录中寻找文件，主要是针对/bin,/sbin下面的可执行文件，usr/share/man下面的说明文件以及几个比较特定目录来查找，所以速度比较快

```
-b # 只查找binary格式的文件
-m # 只查找manual路径下的文件
-s # 只找source来源文件
-u # 搜索不在上述三个路径中的其它特殊文件
```

- locate 从已经创建的数据库var/lib/mlocate里面寻找数据，不直接去读硬盘，所以速度非常快，因为数据库不是实时的，每天会更新一次数据库，我们可以通过updatedb命令主动更新数据库信息

```
-i # 忽略大小写
-c # 不输出文件名，仅输出数量
-l # 输出前几行
-r # 后面可以接受正则
-S # 输出locate所用到的数据库文件相关信息
```

- find 会查找硬盘，但是使用起来比较灵活，可以指定目录查找，还可以根据权限，创建者，更新时间等内容进行查找

```
find /etc -name "*httpd*"   # 在/etc目录下查找所有包含httpd字符串的文件
find /run -type s           # 在/run目录下找到类型为Socket的文件名有哪些
find /home -user dmtsai     # 在/home目录下找到术语dmtsai用户的文件
find /etc -size +50k        # 在/etc目录下找到大于50k的文件
```

shell 环境相关配置和使用：
shell配置简介

- zsh和bash是常用的shell
- bash 的环境变量配置在当前用户目录下的.bash_profile和.bashrc文件或者/etc/bashrc文件
- zsh 的环境变量配置在当前用户目录下的.zshrc文件或者/etc/zshrc文件
- 当shell是交互式登录shell时，读取.bash_profile文件，如在系统启动、远程登录或使用su -切换用户时；
- 当shell是交互式登录和非登录shell时都会读取.bashrc文件;
- .bash_profile只在会话开始时被读取一次，而.bashrc则每次打开新的终端时，都会被读取;
- 如果想让.bash_profile,.bashrc,.zshrc立即生效，可以使用source命令
- /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目录的配置文件中搜集shell的设置
- /etc/bashrc和/etc/zshrc所有用户共有，~ /.bashrc和~/.zshrc当前用户拥有

bash的环境配置文件详解

1. /etc/profile
   是每个使用者在登录时必须去读取的配置文件，包括一些环境变量配置：PATH，MAIL，USER，HOSTNAME等，以及一些bash的基本信息配置
2. /etc/profile.d/*.sh
   /etc/profile会调用/etc/profile.d/下面的脚本，这个目录下规范了操作接口的颜色，语系，vi，which命令别名等信息
3. ~/.bash_profile, ~/.bash_login, ~/.profile
   登录以后读完/etc/profile下的配置以后，接下来会调用每个人个人目录下的文件，会依序读取上述三个文件，只要有一个文件存在就不会继续读取后面的文件了

一般这三个文件会调用.bashrc脚本，还会设置~/bin目录到PATH环境变量，如果修改了.bashrc，我们可以用source命令立即让其生效

4. ~/.bashrc
   一般用户修改环境配置的时候，只需要修改.bashrc这个文件即可
5. ~/.bash_history
   所有的历史命令都记录在这里
6. ~/.bash_logout
   系统退出登录时调用这个shell脚本

shell状态查看和切换

```
cat /etc/shells   => 查看系统拥有的shellchsh -s /bin/zsh  => 从当前的shell切换到zsh
chsh -s /bin/bash => 从当前shell切换到bash
source .zshrc     => 让.zshrc修改立即生效
echo $SHELL       => 查看当前使用的shell环境
```

history命令

history命令用于显示历史执行的命令

```
history 200         # 显示最近执行的前200条记录
echo $HISTFILESIZE  # 用于设置history最多保存的命令条数
```

clear 命令

清空当前shell

!number 命令

执行history中历史记录行号对应的命令

printenv命令

查看有效的环境变量列表

set命令

显示环境变量和shell变量

env命令

列出当前系统的环境变量

export命令

export命令主要是用于设置或显示环境变量

```
export -p          # 列出当前环境变量值
export MYENV=7     # 定义环境变量并赋值
```

source命令

- source命令让控制台强制去读文件（一般控制台直只有在启动的时候会读取必要的文件）
- source命令和普通的执行脚本的命令不同之处是，source命令可以让脚本在父程序中执行，所以脚本中设置的变量等内容都会在父程序中生效

command命令

command命令导致shell将指定的命令和变量视为简单命令，禁止了shell功能查询。通常，当 /（斜杠）不在命令（表示特定的路径）之前时，shell 通过搜索以下类别找到命令：

- 特殊 shell 内置
- shell 函数
- 常规的 shell 内置
- PATH 环境变量

如果以上查询中存在具有与常规的内置相同名称的函数，那么系统使用该函数。command 命令允许您直接调用具有与某个函数相同名称的内置命令。

```
command -v md5 "hello world"   # 返回0表示找到该命令并执行成功，返回大于0的数字表示找不
```

linux中常用的环境变量

变量名称	含义
SHELL	当前使用的脚本解析器
PATH	执行命令的搜索路径
USER	当前用户
HOME	用户的工作目录
IFS(Internal Field Seprator)	内部的域分隔符，默认值是“空格，tab, 和换行”，常见使用场景“$*”,for循环

shell常用快捷键

快捷键	功能

快捷键	功能
ctrl + a (fn + <=)	移动到行首
ctrl + e (fn + =>)	移动到行末
ctrl + f	光标后移一个字符
ctrl + b	光标前移一个字符
ctrl + d	删除光标处的字符
ctrl + k	剪切当前位置到行末的内容
ctrl + u	剪切当前位置到行首的内容
ctrl + p	移动到上一个执行的命令
ctrl + n	移动到下一个执行的命令
ctrl + j	执行命令，相当于按下回车键
ctrl + r	反向递增条件搜索执行过的命令

sh命令

执行某个脚本

```
-n            # 检查语法错误
-x            # 执行脚本，并将执行到的内容显示在屏幕上，便于调试
-v            # 在执行之前，先把脚本内容打印到屏幕上
```

文本操作相关命令：

几个文件查阅的命令

- cat: 由第一行开始显示文件内容
- tac: 从最后一行开始显示文件内容
- nl: 显示的时候顺便输出行号
- more：一页一页的显示文件内容
- less： 和more类似，但是可以向前翻页
- head：只看头几行
- tail：只看尾部几行
- od：以二进制的方式显示文件内容

cut命令

cut命令用来显示行中的指定部分，删除文件中指定字段

该命令有两项功能，其一是用来显示文件的内容，它依次读取由参数file所指明的文件，将它们的内容输出到标准输出上；其二是连接两个或多个文件，如cut fl f2 > f3将把文件fl和几的内容合并起来，然后通过输出重定向符“>”的作用，将它们放入文件f3中

fmt命令

fmt命令读取文件的内容，根据选项的设置对文件格式进行简单的优化处理，并将结果送到标准输出设备

```
-c或--crown-margin                          # 每段前两列缩排；
-p<列起始字符串>或-prefix=<列起始字符串>    # 仅合并含有指定字符串的列，通常运用在程序语
-s或--split-only                            # 只拆开字数超出每列字符数的列，但不合并字数
-t或--tagged-paragraph                      # 每列前两列缩排，但第1列和第2列的缩排格式不
-u或--uniform-spacing                       # 每列字符之间都以一个空格字符间隔，每个句子
-w<每列字符数>                              # 设置每列的最大字符数。
```

echo命令

```
echo ~  # 打印当前用户目录
echo $((2 + 2))    # 打印算法表达式  $((expression))表示算术表达式展开
echo $(($((5**3)) * 3))  # 算术表达式展开支持嵌套  =5*5*5*3
echo Front-{A,B,C}-Back  # 花括号展开：Front-A-Back Front-B-Back Front-C-Back
echo $USER               # 变量展开
echo $(ls)               # 命令替换
ls -l $(which cp)        # 命令替换
echo -e "\n" '$@=====';  # -e参数对特殊字符进行转译
```

sort命令

根据字符串内容排序，用法如下：

```
ls | sort     # 默认降序
ls | sort -s  # 升序
```

uniq 命令
去除重复的行

```
uniq -d  # 只显示重复的行
uniq -i  # 比较时忽略大小写
```

grep 命令
grep命令是一个强大的文本搜索工具：

```
cat app.js | grep let;
grep let -n app.js;         # -n 显示行号
grep let -i app.js:         # -i 不区分大小写
grep let -v app.js          # -v 只显示不匹配的文本行
grep let -w app.js          # -w 匹配整个单词
grep -e let -e app app.js   # -e 多模式匹配，可以匹配多个不同的字符串，它们之间是或的关
grep -E "app|let" app.js    # -E 指定扩展正则表达式匹配，比如 |，?,+,()等操作，等价于eg
grep -B 3 -A 2 let app.js   # -B 打印匹配的前面几行 -A打印匹配的后面几行
grep -C2 let app.js         # -C2 相当于 -B 2 -C 2
```

wc命令
用于显示文本的行数，字节数以及单词的数量

```
wc -l test.js   # 显示文本行数
wc -w test.js   # 显示文本单词数量
wc -c test.js   # 显示文本的字节数
```

head命令
打印文本开头几行

```
head -n 5 app.js  # 显示文本的前五行
```

tail命令
打印文本的末尾几行

```
tail -n 5 app.js  # 打印文本的末尾5行
```

od命令
od命令用于输出文件的八进制、十六进制或其它格式编码的字节，通常用于显示或查看文件中不能直接显示在终端的字符

```
echo abcdef g > tmp         # 先准备一个tmp文件
od -b tmp                   # 以八进制的方式输出
=>  0000000 141 142 143 144 145 146 040 147 012         # 左侧是地址格式0000011od -c tm
=>  0000000   a   b   c   d   e   f       g  \n     0000011    od -t d1 tmp
=>  0000000   97   98   99  100  101  102   32  103   100000011od -N 2 -j 2 -c tmp
=>  0000002   c   d0000004od -w3 -b tmp               # -w3表示每行只输出3个字节
=>  0000000 141 142 1430000003 144 145 1460000006 040 147 0120000011
```

xxd命令
xxd命令是对于标准输入或者给定的文件，显示其16进制的内容

- -b以二进制的方式显示
- -c每行输出多少个字节
- -g把几个字节组成一组

awk命令
awk是一种编程语言，用于在linux/unix下对文本和数据进行处理。数据可以来自标准输入(stdin)、一个或多个文件，或其它命令的输出。它支持用户自定义函数和动态正则表达式等先进功能，是linux/unix下的一个强大编程工具。它在命令行中使用，但更多是作为脚本来使用。awk有很多内建的功能，比如数组、函数等，这是它和C语言的相同之处，灵活性是awk最大的优势。
awk的使用方式：

```
awk 'BEGIN{ commands } pattern{ commands } END{ commands }'
```

- 第一步：执行BEGIN{ commands }语句块中的语句，BEGIN语句块在awk开始从输入流中读取行之前被执行，这是一个可选的语句块，比如变量初始化、打印输出表格的表头等语句通常可以写在BEGIN

语句块中

- 第二步：从文件或标准输入(stdin)读取一行，然后执行pattern{commands}语句块，它逐行扫描文件，从第一行到最后一行重复这个过程，直到文件全部被读取完毕。pattern语句块中的通用命令是最重要的部分，它也是可选的。如果没有提供pattern语句块，则默认执行{ print }，即打印每一个读取到的行，awk读取的每一行都会执行该语句块
- 第三步：当读至输入流末尾时，执行END{ commands}语句块。END语句块在awk从输入流中读取完所有的行之后即被执行，比如打印所有行的分析结果这类信息汇总都是在END语句块中完成，它也是一个可选语句块。

示例如下：

```
echo -e "A line \nB line 2" | awk 'BEGIN{ print "Start" } { print } END{ print "End" }'
=>  StartA line 1B line 2Endecho -n "$IFS" | xxd -g 1 | awk -F":" '{print $2}' | awk -F
=> 20 09 0a       # 打印分隔符(IFS)的16进制
```

awk命令功能很强大，详解介绍见:awk命令

sed命令

sed是stream editor的简称，也就是流编辑器。它一次处理一行内容，处理时，把当前处理的行存储在临时缓冲区中，接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非你使用重定向存储输出

```
sed -i "s/require/require1/" app.js  # 遍历app.js中的每一行，并将require替换成require1,
sed '/require/a xxxx' app.jd         # 遍历app.js中的每一行，遇到满足条件匹配规则，会在
sed '/1.1.1.1/d' app.js              # -d表示删除满足条件的那一行
nl app.js | 'sed 2,5d'              # 将app.js中的2到5行删除然后打印出来# 常用模式
a    # 新增（下一行）
c    # 取代
d    # 删除
i    # 新增（上一行）
s    # 取代，和正则表达式一起使用# 参数说明
-f   # 修改后的内容写入文件
-i   # 直接修改原始文件，而不打印到控制台
-n   # 只会打印出经过sed处理的那几行内容
```

read命令

接受来自键盘的输入并赋值到某个变量

```
read atest                # 等待键盘输入并赋值给变量atest
read -p "message" atest   # -p参数打印一些提示信息然后等待输入
read -t 30 atest          # -t参数表示最多等待30s输入时间
```

tr命令

用于对于一段文字中的字符作删除或者替换操作

```
cat PlatformDeploy/base.sh | tr -d "auto"    # 删除base.sh中所有是a，u，t，o的字母
cat PlatformDeploy/base.sh | tr  [a-z] [A-Z] # 将base.sh中所有的小写字母替换为大写字母
```

col命令

col命令用于将tab键转换成对等的空白键

```
cat PlatformDeploy/base.sh | col -x
```

join命令

处理两个文件中的数据，将两个文件中“有相同数据”的那一行加在一起

```
join test1.js test2.js           # 根据test1.js和test2.js中每一行中第一个字段作区分，如
join -t ':' test1.js test2.js    # -t参数指定分隔符
join -1 3 test1.js -2 4 test2.js # 根据test1.js中的第三个字段和test2.js中的第四个字段作
```

paste命令

只是简单的将两个文件中相同的行合并在一起并输出

```
paste file1 file2
```

split命令

用于将大的文件或者输入流按照字节大小或者行数分为多个文件

```
split -b 300k file1      # 将file1文件按照每300k的大小分为多个文件
split -l 20 file1        # 将file1文件按照每20行的行数分为多个文件
PREFIX                   # 该参数可以指定文件前缀
```

diff命令
用于对比两个文件，主要用于两个文件的“行”为单位进行处理
cmp命令
用于对比两个文件，以字节为单位进行对比，主要用于二进制文件的对比
patch命令
可以根据diff命令出来的结果对旧的文件进行升级修补
pr命令
进行文件打印命令
test命令
用于测试文件类型及对比文件信息，也可以使用[]来判断结果是否正确，效果一样

```
test -e test.js && echo "exist" || echo "not exist" # -e参数判断文件是否存在
test -f test # 是否存在且为文件
test -d test # 是否存在且为文件夹
```

seq命令
用来序列化一串数据

```
seq 1 20     # 产生1-20的顺序数字
seq 1 2 30   # 步长为2，默认为1
seq -s       # 指定分隔符，默认为\n
```

用户/角色/权限相关命令：
id 命令
查看个人的用户信息，个人id，以及所属的组id

```
id
=> uid=501(zhangdianpeng) gid=20(staff) groups=20(staff),701(com.apple.sharepoint.group
```

groups 命令
查看用户所在的所有群组信息
usermod 命令
修改用户信息，包括群组，密码等信息
gpasswd 命令
群组管理员将用户加入他所在群组
newgrp 命令
变更用户的有效群组
useradd 命令
用来创建用户账号
passwd 命令
修改用户密码
chage 命令
列出用户密码的详细信息，过期时间啥的
userdel 命令
删除用户及其相关信息
chsh 命令
修改用户的默认shell
groupadd 命令
新建群组
groupmod 命令
修改群组
groupdel 命令
删除群组相关信息
ls -l filename
查看文件的执行权限，返回10个字符的文件属性：第一个字符表示文件类型，剩余9个表示文件所有者，文件组所有者以及其他人的读，写，执行权限
chmod
修改文件所有者，文件所有组，其他人的权限

选项	功能
u g o	分别表示用户，组，其他人

选项	功能
x w r a	分别表示执行权限，写权限，读权限，所有权限
+ -	分别表示添加权限，删除权限

```
chmod 765 file          # 给User、Group、及Other的权限分别设置为7，6，5
chmod ugo+r file1.txt   # file1.txt文件设置User、Group、及Other都可以读取，相当于chmod
chmod u+x ex1.py        # 给文件所有者赋值可执行权限
chmod o-w file2.txt     # 删除其他人的写权限
chmod o+a file3.txt     # 给其他人赋值所有权限（a表示所有权限）
```

chown 更改文件的所有者和所有组
普通用户不能将自己的文件改变成其他的拥有者，其操作权限一般为管理员。

```
chown mail:mail log2012.log   # 将文件的用户和用户组改为mail和mail
chown :mail log2012.log       # 将文件的用户组改为mail
chown -R youdata logs         # -R参数表示更改当前文件以及子文件夹的所有者和所有组
```

chgrp 改变文件所属用户组

```
chgrp -R user smb.conf  # -R参数表示递归修改
```

umask命令

- umask用来指定新建文件时的权限，用户每次进入系统，umask命令都会被执行，并自动设置掩码来限制新建文件的权限。
- umask命令可以指定哪些权限将在新文件的默认权限中被删除，与chmod命令恰恰相反
- umask默认值是022

```
umask 022 # 此时你新建文件的权限将是622
```

lsattr和chattr命令
这两个命令是用来查看和改变文件、目录属性的，与chmod这个命令相比，chmod只是改变文件的读写、执行权限，更底层的属性控制是由chattr来改变的

```
# chattr的用法: chattr [+-=] [ASacdijsu] fileName+ ： 在原有参数设定基础上，追加参数。
- ： 在原有参数设定基础上，移除参数。
= ： 更新为指定参数设定。A: 文件或目录的 atime (access time)不可被修改(modified)，可以有
S: 硬盘I/O同步选项，功能类似sync。
a: 即append，设定该参数后，只能向文件中添加数据，而不能删除，多用于服务器日志文件安全，
c: 即compresse，设定文件是否经压缩后再存储。读取时需要经过自动解压操作。
d: 即no dump，设定文件不能成为dump程序的备份目标。
i: 设定文件不能被删除、改名、设定链接关系，同时不能写入或新增内容。i参数对于文件 系统的
j: 即journal，设定此参数使得当通过mount参数: data=ordered 或者 data=writeback 挂 载的文
s: 保密性地删除文件或目录，即硬盘空间被全部收回。
u: 与s相反，当设定为u时，数据内容其实还存在磁盘中，可以用于undeletion。
各参数选项中常用到的是a和i。a选项强制只可添加不可删除，多用于日志系统的安全设定。而i是更
```

```
# lsattr用法: lsattr /etc/resolv.conf
```

打包和解压相关命令:
tar 命令

```
tar -xvf FileName.tar               # 将tar包解到当前目录中   -f指定包名字， -x表示解开
tar -cvf FileName.tar DirName       # 将DirName中的内容打包到fileName.tar中  -c表示建立
tar -zcvf FileName.tar.gz DirName   # 压缩包
tar -zxvf
```

gzip命令
创建压缩扩展名为*.gz的文件名

```
gzip -v services    # 将services文件夹压缩为services.gz文件，并删除原文件，-v参数显示压
gzip -d serivces.gz # -d参数表示解压
gzip -8 services    # 设置压缩等级进行压缩，默认为6，-1最快，-9最慢，但压缩比最好
```

zcat／zmore／zless命令
用于读取gzip压缩后的文件内容

bzip2命令
bzip2和gzip使用方式一模一样，但是比gzip压缩比更高，但是压缩时间更长，压缩扩展名为.bz2
bzcat/bzmore/bzless命令
用于读取bzip2压缩后的文件内容
xz命令
比bzip2压缩比更高的压缩软件，扩展名为.xz
xzcat/xzless/xzmore命令
用于读取xz压缩的文件内容
常用工具命令：
cal命令
显示日历命令

```
cal    # 显示当月的日历
cal -y # 显示当年的日历
```

date命令
显示当前日期
md5命令和md5sum命令
md5命令和md5sum命令都是采用MD5报文摘要算法(128位)计算和检查文件的校验和,mac上有md5命令，而linux系统是有md5sum命令

- -b：二进制模式读取文件；
- -t或–text：把输入的文件作为文本文件看待；
- -c：从指定文件中读取MD5校验和，并进行校验；
- –status：验证成功时不输出任何信息；
- -w：当校验不正确时给出警告信息

bc命令
计算器命令，可以直接执行加减乘除等计算操作
mail命令
发送和接收邮件命令

```
mail -s test -c admin@aispider.com  root@aispider.com< file
```

开关机命令

- halt：进入系统停止模式，屏幕可能出现一些讯息
- poweroff：进入系统关机模式，直接关机没有提供电力
- reboot：直接重新开机
- suspend：进入休眠模式
- shutdown：可以在指定时间进行关机操作

sync命令
强制将内存中没有被更新的数据写到硬盘中，所以在关机前，可以使用sync命令，防止内存中的数据未保存
shell脚本相关知识：
shell中特殊字符的含义

```
$     # 当前shell的PID
?     # 上一个命令的返回结果
#     # 表示注释
```

declare命令
declare用于声明变量，可以指定变量类型等操作

```
-a  # 将变量定义成array
-i  # 将变量定义成int
-x  # 将变量导出为环境变量
-r  # 将变量设置为只读模式
```

双引号和单引号
使用双引号，我们可以阻止单词分割，而单引号可以禁止所有的展开（参数展开，表达式展开，命令替换等）

```
ls -l "two words.txt"
echo "$USER $((2+2)) $(cal)"   # 双引号中，参数展开，算术表达式展开，命令替换仍然生效
```

变量部分替换命令

```
${var#pattern}                    # 从变量头部开始匹配模式，将符合的最短数据删除
${var##pattern}                   # 从变量头部开始匹配模式，将符合的最长数据删除
${var%pattern}                    # 从变量尾部开始匹配模式，将符合的最短数据删除
${var%%pattern}                   # 从变量尾部开始匹配模式，将符合的最长数据删除
```

```
${var/oldPattern/newPattern}        # 将第一个符合旧模式的数据替换为新模式
${var//oldPattern/newPattern}       # 将全部符合旧模式的数据替换为新模式
${var:offset}／${var:offset:length} # 从变量第offset个字符开始，提取length个字符
${var:-word}                        # 若var没有设置或为空，展开结果是word的值并且word的
${var:?word}                        # 若var没有设置或为空，这种展开导致脚本带有错误退出
${var:+word}                        # 若var没有设置或为空，展开结果为空。若var不为空，展
${#var}                             # 展开成由var所包含的字符串的长度。通常，var是一个字
${var,,}                            # 将var全部展开成小写字母
${var,}                             # 将var的首字母展开成小写字母
${var^^}                            # 将var的全部展开为大写字母
${var^}                             # 将var的首字母为大写字母
```

shebang in linux(!#)
Shebang这个符号通常在Unix系统的脚本中第一行开头中写到，它指明了执行这个脚本文件的解释程序:

- 如果脚本文件中没有#!这一行，那么它执行时会默认用当前使用Shell去解释这个脚本(即：$SHELL环境变量）。
- 如果#!之后的解释程序是一个可执行文件，那么执行这个脚本时，它就会把文件名及其参数一起作为参数传给那个解释程序去执行。
- 如果#!指定的解释程序没有可执行权限，则会报错“bad interpreter: Permission denied”。
- 如果#!指定的解释程序不是一个可执行文件，那么指定的解释程序会被忽略，转而交给当前的SHELL去执行这个脚本。
- 如果#!指定的解释程序不存在，那么会报错“bad interpreter: No such file or directory”。
- 注意：#!之后的解释程序，需要写其绝对路径（如：#!/bin/bash），它是不会自动到$PATH中寻找解释器的。
- 当然，如果你使用"bash test.sh"这样的命令来执行脚本，那么#!这一行将会被忽略掉，解释器当然是用命令行中显式指定的bash
  命令的顺序执行
- cmd1;cmd2: cmd1和cmd2顺序执行，两者没有任何关联
- cmd1&&cmd2: 只有cmd1执行正确才会执行cmd2
- cmd1||cmd2: 只有cmd1执行错误才会执行cmd2
  printf命令
  类似c语言中的printf函数一样，可以对文本进行格式化打印

```
printf '%10s\t %5i %5i %5i %8.2f\n' $(cat printfTest | grep -v Name)
```

"$"开头的含义

- $? : 上一条命令的返回结果，一般情况下，0表示返回成功，大于0的表示返回失败
- $# : 输出命令行参数或者函数参数的个数
- $0 : 命令行第0个参数，表示命令行执行的路径
- $1/$2/$3: 输出第一个/第二个／第三个命令行参数
- $*: 展开成一个从1开始的位置参数列表。当它被用双引号引起来的时候，展开成一个由双引号引起来的字符串，包含了所有的位置参数，每个位置参数由shell变量IFS的第一个字符(默认为一个空格)分隔开
- $@: 展开成一个从1开始的位置参数列表。当它被用双引号引起来的时候，它把每一个位置参数展开成一个由双引号引起来的分开的字符串
  我们写一个脚本shell.sh如下：

```
#!/bin/bash
# print_params函数分别打印参数，第0-第4个参数
print_params () {echo "\$# = $#";echo "\$0 = $0";echo "\$1 = $1";echo "\$2 = $2";echo "
}
pass_params () {echo -e "\n" '$*=====';print_params   $*;echo -e "\n" '"$*"=====';print
}
pass_params "word" "words with spaces"
```

运行上述脚本(./shell.sh)输出结果如下：

- $0的结果都一样，是"./shell.sh"
- 没有加双引号的*和*和*和@输出结果一样，都是按照空格将参数分开传进来的
- 加了双引号的*和*和*和@就结果完全不一样，前者把所有的字符串拼接成一个参数传入，后者却按照传入第一个参数的方式（两个参数）传入到第二个参数

```
$*=====
$# = 4
$0 = ./shell.sh
$1 = word
$2 = words
```

```
$3 = with
$4 = spaces"$*"=====
$# = 1
$0 = ./shell.sh
$1 = word words with spaces
$2 =
$3 =
$4 =$@=====
$# = 4
$0 = ./shell.sh
$1 = word
$2 = words
$3 = with
$4 = spaces"$@"=====
$# = 2
$0 = ./shell.sh
$1 = word
$2 = words with spaces
$3 =
$4 =
```

linux中的数组

```
arr[1] = "hello";               # 给数组赋值方式1
arr= ("hello", "world", "how", "are", "you");     # 给数组赋值方式2
echo $arr[1];                   # 通过下标访问数组元素
echo ${arr[*]};                 # 通过*访问数组中所有元素，和位置参数一样
echo "${arr[*]}";               # 加引号后的区别与位置参数"$*"加引号一致
echo ${arr[@]};                 # 通过@访问数组中所有元素，和位置参数一样
echo "${arr[@]}"                # 加引号后的区别与位置参数"$@"加引号一致，这个最常用
echo ${#arr[@]}                 # 输出数组元素的长度
arr+=(d e f)                    # 在数组末添加元素
unset arr                       # 删除数组中的所有元素
unset 'arr[2]'                  # 删除数组中下标为2的元素
```

for循环

linux中的for循环有以下三种语法，linux下for循环中可以使用break和continue关键字来跳出循环

```
#语法一：分别列出以环境变量IFS分割出的字符串
for i in 1 2 3 4 5   do    echo "$i-->$(uptime)"  done#语法二：遍历命令的执行结果
for i in `ls ./*.tar.gz`   do    tar -zxvf $i >/dev/null  done #语法三：类c语言的for循环
sum=0
for (( i=1; i<=100; i++ ))  do    sum=$(( $sum + $i ))  done  echo "1+2+3+...+100=$sum"
IFS=$'\n';    # 只使用换行符分割，而不使用空格。这里不能使用'\n',这个时候是把斜杠，n作为
for i in `ls -l`
do
echo "$i-------"
done
IFS=IFS_COPY# 用for循环遍历数组并计算数组中所有元素的和
sum=0;
arr=(23 33 44)for item in ${arr[@]}dosum=$(($sum + $item))doneecho "sum of arr = $sum"
```

磁盘和文件系统相关命令：

df命令

df命令用于列出文件系统的整体磁盘使用量

```
df -a       # 列出所有文件系统的使用量，包括系统特有的/proc等文件系统
df -k       # 以KBytes的容量显示文件系统
df -m       # 以MBytes的容量显示文件系统
df -h       # 自动选择显示单位进行显示
df -i       # 不显示磁盘容量，而显示inode数量
df -h /etc  # 如果指定目录，会自动分析该目录或者文件所在的partition，并将该partition的容
```

du命令

du命令用于查看某个文件的下的磁盘使用情况

```
du -s     # 列出当前文件的大小
du -s *   # 列出当前文件夹下每个文件的大小
```

```
du -S     # 不包括子目录的统计
du        # 列出所有文件，包括子文件，子文件的子文件
```

fuser命令
通过fuser命令我们可以找出使用该文件或者文件系统的程序

```
-u          # 除了程序的PID之外，同时列出该程序的拥有者
-m          # 查看该文件所在的文件系统的情况
-v          # 可以列出程序的相关指令的完整性
fuser -uv . # 查看当前目录被哪些程序使用
fuser -umv .# 查看当前目录所在的文件系统的程序使用情况
```

lsof命令
列出被程序所打开的文件和文件名

```
lsof                 # 列出目前系统上所有已经被打开的文件和设备
lsof -u root -a -U   # 列出root所打开的socket文件，-a表示且的关系，-U表示是socket程序
lsof +d /dev         # 列出目前系统上面所有的被启动的周边设备，+d后面接目标，表示指定目
```

磁盘分区相关命令

- lsblk 列出系统上所有磁盘列表
- blkid 列出每个磁盘设备的UUID参数，每个设备的唯一识别码
- parted 列出磁盘的分区情况（磁盘的总容量／分区表格式／磁盘的模块名称／逻辑物理扇区容量）
- fdisk MBR分区表使用fdisk命令进行分区
- gdisk GPT分区表使用gdisk命令进行分区
- parted 可以兼容MBR和GPT分区表，进行分区
- partprobe 更新Linux核心分区表信息
- mkfs 格式化磁盘命令
- xfs_repair 检查并修复文件系统的错误
- mount 挂载文件系统
- umount 卸载文件系统
- dd 可以创建大型文件来取代磁盘无法分区的情况
- /etc/fstab 可以设置开机启动自动挂载，利用mount -a检查语法是否正确
- dd命令 dd命令用于复制文件并对原文件的内容进行转换和格式化处理。dd命令功能很强大的，对于一些比较底层的问题，使用dd命令往往可以得到出人意料的效果。用的比较多的还是用dd来备份裸设备

xfsdump命令用于备份文件系统

```
1.不仅可以进行完整备份，还可以进行累计备份
2.只能备份已经挂载的文件系统
3.只能对完整的文件系统进行备份，不能备份其它目录
4.只有root权限可以操作
```

xfsrestore命令用于还原文件系统

```
1.用于对xfsdump命令备份的数据进行还原
2.xfsrestore -l 用于查阅xfsdump备份了哪些内容
3.还原的过程同名文件会被覆盖，其它新的文件会被保留
```

定时任务相关命令：
at命令
at进行单一的工作调度，可以让系统在指定时间执行某些脚本。
可以使用/etc/at.allow和/etc/at.deny这两文件对at的使用进行限制，/etc/at.allow指定了容许使用at的使用者，如果这个文件不存在那么会去判断/etc/at.deny,这个文件指定了不容许指定at命令的用户，如果两个文件都不存在，那么只有root用户才可以指定at命令

```
at now + 5 minutes                                          # 在五分钟后执行命令，按下a
at> /bin/mail -s "testing at job" root < /root/.bashrc      # 5分钟后发邮件给root
at> <EOT>                                                   # 按下contrl + D 结束设置
at -l                                                       # 显示目前存在的所有的at调度
at -c 2                                                     # 显示调度号为2的详细内容
at -m                                                       # 执行完调度任务会发送一条计
```

batch命令
batch命令和at命令基本一致，只是batch命令更人性化，它是系统的工作负载小于0.8以后才会执行任务
crontab命令

- crontab进行循环型的工作调度，可以让系统在每天，每小时的指定时间执行某些任务

- 可以使用/etc/cron.allow和/etc/cron.deny这两文件对crontab的使用进行限制，/etc/cron.allow指定了容许使用crontab的使用者，如果这个文件不存在那么会去判断/etc/cron.deny,这个文件指定了不容许指定crontab命令的用户，如果两个文件都不存在，那么只有root用户才可以指定crontab命令
- 如果是系统性的定时任务，一般不需要crontab -e来编辑，只需要编辑/etc/crontab这个文件就可以了，只有root才可以编辑
- 每个用户的crontab任务最终都存放在/var/spool/cron/crontabs中
- 如果是自己开发的软件可以使用/etc/cron.d/newfile，比较独立，全新的文件目录
- 如果是固定每小时，每日，每周，每天指定的特别工作，如果与系统维护有关，建议放在/etc/cron.daily,/etc/cron.monthly, /etc/cron.hourly, /etc/cron.weekly文件里执行

```
crontab -e    # 编辑系统的crontab任务
crontab -l    # 查阅系统的crontab任务
crontab -r    # 移除所有的crontab任务
```

anacron命令

- anacron会定时去检查有没有相关的调度任务被执行，如果超过执行期限，就会执行该调度任务
- 主要为了解决去执行由于某些原因比如关机导致没有执行的调度任务
- anacron每小时都会被执行一次

```
anacron -s  # 开始一连串的执行工作，根据记录文件顺序判断是否要执行
anacron -n  # 立刻执行未执行的任务，不进行延迟
anacron -f  # 强制执行而不会判断时间记录文件的时间戳
```

文件的编译和运行:

gcc命令

对c语言编写的文件进行编译，编译成可执行的二进制文件

```
gcc hello.c                      # 直接编译成out.c的二进制文件
gcc -c hello.c                   # 将hello.c转换为目标文件hello.o
gcc -o hello hello.o             # 将hello.o转化为二进制文件hello
gcc -c hello1.c hello.c          # 将两个.c文件编译中间产物.o文件
gcc -o hello hello1.o hello.o    # 将hello1.o和hello.o编译成二进制文件hello，之所以要有
```

make命令

对gcc命令的封装，用于快速编译二进制文件，make的特点如下：

- 简化编译时需要下达的指令
- 如果编译完成后，有源文件作了修改，make会针对改动的源文件进行编译，而不会编译其它object文件
- 可以依照相依性更新可执行文件

make首先要编写一个makfile文件，如下：

```

# makefile文件编写
main: hello.o hello1.o                    # main要编译成的目标二进制文件，以及相依的.
make clean main                      # 先执行clean，再进行编译二进制文件
```