linux的基本原则
1、由目的单一的小程序组成，组合小程序完成复杂任务
2、一切皆文件
3、尽量避免捕获用户接口
4、配置文件保存为纯文本格式

命令格式
    命令 选项... 参数

选项：
    短选项
    长选项
参数：

list: ls
ls
    -l:长格式
        文件类型
            -：普通文件（f）
           d:目录文件
           b:块设备文件
           c:字符设备文件
           l:符号链接文件
           p:命令管道文件
           s:套接字文件
        文件权限
        文件硬链接的次数
        文件的属主
        文件的数组
        文件大小，单位为字节
        时间戳：最近一次被修改的时间
            访问
            修改：文件内容发生改变
            改变：改变元数据
        文件名
    -h:做单位转换
    -a:显示以.开头的文件
        .:表示当前目录
        ..:表示父目录文件
    -A:显示所有文件但是不显示.and..
    -d:显示目录属性
    -i:index node,inode 显示缩引节点号
    -r:逆序显示
    -R:递归显示

cd :change directory切换当前目录

type：显示指定命令属于哪种类型

列出，列表

文件夹：

命令类型：
    内置命令（shell内置），内部，内建
    外部命令：在文件系统的某个路径下有一个与命令名称相应的可执行文件

环境变量：命名的内存空间
    变量赋值
        NAME=JERRY

    printenv：显示环境变量

    PATH：使用冒号分割的路径

    hash：缓存O（1）

date:时间管理
LINUX：RTC
ntp：网络时间协议

硬件时钟：clock hwclock
系统时钟：date
获得命令的使用帮助：
内部命令：
    help COMMAND
外部命令：COMMAND --help
命令手册：
man COMMAND

whatis COMMAND 显示命令摘要信息

分章节
1：用户命令（/bin,/usr/bin,/usr/local/bin）
2：系统调用（字符集不匹配时：export LANG=en）
3：库用户
4：特殊文件（设备文件）
5：文件格式（配置文件的语法）
6：游戏
7：杂项
8：管理命令（/sbin,/usr/sbin,/usr/local/sbin）

<>必须给出的内容
[]可选
...可以出现多次
|：多选一
{}：分组

man:
    NAME：命令名称
    SYNOPSIS（大纲摘要）：用法说明，包括可用的选项
    DESCRIPTION：命令功能的相近说明，可能包括每个选项的意义
    OPTION：说明每一个选项的意义
    FILES：此命令的配置文件
    BUGS:
    EXAMPLES：使用示例
    SEE ALSO：另外参照

DATE：可以修改系统时间

翻屏：
    向后翻一屏：SPACE
    向前翻一屏：b
    向后翻一行：ENTER
    向前翻一行：k
查找：
/KEYWORD：向前
n：下一个
N：前一个
？KEYWORD：向后

q：退出

date +"This year is %Y.%n Today is %d."

将系统时间同步到硬件时间：hwclock -w
将硬件时间同步到系统时间：hwclock -s

hwclock
    -w:
    -s:
在线文档：
info COMMAND

/usr/share/doc ：主要看内核说明文档

cal：日历

查看是内部命令还是外部命令：type echo
内部命令的帮助：help echo
echo：显示一行、
echo - n不显示换行

printf

Windows：PE
Linux：ELF

根文件系统详解：
文件系统：
rootfs：根文件系统
FHS:Linux文件系统层级标准
/boot：系统启动相关的文件，如内核、磁盘，以及grub（bootloader）
/dev：设备文件
    设备文件：
        块设备：随机访问的设备,数据块
        字符设备：线性访问，按字符为单位
        !!!!!!!!!!!!!!!!!!!!!!!!(移除耳机）!!!!!!!!!!!!!!!!!!!!!!!
        设备号：主设备号，次设备号
/etc：配置文件
/home：用户的家目录，每一个用户的家目录通常默认为/home/USERNAME
/root：管理员的家目录
/lib：库文件和内核模块文件
    /lib/modules：内核模块文件
    静态库，   .a
    动态库，.dll， .so(shared object)
/media:挂载点目录，通常挂载移动设备（对于centos来说该目录挂载的为虚拟镜像）
/mnt：额外的临时文件系统
/opt：可选目录，第三方程序安装目录
/proc：伪文件系统，内核映射文件
/sys：伪文件系统，根硬件设备相关的属性映射文件
/tmp：临时文件
/var：可变化的文件
/bin：可执行文件，用户命令，启动相关
/sbin：管理命令，与启动相关
/usr：shared ，read-only 全局共享的只读文件
    /usr/bin
    /usr/sbin
    /usr/lib
/usr/local：第三方软件安装，非关键性
    /usr/local/bin
    /usr/local/sbin
    /usr/local/lib

命名规则：
1、长度不能超过255个字符。
2、不能使用/当文件名
3、严格区分大小写

相对路径：
绝对路径：

使用一个操作系统时所用到的功能和工作
编辑文件
目录管理
运行程序
设备管理
软件管理
进程管理
网络管理

文件管理
ls
cd
pwd
mkdir：创建空目录
    -p：递归创建
    -v：verbose详细信息
mkdir -pv /mnt/test/x/m /mnt/test/y
mkdir -pv /mnt/test/{x/m，y}
花括号：命令行展开
mkdir {a,d}_{b,c}可以创建四个文件夹
波浪线展开

删除目录：rmdir 删除目录，只能删除空目录
    rmdir -p只能删除一线单传的

文件的创建和删除：
touch：用来改变时间戳
    -a：访问时间
    -m：更改时间
    -t：修改到指定的时间
    -c：不创建文件

stat：查看文件的状态

创建文件，还可以使用文件编辑器

ASCII:美国标准信息交换代码
128不同的字符：
    二进制：

file：确定文件类型

\rm使用原命令
    -i提示是否删除
    -f强制删除
    -r递归删除所有文件
rm -rf

复制和移动文件
cp:
只能复制一个文件到一个文件
多个文件到一个目录
cp -r 复制目录
   -a ：归档复制，常用于备份q
   -p ：保留属主
   -L ：复制文件而不是链接
   -P ：如果为链接则为复制链接
   -f ：强行复制
   -i ：提示

mv：移动文件(目录均可)

install 只能为文件而不能为目录
    -d ：创建目录
       ：复制之后具有执行权限
    -m ：指定权限

//2-4end

//3-1

tree：查看目录树

目录管理：
ls cd pwd mkdir rmdir tree
文件管理
touch stat file rm cp mv nano
日期时间
date clock hwclock cal

查看文本：
cat tac more less head tail

cat：链接并显示
    -n ：显示行号
    -e ：显示换行符
    -v ：

Ctrl+c ：结束命令

分屏显示：
more less head
less：类似man命令显示格式比较好

tail -f 查看文件尾部，不退出，等待显示后续加至此文件的新内容

文本处理：
cut join sed awk

cut:
    -d：指定字段分隔符，默认是空格
    -f：指定要显示的字段
        -f 1：显示一
        -f 1,3：显示一三
        -f 1-3：显示一到三
//3-2
文本排序：sort
    -n：按着数值的大小排序
    -r：反向排序
    -t：字段分隔符
    -k：以哪个字段为关键字进行排序
    -u：重复的只显示一次 这个命令可以 sort -u $(find /)这样重复的就只显示一次
    -f：排序时忽略字符大小写
uniq：相邻的重复行进行忽略
    -d:打印重复的行
    -c：统计某一行重复的次数

文本统计：wc（word count）
    行数 单词数 字节数
    -c：字节数
    -m：字符数
    -l：行数
    -w：单词数
    -L：最长的一行包含多少个字符

字符处理命令：tr--实现转换或删除字符
tr 'a' 'A' < 文件
tr -d ：删除字符

bash及其特性：
shell：外壳
gui：gnome，kde，xfce
cli：sh，csh，ksh，bash，tcsh，zsh

进程：每个进程看来，当前主机上只存在和内核和当前进程
进程是程序的副本，进程是程序执行实例

用户工作环境：
bash：设置可能不同
    管理员#
    普通用户$
pstree：显示层次

bash：特性
1、命令历史
2、管道、重定向
3、命令别名
4、命令行编辑
5、命令行展开
6、文件名通配
7、变量
8、编程

命令行编辑：
光标跳转：
    ctrl+a：跳到命令行首
    ctrl+e：跳到命令行尾
    ctrl+u：删除光标以前的内容
    ctrl+k：删除光标以后的内容
    ctrl+左右箭头：一次跳一个单词
    ctrl+l：清屏
命令历史：
    上下箭头
    history：查看命令历史
    -c：清空命令历史
    -d：删除指定位置的命令 history -d 500 3 删除三个历史命令

-w：保存命令历史至历史文件中

环境变量
PATH：命令搜索路径
HISTSIZE：命令历史大小缓冲区大小 echo $HISTSIZE显示大小

命令历史的使用技巧：
！n 执行命令历史中第n条命令
！-n执行倒数第几行
！！执行上次的命令
！string：执行最近一次字符串开头的命令
！$：引用上一个命令的参数
esc .
alt+.

命令别名：
alias CMDALIAS='COMMAND'
unalias 撤销命令别名
\CMD不使用参数

命令替换：$(CMD),反引号 `CMD`
把命令中某个子命令替换为其执行结果的过程
echo “The current directory is $(pwd)."

file-2013-02-28-14-53-31.txt

touch ./file-$(date +%F-%H-%M-%S).txt

bash支持的引号：
``：命令替换
“”：弱引用，可以实现变量替换
‘’：强引用，不完成变量替换

文件名通配,globbing
*：任意长度的任意字符
?：任意单个字符
[]：匹配指定范围内的任意单个字符
    [abc],[a-m],[a-z],[A-Z],[0-9],[a-zA-Z], [0-9a-zA-Z]
[^]:匹配指定范围之外的任意单个字符
文件中包含空白字符:touch 'a b'
[[:space:]]:空白字符
[[:punct:]]:标点符号
[[:lower:]]:小写字母
[[:upper:]]:大写字母
[[:alpha:]]:大小写字母
[[:digit:]]:数字
[[:alnum:]]:数字和大小写字母

例如识别字母开头中间有空格字母结尾的字符串
ls [[:alpha:]]*[[:space:]]*[[:alpha:]]

man 7 glob ：获得上面的列表帮助

linux用户及权限

安全上下文

权限：
r，w，x
文件：
r：可读，可以使用类似cat等命令查看文件内容
w：可写，可以编辑或删除此文件
x：可执行，exacutable，可以在命令提示符下当作命令提交给内核运行

目录：
r：可以对此目录执行ls以列出内部的所有文件
w：可以在此目录创建文件
x：可以使用cd切换进此目录，也可以使用ls -l 查看内部文件的详细信息
默认文件一般不给执行权限，但是目录因该有

用户：UID， /etc/passwd
组：GID，/etc/group

影子口令：
用户：/etc/shadow
组：/etc/gshadow

用户类别：
管理员：id号为0
普通用户：1-65535
    系统用户：1-499   不允许登录系统
    普通用户：500-60000

用户组：
管理员组：
普通组：
    系统组：
    一般组：

用户组类别：
    基本组：用户的默认组

    私有组：创建用户组时，如果没有为其指定所属的组，系统会自动为其创建一个与用户名同名的组
    附加组，额外组：

进程：tom tom
对象：

/etc/passwd
account：账户
password：密码
UID：
GID：基本组ID
comment：注释
HOME DIR：家目录
shell：用户的默认shell

cat /etc/shadow
cat /etc/shells
cat /etc/group
cat /etc/gshadow

/etc/shadow
account：登录名
encrypted password：加密的密码

加密方法：
    对称加密：加密和解密使用同一个密码
    公钥加密：每个密码都成对出现，一个为私钥一个为公钥
    单向加密，散列加密：提取数据特征码，常用于数据完整性校验
        1、雪崩效应   md5sum 就可以计算其特征码
        2、定长输出
            md5sum： Message Digest(信息摘要)，128(一位表示4位)位定长输出
            sha1sum：Secure Hash Aigorithm ，160位定长输出

useradd 用户名
tail -1 /etc/passwd
tail -1 /etc/group
tail -1 /etc/shadow

cd /etc/default
cat /etc/default/adduser 确定添加用户时的默认选项

groupadd

用户管理：
    useradd userdel usermod passwd chsh chfn finger id chage
组管理：
    groupadd groupdel groupmod gpasswd
权限管理：
    chwon chgrp chmod umask

更改ssh登录提示格式：/etc/update-motd.d
//4-01

useradd
    -u UID: 1000
    -g GID: 基本组
    -G GID: 附加组或额外组可以有多个中间用逗号隔开
    -c "注释信息"
    -d /path/to/somedirectory
    useradd -c "Tony Blare" -d /home/blare user4
    -s shell:给用户分配默认shell
    useradd -s /sbin/nologin user5
    useradd -s /bin/tcsh user6
    -m -k   :强制创建家目录 etc/skel
    -M   : /etc/login.dfgs 强制不创建家目录即使默认配置中有家目录
    -r   : 添加系统帐号
/etc/shells:指定当前系统可用的安全shell

userdel :删除用户
  -r : 删除其家目录

id: 查看用户id号
  -n : 显示用户名
  -u : 显示uid
  -g : 显示gid
  -G : 显示额外组
finger username: 查看用户帐号信息

修改用户信息:
usermod:
  -u : 修改id号
  -g : 修改基本组
  -G : 修改附加组 与-a一起使用增加额外附加组 不使用-a则会覆盖以前的额外组
  -c : 修改注释信息
  -d : 与-m一起使用，即创建新的家目录还将以前的家目录文件拷贝过去
  -s : 修改shell
  -l : 修改登录名
  -e : 过期时间
  -f : 非活动时间
  -L : 锁定帐号
  -U : 解锁帐号

chsh : chsh user2 : 修改用户的默认shell

chfn : 修改用户的注释信息
     : chfn user3

密码管理:
passwd [USERNAME]
  --stdin:从标准输入接口接受密码
  echo "radhat" | passwd --stdin user4 管道输入   ubuntu竟然不支持
  -l: 锁定用户帐号 即在/etc/shadow文件中密码的前面加入!!即可禁止登陆
  -u: 解锁用户帐号
  -d: 删除用户密码

pwck: 检查用户帐号完整性

组管理:
创建组: groupadd
  -g: 指定Gid
  -r: 添加系统用户

groupmod: 修改组
  -g : gid
  -n : 组名

groupdel: 删除组

gpasswd : 给组设定密码

newgrp: 登录到一个新组中

练习
1、创建一个用户mandriva，其ID为2002，基本组为distro（组ID为3003），附加组为linux。

```
groupadd -g 3003 distro
groupadd linux
useradd -u 2002 -g distro -G linux mandriva

2、创建一个用户fedora，其全名为Fedora Community，默认shell为tcsh
useradd -c "Fedora Community" -s /bin/tcsh fedora

3、修改mandriva的ID号为4004，基本组为linux，附加组为distro和fedora
usermod -u 4004 -g linux -aG distro,fedora mandriva

4、给fedora加密码，并设定其密码最短使用期限为2天，最长为50天
echo "fedora" | passwd --stdin fedora
passwd -n 2 -x 50 fedora
5、将mandriva的默认shell改为/bin/bash;
chsh mandriva
6、添加系统用户hbase，且不允许其登录系统;
useradd -r hbase -s /sbin/nologin

chage:
  -d：最近一次的修改时间
  -E：过期时间
  -I：非活动时间
  -m：最短使用期限
  -M：最长使用期限
  -w：警告时间
//04-02
权限管理：
r:
w:
x:

三类用户：
u：属主
g：属组
o：其他用户

chown：改变文件属主（只有管理员可以使用此命令）

 -R：修改目录及其内部文件或内部文件的属主
 --reference=/path/to/somefile file：设置为与路径下相同的文件的属主信息
chgrp：修改组名
 -R：递归修改
 --reference=/path/to/somefile file：设置为与路径下相同的文件的属组信息

chown USERNAME:GROUPNAME file
chown USERNAME,GROUPNAME file

chmod：修改文件权限

修改三类用户的权限
  -R:
  --reference=/path/to/somefiel fiel

修改某类用户或某些类用户权限
u，g，0，a
chmod 用户类别=MODE file
chmod u=rwx file
chmod g=rw-
chmod o=rx file
chmod g=r，u=r
chmod g0=w file

修改某类用户某位或某些权限
u，g，o，a
chmod 用户类别+  - -mod file

chmod u-x
chmod u+x，g-x
chmod a+x
chmod +x//三个用户同时操作
chmod u-wx file

练习
```

1、新建一个没有家目录的用户openstack
useradd -M openstack
2、复制/etc/skel为/home/openstack
cp -r /etc/skel /home/openstack
3、改变/home/openstack及其内部文件的属主属组为openstack
chown -R openstack:openstack openstack/
4、/home/openstack及其内部的文件，属组和其他用户没有任何访问权限
、、chmod 700 openstack/ ：一般不能让文件具有执行权限
chmod -R go= openstack

手动加一个用户hive，基本组为hive （5000）附加组为mygroup
nano /etc/group
nano /etc/passwd
nano /etc/shadow
cp -r /etc/skel /home/hive

openssl passwd
whatis passwd
man sslpasswd
openssl passwd -1 -salt

umask:遮罩码
创建文件：666-umask
创建目录：777-umask

root用户umask为022
普通用户umask为002

umask 022 修改当前用户的umask

umask 023
文件：666-023=643
：计算结果默认文件不能带有执行权限，当计算结果有执行权限时自动加一 所以结果为644
目录：777-023=754

站在用户的登录角度来说，SHELL的类型：
登录式shell
  正常通过其他终端登录
  su - USERNAME
  su -l USERNAME

非登录式shell：
  su USERNAME
  图形终端下打开命令窗口
  自动执行的shell脚本

bash的配置文件
全局配置
  /etc/profile , /etc/profile.d/*.sh, /etc/bashrc
个人配置
 ~/.bash_profile , ~/.bashrc

profile类的文件
  设定环境变量
  运行命令或脚本

bashrc类的文件
  设定本地变量
  定义命令别名

登录式shell如何获取配置文件
/etc/profile --> /etc/profile.d/*.sh -->~/.bash_profile --> ~/.bashrc-->/etc/bashrc

非登录式shell如何配置文件
~/.bashrc --> /etc/basrc --> /etc/profile.d/*.sh

bash:脚本解释器

计算机组成

运算器、控制器、CPU
存储器：RAM
输入设备/输出设备

程序：指令和数据

控制器：指令
运算器：
存储器：

地址总线：内存寻址
数据总线：传输数据
控制总线：控制指令

寄存器：cpu暂时存储器

I/O：硬盘，

程序

INPUT设备：

OUTPUT设备

系统设定
    默认输出设备：标准输出 STDOUT 用1表示

    默认输入设备：标准输入 STDIN   用0表示

    标准错误输出：STDER 2
标准输入：键盘
标准输出和错误输出：显示器

I/O重定向：

对于linux：
输出重定向：> 覆盖输出：ls /etc/var > /tmp/var.out cat /etc/fstab > /tmp/var.out
            >>:追加输出，可以保留原文件内容，而且继续保留原文件内容

set
  -C：开启功能 如果文件中初始有数据禁止使用覆盖方式写入

  +C：关闭功能
set -C时强制覆盖输出 则使用 >| 即可

2>:重定向错误输出
2>>:追加输出

实现错误和正确的都定向输出：ls /var > /tmp/var3.out 2> /tmp/err.out

&>:重定向标准输出或错误输出至同一个文件

输入重定向：<
tr 'a-z' 'A-Z' < /etc/fstab

<< ：在此处生成文档

cat << END

cat >> /tmp/myfile.txt << EOF :将此处生成的文件重定向至指定文件

管道：前一个命令的输出，作为后一个命令的输入

```
echo "hello ,world" | tr 'a-z' 'A-Z'
echo "radhat" | passwd --stdin USER
cat /etc/passwd | sort
cut -d : -f1 /etcpasswd | sort
cut -d : -f1 /etc/passwd | sort | tr 'a-z' 'A-Z' >> /tmp/username.txt
```

命令1 | 命令2 | 命令3

tee:即保存又输出至屏幕中

```
echo "hello.world" | tee /tmp/hello.out
```

wc -l /etc/passwd | cut -d / -f1 :管道实现只输出行数
wc -l /etc/passwd | cut -d ' ' -f1 也可以实现
ls -l /usr/bin | head -1
ls /usr/bin/ | wc -l

练习：
1、统计/usr/bin/ 目录下的文件个数
 ls /usr/bin | wc -1
2、取出当前系统上所有用户的shell，要求，每种shell只显示一次，并且按顺序进行显示
 cut -d : -f 7 /etc/passwd | sort -u
3、思考：如何显示/var/log目录下每个文件的内容类型
 file /var/log/*
 file $(ls /var/log)
4、取出/etc/inittab文件的第六行
 head -6 /etc/inittab | tail -1
5、取出/etc/passwd
文件中倒数第9个用户的用户名和shell，显示到屏幕上并将其保存至/tmp/users文件中
 sort -r /etc/passwd | cat -n 先逆序输出然后显示行号
 tail -9 /etc/passwd | head -1 | tee /tmp/users
 tail -9 /etc/passwd | head -1 | cut -d : -f 1,7 | tee /tmp/users
6、显示/etc目录下所有以pa开头的文件，并统计其个数
 ls -d /etc/pa* | wc -l 要显示为目录不然ls会显示目录中的内容
7、不使用文本编辑器，将alias cls=clear 一行内容添加至当前用户的.bashrc文件中
 echo "alias cls=clear" >> .bashrc

文本查找的需要
grep，egrep，fgrep

grep：根据模式，搜索文本，并将符合模式的文本行显示出来
pattern：模式 文本字符和正则表达式的元字符组合而成的匹配条件
grep 'root' /etc/passwd
 -i：忽略字符大小写
 grep --color 'root' /etc/passwd
 -v：反向查找，显示没有被模式匹配到的行
 -o：只显示被模式匹配到的串

*：
？：
[]
[^]:

正则表达式：regular expression
元字符：
.：匹配任意单个字符
[]:匹配指定范围内的任意单个字符
[^]:匹配指定范围外的任意单个字符
    字符集合：[:digit:],[:lower:],[:upper:],[:punct:],[:space:],[:alpha:],[:alnum:]

grep 'r..t' /etc/passwd

字符次数（贪婪模式）：
*：匹配其前面的字符任意次
 a, b, ab, aab, acb, adb, amnb
 a*b :
 a.*b :ab , aab, acb, adb,amnb
.*:任意长度的任意字符

\?：匹配其前面的字符1次或0次

\{m,n\}:匹配其前面的字符至少m次，至多n次
  \{1,\}:至少一次
  \{0,3\}:最多三次

grep 'a\{1,3\}b' test.txt :匹配a至少一次之多三次
grep 'a.\{1,3\}b' test.txt :匹配任意字符至少一次之多三次

位置锁定：
^:锁定行首，此字符后面的任意内容必须出现在行首
$:锁定行尾，此字符前面的任意内容必须出现在行尾
^$：空白行
grep '[[:digit:]]$' /etc/inittab 查找以数字结尾的行

\<或者\b:锚定词首，其后面的任意字符必须作为单词首部出现
\>或者\b:锚定词尾，其前面的任意字符必须作为单词的尾部出现

\<root\>:只寻找root这个单词

grep "root\>" test2.txt

分组：
\(\)
 \(ab\)*:将ab作为一个整体可以出现任意次
 后向引用
 \1:引用第一个左括号以及与之对应的右括号所包括的所有内容
 \2:
 \3:

 grep "\(l..e\).*\1" test3.txt ： 这样实现前后一致才可以匹配
 grep "\([0-9]\).*\1$" /etc/inittab

He love his lover.
She like her liker.
he like his lover

He love his lover.
She like her lover.
He like his liker.
She love her liker.

grep "l..e" test3.txt

grep '\([0-9]\).*\1$' /etc/inittab

l..e

//5-1
正则表达式：
  Basic REGEXP：基本正则表达式
  Extended REGEXP：扩展正则表达式

grep：使用基本正则表达式定义的模式来过滤文本的命令
  -i：忽略字符大小写
  -v：反向搜索
  -o：只显示匹配到的字符串
  --color ：显示颜色
  -E：使用扩展正则表达式
  -A #：grep -A 2 "^core id" /proc/cpuinfo 表示附加后面两行
  -B #：附加显示前面#行
  -C #：上下都#行

扩展的正则表达式：
字符匹配：
与基本相同
.
[]
[^]

次数匹配
*:任意次
?:匹配零次或一次
+:匹配其前面的字符至少一次 等于基本表达式\{1,\}
{m，n} ：次数匹配

锚定字符相同
^:行首
$:行尾

\<:词首
\>:词尾

分组：
():真正意义的分组
\1,\2,\3,...

或者
| :a or b

C|cat :Cat or cat .是 C或cat
grep -E 'C|cat' test4.txt
grep -E "(C|c)at" test4.txt 分组找到Cat或cat
grep -E "^[[:space:]]+" /boot/grub/grub.conf :以至少一个空白字符开头的
grep -E =egrep

找出ifconfig命令结果中的1-255之间的整数

\<([1-9] | [1-9] [0-9] | 1[0-9] [0-9] | 2[0-4][0-9] | 25 [0-5])\>

ifconfig | egrep '\<([1-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>'

\.

上面的是错的下面的是对的。
egrep '\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>'
ifconfig | egrep
'\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>\.\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>\.\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>\.\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>'

ifconfig | egrep -o
'(\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>\.){3}\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>'
ipv4:
5类：A B C D E

A:1-127
B:128-191
C:192-223

只找IP地址：

egrep -o
'\<([1-9]|[1-9][0-9]|1[0-9][0-9]|2[01][0-9]|22[0-3])\>(\.\<([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>){2}\.\<([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\>'

//05-02
grep，egrep，
fgrep：fast快速的grep，不支持正则表达式

shell编程：

编程语言：机器语言，汇编语言，高级语言

静态语言：编译型语言
  强类型（变量）
  关键字：
  事先转换成可执行格式
  c，c++，java，c#

动态语言：解释型语言，on the fly
  边解释边执行
  PHP、SHELL、PYTHON、PERL

面向过程：shell，c
面向对象：java，Python，perl，c++

变量：内存空间，命令

内存：遍址的存储单元

进程：
1+100;
1+1000000

变量：内存空间
1,10000所占去的内存不同
字符型10：16bit
数值10：1010 4bit

变量类型：事先确定数据的存储格式和长度
  字符
  数值
    整形：
    浮点型：11.23
  2013/10/10 , 64bit
  99999：24bit
  真、假
逻辑：1+1>2
逻辑运算：与、或、非、异或
1：真
0：假

1 & 0 = 0
0 & 1 = 0
0 & 0 = 0
1 & 1 = 1

或

非：单目运算符
! 真 = 假
! 假 = 真

异或：

相同为假
不同为真

整形，8bit ：256
0-255,溢出

shell编程：弱类型编程语言

强：变量在使用前，必须事先声明，甚至还需初始化
弱类型：变量用时申明，甚至不区分类型

11+c =

显示转换
隐式转换

变量赋值：var_name=value

  脚本编程

变量名称：
  1、只能包含字母数字和下划线，并且不能数字开头
  2、不应该跟系统中已有的环境变量重名
  3、最好做到见名知意

bash变量类型
   环境变量
   本地变量（局部变量）
   位置变量
   特殊变量

本地变量：整个bash进程
  bash：

（set） NAME=jerry
echo $(NAME)
声明局部变量
local VARNAME=VALUE：作用域为当前代码段

环境变量：作用域为当前shell进程及其子进程
export VARNAME=VALUE
"导出"

脚本在执行时会启动一个shell进程
命令行中启动的脚本会继承当前shell环境变量
系统自动执行的脚本（非命令行启动）就需要自我定义需要各环境变量

位置变量：
$1,$2,...

特殊变量：
$?:上一个命令的执行状态返回值

程序执行，可能有两类返回值
程序执行结果
程序状态返回代码
0：正确执行
1-255：返回错误类型 1， 2, 127 系统预留

输出重定向
>
>>
2>
2>>
&>

/dev/null :软件设备 ，bit bucket位桶，数据黑洞

撤销变量：
unset VARNAME

查看当前shell中的所有变量：set

查看当前shell中的环境变量：
printenv
env
export

定义一堆字符串
ANIMALS=pig
ANIMALS=$ANIMALS:goar

export PATH=$PATH:/usr/local/apache/bin 一种追加赋值的方法

shell默认定义为字符串变量
A=2
B=3
C=$A+$B
echo $C
显示结果为2+3

引用变量：${VARNAME},括号有时可省略，
""：弱引用，用""时显示的内部变量不会被引用
''：强引用，''里面的东西直接被引用

脚本：命令的堆砌，按实际需要，结合命令流程控制机制实现的源程序

shebang：魔数
#! /bin/bash
注释行，不执行

练习：写一个脚本完成一下任务
1、添加5个用户，user1,....user5
2、每个用户的密码同用户名，而且要求，添加密码完成后不显示passwd命令的执行结果信息
3、每个用户添加完成后，都要显示用户某某已经添加成功

useradd user1

```
echo 'user1' | passwd --stdin user1 > /dev/null
echo "user1添加成功"
useradd user2
echo 'user2' | passwd --stdin user2 > /dev/null
echo "user2添加成功"
useradd user3
echo 'user3' | passwd --stdin user3 > /dev/null
echo "user3添加成功"
```

练习：写一个脚本，完成一下任务
1、使用一个变量保存一个用户名
2、删除此变量中的用户，并且一并删除其家目录：
3、显示”用户删除完成“类的信息

```
NAME=user1
userdel -r $NAME
echo "用户$NAME删除成功"

NAME=user2
userdel -r $NAME
echo "用户$NAME删除成功"

NAME=user3
userdel -r $NAME
echo "用户$NAME删除成功"
```

//5-3
条件判断：
  如果用户不存在
      添加用户，给密码并显示添加成功
  否则
      显示如果已经存在，没有添加

如果用户存在，就显示用于已存在，否则，就添加用户

```
id user1 &> /dev/null && echo "user1 is a user" || useradd user1
```

如果用户不存在，就添加，否则显示其已存在

```
! id user1 &> /dev/null && useradd user1 || echo "user1 exists"
```

如果用户不存在，就添加并给密码：否则，显示其已经存在

```
! id user1 &> /dev/null && useradd user1 && echo "user1" | passwd --stdin user1 ||
echo "user1 exists"
```

bash中如何实现条件判断？
条件测试类型：
    整数测试
    字符测试
    文件测试

条件测试的表达式：

```
    [ expression ]
    [[ expression ]]
    test expression
```

整数比较：
  -eq：测试两个整数是否相等：比如[$A -eq $B]
  -ne：测试两个整数是否不等，不等为真，相等为假
  -gt：测试一个数是否大于另一个数：大于为真，否则为假
  -lt：测试一个数是否小于另一个数：小于为真，否则为假
  -ge：大于或等于
  -le：小于或等于

命令间的逻辑关系：
  逻辑与：&&
     第一个条件为假时，第二个条件不再判断，最终结果已经确定
     第一个条件为真时，第二个条件必须得判断
  逻辑或：||
     第一个为假时，
  ！：取反

```
id user1 &> /dev/null && echo "hello.student"
```

如果用户不存在，就添加用户user6

```
! id user6 && useradd user6
```

```
id user6 || useradd user6

如果/etc/inittab文件的行数大于100，就显示好大的文件
[ `wc -l /etc/inittab | cut -d ' ' -f 1` -gt 100 ] && echo "It is a big file" ||
echo "It is a small file"

变量名称：
    1、只能包含字母，数字和下划线，并且不能以数字开头；
    2、不应该跟系统中已有的环境变量重名；
    3、最好做到见名知意；

如果用户存在，就显示用户已经存在;否则，就添加此用户；
id zhangshuo &> /dev/null && echo "zhangshuo is exisit" || useradd zhangshuo

如果用户不存在，就添加;否则，显示其已经存在
!  id zhangshuo &> /dev/null && useradd zhangshuo || echo "zhangshuo exists"

如果用户不存在，添加并且给密码，否则，显示其已经存在
! id zhangshuo &> /dev/null && useradd zhangshuo && echo "zhangshuo" | passwd
--stdin zhangshuo || echo "zhangshuo exists"

练习，写一个脚本，完成一下要求；
1、添加3个用户user1 ，user2,user3，但要先判断用户是否存在，不存在而后再添加；
2、添加完成后，显示一共添加了几个用户;当然，不能包括因为实事先存在而没有添加的;
3、最后显示当前系统一共有多少个用户

练习，写一个脚本，完成以下要求：
给定一个用户：
  1、如果其UID为0,就显示此为管理员
  2、否则，就显示其为普通用户；

[ $(id -u root) -eq 0 ] && echo "此用户为管理员" || echo "该用户为普通用户
"

如果UID为0那么
  显示为管理员
否则
  显示普通用户

NAME=user16
USERID=$(id -u $NAME)
if [ $USERID -eq 0 ];then
  echo "Admin"
else
  echo "common user"
fi




练习：写一个脚本完成一下任务
1、使用一个变量保存一个用户名
2、删除此变量中的用户，并一并删除其家目录
3、显示"用户删除完成"类的信息

条件判断，控制结构

单分支的if语句
if 判断条件; then
  statement1
  statement2
  ...
fi

双分支的if语句：
if 判断条件; then
  statement1
  statement1

else
  statement3

fi
```

如果用户存在则显示用户存在

```
#!/bin/bash
NAME=user1
if id $NAME &> /dev/null;then
   echo "$NAME 用户存在"
else
   echo "$NAME 用户不存在"
fi
```

如果用户存在显示用户存在，用户不存在则建立用户并命名密码

```
if id $NAME &> /dev/null ;then
   echo "$NAME 用户存在"
else
   useradd user1
   echo "user1" | passwd --stdin user1
   echo "$NAME 用户不存在，所以建立了此用户并赋予与用户名相同的密码"
fi
```

练习：写一个脚本
判断当前系统上是否有用户的默认shell为bash
  如果有，就显示有多少个这类用户;否则，就显示没有这类用户

```
#!/bin/bash
USERBASHNUM=`grep "\<bash\>" /etc/passwd | wc -l`
if [ $USERBASHNUM -eq 0 ];then
    echo "no such user"
else
    echo "$USERBASHNUM users"
fi
```

练习：写一个脚本
给定一个文件，比如/etc/inittab
判断这个文件中是否有空白行
如果有，则显示空白行数否则，就显示没有空白行

```
#!/bin/bash
#by zhangshuo 2018/01/15
FILENAME=/etc/inittab
grep "^$" $FILENAME &> /dev/null
RES=$?
if [ $RES -eq 0 ];then
    BACKNUM=`grep "^$" $FILENAME | wc -l`
    echo "$BACKNUM backline in $FILENAME"
else
    echo "no back line"
fi
```

练习：写一个脚本
给定一个用户，判断其UID与GID是否一样
如果一样，就显示此用户为“GOOD GUY”：否则，就显示此用户为“BAD GUY”

```
#!/bin/bash
#by zhangshuo 2018/01/15
USERNAME=zhangshuo
USER_UID=`id -u $USERNAME`
USER_GID=`id -g $USERNAME`
if [ $USER_UID -eq $USER-GID ];then
    echo "good guy"
else
    echo "bad guy"
fi
```

进一步要求，不使用id命令获得其id号

练习：写一个脚本
给定一个用户，获取其密码的警告期限：
而后判断用户最近一次修改密码时间距离今天是否已经小于警告期限
  提示：算数运算的方法$[$A-$B]:表示变量A的值减去变量B的值的结果
如果小于，则显示“warning”：否则，就显示“OK”

```
#!/bin/bash
```

```
#
USERNAME=zhangshuo
DATEWARING=`grep "$USERNAME" /etc/shadow | cut -d : -f 5`

DATENOW=$(echo "`date +%s`/86400" | bc)

DATECHAGE=`grep "$USERNAME" /etc/shadow | cut -d : -f 3`

let DATEERR=$DATENOW-$DATECHAGE

if [ $DATEERR -gt $DATEWARING ];then
        echo "warning"
else
        echo "ok"
fi
```

练习：写一个脚本
判定命令历史中命令的总条目是否大于1000;如果大于，就显示“some command will gone”：否则显示“OK”

shell中如何进行算数运算

```
A=3
B=6
```

1、let 算数运算表达式

```
    let C=$A+$B
```

2、$[算数运算表达式]

```
    C=$[$A+$B]
```

3、$(($A+$B))

4、expr 算数运算表达式，表达式中各操作数及运算符之间要有空客，而且要使用命令引用

```
   C=$(expr $A + $B)
   C=`expr $A + $B`

A=3
B=6
let C=$A+$B

//06-01
exit:退出脚本
exit #
```

如果脚本没有明确定义退出状态码，那么，最后执行的一条命令的退出码即为脚本的退出状态码

测试方法：

```
[ expression ]
[[            ]]
test expression
```

bash中常用的测试条件有三种

整数测试

```
  -gt
  -le
  -ne
  -eq
  -ge
  -lt

INT1=63
INT2=77
[ $INT1 -eq $INT2 ]
[[ $INT1 -eq $INT2 ]]
test $INT1 -eq $INT2

if [ grep "^$USERNAME\>" /etc/passwd ];then
```

此条件不要使用中括号

文件测试：

-e FILE: 测试文件是否存在
-f FILE: 测试文件是否为普通文件
-d FILE: 测试指定路径是否为目录
-r FILE: 测试当前用户为指定文件是否有读权限
-w FILE: 测试当前用户为指定文件是否有写权限

-x FILE：测试当前用户为指定文件是否有执行权限

测试脚本是否有语法错误：
bash -n 脚本好像不好用
bash -x 显示每一步执行的程序并显示出来

[-e /etc/inittab]:测试inittab文件是否存在
[-x /etc/rc.d/rc.sysinit]:测试指定文件是否有执行权限

```
FILE=/etc/inittab
if [ ! -e $FILE ];then
  echo "NO $FILE"
  exit 8
  ：这个数字是自己定义的退出码，如果没有定义退出码，则最后执行的一条命令的退出码即为脚本的退出状态码

fi
```

知识补充：多分支if语句

```
if 判断条件1;then
  state
elif 判断条件2;then
  state
elif 判断条件;then
  state
else
  state
fi
```

练习：写一个脚本
给定一个文件：
如果是一个普用文件，就显示之
如果是一个目录，亦显示之
否则，此为无法识别之文件

```
#!/bin/bash
#by zhangshuo 2018/01/15

FILENAME=/etc/inittab

if [ -e $FILENAME ];then
        if [ -f $FILENAME ];then
                echo "it is a common file"
        elif [ -d $FILENAME ];then
                echo "it is a dir"
        else
                echo "cat recgnize"
        fi
else
        echo "no such file"
        exit 4
fi
```

bash变量的类型：
  本地变量（局部变量）
  环境变量
  位置变量 ： $1,$2...
             shift n
  特殊变量

```
./filetest.sh /etc/fatab /etc/inittab
$1:/etc/fatab
$2:/etc/inittab
```

练习：写一个脚本
能接受一个参数{文件路径}
判定：此参数如果是一个存在的文件;并显示其文件类型，否则就显示"no such file "

```
#!/bin/bash
```

```
#by zhangshuo 20180115

if [ $# -eq 0 ];then
    echo "usage !!!!!!!!!!!"
    exit 2
fi

if [ -e $1 ];then
    if [ -f $1 ];then
        echo "common file"
    elif [ -d $1 ];then
        echo "dir"
    else
        echo "unkonw"
    fi
else
    echo "no such file"
    exit 3
fi


特殊变量：
    $?：返回码
    $#:参数的个数
    $*:参数列表
    $@:参数列表

练习：写一个脚本
给脚本传递两个参数{整数};
显示此两者之和。之积

#!/bin/bash
#by zhangshuo 20180115
if [ $# -lt 2 ];then
    echo "usage!!!!!!!!!!!"
    exit 2
fi

let SUM=$1+$2
let SQR=$1*$2

echo "SUM is $SUM"
echo "SQR is $SQR"

//06-02
处理文本：工具
grep，sed（数据流编辑器），awk

sed基本用法：
sed ：stream editer
  行编辑器（全屏编辑器：vi）

sed：模式空间
默认不编辑原文件，仅对模式空间中的数据做处理

sed [options] 'AddressCommand' file ...
  -n:静默模式 不显示模式空间中的内容 不然匹配到的行会显示两次
  -i: 直接修改原文件
  -e SCRIPT -e SCRIPT...:可以同时执行多个脚本
  -f /PATH/TO/SED_SCRIPT
     sed -f /path/to/scripts file :挨个处理脚本中的操作
  -r ： 使用扩展正则表达式

有三种方式可以指定命令行上的多重指令：
1、用分号分隔指令
    sed 's/ MA/ ,Massachusetts/;s/ PA/, Pennsylvania/' list
2、在每个指令前放置-e
    sed -e 's/ MA/ ,Massachusetts/' -e 's/ PA/, Pennsylvania/' list
3、使用bash的分行指令功能。在输入单引号后按RETURN键，就会出现多行输入的提示符(>).
    sed '
    s/ MA/, Massachusetts/
    s/ PA/, Pennsylvania/
    s/ CA/, California/' list
```

Address:
1.StartLine,EndLine
比如1,100
$:最后一行
$-1：倒数第二行
2./RegExp/
/^root/
3./pattren1/,/pattern2/:表示被第一次模式匹配到的行到第二次模式匹配到的行中间的所有行
4.LineNumber :表示精确某行
5.StartLine,+N
从startLine开始，向后的N行

command:
d：删除符合条件的行
```
sed '1,2d' /etc/fstab
sed '/oot/d' /etc/fstab
sed '1,+2d' /etc/fstab
```
p:显示符合条件的行
```
sed '/^\//d' /file :删除以斜线开头的行
sed -n '/^\//p' /file :只显示匹配到的行
```
a \string:在指定的行后面追加新行，内容为"string"
```
sed '/^\#/a \# hello world' /etc/fstab :
```
i \string :在指定的行前面添加新行，内容为string
r file :将指定的文件的内容添加至符合条件的行处
```
sed '2r /etc/issue' /etc/fstab
sed '$r /etc/issue' /etc/fstab
sed '1,2r /etc/issue' /etc/fstab
```
w file:将地址指定范围内的内容另存至符合条件的行处
```
sed '/oot/w /tmp/oot.txt' /etc/passwd
```
s/pattern/string/:查找并替换 :默认每行中第一次被模式匹配到的串
```
sed 's/oot/OOT/' /etc/passwd
```
修饰符
g：全局替换
```
sed 's/oot/OOT/g' /etc/passwd
sed 's/^\//#/g' /etc/passwd
```
i:忽略字符大小写
```
s///:s###:s@@@:s!!!
```

后向引用
```
\(\),\1,\2
```

```
l..e:like-->liker
     love-->lover
```

&:引用模式匹配整个串

```
sed 's#l..e#&r#' sed.txt
sed 's#\(l..e\)#\1r#' sed.txt :后向引用
sed 's#l\(..e\)#L\1#' file :引用一部分

history | sed 's#[[:space:]]##g'
history | sed 's/^[[:space:]]*//' | cut -d ' ' -f 1
```

sed练习：
1、删除/etc/grub.conf文件中行首的空白字符
```
sed 's#^[[:space:]]*##g' sed.txt
sed -r 's#^[[:space:]]+##g' sed.txt
```
2、替换/etc/inittab文件中"id:3:initdefault:"一行中的数字为5
```
sed 's#\(id:\)[0-9]\(:initdefault\)#\15\2#g' /etc/inittab
```
3、删除/etc/inittab文件中的空白行
```
sed '/^$/d' sed.txt
```

删除/etc/inittab文件中开头的#号
```
sed 's@^#@@g' /etc/inittab
```

4、删除/etc/inittab文件中开头的#号，但要求#号后面必须有空白字符
```
sed 's@^#[[:space:]]\+@@g' /etc/inittab
```

5、删除某文件中开头的#号及后面的空白字符，但要求#号后面必须有空白字符

6、删除某文件中以空白字符后面跟#类的行中的开头的空白字符及#

```
   sed 's@^[[:space:]]\+#@@g' /etc/inittab
```

7、取出一个文件路径的目录名称

```
   echo "/etc/rc.d" | sed -r 's@^(/.*/)[^/]+/?@\1@g'
   取出文件的基名
   echo "/etc/rc.d" | sed -r 's@^/.*/([^/]+/?)@\1@g'

#abc
 # hello world
    # hi world
```

练习：
传递一个用户名参数给脚本，判断此用户的用户名跟其基本组的组名是否一致，并将结果显示出来。

```
#!/bin/bash
#by zhangshuo 20180116

if [ $# -eq 0 ];then
    echo "usage!!!!"
    exit 2
fi

USERUID=`id -un $1`
USERGID=`id -gn $1`

if [ $USERUID == $USERGID ];then
    echo "same"
else
    echo "dont same"
```

fi字符测试：
==：相等为真不等为假（等号两段必须有空格）
!=：不等为真，等则为假
>
<
-s string:测试字符串是否为空，空则真，不空则假
-n string:测试指定字符串是否不空，不空为真，空则为假

写一个脚本
传递一个参数（单字符串就行）给脚本，如果参数为q、Q、quit或Quit,就退出脚本；否则，就显示用户的参数：

```
#!/bin/bash
#by zhangshuo 20180116

if [ $# -eq 0 ];then
    id
    exit 2
fi

if echo $1 | egrep '\<(q|Q|quit|Quit)\>' &> /dev/null ;then
    exit 2
else
    echo $1
fi

if [ $1 == 'q' -o $1 == 'Q' -o $1 == 'quit' -o $1 == 'Quit' ];then
    exit 2
else
    echo $1
fi
```

练习：
传递三个参数给脚本，第一个为整数，第二个为算数运算符，第三个为整数，将计算结果显示出来，
要求保留两位精度。形如： ./calc.sh 5/2

```
echo $(echo "scale=2;$1$2$3" | bc)
bc <<< "scale=2;5/2"

#!/bin/bash
#by zhangshuo 20180116

echo "使用*时需要使用逃逸字符即\*"
```

```
echo `echo "scale=2;$1$2$3" | bc`


练习：(line1491.sh)
传递三个参数给脚本，参数均为用户名，将此些用户的账号信息提取出来后放置于/tmp/testusers.txt文件中，并要求
每一行行首有行号。

写一个脚本：(line1495.sh)
判断当前主机的CPU生产商，其信息在/proc/cpuinfo文件中vendor_id 一行中。
如果其生产商为AUTHENTICamd,就显示其为AMD公司；
如果其生产商为GenuineInter，就显示其为Inter公司；
否则，就说其为非主流公司；

写一个脚本：(line1501.sh)
给脚本传递三个参数，判断其中的最大数和最小数，并显示出来。

循环：进入条件，退出条件
for
while
until

for 变量 in 列表 ;do
    循环体
done

遍历完成之后，退出：

如何生成列表：
{1..100}//整数列表

seq 5 10
seq 10
seq 启示数 步进长度 结束数

1,...,100

declare -i SUM=0

for I in $(seq 1 1 100);do
    let SUM=$[$SUM+$I]
done

echo "the sum is :$SUM"

#!/bin/bash
#by zhangshuo 20180116

declare -i SUM=0
for I in `seq 1 1 100`;do
    let SUM+=$I
done

echo $SUM

写一个脚本：(line1532.sh)
1、设定变量FILE的值为/etc/passwd
2、依次向/etc/passwd中的每个用户问好，并显示对方的shell，型如：
    Hello，root，your shell : /bin/bash
3、统计一共有多少个用户

#!/bin/bash
#by zhangshuo 20180116

declare -i NUM=0

for I in `cat /etc/passwd | cut -d : -f 1`;do
    #echo "helo $I,your shell :`grep "$I" /etc/passwd | cut -d : -f 7`"
    USERSHELLS=$(grep "^\<$I:" /etc/passwd | cut -d : -f 7)
    if [ $USERSHELLS == '/bin/bash' ];then
        echo "helo $I,your shell : $USERSHELLS"
        let NUM+=1
    fi
```

```
done

echo $NUM
```

只向默认shell为bash的用户问好

写一个脚本：(line1540.sh)
1、添加10个用户user1到user10，密码同用户名 但是要求只有用户不存在的条件下才能添加

扩展：
接受一个参数：
add：添加用户user1..user10
del：删除用户user1..user10
其他：退出　　再继续扩展：--add user1,user2,user3,hello,hi(line1547.sh)

```
#!/bin/bash
#by zhangshuo 20180116

if [ $# -eq 0 ];then
    echo "usage!!!!"
    exit 2
fi

if [ $1 == '--add' ];then
    for I in `seq 1 1 10`;do
        if ! id user$I &> /dev/null;then
            echo "add user$I"
            useradd user$I
            echo "user$I" | passwd --stdin user$I &> /dev/null
        else
            echo "user$I exisit"
        fi
done
elif [ $1 == '--del' ];then
    for I in {1..10};do
        if id user$I &> /dev/null;then
            echo "del user$I"
            userdel -r user$I
        else
            echo "no user to del"
        fi
    done
else
    exit 0
fi

#!/bin/bash
#by zhangshuo 20180117

if [ $# -lt 2 ];then
        echo "usage!!!!!!"
        exit 2
fi

if [ $1 == '--add' ];then
        echo "add users"
        for I in `echo $2 | sed 's@,@ @g'`;do
                if id $I &> /dev/null;then
                        echo "user $I exisit"
                else
                        echo "add user $I done"
                fi
        done
fi
```

写一个脚本：(line1549.sh)
计算100以内所有能被3整除正整数的和
取模，取余 %
3%2=1
100%55=45

写一个脚本：(line1555.sh)
计算100以内所有奇数的和以及所有偶数的和，分别显示之：

```
#by zhangshuo 20180116

declare -i JSUM=0
declare -i OSUM=0

for I in `seq 1 100`;do
    let MID=$I%2
    if [ $MID -eq 0 ];then
        let OSUM+=$I
    else
        let JSUM+=$I
    fi

done

echo "偶数的和为：$OSUM"
echo "素数的和为：$JSUM"

let I=$[$I+1] 相当于 let I++
SUM=$[$SUM+1]

let SUM+=$I

-=
/-
%=
```

写一个脚本：(line1558.sh)
分别显示当前系统上所有默认shell为bash的用户和默认shell为/sbin/nologin的用户，并统计各类shell下
的用户总数。显示结果形如：
BASH,3users，they are:
root，redhat，gentoo
NOLOGIN,2users，they are:
bin，ftp

```
declare -i SUM=0  //声明变量类型
    -i ：整型
    -x ：声明为环境变量
```

测试：
整数测试
```
        -le
        -lt
        -ge
        -gt
        -eq
```
字符串测试
```
        ==
        !=
        >
        <
        -n
        -z
```
文件测试
```
        -e
        -f
        -d
        -r
        -w
        -x
```
组合测试条件
```
        -a:与关系&&
        -o:或关系||
        ! :非关系
```

```
1<$#<=3
```
[$# -gt 1 -a $# -le 3] 或 [$# -gt 1] && [$# -le 3] 写法

vim编辑器

文本编辑器，字处理器
ASCII

nano ，sed，

vi：Visual Interface 可视化接口
vim：vi improved

全屏编辑器，模式化编辑器

vim模式：
编辑模式（命令模式）
输入模式
末行模式

默认处于编辑模式

模式转换：
编辑模式-->输入模式：
  i：在当前光标所在字符的前面，转为输入模式
  a：在当前光标所在字符的后面，转为输入模式
  o：在当前光标所在行的下方，新建一行，并转为输入模式

  I：在当前光标所在行的行首，转换为输入模式
  A：在当前光标所在行的行尾，转换为输入模式
  O：在当前光标所在行的上方，新建一行，并转为输入模式

输入模式-->编辑模式
  ESC

编辑模式-->末行模式：
  :
  10d：删除第十行
  10,20d：删除第十到第二十行

末行模式-->编辑模式
  ESC，ESC

一、打开文件
vim /path/to/somefile
  vim +# file ：光标直接显示到#行
  vim + file：光标处在最后一行上
  vim +/PATTERN ：打开文件，定位至第一次被PATTERN匹配到的行的行首

二、关闭文件
1、末行模式下关闭文件
  w：保存
  q：退出
  wq：保存并退出
  q!：不保存并退出
  w!：强行保存（只有管理员才可以）
  wq --> ：x 相同的功能
2、编辑模式下退出
ZZ：保存退出

三、移动光标（编辑模式）
1、逐字符自动
  h：左
  l：右
  j：下
  k：上
  #h：移动#个字符

2、按单词为单位移动
  w：移至下一个单词的词首
  e：跳至当前或下一个单词的词尾
  b：跳至当前或前一个单词的词尾

#w:

3、行内跳转：
0：绝对行首
^：行首的第一个空白字符
$：绝对行尾

4、行间跳转
#G：跳转至第#行
G：跳至结尾

末行模式下，直接给出行号即可实现行间跳转

四、翻屏操作
Ctrl+f：向下翻一屏
Ctrl+b：向上翻一屏

Ctul+d：向下翻半屏
Ctul+u：向上翻半屏

五、删除单个字符
x：删除光标所在处的单个字符
#x：删除光标所在处及向后的共#个字符

六、删除命令：d
d命令跟行内跳转命令组合使用
#d跳转字符：5dw：删除五个单词 d0：删除至行首 d$：删除至行尾

dd：删除当前光标所在行
#dd：删除包括当前光标所在内的#行：

末行模式下：
StartADD，EndADDd：删除多行
.：表示当前行
$：最后一行
+#:相对表示法
$-#:到最后一行之前多少行

七、粘贴命令 p
P：如果删除或复制为整行内容，则粘贴至光标所在行的下方，如果复制或删除的内容为非整行，则粘贴至光标所在字符的后面
p：如果删除或复制为整行内容，则粘贴至光标所在行的上方，如果复制或删除的内容为非整行，则粘贴至光标所在字符的下面

八、复制命令 y
用法同d命令

九、修改：先删除内容，在转换为输入模式
c：用法同d

十、替换 ：r替换单个字符
R：替换模式

十一、撤销编辑操作 u
u：撤销前一次的操作 可以连续使用u 最多可以撤销50次操作

#u：直接撤销最近#次编辑

撤销最近一次撤销：Crtl+r

十二、重复前一次编辑操作
.

十三、可视化模式
v：按字符选取
V：按矩形选取

十四、查找
/PATTERR
?PATTERR
n
N

十五、查找并替换
在末行模式下使用s命令
ADDR1,ADDR2s@PATTER@STRING@gi
.,$-1s/he/HE/g

%:表示全文

练习：将/etc/yum.repos.d/server.repo文件中的ftp://instructor.example.com/pub替换为http://172.16.0.1/yum

%s/ftp:\/\/instructor\.example\.com\/pub/http:\/\/172.16.0.1\/yum/g
%s@ftp://instructor.example.com/pub@http://172.16.0.1/yum@gi

十六、如何使用vim编辑多个文件
vim FILE1 FILE2 FILE3
: next切换至下一个文件
: prev切换至前一个文件
: last切换至最后一个文件
: first切换至第一个文件

: qa 全部退出

十七、分屏显示一个文件
Ctrl+w，s：水平拆分窗口
Ctrl+w，v：垂直拆分窗口
Ctrl+w，ARROW：窗口间切换

: qa 关闭所有窗口

十八、分窗口显示多个文件
vim -o ：水平分割
vim -O：垂直分割显示

十九、将当前文件中的部分内容另存为零为一个文件
末行模式下使用w命令
: w
: ADDR1,ADDR2w /path/to/somewhere

二十、将另外一个文件的内容填充在当前文件中
: r /path/to/somefile

二十一、跟shell交互
: ! COMMAND

二十二、高级话题
1、显示或取消显示行号
: set number
: set nu

: set nonu

2、显示或忽略区分字符大小写
: set ignorecase
: set ic

: set noignorecase
: set noic

3、设定自动缩进
: set autoindent
: set ai
: set noai

4、查找到的文本高亮显示或取消
: set hlsearch
: set nohlsearch

5、语法高亮
: syntax on
: syntax off

二十三、配置文件
/etc/vimrc
~/.vimrc

Vimtutor

//07-03
文件查找：
locate:
    非实时，查找是根据全系统文件数据库进行的；
    locate passwd
updatedb,手动生成文件数据库
优点 速度快

find：
    实时
    精确
    支持众多查找标准
    遍历指定目录中的所有文件完成查找，速度慢；

find 查找路径 查找标准 查找到以后的处理动作
查找路径：默认为当前目录
查找标准：默认为制定路径下的所有文件
处理动作：默认为显示

匹配标准：
    -name 'FILENAME' 对文件名做精确匹配
    # find /etc -name 'passwd'
        文件名通配
            *：任意长度的任意字符
            ？：任意单个字符
            []:匹配范围内的字符
    -iname 'FILENAME' 文件名匹配时不区分大小写
    -regex PATTERN 基于正则表达式进行文件名匹配

    -user USERNAME :根据文件的属主查找文件
    # find /etc -user zhangshuo
    -group GROUPNAME:根据组查找

    -uid UID:根据UID查找
    -gid GID:根据GID查找

    -nouser：查找没有属主的文件
    -nogroup:查找没有属组的文件

    -type 文件类型查找
        f:普通文件
        d:目录文件
        c:字符设备
        b:块设备
        l:符号链接
        p:管道设备
        s:套接字设备
    # find /etc -type d
    # find /tmp -type s

    -size 文件大小查找
        [+|-]#k:
        [+|-]#M:
        [+|-]#G:
    find /etc -size 10k -ls

组合条件：默认为&&条件
    -a:&&
    -o:||
    -not:!
    # find /tmp -nouser -type d
    # find /tmp -not -type d //表示查找非目录的文件
查找/tmp目录下，不是目录，并且还不能是套接字类型的文件
    #find /tmp -not -type d -a -not -type s
    #find /tmp -not \(-type d -o -type s\)

查找/tmp/test目录下，属主不是user1，也不是user2的文件
```
    find /tmp/test -not \( -user user1 -o -user user2 \)

    -mtime:修改时间(天)
    -ctime:改变时间
    -atime:访问时间
        [+|-]#
        # find /tmp -mtime +5 -a -type -f -a -nouser

    -mmin:修改时间(分钟)
    -cmin:改变时间
    -amin:访问时间
        [+|-]#
```
查找/tmp目录下，一周之前访问的文件
```
# find /etc -atime +7

    -perm MODE:精确匹配给定权限
          /MODE：只要包含一个相同的权限即可
          -MODE：文件权限要完全包含给定权限时才匹配
    find ./ -prem -001
```
动作：
```
    -print:显示
    -ls:类似ls -l 的形式显示每一个文件的详细信息
    -ok COMMAND {} \; 每次操作都需要用户确认
    -exec COMMAND {} \;

    find ./ -prem -006 -exec chmod o-x {} \;
    find ./ -type d -exec chmod a+x {} \;
```
查找组有读权限的文件并改名为~.new
```
# find ./ -perm -020 -exec mv {} {}.new \;
```
找到目录下所有文件名为.sh结尾的并将其他用户的执行权限去掉
```
# find ./ -name '*.sh' -a -perm -111 -exec chmod o-x {} \;
```
练习：
1、查找/var目录下属主为root并且属组为mail的所有文件
```
# find /var -user root -a -group mail
```
2、查找/usr目录下不属于root，bin，或student的文件
```
# find /usr -not \( -user root -a -user bin -a -user zhangshuo \)
```
3、查找/etc目录下最近一周内内容修改过且不属于root及student用户的文件
```
# find /etc -atime -7 -a -not -user root -a -not -user zhangshuo -ls
```
4、查找当前系统上没有属主或属组且最近一天内曾被访问过的文件，并将其属主属组都写改为root
```
# find / -nouser -a -nogroup -a -atime -7 -exec chown root:root {} \;
```
5、查找/etc目录下大于1M的文件，并将其文件名写入/tmp/etc.largefiles文件中
```
# find /etc -size +1M -exec ls {} > /tmp/etc.largefiles \;
```
6、查找/etc目录下所有用户都没有写权限的文件，显示出其详细信息
```
# find /etc  -not -perm /222 -ls
```

特殊权限
passwd;s

SUID:运行某程序时，相应进程的属主是程序文件自身的属主，而不是启动者；
```
    chmod u+s FILE
    chmod u-s FILE
        如果FILE本身原来就有执行权限，则SUID显示为s,否则显示为S；
```
SGID:运行某程序时，相应进程的属组是程序文件自身的属组，而不是启动者所属的基本组；
```
    chmod g+s FILE
    chmod g-s FILE
```
sticky:在一个公共目录，每个都可以创建文件，删除自己的文件，但不能删除别人的文件。
```
    chmod o+t DIR
    chmod o-t DIR
```

000
001
...
111

chmod 2755 /backup/test 在以前的基础上右多了一个位

```
umask 0022


文件系统访问列表
tom
    tom,tom基本组
JERRY:other: rw-

普通用户没有权限执行chmod

FACL:Filesystem Access Control List
利用文件扩展保存额外的访问控制权限

加入FACL之后文件的访问顺序:
    Owner -> facl.user -> Group -> facl,group -> Other

setfacl
    -m: 设定
        u:UID:perm
        d:u:UID:prem 给目录创建默认的访问列表
        g:GID:perm
        d:g:UID:prem
    -x: 取消
        u:UID
        g:GID

getfacl

setfacl -m u:hadoop:rw /path/to/some/file
setfacl -m g:mygroup:rw /path/to/some/file



关于几个命令

whoami 我是谁

who //显示当前系统登陆的用户有哪些
    -r:显示用户级别

写一个脚本:
每隔5秒钟，就来查看hadoop是否已经登陆，如果登陆，显示已经登陆，并退出;
sleep 睡眠

w: 显示当前登录的用户并显示其动作

last: 显示登陆日志其实为 显示/var/log/wtmp文件 显示用户登陆历史及系统重启历史
    -n #:显示最近#次的相关信息
lastb, 显示/var/log/btmp,显示用户错误的登陆尝试
    -n #:

lastlog:显示每一个用户最近一次成功登陆信息
    -u USERNAME:显示特定用户最近的登陆信息

basename /etc/abc/me
    $0:执行脚本时的脚本路径及名称

mail 邮件服务

hostname:显示或更改主机名
如果当前主机的主机名不是www.magedu.com,就将其改为www.magedu.com

[ $(hostname) != 'www.magedu.com' ] && hostname www.magedu.com

如果当前主机的主机名是localhost,就将其改为www.magedu.com
[ $(hostname) == 'www.magedu.com' ] && hostname www,magedu.com

如果当前主机的主机名为空，或者为(none),或者为localhost,就将其改为www.magedu.com
[ -z $(hostname) ] || [ $(hostname) == '(none)' -o $(hostname) == 'localhost' ] &&
hostname www.magedu.com

生成随机数
RANDOM:0-32768
```

随机数生成器：熵池
/dev/random
/dev/urandom

写一个脚本：(line2031.sh)(random.sh)
利用RANDOM生成10个随机数，并找出其最大值（排序）

```
#!/bin/bash
#by zhangshuo 20180118

MAXNUM=0
MINNUM=32768

for I in {1..10};do
    RANDOMNUM=$RANDOMNUM:`echo $RANDOM`
done

echo $RANDOMNUM

for I in `echo $RANDOMNUM | sed 's@:@ @g'`;do
    echo $I
    if [ $I -gt $MAXNUM ];then
        MAXNUM=$I
    elif [ $I -lt $MINNUM ];then
        MINNUM=$I
    fi
done

echo "MAXNUM is $MAXNUM"
echo "MINNUM is $MINNUM"
```

终端类型：
    console：控制台
    pty：物理终端（VGA）
    tty：虚拟终端（VGA）
    ttys：串行终端
    pts/#:伪终端

面向过程
    控制结构
        顺序结构
        选择结构
        循环结构

选择结构：
if 单分支、双分支、多分支
if CONDITION ;then

case语句：选择结构 (case.sh)

```
case SWITCH in
value1)
    statement
    ...
    ;;
value2)
    statement
    ...
    ;;
*)
    statement
    ...
    ;;
esac

#!/bin/bash
#by zhangshuo 20180118
```

```
if [ $# -eq 0 ];then
    echo "usage!!!!!"
    exit 2
fi

case $1 in
[a-z])
    echo "$1 is a char"
    ;;
[0-9])
    echo "$1 is a num"
    ;;
*)
    echo "unknow"
    ;;
esac
```

写一个脚本(line2074.sh)
只接受参数start ,stop,restart,status其中之一并显示之

```
#!/bin/bash
#by zhangshuo 20180118

if [ $# -eq 0 ];then
    echo "usage!!!!!"
    exit 2
fi

case $1 in
'start')
    echo "start"
    ;;
'stop')
    echo "stop"
    ;;
'restart')
    echo "restart"
    ;;
'status')
    echo "status"
    ;;
*)
    echo "usage:`basename $0` {start|stop|restart|status}"
    ;;
esac
```

写一个脚本，可以接受选项及参数，而后能够获取每一个选项，及选项的参数，并能根据选项及参数做出
特定的操作，比如：
adminusers.sh --add tom,jerry --del tom,blair -v | --verbose -h|--help

```
#!/bin/bash
#by zhangshuo 20180118

DEBUG=0
ADDUSER=0
DELUSER=0

for I in `seq $#`;do
if [ $# -gt 0 ];then
    case $1 in
    '-v'|'--verbose')
        DEBUG=1
        shift 1
        ;;
    '--add')
        ADDUSER=1
        ADDUSERS=$2
        shift 2
        ;;
    '--del')
        DELUSER=1
```

```
            DELUSERS=$2
            shift 2
            ;;
        *)
            echo "gg"
            ;;
        esac
    fi
    done

    echo $DEBUG $ADDUSER $DELUSER $ADDUSERS $DELUSERS

    if [ $ADDUSER -eq 1 ];then
        for USER in `echo $ADDUSERS | sed 's@,@ @g'`;do
            if id $USER &> /dev/null;then
                [ $DEBUG -eq 1 ] && echo "user $USER exisit"
            else
                [ $DEBUG -eq 1 ] && echo "add $USER finished"
            fi
        done
    elif [ $DELUSER -eq 1 ];then
        for USER in `echo $DELUSERS | sed 's@,@ @g'`;do
            if id $USER &> /dev/null;then
                [ $DEBUG -eq 1 ] && echo "del $USER finished"
            else
                [ $DEBUG -eq 1 ] && echo "no such user"
            fi
        done
    fi
```

练习：写一个脚本showlogged.sh，其用法格式为：

showlogged.sh -v -c -h| --help

其中，-h选项只能单独使用用于显示帮助信息；-c选项时，显示当前系统上登陆的所有用户数：如果同时使用了-v选项，则既显示同时登陆的用户数，又显示登陆的用户的相关信息：如

```
Logged users: 4.
they are:
Root tty2 feb 18 02:41
```

```
#!/bin/bash
#by zhangshuo 20180118

declare -i SHOWNUM=0
declare -i SHOWUSERS=0

for I in `seq $#`;do
    if [ $# -gt 0 ];then
        case $1 in
        '-h'|'-help')
            echo "usage :`basename $0` -h|--help -c|--count"
            exit 0
            ;;
        '-c'|'--count')
            let SHOWNUM=1
            shift
            ;;
        '-v'|'--verbose')
            let SHOWUSERS=1
            shift
            ;;
        *)
            echo "usage :`basename $0` -h|--help -c|--count"
            exit 2
            ;;
        esac
    fi
done

if [ $SHOWNUM -eq 1 ];then
    echo "Logged users:`who | wc -l`"
fi

if [ $SHOWUSERS -eq 1 ];then
    echo "They are:"
```

```
    who
fi

分区partition
    文件系统
MBR 主引导记录位于零磁道零扇区（512byte）
master boot record
main boot record
    446byte：bootloader，程序，
    64bytes：
        16bytes：标识一个分区
    最后两个字节：
        magic number ： 模数
        标记mbr是否有效

文件系统管理
重新创建文件系统会损坏原有文件

硬链接，符号链接
符号链接权限都为777
ln [-s -v] SRC DEST

硬链接：
    1、只能对文件创建，不能应用于目录
    2、不能跨文件系统
    3、创建硬链接会增加文件被链接的次数。

符号链接：
    1、可应用与目录
    2、可以跨文件系统
    3、不会增加被硬链接的次数
    4、其大小为指定的路径所包含的字符个数

du -s 显示文件夹的大小

df -h 显示分区的使用情况
    -i 显示INODE
    -p 不换行显示

设备文件
    块设备：按块为单位，随机访问的设备
    字符设备：按字符为单位，线性设备

ls -l /dev
    字符设备的两个号码
    第一个为：主设备号
    第二个为：次设备号

mknod ： 创建块设备或字符设备
mknod -m MODE （权限）

使用tty命令可以查看当前使用的客户端类型
echo "hello" >> /dev/pts/6

硬盘设备的设备文件名：
IDE ,ATA ： hd
STAT:sd
SCSI:sd
USB:sd
    a,b,c,....来区别同一种类型下的不同设备

IDE:
    第一个IDE口：主、从
        /dev/hda  /dev/hdb
    第二个IDE口：主、从
        /dev/hdc  /dev/hdd

fdisk -l 查看当前系统分区情况

Linux支持的文件系统
VFS：虚拟文件系统

cat /proc/partitions :查看当前内核已经识别的分区
```

partprobe：通知内核重读分区表

管理磁盘分区
fdisk /dev/sdb
    p:显示当前硬盘的分区情况
    w:保存退出
    n:创建新分区
        e:扩展分区
        p:主分区
    d:删除一个分区
    q:不保存退出
    t:修改分区类型
    l:显示文件系统类型

mkfs:make file system
    -t:FSTYPE:
    ext2
    ext3

mkfs -t ext2 = mkfs.ext2
mkfs -t ext3 = mkfs.ext3

专门管理ext系列文件系统

mke2fs :创建ext2格式文件系统
    -j :创建日志文件系统即ext3
    -b BLOCK_SIZE:指定块大小，默认为4096 可用取值为1024 2048 4096
    -L LABEL：指定分区卷标
    -m #:制定预留给超级用户的块数百分比
    -i
    #:用于指定为多少个字节空间创建一个inode，默认为8192，这里给出的数值应该为块大小的2^n倍
    -N：指定inode个数
    -F：强制创建文件系统
    -E：用户指定额为文件系统属性

cat /proc/filesystems 查看当前内核支持那些文件系统

对于VFAT文件系统的优盘来说
#vi /etc/mtools.conf
在最后一行增加 mtools_skip_check=1。
修改卷标命令为：mlabel -i /dev/sdb1 ::name

blkid：查询或查看磁盘设备的相关属性
    UUID
    TYPE
    LABEL

e2label :用于查看或定义卷标
    e2label:设备文件 卷标：设定卷标

tune2fs:调整文件系统的相关属性
    -j:不损害原有数据，将ext2升级为ext3
    -L LABEL：设定或修改卷标
    -m #:调整预留百分比
    -r #:指定预留块数
    -o:设定默认挂在格式
        acl
    -c #:指定挂在次数达到#次之后进行自检，0或-1表示关闭此功能
    -i #:每挂载使用多少天后进行自检：0或-1表示关闭此功能
    -l :显示超级快中的信息

dumpe2fs:显示文件属性信息
    -h:只显示超级块信息

fsck：检查并修复Linux文件系统
    -t FSTYPE:文件系统类型
    -a :自动修复

e2fsck:专用于修复ext2/ext3文件系统
    -f:强制检查
    -p:自动修复
    -a:自动修复

挂载：将新的文件系统关联至当前文件系统
卸载：将某文件系统与当前根文件系统的关联关系予以移除

mount :挂载
mount 设备 挂载点
    设备文件：/dev/sdb1
        卷标：LABEL=""
        UUID：UUID=""
    挂载点：目录
        要求：
            1、此目录没有被其他进程使用
            2、目录要事先存在
            3、目录中的原有的文件将会暂时隐藏

挂载完成后要通过挂载点访问文件

umount：卸载某文件系统
    umount 设备
    umount 挂载点

    卸载注意事项：
        挂载的设备没有进程使用

mount：显示当前系统已经挂载的设备及挂载点
mount [options] [-o options] DEVICE MOUNT_POINT
    -a:表示挂载/etc/fstab 文件中定义的所有文件系统

    -n:默认情况下，mount命令每挂载一个设备，都会把挂载的设备信息保存至/etc/mtab文件；使用
        -n选项意味着不写入
    -t
    FSTYPE:指定正在挂载设备上的文件系统的类型，不使用次选项时，mount会调用blkid命令获取
            对应文件系统类型
    -r:只读挂载 挂载光盘时常用此选项
    -w:读写挂载

    -o:指定额外的挂载选项，也即指定文件系统启用的属性
        remount：重新挂载当前文件系统
        ro:挂在为只读
        rw：读写挂载

    mount -o remount，ro 。。。

swap：交换空间

增加交换分区：
    1、新建一个分区
    2、更改分区类型为交换空间 82
    3、通知内核重新读取 partprobe /dev/sdb
    4、创建交换分区类型文件系统

创建交换分区：
mkswap /dev/sdb2
    -L LABEL：卷标

swapon /dev/sdb2 ：启用交换空间
    -a:启用所有的定义在/etc/fstab文件中的交换设备
swapoff /dev/sdb2 :关闭交换空间

free:查看物理内存与交换分区的使用情况
    -m：以m的形式显示

回环设备
loopback，使用软件来模拟实现硬件

创建一个镜像文件，1G

dd命令
    if=数据来源
    of=数据存储目标

    bs=1 blocksize 块大小单位为字节
    count=2
dd if=/dev/sda of=/mnt/usb/mbr.backup bs=512 count=1 将MBR保存到优盘文件
dd if=/mnt/usb/mbr.backup of=/dev/sda bs=512 conut=1

cat /dev/cdrom > /root/rhe15.iso 直接制作镜像文件

dd if=/dev/zero of=/var/swapfile bs=1M count=1024
dd if=/dev/zero of=/var/swapfile seek=1023 bs=1M count=1024

mount命令，可以挂载iso镜像
mount DEVICE MONT POINT
    -o loop:挂载本地回环设备
mount -o loop ubuntu-16.04-desktop-amd64.iso /media/zhangshuo/iso/   挂载iso镜像文件
umount /media/zhangshuo/iso 卸载iso镜像文件

wget ftp://172.16.0.1/pub/isos/rhci-5-8-1.iso

文件系统的配置文件 /etc/fstab
    os在初始化时，会自动挂载此文件中定义的每一个文件系统

使用空白隔开的六个字段

要挂在的设备   挂载点 文件系统类型   挂载选项   转储频率（每多少天做一次备份）   文件系统检测次序（只有根可以为1，0表示不检查）
/dev/sdb1   /rum/media/zhangshuo/usb1   vfat   defaults   0   0

mount -a :挂载/etc/fstab文件中定义的所有文件系统

fuser :验证进程正在使用的文件或套接字文件
    -v:查看某文件上正在运行的进程
    -k: kill
    -m:name

    fuser -km MOUNT_POINT:终止正在访问此挂载点的所有进程

练习：
1、创建一个5G的分区，文件系统为ext3，卷标为MYDATE，块大小为1024，预留管理空间为磁盘分区
    的3%，要求开机后可以自动挂载至/data 目录，并且自动挂载的设备要使用卷标进行引用
2、创建一个本地回环文件/var/swaptemp/seapfile来用于swap，要求大小为512MB，卷标为SWAP-FILE，并且
    开机自动启动此交换设备
    mkdir /var/swaptemp
    dd if=/dev/zero of=/var/swaptemp/swapfile bs=1M count=512
    mkswap LABEL=SWAP-FILE /var/swaptemp/swapfile

    /etc/fstab
    /var/swaptemp/swapfile swap   swap   defaults   0 0
3、上述第一问，如何让其自动挂载的同时启动ACL功能
    /ect/fstab
    LABEL='MYDATE' /data   ext3   defaults,acl 0 0

压缩、解压缩命令
压缩格式：gz，bz2,xz,zip,z

压缩算法：算法不同，压缩比也会不同；

comprees:FILENAME.z
uncompress

gzip:.gz
    gzip PATH/TO/SOMEFILE:压缩完成后会删除源文件
        -d:解压缩
        -#：1-9：指定压缩比，默认为6
gunzip:

```
    gunzip /PATH/TO/SOMEFILE.gz :压缩完成后会删除源文件

    zcat:不解压的情况下，查看压缩后的文件内容
        zcat test.gz

bzip2:.bz2
比gzip有着更大压缩比的压缩工具，使用格式近似gzip
    bzip2 /PATH/TO/SOMEFILE
        -d：解压缩
        -#：1-9 ，默认为6的压缩比
        -k:压缩时保留源文件

    bunzip2 /PATH/TO/SOMEFILE:解压缩文件
    bzcat /PATH/TO/SOMEFILE 不用解压缩直接查看文件内容
xz:.xz
    xz /PATH/TO/SOMEFILE
        -d:解压缩文件
        -#:1-9,默认为6
        -k:压缩时保留源文件

    unxz /PATH/TO/SOMEFILE:解压缩文件
    xzcat:
    xzdec:

zip:即归档又压缩的工具
    zip FILENAME.zip FILE1 FILE2 ...: 压缩后不删除源文件
    unzip FILENAME.zip 解压缩文件


archive:归档，归档本身并不意味者压缩

tar:归档工具
    -c:创建归档文件
    -f  FILE.tar ：操作的归档文件
    -x:展开归档
    --xattrs:归档时，保留文件的扩展属性信息
    -t:不展开归档，直接查看归档了哪些文件

    -zcf:归档并调用gzip压缩
    -zxf:调用gzip解压缩并展开归档

    -jcf: bzip
    -jxf:

    -Jcf:xz
    -Jxf:

    -jtf:不解压缩直接查看压缩包内容

RAR压缩和解压文件
1、解压文件
    rar x FILENAME.rar
2、压缩文件
    rar a FILENAME.rar /path

cpio:归档工具

#!/bin/bash
#by zhangshuo 20180121

echo -n "input 2 num:"
read NUMBER1 NUMBER2
echo "$NUMBER1 plus $NUMBER2 equel:$[$NUMBER1 + $NUMBER2]"

read：读取键盘内容并赋值给变量
    -p:直接提示内容
    -t:超时事件限制
#!/bin/bash
#by zhangshuo 20180121

read -t 3  -p "input 2 num:" NUMBER1 NUMBER2

[ -z $NUMBER1 ] && NUMBER1=1 && echo -n "1 + "
```

```
[ -z $NUMBER2 ] && NUMBER2=1 && echo  "1"

echo "$NUMBER1 plus $NUMBER2 equel:$[$NUMBER1 + $NUMBER2]"
```

练习：写一个脚本（line2397.sh）
从键盘让用户输入几个文件，脚本能够将此几个文件归档压缩成一个文件

```
#!/bin/bash
#by zhangshuo 20180121

read -p "three files: " FILE1 FILE2 FILE3

read -p "name: " FILENAME

read -p "type: " TYPE

case $TYPE in
gzip)
    tar -zvcf ${FILENAME}.tar.gz $FILE1 $FILE2 $FILE3
    ;;
bzip2)
    tar -jvcf ${FILENAME}.tar.bz2 $FILE1 $FILE2 $FILE3
    ;;
xz)
    tar -Jvcf ${FILENAME}.tar.xz $FILE1 $FILE2 $FILE3
    ;;
*)
    echo "unkonw"
    exit 2
    ;;
esac
```

while循环，适用于循环次数未知的场景,要有退出条件
语法：

```
    while CONDITION ;do
        statement
        ...
    done
```

就算100以内所有整数的和（line2411.sh）

```
#!/bin/bash
#by zhangshuo 20180121

declare -i I=1
declare -i SUM=0

while [ $I -le 100 ];do
    let SUM+=$I
    let I++
done

echo $SUM $I
```

给定一个字符串并转换为大写，当输入quit时退出while quit (line2413.sh)

```
#!/bin/bash
#by zhangshuo 20180121


read -p "input some string: " STRING

while [ $STRING != 'quit' ];do
    echo $STRING | tr 'a-z' 'A-Z'
    read -p "input some string: " STRING

done
```

每隔五分钟查看一次某用户是否登陆系统（line2415.sh）

```
#!/bin/bash
#by zhangshuo 20180121

RES=1

while [ $RES -eq 1 ];do
    if who | grep "jialei" &> /dev/null;then
        RES=0
        echo " jialei in"
    else
        sleep 5
    fi
done
```

写一个脚本：（line2417.sh）
1）显示一个菜单给用户

```
d|D) show disk usages
m|M) show memory usages
s|S) show swap usages
*)quit
```

2)当用户给定选项后显示相应的内容；

扩展：
当用户选择完成，显示相应消息后，不退出；而让用户再一次选择，再次显示相应内容
除非用户使用quit退出

```
#!/bin/bash
#by zhangshuo 20180121

#echo "d|D) show disk usages"
#echo "m|M) show memory usages"
#echo "s|S) show swap usages"
#echo "*) quit"

cat << EOF
d|D) show disk usages
m|M) show memory usages
s|S) show swap usages
*) quit
EOF

read -p "chose one: " CHOSICE

while [ $CHOSICE != 'quit' ];do
case $CHOSICE in
d|D)
    df -h
    ;;
m|M)
    free
    ;;
s|S)
    free
    ;;
*)
    echo "unknow"
esac
read -p "chose one: " CHOSICE
done


echo -e "\033[31mhello\033[0m,world" 用于控制输出的颜色设置

IDE:133Mbps
STAT:300Mbps    600Mbps 6Gbps
usb3.0：480Mbps
SCSI:small computer system interface
    10000,15000

RAID:
级别，仅代表磁盘组织方式不同，没有上下之分
0：条带
    性能提升：读，写
```

    冗余能力：无
1：镜像
    性能提升：写能力下降，读性能提升
    冗余能力：有
2：
3：
4：
5：
    性能表现：读写提升
    冗余能力：有
    空间利用率：（n-1）/n
10：
    性能表现：读写提升
    冗余能力：有
    空间利用率：0.5
01：
    性能表现：读写提升
    冗余能力：有
    空间利用率：0.5
50：
    性能表现：读写提升
    冗余能力：有
    空间利用率：(n-2)/n

JBOD：
    性能表现：无
    冗余能力：无
    空间利用率：1
Hadoop
    HDFS

硬件RAID
软件RAID

逻辑RAID
/dev/md0 表示raid逻辑设备
/dev/md1

当一个系统完成磁盘RAID后如果系统崩溃则数据无法找回，所以制作RAID时应将要制作的设备标记为内核可识别的设备，即在磁盘分区时就行该指定磁盘类型。RAID的类型为FD。

md：multi disks 软件raid
mdadm：将任何块设备做成RAID
模式化的命令：
    创建模式
        -C
            专用选项
                -l：级别
                -n：设备个数
                -a （yes no）：自动为其创建设备文件
                -c：chunk大小即数据块大小 默认为64k
                -x：指定空闲盘个数
    管理模式
        --add,--remove
        mdadm /dev/md# --fail /dev/sdb7 模拟将/dev/sdb7进行损坏
        mdadm /dev/md1 -f /dev/sdb7 模拟硬盘损毁
        mdadm /dev/md1 -r /dev/sdb7 卸载坏的硬盘
        mdadm /dev/md1 -a /dev/sdb7 增加一块好的硬盘到raid
    监控模式
        -F
    增长模式
        -G
    装配模式
        -A

创建RAID0
    2G：
        4：512MB
        2：1G

创建raid0   mdadm -C /dev/md0 -a yes -l 0 -n 2 /dev/sdb{5,6}
查看当前raid   cat /proc/mdstat

创建raid1
    2G：
        2：2G

mdadm -D /dev/md1 :查看磁盘阵列的详细信息

停用阵列：
    mdadm -S /dev/md#
        --stop
启动阵列
    mdadm -A /dev/md#

完整删除RAID信息
1、查看RAID磁盘阵列信息，确认一下要关闭哪个。如：关闭md0这个阵列
    cat /proc/mdstat
2、卸载md0这个阵列的挂载点
    umount /mnt/RAID5
3、停止md0这个阵列，并释放与该阵列相关的所有资
    mdadm -S /dev/md0
4、清除成员磁盘当中阵列的超级块信息，这一步很重要！
    mdadm --zero-superblock /dev/sdb[1-3,5]
5、删除或注释/etc/fstab上的挂载信息
6、删除或注释/etc/mdadm.conf对应的RAID信息

watch：周期性的执行命令，并以全屏方式显示结果
    -n #：指定周期长度，单位为秒，默认为2
    watch -n # 'COMMAND'

mdadm -D --scan > /etc/mdadm.conf 将raid信息保存至文件以后便可开机装配

mke2fs -E stride 16 :格式化时指定条带大小这样可以提高raid性能。chuck大小默认为64k 磁盘块大小为4k所以这里数字为16.注意用优盘做raid时它的chunk大小为512k。

RAID5:
    2G：3，1G
    madam -C /dev/md2 -a yes -l 5 -n 3 -x 1 /dev/sdb{1,2,3}

lsmod:显示已经加载到内核中的模块状态信息

mdadm：用户空间工具只是管理工具，对于raid其实还是内核中raid模块进行的管理。

MD,DM
        /dev/md#
        meta device

DM:Device Mapper
    逻辑设备
        RAID,LVM2

DM:LVM2
    线性 用md也可实现，类似jbod
    镜像 md也可以实现既raid
    快照 数据备份
    多路径

逻辑卷的快照必须和逻辑卷在同一个卷组中切必须留有足够的空间给快照卷

将物理卷加入到卷组中时会给物理卷分块类似chunk此时的块叫做pe：physical extend 物理盘区 既逻辑存储单位

将pe放到逻辑卷中，即逻辑卷的存储空间是由若干个pe组成的。此时的pe叫做le：logical extend 逻辑盘区

physical volume pv:物理卷
    pvcreate,pvremove,pvscan,pvdisplay,pvmove
    pvcreate /dev/sdb{11,12}
    pvs :查看当前系统的pv
    pvdisplay
volume group vg:卷组类似扩展分区
    vgcreate,vgremove,vgextend,vgreduce,vgs,vgdisplay,vgscan
    vgs

vgcreate:创建卷组
-s：可以指定pe大小默认为4m 该选项可以带单位用时man一下
vgcreate myvg /dev/sdb{11,12}

vgremove vg_name ： 移除该vg

vgcreat VG_NAME /PATH/TO/PV
-s #:PE大小，默认为4MB

vgcreate -s 8M myvg /dev/sdb{11,12}

缩小一个vg的过程
1、pvmove 移除一个pv该过程会自动复制数据
pvmove /dev/sdb12
2、vgreduce myvg /dev/sdb12
3、pvremove 真正移除pv

增加一个vg的过程
1、pvcreate /dev/sdb12
2、vgextend myvg /dev/sdb12

logical volume：逻辑卷
lvcreate,lvremove,lvextend,lvreduce,lvresize,lvs,lvdisplay
lvs

lvcreate -n LV_NAME -L #G VG_NAME

lv创建好以后其设备目录为/dev/vgname/lvname，但是其实真正的lv设备是在/dev/mapper下面

删除一个lv
1、卸载
umount /run/media/zhangshuo/vg0/
2、删除
lvremove /dev/myvg/testlv

练习：创建一个由两个物理卷组成的大小为20G的卷组myvg，要求其PE大小为16M，而后在此卷组中创建一个
大小为5G的逻辑卷lv1，此逻辑卷要能在开机后自动挂载至/users目录，且支持ACL功能

1、扩展逻辑卷：
lvextend ： 扩展其物理边界
-L [+]# /PATH/TO/LV

resize2fs ： 扩展其文件系统边界
resize2fs /PATH/TO/LV 5G 修复逻辑边界到5g
resize2fs -p /PATH/TO/LV 扩展至最大

2、缩减逻辑卷
注意：
1、不能在线缩减，要先卸载再缩减
2、确保缩减后的空间大小依然能存储原有的所有数据
3、在缩减之间应线强行检查文件，以确保文件系统处于一致性状态：

df -lh 确保大小足够
umount 卸载挂载
e2fsck -f /PATH/TO/LV 检查文件系统

resize2fs
resize2fs /PATH/TO/PV 3G 缩小至3G

lvreduce -L [-]# /PATH/TO/LV

重新挂载

3、快照卷
1、生命周期为整个数据时常：在这段时常内，数据的增长量不能超出快照卷大小：
2、快照卷应该是只读的
3、跟原卷在同一卷组内

lvcreate
    -s :指定为快照卷
    -p r|w 指定权限，一般快照用于数据备份使用所以快照卷不允许修改所以应该为只读

lvcreate -L # -n SLV_NAME -s -p r /PATH/TO/LV

脚本编程控制结构
        顺序
        选择
            if
            case
        循环
            for
            while
            until

while CONDITION; do
    statment
done

进入循环，条件满足
退出循环，条件不满足

until CONDITIONL;do
    statement
done

进入循环，条件不满足
退出循环，条件满足
（line2647.sh）

每隔五秒钟查看zhangshuo是否登陆通过until实现（line2649.sh）

for ((expr1 ; expr2 ;expr3));do
    循环体
done

100以内所有整数的和for另一种用法实现（line2656.sh）

写一个脚本（line2658.sh）
1、通过ping命令测试192.168.0.151到192.168.0.254之间的所有主机时候在线
    如果在线，就显示“IP is up 。”其中的IP要换为真正的IP地址，且以绿色显示：
    如果不在线就显示“ip is down”，其中的IP要换为真正的IP地址，且以红色显示

要求：分别使用while，until，和for循环实现
echo -e "\033[31mhello\033[0m,world" 用于控制输出的颜色设置

ping
    -c：指定ping次数
    -w：指定等待时间，超时则退出

awk ‘pattern{action}' file
    print $1 :显示第一段

Df -Ph | awk '{print $1}'
awk -F : '{print $1,$3}' /etc/passwd 以：为分割字段，输出第一三列

fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]":显示设备信息

fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" | awk -F : '{print $1}':只显示设备名称

写一个脚本（前提，请为虚拟机新增一块硬盘，假设它为/dev/sdb），为指定的硬盘创建分区
1、列出当前系统上所有的磁盘，让用户选择，如果用户选择quit则退出脚本，如果用户选择错误，就让用户重新选择
2、当用户选择后，提醒用户确认接下来的操作可能会毁坏数据，并请用户确认：如果用户选择y就继续，否则，让用户重新选择
3、抹除那块硬盘上的所有分区（提示，抹除所有分区后执行sync命令，并让脚本睡3秒后再分区）；并为其创建三个分区，第一个为20M，第二个为

512M，第三个为128M，且第三个为swap分区类型，（提示，将分区命令通过echo传送给fdisk即可实现）

```
fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]"
fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" | awk -F : '{print $1}'
```

取出第一段
脚本来实现（line2675.sh）

dd if=/dev/zero of=/dev/sdb bs=512 count=1 ：抹除硬盘的MBR既删除所有分区
执行上述命令之后执行
sync将数据同步到磁盘
sleep 3 睡3秒防止操作太快磁盘无法正常工作

```
#!/bin/bash
#by zhangshuo 20180123
#
fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" | awk -F : '{print $1}'

read -p "input you disk: " DISK

until [ $DISK == 'quit' ] || fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" | awk -F : '{print $1}' | grep "^Disk ${DISK}$" &> /dev/null ;do
    read -p "input you disk: " DISK
done

echo "your choice is $DISK"

echo "waring the dask will gg!!!!,continu?"

read -p "y/n: " CHOICE

until [ $CHOICE == 'n' -o $CHOICE == 'y' ];do

    read -p "y/n: " CHOICE
done

echo $CHOICE

if [ $CHOICE == 'y' ];then
    #当操作某个设备时应首先保证该设备是处于没有挂载状态
    for I in  `mount | grep "$DISK" | awk '{print $1}'`;do
        umount $I
        echo "umount $I done"
    done
    #卸载完设备以后便可以开始格式化
    echo "stat repart $DISK"
    #清除但前磁盘所有分区
    dd if=/dev/zero of=$DISK bs=512 count=1 &> /dev/null
    sync
    sleep 3
echo 'n
p
1

+1G
n
p
2

+1G
n
p
3

+1G
n
e


n

+1G
n
```

```
+1G
n

+1G
n

+1G
n


t
1
fd
t
2
fd
t
3
fd
t
5
fd
t
6
fd
t
7
fd
t
8
fd
w'| fdisk $DISK &> /dev/null
    sync
    sleep 3

    partprobe

    echo "done"

    fdisk -l $DISK

    #此时分区结束，开始建立RAID及LVM

else
    exit 0
fi


网络管理

点分十进制：
0-255

221.34.23.12

Ip地址与端口的绑定就是套接字文件

网络地址：
主机地址：

A类：第一段表示网络地址后三段表示主机地址   255.0.0.0   8位长度的掩码
    0   000 0001 - 0    111 1111
    127个A类，127用于回环，1-126
    容纳多少个主机：2^24-2
    主机位全0：网络地址
    主机位全1：广播地址
B类：前两段表示网络地址，后两段表示主机地址 255.255.0.0 16位长度的掩码
    10  00  0000 - 10   11  1111
    128-191
    64个B类网络，2^14方个
    容纳多少个主机：2^16-2
C类：前三段表示网络地址，最后一段表示主机地址   255.255.255.0 24位长度的掩码
```

110 0   0000 - 110 1   1111
192-223
32个C类地址，2^21次方个C类网络
容纳多少个主机：2^8-2
D
1110   0000 - 1110 1111
224-239
E

私有地址用于建立局域网可以随意使用的地址，但是不可以连接到外网：
A类：10.0.0.0/8
B类：172.16.0.0/16-172.31.0.0/16
C类：192.168.0.0/24-192.168.255.0/24

路由，选路

主机接入网络需要配置如下信息：

IP:
NETMASK掩码:
GATEWAY网关:
HOSTNAME主机名：
DNS1：服务器地址
DNS2：备用dns
DNS3:
以上所有配置手动指定
DHCP:
随机自动分配给的地址169.254

路由

Linux:
lo：本地回环
以太网卡：ethX
pppX：点对点连接

在系统中如果直接操作设备文件会很麻烦，所以linux引入了设备名称，这样可以直接引用该设备
RHEL6中定义网卡名称的文件：/etc/udev/rules.d/70-persistent-net.rules
RHEL5中定义网卡别名的文件：/etc/modprobe.conf 通过alias别名定义的

修改Centos7ifconfig命令的网卡名称
1、vim /etc/default/grub
在GRUB_CMDLINE_LINUX的最后，加上 net.ifnames=0 biosdevname=0 的参数
GRUB_CMDLINE_LINUX="rd.lvm.lv=rootvg/usrlv rd.lvm.lv=rootvg/swaplv crashkernel=auto vconsole.keymap=us rd.lvm.lv=rootvg/rootlv vconsole.font=latarcyrheb-sun16 rhgb quiet net.ifnames=0 biosdevname=0"
2、 grub2-mkconfig -o /boot/grub2/grub.cfg
3、 mv /etc/sysconfig/network-scripts/ifcfg-enp0s3 /etc/sysconfig/network-scripts/ifcfg-eth0
4、将复制过来的文件内容替换为eth0
5、 reboot

ifconfig
-a:显示所有接口的配置信息

ifconfig ethX IP/MASK {up|down}
配置的地址立即生效，但重启网络服务或主机，都失效

网络服务：
RHEL5: /etc/init.d/network {start|stop|restart|status}
RHEL6: /etc/init.d/NetworkManager {start|stop|restart|status}

网关：
route
add:添加
-host:主机路由
-net:网络路由
-net 0.0.0.0
J 表示可以通过192这个网关到达10网络
route add -net|-host DEST gw NEXTHOP
route add default   gw NEXTHOP 添加默认路由

del:删除
-host
-net

route del -net 10.0.0.0/8 [gw NEXTHOP]
route del -net 0.0.0.0 删除默认路由
route del default

所做出的改动或重启网络服务后失效；

查看：
route -n ：以数字方式显示各主机或端口等相关信息

查看网卡UUID：
nmcli con show   或   nmcli con list

查看网卡mac地址：
nmcli dev show   或   nmcli dev list

查看当前网卡速度命令：
dmesg | grep "eth0"

查看硬件设备生产商
lspci

网络配置文件：
/etc/sysconfig/network

网络接口配置文件：
/etc/sysconfig/network-scripts/ifcfg-INTERFACE_NAME
DEVICE=:关联的设备名称，要与文件名的后半部"INTERFACE_NAME"保持一致；
BOOTPROTO={static|none|dhcp|bootp}:引导协议：要使用静态地址，使用static或none；dhcp表示使用DHCP服务器获取地址；
IPADDR=：IP地址
NETMASK=：子网掩码
GATEWAY=:设定默认网关
ONBOOT=：开机时是否自动激活此网络接口
HWADDR=:硬件地址，要与硬件中的地址保持一致，可省：
USERCTL={yes|no}:是否允许普通用户控制此接口
PEERDNS={yes|no}:是否在BOOTPROTO为dhcp时接受由DHCP服务器指定的DNS地址：

不会立即生效，但重启网络服务或主机都会生效：

路由：
/etc/sysconfig/network-scripts/route-ethx
添加格式一：新建文件
DEST{目标} via{下一跳}   NEXTHOP{}
192.168.10.0/24 via 10.10.10.254

添加格式二：
ADDRESS0=
NETMASK0=
GATEWAY0=

DNS服务器指定方法只有一种：
/etc/resolv.conf
nameserver DNS_IP_1
nameserver DNS_IP_2

指定本地解析：
/etc/hosts
主机IP 主机名 主机别名
172.16.0.1 www.zhangshuo.com   www

DNS-->/etc/hosts-->DNS

配置主机名：
hostname   HOSTNAME
立即生效，但不是永久生效

```
/etc/sysconfig/network
HOSTNAME=


ifconfig :老旧

iproute2
        ip
            link:配置网络接口属性
                show
                    -s:显示统计信息
                    ip -s link show
                set
                    ip link set DEV{up|down}
            addr：协议地址
                add
                    ip  addr add ADDRESS dev DEV label ethx:x
                    ip addr add 10.3.3.3/8 dev eth1 label eth1:1
                del
                    ip  addr del ADDRESS dev DEV label ethx:x
                show

                flush 清除
                    ip addr flush eth1 to 10/8

            route：路由



一块网卡可以使用多个地址：
网络设备可以别名：
eth0
        etchX:X
        ifconfig eth0:0 192.168.1.105/24

配置方法：
    ifconfig    ethx:x IP/NETMASK

    /etc/sysconfig/network-scripts/ifcfg-ethx:x
    DEVICE=ethx:x

    非主要地址不能使用DHCP动态获取



软件包管理

应用程序：
        程序与，Architecture体系架构有很大的关系
C语言：源代码-->(编译)二进制格式
脚本；解释器（二进制程序）

源代码：-->编译-->链接-->运行
    程序：
        库
            静态
            动态

            静态链接
            动态链接
                共享库.so

配置文件：
dir=/path/to/somewhere

程序的组成部分：
    二进制程序
    库
    配置文件
    帮助文件

/boot
/etc
```

/usr
/var
/dev
/lib
/tmp
/bin
/sbin
/proc
/sys
/mnt
/media
/home
/root
/misc
/opt
/srv

/usr/share/man：帮助文件

/etc:配置文件,/bin,/sbin：二进制程序,/lib：库文件
    系统启动就需要用到的程序，这些目录不能挂载额外的分区，必须挂载在根文件系统的分区上

/usr/
    bin
    sbin
    lib

    提供操作系统核心功能，可以单独分区

/usr/local
    bin
    sbin
    lib
    etc
    man

    第三方程序安装目录，建议单独分区

/proc
/sys
        不能单独分区，默认为空

/dev:设备，不能单独分区：

    udev:利用内核信息动态创建设备文件，使得设备文件可以按需创建不再是将所有文件都放在这里
/root:不能单独分区

/var:日志信息和运行信息存放目录 建议单独分区

/boot:内核 建议单独分区因为文件系统一般是靠LVM托管，开机时bootloader要找内核

程序：指令+数据
        指令：芯片
        CUP：普通指令，特权指令
        指令集

C语言：
        Powerpc CPU 编译为 二进制

        不能在x86:运行

软件包管理器
打包成一个文件：二进制程序，库文件，配置文件，帮助文件
生成数据库，追踪所安装的每一个文件

软件包管理器的核心功能
1、制作软件包

2、安装、卸载、升级、查询、校验：

Redhat ，suse，debian

redhat，suse RPM

Debian：dpt

依赖关系：
        x-->y-->z
前端工具：yum apt-get
后端工具：rpm，dpt

yum：Yellowdog Update Modifier

rpm命令：
        rpm：
            数据库/var/lib/rpm
        rpmbuid：用于创建rmp软件包

安装、查询、卸载、升级、校验、数据库重建、验证数据包的合法性等工作；

rpm命名：
包：组成部分
    主包：
        bind-9.7.1-1.i586.el5.rpm
    子包：
        bind-libs-9.7.1-1.i586.el5.rpm
包名格式
    name-version-release.arch.rpm
    bind-major.minor.release-release.aech.rpm

主版本号：重大改进
次版本号：某个子功能发生重大改变
发行号：修正了部分bug，调整了一点功能

bind-9.7.1.tar.gz这是下载的文件

rpm包：
    二进制格式：安装简单

        rpm包作者下载源程序，编译配置完成后，制作成rpm包此时会加上自己的发行号，即后面的发行号。
    源码格式：能够更适合自己的硬件平台

rpm

1、安装
rpm
    -i /PATH/TO/PACKAGE_FILE
    -h:以#显示安装进度：每个#表示2%；
    -v:显示详细过程
    -vv:更纤细的过程

rpm -ivh /PATH/TO/PACKAGE_FILE

    --nodeps:忽略依赖关系：
    --replacepkgs:重新安装，替换原有安装：
    --force:强行安装，可以实现重装或降级安装

2、查询
rpm -q PACKAGE_NAME ：查询指定的包是否已经安装
    --changelog ：rpm包的修改日志
    rpm -q bash
rpm -qa :查询已经安装的所有包
    rpm -qa python*

rpm -qi PACKAGE_NAME ：查询指定包的说明信息
rpm -ql PACKAGE_NAME:查询指定包安装后生成的文件列表
rpm -qc PACKAGE_NAME:查询安装包的配置文件
rpm -qd PACKAGE_NAME:查询指定包安装的帮助文件；

rpm -q --scripts PACKAGE_NAME：查询指定包包含的脚本
    脚本包含四部分，安装前，安装后，卸载前，卸载后

rpm -qf /path/to/somefile:查询指定的文件是由哪个rpm包安装生成的：

如果某rpm包尚未安装，我们需要查询其说明信息、安装以后会生成的文件；

rpm -qpi /PATH/TO/PACKAGE_FILE

3、升级
rpm -Uvh /PATH/TO/NEW_PACKAGE_FILE：如果装有老版本的，则升级：否则，则安装
rpm -Fvh /PATH/TO/NEW_PACKAGE_FILE:如果安装有老版本的，则升级，否则，退出

4、降级
rpm -Uvh --oldpackage /PATH/TO/NEW_PACKAGE_FILE
rpm -Fvh --oldpackage /PATH/TO/NEW_PACKAGE_FILE

5、卸载
rpm -e PACKAGE_NAME
        --nodeps 如果有安装包依赖此安装包则禁止卸载，如果要强行卸载则需要解除依赖

6、校验
rpm -V PACKAGE_NAME

校验结果会用如下格式表示

```
       S file Size differs
       M Mode differs (includes permissions and file type)
       5 digest (formerly MD5 sum) differs
       D Device major/minor number mismatch
       L readLink(2) path mismatch
       U User ownership differs
       G Group ownership differs
       T mTime differs
       P caPabilities differ
```

7、重建数据库
    rpm
        --rebuilddb:重建数据库，一定会重新建立。
        --initdb:初始化数据库，没有才建立，有就不用建立

8、校验来源合法性，及软件完整性
加密类型：
        对称加密：加密和解密使用同一个密码
        公钥加密：每个密码都成对出现，一个为私钥一个为公钥
                一对儿密钥，公钥，私钥：公钥隐含与私钥中，可以提取出来，并公开出去
        单向加密，散列加密：提取数据特征码，常用于数据完整性校验

ls /etc/pki/rpm-gpg 保存红帽对外发布的校验码
    RPM-GPG-KEY-redhat-release

rpm --import   /etc/pki/RPM-GPG-KEY-redhat-release 导入公钥

rpm -K /PATH/TO/PACKAGE_NAME
    dsa,gpg:验证来源合法性，也即验证签名：可以使用--nosignature,忽略此项
    sha1，md5:验证软件包完整性：可以使用--nodigest,忽略此项

要解决rpm包的依赖关系很是复杂所有制作了yum软件包管理器
该软件包为C/S架构，Client，Servier 即客户端服务器架构

yum repository yum仓库
    文件服务
        ftp：//
        web：//
        file：///

yum自己的客户端
    配置文件（该客户端的各种配置信息）
        写明服务器地址

仓库文件中都会存在repodata文件夹，文件夹中就包含了该yum仓库的信息

HEML：Hypertext mark language
XML:extended Mark Language

xml,json:半结构化的数据在不同操作系统之间实现数据传输

yum仓库中的元数据文件；
perimary(首要的最早的).xml.gx
    所有RPM包的列表
    依赖关系
    每个RPM安装生成的文件列表
filelists.xml.gz
    当前仓库中所有RPM包的所有文件列表：
other.xml.ge
    额外信息，rpm包的修改日志：
repomd.xml
    记录的是上面三个文件的时间戳和校验和：

comps*.xml:RPM包分组信息

man yum

```
command is one of:
        * install package1 [package2] [...]
        * update [package1] [package2] [...]
        * update-to [package1] [package2] [...]
        * update-minimal [package1] [package2] [...]
        * check-update
        * upgrade [package1] [package2] [...]
        * upgrade-to [package1] [package2] [...]
        * distribution-synchronization [package1] [package2] [...]
        * remove | erase package1 [package2] [...]
        * autoremove [package1] [...]
        * list [...]
        * info [...]
        * provides | whatprovides feature1 [feature2] [...]
        * clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]
        * makecache [fast]
        * groups [...]
        * search string1 [string2] [...]
        * shell [filename]
        * resolvedep dep1 [dep2] [...]
           (maintained for legacy reasons only - use repoquery or yum provides)
        * localinstall rpmfile1 [rpmfile2] [...]
           (maintained for legacy reasons only - use install)
        * localupdate rpmfile1 [rpmfile2] [...]
           (maintained for legacy reasons only - use update)
        * reinstall package1 [package2] [...]
        * downgrade package1 [package2] [...]
        * deplist package1 [package2] [...]
        * repolist [all|enabled|disabled]
        * repoinfo [all|enabled|disabled]
        * repository-packages <enabled-repoid>
        <install|remove|remove-or-reinstall|remove-or-distribution-synchronization>
        [package2] [...]
        * version [ all | installed | available | group-* | nogroups* | grouplist |
        groupinfo ]
        * history
        [info|list|packages-list|packages-info|summary|addon-info|redo|undo|rollback|n
        ew|sync|stats]
        * load-transaction [txfile]
        * updateinfo [summary | list | info | remove-pkgs-ts | exclude-updates |
        exclude-all | check-running-kernel]
        * fssnapshot [summary | list | have-space | create | delete]
        * fs [filters | refilter | refilter-cleanup | du]
        * check
        * help [command]
```

yum的配置信息在/etc/yum.conf

```
[main]
cachedir=/var/cache/yum/$basearch/$releasever
```

```
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
#严格检查平台版本
exactarch=1
#过期的要不要废弃
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=5
bugtracker_url=http://bugs.centos.org/set_project.php?project_id=23&ref=http://bugs.centos.org/bug_report_page.php?category=yum
#发行的版本号
distroverpkg=centos-release


#  This is the default, if you make this bigger yum won't see if the metadata
# is newer on the remote and so you'll "gain" the bandwidth of not having to
# download the new metadata and "pay" for it by yum not having correct
# information.
#  It is esp. important, to have correct metadata, for distributions like
# Fedora which don't keep old packages around. If you don't like this checking
# interupting your command line usage, it's much better to have something
# manually check the metadata once an hour (yum-updatesd will do this).
# metadata_expire=90m

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d

如何为yum定义repo文件/etc/yum.repos.d
[Repo_ID]
name=Description
baseurl=
    ftp://
    http://
    file:///
enabled={0|1}是否启用
gpgcheck={0|1}是否启用校验
gpgkey=

yum [options] [command] [package ...]
    -y:自动回答为yes
    --nogpgcheck:

list:列表
    all:所有的
    availabel:可用的，仓库中有但尚未安装的
    installed：已经安装的
    updates:可用的升级

clean:清理缓存
    all:
repolist:[all|enabled|disabled] 列出当前系统的仓库

install：安装软件包
yum install PACKAGE_NAME

update:升级
update_to ：升级为指定版本

remove | erase :卸载

info :详细信息

provides | whatprovides :查看指定的文件或特性是由哪个包安装生成的；
yum provides /etc/inittab :查看某个文件属于哪个安装包，即rpm -qf /path/to/somewhere

group info :组详细信息
group list
group install
group remove
group update
```

如何创建yum仓库
createrepo /yum/VT/
1、在/etc/yum.repo中创建一个文件
[Base]
name=Centos7 CDROM Server
baseurl=file:///media/zhangshuo/cdrom
enabled=1
gpgcheck=0

RPM安装：
        二进制格式：
        源程序-->编译-->二进制格式
            有些特性是编译选定的，如果编译未选定此特性，将无法使用：
            rpm包的版本会落后与源码包，甚至落后很多：bind-9.8.7，bind-9.7.2

定制：手动编译安装

编译环境，开发环境
开发库，开发工具

Linux：C，
GNU:C

C，C++
gcc:GNU C Complier,c
g++:

如果手动使用gcc编译器，当要编译和链接的文件很多时要指定很多编译次序，此时对于大项目来说会很复杂，所以有了make工具

make:项目管理工具
    makefile：定义了make(gcc,g++)gcc按何种次序去编译这些源程序文件中的源程序

automake,-->makefile.in-->makefile
autoconf,-->configure

100个可选特性

make install

编译安装的三步骤
前提：准备开发环境（编译环境）
安装“Development Tools”和“Development Libraries”

#tar
#cd
#./configure
        --help:帮助
        --prefix=/PATH/TO/SOMEWHERE:软件安装路径
        --sysconfdir=/PATH/TO/CONFFIE_PATH:配置文件路径
        功能：1、让用户选定编译环境：2、检查编译环境：
#make
#make install

更改环境变量/etc/profile
    PATH=$PATH:/usr/local/nginx/sbin加入此行可以改变环境变量
    export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
或
在/etc/profile.d/目录下建立一个以.sh为名称后缀的文件，在里面定义export PATH=$PATH:/PATH/TO/SOMEWHERE

source /etc/profile :可以通知内核重新读取该文件

tar xf tenfine-1.4.2.tar.gz
cd tengine-1.4.2
./configure --prefix=/usr/local/tengine --conf-path=/etc/tengine/tengine.conf
make
make install

1、修改PATH环境变量，以能够识别此程序的二进制文件路径
    修改/etc/profile文件

在/etc/profile.d/目录建立一个以.sh为名称后缀的文件，在里面定义export PATH=$PATH:/PATH/TO/SOMEWHERE

2、默认情况下，系统搜索库文件的路径/lib,/usr/lib；要增添额外搜索路径：在/etc/ld.so.conf.d/中创建以.conf为后缀名的文件，而后把要添加的路径直接写至此文件中

```
# ldconfig通知系统重新搜寻库文件
    -v：显示重新搜寻库的过程
```

3、头文件：输出给系统

默认：/usr/include

增添头文件搜寻路径，使用链接进行

将/usr/local/tengine/include/ 链接到/usr/include/目录下

两种方式

```
ln -sv /usr/local/tengine/include/* /usr/include/   或
ln -sv /usr/local/tengine/include   /usr/include/tengine
```

4、man文件路径：安装在--prefix指定的目录下的man目录：/usr/share/man

1、man -M /path/to/man_dir command

2、在/etc/man_db.conf 中添加一条MANPATH

删除CentOS更新后的旧内核

1.查看系统当前内核版本:

```
# uname -a
```

2.查看系统中全部的内核RPM包:

```
# rpm -qa | grep kernel
```

3.删除旧内核的RPM包

```
yum remove kernel-2.6.18-194.el5
yum remove kernel-devel-2.6.18-194.el5
```

netstat命令

-r:显示路由表

-n:以数字方式显示

-t:建立的tcp链接

-u:显示udp链接

-l:显示监听状态的链接 (多个选项可以同时使用)

-p:显示监听指定的套接字的进程的ID号及进程名

```
# netstat -tulnp 显示tcp链接udp链接监听状态链接及进程id等所有信息
```