linux

https://www.php.cn/linux/linux-system-contents.html

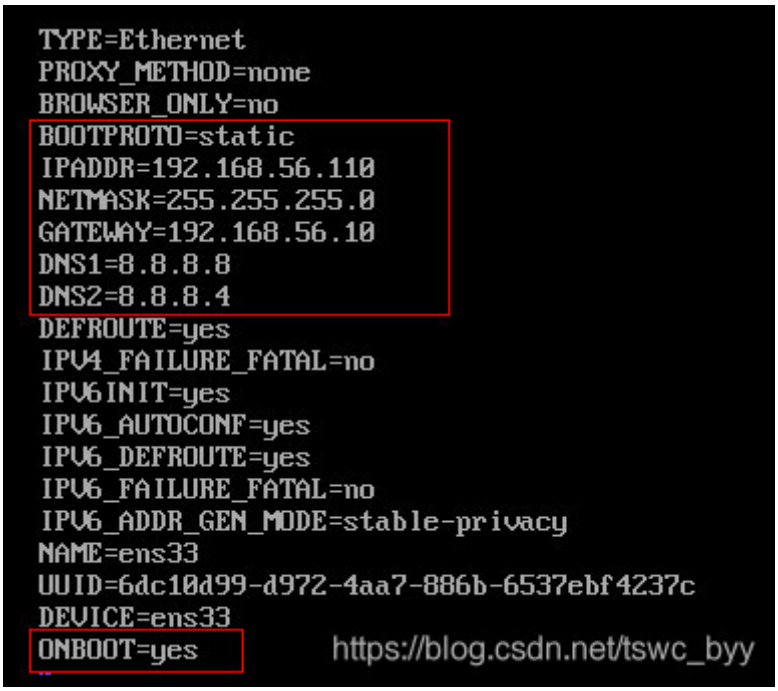
https://www.php.cn/linux-421910.html

https://blog.csdn.net/qq_36119192/article/details/82752515

配置信息

1、网络配置

```
vi /etc/sysconfig/network-scripts/ifcfg-ens33
```



在修改网络配置后需要重启网络

```
service network restart
```

能够ping通自己的ip则证明网络配置成功

```
ping 192.168.137.110
```

2、关闭防火墙

查看防火墙状态

```
systemctl status firewalld.service
```

关闭防火墙

```
systemctl stop firewalld.service
```

永久关闭防火墙

```
systemctl disable firewalld.service
```

3、设置主机名

需要修改瞬态主机名以及静态主机名，需要注意的是linux对大小写敏感，需要注意规范书写

修改瞬态主机名：

```
vi /etc/sysconfig/network
```

追加内容

```
NETWORKING=yes
HOSTNAME=master      #此处为你想设置的主机名
```

修改静态主机名

```
vi /etc/hostname
```

将原内容全部删除 加上自己的主机名

```
master
```

4、hosts设置

```
vi /etc/hosts
#在文本后追加
192.168.137.110  master
```

5、免密钥登录配置

```
ssh-keygen -t rsa
```

四下回车，然后进入ssh文件夹

```
cd ~/.ssh
```

查看文件内容

```
[root@master ~]# cd ~/.ssh
[root@master .ssh]# ls -l
total 8
-rw-------. 1 root root 1675 Nov 26 17:34 id_rsa    #密钥
-rw-r--r--. 1 root root  393 Nov 26 17:34 id_rsa.pub  #公钥
```

将公钥文件发送给自己

```
ssh-copy-id -i id_rsa.pub root@master
```

验证是否成功（若不需要输入密码则代表成功）

```
ssh master
```

linux系统启动过程

Linux系统的启动分为5个阶段

1、内核的引导

操作系统接管硬件以后，首先读入 /boot 目录下的内核文件

2、运行init

init 进程是系统所有进程的起点

Linux允许为不同的场合，分配不同的开机启动程序，这就叫做"运行级别"（runlevel）。也就是说，启动时根据"运行级别"，确定要运行哪些程序.Linux一共有7个运行级别0~6

运行级别0：系统停机状态，系统默认运行级别不能设为0，否则不能正常启动

运行级别1：单用户工作状态，root权限，用于系统维护，禁止远程登陆

运行级别2：多用户状态(没有NFS)

运行级别3：完全的多用户状态(有NFS)，登陆后进入控制台命令行模式

运行级别4：系统未使用，保留

运行级别5：X11控制台，登陆后进入图形GUI模式

运行级别6：系统正常关闭并重启，默认运行级别不能设为6，否则不能正常启动

3、系统初始化

4、建立终端

5、用户登录系统

命令

查看命令及档案格式

```
man ls
info cd
#使用man、info查询不知道含义的指令或者档案格式
```

而如果想要架设一些其他的服务，或想要利用一整组软体来达成某项功能时，可到/usr/share/doc 底下查一查有没有该服务的说明档.

按键	进行工作
空白键	向下翻一页
[Page Down]	向下翻一页
[Page Up]	向上翻一页
[tab]	在node 之间移动，有node 的地方，通常会以* 显示。
[Enter]	当游标在node 上面时，按下Enter 可以进入该node 。
b	移动游标到该info 画面当中的第一个node 处
e	移动游标到该info 画面当中的最后一个node 处
n	前往下一个node 处
p	前往上一个node 处
u	向上移动一层
s(/)	在info page 当中进行搜寻
h, ?	显示求助选单
q	结束这次的info page

关机操作

```
#在进行关机操作前应先运行sync命令（将内存中的数据存到磁盘中）
[root@master opt]# sync
#现在关机
shutdown -h now
#在22：23关机
shutdown -h 22:23
#十分钟后关机
shutdown -h +10
#系统立刻重新开机
shutdown -r now
#20分钟后关机，并进行提示
shutdown -h +20 'the system will be shut down after 20 minuites'
#仅发出警告参数(同时会报无法解析时间范围：Failed to parse time specification)
shutdown -k "this system will be shutdown"
```

切换终端

当使用文字模式进入linux时，有6个终端（tty1~tty6）可以使用，切换命令为Ctrl+Alt+F1~F6

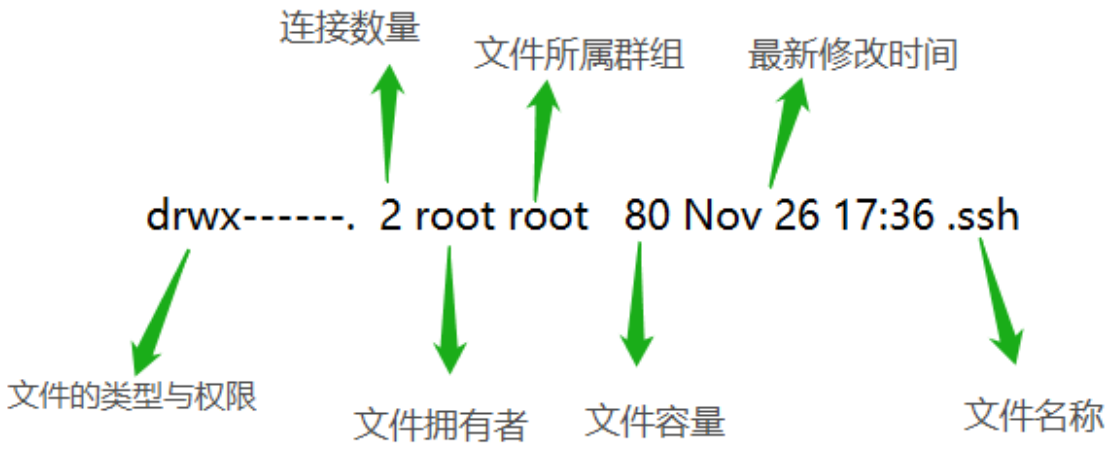
在man 的时候， man page 显示的内容中，指令(或档案)后面会接一组数字，这个数字若为 1, 5, 8 ，表示该查询的指令(或档案)意义为何?

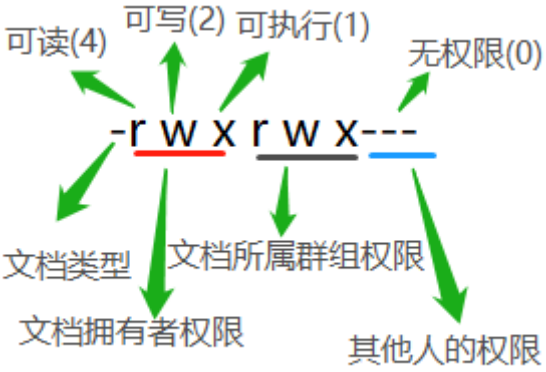
```
1：  一般使用者可以使用的指令或可执行档案  5：一些设定档的档案内容格式  8：系统管理员能够使用的管理指令。
```

文件操作

元件	内容	叠代物件	r	w	x
档案	详细资料data	文件资料夹	读到文件内容	修改文件内容	执行文件内容
目录	档名	可分类抽屉	读到档名	修改档名	进入该目录的权限(key)

文件的权限





示例:

dr-xr-xr-x. 17 root root 224 Nov 26 16:33 ..

为一个文件夹、文件权限为555，连接数量为17，所有者和所属群组为root,文件容量为244(**预设单位为**bytes**),最近的修改时间为11月26日16：33

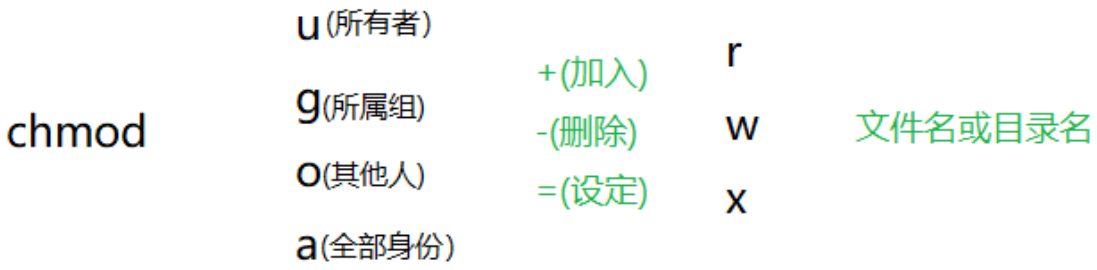
```
. ：代表当前的目录，也可以使用./ 来表示；
.. ：代表上一层目录，也可以../ 来代表。
如果文档名前面多了一个.则代表这个档案为隐藏文档
Linux档名的限制为：单一档案或目录的最大容许档名为255 个英文字元或128 个中文字元；
```

更改文件权限

```
chmod [-R] 权限值 文件名
#-R（注意是大写）选项表示连同子目录中的所有文件，也都修改设定的权限
```

示例

```
[root@master opt]# vi hell.txt
[root@master opt]# ll
total 4
-rw-r--r--. 1 root root 29 Nov 27 10:49 hell.txt
[root@master opt]# chmod  777 hell.txt
[root@master opt]# ll
total 4
-rwxrwxrwx. 1 root root 29 Nov 27 10:49 hell.txt
```



示例：

```
[root@master opt]# chmod u=rxw,go=rx hell.txt
[root@master opt]# ll
total 4
-rwxr-xr-x. 1 root root 29 Nov 27 10:49 hell.txt
```

更改文件拥有者

```
chown [-R]帐号名称:群组名称档案或目录
#-R : 进行递回(recursive)的持续变更，亦即连同次目录下的所有档案都变更
```

示例

```
[root@master opt]# chown bin hell.txt
[root@master opt]# ll
total 4
-rwxr-xr-x. 1 bin root 29 Nov 27 10:49 hell.txt
```

更改文件所属群组

```
chgrp [-R] dirname/filename
#-R : 进行递回(recursive)的持续变更，亦即连同次目录下的所有档案、目录都更新成为这个群组之意。
常常用在变更某一目录内所有的档案之情况。
```

示例：

```
[root@master opt]# chgrp users hell.txt
[root@master opt]# ll
total 4
-rwxr-xr-x. 1 bin users 29 Nov 27 10:49 hell.txt
```

将文件拥有者与所属群组改回root

```
[root@master opt]# chown root:root hell.txt
[root@master opt]# ll
total 4
-rwxr-xr-x. 1 root root 29 Nov 27 10:49 hell.txt
```

复制文件

```
cp 来源文档 目标文档
```

示例:

此操作是在来源文档的路径下进行的，可不写来源文档路径，但若不在来源文档路径下，需写出来源文档的路径

```
[root@master opt]# cp hell.txt /etc/hell.txt
[root@master opt]# cat /etc/hell.txt
hello word!
welocme to linux
```

创建文件夹

创建单级空文件夹

```
mkdir 名称
#示例:
mkdir test
```

创建多级文件夹

```
mkdir -p 名称/名称/名称/...
#示例:
mkdir -p test1/test2/test3
```

删除文件操作

删除单个文件夹

```
rmdir 名称
#示例:
rmdir test
```

删除多级文件夹

```
rmdir 名称/名称/名称/...
#示例：
rmdir test1/test2/test3
```

移除文档或目录

```
rm 文档或目录
可选参数：
-f ：就是force 的意思，忽略不存在的档案，不会出现警告讯息；
-i ：互动模式，在删除前会询问使用者是否动作
-r ：递回删除啊！最常用在目录的删除了！
```

```
#删除文档
rm test1.txt
#删除文件夹
rm -r test
```

查看文档内容

```
#从第一行开始显示文件内容（顺序）
[root@master opt]# cat hell.txt
hello word!
welocme to linux
#从最后一行开始显示文件内容（倒序）
[root@master opt]# tac hell.txt
welocme to linux
hello word!
#查看内容并输出行号
[root@master opt]# nl hell.txt
     1  hello word!
     2  welocme to linux
[root@master opt]# cat -n hell.txt
     1  hello word!
     2  welocme to linux
#一页一页的显示内容
more hello.txt
#一页一页的显示内容并且能够向前翻页
less hello.txt
#只看头几行
head -行数 文档名称
head -2 hello.txt
#只看尾几行
tail -行数 文档名称
tail -1 hell.txt
#以二进制的方式读取文档内容
od hell.txt
#取文档的指定范围行
示例： 20-30行
head -30 sdf.conf | tail -n
#取文档的指定范围行并显示正确的行号
cat -n sdf.conf |head -n 20|tail -n 10
#找到某个单词或字母或句子的ASCII对照
echo hello | od -t oCc
[root@master etc]# echo hello |od -t oCc
0000000 150 145 154 154 157 012
```

```
          h   e   l   l   o  \n
0000006
```

设置文档隐藏属性

```
chattr [+-=][ASacdistu]档案或目录名称
选项与参数：
+ ：增加某一个特殊参数，其他原本存在参数则不动。
- ：移除某一个特殊参数，其他原本存在参数则不动。
= ：设定一定，且仅有后面接的参数

A ：当设定了A 这个属性时，若你有存取此档案(或目录)时，他的存取时间atime 将不会被修改，
     可避免I/O 较慢的机器过度的存取磁碟。(目前建议使用档案系统挂载参数处理这个项目)
S ：一般档案是非同步写入磁碟的(原理请参考前一章sync的说明)，如果加上S这个属性时，
     当你进行任何档案的修改，该更动会『同步』写入磁碟中。
a ：当设定a 之后，这个档案将只能增加资料，而不能删除也不能修改资料，只有root 才能设定这属性
c ：这个属性设定之后，将会自动的将此档案『压缩』，在读取的时候将会自动解压缩，
     但是在储存的时候，将会先进行压缩后再储存(看来对于大档案似乎蛮有用的！)
d ：当dump程序被执行的时候，设定d属性将可使该档案(或目录)不会被dump 备份
i ：这个i 可就很厉害了！他可以让一个档案『不能被删除、改名、设定连结也无法写入或新增资料！』
     对于系统安全性有相当大的助益！只有root 能设定此属性
s ：当档案设定了s 属性时，如果这个档案被删除，他将会被完全的移除出这个硬碟空间，
     所以如果误删了，完全无法救回来了喔！
u ：与s 相反的，当使用u 来设定档案时，如果该档案被删除了，则资料内容其实还存在磁碟中，
     可以使用来救援该档案喔！
注意1：属性设定常见的是a 与i 的设定值，而且很多设定值必须要身为root 才能设定
注意2：xfs 档案系统仅支援AadiS 而已
```

```
#建立空文件夹
touch test
#赋予i属性后进行重命名操作
[root@master opt]# chattr +i test
[root@master opt]# mv test test2
mv: cannot move ‘test’ to ‘test2’: Operation not permitted
#去除i属性后进行重命名
[root@master opt]# chattr -i test
[root@master opt]# mv test test1
[root@master opt]# ls
hell.txt  test1
```

查看文件类型：

file 文件

```
[root@master opt]# file test1
test1: empty
[root@master opt]# file hell.txt
hell.txt: ASCII text
```

指定档名的搜索

```
[root@master opt]# which ifconfig
/usr/sbin/ifconfig
#可加参数 -a 可将所有由PATH目录中可以找到的指令均列出，而不止第一个被找到的指定命令
[root@master opt]# which -a .conf
/usr/bin/which: no .conf in
(/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin)
```

档案档名的搜索

```
[root@master opt]# whereis ifconfig
ifconfig: /usr/sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
[root@master opt]# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz
```

whereis的查询速度比find快，因为whereis只查询特定的目录，而非全系统查询

find

find [PATH] [option] [action]

```
选项与参数：
1. 与时间有关的选项：共有-atime, -ctime 与-mtime ，以-mtime 说明
   -mtime n ：n 为数字，意义为在n 天之前的『一天之内』被更动过内容的档案；
   -mtime +n ：列出在n 天之前(不含n 天本身)被更动过内容的档案档名；
   -mtime -n ：列出在n 天之内(含n 天本身)被更动过内容的档案档名。
   -newer file ：file 为一个存在的档案，列出比file 还要新的档案档名
```

示例：

将过去系统24小时内有变动过内容的档案列出

```
find / -mtime 0
```

压缩文件

将文档/文件进行压缩(gzip/bzip2/xz)

bzip2的压缩率要高于gzip，但在压缩大文件时，bzip2所需时间要高于gzip,xz的压缩率最高但所耗时间长

```
#使用gzip进行压缩
gzip 文件/文档名
-v ：可以显示出原档案/压缩档案的压缩比等信息
[root@master opt]# gzip -v test1
test1:  gzip: test1: Operation not permitted
  0.0% -- replaced with test1.gz
[root@master opt]# ls
hell.txt  test1  test1.gz
#在对文档进行压缩时，会直接覆盖原文档
[root@master opt]# gzip -v hell.txt
hell.txt:    -6.9% -- replaced with hell.txt.gz
[root@master opt]# ls
hell.txt.gz  test1  test1.gz
#使用bzip2进行压缩
[root@master opt]# bzip2 -v hell.txt
  hell.txt:  0.446:1, 17.931 bits/byte, -124.14% saved, 29 in, 65 out.
```

```
[root@master opt]# ls
hell.txt.bz2  test1
```

读取压缩后的文档内容

```
#使用gzip
gcat 压缩文档名称
[root@master opt]# zcat hell.txt.gz
hello word!
welocme to linux
#使用bzip2
bzcat 压缩文档名称
[root@master opt]# bzcat hell.txt.bz2
hello word!
welocme to linux
```

查找压缩文档内的内容

```
#使用gzip
gcat '内容' 压缩文档名称
[root@master opt]# zgrep 'hell' hell.txt.gz
hello word!
#查找内容并显示行号
[root@master opt]# zgrep -n 'o' hell.txt.gz
1:hello word!
2:welocme to linux
#使用bzip2
[root@master opt]# bzgrep 'h' hell.txt.bz2
hello word!
[root@master opt]# bzgrep -n 'e' hell.txt.bz2
1:hello word!
2:welocme to linux
```

解压缩已压缩的文档

```
gzip -d 压缩文件名称
[root@master opt]# gzip -d hell.txt.gz
[root@master opt]# ls
hell.txt  test1
```

打包指令

-z : 使用 gzip 来压缩和解压文件

-v : --verbose 详细的列出处理的文件

-f : --file=ARCHIVE 使用档案文件或设备，这个选项通常是必选的

-c : --create 创建一个新的归档（压缩包）

-x : 从压缩包中解出文件

```
#压缩
tar -zcvf 压缩后的压缩名   目标压缩文件名
[root@master opt]# tar -zcvf test1.gz hell.txt
hell.txt
[root@master opt]# ls
hell.txt  test1  test1.gz
#解压缩
tar -zxvf 压缩文件名
[root@master opt]# tar -zxvf test1.gz
hell.txt
[root@master opt]# ls
hell.txt  test1  test1.gz
```

zip文件

压缩为zip文件

zip 压缩后的文件名.zip 目标压缩文件名

```
zip te.zip test1
```

解压zip文件

unzip 目标解压文件名

目录存放

目录	应放置档案内容
第一部份：FHS 要求必须要存在的目录	
/bin	系统有很多放置执行档的目录，但/bin比较特殊。因为/bin放置的是在单人维护模式下还能够被操作的指令。 在/bin底下的指令可以被root与一般帐号所使用，主要有：cat, chmod, chown, date, mv, mkdir, cp, bash等等常用的指令。
/boot	这个目录主要在放置开机会使用到的档案，包括Linux核心档案以及开机选单与开机所需设定档等等。 Linux kernel常用的档名为：vmlinuz，如果使用的是grub2这个开机管理程式，则还会存在/boot/grub2/这个目录喔！
/dev	在Linux系统上，任何装置与周边设备都是以档案的型态存在于这个目录当中的。你只要透过存取这个目录底下的某个档案，就等于存取某个装置啰～比要重要的档案有/dev/null, /dev/zero, /dev/tty , /dev/loop, */ dev/sd*等等
/etc	系统主要的设定档几乎都放置在这个目录内，例如人员的帐号密码档、各种服务的启始档等等。一般来说，这个目录下的各档案属性是可以让一般使用者查阅的，但是只有root有权力修改。FHS建议不要放置可执行档(binary)在这个目录中喔。比较重要的档案有： /etc/modprobe.d/, /etc/passwd, /etc/fstab, /etc/issue等等。另外FHS还规范几个重要的目录最好要存在/etc/目录下喔：/etc/opt(必要)：这个目录在放置第三方协力软体/opt的相关设定档/etc/X11/(建议)：与X Window有关的各种设定档都在这里，尤其是xorg.conf这个X Server的设定档。/etc/sgml/(建议)：与SGML格式有关的各项设定档/etc/xml/(建议)：与XML格式有关的各项设定档
/lib	系统的函式库非常的多，而/lib放置的则是在开机时会用到的函式库，以及在/bin或/sbin底下的指令会呼叫的函式库而已。什么是函式库呢？妳可以将他想成是『外挂』，某些指令必须要有这些『外挂』才能够顺利完成程式的执行之意。另外FHS还要求底下的目录必须要存在：/lib/modules/：这个目录主要放置可抽换式的核心相关模组(驱动程式)喔！
/media	media是『媒体』的英文，顾名思义，这个/media底下放置的就是可移除的装置啦！包括软碟、光碟、DVD等等装置都暂时挂载于此。常见的档名有：/media/floppy, /media/cdrom等等。
/mnt	如果妳想要暂时挂载某些额外的装置，一般建议妳可以放置到这个目录中。在古早时候，这个目录的用途与/media相同啦！只是有了/media之后，这个目录就用来暂时挂载用了。
/opt	这个是给第三方协力软体放置的目录。什么是第三方协力软体啊？举例来说，KDE这个桌面管理系统是一个独立的计画，不过他可以安装到Linux系统中，因此KDE的软体就建议放置到此目录下了。另外，如果妳想要自行安装额外的软体(非原本的distribution提供的)，那么也能够将你的软体安装到这里来。不过，以前的Linux系统中，我们还是习惯放置在/usr/local目录下呢！
/run	早期的FHS 规定系统开机后所产生的各项资讯应该要放置到/var/run 目录下，新版的FHS 则规范到/run 底下。由于/run 可以使用记忆体来模拟，因此效能上会好很多！

目录	应放置档案内容
/sbin	Linux有非常多指令是用来设定系统环境的，这些指令只有root才能够利用来『设定』系统，其他使用者最多只能用来『查询』而已。 放在/sbin底下的为开机过程中所需要的，里面包括了开机、修复、还原系统所需要的指令。 至于某些伺服器软体程式，一般则放置到/usr/sbin/当中。至于本机自行安装的软体所产生的系统执行档(system binary)，则放置到/usr/local/sbin/当中了。常见的指令包括：fdisk, fsck, ifconfig, mkfs等等。
/srv	srv可以视为『service』的缩写，是一些网路服务启动之后，这些服务所需要取用的资料目录。常见的服务例如WWW, FTP等等。举例来说，WWW伺服器需要的网页资料就可以放置在/srv/www/里面。不过，系统的服务资料如果尚未要提供给网际网路任何人浏览的话，预设还是建议放置到/var/lib 底下即可。
/tmp	这是让一般使用者或者是正在执行的程序暂时放置档案的地方。这个目录是任何人都能够存取的，所以你需要定期的清理一下。当然，重要资料不可放置在此目录啊！因为FHS甚至建议在开机时，应该要将/tmp下的资料都删除唷！
/usr	第二层FHS 设定，后续介绍
/var	第二层FHS 设定，主要为放置变动性的资料，后续介绍

目录	应放置档案内容
第一部份：FHS 要求必须要存在的目录	
/usr/bin/	所有一般用户能够使用的指令都放在这里！目前新的CentOS 7 已经将全部的使用者指令放置于此，而使用连结档的方式将/bin 连结至此！也就是说， /usr/bin 与/bin 是一模一样了！另外，FHS 要求在此目录下不应该有子目录！
/usr/lib/	基本上，与/lib 功能相同，所以/lib 就是连结到此目录中的！
/usr/local/	系统管理员在本机自行安装自己下载的软体(非distribution预设提供者)，建议安装到此目录， 这样会比较便于管理。举例来说，你的distribution提供的软体较旧，你想安装较新的软体但又不想移除旧版， 此时你可以将新版软体安装于/usr/local/目录下，可与原先的旧版软体有分别啦！你可以自行到/usr/local去看看，该目录下也是具有bin, etc, include, lib...的次目录喔！
/usr/sbin/	非系统正常运作所需要的系统指令。最常见的就是某些网路伺服器软体的服务指令(daemon)啰！不过基本功能与/sbin 也差不多， 因此目前/sbin 就是连结到此目录中的。
/usr/share/	主要放置唯读架构的资料档案，当然也包括共享文件。在这个目录下放置的资料几乎是不分硬体架构均可读取的资料， 因为几乎都是文字档案嘛！在此目录下常见的还有这些次目录：/usr/share/man：线上说明文件/usr/share/doc：软体杂项的文件说明/usr/share/zoneinfo：与时区有关的时区档案
第二部份：FHS 建议可以存在的目录	
/usr/games/	与游戏比较相关的资料放置处
/usr/include/	c/c++等程式语言的档头(header)与包含档(include)放置处，当我们以tarball方式(*.tar.gz 的方式安装软体)安装某些资料时，会使用到里头的许多包含档喔！
/usr/libexec/	某些不被一般使用者惯用的执行档或脚本(script)等等，都会放置在此目录中。例如大部分的X视窗底下的操作指令，很多都是放在此目录下的。
/usr/lib/	与/lib/功能相同，因此目前/lib 就是连结到此目录中
/usr/src/	一般原始码建议放置到这里，src有source的意思。至于核心原始码则建议放置到/usr/src/linux/目录下。

vim编辑器

进入编辑模式

```
i/a/o/I/A/O
```

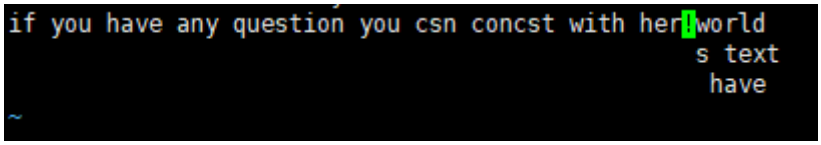
退出编辑（先执行esc，推出编辑模式）

注意：vim对大小写敏感

```
#退出编辑页码
:q
#保存内容并退出编辑页
:wq
#强制退出（进行了编辑，但不想保存，且权限不够）
:q!
#强制保存并退出(操作权限不够)
:wq!
#强制保存
:w!
```

常用的编辑命令

命令	涵义
dd	删除游标所在的行
ndd	删除游标向下n行
gg	将光标跳到文档开头
G	将光标条道文末
yy	复制游标所在的行
p/P	粘贴复制的内容到游标的下一行/上一行
nyy	复制游标向下的n行
/word	向游标之下寻找字符串
?word	向游标之上寻找字符串
n/N	重复向下/向上搜索
x/X	向前/向后删除字元（游标处）
nx	连续向后删除n个字元
n	n为数字。游标向下移动n列
:n1,n2s/word1/word2/g	n1与n2为数字。在第n1与n2列之间寻找word1这个字串，并将该字串取代为word2！举例来说，在100到200列之间搜寻vbird并取代为VBIRD则：『:100,200s/vbird/VBIRD/g』。
:1,$s/word1/word2/g	从第一列到最后一列寻找word1字串，并将该字串取代为word2！
:1,$s/word1/word2/gc	从第一列到最后一列寻找word1字串，并将该字串取代为word2！且在取代前显示提示字元给使用者确认(confirm)是否需要取代！
u	**复原**前一个动作。
:set nu	显示行号，设定之后，会在每一列的字首显示该列的行号
:set nonu	与set nu 相反，为取消行号！
ctrl+v	先择区块，之后按y即复制，将光标移动到想要粘贴的位置按p粘贴



多视窗情况下的按键功能	
:sp [filename]	开启一个新视窗，如果有加filename， 表示在新视窗开启一个新档案，否则表示两个视窗为同一个档案内容(同步显示)。
[ctrl]+w+ j [ctrl]+w+↓	按键的按法是：先按下[ctrl] 不放， 再按下w 后放开所有的按键，然后再按下j (或向下方向键)，则游标可移动到下方的视窗。
[ctrl]+w+ k [ctrl]+w+↑	同上，不过游标移动到上面的视窗。
[ctrl]+w+ q	其实就是:q 结束离开啦！举例来说，如果我想要结束下方的视窗，那么利用[ctrl]+w+↓ 移动到下方视窗后，按下:q 即可离开， 也可以按下[ctrl]+w+q 啊！
:e!	**将文档恢复为初始状态**

head

```
head -n number filename
```

使用head命令若不加参数则默认显示前十行

```
#查看文件前3行
head -n 3 helloText
#显示文件名
[root@master opt]# head -v helloText
==> helloText <==
hello linux:
#退出编辑页码
:q
#保存内容并退出编辑页
:wq
#强制退出（进行了编辑，但不想保存，且权限不够）
:q!
#强制保存并退出(操作权限不够)
:wq!
#强制保存
#不显示文件名
[root@master opt]# head -p helloText
#显指文件前指定字符数内容
[root@master opt]# head -c 10 helloText
hello linu[root@master opt]#
```

文件控制

cp

复制命令

```
#复制文件到指定文件
cp 参数 源文件对象 目标文件对象
cp hello.txt hell.txt
#强行复制文件或目录
cp -f 源文件对象 目标文件对象
#复制前进行询问
cp -i hello.txt hell.txt
[root@master opt]# cp -i hello.txt hell.txt
cp: overwrite ‘hell.txt’?
#对源文件建立硬连接而非复制文件
[root@master opt]# cp -l hello.txt hell.txt
cp: overwrite ‘hello.txt’? y
#保留源文件或目录的属性
cp -p hello.txt hell.txt
#显示执行详情
cp -v hello.txt hell.txt
#当文件的更改时间或对应名称不存在时才进行复制
cp -u hello.txt hell.txt
#将文件、目录复制给其他虚拟机
```

scp

对于不同主机之间的复制操作

```
#将当前主机文件复制给其他主机
scp 源文件路径 远程用户名@IP地址：远程目标文件的绝对路径
scp /opt/hello.txt root@192.168.137.111:/opt/
#将其他主机的文件复制到当前主机
scp 远程用户名@IP地址：远程目标文件的绝对路径 当前主机的目标路径
```

touch

功能：

1、如果文件不存在，则新建文件夹

2、如果文件存在，则更新时间标签

```
# 只更改读取时间（使用ls/ll中的--full-time显示的是文件的修改时间 使用-a参数较少）
touch -a hell.txt
[root@master vitest]# ls --full-time
total 16
-rw-r--r--. 1 root root 106 2020-11-28 16:12:07.635332935 +0800 hello
-rw-r--r--. 1 root root 107 2020-11-28 16:24:16.033358292 +0800 helloText
-rwxr-xr-x. 1 root root  29 2020-11-30 15:36:11.403308510 +0800 hell.txt
-rw-r--r--. 1 root root  92 2020-11-28 15:16:59.585217773 +0800 text
[root@master vitest]# touch -a hell,txt
[root@master vitest]# ls --full-time
total 16
-rw-r--r--. 1 root root 106 2020-11-28 16:12:07.635332935 +0800 hello
-rw-r--r--. 1 root root 107 2020-11-28 16:24:16.033358292 +0800 helloText
-rw-r--r--. 1 root root   0 2020-11-30 15:52:57.711343542 +0800 hell,txt
-rwxr-xr-x. 1 root root  29 2020-11-30 15:36:11.403308510 +0800 hell.txt
-rw-r--r--. 1 root root  92 2020-11-28 15:16:59.585217773 +0800 text
#更改文件被修改时间（使用ls/ll中的--full-time显示的是文件的修改时间）
[root@master vitest]# touch -m hell.txt
```

```
[root@master vitest]# ls --full-time hell.txt
-rwxr-xr-x. 1 root root 29 2020-11-30 16:02:11.976362837 +0800 hell.txt
#不创建任何文件或目录
[root@master vitest]# touch -c /opt/hell
[root@master vitest]# ls
hello  helloText  hell,txt  hell.txt  text
#解析时间字符串并使用它代替当前时间
touch -d hell.txt 12:30:55
```

mv

1、重命名

2、移动

```
#更改文件名
mv 原文件名 修改文件名
[root@master vitest]# mv hell.txt helloword.txt
[root@master vitest]# ls
hello  helloText  helloword.txt  hell,txt  text
#移动文件/文件夹
[root@master vitest]# mv helloword.txt /opt
[root@master vitest]# ls ../
hellolinux  helloText  hello.txt  helloword.txt  hell.txt  test1  test1.gz
vistest  vitest
#-i 参数，当目标地址有该文件时则会询问用户是否覆盖，若没有则直接移动
mv -i text /opt
#当源文件比目标文件新或者目标文件不存在时才执行此操作
mv -u text /opt
#当目标地址中与源文件/目录重复，则进行覆盖
mv -f text /opt
#显示详细信息
[root@master vitest]# mv -v text /opt
mv: overwrite ‘/opt/text’? y
‘text’ -> ‘/opt/text’
#若需要覆盖文件则先进行备份再进行覆盖
mv --backup
```

mkdir

mkdir 参数 目标文件

```
#创建文件夹
mkdir data
#创建文件并同时设定权限
mkdir -m 权限 文件夹名称
#若创建的目录上级不存再则会一并创建上级目录
mkdir -p /opt/dat/vi
[root@master vitest]# ls ../
dat  hellolinux  helloText  hello.txt  helloword.txt  hell.txt  test1  test1.gz
text  vistest  vitest
```

rm

删除文件夹/目录、

```
#删除文件夹
rm -r 目标文件夹名称
rm -r dat
#强制删除文件夹
rm -rf 目标文件夹
rm -rf /opt/dat
#删除空文件夹
rm -d /dat
```

rmdir

删除空文件夹

```
rmdir 目标文件夹名称
rmdir ds
rmdir -p 删除指定目录后 若父目录为空则一并删除
rmdir -p /opt/dat/sd
```

grep

查找文件中符合条件的字符串

```
#以递归的方式查找符合条件的文件（例：查找此目录下存在hello的文件）
grep -r 字符串
[root@master opt]# grep -r hello
hell.txt:hello word!
Binary file .helloText.swp matches
hellolinux:hello linux:
vitest/hello:#hello world
vitest/helloText:#hello world
helloText:hello linux:
vistest:hello linux:
hello.txt:hello linux:
helloword.txt:hello word!
#查找文件中包含查找字符的行
grep 查找的字符串 文件名
[root@master opt]# grep hello helloText
hello linux:
#以递归的方式查找不符合条件的文件
grep -v 字符串 目标文件
[root@master opt]# grep -v hello hell.txt
welocme to linux
#统计文件中包含字符串的行数
grep -c "字符串" 文件名称
[root@master opt]# grep -c "hello" hell.txt
1
#输出包含该字符串的行数及其内容
grep "字符串" -n 文件名称
[root@master opt]# grep "hello" -n hell.txt helloText
hell.txt:1:hello word!
helloText:1:hello linux:
#将匹每一个配项单独一行输出
grep -o "字符串" 文件名称
[root@master opt]# grep -o "he" helloText
he
he
```

```
#统计某一字符在文件中出现的次数
grep -o "字符串" 文件名称 | wc -l
[root@master opt]# grep -o "he" helloText | wc -l
2
#忽略大小写
gerp -i "字符串" 文件名
[root@master opt]# grep -i "hE" helloText
hello linux:
if you have any question you can concat with her!
#忽略大小写，并输出出现行数
[root@master opt]# grep -ic "hE" helloText
2
```

ps

查看进程

```
#查看一个终端中的所有进行
ps -a
#查看系统中的所有进程
ps -aux
#查看所有进行程
ps -e
#查看当前按shell产生的进程
ps -l
#查看进程数量
ps aux | wc -l
```

find

在目录中查找文件

tips:

ll 不是命令 应写作（ls -l)
ls -l 命令默认显示的文件大小单位为byte
若想显示单位则可使用ls -lh命令
若想指定显示单位则可以使用ls -l block-size=单位名称 如 ls -l --block-size=m

```
#根据文件名称
find 路径 -name 文件名称
[root@master /]# find /opt/ -name hell.txt
/opt/hell.txt
#不区分大小写查找文件
find 路径 -iname 文件名名称
[root@master opt]# find /opt -iname HEllo.txt
/opt/hello.txt
#根据大小进行查找
在find命令中的-size 参数中默认单位为b，其大小为512byte
#查找大小为2b的文件
[root@master opt]# find /opt  -size 2
/opt/helloText
[root@master opt]# find /opt  -size 2b
/opt/helloText
#查找大小小于1b的文件、
[root@master opt]# find /opt -size -1b
```

```
/opt/test1
#查找大小大于7b的文件
find /opt -size +7b
#查找大小大于3b小于6b的文件
find /opt -size -3b -size + 7b
#根据权限查找
find 路径 -perm 权限
fins /opt -perm 777
#根据用户名称/所属群组进行查找
find 路径 -user 用户名
find /opt -user root
find 路径 -group 群组名称
find /opt -group root
#根据时间进行查找
#被更改过的文件
find -mtime 时间
//一天内被更改过的
find -mtime 1
//一天前被更改过的
find -mtime +1
#被访问的文件
find -atime 时间
#状态被改变的文件
find -ctime 时间
```

kill

关闭进程

```
kill -l<信息编号>
若没有信息编号列出全部的信息名称
```

用户/群组

创建

useradd 用户名称　//创建新用户

passwd 用户名称　//更改用户密码

切换用户

su 用户名称

删除

userdel 用户名称 //删除用户

groupdel 群组名称　//删除群组

shell

执行shell的三种方式

1、bash shell文件名称

　bash hello.sh

2、sh shell文件名称

sh hello.sh

3、./shell文件名称(需要开通权限)

./hello.sh

简单的输出脚本

```
#!/bin/bash
#!是一个约定的标记，告诉系统这个脚本需要什么解释器来执行，即使用哪一种shell

示例：
#创建.sh文件
vi hello.sh
#文件内写入内容
#!/bin/bash
echo "hello linux"
#执行文件
bash hello.sh
```

输出提示信息

```
read -p "提示内容" s
```

shell变量

在定义变量时，变量名不加美元符号(php语言中需要)

注意：

1、变量名只能使用英文字母、数字和下划线，且首字母不能以数字开头

2、中间不能有空格（尤其是等号左右），可以使用下划线

3、不能使用标点符号

4、不能使用bash里面的关键字

变量的使用

对于已经定义过的变量，只要在变量名前面加美元符号即可

对于变量名外的跨括号可加可不加，加花括号是为了帮助解释器识别变量的边界

```
#!/bin/bash
user_name="superman"
echo $user_name
echo "my name is ${user_name}, you can not beat me"

[root@master shell]# bash test.sh
superman
my name is superman, you can not beat me
```

只读变量

readonly 变量名称

对于只读变量不能再次进行修改

```
#!/bin/bash
url=www.bing.com
readonly url
url=www.gogle.com //对于变量进行再次赋值

[root@master shell]# bash readonly.sh
readonly.sh: line 4: url: readonly variable  //赋值失败
```

删除变量

unset 变量名称

```
#!/bin/bash
url=https://www.cnblogs.com/shiningrain/
unset url
echo ${url}

[root@master shell]# bash del.sh
```

变量类型

1、局部变量：在脚本或命令中定义，仅在当前shell示例中有效，其他shell启动的程序不能访问局部变量

2、环境变量：所有的程序，包括shell启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。必要的时候shell甲苯也可以定义环境变量

3、shell变量：shell变量是由shell程序设置的特殊变量。shell变量中有一部分是环境变量，有一部分是局部变量，这些变量保证了shell的正常运行。

shelll字符串

对于字符串可以使用单引号 也可以使用双引号，也可以不用引号（不使用引号要注意不能含有空格）

双引号的优点：

双引号例可以有变量

双引号里面可以出现转义字符

```
#!/bin/bash
str=this_is_a_string
str1='this is a string'
str2="this is a string"
echo $str
echo ${str1}
echo "this is what i write ${str2}"

[root@master shell]# bash str.sh
this_is_a_string
this is a string
this is what i write this is a string
```

拼接字符串

使用双引号拼接，使用单引号拼接

```
#!/bin/bash
your_id="SpongeBob"
#使用双引号拼接
greeting="hello,"$your_id"!"
greeting1="hello,${your_id}!"
echo $greeting $greeting1
#使用单引号拼接
greeting2='hello,'$your_id'!'
greeting3='hello,${your_id}!'
echo $greeting2 $greeting3

[root@master shell]# bash concat.sh
hello,SpongeBob! hello,SpongeBob!
hello,SpongeBob! hello,${your_id}!
```

获取字符串长度

echo ${#字符串名称}

示例:

echo ${#your_id}

提取子字符串

echo ${string:起始位置:结束位置}（起始位置的id默认为从0开始而非1）

示例

```
#!/bin/bash
string="abc"
echo ${string:1:2}

#!/bin/bash
bc
```

查找子字符串

```
eccho` expr index "$字符串名称" 搜索的子字符串`（此处echo后的为``不是单引号
echo `expr index "$greeting1" ho`
```

数组

创建数组的三种方式

1、数组名称（value1 value2 value3...）数组值以空格分开

2、数组名称（

value1

value2

value3

.....

)

3、数组名称[下标]=value

读取数组

${数组名称[下标]}

使用@符号可以获取数组中的所有元素

echo ${array[@]}

```
#!/bin/bash
#创建数组
array1=(10 29 238 84 094 2 45 23)
array2=(1123
        5
        546534
        653
        5432)
array3[0]=11
array3[1]=230
array3[2]=34
array3[30]=90
#读取数组
echo ${array1[3]}
value=${array2[0]}
echo $value
echo ${array3[@]}

[root@master shell]# sh array.sh
84
1123
11 230 34 90
```

获取数组长度

变量名=${#数组名[@]}　变量名= ${#数组名[*]}

获取数组单个元素的长度

变量名=${#数组名[下标]}

```
#获取数组长度
length=${#array1[@]}
echo $length
length2=${#array2[*]}
echo $length2
#获取数组单个元素的长度
lengthn=${#array3[2]}
echo $lengthn
```

函数

```
[ function ] funname [()]

{
    action;
    [return int;]

}
```

```
#!/bin/bash
read -p "请输入参数a" a
read -p "请输入参数b" b
demofun(){
        if [[ a > b ]]
        then
                echo "$a > $b "
        elif [[ a < b ]]
        then
                echo "$a < $b "
        else
                echo "$a = $b"
        fi
        return $(($a + $b))
}
echo " 开始执行函数 "
demofun
echo " 两个数字之和为$? !"
echo " 函数执行结束 "
```

函数传递参数

```
#!/bin/bash
        funParam(){
        echo "第一个参数为 $1"
        echo "第二个参数为 $2"
        #错误写法
        echo "第十个参数为 $10"
        echo "第十个参数为 ${10}"
        echo "第十一个参数为 ${11}"
        echo "参数总数有 $# 个"
        echo "作为一个字符串输出所有参数 $* "
}
funParam 1 2 3 4 5 6 7 8 9 23 34
#$10不能获取第十个参数，获取第十个参数需要${10} 当n>=10时需要使用${n}来获取参数
```

传递参数

echo"$第几位参数 "（默认0 为执行文件名称）

计算参数的个数

echo"$#"

将传入的参数作为一个字符串进行输出

两种方式 $* $@(两者的区别在于$*仅传递了一个参数，而$@则传入了传参个数的参数)

```
#!/bin/bash
echo "shell传参实例"
echo "执行的文件名:$0";
echo "第一个参数:$1";
echo "第二个参数:$2";
echo "第三个参数:$3";
echo "第四个参数:$4";
echo "第五个参数:$5";
echo "参数个数为:$#";
echo "将传递的参数作为一个字符串进行输出 ：  $*";
echo "将传递的参数作为一个字符串进行输出 ：  $@";
#$*与$@的区别
for i in "$*";do
        echo $i
done

for j in "$@";do
        echo $j
done

[root@master shell]# sh parameter.sh 1 2 3 33 56
shell传参实例
执行的文件名:parameter.sh
第一个参数:1
第二个参数:2
第三个参数:3
第四个参数:33
第五个参数:56
参数个数为:5
将传递的参数作为一个字符串进行输出 ：  1 2 3 33 56
将传递的参数作为一个字符串进行输出 ：  1 2 3 33 56
1 2 3 33 56
1
2
3
33
56
```

基本运算符

在表达式与运算符之间需要以空格进行分隔

运算符	说明	举例
=	检测两个字符串是否相等，相等返回 true。	[$a = $b] 返回 false。
!=	检测两个字符串是否相等，不相等返回 true。	[$a != $b] 返回 true。
-z	检测字符串长度是否为0，为0返回 true。	[-z $a] 返回 false。
-n	检测字符串长度是否为0，不为0返回 true。	[-n $a] 返回 true。
str	检测字符串是否为空，不为空返回 true。	[$a] 返回 true。

```
`expr 变量1 运算符 变量二`
val=`expr 2 + 3`
echo $val
echo`expr 2 ^ 3`
```

```
#!/bin/bash
val=`expr 2 + 1`
echo "2 + 1 = $val"
echo "4 + 4 = `expr 4 + 4 ` "
echo "4 % 2 = `expr 4 % 2`"
echo "6 / 3 = `expr 6 / 3`"
echo "3 * 2 = `expr 3 \* 2`"
a=2
b=4
if [ $a == $b ]
then
echo "a等于b"
fi
if [ $a != $b ]
then
echo "a不等于b"
fi


[root@master shell]# sh cal.sh
2 + 1 = 3
4 + 4 = 8
4 % 2 = 0
6 / 3 = 2
3 * 2 = 6
a不等于b
```

关系运算符

关系运算符只支持数字，不支持字符串，除非字符串的值式数字

运算符	说明	举例
-eq	检测两个数是否相等，相等返回 true。	[$a -eq $b] 返回 false。
-ne	检测两个数是否不相等，不相等返回 true。	[$a -ne $b] 返回 true。
-gt	检测左边的数是否大于右边的，如果是，则返回 true。	[$a -gt $b] 返回 false。
-lt	检测左边的数是否小于右边的，如果是，则返回 true。	[$a -lt $b] 返回 true。
-ge	检测左边的数是否大于等于右边的，如果是，则返回 true。	[$a -ge $b] 返回 false。
-le	检测左边的数是否小于等于右边的，如果是，则返回 true。	[$a -le $b] 返回 true。

```
#!/bin/bash
a=45
b=22
c=22
echo "a = $a ,b = $b ,c = $c"
#两个变量是否相等
if [ $a -eq $b ]
then
echo "$a -eq $b : a 等于 b"
else
echo "$a -eq $b : a 不等于 b "
fi
#两个变量是否不相等
if [ $a -ne $b ]
then
echo "$a -ne $b : a与b不相等"
else
echo "$a -ne $b : a与b相等"
fi
#左边的数是否大于右边的数
if [ $a -gt $b ]
then
echo "$a -gt $b : a > b"
else
echo "$a -gt $b : a < b"
fi
#左边的数是否小于右边的数
if [ $a -lt $b ]
then
echo "$a -lt $b : a < b"
else
echo "$a -lt $b : a > b"
fi
#左边的数是否大于等于右边的数
if [ $b -ge $c ]
then
echo "$b -ge $c : b >= c"
else
echo " $b -ge $c : b < c"
fi
#左边的数是否小于等于右边的数
if [ $a -le $c ]
then
echo "$a -le $c : a <= c"
else
echo "$a -le $c : a > c"
fi

[root@master shell]# sh op.sh
a = 45 ,b = 22 ,c = 22
45 -eq 22 : a 不等于 b
45 -ne 22 : a与b不相等
45 -gt 22 : a > b
45 -lt 22 : a > b
22 -ge 22 : b >= c
45 -le 22 : a > c
```

布尔运算符

运算符	说明	举例
!	非运算，表达式为 true 则返回 false，否则返回 true。	[! false] 返回 true。
-o	或运算，有一个表达式为 true 则返回 true。	[$a -lt 20 -o $b -gt 100] 返回 true。
-a	与运算，两个表达式都为 true 才返回 true。	[$a -lt 20 -a $b -gt 100] 返回 false。

```
#!/bin/bash
a=22
b=10
echo "a = $a ,b = $b"
#非运算
if [ $a != $b ]
then
echo "$a != $b : 返回true"
else
echo "$a != $b : 返回false"
fi
#或运算
if [ $a -lt 20 -o $b -lt 20 ]
then
echo "$a -lt 20 -o $b -lt 20 : 返回true"
else
echo "$a -lt 20 -o $b -lt 20 : 返回false"
fi
#与运算
if [ $a -gt 10 -a $b -lt 20  ]
then
echo "$a -gt 10 -a $b -lt 20 : 返回true"
else
echo "$a -gt 10 -a $b -lt 20 : 返回false"
fi

[root@master shell]# sh boolean.sh
a = 22 ,b = 10
22 != 10 : 返回true
22 -lt 20 -o 10 -lt 20 : 返回true
22 -gt 10 -a 10 -lt 20 : 返回true
```

逻辑运算

运算符	说明	举例
&&	逻辑的 AND	[[$a -lt 100 && $b -gt 100]] 返回 false
||	逻辑的 OR	[[$a -lt 100 || $b -gt 100]] 返回 true

注意 逻辑运算符需要双层中括号

```
#!/bin/bash
a=22
b=15
c=24
echo $a ,$b ,$c
#逻辑运算与
if [[ $a -gt $b && $a -lt $c ]]
then
echo "$a -gt $b && $a -lt $c : true "
else
echo "$a -gt $b && $a -lt $c : false"
fi
#逻辑运算或
if [[ $a -gt $b || $a -gt $c ]]
then
echo "$a -gt $b || $a -gt $c : true"
else
echo "$a -gt $b || $a -gt $c : false"

[root@master shell]# sh logic.sh
22 ,15 ,24
22 -gt 15 && 22 -lt 24 : true
22 -gt 15 || 22 -gt 24 : true
```

字符串运算符

在对字符串运算符进行判断时，若字符串中含有空格，shell解析时会将其认为是多个参数

，在进行判断是不知道获取哪个参数，即会报错误：line 14: [: too many arguments

解决方法：在进行判断时将 $字符串名称 写作："$字符串名称"

运算符	说明	举例
=	检测两个字符串是否相等，相等返回 true。	[$a = $b] 返回 false。
!=	检测两个字符串是否相等，不相等返回 true。	[$a != $b] 返回 true。
-z	检测字符串长度是否为0，为0返回 true。	[-z $a] 返回 false。
-n	检测字符串长度是否不为 0，不为 0 返回 true。	[-n "$a"] 返回 true。
$	检测字符串是否为空，不为空返回 true。	[$a] 返回 true。

在检测字符串是否为空时，若字符串中间有空格则写作："$字符串名称"

文件测试运算符

操作符	说明	举例
-b file	检测文件是否是块设备文件，如果是，则返回 true。	[-b $file] 返回 false。
-c file	检测文件是否是字符设备文件，如果是，则返回 true。	[-c $file] 返回 false。
-d file	检测文件是否是目录，如果是，则返回 true。	[-d $file] 返回 false。
-f file	检测文件是否是普通文件（既不是目录，也不是设备文件），如果是，则返回 true。	[-f $file] 返回 true。
-g file	检测文件是否设置了 SGID 位，如果是，则返回 true。	[-g $file] 返回 false。
-k file	检测文件是否设置了粘着位(Sticky Bit)，如果是，则返回 true。	[-k $file] 返回 false。
-p file	检测文件是否是有名管道，如果是，则返回 true。	[-p $file] 返回 false。
-u file	检测文件是否设置了 SUID 位，如果是，则返回 true。	[-u $file] 返回 false。
-r file	检测文件是否可读，如果是，则返回 true。	[-r $file] 返回 true。
-w file	检测文件是否可写，如果是，则返回 true。	[-w $file] 返回 true。
-x file	检测文件是否可执行，如果是，则返回 true。	[-x $file] 返回 true。
-s file	检测文件是否为空（文件大小是否大于0），不为空返回 true。	[-s $file] 返回 true。
-e file	检测文件（包括目录）是否存在，如果是，则返回 true。	[-e $file] 返回 true。

echo命令

操作符	名称
\n	换行 echo "hell \n word"，使用前须在echo后加转义符 -e
\n	不换行，使用前须在echo后加转义符 -e
>文件名称	将显示结果定向至指定文件，注意此方法会将原文件的内容替换掉
>>文件名	将显示结果定向至指定文件，此方法为将内容追加到目标文件中
`date`	显示当前时间

```
echo "this is the second test" >> /opt/hello.txt
echo `date`
```

printf

printf比echo的移植性更好

```
printf format-string [arguments]
format-string 格式控制字符串
arguments 参数列表，如果没有arguments那么%s 用null代替，%d用0代替
```

```
#!/bin/bash
printf "hello,shell\n"
printf "%-10s %-8s %-4s\n" 姓名 性别 体重kg
printf "%-10s %-8s %-4.2f\n" 张三 女 59.34
printf "%-10s %-8s %-4.2f\n" 对三 男 66.56
printf "%-10s %-8s %-4.2f\n" 可高 男 55.6
printf "%-10s %-8s %-4.2f\n" 黄二万 女 45
#指定一个参数 但多出的参数仍旧会按格式输出
printf %s sd sjei djks
printf "%s \n" sd sdhj ui
printf "%s %s %s\n" s d g hs n
#如果没有arguments那么%s会用null代替，%d会用0代替
printf "%s and %d\n"

[root@master shell]# sh printf.sh
hello,shell
姓名      性别    体重kg
张三      女      59.34
对三      男      66.56
可高      男      55.60
黄二万    女      45.00
sdsjeidjkssd
sdhj
ui
s d g
hs n
 and 0
```

序列	说明
\a	警告字符，通常为ASCII的BEL字符
\b	后退
\c	抑制（不显示）输出结果中任何结尾的换行字符（只在%b格式指示符控制下的参数字符串中有效），而且，任何留在参数里的字符、任何接下来的参数以及任何留在格式字符串中的字符，都被忽略
\f	换页（formfeed）
\n	换行
\r	回车（Carriage return）
\t	水平制表符
\v	垂直制表符
\	一个字面上的反斜杠字符
\ddd	表示1到3位数八进制值的字符。仅在格式字符串中有效
\0ddd	表示1到3位的八进制值字符

流程控制

```
#!/bin/bash
#for循环
for str in "this is a string"
do
        echo $str
done

for loop in 1 2 3 4 5 6
do
        echo $loop
done
#if else与test命令连用
num1=$[2 * 3]
num2=$[2 + 3]
if test $[num1] -eq $[num2]
then
echo "两个数字相等"
else
echo "两个数字不相等"
fi
#while循环
a=2
while(($a < 10))
do
        echo $a
        let "a++"   //自加操作    let"a--"自减操作
done
```

统计词频

```
#截取字符串
${string: start :length}
#示例
var2=${s:4:2}
```

分割字符串

tr

tr命令的输出结尾没有换行符

使用tr替换命令

```
#此处是将空格转换为换行符（此方法并不适用于有标点符号的字符串）
[root@master shell]# echo "hello world" | tr " " "\n"
hello
world
```

-s 将列出的重复字符的每个输入序列替换为该字符的单次出现。

```
[root@master shell]# echo "hhelo dfdfwwwvers" | tr -s "hw"
helo dfdfwvers
```

-d删除指定字符

大小写转换

```
#方式一：
'a-z' 'A-Z'
[root@master shell]# cat hello.txt | tr 'a-z' 'A-Z'
THE DAY IS SUNNY THE THE
THE SUNNY IS IS
#方式二：
[:lower:] [:upper:] #注意两个[]之间有空格,该操作仅让输出格式改变，并未改变源文件
#将文件内容转换为大写格式且将文件内容追加到另一个文件中
cat hello.txt |tr [:lower:] [:upper:] >> terst.txt
[root@master shell]# cat terst.txt
THE DAY IS SUNNY THE THE
THE SUNNY IS IS
```

uniq

uniq -c 代表打印每行在文本中出现的此时

sort排序

-n 根据数字大小进行拍寻

-r 将排序结果逆向显示

```
echo "the day is sunny the the" | tr " " "\n" |uniq -c |sort -r
```

在使用uniq之前，最好先使用sort将分割的字符串进行分组，，-c原理是字符串相同则加一，如果不进行先排序的话将无法统计准确的数目

```
#错误结果
[root@master shell]# cat hello.txt|tr "\n" " "|xargs -d" " -n1|uniq -c
      1 the
      1 day
      1 is
      1 sunny
      3 the
      1 sunny
      2 is
#正确结果
[root@master shell]# cat hello.txt|tr "\n" " "|xargs -d" " -n1|sort|uniq -c
      1 day
      3 is
      2 sunny
      4 the
```

xargs

xargs可以将单行或多行文本输入转换为其他格式，例如多行变单行，单行变多行

xargs可以将空白以及换行统一使用空格替换

命令格式

xargs -item command

多行输入单行输出

```
[root@master shell]# cat hello.txt
the day is sunny the the
the sunny is is
[root@master shell]# cat hello.txt |xargs
the day is sunny the the the sunny is is
```

-n选项多行输出

```
[root@master shell]# cat hello.txt |xargs -n1  #n后面的数字为几则每行输出几个字符
the
day
is
sunny
the
the
the
sunny
is
is
[root@master shell]# cat hello.txt |xargs -n3
the day is
sunny the the
the sunny is
is
```

-d定义一个定界符

```
[root@master shell]# echo "hellovshujvsjdksv" |xargs -dv  #注意不要
```

```
hello shuj sjdks
#若存在转义符则可以通过但单引号将转义符引起来而达到预期效果
[root@master shell]# echo "hello\word\sdfs\dssa" |xargs -d'\'
hello word sdfs dssa
[root@master shell]# echo "hello\\word\\nsdfs\\ndssa" |xargs -d"\\"
hello word nsdfs ndssa
#对于想要分割的字符串为系统会认定为转义符的字符串，可进行反转义操作
[root@master shell]# echo "hello\nword\nsdfs\ndssa" |xargs -d"\\\n"
hello nword nsdfs ndssa
#对于换行等其他转义符 可以指使用进行划分
[root@master shell]# cat hello.txt
the day is sunny the the
the sunny is is
[root@master shell]# cat hello.txt|xargs -d"\n"
the day is sunny the the the sunny is is
```

sed

替换、删除、更新文件中的内容，以行为处理单位

格式:

sed command(这对每行要进行的处理) **file**(要处理的文件)

/2/d中的 d 表示删除，意思是说，只要某行内容中含有字符 2，就删掉这一行。（sed 所谓的删除都是在模式空间中执行的，不会真正改动 roc.txt 原文件

```
#删除指定行
sed '2d' hello.txt  #删除第二行
sed '2,5d' hello.txt  #删除第2~5行
sed '3,$d'   #删除第三行开始的所有内容（包括第三行）
#插入新文本\追加新文本
行号i(a) \新文本内容（反斜杠后换行） 文件名称
[root@master shell]# sed '2a\
this is the insert line' hello.txt
the day is sunny the the
the sunny is is
this is the insert line
[root@master shell]# sed '2i\
this is the insert line' hello.txt
the day is sunny the the
this is the insert line
the sunny is is
#替换文件内容 -c
[root@master shell]# sed '2c\
this is the new tetx' hello.txt
the day is sunny the the
this is the new tetx
#将指定行的内容写入文件中 (此操作会覆盖原文件内容)
sed -n '1,2w terst.txt' hello.txt
#打印文件内容
sed -n '1,3p' hello.txt  #n选项经常和 p 配合使用，其含义就是，输出那些匹配的行。
```

awk

行匹配语句awk ''只能使用单引号

```
#每行按空格或TAB分割
```

```
awk'{print $1,$4}'  #输出文本中的第一项和第四项，按所写的顺序输出
#格式化输出
[root@master shell]# awk '{printf"%-8s %-10s\n",$1,$4}' hello.txt
the      sunny
the      is
#指定分割字符
awk -F分割符号
实例：
awk -F, '{print$1,$3}' hello.txt                #按照逗号进行分割
awk -f'[, ]' '{print$1,$3}' hello.txt           #多个分割符，先使用逗号进行分割 再使用空格进行分割
awk 'BEGIN{FS=","} {print$1,$3}' hello.txt      #内建变量
#设置变量
awk -v变量名称=变量值
[root@master shell]# awk -va=1 '{print $1,$1+a}' hello.txt
the 1
the 1
2 3
#过滤
awk '过滤条件' 文件名称
awk '$1 > 2' hello.txt              #过滤第一列大于2的行
awk '$1 > 2 && $2=="day" {pritn$1,$2}' hello.txt    #过滤出第一列大于2且第二列为day的行
#指定输出分割符
[root@master shell]# awk '{print$1,$2,$3}' OFS="$" hello.txt        #使用$分割第一列、第二列、第三列
the$day$is
the$sunny$is
2$$
#匹配字符串
~ 表示模式开始。// 中是模式
[root@master shell]# awk '$2~/day/ ' hello.txt      #输出第第二列中含有day的行
the day is sunny the the
#忽略大小写
awk 'BEGIN{IGNORECASE=1} /所要匹配的字符/' 文件名称
[root@master shell]# awk 'BEGIN{IGNORECASE=1} /The/ ' hello.txt
the day is sunny the the
the sunny is is
#取反
[root@master shell]# awk '$1 !~ /the/ {print $1,$2}' hello.txt   #筛选第一列不为the的行
2
#awk脚本
BEGIN{执行前语句}
END{处理完所有的执行后要执行的语句}
{处理每一行索要执行的语句}
```

将字符串转换为数字

$((变量名称))

磁盘管理

df

检查文件西戎的磁盘空间占用情况

```
#查看全部磁盘的占用情况
df -h
#查看指定目录的磁盘占用情况
df -h 目录名称
```

du

检查磁盘空间使用情况

```
#查看当前目录每个文件夹的占用情况 （在其后加上目录名称即可查看指定目录）
du --max-depth=1 -h    #最后一行输出内容为统计占用多少磁盘
#计算文件大小
du -sh 文件目录
```

fdisk

磁盘分区

```
fdisk -l             #列出所有的磁盘情况
df /                 #找出系统中根目录所在的磁盘（重点是找磁盘文件名）
```

当root用户出现 Operation not permitted的问题时，首先使用lsattr查看文件的隐藏属性，再使用chattr -属性名称 去除属性，随后便能对文件进行正常操作

linux查看默认网关

```
netstat -r
```

对系统进行更新

```
yum -y update
```

检测tcp/udp端口情况

```
nmap -sTU localhost
```

将本地文件直接上传至linux

```
#查看系统自带的软件包信息
yum provides */rz
如果有Filename    : /usr/bin/rz 则进行下一步
yum -y install lrzsz
使用rz命令即可将本地文件上传至linux
将文件加载到本地
sz 文件名
```

根据传入的日期计算出所需要的日期

```
#!/bin/bash
read -p "请输入日期(例:20201208): " s
```

```
#echo "数字长度为${#s}"
#判断输入是否为纯数字且为8位
if [[ "$s" =~ ^[0-9]+$ && ${#s} == 8 ]]
then
#截取年份
var1=${s:0:4}
#截取月份
var2=${s:4:2}
#截取天
var3=${s:6:2}
var1=$(($var1))
var2=$((10#$var2))
var3=$((10#$var3))
#输出年、月、日
echo "$var1年$var2月$var3日"
#判断该年份为闰年还是平年的表达式
val1=`expr $var1 % 4`
val2=`expr $var1 % 100`
val3=`expr $var1 % 400`
#判断年份与月份是否为正确日期
if [[ $var1 -eq 0 || $var2 -lt 1 || $var2 -gt 12 ]]
then
    echo "无效日期"
    exit;
else
#判断是闰年还是平年
if [[ $val1 == 0 && $val2 != 0 || $val3 == 0 ]]
then
    year=1
        echo "$var1是闰年"
else
    year=2
        echo "$var1是平年"
fi
#判断日期中的天是否为正确日期
case $var2 in
    1|3|5|7|8|10|12)
        if [[ $var3 -gt 31 || $var3 -eq 0 ]]
        then
            echo "请输入正确的日期"
            exit
        else
        echo "当月第一天日期为：`date -d "$var1-$var2-01" "+%Y-%m-%d"`"
        printf "当月最后一天为：%.4d-%.2d-%.2d \n" $var1 $var2 31
        fi
        ;;
    4|6|9|11)
        if [[ $var3 -gt 30 || $var3 -eq 0 ]]
        then
            echo "请输入正确的日期"
            exit
        else
        printf "当月第一天为：%.4d-%.2d-%.2d \n" $var1 $var2 01
        printf "当月最后一天为：%.4d-%.2d-%.2d \n" $var1 $var2 30
        fi
        ;;
#当月份为2月时根据闰年和平年来计算当月信息
    2)
```

```
        if [[ $year -eq 1 ]]
        then
        if [[ $var3 -gt 29 || $var3 -eq 0 ]]
                then
                        echo "请输入正确的日期"
                        exit
                else
                        printf "当月第一天为：%.4d-%.2d-%.2d \n" $var1 $var2 01
                        printf "当月最后一天为：%.4d-%.2d-%.2d \n" $var1 $var2 29
        fi
        fi
        if [[ $year -eq 2 ]]
        then
             if [[ $var3 -gt 28 || $var3 -eq 0 ]]
                then
                        echo "请输入正确的日期"
                        exit
                else
                        printf "当月第一天为：%.4d-%.2d-%.2d \n" $var1 $var2 01
                        printf "当月最后一天为：%.4d-%.2d-%.2d \n" $var1 $var2 28
                fi
                fi
                ;;
esac
    #输出当天是星期几
    day=`date -d "$var1-$var2-$var3" +%A`
    echo "当天为星期：  $day"
    #上上个月最后一天为上月第一天-1
    day2=`date -d "$(date -d "- 1 month $var1-$var2-01") - 1 day" "+%Y-%m-%d"`
    echo "上上个月最后一天为：$day2"
    #输出下周的日期
    day3=`date -d "+ 1 week $var1-$var2-$var3" +%A`
    #根据下周的日期计算下周周一的日期
    if [ $day3 == Monday ]
    then
        mon=`date -d "+ 1 week $var1-$var2-$var3" "+%Y-%m-%d"`
        echo "下周周一日期为：$mon"
    elif [[ $day3 == Tuesday ]]
    then
        mon=`date -d "$(date -d "+ 1 week $var1-$var2-$var3") - 1 day" "+%Y-%m-
%d"`
                echo "下周周一的日期为：$mon"
    elif [[ $day3 == Wednesday ]]
        then
                mon=`date -d "$(date -d "+ 1 week $var1-$var2-$var3") - 2 day"
"+%Y-%m-%d"`
                echo "下周周一的日期为：$mon"
    elif [[ $day3 == Thursday ]]
        then
                mon=`date -d "$(date -d "+ 1 week $var1-$var2-$var3") - 3 day"
"+%Y-%m-%d"`
                echo "下周周一的日期为：$mon"
    elif [[ $day3 == Friday ]]
        then
                mon=`date -d "$(date -d "+ 1 week $var1-$var2-$var3") - 4 day"
"+%Y-%m-%d"`
                echo "下周周一的日期为：$mon"
    elif [[ $day3 == Saturday ]]
```

```
        then
                mon=`date -d "$(date -d "+ 1 week $var1-$var2-$var3") - 5 day"
"+%Y-%m-%d"`
                echo "下周周一的日期为：$mon"
    elif [[ $day3 == Sunday ]]
        then
                mon=`date -d "$(date -d "+ 1 week $var1-$var2-$var3") - 6 day"
"+%Y-%m-%d"`
                echo "下周周一的日期为：$mon"
fi
fi
else
    echo "请输入八位数日期！！"
    exit
fi
#输出该月的日历
cal $var3 $var2 $var1
```

DNS

dns域名解析服务，就是将域名和ip之间做相应的转换，利用TCP和UDP的的53号端口

DNS系统作用：

正向解析：根据域名查找对应的IP地址

反向解析：根据IP查找对应的域名

DNS服务器的分类：

rpm安装jdk

命令格式：

rpm -ivh rpm_package_name

“-i”参数指明是要安装这个package，而“-v”这个参数则使输出信息增加，“-h”表示在安装过程中显示hashes作为在安装过程的一个进度条。

在安装jdk时，若出现以下错误，需要查看所下载的安装包是否正确

```
warning: jdk-11.0.9_linux-aarch64_bin.rpm: Header V3 RSA/SHA256 Signature, key
ID ec551f03: NOKEY
Preparing...                          ################################# [100%]
    package jdk-11.0.9-2000:11.0.9-ga.aarch64 is intended for a different
architecture
```

例：

正确的包为：jdk-11.0.9_linux-x64_bin.rpm

错误包：jdk-11.0.9_linux-aarch64_bin.rpm

rpm会默认安装在/usr/java目录下

验证安装

```
[root@master jdk]# ll /usr/bin/java
lrwxrwxrwx. 1 root root 22 Dec  8 09:47 /usr/bin/java -> /etc/alternatives/java
[root@master jdk]# ll /etc/al
aliases       aliases.db    alternatives/
[root@master jdk]# ll /etc/alternatives/java
lrwxrwxrwx. 1 root root 29 Dec  8 09:47 /etc/alternatives/java -> /usr/java/jdk-
11.0.9/bin/java
[root@master jdk]# java -version
java version "11.0.9" 2020-10-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.9+7-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.9+7-LTS, mixed mode)
```

修改环境变量：

vi /etc/profile

```
JAVA_HOME=/usr/java/default
PATH=$PATH:$JAVA_HOME/bin
export JAVA_HOME PATH
```

使变量生效：source /etc/profile

输出JAVA_HOME、PATH查看变量是否生效成功

```
[root@master jdk]# echo $JAVA_HOME
/usr/java/default
[root@master jdk]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/usr/java/default/bi
n
```

搭建tomcat服务器

安装部署tomcat

https://blog.csdn.net/qq_21077715/article/details/85541685

配置tomcat自启

https://blog.csdn.net/qq_21077715/article/details/85474967

mysql安装

1、查看系统中是否有自带的mysql

```
#检查系统中是否有自带的mysql
rpm -qa mysql            #若什么都没有显示则代表没有
#如果有自带的mysql
rpm -e mysql    #普通删除模式，若提示有依赖的其他文件时，进入下一步
rpm -e --nodeps mysql   #强力删除模式
```

2、卸载mysql

如果第一步系统中没有自带的mysql则可忽略此步骤

```
service mysqld status   #查看Mysql状态
#出现以下状态则表示没有mysql服务(可忽略步骤3、4)
[root@master software]# service mysql status
Redirecting to /bin/systemctl status mysql.service
Unit mysql.service could not be found.
#若出现mysql服务已启动进入下一步
```

3、关闭mysql服务

```
service mysql stop
```

4、卸载MySQL

```
rpm -e --nodeps mysql
```

5、查找mysql相关的残留目录

```
whereis mysql
find / -name mysql
```

6、删除残留

```
rm -rf /usr/lib64/mysql
rm -rf /usr/share/mysql
```

7、删除mysql用户及用户组

```
id mysql            #查看mysql用户及用户组
userdel mysql       #删除mysql用户及用户组
```

8、下载mysql(此处为mysql8)

```
wget https://dev.mysql.com/get/Downloads/MySQL-8.0/mysql-8.0.11-linux-glibc2.12-x86_64.tar.gz
```

一般会默认下载到/uer/local/src目录下，将其移动到/usr/local/mysql，此处的安装路径为/opt/software/mysql/mysql8

9、解压MySQL

```
 tar -zxvf mysql-8.0.11-linux-glibc2.12-x86_64.tar.gz
```

10、对解压后的文件进行重命名

```
mv mysql-8.0.11-linux-glibc2.12-x86_64 mysql8
```

11、在mysql8创建data文件

```
mkdir data
```

12、初始化数据库

```
 ./bin/mysqld --initialize
 #成功后会提示 initializing of server has completed
```

13、在mysql8/support-files目录下创建my-config.cnf文件，并将文件复制道/etc/my.cnf

```
[root@master support-files]# touch my-config.cnf
 cp -a /opt/software/mysql/mysql8/support-files/my-config.cnf /etc/my.cnf
```

14、修改my.cnf文件

```
basedir = /usr/local/mysql8

datadir = /usr/local/mysql8/data

prot = 3306

socket = /tmp/mysql.sock

init_connect='SET NAMES utf8mb4'
```

15、建立服务

```
cp -a /usr/local/mysql8/support-files/mysql.server /etc/init.d/mysqld
```

16、添加系统服务

```
chmod +x /etc/rc.d/init.d/mysqld
chkconfig --add mysqld
```

17、查看是否添加成功

```
[root@master etc]# chkconfig --list

Note: This output shows SysV services only and does not include native
      systemd services. SysV configuration data might be overridden by native
      systemd configuration.

      If you want to list systemd services use 'systemctl list-unit-files'.
      To see services enabled on particular target use
      'systemctl list-dependencies [target]'.

mysqld          0:off   1:off   2:on    3:on    4:on    5:on    6:off
netconsole      0:off   1:off   2:off   3:off   4:off   5:off   6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

18、配置全局环境变量

```
#编辑/etc/profile文件
vi /etc/profile
在文件底部添加以下配置信息
```

```
https://www.cnblogs.com/yunian139/p/11804965.html
```