<https://www.cnblogs.com/xiaoyao-lxy/p/5561728.html#wu>

# [一个小白安装Win10、CentOS 7双系统的经历](https://www.cnblogs.com/xiaoyao-lxy/p/5561728.html)

　　最近心血来潮，想学Linux，所以根据网上的建议，买了鸟哥的书。为了以后学习方便，准备安装CentOS7。准备好镜像文件和U盘后，4号晚上八点多开始施工，本以为几个小时就可以弄好的东西，第一次让我明白了什么叫理想与现实的差距。鉴于CentOS7安装的繁琐程度和鸟哥的火热程度，我决定把我安装的全过程写下来，希望能帮助到想要学习的小白们。在这篇教程里，我把我遇到的问题也写了出来，因为我从解决这些问题的过程中，学到了很多知识，如果你也是个小白的话，希望你耐心看下去。

　　另外，欢迎大神们指点批评~

**事先声明一下，在CentOS 7安装过程中，没有启动项的选择，就是说，安装好CentOS后，我们打不开win10，所以在这个问题发生之前，我们就要想好解决措施，因此，我推荐先看第五步，**[**点此跳转**](https://www.cnblogs.com/xiaoyao-lxy/p/5561728.html#wu)

# ****工具及材料****

　　1、一台联网的电脑，或者可以先把软件下载好

　　2、两个U盘，8G以上最好

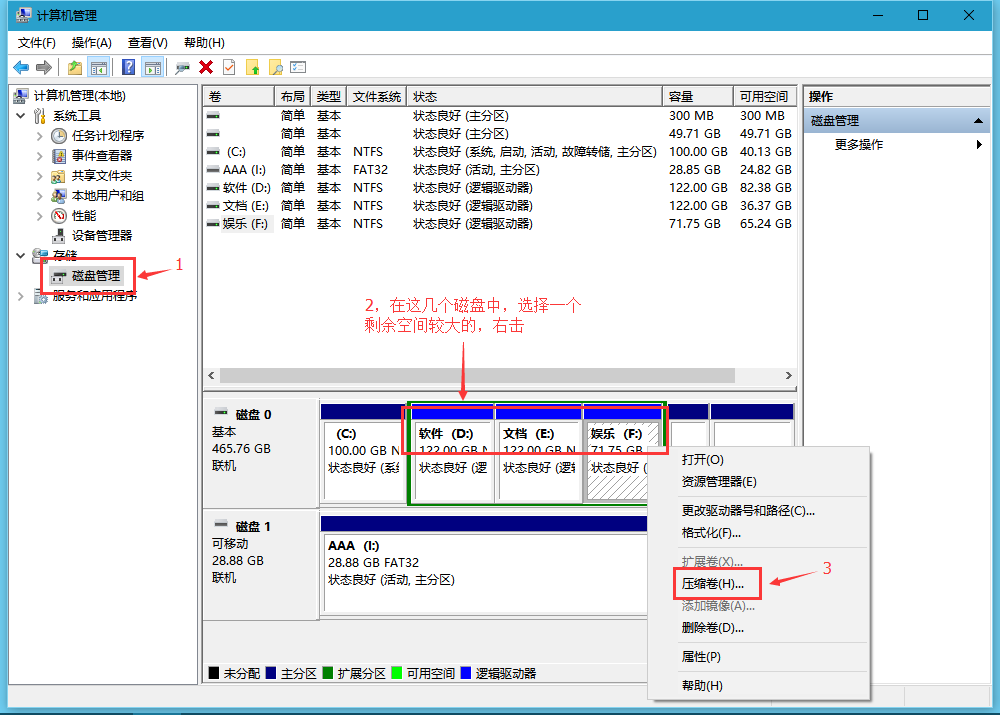
　　3、需要的文件：CentOS7镜像文件（ISO文件）

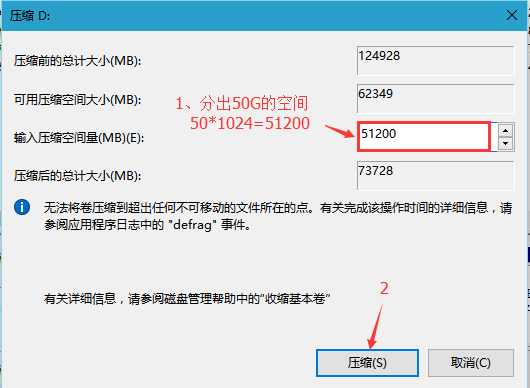
　　4、需要的软件：UltraISO、老毛桃、EasyBCD

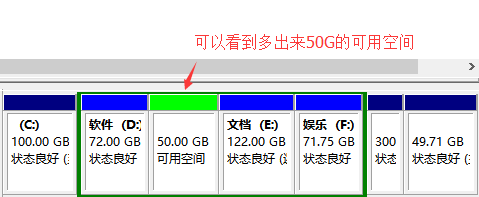
# ****第一步、在Windows中为Linux分出磁盘空间****

　　------------------------------有两颗硬盘的壕们自行[跳过](https://www.cnblogs.com/xiaoyao-lxy/p/5561728.html#tiao)-------------------------------

　　右击“此电脑” -- “管理”







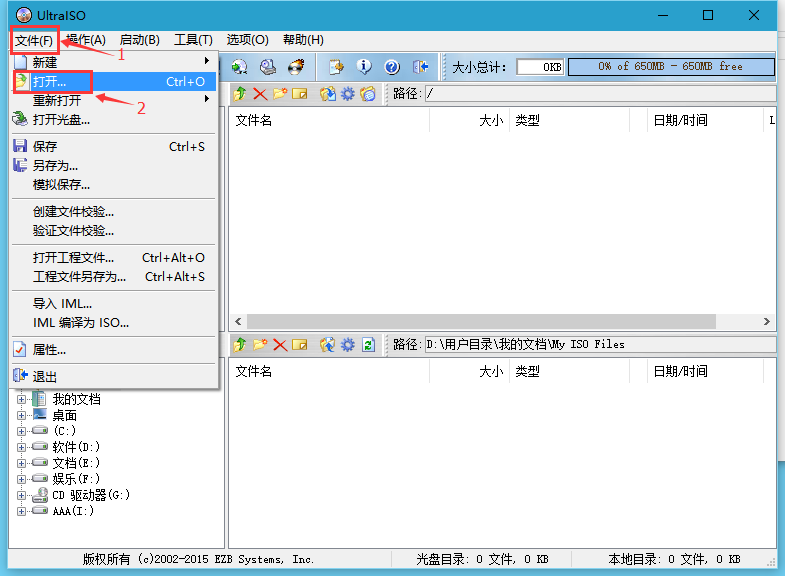
　　放心吧，不会弄坏电脑的

# ****第二步、制作U盘启动盘****

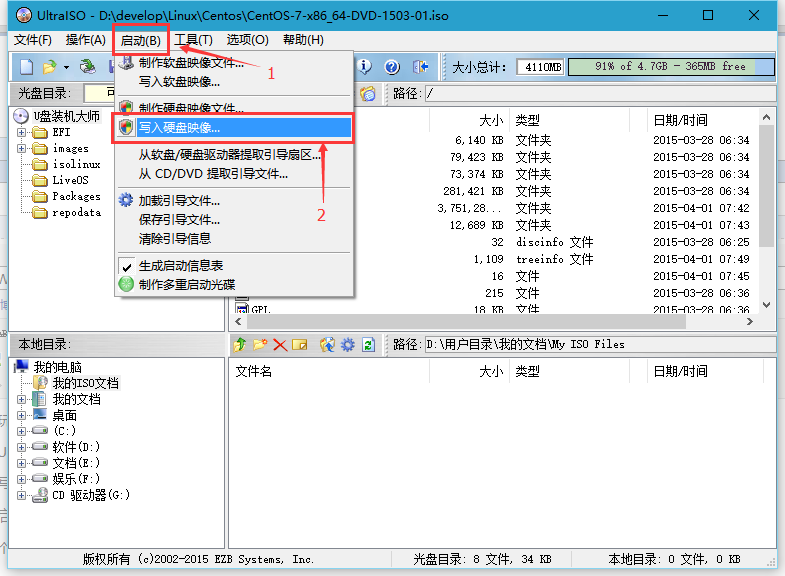
　　光这一步就花了将近两个小时，其实并不难，只是我当时看的教程是早期的，所以走了好多弯路。我们使用UltraISO，即软碟通，一定要去网上下载最新版的，即使是试用也没关系，因为我们只用一下就好。下面是制作过程

　　1、打开ISO文件

　　点击“文件” -- “打开”，选择你保存ISO文件的地方



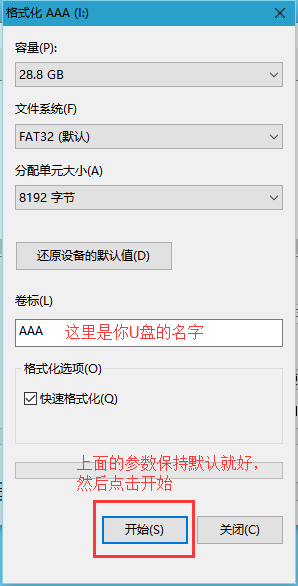
　　2、写入U盘



**在写入的时候，软件会格式化U盘，所以在写入前一定要注意把U盘中重要的数据进行备份**

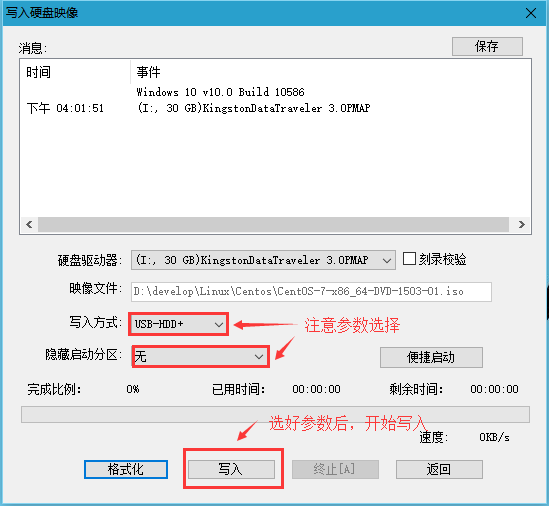
　　尽管会自动格式化，但是为了以防万一，我们还是自己格式化一下

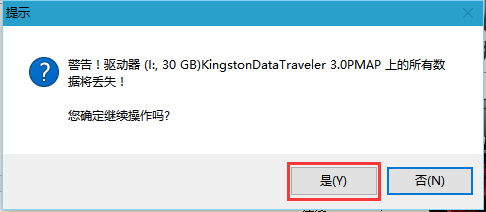


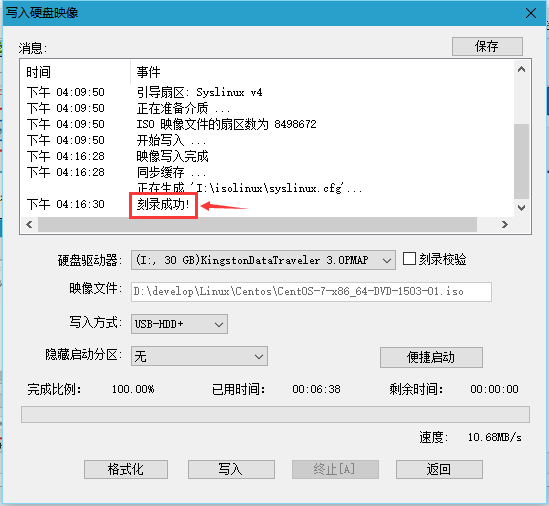


　　然后一路确定~~~

　　格式化完成后，进行写入，注意参数的选择

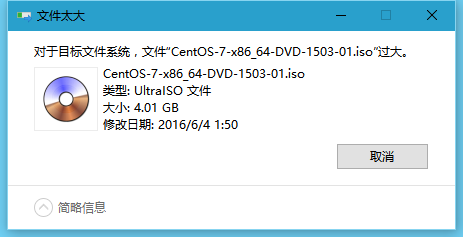






　　到此，第一步就完成了，全程也就5、6分种。我之所以花费了两个小时，是因为当时看的教程说，要在写入前删除几个文件，然后在写入完把ISO原文件拷贝进去。

　　那么问题就来了，U盘启动盘做好后，它的类型是FAT32的，也就是说，大于4G的文件是拷贝不进去的，而我的ISO文件恰好4.01G，所以在拷贝的时候就出现了下面这个情况



　　当时我就无语了，我的U盘是32G的，怎么放不进4G的东西呢，所以我就怀疑是不是买到了假U盘......但是想了想，前几天还用这个U盘转移了23G的PS教程，怎么可能是假的呢？于是就去BD，得到的结果是：FAT32格式下，单个文件最大不超过4G。

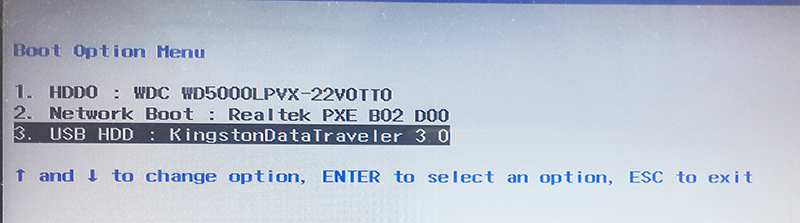
　　那怎么办？当时我很机智的（后来证明很蠢）把U盘重新格式化了一下，格式化成了NTFS格式，这样就对文件没限制啦，于是继续重复上述步骤......然而并没有什么卵用。还是拷贝不进去。怎么可能，我把格式改了呀，但是打开一看，发现它又变成FAT32了，所以继续BD。发现几乎所有刻录软件都把U盘格式化成了FAT32，因此，我用另外的软件将写入后的U盘改成了NTFS，然后把ISO文件拷贝进去。这样做的结果是，开机时，电脑根本识别不到这个启动盘......无奈，开机后继续做，就这样，花了两个多小时，然后在一篇文章的角落中发现现在的ISO文件刻录时已经不需要拷贝原文件了，抱着死马当做活马医的态度，再一次重复上述刻录步骤后，直接重启电脑，结果成功识别。

　　所以，在这里我想提醒各位小白筒子：**看教程的时候，一定要注意教程的发布时间**。如果是很久前的教程，那就要小心了，因为变化可能非常大。

# ****第三步、利用U盘引导****

（由于U盘安装时无法截图，所以以下过程是在虚拟机中进行的。）

　　1、重启电脑后，狂按F12进去boot界面，选择USB启动



　　进入画面

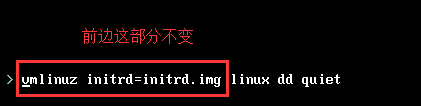


**注意不能直接选“Install CentOS 7”或“Test this media &  istall CentOS 7”**，我当时在这里就花费了将近三个小时，泪奔.......

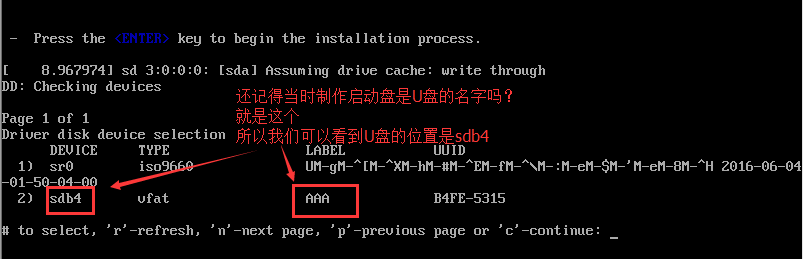
　　因为我们是用U盘安装的，所以要指定U盘位置。因此，我们先按下Tab键，屏幕下方的倒计时变为以下字符串（有的筒子可能会遇到按Tab没反应的情况，那可能是你选择了“Troubleshoot ing”这一项，只要将选择光标移上去就好了）



　　将它改为下边这样，然后回车



　　就可以看到这个界面

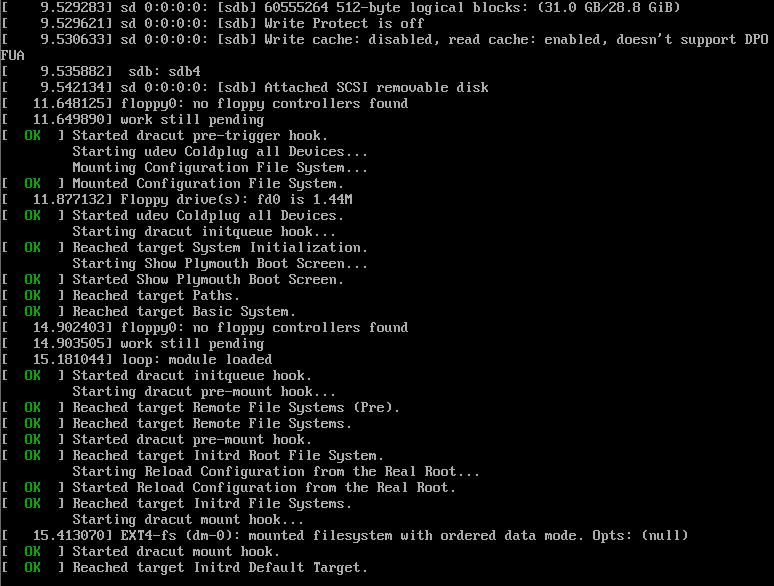


　　这样我们就得到了U盘的位置，**一定要记好你的U盘位置**，然后我们重启电脑（当然只能用电源键强行关机了），按F12进入boot，选择U盘启动，按Tab键（重复上边步骤），然后将字符串改为下边这样，回车：



于是就出现了一大堆的字符串和OK，不用理会，等着就好，上一张图作参考：

sda4



　　在这一步中，其实问题也不大，但是！！！我看了好多教程，里边各种方法都有，而我却一个也没试成功，以至于成功进去安装界面时，已经是凌晨两点多了

　　下面是网络上最常（坑）见（人）的三种处理方法：

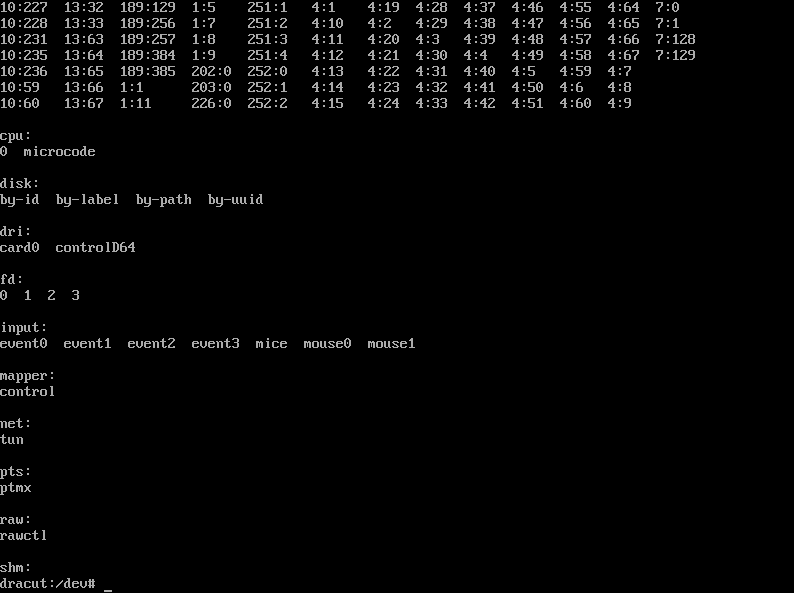
　　1、



　　它为什么排第一，不解释，红框里的一句话，折腾了我一个多小时，甚至一度怀疑我的U盘启动盘制作出错，回去又做了好几遍

　　2、“直接进入，然后系统会报错，这是我们输入cd /dev/ls\* ，从这里看到U盘的位置后重启电脑”

　　这个教程已经告诉我们直接安装是不对的，要指定U盘位置，我按照这个教程小心翼翼试了好多遍，无果。后来注意到了1s和ls的区别，即第一个是字母还是数字，继续试，无果，最后想到了ls与\*间有没有空格，加了空格后，成功进入下边界面



　　完全看不懂

　　3、将        vmlinuz initrd=initrd.img inst.stage2=hd:LABEL=CentOS\x207\x20x86\_64 rd.live.check quiet

　　     改为     vmlinuz initrd=initrd.img inst.stage2=hd:/dev/sdb1 quiet

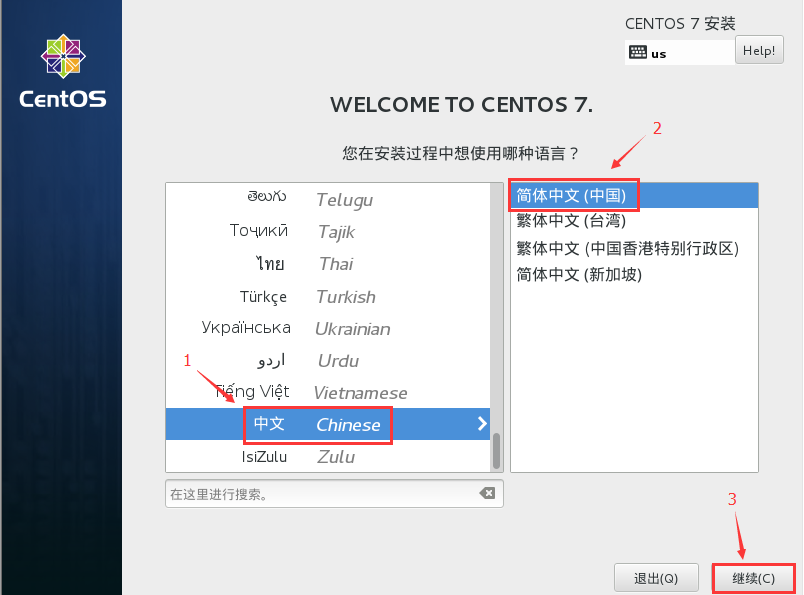
　　　　这里sdb1为U盘所在分区，也可能是sdb2 ，可根据实际情况修改。

　　对于这个方法，虽然是正确的，但是我还是忍不住想吐槽一句：你倒是告诉我根据实际情况是怎么个根据法啊，一点判断条件都没有，衰衰的我只能从1试到4

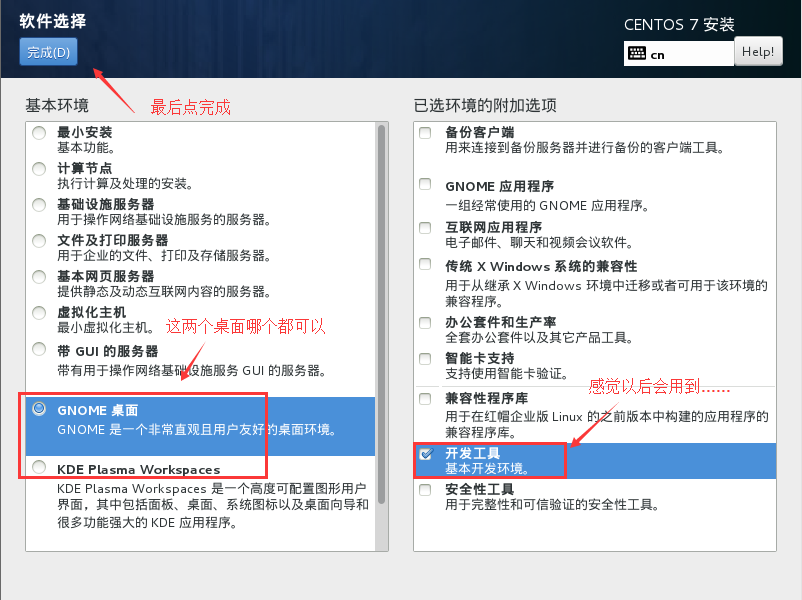
　　以上摘取的三个方法中，我是用第三个方法试了四遍才成功的，看到图形界面后，差点感动哭了

# ****第四步、安装****

　　不久之后，就到了这个界面，终于不是冷冰冰的命令行了，内心还是有点小激动的~O(∩\_∩)O



**注意一定要进去选一个桌面**，我当时很悲哀的忘了选，结果进去以后全是命令行，关机都不会......无奈，强行关机，再重新安装



　　然后是分区，分区这里要注意，一定要选择我们**在第一步中分出来的空闲磁盘，否则会覆盖掉磁盘里原有的信息，切记切记。“硬盘有价，数据无价”**。

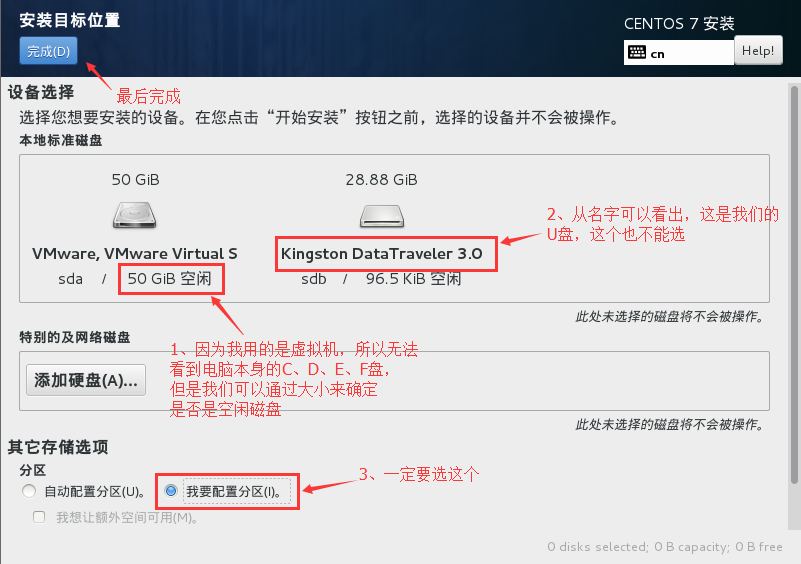
　　分区计划

　　　　/：　　 　　30G

　　　　swap： 　　8G

　　　　/home：　  11.7G

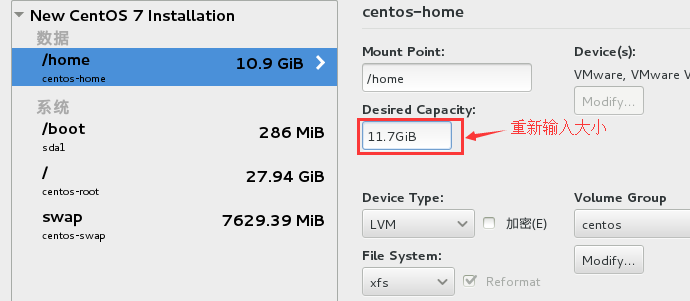
　　　　/boot： 　  300M





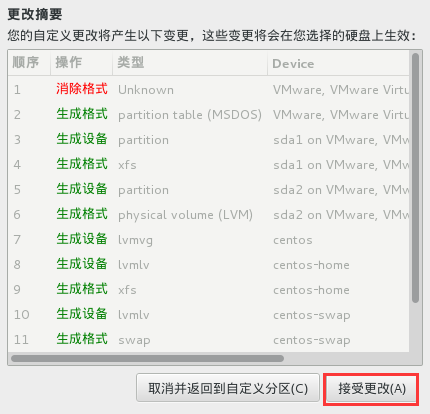
　　同样的方法建立swap、/boot、/home分区，这里需要注意的是swap的大小，一般来说，应该是实际内存的两倍，但是实际内存很大的话，设置8G也够了

　　分区后发现大小不是我们设定的，所以在右边重新设置大小，但是磁盘不能完全被分割，所以最后swap会小一点点（记得设置swap和boot时单位是MB）



　　至于这个分区的类型，也就是Device Type这里，按照它默认就好，即/、swap、/home是LVM类型，/boot是标准分区

　　分好区后，点击“完成”，弹出下面的窗口，直接点击“接受更改”



　　到现在，全部设置完毕，开始安装





　　注意在设置账户时，还会设置一个密码，而那个密码是针对账户来说的，和root密码并不一样，所以要将两个密码记清楚（懒癌晚期的我设置成一样的了），而且如果你的密码设置太简单的话，要点两次完成才能退出

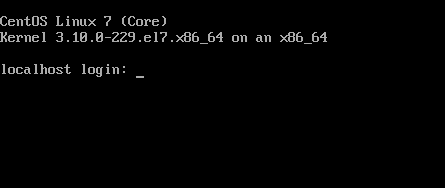


　　重启后使用刚才设置的账号和密码登录就可以了

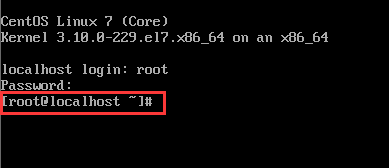
　　这一步也花了很久很久，主要原因有两个：

　　　　一：分区的时候，不知道该选择LVM还是标准分区，纠结了好久。就像上边说过的，默认就好

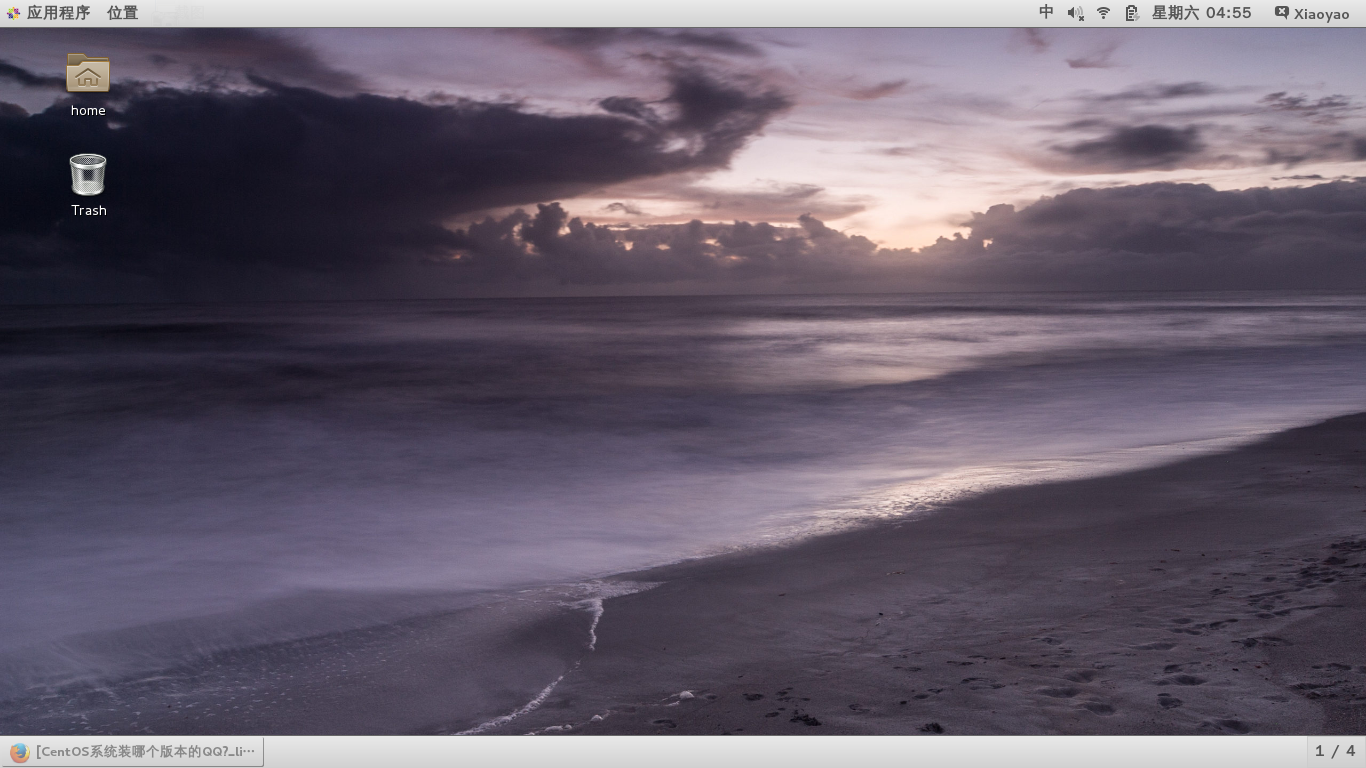
　　　　二：软件选择的时候，忘了选桌面，而是选择了最小安装，进去以后是这样的



　　　　经过BD，发现这是要输入账户名和密码，那就按照要求来（Linux安全做得很好，输入密码时不会回显）



　　　　然后，红框里的东西，其实时已经安装好的标志，只不过没有安装桌面。但是当时我不认识啊，还以为又出错了，OK，关机，重启，再来一遍，还是这样。弄了三次后，才弄明白这已经安装好了，可以直接输入指令来操作了，但是我不会，好吧，再来一次，这次选了GNOME，成功进入Linux



　　　　可以看到，4:55了，当时天已经亮了

　　　　顺便说一句，Linux没法直接安装QQ

**第五步、找到win10的引导**

　　前面也说过，CentOS安装过程中，是不可以选择启动项的，那现在我们就要解决这个问题

　　方法一：在Linux中配置grub

　　　　具体实现请参考：http://tieba.baidu.com/p/3152957061

　　　　先说明一下，小白们就不要进去了，啥也看不懂，我们用第二种

　　方法二：

　　　　1、恢复win10的MBR

　　　　之所以推荐大家先看这一步的原因就在这里，我们要在还是Windows的情况下，先做好一个winPE启动盘，这也是要用两个U盘的原因

　　　　我用的软件是老毛桃（大白菜也可以，都很无脑，基本不需要什么设置）具体实现请参考：http://www.laomaotao.org/softhelp/syjc/1193.html

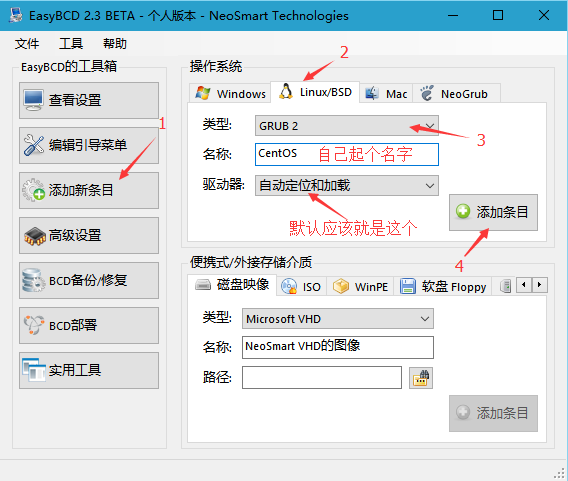
　　　　这里还有一个小问题，就是装机版和UEFI版，现在大部分电脑应该都支持UEFI，但我当时是用学校图书馆的破电脑做的，做UEFI是总是提示读取不了，无奈只能做装机版，不过后来感觉没多大区别。而且上边这个教程也有点老了，我做好启动盘启动后，在开始菜单并没有找到他说的“分区助手”，所以用了桌面上的一个分区软件，里边也有“重建MBR”这么个东西。如果到时候筒子们不会用那个软件，请自行BD（经过CentOS的摧残后，我对度娘的爱意更上一层楼）。

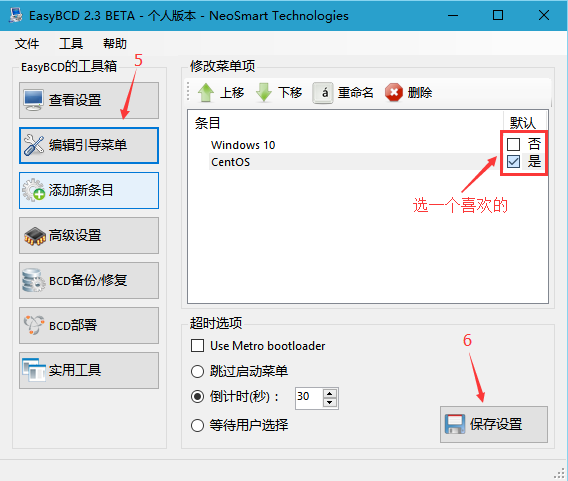
**从最上边跳下来的筒子们，你们现在可以跳回去了~~~**[**点此坐火箭**](https://www.cnblogs.com/xiaoyao-lxy/p/5561728.html#yi)

　　　　2、将Cent0S 7加入到启动项中

　　　　恢复MBR后还没完，这时候你可以进入win10了，但是，你会发现启动项中又没有CentOS了（我的内心是崩溃的）

　　　　所以我们要用到EasyBDC这个软件（注册码直接BD）





　　　　重启电脑后，就可以看到启动项了

　　至此，我们的整个安装过程就完成了

　　不过安装完后，我还遇到了几个小问题

　　第一个：就是用EasyBCD加入启动项后，win10关不了机，每次关机就自动重启

　　　　解决方法：取消win10“快速启动”。

　　第二个：可以正常开关机后，再打开“快速启动”，发现找不到CentOS启动项

　　　　解决方法：还是取消win10“快速启动”。试了很多次，发现只要开了这个功能，就无法使用CentOS

　　OK，所有问题全部解决，欢迎小白们一起学习交流，欢迎大神们指点，水平有限，不喜勿喷

**boot分区**：

                     作用：引导分区，包含了系统启动的必要内核文件，即使根分区顺坏也能正常引导启动 一般这些文件所占空间在200M以下，

                     分区建议：分区的时候可选100M-500M之间,如果空间足够用，建议分300-500M。避免由于长期使用的冗余文件塞满这个分区。

　　　　　　分区格式：建议ext4，按需求更改 500MB

**/分区（根分区）：**

作用：所有的文件都从这里开始，你可以比喻为Windows的C盘，但其实有区别。如果你有大量的数据在根目录下（比如FTP等）可以划分大一点的空间

                      分区建议：建议15G以上。看需求，根分区和home分区的大小就类似C盘和D盘的空间分布一样，主要占空间在哪儿就在那里分大容量

                      分区格式：建议ext4，按需求更改 20G

**swap分区：**

   　　　　　作用：类似于Windows的虚拟内存，在内存不够用时占用硬盘的虚拟内存来进行临时数据的存放，而对于linux就是swap分区

                     分区建议：建议是物理内存大小的2倍，比如你电脑是4G的物理内存，swap分区可以是8G

　　　　　　分区格式：swap格式 8G内存 12G

**可选的分区**

**home分区**

　　　　       作用：存放用户数据，HOME的结构一般是 HOME/userName/userFile，如果不分则默认在/目录下

                     分区建议：如果用户数据多可以将此分区适当增大，请参考“根分区”分区建议；一般硬盘的主要容量几乎都在Home分区和根分区下

                     分区格式：建议ext4，按需求更改6GB

**var分区**

　　　　      作用：用于log日志的文件的存放，如果不分则默认在/目录下

　　　　      分区建议：如果你安装的linux是用于服务器或者经常做日志分析，请划分var分区，避免日志文件不断膨胀塞满导致根分区而引发问题。

                    分区格式：建议ext4，按需求更改2GB

<https://blog.csdn.net/qq_34889607/article/details/78488145>

案例背景  
       本案例是在已安装了windows 7系统的基础上，在一个空闲的分区上安装CentOS 7 Linux系统。当成功安装Linux系统后并不会像Centos 6版本那样保留windows 7系统的启动菜单。而是只有Centos7的系统启动菜单。必须在进入Linux系统后由管理员在/boot/grub2/grub.cfg文件中手动添加windows 7的启动菜单。  
  
方案一：将以下内容先写入到/etc/grub.d/40\_custom文件中，重新生成grub.cfg启动菜单文件。  
第1步，编辑/etc/grub.d/40\_custom文件。  
vi   /etc/grub.d/40\_custom   添加如下内容(菜单内容参考方案一)  
menuentry   'win7' {     
     insmod  part\_msdos   
     set  root='hd0,msdos1'  
     search  -f   /windows/notepad.exe   --set-root  
     chainloader  +1  
}  
  
第2步，更新grub.cfg菜单文件。  
grub2-mkconfig  -o   /boot/grub2/grub.cfg  
grep   win7   /boot/grub2/grub.cfg  
  
第3步，将win7设置为默认启动。（根据个人实际需要考虑是否做这一步）  
grub2-set-default   'win7'     将win7菜单设置为默认启动  
grub2-editenv   list          显示当前的默认启动菜单  
  
第4步，reboot重启Linux系统，验证是否有win7的启动菜单，以及能否正常启动win7系统。  
  
注：查看启动菜单有哪些  
       grep menu /boot/grub2/grub.cfg  
  
－－－－－－－－－－－－－－－－－－－  
附录：  
vi   /etc/grub.d/40\_custom   添加如下内容(最优化的菜单内容参考方案一)  
menuentry   'win7' {               #指定菜单标题  
     insmod  part\_msdos          #加载msdos分区模块  
     set  root='hd0,msdos1'      #设置根分区为第1块硬盘(hard disk)的msdos1分区  
     search  -f   /windows/notepad.exe   --set-root         #将搜索到的notepad.ext文件所在的分区设置为根  
     chainloader  +1          #链接引导  
}  
  
/etc/grub.d/40\_custom  菜单文件内容参考方案二  
menuentry  'win7'  {  
    set   root(hd0,1)  
    chainloader   +1  
}  
  
/etc/grub.d/40\_custom  菜单文件内容参考方案三  
menuentry  'win7'  {  
     insmod   chain  
     insmod   ntfs  
     set   root(hd0,msdos1)  
     chainloader   +1

}

作为一个菜鸟，在接触Linux的过程中，经常会导致Linux出现各种错误，可恢复的或者是不可恢复的（至少在目前的情况下），所以，对于我而言，Linux的备份和还原就尤为的重要。

1.备份

a)可以直接通过tar对整个文件系统（'/‘）进行备份，但是有几点需要注意：

i. 不能备份以下几个文件（目录）

1. 当前压缩文件

2. /proc文件夹

3. /lost+found文件夹

4. /mnt文件夹

5. /sys文件夹

6. /media文件夹

b)所以，命令为：

tar cvpzf backup.tar.gz --exclude=/proc --exclude=/lost+found --exclude=/backup.tar.gz --exclude=/mnt --exclude=/sys --exclude=/media /

1.注意：p选项代表保存当前权限

2.还原

a) Linux可以再正在远行的系统中还原系统，如果当前启动无法启动，可以通过live cd来启动并执行恢复操作

b) 操作如下

tar xcpfz backup.tar.gz -C /

c) 需要额外创建目录

i. mkdir proc

ii. mkdir lost+found

iii. mkdir mnt

iv. mkdir sys

ngrok

<https://www.centos.bz/2017/11/centos%E4%B8%8Bngrok%E6%9C%8D%E5%8A%A1%E5%99%A8%E6%90%AD%E5%BB%BA%E5%8F%8Angrok%E5%AE%A2%E6%88%B7%E7%AB%AF%E4%BD%BF%E7%94%A8/>

vim /etc/profile.d/go.sh

ngrok生成目录

/usr/local/go/bin/

# ngrok搭建（Windows服务端+Windows客户端）

1.go环境搭建（需要Linux系统）

1）下载源码，可以在http://www.golangtc.com/download

2）将其解压到/usr/local目录下：  
tar -C /usr/local -xzf go1.9.2.linux-amd64.tar.gz

3).在root环境下执行如下命令：  
mkdir $HOME/go  
echo ‘export GOROOT=/usr/local/go’>> ~/.bashrc  
echo ‘export GOPATH=$HOME/go’>> ~/.bashrc  
echo ‘export PATH=$PATH:$GOROOT/bin’>> ~/.bashrc  
source $HOME/.bashrc

4). 安装go get工具  
yum install mercurial git bzr subversion

2.获取源码

git clone https://github.com/inconshreveable/ngrok.git

3. 编译  
1). 配置环境变量

export NGROK\_DOMAIN=”ngrok.example.com”

ngrok.example.com替换成你自己的域名。

2). 生成自签名ssl证书

cd ngrok

openssl genrsa -out rootCA.key 2048

openssl req -x509 -new -nodes -key rootCA.key -subj “/CN=$NGROK\_DOMAIN” -days 5000 -out rootCA.pem

openssl genrsa -out device.key 2048

openssl req -new -key device.key -subj “/CN=$NGROK\_DOMAIN” -out device.csr

openssl x509 -req -in device.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out device.crt -days 5000

cp rootCA.pem assets/client/tls/ngrokroot.crt

cp device.crt assets/server/tls/snakeoil.crt

cp device.key assets/server/tls/snakeoil.key

4. 交叉编译生成windows客户端  
上述编译过程生成的服务端和客户端都是linux下的，不能在windows下用。如果想编译生成windows客户端，需要重新配置环境并编译。 交叉编译过程如下：

进入go目录，进行环境配置  
cd /usr/local/go/src/

GOOS=windows GOARCH=amd64 CGO\_ENABLED=0 ./make.bash  
进入ngrok目录重新编译  
cd /usr/local/src/ngrok/

GOOS=windows GOARCH=amd64 make release-server release-client  
编译后，就会在bin目录下生成windows\_amd64目录，其中就包含着windows下运行的服务器和客户端程序。

此时会遇到错误

Set $GOROOT\_BOOTSTRAP to a working Go tree >= Go 1.4.

解决办法是，下载golang1.4版本，然后执行下面命令。

 https://storage.googleapis.com/golang/go1.4.2.darwin-amd64-osx10.8.tar.gz

tar zxvf [go1.4.2.darwin-amd64-osx10.8.tar.gz]  
cp go/ $home/go-bootstrap/  
GOROOT\_BOOTSTRAP=$home/go-bootstrap/  
export GOROOT\_BOOTSTRAP

5.客户端及服务端配置。

以3389远程桌面为例。

客户端新建一个ngrok.cfg文件，内容如下：

server\_addr: “ngrok.example.org:1180”  
trust\_host\_root\_certs: false

tunnels:  
mstsc:  
subdomain: “mstsc”  
remote\_port: 4443  
proto:  
tcp: “127.0.0.1:3389″

然后写一个批处理，内容如下：

ngrok -config=ngrok.cfg start mstsc

服务端也写一个批处理，内容如下:

ngrokd.exe -domain=”ngrok.example.org” -httpAddr=”:801″ -httpsAddr=”:802″ -tunnelAddr=”:1180″

6.参考

1.http://www.360doc.com/content/17/0524/18/29401987\_656833431.shtml

2.https://www.jianshu.com/p/0146801c1178

3.https://www.jianshu.com/p/4f79ae4f081c

# 安装go最新版本go1.4

最近想学习下go语言，把安装环境过程记录如下:

假定要把go安装到/home/xxx/golang目录下：

1. 安装go 编译器

cd  /home/xxx/golang

下载：wget https://storage.googleapis.com/golang/go1.4-bootstrap-20161024.tar.gz

解压：tar zxvf go1.4-bootstrap-20161024.tar.gz -C go1.4

cd go1.4/src

./allbash

此处需要等待几分钟

运行如下命令，可以确认是否安装

/home/xxx/golang/go1.4/bin/go version

go version go1.4-bootstrap-20161024 linux/amd64

---------------------

作者：xxxbigbug

来源：CSDN

原文：https://blog.csdn.net/zhiyuan\_2007/article/details/62423319

版权声明：本文为博主原创文章，转载请附上博文链接！

Liunx 开放端口

firewall-cmd --zone=public --add-port=1001/tcp --permanent

firewall-cmd --reload

setenforce是Linux的selinux防火墙配置命令 执行setenforce 0 表示关闭selinux防火墙。  
  
setenforce命令是单词set（设置）和enforce(执行)连写，另一个命令getenforce可查看selinux的状态。

setenforce 0

getenforce

nginx

检查配置

../sbin/nginx -t

重启

../sbin/nginx -s reload

**/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf**

service mysql restart

netstat –antp

vi /etc/my.cnf

查找文件

find / -name 'go\*.tar.gz'

查找目录

find / -name 'go' -type d

source /etc/profile

GOOS=linux GOARCH=amd64 make release-server release-client

GOOS=windows GOARCH=amd64 make release-client

GOOS=windows GOARCH=amd64 make release-client release-server

setsid /usr/local/ngrokclient/ngrok -config=/usr/local/ngrokclient/ngrok.cfg -log=/usr/local/ngrokclient/log.log start api ssh

vim 按ESC键 跳到命令模式，然后：

1. **:w 保存文件但不退出vi**
2. **:w file 将修改另外保存到file中，不退出vi**
3. **:w! 强制保存，不推出vi**
4. **:wq 保存文件并退出vi**
5. **:wq! 强制保存文件，并退出vi**
6. **q: 不保存文件，退出vi**
7. **:q! 不保存文件，强制退出vi**
8. **:e! 放弃所有修改，从上次保存文件开始再编辑**

## 命令历史

以:和/开头的命令都有历史纪录，可以首先键入:或/然后按上下箭头来选择某个历史命令。

## vim的模式

正常模式（按Esc或Ctrl+[进入） 左下角显示文件名或为空  
插入模式（按i键进入） 左下角显示--INSERT--  
可视模式（不知道如何进入） 左下角显示--VISUAL--

# CentOS7 基本操作

一、查看版本号：

1.查看CentOS的版本号：cat /etc/centos-release

2.查看内核版本号：uname -r

3.查看操作系统位数：getconf LONG\_BIT

4.其他方法



二、centOS7下实践查询版本/CPU/内存/硬盘容量等硬件信息

    参阅：https  :// blog . csdn . net / dream\_broken/article/details/52883883

三、查看CPU信息：cat /proc/cpuinfo

四、查看内存信息：cat /proc/meminfo

————————————————

版权声明：本文为CSDN博主「one312」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/one312/article/details/81059301>

1.系统

  1.1版本

  uname -a 能确认是64位还是32位，其它的信息不多

[root@localhost ~]# uname -a

Linux localhost.localdomain 3.10.0-327.el7.x86\_64 #1 SMP Thu Nov 19 22:10:57 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux

  more /etc/\*release 可以看到更多信息

[root@localhost ~]# more /etc/\*release

::::::::::::::

/etc/centos-release

::::::::::::::

CentOS Linux release 7.2.1511 (Core)

::::::::::::::

/etc/os-release

::::::::::::::

NAME="CentOS Linux"

VERSION="7 (Core)"

ID="centos"

ID\_LIKE="rhel fedora"

VERSION\_ID="7"

PRETTY\_NAME="CentOS Linux 7 (Core)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:centos:centos:7"

HOME\_URL="https://www.centos.org/"

BUG\_REPORT\_URL="https://bugs.centos.org/"

CENTOS\_MANTISBT\_PROJECT="CentOS-7"

CENTOS\_MANTISBT\_PROJECT\_VERSION="7"

REDHAT\_SUPPORT\_PRODUCT="centos"

REDHAT\_SUPPORT\_PRODUCT\_VERSION="7"

::::::::::::::

/etc/redhat-release

::::::::::::::

CentOS Linux release 7.2.1511 (Core)

::::::::::::::

/etc/system-release

::::::::::::::

CentOS Linux release 7.2.1511 (Core)

  1.2核数

    cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c

[root@localhost ~]# cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c

1 Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz

   1个逻辑CPU，i5型等信息

[root@localhost ~]# cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c

8 Intel(R) Xeon(R) CPU E7-4820 v2 @ 2.00GHz

   8个逻辑CPU

  cat /proc/cpuinfo | grep physical | uniq -c

[root@localhost ~]# cat /proc/cpuinfo | grep physical | uniq -c

1 physical id : 0

1 address sizes : 42 bits physical, 48 bits virtual

  实际上是一颗一核的CPU

[root@localhost ~]# cat /proc/cpuinfo | grep physical | uniq -c

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

1 physical id : 0

1 address sizes : 40 bits physical, 48 bits virtual

  由8个1核的CPU组成8核

  cat /proc/cpuinfo可以看到更为详细的信息

[root@localhost ~]# cat /proc/cpuinfo

processor : 0

vendor\_id : GenuineIntel

cpu family : 6

model : 94

model name : Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz

stepping : 3

microcode : 0x74

cpu MHz : 2304.004

cache size : 6144 KB

physical id : 0

siblings : 1

core id : 0

cpu cores : 1

apicid : 0

initial apicid : 0

fpu : yes

fpu\_exception : yes

cpuid level : 22

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant\_tsc arch\_perfmon pebs bts nopl xtopology tsc\_reliable nonstop\_tsc aperfmperf eagerfpu pni pclmulqdq ssse3 fma cx16 pcid sse4\_1 sse4\_2 x2apic movbe popcnt tsc\_deadline\_timer aes xsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch ida arat epb pln pts dtherm hwp hwp\_noitfy hwp\_act\_window hwp\_epp fsgsbase tsc\_adjust bmi1 hle avx2 smep bmi2 invpcid rtm rdseed adx smap xsaveopt xsavec xgetbv1 xsaves

bogomips : 4608.00

clflush size : 64

cache\_alignment : 64

address sizes : 42 bits physical, 48 bits virtual

power management:

1.3运行模式

getconf LONG\_BIT  CPU运行在多少位模式下

[root@localhost ~]# getconf LONG\_BIT

64

如果是32，说明当前CPU运行在32bit模式下, 但不代表CPU不支持64bit

cat /proc/cpuinfo | grep flags | grep ' lm ' | wc -l 是否支持64位

[root@localhost ~]# cat /proc/cpuinfo | grep flags | grep ' lm ' | wc -l

1

结果大于0, 说明支持64bit计算. lm指long mode, 支持lm则是64bit

1.4计算机名

  hostname

[root@localhost ~]# hostname

localhost.localdomain

1.5.查看环境变量

  env

[root@localhost ~]# env

XDG\_SESSION\_ID=4

HOSTNAME=localhost.localdomain

SELINUX\_ROLE\_REQUESTED=

TERM=vt100

SHELL=/bin/bash

HISTSIZE=1000

SSH\_CLIENT=192.168.174.1 58896 22

SELINUX\_USE\_CURRENT\_RANGE=

SSH\_TTY=/dev/pts/0

USER=root

1.6系统允许多长时间了/负载数

uptime

[root@localhost proc]# uptime

10:55:01 up 1:28, 2 users, load average: 0.00, 0.01, 0.05

1.当前时间10:55:01

2.系统运行了多少时间,1:28（1小时28分)

3.多少个用户,2 users

4.平均负载：0.00, 0.01, 0.05，最近1分钟、5分钟、15分钟系统的负载

   直接查看平均负载情况 cat /proc/loadavg

[root@localhost proc]# cat /proc/loadavg

0.00 0.01 0.05 4/524 7152

 除了前3个数字表示平均进程数量外，后面的1个分数，分母表示系统进程总数，分子表示正在运行的进程数；最后一个数字表示最近运行的进程ID

2.资源

  2.1内存

   cat /proc/meminfo内存的详细信息

[root@localhost proc]# cat /proc/meminfo

MemTotal: 1001332 kB

MemFree: 99592 kB

MemAvailable: 420940 kB

Buffers: 1064 kB

Cached: 405292 kB

SwapCached: 0 kB

Active: 412548 kB

Inactive: 250192 kB

Active(anon): 205264 kB

Inactive(anon): 58460 kB

Active(file): 207284 kB

Inactive(file): 191732 kB

Unevictable: 0 kB

Mlocked: 0 kB

SwapTotal: 2097148 kB

SwapFree: 2097140 kB

Dirty: 0 kB

Writeback: 0 kB

AnonPages: 256416 kB

Mapped: 103344 kB

Shmem: 7340 kB

Slab: 126408 kB

SReclaimable: 69912 kB

SUnreclaim: 56496 kB

KernelStack: 10416 kB

PageTables: 15520 kB

NFS\_Unstable: 0 kB

Bounce: 0 kB

WritebackTmp: 0 kB

CommitLimit: 2597812 kB

Committed\_AS: 1578872 kB

VmallocTotal: 34359738367 kB

VmallocUsed: 170756 kB

VmallocChunk: 34359564288 kB

HardwareCorrupted: 0 kB

AnonHugePages: 75776 kB

HugePages\_Total: 0

HugePages\_Free: 0

HugePages\_Rsvd: 0

HugePages\_Surp: 0

Hugepagesize: 2048 kB

DirectMap4k: 73600 kB

DirectMap2M: 974848 kB

DirectMap1G: 0 kB

 MemTotal总内存，MemFree可用内存

 free -m(-m，单位是m，如果是-g，单位是g）查看可用内存

[root@localhost proc]# free -m

total used free shared buff/cache available

Mem: 977 360 97 7 520 411

Swap: 2047 0 2047

 空闲内存total-used=free+buff/cache

我们通过free命令查看机器空闲内存时，会发现free的值很小。这主要是因为，在linux中有这么一种思想，内存不用白不用，因此它尽可能的cache和buffer一些数据，以方便下次使用。但实际上这些内存也是可以立刻拿来使用的。

3.磁盘和分区

  3.1查看各分区使用情况

   df -h

[root@localhost ~]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/centos-root 45G 22G 24G 48% /

devtmpfs 906M 0 906M 0% /dev

tmpfs 921M 96K 921M 1% /dev/shm

tmpfs 921M 1004K 920M 1% /run

tmpfs 921M 0 921M 0% /sys/fs/cgroup

/dev/sda1 497M 195M 303M 40% /boot

tmpfs 185M 0 185M 0% /run/user/1001

tmpfs 185M 0 185M 0% /run/user/0

[root@localhost ~]#

  3.2查看指定目录的大小

   du -sh <目录名>

[root@localhost ~]# du -sh /root

1.2G /root

3.3查看所有分区

   fdisk -l

[root@localhost proc]# fdisk -l

磁盘 /dev/sda：32.2 GB, 32212254720 字节，62914560 个扇区

Units = 扇区 of 1 \* 512 = 512 bytes

扇区大小(逻辑/物理)：512 字节 / 512 字节

I/O 大小(最小/最佳)：512 字节 / 512 字节

磁盘标签类型：dos

磁盘标识符：0x000a0cd4

设备 Boot Start End Blocks Id System

/dev/sda1 \* 2048 1026047 512000 83 Linux

/dev/sda2 1026048 62914559 30944256 8e Linux LVM

磁盘 /dev/mapper/centos-root：29.5 GB, 29490151424 字节，57597952 个扇区

Units = 扇区 of 1 \* 512 = 512 bytes

扇区大小(逻辑/物理)：512 字节 / 512 字节

I/O 大小(最小/最佳)：512 字节 / 512 字节

磁盘 /dev/mapper/centos-swap：2147 MB, 2147483648 字节，4194304 个扇区

Units = 扇区 of 1 \* 512 = 512 bytes

扇区大小(逻辑/物理)：512 字节 / 512 字节

I/O 大小(最小/最佳)：512 字节 / 512 字节

磁盘 /dev/mapper/centos-docker--poolmeta：33 MB, 33554432 字节，65536 个扇区

Units = 扇区 of 1 \* 512 = 512 bytes

扇区大小(逻辑/物理)：512 字节 / 512 字节

I/O 大小(最小/最佳)：512 字节 / 512 字节

磁盘 /dev/mapper/docker-253:0-101330881-pool：107.4 GB, 107374182400 字节，209715200 个扇区

Units = 扇区 of 1 \* 512 = 512 bytes

扇区大小(逻辑/物理)：512 字节 / 512 字节

I/O 大小(最小/最佳)：65536 字节 / 65536 字节

3.4查看所有交换分区

swapon -s

[root@localhost proc]# swapon -s

文件名 类型 大小 已用 权限

/dev/dm-1 partition 2097148 8 -1

4.网络

  4.1查看所有网络接口的属性

  ifconfig

[root@localhost proc]# ifconfig

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 172.17.0.1 netmask 255.255.0.0 broadcast 0.0.0.0

ether 02:42:e1:b8:a5:4f txqueuelen 0 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.174.129 netmask 255.255.255.0 broadcast 192.168.174.255

inet6 fe80::20c:29ff:fe50:b3b4 prefixlen 64 scopeid 0x20<link>

ether 00:0c:29:50:b3:b4 txqueuelen 1000 (Ethernet)

RX packets 28649 bytes 38411280 (36.6 MiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 8937 bytes 1226914 (1.1 MiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536

inet 127.0.0.1 netmask 255.0.0.0

inet6 ::1 prefixlen 128 scopeid 0x10<host>

loop txqueuelen 0 (Local Loopback)

RX packets 4 bytes 340 (340.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 4 bytes 340 (340.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255

ether 00:00:00:00:00:00 txqueuelen 0 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

 4.2 带宽

ethtool 网卡名

[root@localhost php-tomcat-base]# ethtool ens192

Settings for ens192:

Supported ports: [ TP ]

Supported link modes: 1000baseT/Full

10000baseT/Full

Supported pause frame use: No

Supports auto-negotiation: No

Advertised link modes: Not reported

Advertised pause frame use: No

Advertised auto-negotiation: No

<span style="color:#ff0000;">Speed: 10000Mb/s</span>

Duplex: Full

Port: Twisted Pair

PHYAD: 0

Transceiver: internal

Auto-negotiation: off

MDI-X: Unknown

Supports Wake-on: uag

Wake-on: d

Link detected: yes

看Speed

4.3查看路由表

  route -n

[root@localhost proc]# route -n

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

0.0.0.0 192.168.174.2 0.0.0.0 UG 100 0 0 eth0

172.17.0.0 0.0.0.0 255.255.0.0 U 0 0 0 docker0

192.168.122.0 0.0.0.0 255.255.255.0 U 0 0 0 virbr0

192.168.174.0 0.0.0.0 255.255.255.0 U 100 0 0 eth0

[root@localhost proc]#

 4.4查看所有监听端口

netstat -lntp

[root@localhost ~]# netstat -lntp

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 0.0.0.0:111 0.0.0.0:\* LISTEN 743/rpcbind

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 1740/nginx: master

tcp 0 0 0.0.0.0:81 0.0.0.0:\* LISTEN 1740/nginx: master

tcp 0 0 192.168.122.1:53 0.0.0.0:\* LISTEN 2194/dnsmasq

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 1543/sshd

tcp 0 0 127.0.0.1:631 0.0.0.0:\* LISTEN 12610/cupsd

tcp 0 0 0.0.0.0:40536 0.0.0.0:\* LISTEN 19964/rpc.statd

tcp 0 0 127.0.0.1:25 0.0.0.0:\* LISTEN 2133/master

tcp6 0 0 :::111 :::\* LISTEN 743/rpcbind

tcp6 0 0 :::81 :::\* LISTEN 1740/nginx: master

tcp6 0 0 :::22 :::\* LISTEN 1543/sshd

tcp6 0 0 ::1:631 :::\* LISTEN 12610/cupsd

tcp6 0 0 ::1:25 :::\* LISTEN 2133/master

tcp6 0 0 :::35420 :::\* LISTEN 19964/rpc.statd

[root@localhost ~]#

4.5查看所有已经建立的连接

netstat -antp

[root@localhost ~]# netstat -antp

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 0.0.0.0:111 0.0.0.0:\* LISTEN 743/rpcbind

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 1740/nginx: master

tcp 0 0 0.0.0.0:81 0.0.0.0:\* LISTEN 1740/nginx: master

tcp 0 0 192.168.122.1:53 0.0.0.0:\* LISTEN 2194/dnsmasq

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 1543/sshd

tcp 0 0 127.0.0.1:631 0.0.0.0:\* LISTEN 12610/cupsd

tcp 0 0 0.0.0.0:40536 0.0.0.0:\* LISTEN 19964/rpc.statd

tcp 0 0 127.0.0.1:25 0.0.0.0:\* LISTEN 2133/master

tcp 0 0 172.31.4.233:22 121.34.147.13:57190 ESTABLISHED 29540/sshd: cavan [

tcp 0 0 172.31.4.233:22 121.34.147.13:54544 ESTABLISHED 27077/sshd: cavan [

tcp 0 96 172.31.4.233:22 219.137.32.66:60645 ESTABLISHED 30315/sshd: root@pt

tcp 0 0 172.31.4.233:22 121.34.147.13:56319 ESTABLISHED 28703/sshd: cavan [

tcp6 0 0 :::111 :::\* LISTEN 743/rpcbind

tcp6 0 0 :::81 :::\* LISTEN 1740/nginx: master

tcp6 0 0 :::22 :::\* LISTEN 1543/sshd

tcp6 0 0 ::1:631 :::\* LISTEN 12610/cupsd

tcp6 0 0 ::1:25 :::\* LISTEN 2133/master

tcp6 0 0 :::35420 :::\* LISTEN 19964/rpc.statd

[root@localhost ~]#

4.6 某端口使用情况

lsof -i:端口号

[root@localhost mysql]# lsof -i:22

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

sshd 1150 root 3u IPv4 18264 0t0 TCP \*:ssh (LISTEN)

sshd 1150 root 4u IPv6 18273 0t0 TCP \*:ssh (LISTEN)

sshd 2617 root 3u IPv4 20437 0t0 TCP localhost.localdomain:ssh->192.168.174.1:60426 (ESTABLISHED)

[root@localhost mysql]#

或者

netstat -apn|grep 端口号

[root@localhost mysql]# netstat -apn|grep 22

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 1150/sshd

tcp 0 96 192.168.174.136:22 192.168.174.1:60426 ESTABLISHED 2617/sshd: root@pts

tcp6 0 0 :::22 :::\* LISTEN 1150/sshd

5.进程

  5.1查看所有进程

  ps -ef,使用ps -ef|gerp tomcat过滤

[root@localhost ~]# ps -ef

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 09:26 ? 00:00:03 /usr/lib/systemd/systemd --switched-root --system --deserialize 21

root 2 0 0 09:26 ? 00:00:00 [kthreadd]

root 3 2 0 09:26 ? 00:00:00 [ksoftirqd/0]

root 6 2 0 09:26 ? 00:00:00 [kworker/u256:0]

root 7 2 0 09:26 ? 00:00:00 [migration/0]

root 8 2 0 09:26 ? 00:00:00 [rcu\_bh]

root 9 2 0 09:26 ? 00:00:00 [rcuob/0]

root 10 2 0 09:26 ? 00:00:00 [rcuob/1]

root 11 2 0 09:26 ? 00:00:00 [rcuob/2]

root 12 2 0 09:26 ? 00:00:00 [rcuob/3]

root 13 2 0 09:26 ? 00:00:00 [rcuob/4]

root 14 2 0 09:26 ? 00:00:00 [rcuob/5]

root 15 2 0 09:26 ? 00:00:00 [rcuob/6]

root 16 2

ps -aux可以看到进程占用CPU,内存情况

[root@localhost ~]# ps -aux

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.6 126124 6792 ? Ss 09:26 0:03 /usr/lib/systemd/systemd --switched-root --system --deserialize 21

root 2 0.0 0.0 0 0 ? S 09:26 0:00 [kthreadd]

root 3 0.0 0.0 0 0 ? S 09:26 0:00 [ksoftirqd/0]

root 6 0.0 0.0 0 0 ? S 09:26 0:00 [kworker/u256:0]

root 7 0.0 0.0 0 0 ? S 09:26 0:00 [migration/0]

root 8 0.0 0.0 0 0 ? S 09:26 0:00 [rcu\_bh]

root 9 0.0 0.0 0 0 ? S 09:26 0:00 [rcuob/0]

root 10 0.

5.2实时显示进程状态

  top

[root@localhost ~]# top

top - 11:29:02 up 2:02, 2 users, load average: 0.12, 0.04, 0.05

Tasks: 408 total, 2 running, 406 sleeping, 0 stopped, 0 zombie

%Cpu(s): 1.3 us, 0.3 sy, 0.0 ni, 98.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 1001332 total, 92184 free, 370332 used, 538816 buff/cache

KiB Swap: 2097148 total, 2097140 free, 8 used. 419124 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

3001 gdm 20 0 1419364 122996 45860 S 1.0 12.3 0:06.50 gnome-shell

276 root 20 0 0 0 0 S 0.3 0.0 0:09.72 kworker/0:1

3765 root 20 0 142864 5128 3876 S 0.3 0.5 0:00.77 sshd

7740 root 20 0 146452 2384 1432 R 0.3 0.2 0:00.17 top

1 root 20 0 126124 6792 3912 S 0.0 0.7 0:03.58 systemd

2 root 20 0 0 0 0 S 0.0 0.0 0:00.03 kthreadd

3 root 20 0 0 0 0 S 0.0 0.0 0:00.37 ksoftirqd/0

6 root 20 0 0 0 0 S 0.0 0.0 0:00.39 kworker/u256:0

7 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0

8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu\_bh

9 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcuob/0

10 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcuob/1

11 root 20 0 0

6.用户

  6.1查看活动用户

   w

[root@localhost ~]# w

11:32:36 up 72 days, 20:50, 3 users, load average: 0.00, 0.01, 0.05

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

cavan pts/0 121.34.147.13 09:47 1:14m 0.17s 0.17s sshd: cavan [priv]

root pts/1 219.137.32.66 11:19 4.00s 0.05s 0.00s w

cavan pts/3 121.34.147.13 10:14 20:44 0.38s 0.11s vim Dockerfile

[root@localhost ~]#

6.2查看指定用户的信息

id <用户名>

[root@localhost ~]# id root

uid=0(root) gid=0(root) groups=0(root)

[root@localhost ~]# id cavan

uid=1001(cavan) gid=1001(cavan) groups=1001(cavan)

[root@localhost ~]#

6.3查看用户登录日志

last

[root@localhost 1]# last

root pts/0 192.168.174.1 Mon Oct 24 09:51 still logged in

(unknown :0 :0 Mon Oct 24 09:27 still logged in

reboot system boot 3.10.0-327.el7.x Mon Oct 24 09:26 - 11:35 (02:09)

root pts/0 192.168.174.1 Fri Oct 21 09:41 - 18:44 (09:03)

(unknown :0 :0 Fri Oct 21 09:15 - 18:44 (09:28)

reboot system boot 3.10.0-327.el7.x Fri Oct 21 09:15 - 11:35 (3+02:20)

root pts/1 192.168.174.1 Thu Oct 20 10:05 - 18:13 (08:08)

root pts/0

6.4查看系统所有用户

cut -d: -f1 /etc/passwd

[root@localhost ~]# cut -d: -f1 /etc/passwd

root

bin

daemon

adm

lp

sync

shutdown

halt

mail

operator

games

ftp

nobody

dbus

polkitd

abrt

unbound

colord

usbmuxd

ntp

————————————————

版权声明：本文为CSDN博主「夢\_殤」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/dream\_broken/article/details/52883883