1、1969年unix🡪1991年linux诞生

2、TCP/IP就是在unix上开发和发展出来的。Unix促使了C语言的诞生。

3、70年代中后期unix面临版权回收问题，直接或间接的导致了新的类Unix系统的诞生以及自由软件运动的的建立和发展。

1984，Richard Stallman发起了自由软件运动，并成立了自由软件基金会和GNU项目。想开发一个类Unix系统用于解决Unix的版权问题以及软件源代码面临闭源的问题。（后来没有流行起来）

80年代初期，一个大学教授谭邦宁开发了一个Minix系统（可以运行于x86 PC平台，但是是出于教学目的开发的，无发满足商用。）

Linux系统诞生于芬兰赫尔辛基大学的一位计算机系的学生 Linus Torvalsds,于是他找到了邦宁教授通过使用Minix系统和阅读Minix源代码，开始了Minux系统的锥形的设计和开发。

GPL全称为General Public License，中文名为通用公共许可，是一个最著名的开源许可协议，开源社区的Linux内核 就是在GPL许可下发布的。GPL许可是自由软件基金会创建的。

GPL许可的核心，是保证任何人有共享和修改自由软件的自由，任何人有取得，修改和重新发布自由软件的源代码权利，但都同时给出具体更改的源代码。

1991年，Linus Toralds 公开了Linux0.02版（<http://www.kernel.org>）

4、yum grouplist 查看已安装的软件包

Yum groupinstall “Compatibility libraries” “base” “Developmentools”

不要使用yum的删除功能删除软件，会删除相关依赖，导致意外问题。

5、不要忘记更改设置：

Vi /etc/sysconfig/network-scripts/ifcfg-eth0

ONBOOT=yes

提示：vi是编辑命令，相当于记事本，输入命令后，按i编辑修改，

保存命令：wq

启动或重启网卡：

ifup eth0

/etc/init.d/network restart这个更好可以TAB补全！

==等价于service network restart

6、ifconfig ping+ip

## 7、CentOS系统时间同步的步骤如下：

新装的CentOS系统服务器可能设置了错误的,需要调整时区并调整时间.

如下是CentOS系统使用NTP来从一个时间服务器同步  
cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime  
ntpdate us.pool.ntp.org

很简单吧！

下面解析一下,第一句是把当前时区调整为上海就是+8区,想改其他时区也可以去看看/usr/share/zoneinfo目录;

然后第二句是利用ntpdate同步标准时间.

没有安装ntpdate的可以yum一下:  
yum install -y ntpdate

加入定时计划任务，每隔10分钟同步一下时钟  
crontab -e  
0-59/10 \* \* \* \* /usr/sbin/ntpdate us.pool.ntp.org | logger -t NTP

<http://os.51cto.com/art/201004/192549.htm>

## 8、时间中的CST是什么意思呢？

CST China Standard Time UTC+8:00 中国沿海时间(北京时间)

## 9、如果ip获取到了但是连不上，检查几个点：

1.关闭linux防火墙

/etc/init.d/iptables stop

chkconfig iptables off

2.检查vmvare的网段

3.ping ip地址 –t 如果物理连接是ok的。（物理连接通不通）

4.telnet linux\_ip 22 <==connected通了（SSH服务通不通）

## 10、命令

**创建目录**

①mkdir /data

②cd /

mkdir data

**查看data目录本身**

ls –ld /data

相对路径：不从/开始，而是从当前目录开始。例：data/

绝对路径：从/开始的目录，就叫绝对路径。例：/data/

**查看当前路径**

pwd

创建文件

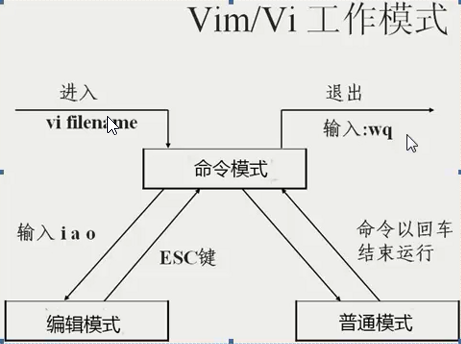
touch

**linux常见的编辑器**

vi ==相当于记事本，有编辑功能，比较弱

vim ==相当于editplus、emeditor、notpad++功能较强

**编辑文件**



vi 文件 //进入命令模式

i或a编辑 //进入编辑模式

esc //退回命令模式

：wq //保存并退出

a      //在当前光标位置的右边添加文本   
i       //在当前光标位置的左边添加文本

:w       //保存文件   
:q          //退出编辑器，如果文件已修改请使用下面的命令

**linux自动补全快捷键**

Tab //路径，命令补全

**echo打印命令**

echo “echo打印的命令" >xuanmiao.txt

将文本“echo打印的命令”写入xuanmiao.txt若xuanmiao.txt 不存在则创建

echo “echo打印的命令” >>xuanmiao.txt

将文本“echo打印的命令”追加到xuanmiao.txt文件结尾（是追加不是覆盖）

**特殊字符命令：**

“>或1>”是重定向，会清除文件的所有数据并写入；

“>>或1>>”是追加重定向，在文件结尾追加内容，不会删除已有命令；

正常输出（stdout）：代码为1，使用>或1>和>>或1>>

错误输出（stderr）：代码为2，使用2>或2>>

<或<0 输入重定向

<<或<<0 追加输入重定向

**cat**

**1.**

在使用 "cat > filename.txt" 创建文件时，可采用下面步骤:  
  
1、键入cat > filename.txt 后回车   
2、录入文本内容；   
3、键入回车；   
4、在键盘上按下Ctrl+D (或者键入Control－D)。 (退出)

**2．**

[root@xuanmiao data]# cat >>xuanmiao.txt<<aa

> 哈哈

> aa

[root@xuanmiao data]# cat xuanmiao.txt

哈哈

aa可以用任意成对内容替换

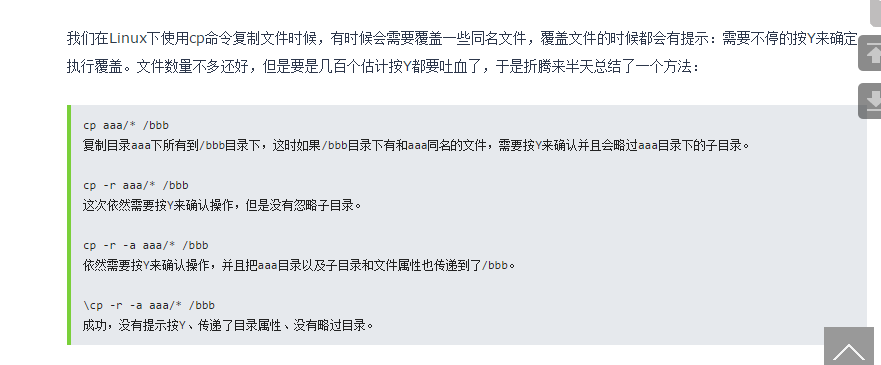
**cp**

cp –a //拷贝目录 –a相当于-pdr

-r:递归，用于复制目录；

-p:保持属性

cp –a //目录/\* 拷贝目录下的所有文件



#### bash内置命令：用help查帮助

**alias**

**bg**

**break**

**cd**

**continue**

**declare**

**echo**

**eval**

**exec**

**exit**

**export**

**fg**

**help**

**history**

**jobs**

**kill**

**let**

**local**

**logout**

**printf**

**pwd**

**read**

**return**

**set**

**shift**

**source**

**test**

**times**

**trap**

**ulimit**

**umask**

**unalias**

**unset**

#### rm -f强制删除 –r删除目录

替代方法 mv(相当于回收站)

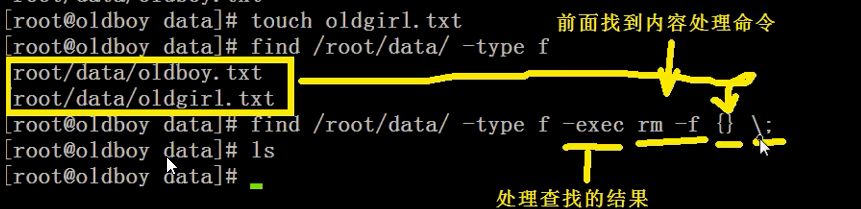
find

find /root/data –type f(文件) d(目录)

find删除

find /root/data/ -type f –exec rm –f {} \:

find /root/data/ -type f|xargs rm –f



\ 让一个有意义的字符，脱掉马甲。

.. 上级目录

. 当前目录 \.代表点号

| 管道（把前一个命令结果的输出交给后一个程序继续处理）

面试题：

删除一个目录下的所有文件，但是保留一个指定文件

**head 头部** 取文件的前n行 默认前10行

**tail 尾巴** 取文件的最后n行 默认最后10行

-f 跟踪一个文件尾部的实时变化

**demo:**

for n in `seq 105 200`;do echo $n>>demo;sleep 1;done

**demo:**

head -1 test.txt只显示文本的第一行

grep –v “除了第一行以外的文本”

1. **awk 一门语言，过滤内容（取列）**

awk ‘{print $1}’文件 <==$1第一列，$2第二列

1. **grep \*\*\*\*\*** 过滤器，把想要的或者不想要的分离开。-v 排除
2. **sed \*\*\*\*\* == stream editor**过滤：sed -n’/过滤的内容/处理的命令’ 文件

–n 取消sed默认输出

-i 改变文件内容

demo：

[root@xuanmiao data]# sed '/嗯哼/d' test.txt

hello Eof

[root@xuanmiao data]# sed -n '/嗯哼/p' test.txt

嗯哼

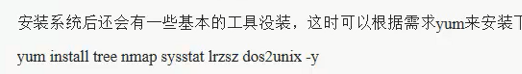
处理的命令：p print打印， d delete删除



head -30 demo|tail -11

tail -81 demo|head -11（这两个体现了思想head和tail的思想）

sed -n 20,30p demo或sed -n '20,30p' demo(简单高效的方法)



软件安装方法：

1. yum
2. 软件包下载下来 rpm格式 rpm –ivh 包名
3. 源代码 ./configure;make;make install

yum update不能轻易做（更新可能会导致软件无法使用）

线上系统尽量不做，除非有已知的安全漏洞。

查看Linux 版本：

登录到linux执行cat /etc/issue

**yum linux包管理器（可以解决依赖问题）（也是调用了rpm帮你找到依赖关系包并下载）**

dmeo :

yum install tree(包名) -y

**rpm –iv（安装、显示输出）**

demo:

rpm –ivh 包名.rpm(提前下载好的)

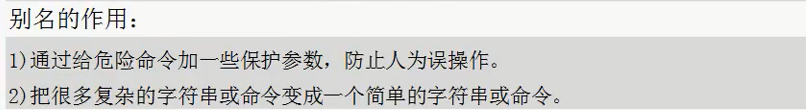
**最大的问题： 依赖问题不好解决**

**如何使操作不提示直接进行：**

1.使用命令的源路径

2.在命令前加\

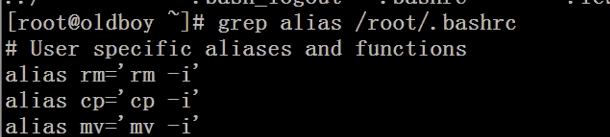
**因为linux里面有个别名的概念（使用前需要确认）如何cp实际上为cp –i,cp不需要确认但是cp -i需要确认**



**别名定义位置：针对root用户： /root/.bashrc**

**如果希望所有用户都生效：/etc/bashrc 或/etc/profile**

**生效： source /etc/profiles**



使用alias查找当前别名

demo:

alias xuanmiao="echo This\'s xuanmiao\'s Linux0.0"

unalias 取消别名

**在man里面可以使用/进行查找按n键查找下一个**

**less 工具**

Less 工具也是对文件或其它输出进行分页显示的工具，应该说是linux正统查看文件内容的工具，功能极其强大。less 的用法比起 more 更加的有弹性。在 more 的时候，我们并没有办法向前面翻， 只能往后面看，但若使用了 less 时，就可以使用 [pageup] [pagedown] 等按键的功能来往前往后翻看文件，更容易用来查看一个文件的内容！除此之外，在 less 里头可以拥有更多的搜索功能，不止可以向下搜，也可以向上搜。

#### 乘法口诀

touch cfkj.sh

vim cfkj.sh

输入以下内容：

#!/bin/awk -f

BEGIN{

for (i=1;i<=9;i++)

{

  for (m=1;m<=i;m++)

  {

    printf m"\*"i"="m\*i"  "

  }

  print

}

}

chmod 755 cfkj.sh

./cfkj.sh

运行效果

#### 

**~ 代表用户的家目录 例如：对root~代表/root**

**seq ==sequence n1 n2 n3 n1起始数字n2增量n3结束数字**

**seq –s = 5**

**输出结果：**



**find \\*结尾字符 find要加一个转义字符/**

**ls \*结尾字符**

**闲谈莫论他人是，静坐多思自己非。**

**修改用户**

usermod -g group loginname   
强行设置某个用户所在组  
usermod -G groups loginname  
把某个用户改为 group(s)   
usermod -a -G groups loginname  
把用户添加进入某个组(s）

# shell 1>&2 2>&1 &>filename重定向的含义和区别

投稿：mdxy-dxy 字体：[[增加](javascript:turnbig()) [减小](javascript:turnsmall())] 类型：转载 时间：2015-04-15[我要评论](http://www.jb51.net/article/64183.htm#comments)

这篇文章主要介绍了shell 1>&2 2>&1 &>filename重定向的含义和区别,需要的朋友可以参考下

当初在shell中, 看到">&1"和">&2"始终不明白什么意思.经过在网上的搜索得以解惑.其实这是两种输出.

在 shell 程式中，最常使用的 FD (file descriptor) 大概有三个, 分别是:

**0 是一个文件描述符，表示标准输入(stdin)**  
**1 是一个文件描述符，表示标准输出(stdout)**

**2 是一个文件描述符，表示标准错误(stderr)**

在标准情况下, 这些FD分别跟如下设备关联:   
stdin(0): keyboard 键盘输入,并返回在前端   
stdout(1): monitor 正确返回值 输出到前端   
stderr(2): monitor 错误返回值 输出到前端

**举例说明吧:**

当前目录只有一个文件 a.txt.   
[root@redhat box]# ls   
a.txt   
[root@redhat box]# ls a.txt b.txt   
ls: b.txt: No such file or directory 由于没有b.txt这个文件, 于是返回错误值, 这就是所谓的2输出   
a.txt 而这个就是所谓的1输出

**再接着看:**

[root@redhat box]# ls a.txt b.txt 1>file.out 2>file.err   
执行后,没有任何返回值. 原因是, 返回值都重定向到相应的文件中了,而不再前端显示   
[root@redhat box]# cat file.out   
a.txt   
[root@redhat box]# cat file.err   
ls: b.txt: No such file or directory   
一般来说, "1>" 通常可以省略成 ">".   
即可以把如上命令写成: ls a.txt b.txt >file.out 2>file.err   
有了这些认识才能理解 "1>&2" 和 "2>&1".   
1>&2 正确返回值传递给2输出通道 &2表示2输出通道   
如果此处错写成 1>2, 就表示把1输出重定向到文件2中.   
2>&1 错误返回值传递给1输出通道, 同样&1表示1输出通道.   
举个例子.   
[root@redhat box]# ls a.txt b.txt 1>file.out 2>&1   
[root@redhat box]# cat file.out   
ls: b.txt: No such file or directory   
a.txt   
现在, 正确的输出和错误的输出都定向到了file.out这个文件中, 而不显示在前端.   
补充下, 输出不只1和2, 还有其他的类型, 这两种只是最常用和最基本的.

**>是重定向符，就是把前面输出的内容重定向到后面指定的位置，比如（例1）：**

[?](http://www.jb51.net/article/64183.htm)

|  |  |
| --- | --- |
| 1 | echo "一些内容" > filename.txt |

上面例子会把 "**一些内容**" 写入到**filename.txt** 文件中。

**>**前是可以加数字来说明把什么内容重定向到文件中，默认是把标准输出重定向到文件中，所以下面这个例子和上面那个是一样的（例2）：

[?](http://www.jb51.net/article/64183.htm)

|  |  |
| --- | --- |
| 1 | echo "一些内容" 1> filename.txt |

如果是错误信息就不会输出到filename.txt（例3）：

[?](http://www.jb51.net/article/64183.htm)

|  |  |
| --- | --- |
| 1  2 | $ ls nodir 1> filename.txt  $ ls: nodir: No such file or directory |

上面这个例子中nodir不存在，所以通过ls命令查询时错误信息会输出到 2(stderr)，但我们指定的是把1重定向到filename.txt，所以上面命令执行完后，filename.txt中是没有内容的。但是执行下面命令就会把错误信息写入到filename.txt中（例4）：

[?](http://www.jb51.net/article/64183.htm)

|  |  |
| --- | --- |
| 1  2  3 | $ ls nodir 2> filename.txt  $ cat filename.txt  $ ls: nodir: No such file or directory |

**& 是一个描述符，如果1或2前不加&，会被当成一个普通文件。**

**1>&2 意思是把标准输出重定向到标准错误.**

**2>&1 意思是把标准错误输出重定向到标准输出。**

**&>filename 意思是把标准输出和标准错误输出都重定向到文件filename中**

我们再看一个例子（列5）：

[?](http://www.jb51.net/article/64183.htm)

|  |  |
| --- | --- |
| 1  2  3 | $ ls nodir 1> filename.txt 2>&1  $ cat filename.txt  $ ls: nodir: No such file or directory |

上面例子把 **标准输出** 重定向到文件 **filename.txt**，然后把 **标准错误** 重定向到 **标准输出**，所以最后的错误信息也通过标准输出写入到了文件中，比较例3，4，5，就能明白其作用。

下面是来自百度知道的内容，大家可以参考下：

问：Linux重定向中 >&2 怎么理解？  
问题补充：echo "aaaaaaaaaaaaaaaa" >&2 怎么理解？

答：

>&2 即 1>&2 也就是把结果输出到和标准错误一样；之前如果有定义标准错误重定向到某log文件，那么标准输出也重定向到这个log文件  
如：ls 2>a1 >&2 （等同 ls >a1 2>&1）  
把标准输出和标准错误都重定向到a1，终端上看不到任何信息。

位置参数：

