# Oeasy教您玩转linux系列实验1

## 0安装环境

### virttualbox 备份 设置 快照

## 零基础快速上手linux

### 图形与终端

#### 查看内核版本uname

##### Linux是什么uname

###### Linux,发音

###### Linux不是操作系统,

linux是操作系统(加超莲)的内核Linux [ˈlɪnəks] [ˈlɪnʊks] 两种发音 **丽娜克斯** 和 **李扭克斯** 都可以

###### (https://baike.baidu.com/item/内核)名称

###### https://github.com/torvalds/linux

###### https://www.kernel.org/

##### 如何查看内核 uname 命令

###### 选项—help可以看到到相关参数

##### Logo，它的名字叫做Tux

这只企鹅就是吉祥物tux，为什么企鹅,芬兰有企鹅么?原来北极还真有,后来让人给吃没了.发音是[tʌks].

南极不属于任何一个国家,linux不属于任何一个公司.不知道我们聊天用的那只站起来带微博的企鹅是怎么来的.

Tux是在Linus Torvalds和Alan Cox的建议下，由Larry Ewing于1996年创作的，这是来自维基百科的解释。但历史总是比它看上去要复杂。我们知道，当时大多数的讨论都是通过邮件列表进行的。在1996 年，Linux内核正处于繁忙开发之中，最有名的领袖就是Linus Torvalds（创始人）和Alan Cox（排行第二）。开发者们都想要一个Linux标识，去直面Windows。很多人在讨论鲨鱼，或者其他和BSD精灵一样强的东西。但是 Torvalds提出了企鹅的想法，原文是：

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

Re: Linux Logo

Linus Torvalds (torvalds@cs.helsinki.fi)

Sun, 12 May 1996 09:39:19 +0300 (EET DST)

.

Umm.. You don't have any gap to fill in.

"Linus likes penguins". That's it. There was even a headline on it in some Linux Journal some time ago (I was bitten by a Killer Penguin in Australia - I'm not kidding). Penguins are fun.

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

早期的linux邮件列表https://lkml.org/lkml/1996/5/12/136 ,发现了kernel组里面在96年的时候讨论了为什么企鹅tux成为了吉祥物...oeasy也想有个吉祥物...用什么东西做oeasy的吉祥物合适呢?

事实上，Torvalds只是被澳大利亚国家动物园的一只小企鹅轻轻地咬了一下，但这个想法就这样提出来了。

他也有可能是被一部简短的电影《Creature Comforts》所影响，里面有一只相似的企鹅。

在解释理由的时候，发起者如是说：

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

All the other logos were too boring - I wasn't looking for the

"Linux Corporate Image", I was looking for something \_fun\_ and

sympathetic to associate with Linux. A slightly fat penguin that sits

down after having had a great meal fits the bill perfectly.

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

当时产生了很多标识，事实上，Tux并没有赢过其他的标识。所以，它算是一个吉祥物（mascot），而不能说是标识（logo）。

1996年，Larry Ewing用Gimp创作了这只可爱的企鹅。这幅图片遵守GNU革奴，源文件可以在作者的博客找到。

而Tux这个名字则来自James Hughes：https://lkml.org/lkml/1996/6/10/79  
-----------------------  
Re: Let's name the pguin! (was: Re: Linux 2.0 really \_is\_ released..)  
.  
James Hughes ([jamesh@interpath.com](mailto:jamesh@interpath.com))  
Mon, 10 Jun 1996 20:25:52 -0400  
.  
(T)orvalds (U)ni(X) --> TUX!  
-----------------------  
  
最初，Tux并没有获得预想中的赞赏。毕竟，Torvalds是在用一只胖胖的企鹅来和当时最强的公司微软作对抗。何况Tux并没有在标识比赛中取胜。但Tux最终在野兽微软身边找到了自己的位置，还是多亏了Torvalds：  
-----------------------

https://isc.tamu.edu/~lewing/linux/

德克萨斯A&M大学

http://vger.kernel.org/vger-lists.html邮件列表 找到kernel

##### 什么机器跑着linux?

世界前500的计算机(https://top500.org/lists/top500/2020/06/),中,90%以上的操作系统是linux,而且越来越多.第四的是来位于我国无锡国家超算中心(超级计算机中心)(http://www.nsccwx.cn/),2016年上线的神威太湖之光https://top500.org/system/178764/

「」

看看他有多少核心有多少内存,系统是Sunway RaiseOS 2.0.5

###### 每个人手里了计算机云android

###### 小米路由器(http://www.miwifi.com/)使用的OpenWrt(https://openwrt.org/)

###### 谷歌公司推出的vr系统daydream(https://arvr.google.com/daydream/)

我们可以用netcraft(idc(Internet Data Center)观察公司)(https://searchdns.netcraft.com/)搜索(https://searchdns.netcraft.com/?restriction=site+contains&host=microsoft.com&position=limited)微软子域名(微软官网(http://microsoft.com)名下的网站,发现大多数服务器用的也是linux,谷歌、淘宝、腾讯基本都是linux

###### 微软公司目前也在windows系统里面提供wsl供大家来使用.

##### 两条学习道路

###### 关于Linux的学习方法，有两种：一种是从GUI(图形界面)入手，轻松快速,但是很难深入.另一种方法则是从shell(Linux命令行)入手，Linux命令是才是精髓所在，一切的东西都要找到最根本的方式,但是缺点是学习过程枯燥乏味,非常难走。

###### gui与cli两种

###### 打开linux终端terminal的四种方式

双击xfce

在指定文件夹右键

所有应用程序,终端模拟器

1991年做出来的 minix的一个改进版

Linux在数据中心操作系统上的份额可能高达70％

其中一个最显着的趋势是近年来微软的转变。 如果你十年前告诉软件开发人员，Windows操作系统将在2016年“运行Linux”，他们中的大多数人会歇斯底里地笑了。 但是Windows Linux子系统（WSL）的存在和普及，以及最近宣布的诸如Docker的Windows端口（包括Windows上的Linux容器）支持等功能都证明了Linux具有的影响 - 以及 显然将会继续存在 - 跨软件世界。在全球顶尖的500台超级计算机中，有约60%安装了Linux操作系统。

目前世界上运算速度最快的超级计算机“神威太湖之光”（Sunway TaihuLight），运行的国产操作系统神威睿思（RaiseOS 2.0.5），就是基于Linux的。

这就涉及到Android（安卓）是什么，依旧是某百科答案：

*Android是一种基于Linux的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，由Google公司和开放手机联盟领导及开发。*

###### uname可以查看内核版本

注意linux命令是大小写敏感的,一般命令名为小写

###### 切换界面

实验楼的操作系统实验

##### 其他内核

其他系统的内核xnu(xnu is not unix) 一个混合内核, 奠基于卡耐基·梅隆大学开发的 Mach kernel , 它的组件来自 FreeBSD 和用于编写驱动程序的 C++ API “。(mac、ios的内核)等

Windows的内核的名字我们不知道,只能通过version看到版本号.

索尼 PS4 的操作系统就是基于 ---

###### Gnu有个还没有实现的内核叫做hurd

###### 那么这个内核究竟在哪呢?

#### 什么是发行版distro

##### 乌班图ubuntu

###### 字符界面里面看到了这个内核,那前面这个ubuntu是什么?

###### 其名称来自非洲南部祖鲁语或豪萨语的“ubuntu"一词，意思是“人性”“我的存在是因为大家的存在"，是非洲传统的一种价值观。

###### linux是内核,ubuntu是发行版.

##### 发行版的测试

Linux发行版是各种应用程序（软件）的集合

###### http://mirrors.kernel.org/内核上对于各个发行版评价

###### https://distrowatch.com/

###### https://distrowatch.com/search.php

##### 发行版的关系

###### 发行版是基于发行版的 比如,前十的发行版都是基于RHEL和debian的Debian [ˈdɛbiən] 念 得(dei)变

Debian严格来说不是一种linux，而是一个unix like操作系统计划的统称，除了linux/freebsd内核外，还有hurd/opensolaris内核，目前只有linux和freebsd相对成熟。

https://www.debian.org/CD/http-ftp/可以看到各种架构

Mint基于ubuntu开发

ubuntu(基于debian稳定版)

kali基于debian

MX Linux（基于Debian稳定版本开发）

Cat /etc/issue

cat /etc/os-release

cat /proc/version

Debian是免费发行版,1993.12月第一次出现.的作者是.名字来源是.debian是一个独立的组织,Debian组织着5万个以上的软件包和320百万行代码,各种项目的负责人是选出来的.

支持的架构也多,比如龙芯兼容的mips

Debian网站

Deepin曾经进过前十,是国产的linux发行版,界面非常漂亮

你想过自己做个linux发行版吗?从哪个发行版衍生呢?

Rhel是最早的商业发行版1994出现,99年上市,首日收益历史第八,2012年10亿美元回报,在商用服务器上使用非常多,centos

Centos、fedora、mandriva免费基于rhel企业版收费

https://sourceforge.net/projects/uck/

Distro hop 折腾发现版来玩的人

https://distrotest.net/

**Fedora是免费开源社区版,rhel是商业开源软件,商业软件的意思是可以通过服务挣钱,开源的意思是开放源代码,**

fedora是redhat的小白鼠,有了特性先在社区里面玩去

Fedore是社区版本，红帽赞助。一般一些新的产品会在fedora优先发布，供大家体验，同时发现bug或者提出更好的建议。等到很稳定的时候，就加入到redhat的版本。

Redhat市值非常高已经被ibm所收购.

centos的代码与rhel完全一致，只是修改为符合开源协议的版本，以及修改发行版名称和源等内容。把收费的组件和客户的支持都去掉了.内核和功能代码完全一致，因此稳定性也基本与rhel一致，相当于官方免费版。

centos的免费分发版本，实际上是大大促进了RHEL系统在Linux领域的实际占有份额。也为招揽潜在的付费商业客户提供了巨大的优势。如果centos能成为事实上的linux领域标准操作系统，就像windows和android。那么各类商业客户也非常容易倾向于使用RHEL。如果从这个目的出发，Centos会一直保持免费开源，并会保持和RHEL版本的高度一致性。基于这个目标，我们甚至可以推测，被RedHat收购了的Centos将比之前更“像”RHEL。

德国有人做了发行版suse, Novell对SUSE收购

###### 非标准的linux

Google的安卓

三星的Tizen

Ubuntu touch

Lg的webos

Firefox OS

###### 其他类unix系统

Bsd(mac、ios)

Solaris

Unix

#### 查看当前路径pwd

##### Cd=change directory

###### 我在哪里pwd

###### 这里有什么ls

###### 这些东西有的是文件、有的是文件夹.

###### 我们的各种程序、数据都是以文件的形式存储的.

##### pwd

###### 直接运行pwd

###### 在图形界面找到位置

###### 路径的表示

把路径从gui直接拖动到terminal

###### 这是一个command,他在硬盘里面还是内存里面呢?

##### 查看系统信息cat

###### 内核在哪

内核最开始在硬盘里面,在/boot目录下,系统开机之后被加载到内存里

内核直接在内存运行

###### 内核都干什么

内核初始化initial之后,就接管了硬件,可以分配系统资源(cpu,内存,以及外部设备的资源等). 内核是操作系统中离硬件最近的一层,提供一些基本的功能,让cpu内存和外设可以接受外面的指令.

外面的指令比如运行某个程序,cpu把他从硬盘读取到内存,分配cpu资源让执行.比如问询一下当前时间或者.

###### 内存

什么是内存

Cat 是命令只运行cat 观察

后面的是参数,你到底要查看谁呢?

/proc/meminfo是硬盘里面的一个文件

可以用ls 看到他

cat /proc/meminfo

内核虽然自己在内存里面,但是可以把内存空间分页,然后可以支持不同的程序同时访问内存

如果内存不够了怎么办?

Cat /proc/vmstat

虚拟内存

虚拟内存

换个参数

Cat /proc/zoneinfo

Dma是什么

内核本身就在内存里面,怎么还能控制内存呢?靠的是cpu

###### cpu

什么是cpu

cat /proc/cpuinfo

这里面可以观察到当前cpu支持的指令集

Cat /proc/version

观察当前版本内核版本,可以看到当前的指令集,当时就是从kernel.org下载的c文件,然后用这个版本的gcc编译出来的内核,叫做这个名字

什么是乌班图呢?

Cat /proc/version

cat /etc/issue

cat /etc/\*release

分别看看里面有什么

##### man

#### 查看手册man

##### =manual

###### Man pwd

###### Man ls

###### Man man

快捷键

按h可以进入帮助

按q推出

Ctrl+f向下翻页

Ctrl+B 向上翻页

j向下一行

/搜索

Q退出

Pwd –help不能用,找到他

##### man和-help对比

###### man更详细

###### help更简略

#### 这儿都有啥ls

##### what is ls

##### ls可以有参数吗?

###### ls 看看当前位置都有什么

把路径从terminal复制粘贴到地址栏

把路径从gui直接拖动到terminal

###### clear进行清屏操作 ctrl+c

###### 总结三板斧、互相找路径

###### 参数

可以看/ 可以看用户文件夹 可以看桌面

Ls /b …tab键,然后就可以有变化选择一个路径 自动补全功能

在cli粘贴的时候就是ctrl+shift+v

在cli复制就是ctrl+shift+=c

##### 选项-开关

Options

ls -l 我们可以看到颜色文件夹是蓝色的

ls -a 可以看到隐藏文件

-a -l

可以写到一起-al

写成la可以么

-h可以分开写 human readable

ls -lah

ls -hal

六种排列组合

##### 既使用参数又使用选项

#### 那啥在哪whereis?

##### Whatis pwd

##### Whereis pwd

##### 找到文件夹具体位置

##### 使用绝对路径执行

##### https://www.cnblogs.com/shengs/p/4088738.html

##### 命令执行过程990915

##### Whereis clear

##### Whereis whatis

##### Whereis whereis

##### 灵魂第二问

##### 发现有两个以上的地址,怎么办,哪个是?

#### 到底哪个which

##### 灵魂之问whatis clear

##### 灵魂二问whereis clear

##### 灵魂三问 which clear

##### Whatis which

##### Whereis which

##### Which which

##### 内建命令

##### https://en.wikipedia.org/wiki/Shell\_builtin

##### Which pwd

##### Which /bin/pwd这个就是一个存储于硬盘的外部命令

##### Pwd –help

##### /bin/pwd –help

##### 思考

###### 如果我把这个pwd删除了会怎么样

###### pwd 还能运行么?

###### 硬盘文件删除了 pwd还能运行么

#### 清空屏幕clear

##### 灵魂三问

#### 命令行快捷键总结

##### 复制粘贴

##### 记忆命令

###### 上下箭头

###### Ctrl+R搜索命令历史

##### 快速移动

###### 方向键

###### 按照单词移动

###### 直接到头 ctrl+a

###### 直接到尾 ctrl+e

##### 快速删除

###### 直接忽略 ctrl+c

###### 全部删除 Ctrl+u

###### 删除到尾 Ctrl+k

###### Ctrl+d log out登出系统 登出和关闭terminal有区别吗,用户和系统之间有个session,logout session就断了

### 终端应用实例

#### Yes命令作用不断输出y

##### Man yes观察选项和参数

###### 参数

Yes oeasy

###### 选项

Man --help

Man –version

###### 回头看

回头看man ls(ls是man的参数还是选项)

Ls -a是参数还是选项

Man pwd是什么

#### Apt软件包管理

##### 如何安装应用

###### https://www.debian.org/mirror/list

###### 先搜索

###### Man Apt-get

###### 如果你要装的这个包甲依赖于那个包乙,而那个包乙又依赖于包丙,这复杂的链路,你一个个下载太麻烦了,于是可以交给这个命令来做这个事情.

###### 尝试update,不允许

#### 新命令sudo

##### 先三问再type

##### Apt-get更新 update

sudo 这个时候注意方向键控制光标,ctrl+a到终端开始位置,ctrl+e到终端结尾位置

升级man命令

先三问

Man --version

apt-get upgrade man

man –version

#### linuxlogo

##### 三问

##### 观察所在位置

##### man显示

##### 这就是新安装的软件

#### apt-get remove

###### 发现提示当中有可以remove的程序

###### 试试remove删除应用的方法 remove一个

##### Autoremove的方式

##### 继续观察man apt-get

###### Ls /etc/apt/sources.list

#### Cat拼合文本concatenate

##### Cat /etc/apt/sources.list

##### 找到地址,用火狐访问

#### 换源

##### 搜apt-get换源

##### https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu

##### 控制版本

##### 修正list文件

##### 换源成功

##### 还能装什么应用

#### 文字处理Figlet

###### 直接运行 写文字Oeasy

###### 三问

###### Man

###### https://aotu.io/notes/2016/11/22/figlet/index.html

###### 命令行世界的banner

#### toilet

##### 安装

###### 随时三问

###### Man

##### Toilet

###### 直接运行,写文字oeasy

###### Man 观察

#### 火焰动态图形

##### Aafire

###### 安装

###### 三问

###### 都有些什么

#### Sl

##### Apt-get install sl

##### Man sl

##### 这个东西装在哪里了?

##### 灵魂三问

#### 水族馆

asciiquarium

##### 新建终端

###### 新建终端选项卡快捷键

###### 分离选项卡

###### 在选项卡之间跳转

##### 用终端打开终端?

找到对应命令

找到文件位置

##### 为终端设置快捷键

###### Ctrl+T

#### 🎁CMatrix

##### 安装

##### 手册

##### 玩耍

##### Whereis …

##### 在服务器什么位置

##### apt-cache showsrc cmatrix

#### apt-cache

##### man apt-get

##### apt之后按tab观察提示

##### apt-cache search matrix

#### 应用管理aptitude

##### Aptitude

##### 安装运行

##### 观察

#### Cowsay

##### 先安装

##### 然后man

##### 然后说句话

##### Cowsay -l

##### 在/usr/games里面可以找到源代码

#### box

##### ascii view

asciiview elephant.jpg -driver curses

sudo apt install aview

$ sudo apt install imagemagick

#### 查找功能 apropos

##### apropos “list”

##### apropos Chinese

##### apropos apt

#### 中文输入法fcitx

##### 打开输入法

##### 直接运行sogou

##### 切换输入法

##### 重启环境

##### 观察输入法

##### 找到输入法

##### 更新输入法

##### 右键快捷方式观察应用名称,然后直接在terminal跑

##### 也可以输入中文

##### 观察terminal的编码格式,修改一下观察在运行apt-get会有乱码出现

#### 管道pipe|

##### Cowsay ls

##### Ls|cowsay

###### Pwd

Pwd」cowsay

Pwd|toilet

Pwd|toilet|cowsay

###### 本来应该输出到屏幕标准输出的字节流,通过管道流到了某一个命令那里

Ls|donkeysay

Ls | figlet

Ls |more

ls |less

Ls|figlet|cowsay

█ **date | figlet** ▐▌

Box

http://bbs.liuli9.cn/244.html

#### more

#### less

#### Fortune

##### 下载

##### 文档

##### 运行

█ **fortune ascii-art** ▐▌

##### 管道

##### 两边分别带参数

**fortune | cowsay -f duck**

#### 其他fortune-zh

##### Apt-cache search fortune

##### 安装zh

##### 然后让牛说中文

##### 然后让狐狸说中文火狐

##### 通过fortune 输出重定向两首古诗文本文件

##### cat 悯农.txt 静夜思.txt >古诗两首.txt

##### cat -n 古诗两首.txt

### 桌面应用举例

#### linux的界面gui 估衣

##### 桌面gui打开关于xfce

###### 查看关于

###### echo $DESKTOP\_SESSION

###### $echo $DESKTOP\_SESSION查看应用的桌面程序

Xfce — Xfce体现了模块化和可重用性的传统 UNIX哲学。它由许多组件组成，这些组件可提供人们可以期望的现代桌面环境的全部功能，同时又保持相对轻巧。它们是单独包装的，您可以从可用的包装中选择以创建最佳的个人工作环境，就是个轻量级的桌面。KDE Plasma — KDE Plasma桌面环境是一个熟悉的工作环境。Plasma提供了现代桌面计算体验所需的所有工具，因此您从一开始就可以高效地工作。功能最丰富，占用资源最多。GNOME — GNOME GNOME [ɡˈnoʊm] [noʊm] 两种发音 **格弄姆** **弄姆**

**隔奴目** 都可以

XFCE是一个桌面环境，就像GNOME和KDE。它包含了一系列应用程序，比如根窗口（？）、窗口管理器、文件管理器、面板等等。XFCE是用GTK2 toolkit写的，同时也包含了其自己的开发环境（库、守护进程等），和其他大型的桌面环境差不多。但与GNOME和KDE不同的是，XFCE是一个轻量级的桌面环境，设计上软多地参考了CDE而不是Windows或Mac。其开发周期较长，但很可靠且运行速度非常快。，XFCE特别适合用于老硬件的环境。

为什么使用XFCE？  
下面列出了一些使用XFCE（主观的）理由：  
速度。比任何一个主要的桌面环境都要快。

###### Cinnamon，MATE，KDE

KDE桌面图形环境是Linux几大桌面里的老大，也是第一个诞生出来的linux桌面环境，所以相对来说KDE桌面环境和KDE软件生态这两个方面会更全面更稳健更完善更完整，体验会更好。KDE官方不仅仅是做桌面，KDE旗下软件生态这一块真的是其他桌面没法比的。

GNOME [ɡˈnoʊm] [noʊm] 两种发音 **格弄姆** **弄姆** 都可以

如果你以前听过这个传闻：当 GNOME3 第一次发布时，很多 GNOME 用户还没有准备好放弃 GNOME 2。

[Mate](https://link.zhihu.com/?target=https%3A//mate-desktop.org/)（以 *马黛茶(yerba mate)*植物命名）项目的开始是为了延续 GNOME 2 桌面，刚开始时它使用 GTK 2（GNOME 2 所基于的工具包），然后又合并了 GTK 3。由于 Linux Mint 的简单易用，使得该桌面变得非常流行，并且从那时起，它已经普遍用于 Fedora、Ubuntu、Slackware、Arch 和许多其他 Linux 发行版上。今天，Mate 继续提供一个传统的桌面环境，它的外观和感觉与 GNOME 2 完全一样，使用 GTK 3 工具包

###### Linux的主要场景是服务器,一般使用ssh命令行进行连接,所以图形界面并不重要.五色令人目盲.

#### 修改文本gedit

##### 新建文档

##### 灵魂三问

##### man

##### 修改 用浏览器 source.list拖过来

##### 然后改乱

##### 观察apt-get update

#### 桌宠Oneko

#### 新建终端、新建终端页

##### 终端所在位置

#### Xeyes

###### 安装使用

###### 观察参数

#### 运行火狐

##### 直接运行

##### Man firefox

##### Firefox参数细节

###### Firefox shiyanlou.com

###### 结束进程

###### Firefox -h

###### Firefox -v

###### Firefox –search 123

#### nautilus

##### 从终端开启文件夹

##### Apt-get update

##### Apt-get install nautilus

#### 这个文件装在哪里了?

##### 找到这个文件

##### 我可以在这个gui的地方打开终端

##### 然后在这个终端直接打开这个nautilus就是当前位置了

##### 文件夹的相互拖动

##### 文件的相互拖动

##### 拖动来

##### 我如果在别的位置也想直接打开怎么办?nautilus加上绝对地址,就打开了

#### 更新xfce

$ sudo apt update

$ sudo apt install xfce4

Xfce-about 然后观察

## vim初级入门

### 理解模式

#### 进入与退出

##### 进入退出

###### 进入vi 进入 开始的时候默认进入的是normalmode

###### 注意normal模式下 NormalMode

###### 想要退出怎么办ctrl+c

###### 最下方出现提示

###### 按下:在屏幕左下角会出现:这个时候就进入了命令行模式CommandLineMode

###### 命令行模式来自与古老的行编辑器ex

###### 注意大小写！vi对于命令是大小写敏感的

###### 输入qa!quit all buffers force!

###### 再来一次

#### 观察帮助

###### 观察界面

###### ：q

###### 因为我们什么也没有写所以可以直接q退出

##### help q

#### 打开文件

##### 进入ls > oeasy.txt

##### cat oeasy.txt

##### Vi oeasy.txt 可以打开文件

##### 波浪线～tilder是还没有内容的行紫色行

##### 注意插入的波浪线和前面的波浪线是不一样的

##### 退格又变回道空白波浪线

##### 可见的范围是80\*24个字符大小的范围

##### 用方向键控制光标运动

##### 用hjkl控制光标运动,这个时候hjkl无法输入字母,vim-adventure。Com第一关可以免费玩

##### 用ctrl+g确认当前文件路径和文件名，观察状态栏

##### 要退出ctrl+c

##### 注意normal模式下 :qsa

#### 修改文件

##### 修改文件

###### Vi oeasy.txt

###### 复习上次的上下左右,翻页练习

###### 当前的模式是命令模式

###### 注意一下左下角的状态现在什么也没有

###### 找到第3行摁下i

###### 注意左下角会出现插入两个字,意味着当前进入了插入模式

###### 然后修改文件,输入hjkl

###### 输入完成之后esc,注意左下角插入两个字消失,重新进入了命令模式

###### :q提示尚未保存,强制退出用q!

###### q!强制退出

###### cat发现并未修改

##### 修改文件2

###### 重复上次过程,找到第三行

###### I进入插入模式,观察右下角状态栏

###### 插入hjkl四个字符

###### Esc退出插入模式,进入编辑模式

###### ：q观察提示

###### :w

###### Help :w :write注意要打冒号 w 是write的缩写状态

###### 观察状态栏信息

###### 显示当前文件名字,多少行,多少个字符

###### 保存完毕

###### :q退出

###### Head oeasy.txt

#### 插入状态

##### 准备素材

###### Vi o2.txt 新建并打开o2.txt

###### 开启中文写一首诗

###### 锄禾当午

###### 汗滴下土

###### 谁知盘中

###### 粒皆辛苦

##### i和a

###### i

help i

esc把当前状态切换为命令

把光标移动到当上

我要在“当”字前面插入“日”字

输入命令指令i

状态右下角切换到插入模式

当字前面插入文字,光标放在当上,按下i

输入日,文字出现在当前面日当无

###### A

Help a

Ecs退出插入模式,进入命令模式

把光标移动滴上面

这回要在滴的后面插入字符

摁下a,状态切换到插入状态

输入禾字,这个时候插入字符出现在后面

总结一下i代表insert在当前光标前面插入

A代表after在当前光标后面插入

##### A和I注意A，a不一样，说明vi是大小写敏感的

###### A

Esc切换模式,把光标放在第三行,盘的位置

如果这个时候我i就在盘前面插入

如果a就在盘后面插入

但是这个时候我摁下A,注意这个时候是大写

A和以前的小写a对应不同的两个命令

就在本行的结尾插入,输入一个餐字

###### I

Esc调整到最后一行,摁下一个I,就会到整个行的最前面,插入一个粒字

##### o和O

###### o

esc退出插入模式,我想插入一个结尾

但是我处于命令模式

如何快速在当前行下面插入一个新行?

光标移动到皆字上面,使用命令o

切换到插入状态,写下全文完

###### O

这个时候我想插一个开头“悯农”

Esc切换到命令状态,移动到第一行

O,切换到插入状态

写下悯农

Esc

:wq

保存观察

### 光标行间运动

##### 基本运动

###### 回忆o和O区别，有个实用的技巧

比如你在一句话的中间，处于插入模式，这个时候你想下面加一行，你得移动到末尾在回车很麻烦

如果想快速到行尾，需要esc然后A然后回车，这样简单了一点，如果您觉得esc离您太远或者感觉那个触控板的按键没有感觉，可以用ctrl+c回到正常模式。

也可以ctrl+o 切换到插入-普通模式，插入普通模式就是让你执行一次普通模式的命令，然后还是在插入模式，你可以ctrl+o，o，继续插入

或者要调整位置《c-o》 zz 继续

当然A、I和O也可以使用。

###### 把oeasy.org的主页下载下来wget

###### Hjkl基本运动

###### 回车和j的区别,回车到下一行行首

###### 方向键基本运动

###### 在插入状态也可以用方向键做基本运动

###### 如果左右键配合ctrl就是以单词为单位左右运动

###### 有的时候打字过程中，发现自己好像手指次序不对，抬头一看果然错了，用快捷键

###### <C-h> 删除一个字符 h 是左一格 相当于退格键

###### <C-w> 删除一个单词word

###### <C-u> 删除到行首

###### 这套逻辑和在bash里面输入命令是一致的不用单独记忆

###### 当然上次学的A、I也是移动工具

###### int oeasy = 1

###### double o2z = 2

###### String oThreeZ = “3”

###### 比如在这个前面添加注释

##### 显示行号

###### 在vimrc中配置

###### 显示标尺set ruler

###### 跳转到指定行12G

gg Go to the first line

G Go to the last line

nG Go to line n

n% Go to n% in file 配合标尺

##### 行运动

nG|：移动到当前行的第n列

###### 向上3行3k

###### 向下3行

###### 向左10字符

###### 向右5字符

###### 跳转到最后一行G round

###### 跳转到第一行gg

###### 整理缩进gg=G整理全部代码缩进

###### 这样整理代码就非常迅速了，程序员更多的时候工作在normal状态，浏览、搜索、跳跃、复制、粘贴。真正聚焦下来写代码再切换到插入状态。而其他编辑器永远只有一个插入状态，其实是浪费了很多键盘功能的。这也是Normal模式被称作Normal正常模式、被当作默认模式的原因。

##### 配置文件

###### 出厂配置运行vim

vim -u NONE

###### 显示行号:set number

###### 不显示行号:set nonumber

###### 那这个显示行号在哪里设置的呢？

###### vimrc

###### d$ 全都删了

###### 运行 没有任何配置

###### 或者set number

###### help set number

##### 配置显示信息

###### 蓝布棉门帘

###### " show

###### set ruler " show the current row and column

set ruler                   " 打开状态栏标尺

syntax enable  
syntax on                   " 自动语法高亮

###### set number " show line numbers

###### set nowrap “自动折行

###### set showcmd " 显示正在输入的命令 display incomplete commands

###### set showmode " display current modes

###### set showmatch " jump to matches when entering parentheses 显示配对的括号

###### set matchtime=2 " tenths of a second to show the matching parent匹配括号高亮的时间（单位是十分之一秒）

##### 缩进控制

###### 插入模式下tab 和退格

###### 命令模式下>> <<

###### 查看帮助文件:h >

##### 自动缩进配置

:set tabstop=4 设定tab宽度为4个字符

:set shiftwidth=4 设定自动缩进为4个字符

###### help tabstop

###### 自动缩进

set autoindent " Respect indentation when starting a new line.

set expandtab " Expand tabs to spaces. Essential in Python.

set tabstop=4 " Number of spaces tab is counted for.

set shiftwidth=4 " Number of spaces to use for autoindent.

set backspace=2 " Fix backspace behavior on most terminals.

##### 语法高亮和自动缩进

###### 语法高亮

syntax on " Enable syntax highlighting.

filetype plugin indent on " Enable file type based indentation.

colorscheme murphy " Change a colorscheme.

可以找到所有color对应的文件夹

选择配色模式

也可以下载更多配色模式。

https://vimcolors.com/

##### 屏幕内运动

H Go to top of screen

M Go to medium screen

L Go to bottom of screen

nH Go n line from top

nL Go n line from bottom

zt Bring the current line near the top of your screen

zz Bring the current line to the middle (half) of your screen

zb Bring the current line near the bottom of your screen

##### 翻页

Ctrl-e Scroll down a line

Ctrl-d Scroll down half screen

Ctrl-f Scroll down whole screen

Ctrl-y Scroll up a line

Ctrl-u Scroll up half screen

Ctrl-b Scroll up whole screen

###### 用ctrl+f向下翻页

###### 用ctrl+b向上翻页

：h digraphs-table

#### 光标行内移动

##### 行内移动

###### 0：移动到当前行的第一个字符处 ^：移动到当前行第一个非空字符处 g\_：移动到当前行最后一个非空字符处 $：移动到当前行最后一个字符处

###### 移动到行首0

###### 移动到行尾$

###### 移动到本行文字首^

###### I 其实是 ^i

###### A 其实是$a

###### o 其实是A回车

###### O是ko 也就是kA回车

###### 配合一张图类似于P67的4.4

0 Go to the first character in the current line

^ Go to the first nonblank char in the current line

g\_ Go to the last non-blank char in the current line

$ Go to the last char in the current line

n| Go the column n in the current line

f Search forward for a match in the same line

F Search backward for a match in the same line

t Search forward for a match in the same line, stopping before match

T Search backward for a match in the same line, stopping before match

; Repeat the last search in the same line

, Repeat the last search in the same line backwards

##### 单词间移动

###### 移动到下一个单词头w

###### 移动到上一个单词头b

###### 移动到下一个单词尾e

###### 画一个图类似于P67的4.4

###### 如果是3w就是摁三次w

###### 对应w,2w,3w的图

###### 同理3b就是b三次

###### e就是e三次

##### 单词间移动进阶

###### 光标放到第四行

###### 回忆上次的wbe的基本操作

###### ^回到本行文字首

###### W跳跃的是词组,词组之间用空格分开的

###### W跳跃的是单词,单词之间用各种分割符都可以分割

###### 同理E可以跳到词组的结尾

###### B往回跳词组

##### 句段间运动{}

###### Wget http://users.csc.calpoly.edu/~dekhtyar/466-Fall2010/labs/lab1/tomsawyer.txt

###### （）句子间跳动

###### 3）

###### {}段落间

###### 3}

###### Vi tomsawyer。Txt +8 第八行打开文件

###### ：302

###### 302G

##### 对称的括号之间运动

###### %

##### marker

###### ：marks

ma Mark position with mark "a"

`a Jump to line and column "a"

'a Jump to line "a"

'' Jump back to the last line in current buffer before jump

#### 查找定位

##### 行内查找 f F

###### 给加号左右加上空格

###### ；，

###### String s =”add(oeasy”+o2z+”o3z”+o4z+”o5z”);

###### 如果；按多了一次可以用，回来

##### help search

##### /命令

set ic/noic                  " 在查询及模式匹配时忽赂大小写/不忽略大小写(缺省)

set ws/nows                  " 在搜索时如到达文件尾则绕回文件头继续搜索

向前向后

##### \*查找当前单词

###### 适合找同一个函数的调用，在函数内统一个变量的遍历。

##### 吧tomsawyer变成tomcat

##### 通过查找和cwTomcat<esc>批量修改控制成.

##### n和N配合移动，如果要修改就. 如果不要的话就是n ，往回找就是N

##### 总结.重做的模式，一键移动，一键重做

##### jk可以上下行来做

##### \*/可以nN来移动

##### f+ 可以用；，来移动

##### 这个命令在ed时代就有了一样的s///

###### 今天在火狐也可以通过/来查找相当于ctrlf

###### esc退到命令格式

###### 搜索chapter大写

###### n向下遍历

###### 5n向下找到第五个

###### N向上遍历

###### 3N向上找第三个

###### a进入插入模式

###### 把oeasy变成o2z

###### 高亮显示set hlsearch

##### 快速查找

###### 在当前单词位置+8

##### ?命令

###### Esc退到命令格式

###### 找到中间的100行左右

###### ?oeasy向上遍历

###### n向上遍历

###### N向下遍历

###### 两者正好相反

/ Search forward for a match

? Search backward for a match

n Repeat last search (same direction of previous search)

N Repeat last search (opposite direction of previous search)

:n1,n2s/word1/word2/g

:1,$s/word1/word2/g 或 :%s/word1/word2/g

:1,$s/word1/word2/gc 或 :%s/word1/word2/gc

###### info ed

###### bill joy开发，visual是相对于原来unix文本编辑器ex，ed这样的行编辑器而言

###### vim在terminal中运行，实际上是一个基于终端的文本编辑器

###### 升级到vim

###### Vim stands for Vi IMproved. It used to be Vi Imitation

###### 作者

###### ssh登陆的时候，没有图形界面，但是用vim可以方便的编辑

###### 各发行版的linux、mac、windows都可以使用vim，在前两者中都是预装好的

###### 可以进行升级

###### vim –version

###### windows中可以访问http://www.vim.org在download中下载

###### 其他变种 nvi = new vi

###### elvis=

###### vile vi like emacs

### 文本修改

#### 基本字符插入

##### h i\_CTRL-v i代表insert模式下的

###### 颜文字 复制粘贴过来 用terminal的粘贴功能

###### <C-v>065

###### Unicode.org

###### <C-v>u00bf

###### <C-v>u1234

###### <C-v>u3456

###### 正常模式ga 观察编码方式

###### 二合字符插入非常用字符

###### h digraphs-default

?I

12 二分之一

14

34

38

25

+-

se section

co copyright

rg registed sign

###### 从剪贴板复制粘贴，插入状态，根据shell的快捷键来

##### 计算

###### 寄存器做计算 《C-r》=3+3《C-r》就会输入6

###### 如何查询插入状态的快捷键：h i\_CTRL-r

#### 删除字符

##### x删除当前光标字符X向前退格

###### 首先esc命令模式

###### x删除当前光标字符

###### 继续x继续删除

###### 把oeasy变成o2z

###### 10x可以将x执行10次

#### 撤销命令

##### 回忆如何加注释

##### u撤销一步

###### 一步一步的撤销,一直回到最原始

##### help undo

##### ctrl+r

##### 控制插入的技巧

##### 如果一大块程序一气呵成的话感觉是很好的

##### 但是有的时候，有思路的停顿时候，可以形成esc的习惯，思路继续的时候再o。

##### 这样撤销的时候，可以分段的撤销，不会一下子撤回到最最开始啥都没有的状态。

##### 问题是什么时候停顿，esc呢？当你思路已经中断的，不知道该干什么的时候，当你问自己我是不是需要回到正常模式上下翻翻的时候，就esc吧。

##### 如果你只是在小范围上下翻看，可以使用方向键，当您使用方向键的时候

###### 模式自动先切换到正常

###### 然后移动

###### 然后切换回插入

###### 这样也可以控制撤销的粒度

##### U撤销一行中所有的改变

#### 替换字符

##### r替换当前光标字符

###### 把html变成hxml

###### 把title变成didle

###### 把head变成dead

###### 修改中文一个字

###### 把会玩才会学,变成会学才会玩.

##### R进入替换模式

###### 修改当前单词并一路改下去

###### 注意状态栏

###### 退格回到原始字符

###### 修改会玩才会学

##### ~改变大小写

###### 大小写修正

### 复制剪切

#### dd删除本行

##### dd 删除本行

##### 如果要删除别的行 可以配合方向 然后.

##### 如果连续的 可以…

##### 特别多的话也可以

##### 我要删除10行10dd

##### 100dd全删了吗?

##### G是最后一行,我先到第一行gg

##### dG都删了哈哈哈

##### dgg

##### 观察状态栏

#### 删除之后可以粘贴pP，这个剪贴板到底在哪·？：reg

#### 行命令模式的复制粘贴

二、复制多行

任务：将第9行至第15行的数据，复制到第16行

方法1：（强烈推荐）

：9，15 copy 16  或 ：9，15 co 16

由此可有：

：9，15 move 16  或 :9,15 m 16 将第9行到第15行的文本内容到第16行的后面

\*\* 使用操作符 y 复制文本，使用 p 粘贴文本 \*\*

:10,100>  
第10行至第100行缩进  
:20,80<  
第20行至第80行反缩进

>G一直到最后一行都缩进

1，3w o.txt

3,5t 44

12,34d

#### vim可以写程序，写配置文件，写文档

##### unix上面ken tompson

Ken Thompson 创建了行编辑器

1966 年，贝尔实验室聘用了 Ken Thompson 。Thompson 刚刚在加州大学伯克利分校完成了电气工程和计算机科学的硕士学位。在伯克利他使用一个名为 QED 的文本编辑器，该编辑器在 1965 到 1966 年间被开发用于伯克利分时系统。Thompson 到达贝尔实验室后做的第一件事就是为麻省理工学院兼容分时系统重写 QED。他后来又为 Multics 项目写了另一个版本的QED。在重写过程中，他对程序进行了扩展，以便用户可以在文件中搜索某一行，并使用正则表达式进行替换。

与伯克利的分时系统一样，由麻省理工学院、通用电气和贝尔实验室合作的 Multics 项目试图创建一个可行的商业分时操作系统。最终，AT&T 认为这个项目毫无进展并退出。在没有分时系统的情况下，Thompson 和贝尔实验室资深研究员 Dennis Ritchie，开始怀念分时系统所提供的“交互式计算的感觉”，并着手创建他们自己的版本，该版本最终发展成为 Unix。1969 年 8 月，在妻子和幼子外出去加州度假时，Thompson “给操作系统、shell、编辑器和汇编程序分别分配了一个星期”，将新系统的基本组件组合在一起。

这个编辑器被称为 ed 。它是基于 QED 的，但并不完全是 QED 的复现。 Thompson 决定放弃某些 QED 的功能，弱化了对常规的表达式的支持，因此 ed 只能理解相对简单的正则表达式。QED 允许用户打开多个缓冲区同时编辑多个文件，但是 ed 一次只使用一个缓冲区。QED 可以执行包含命令的缓冲区，而 ed 则不能。这些简化可能是必要的。Dennis Ritchie 曾说过，去掉 QED 的高级正则表达式是“并不大的损失”。

ed 现在是 POSIX 规范的一部分，所以如果你有一个符合 POSIX 的系统，你的电脑上就安装了 ed 。现在，许多 ed 命令都是 Vim 的一部分，因此，这就值得摆弄一番了。例如，你必须使用 w 命令来写入磁盘缓冲区，必须使用 q 命令来退出编辑器。这两个命令可以写在同一行命令中，也就是 wq。ed 与 Vim 一样，是一个模态编辑器；若要从命令模式进入输入模式，取决于你试图如何转换文本，需使用 insert 命令（i）、append 命令（a）或 change 命令（c）。ed 还引入了s/foo/bar/g语法来查找和替换或“替换”文本。

1

2

3

1，5p

10i

111111

Ctrl+c

9，13p

10d

Ctrl+c

9，13p

W

Q

#### :1,40 y

#### :2,4 w oeasy.txt

### 高级编辑

#### operator motion

##### h operator

##### h motion

##### db

#### 总结operator

#### 总结motion

#### 重复工具

##### “oeasy o2z o3z o4z o5z”

##### x 删除当前字符

##### . 相当于x 重做

##### 好像并没有什么用

##### u撤回来 这次dw

##### 我们的操作是dw

##### 重做每次删除一个w

##### 这次的任务是A;

##### int oeasy = 1

##### double o2z = 2

##### String oThreeZ = “3”

##### j.j.j. 这个好像很小心谨慎的样子

##### 不过不用害怕 jj了之后k可以回来 不影响.

##### .多了之后u可以回来

#### 重复深入

##### “oeasy o2z o3z o4z o5z”

##### dw d2w 2dw 都可以

##### 很长也知道多少个单词 怎么办？…要跳过某行怎么办 7w b .

##### 需要插入三个rel="external nofollow"

##### 找到位置然后摁 .

##### 然后 .

##### help .

##### 有的时候需要先插入id=“” 然后class=“” 我们要复制的是class=“”，而不复制id=“”怎么办？

##### 没关系 我们可以用esc退出，结束上一个输入过程，然后再insert balabala esc 这样我们就用esc控制了重复范围的粒度，这个力度其实也是撤销时候对应的粒度

#### cw

#### y

#### 大小写的变化

##### ~……

##### g~5l …

##### g~aw aUaw guaw

###### 可重复

##### :h operator

###### c d y g～ gu gU >

##### :h motion

#### aw、iw aW 、iW

##### daw

##### 假设System.out.println(“oeasy o2z o3z”);光标在a，现在要求删除oeasy，应该怎么办？dbxx 或者 bdw 少了一个词

##### diw 和daw

##### cap

##### 区别

##### daw的可重复性更强

##### 在o4z上面.一下就可以了。

##### 当然也有diW daW

##### :h daw

##### 这种找到最少字符解法的方式叫做vimgolf，有个网站可以去玩一下。

###### String str = “oeasy o2z o3z o4z o5z”

###### 我需要只保留oeasy和o6z 怎么办？

###### 可以d4w（delete 4 words） 也可以4dw（delete word 4 times）但是哪个更好的呢？dw…更好，因为有的时候看不清几个单词，要自己把握少了还得继续dw，多了的话直接接撤销到最开始了，dw…就不会有这种问题，可以一步步撤销还可以移动继续 …

###### 这三种方法哪个最好呢？我们可以从重复复性来看，dbxx重复x bdw重复dw daw重复整体 如果还有类似操作 我们就可以用.来直接完成了

###### <div class= “mystyle”></div>

用diW可以批量操作

养成好习惯之后，下一操作该做的时候，你会发现.能让你会心一笑

###### System.out.println(“oeasy”);

yy p ft2”ci”再输入

##### 删除 xml tag 中的内容

#### 查找配合编辑t（

###### dt(

###### ct”

###### yt

###### dT(

##### c

###### ciw 之后插入一些字母 整体算一个操作 然后移动配合 . 重做操作

OeasyManager。Call(oeasy,30);

在句首，yt，o 《C-r》0 输入，50);

:reg 观察寄存器 存储的是《C-r》0对应第0个寄存器

##### C

##### S和s

###### s删除当前字符，并进入插入状态

###### S删除当前行，并进入插入状态

help S::^C

help s::cl

ciw

vis

cit in tag

https://www.jianshu.com/p/438eef92a94c

###### String s =”add(oeasy”+o2z+”o3z”+o4z+”o5z”);

###### 用；再次查询 用. 代表s + 批量给+左右加空格

###### 如果；按多了一次可以用，回来

能否把 + 替换为 空+空

##### 复制粘贴

###### 控制reg

<li ><a href="./category/game/game.html" target="showFrame">游戏</a></li>"

把其中的target属性 复制下来yt>

然后找到合适的地方粘贴

#### 可视化模式

##### 可视化模式的复制粘贴

###### 正常模式是用操作符加上一个动作

###### 原来是操作+范围

可视模式是先选好范围再使用d、y、c等操作符

1. 定位到下面标记有 ---> 的一行，将光标移动到 "a)" 之后。

2. 接着使用 v 进入可视模式，移动光标到 "first" 的前面。

3. 现在输入 y 以抽出(复制)高亮的文本。

4. 然后移动光标到下一行的末尾：j$

5. 接着输入 p 以放置(粘贴)复制了的文本。然后输入：a second <ESC>。

6. 使用可视模式选中 " item."，用 y 复制，再用 j$ 将光标移动到下一行末尾，

用 p 将文本粘贴到那里。

---> a) this is the first item.

b) this is the firstc

###### 可视化模式初探

###### 可视化模式是选择好了之后输入动作ydc

##### 选择范围控制

###### 可视化的范围就是开头到结尾，

###### 切换开头和结尾o

###### 可视化模式也支持fF，；、/？nN

###### 可视化模式有三种面向

v 面向字符的可视化模式

shift + v 面向行的可视化模式

选择多行之后d或者y

ctrl + v面向列块的可视化模式

oeasy

o2z

o3z

o4z

三种方式，左 下 e 选择 到o4z后面粘贴p

在可视化模式中也可以进行切换

gv 是 重选刚才的选区

可视化模式批量添加注释# 添加注释//

在删除注释

###### vi 意思是visual

可视化编辑 相对于ed来的

对 Thompson 和 Ritchie 来说， ed 已经足够好了。但是其他人则认为它很难用，而且它作为一个淋漓尽致地表现 Unix 对新手敌意的例子而臭名昭著。在 1975 年，一个名叫 George Coulouris 的人在伦敦玛丽皇后学院的 Unix 系统上开发了一个改进版 ed 。Coulouris 利用他在玛丽女王学院的视频显示器开发他的编辑器。与 ed 不同的是，Coulouris 的程序允许用户编辑在屏幕中的一行代码，通过一次次击键的方式来操作行（想象一下在 Vim 中每次编辑一行）。 Thompson 拜访玛丽女王学院时，看到 Coulouris 已经写好的程序，驳斥道他不需要在编辑文件的时候看到它的状态。受此启发，Coulouris 将他的程序命名为 em，或者“为凡人而生的编辑器”。



（George Coulouris）

1976年，Coulouris 把 em 引入了加州大学伯克利分校，在那里他用了一个夏天的时间在 CS 系访学。这是 Ken Thompson 离开伯克利去贝尔实验室工作十年之后的事了。在伯克利，Coulouris 遇到了 Bill Joy，一名伯克利软件发行公司（BSD）的研究生。Coulouris 斯向Joy 展示了 em， Joy 以 Coulouris 的源代码为基础，为扩展 ed 建立了一个名为 ex 的改进版 ed。1978年，1.1 版本的 ex 与第 1 个版本的 BSD Unix 捆绑在一起。ex 在很大程度上与 ed兼容，但它增加了两种模式：一种“开放”模式，这种模式可以使 em 单行编辑成为可能，还有一种“可见”模式，这种模式会占据整个屏幕，并且可以像我们今天所习惯的那样，对整个文件进行实时编辑。



（Bill Joy）

1979 年的第 2 版 BSD 引入了一个名为 vi 的可执行文件，它只在可视模式下打开 ex 。

ex/vi （后来称为 vi）建立了我们现在使用的 Vim 中大多数的约定，但这些约定当时并不是 ed 的一部分。Bill Joy 使用的视频终端是 Lear Siegler ADM-3A，它的键盘没有光标键。而是，**h、j、k 和 l 键**上绘制光标键，所以 Bill Joy 在vi 中就使用这些键来进行光标移动。ADM-3A 键盘上 escape 键位置是今天我们所使用的键盘上的 tab 键，这也就解释了为什么这样一个难以够着的键会被用来实现像退出当前模式这么常见的操作。前缀命令的 **:**字符同样也来自 i，它在常规模式下（即运行 ex 进入的模式）使用 : 作为提示。这解决了一个 ed 中被长期诟病的问题，也就是一旦启动之后，没有任何反馈信息向用户致以问候。在可见模式下，保存和退出需要使用现在仍在使用的经典 wq。“Yanking”和“puttng”、标记、以及用于设置选项的 set 命令都是原始 vi 的一部分。我们今天在 Vim 中使用的的基本文本编辑过程，都是 vi 中使用的特性。



vi 是除 ed 之外唯一与 BSD Unix 捆绑的文本编辑器。在那个时候，Emacs 可能会花费数百美元（这是在 GNU Emacs 之前），所以 vi 变得非常流行。但是 vi 是 ed 的直接衍生版本，这意味着如果没有 AT&T 的源代码，源代码就不能被修改。这促使一些人创建了 vi 的开源版本。 STEVIE （专门为 VI 爱好者的 ST 编辑器）出现于1987年，Elvis 出现于 1990 年，nvi 出现于 1994 年。其中一些克隆版本添加了额外的功能，如语法高亮和窗口分离。尤其是 Elvis ，它的许多功能被整合到 Vim 中，因为许多 Elvis 用户推动了这些功能的加入。)

##### 关于缩进的总结

###### 批量操作

如果想要缩进很多行代码的话就按照下面做

1 //在这里按下'v'进入选择模式

1

1

1

1

1

1//光标移动到这里，再按一次大于号'>'缩进一次，按'6>'缩进六次，按'<'回缩。以下同理

###### 命令方式>>

###### 可视化模式

###### 命令行模式

###### 自动缩进

#### 可视模式选择范围

##### 批量添加注释 批量删除注释

### 批量替换

#### esc批量替换

##### ：s help s

###### 替换试试

##### :3，5s/adfas

##### %s/oeasy/o2z/

##### u

##### %s/oeasy/o2z/g

##### u

##### %s/oeasy/o2z/c

##### U

##### %s/oeasy/o2z/gc

##### 在最下面提示

##### 开头统一加注释

##### 开头统一去除注释

##### 结尾统一加分号

###### %3，5s/$/;$/gc

##### 关于注释方式的总结

#### cw c$

#### V模式y

##### 就可以知道：q这个命令是干什么用的

###### :only

###### Ctrl+g

###### 在正常状态下ZZ命令也可以保存并退出

##### h！

##### h 42

#### vim历史

Bram Moolenaar 创建了 Vim

“Vim”现在是“改进版 Vi”的缩写，而最初代表的是“模拟版 Vi”。和其他许多“vi克隆版本”一样，Vim 始于在一个无法使用 vi 的平台上复现 vi 的一个尝试。在荷兰 Venlo 一家影印公司工作的软件工程师 Bram Moolenaar 想要为他全新的 Amiga 2000 准备一款类似于 vi 的编辑器。Moolenaar 已经习惯了在大学时使用的 Unix 系统上的 vi ，当时他 已经对vi了如指掌。所以在 1988 年，Moolenaar 使用当时的 STEVIE vi克隆版本开始在 Vim 上工作。

（Bram Moolenaar，2006 年加入 Google）

Moolenaar 接触到 STEVIE 缘于其曾经出现在一个叫 Fred Fish 的磁盘上。Fred Fish 是一名美国程序员，每个月都会寄出一张软盘，内含为 Amiga 平台提供的精选可用开源软件。任何人只要支付邮费就可以得到一张这样的磁盘。有若干版本的 STEVIE 曾在 Fred Fish 磁盘上发布。Moolenaar 使用的 STEVIE 版本在 Fred Fish 256 号磁盘上发布。（令人失望的是，Fred Fish 磁盘似乎与 Freddi Fish 没有任何关系。）

Moolenaar 喜欢 STEVIE，但很快就注意到其缺失了很多 vi 命令。因此，在第一次发布 Vim 时，Moolenaar 优先考虑了 vi 的兼容性。当时已经有其他人编写了一系列的 vi 宏，当运行一个合适的 vi 兼容编辑器时，可以求解一个随机生成的迷宫。Moolenaar 能够让这些宏在 Vim 中运行。1991年，Vim 以 Vi模拟为名第一次发布于 Fred Fish 591 号磁盘。Moolenaar 添加了一些特性（包括多级撤销和解决编译器错误的“quickfix”模式），这意味着 Vim 已经完成了对 Vi 的超越。在 1993 年通过 FTP 发布 Vim 2.0 之前，Vim 都仍以 Vi模拟 的身份存在。

在众多互联网合作者的帮助下，Moolenaar 稳健地在 Vim 中加入了一些功能。Vim 2.0 引入了对wrap选项的支持，以及对长行文本进行水平滚动的支持。受到了vi克隆nvi的启发，Vim 3.0 增加了对分割窗口和缓冲区的支持。Vim 现在还将每个缓冲区保存到交换文件中以避免程序崩溃造成文件丢失。Vim 支持语法高亮显示，第一次出现是在 Vim 5.0 中。与此同时，Vim 的受欢迎程度也在不断增长。它被移植到 MS-DOS、 Windows、Mac，甚至被移植到 Unix 与原来的 vi竞争。

2006 年，Vim 被 Linux Journal 读者评为最受欢迎的编辑器。如今，根据 2018 年 Stack Overflow 的开发者调查，Vim 是最受欢迎的文本模式（即终端模拟器）编辑器，受用于 25.8% 的软件开发人员(和 40% 的 Sysadmin / DevOps 人员)。在 1980 年代末和整个 1990 年代，程序员一度发起了“编辑器战争”，将 Emacs 用户与 vi （即最终的 Vim ）用户进行了对比。虽然 Emacs 肯定仍有一些追随者，但有些人认为编辑器战争已经以 Vim 获胜而结束。2018年 Stack Overflow 的开发者调查显示只有 4.1% 的受访者使用 Emacs，也验证了这个事实。

Vim 是如何变得如此成功的？显然，人们喜欢 Vim 所提供的特性。但我认为，Vim 背后的悠久历史表明了它的优势远不仅仅体现在其功能集上。Vim 的代码库可以追溯到 1988 年，当时 Moolenaar 开始研究它。另一方面，“ wq 文本编辑器”——关于 Unix-y 文本编辑器应该如何工作的更广泛的愿景——可以追溯到半个世纪以前。“ wq 文本编辑器”有一些不同的具体表达方式，但在某种程度上要感谢 Bill Joy 和 Bram Moolenaar 对向后兼容性非比寻常的关注，才使好的想法逐渐积累起来。从这个意义上说，“ wq 文本编辑器”是运行时间最长、最成功的开源项目之一，得益于计算机世界中一些最伟大的思想贡献。我不认为“创业公司无视所有先例来创造颠覆性的新软件”的开发方式都是不妥的，但 Vim 提醒我们，这种协作和增量的方式同样能产生奇迹。

### 练习

#### vimtutor zh

## linux文件操作

### linux文件操作

#### apache2的安装

##### 安装

###### 更新源

###### 安装

[阿帕奇](https://wenwen.sogou.com/s/?w=%E9%98%BF%E5%B8%95%E5%A5%87&ch=ww.xqy.chain)音译为Apache，是北美[印地安人](https://wenwen.sogou.com/s/?w=%E5%8D%B0%E5%9C%B0%E5%AE%89%E4%BA%BA&ch=ww.xqy.chain)的一个部落，叫阿帕奇族，在美国的西南部。相传阿帕奇是一个武士，他英勇善战，且战无不胜，被印第安人奉为勇敢和胜利的代表，因此后人便用他的名字为印第安部落命名，而阿帕奇族在印第安史上也以强悍著称。

##### apache2的使用

###### 灵魂三问

###### 手册观察

###### 启动

##### apache2的浏览

###### localhost

###### 找到位置

###### 用cat的方式打开文件

###### 是否可以自己新建网页,下次再说

#### 新建并更新touch

##### 新建一个文件

###### 浏览oeasy.html-没有

###### 新建文件oeasy.html

###### 进入浏览器观察 成功

###### Ls -l 注意日期

###### 再次touch

###### 再次ls -l

###### 然后就更新了日期

##### 新建一组文件

###### 批量新建{2,3,4}.html

###### 在文件夹观察

###### 在浏览器浏览 成功

##### 新建连续范围文件

###### Touch {4..20}.html

###### 观察 ls -lah

###### 顺序有问题

###### 我希望是1.html变成01这样子

##### 首先要删除

###### 找到单字符的文件?.html

###### 然后删除 -delete

###### 观察

##### 然后添加

###### touch {01..30}.html

###### 观察

##### 文件和文件夹有什么不同

#### 标准输出 >

##### 写oeasy.html

###### Ls

###### Ls 》 01.html

###### Sudo ls 〉01.html

##### 修改访问权限

###### Ls -l 01.html

###### 观察到访问权限

###### Sudo chmod 777 01.html

###### Ls -l 01.html

###### 再观察01.html可以修改了

##### 输出重定向

###### Ls 》 01.html

###### 观察网页

###### Ls -l 〉01.html

###### 观察网页

###### Cat 01.html

##### 总结

###### 》输出重定向

###### 把原来输出到屏幕的内容，输出到了一个文件

###### 我想输出指定的文字怎么办？

#### 显示echo

##### 显示内容

###### Echo oeasy

###### 输出重定向

###### Echo oeasy 》 oeasy.html

###### 查看源文件

###### Cat oeasy.html

###### 在浏览器中打开

##### 输出中文

###### Echo oeasy教您玩转linux

###### 然后重定向到oeasy。Html

###### 然后在浏览器中显示

###### 查看源文件

###### 查看本地字符集

###### Locale

###### 控制编码格式再次显示

###### 解决乱码问题

##### 设置字符集

###### Echo 《meta charset=utf-8》

###### 失败

###### Echo “《meta。。。》”

###### Charset的意思是字符集

###### 本句话的意思是说明当前字符集

###### Echo “《meta..》oeasy教你玩转linux”〉 oeasy.html

###### 刷新网页

###### 查看源文件

###### 如何加回车？下次再说。

#### 转义字符

##### 加上转义字符

###### Man echo

###### Echo \x41

###### Echo “\x41”

###### Echo “\x42”

###### Echo “\x43”

###### Echo \x51

###### Echo \50

###### Echo \49

###### Echo \4F

###### Echo “\x4f\x45\x41\x53\x59”

###### Echo “\x6f\x65\x61\x73\x79”

##### 制表符

###### Echo “oeasy\t教程“

###### Echo “oeasy\t教程“>01.html

###### Firefox 01.html &

###### 查看源文件

###### Cat 01.html

##### 退格和重头

###### Echo “oeasy\b123”

###### Echo “oeasy\r123”

###### 输出到01.html

###### Firefox浏览

##### 换行符

###### Echo ”oeasy\n玩转\nlinux”

###### Echo “oeasy\nlinux”>oeasy.html

###### Firefox 。。

###### 我们希望在html里面换行 显然失败了

#### html换行符

##### 搜html换行符

###### Echo oeasy《br》linux

###### 输出到01.html

###### 浏览器浏览

##### 输出中文

###### Echo oeasy教程

###### Echo 《meta charset=utf-8》

###### 在输出中文

##### 输出带标签英文

###### 《h1》《/h1》

###### 《p》《/p》

###### Cat

###### Firefox

###### 太麻烦 能否在源文件基础之上再追加输出

#### 追加输出〉》

##### 追加〉》

###### echo ”oeasy gogogo“〉》01.html

###### cat

###### firefox

##### 追加带标签内容

###### 《hr》

##### 追加cowsay

###### Cowsay

###### Cowsay 〉02.html

###### Firefox

###### Vi 02.html

###### Echo “<pre>”

###### 再追加cowsay

###### Firefox 02.html

###### Cat 02.html >> oeasy.html

###### Firefox

#### Touch文件练习

##### Touch 31.txt

##### touch ../2/「1..28」.txt

##### touch ../3/31

##### slash词源字典

**1540s, "to cut with a stroke of a blade or whip;" 1650s, "to strike violently," perhaps from Middle French esclachier "to break," variant of esclater "to break, splinter" (see slat). Meaning "to clear land" (of trees) is from 1821, American English. In reference to prices, it is attested from 1906. Related: Slashed; slashing. Slash-and-burn for a method of clearing forest for cultivation is from 1919.**

##### .txt

##### touch ../../summer/「4..6」/「1..30」.txt

##### touch ../../ autumn /「7..9」/「1..30」.txt

##### cd ../cd..

##### 这里使用的是文件的相对地址.

##### 如果要使用文件的绝对抵制应该如何呢?

##### touch summer/5/31.txt

##### touch autumn /7/31.txt

##### touch autumn/8/31.txt

##### touch winter/10/31.txt

##### touch winter/10/31.txt

#### 类型是啥Type

##### Type直接回车

##### 试试各种东西的type

###### Type pwd

###### Type man

###### Type clear

###### Type oeasy

##### 对type灵魂三问

###### Whatis

###### Where

###### which

##### 使用type去反问

###### Type whatis

###### Type whereis

###### Type which

###### Type type

##### 查看type文档

###### Man type但是找不到

###### 搜索找到页面

https://en.wikipedia.org/wiki/Type\_(Unix)但是没有细节

https://www.gnu.org/software/bash/manual/html\_node/Bash-Builtins.html有细节

###### 先试试

-t

Type -t whatis

Type -t whereis

Type -t which

Type -t /bin/pwd

-a

三问灵魂

-p

-P

#### Alias

##### alias y=yes

##### y oeasy

##### 恶作剧Alias ls=sl

##### alias yo=yes oeasy

##### man alias

##### Alias dunkeysay=

##### alias

###### ll

###### la

###### lsa

三种命令的总结可执行的程序,内建命令,别名

可以用which来进行区分

Type ls

下面这几个命令，可能是你天天都在敲的。所以，你应该设置成 alias 来提高效率

alias nis="npm install --save "

alias svim='sudo vim'

alias mkcd='foo(){ mkdir -p "$1"; cd "$1" }; foo '

alias install='sudo apt get install'

alias update='sudo apt-get update; sudo apt-get upgrade'

alias ..="cd .."

alias ...="cd ..; cd .."

alias www='python -m SimpleHTTPServer 8000'

alias sock5='ssh -D 8080 -q -C -N -f user@your.server'

#### ls的细节

##### 排序

###### touch 「a-z」

###### touch [A-Z]

###### touch [0-9]

###### /sort 搜索和排序相关的

###### -X排序

###### -r 反序

###### -U无序

###### ,…-m 可以把jar包列表都直接坐好.

###### 可以按照三种方式排序/sort

###### 时间、大小、名称

#### 输出重定位

##### >

who > users

##### >>

##### 追加信息

##### asd 2>error.txt 输出错误信息

##### 保证信息和错误都写入文件

命令 > 文件 2>&1

标准输出和标准错误结合使用  
# grep "root" /etc/passwd > 1.log 2>&1  
2 >& 1 意思是将错误输出重定向到句柄1标准输出;综合起来就是grep命令执行过程中产生的标准输出和错误输出都会被重定向到1.log.中;

0 stdin 标准输入  
1 stdout 标准输出  
2 stderr 标准错误  
stdin 大多数bash指令通过stdin接受输入 也可以使用重定向命令  
stdout 默认的标准输出的位置是终端监视器 可以使用重定向符号改变标准输出的输出位置  
stderr 默认的错误输出的位置是终端监视器 可以使用重定向符号改变标准输出的输出位置  
将标准输出,标准错误分别重定向到不同的文件中

#### 分辨文件类型file

##### 先touch再file

##### Whereis pwd

###### File pwd

#### 观察头部head

##### head oeasy.txt

##### head -n5 oeasy.txt  只看1首

##### 如果想加上行号怎么办

###### Cat -n oeasy.txt

###### 得到这个之后再给pip

###### Cat -n oeasy.txt|head -n5

#### 观察尾部tail

##### 先找到本地的png文件

###### cat pic.png love.txt >pic.img然后融合

##### 首先看看这是个什么文件

##### 是否能在

##### 看看头 看看尾巴

##### tail pic2.png

##### 比较两种不同

###### Cat o.txt |cat -n|tail -5

###### Cat o.txt |tail -5|cat -n

#### 字数统计wc

##### echo “hello”

##### echo “hello”|wc

##### echo “hello world”|wc

##### wc oeasy.txt

#### 标准输出

##### 输出到火狐

#### Cp

##### Pwd

##### cp story.txt poems.txt

##### ls

##### ls \*.txt

##### ls ?????.txt

##### ls p\*.txt

##### mv poems.txt poem.txt

##### ls ????.txt

##### ls ?????.txt

##### ls p\*.txt

##### ls

##### mv story.txt .

##### cp story.txt spring

##### ls spring

##### cp \*.txt summer

##### mv poems.txt autumn

##### mv story.txt winter

##### Cp . ../2/

##### Cp ../3/

##### Tree ../

##### Touch ../../summer/4/{1-30}.txt

##### Touch ../../summer/5/{1-30}.txt

##### Cp ../../summer/5/ ../../sumer/6

##### Tree ../../summer/

#### Rm文件练习

##### 删除单个文件

##### 删除指定文件

###### 新建10个文件同名,但扩展名不同,分别为jpg和cr3,

###### 只删除cr3文件

###### 删除ls

##### rm -I interactive 提示

##### whereis ls删除

###### 看他还好使吗?

##### -r

###### Rm -rf ../../autumn ../../winter

###### Cp ../../spring ../../aumtum

###### Tree ../../autumn

###### Cp ../../spring ../../winter

###### Tree winter

##### 把~和/删空

###### rm -r \*

#### 隐藏文件

##### 有的时候有些文件就不会出现在ls列表下,为什么呢因为他是隐藏文件

##### 如何建立隐藏文件开头是点

##### 建立隐藏文件夹 里面再放隐藏文件

##### 如何查看隐藏文件

##### 如何删除隐藏文件

##### 把普通文件改成隐藏文件

##### 把普通文件夹改成隐藏文件夹

##### .和..可以删除吗?

###### 试试

##### 哪里没有.和..

###### Root那么可以新建.或者..文件夹吗

###### 试试

#### 粉碎文件shred

#### 文件综合练习

##### mv ../../../year ../../../year2020

##### cp ../../../year2020 ../../../year2021

##### rm ../../../year2021/spring/2/29.txt

##### cp ../../../year2021 ../../../year2022

#### 疯狂思考

##### 如何移动ls

##### 移动了又会发生生么

##### mv /bin/ls /root（

##### 还能用吗?

##### 可以用绝对路径使用吗?

##### 为什么这个东西不在这个地方还是能够显示呢?

#### 链接命令ln

##### 软连接-s

###### 什么是软连接

首先源文件oeasy

然后建立软连接

然后ls观察

通过连接进行修改,观察源

删除源文件,再观察,

链接也就断了,很软,指针指向文件名的url

新建源文件,在观察

如果改名呢

###### 为什么要有软连接

如果不用软连接,用cp就会出现很多副本

编辑了一个之后其他的并不会修改

软连接就像快捷方式非常快速

在/etc/init.d目录下创建tomcat服务软连接

ln -s /usr/lib/tomcat8.5.31/bin/catalina.sh /etc/init.d/tomcat

通过ls 观察第二列中的连接数量.

Cowsay和cowthink是链接

##### 硬连接

###### 什么是硬连接

首先源文件oeasy

然后建立软连接

然后ls观察

通过连接进行修改,观察源

删除源文件,再观察

链接没有断开,还是很硬的指向具体存储文件信息的磁盘空间

新建源文件,在观察

指向相应的文件对应的存储地址地址

###### 为什么要有硬连接

软件连接端掉原来的文件名之后,文件就消失了

硬连接直接指向存储文件的磁盘位置

不管原来的文件名如何变化

总是直接指过去

始终连接着

###### 比较软硬连接

软连接连接的是相对路径

硬连接连接的是直接的磁盘数据inode

系统中所有的文件名字是软连接还是硬连接?

###### 道是一个文件夹 里面有阴 阳是对于阴的一个硬连接 三在阳里面 三也在阴里面

#### Find命令

##### 按照name查找

##### 按照type查找

##### 按照深度查找 depth

##### 按照文件名查找 快速touch出12个月每个月31天,然后find加删除

##### 并且删除 并且打开播放列表 打开字幕文件

##### 按照时间ls

##### 按照时间查找

### linux文件夹

#### 文件夹路径

###### 为什么要有文件夹呢?

###### 我们假设,只有一个文件夹,所有的文件都堆在一起,那怎么解决重名的问题

###### 怎么样才能找到我想要找的文件呢?

###### 我的世界在一个文件夹

###### Tomcat在一个文件夹 .

###### Tomcat里面都是tomcat相关的这些文件 .

###### 分门别类 才方便

###### 那我们如何找到文件呢?

###### 每个文件都有名字,知道他的名字就可以找到他.

###### 文件具体在哪呢,在硬盘的某个文件里面,

###### 你的云主机都有硬盘容量,那是总量.

###### 他被划分成几块,每块儿都要在切分,我们的文件就存在某个小块里面.

###### 如何进入desktop

###### Cd desktop

###### 桌面有什么

###### Ls

###### Ls -lah观察 . 和..是什么意思

##### 绝对路径/代表根目录根

###### Pwd

###### 他在哪儿呢,他是一层层进来的.

###### 最根本的是什么,是根目录/

###### Cd /

###### 我们去根目录看看

###### Ls 根目录里面有多少子目录?

###### 这里面就有父子关系,比如乾隆是雍正的儿子,雍正是康熙的儿子.

###### 今天的话叫y染色体,那要是网上追根溯源 到哪儿呢?

###### 清太祖,努尔哈赤,在往上还有没有,还有,但是网上追没头,得有个定义.谁给定义的呢?

[皇太极](https://baike.baidu.com/item/%E7%9A%87%E5%A4%AA%E6%9E%81/179283)在位时规定以[清显祖](https://baike.baidu.com/item/%E6%B8%85%E6%98%BE%E7%A5%96)塔克世的直系子孙为“宗室”，其余伯叔兄弟旁支子孙称“[觉罗](https://baike.baidu.com/item/%E8%A7%89%E7%BD%97/10775332)”。宗室系黄带子，觉罗系[红带子](https://baike.baidu.com/item/%E7%BA%A2%E5%B8%A6%E5%AD%90)，由此以示区别。

这一代代的就像一棵树一样,开枝散叶,但是根儿在哪?

根儿在塔可世,这就是根结点,这就是根,这就是/,这就是root.

认了这个祖就归了宗,属于宗人府来管理了,所有的传宗接代都不是生物行为了,都有一套等级秩序,长幼嫡庶非常复杂.

这里面还有兄弟关系,比如皇太极和多尔衮就是兄弟,他们都是努尔哈赤的儿子.

###### 那我们来看看 我们的/有什么儿子

###### Cd /home

###### Cd /home/shiyanlou

###### Cd /home/shiyanlou/desktop

###### Ls /etc/apt

###### Cat /etc/apt/sources.list

###### Cd /找到根目录所有的文件夹都是从这里排生的

###### 回到用户目录 cd ~

###### tilde 美 [ˈtɪldə]

###### 或者更简单 cd 回车

##### home

###### cd $HOME

###### cd $

###### cd

##### 相对路径

###### . 就是当前目录也叫做工作目录

Ls .

Cat ./fxtic.desktop

注意这个 路径其实是cd的参数 所以一定要加上空格

Nautilus .

Cd .

###### 向上 cd..

.. 就是上一级目录

Ls ..

Cat ../package.json

Nautilus ..

Cd ..

Cd desktop

Cd –

总结. .. - ~ /

Cd 直接回到home文件夹

##### 什么是unix

###### Ls从什么时候开始有的?

###### Mac的terminal也可以支持ls

###### Mac的本质是bsd

###### 观察到来自于unix

##### 

#### 文件夹操作

##### linux新建文件夹mkdir

##### 批量追加字幕效果开头 在结尾

##### cat 悯农.txt 静夜思.txt >古诗两首

##### Man mkdir发现也是david写的做yes的那个哥们

新建一个uni¢𐍈d€文件夹

ʎsɐǝට倒着写文字

##### mkdir $(echo 'uni¢\xF0\x90\x8D\x88d€3')

##### mkdir $(echo 'uni¢\xF0\x90\x8D\x88d€3\\1F601')

##### https://www.wfublog.com/2015/06/unicode-emoji-special-character-table.html

##### 各种unicode字符

##### 建立一个☯️文件夹,里面有阴阳两个字文件夹

##### https://dev.to/thatjoemoore/emojifying-my-bash-prompt

##### 😋

##### mkdir 😋

##### j

打开这个网站：https://unicode.org/emoji/charts/full-emoji-list.html，然后以第一个表情为例，码表为 U+1F600，那么终端输入 echo "\U1F600" 就可以输出这个表情了

-e这个参数。linux上确实是需要的

###### 到桌面

###### 新建一个文件夹叫做year

###### 进入year

建立一个道文件夹

里面分成阴阳

Ls -R

###### 为什么一般文件夹中没有空格

###### 你可以建立一个文件夹带空格的

###### 如果有了文件夹还可以touch出同名文件吗?

###### 如果有了文件夹,还可以mkdir出同名文件夹么

###### Mkdir oeasy tutorial

###### Mkdir “oeasy tutorial”

###### Mkdir oeasy\ tutorial

###### 这个时候想要进入目录的时候需要加上双引号

###### 在year文件夹下面建立一个spring文件夹

###### 再建立文件夹分别是summer

###### 一月里面建立「1-3」

###### Summer里面「4-6」

###### 使用-p parent man

###### Autumn「7-9」

###### Winter「10-12」

##### Tree

###### Ls -R

###### 使用

###### Man tree

###### Tree -a

###### ls -R 递归便利文件夹

##### rm删除文件夹

###### Rm summer

###### 删除文件夹autumn winter

###### Ls在哪?如果直接把/bin删除了会如何

##### cp复制文件夹

###### Ls 观察现在还有几个文件夹

###### 先进入year/spring

###### Cp spring spring2..观察第二春

###### Cp spring ../spring\_backup

###### 观察 然后删除

###### Cp spring summer

###### Man cp

###### Cp -r spring summer

###### Cp -r spring autumn

###### Cp -r summer winter

###### 如果复制bin到bin2

###### 还能执行么?

##### 移动文件夹

mkdir {2007..2009}-{01..12}

###### 新建文件夹并且放入相关文件夹内

###### Mv ../spring \_backup ./year/

###### Mv ../spring ./year/spring\_backup2

###### Ls .

###### 如果移动bin,还能执行ls吗

##### 改名

###### Rm summer

###### Mv spring\_backup summer

###### 如果bin改名还能执行吗?

##### 连带文件夹递归删除

###### Man rm

###### Rm -rf /

###### 传说中的删库跑路

##### 综合练习

###### 建立一年四季

###### 建立一季三个月

###### Mkdir {10-12}

##### 目录跳转

###### 进入7月

###### 从7月进入autumn

###### 从autumn进入winter

###### 从winter进入1月

###### 从1月进入4月

### 压缩与解压

#### tar

##### 参考https://www.shiyanlou.com/courses/1/learning/?id=62

##### man tar

##### shift+g 到最底下找example

##### cvf caf

##### 压缩成不同格式tar。Tar .bz 共三种然后ls比较大小

##### 三个文件夹解压

###### 第一个,进入文件夹unpack1,然后解压

###### 第二个用-C指定目录unpack2解压

###### 第三个用相对目录建立文件夹 ,然后解压 观察

##### 字符串水族馆

* 安装依赖包

root@jacktian-virtual-machine:~# sudo apt-get install libcurses-perl  
root@jacktian-virtual-machine:~# cd /tmp  
root@jacktian-virtual-machine:/tmp# wget http://search.cpan.org/CPAN/authors/id/K/KB/KBAUCOM/Term-Animation-2.4.tar.gz  
root@jacktian-virtual-machine:/tmp# tar -zxvf Term-Animation-2.4.tar.gz  
root@jacktian-virtual-machine:/tmp# cd Term-Animation-2.4/  
root@jacktian-virtual-machine:/tmp/Term-Animation-2.4# sudo perl Makefile.PL && make && make test  
root@jacktian-virtual-machine:/tmp/Term-Animation-2.4# sudo make install

#### zip

#### gzip gunzip 可以看到ls对应的帮助文件

#### rar

##### Man搜索apt-get

###### 使用/对与install进行搜索

###### 观察到好像可以两个包一起装

##### rar unrar

## 用户

### 用户管理

#### 用户查询who

##### who

##### who -H headings

##### who -q

##### who -Hq

##### 新打开一个terminal

##### who -Hq

##### who am I

##### whoami

##### which whoami

##### man who

###### who /var/run/utmp

###### who /var/log/wtmp|sort|uniq

#### 用户管理

##### useradd 灵魂三问

#### 随机生成身份rig:

#### sudo

##### sudo -k

#### su

##### root

##### root文件夹 /root

##### 使用root用户登录之后,用户名会变更而且$会变成#

#### 修改密码passwd

##### https://blog.csdn.net/fanbaodan/article/details/88431957

#### 多用户的概念起源于unix

##### Unix的历史

#### chown 改变所有者

##### Owner的概念

##### 从oeasy换到root

##### 尝试写权限

##### 再从

#### Find按照所有者查找

### 用户组管理

#### chgrp改变文件的所有组

### ssh远程登录

#### ssh登陆

#### 开启ssh登陆服务

Ssh -v 查看详细信息

#### screen

##### https://blog.csdn.net/F8qG7f9YD02Pe/article/details/79694995?utm\_medium=distribute.pc\_aggpage\_search\_result.none-task-blog-2~all~first\_rank\_v2~rank\_v25-9-79694995.nonecase

#### scp

#### 使用用户名远程登录bandit

## 权限管理

### 权限设置

#### 查看权限stat

##### stat 1.txt

#### 修改文件权限

#### 什么是可读

#### 什么是可写

#### 什么是可执行

##### 下载源代码

###### Apt source …

###### Apt build-dep

###### Make

##### 脚本的解释器

什么是指令集?

CPU执行计算任务时都需要遵从一定的规范，程序在被执行前都需要先翻译为CPU可以理解的语言。这种规范或语言就是指令集（ISA，Instruction Set Architecture）。程序被按照某种指令集的规范翻译为CPU可识别的底层代码的过程叫做编译（compile）。x86、ARM v8、MIPS都是指令集的代号。指令集可以被扩展，如x86增加64位支持就有了x86-64。厂商开发兼容某种指令集的CPU需要指令集专利持有者授权，典型例子如Intel授权AMD，使后者可以开发兼容x86指令集的CPU。

 处理器架构就是处理器的硬件架构，称为微架构。是一堆硬件电路，去实现指令集所规定的操作运算。

 是的，指令集决定了处理器的架构，因为处理器架构就是用硬件电路实现指令集。但是具体用什么样的处理器架构，设计怎样的硬件电路，每个人设计的都可以不一样。

4大芯片架构arm，与X86、MIPS、RiSC-V

##### 这里插入c语言历史,c为什么是unix成功可移植的原因.ppt150,

##### 下载clear.c的源文件,见whereis

##### https://www.multicians.org/

##### ken和litche最早在做multics时候,使用的是PL/1这样的语言 他们知道高级语言对于操作系统移植性的好处

##### ls -F可以查看可执行带\* 文件夹带/

1984年，史托曼开始[GNU](http://www.gnu.org/)计划， 这个计划的目的是：建立一个自由、开放的Unix操作系统(Free Unix)。 但是建立一个操作系统谈何容易啊！而且在当时的GNU是仅有自己一个人单打独斗的史托曼～ 这实在太麻烦，但又不想放弃这个计划，那可怎么办啊？  
  
聪明的史托曼干脆反其道而行～『既然操作系统太复杂，我就先写可以在Unix上面运行的小程序，这总可以了吧？』在这个想法上， 史托曼开始参考Unix上面现有的软件，并依据这些软件的作用开发出功能相同的软件，且开发期间史托曼绝不看其他软件的原始码， 以避免吃上官司。后来一堆人知道免费的GNU软件，并且实际使用后发现与原有的专利软件也差不了太多，于是便转而使用GNU软件， 于是GNU计划逐渐打开知名度。

Gnu的工具链一直是在unix上运行的,最早的linux的x86内核也是在unix上用gnu的工具链gcc在unix上编译的,然后在unix下把内核放到磁盘上,然后从磁盘启动接管内存cpu等资源.然后再把gnu的工具链编译之后也放到磁盘上,gnu的x86的工具链是ok的,然后再编译系统中其他的东西,比如ls之类的.最后可以在x86的机器上,进入linux系统,然后用gnu的工具链编译各种工具,甚至编译内核.gnu/linux最早是寄生在unix上的.  
  
虽然GNU计划渐渐打开知名度，但是能见度还是不够。这时史托曼又想：不论是什么软件， 都得要进行编译成为二进制文件(binary program)后才能够执行，如果能够写出一个不错的编译程序，那不就是大家都需要的软件了吗？ 因此他便开始撰写C语言的编译程序，那就是现在相当有名的GNU C Compiler(gcc)！ 这个点相当的重要！这是因为C语言编译程序版本众多，但都是专利软件， 如果他写的C编译程序够棒，效能够佳，那么将会大大的让GNU计划出现在众人眼前！如果忘记啥是编译程序， 请回到第零章去瞧瞧编译程序吧！

 GNU的通用公共许可证：  
  
到了1985年，为了避免GNU所开发的自由软件被其他人所利用而成为专利软件， 所以他与律师草拟了有名的通用公共许可证(General Public License, GPL)， 并且称呼他为copyleft(相对于专利软件的copyright！)。 关于GPL的相关内容我们在下一个小节继续谈论，在这里，必须要说明的是， 由于有GNU所开发的几个重要软件，如：

* Emacs
* GNU C (GCC)
* GNU C Library (glibc)
* Bash shell

##### 可移植portable,意思是可以移动的,可以带着去port港口的,对应可以从一种cpup架构到另一种;posix

;而对于我们就变成了可移植指将植物移动到其他地点种植，后引申为将生命体或生命体的部分转移的器官*移植*，例如转移一个人的肝脏代替另一个人的肝脏，这叫做肝*移植*。

##### 汇编语言什么样子 造成后来很多的软件开发者可以藉由这些基础的工具来进行程序开发！ 进一步壮大了自由软件团体！这是很重要的！不过，对于GNU的最初构想 『建立一个自由的Unix操作系统』来说，有这些优秀的程序是仍无法满足， 因为，当下并没有『自由的Unix核心』存在...所以这些软件仍只能在那些有专利的 Unix平台上工作～～一直到Linux的出现...更多的FSF开发的软件可以参考如下网页：

* <https://www.fsf.org/resources>
* 1.一段简单的C语言代码：
* #include<stdio.h>  
  int main()  
  {  
  printf("hello world/n");  
  return 0;  
  }  
  2.root@xyf-laptop:~/桌面# gcc -O2 -S 1.c  
  得到对应的汇编代码：  
      .file    "1.c"  
      .section    .rodata.str1.1,"aMS",@progbits,1  
  .LC0:  
      .string    "hello world"  
      .text  
      .p2align 4,,15  
  .globl main  
      .type    main, @function  
  main:  
      leal    4(%esp), %ecx  
      andl    $-16, %esp  
      pushl    -4(%ecx)  
      pushl    %ebp  
      movl    %esp, %ebp  
      pushl    %ecx  
      subl    $4, %esp  
      movl    $.LC0, (%esp)  
      call    puts  
      addl    $4, %esp  
      xorl    %eax, %eax  
      popl    %ecx  
      popl    %ebp  
      leal    -4(%ecx), %esp  
      ret  
      .size    main, .-main  
      .ident    "GCC: (Ubuntu 4.3.3-5ubuntu4) 4.3.3"  
      .section    .note.GNU-stack,"",@progbits  
  3.生成目标代码文件1.o：root@xyf-laptop:~/桌面# gcc -O2 -c 1.c  
  4.对文件1.o进行反汇编：root@xyf-laptop:~/桌面# objdump -d 1.o  
  得到：  
  1.o:     file format elf32-i386  
  Disassembly of section .text:  
  00000000 <main>:  
     0:    8d 4c 24 04                    lea    0x4(%esp),%ecx  
     4:    83 e4 f0                         and    $0xfffffff0,%esp  
     7:    ff 71 fc                           pushl  -0x4(%ecx)  
     a:    55                                  push   %ebp  
     b:    89 e5                             mov    %esp,%ebp  
     d:    51                                  push   %ecx  
     e:    83 ec 04                        sub    $0x4,%esp  
    11:    c7 04 24 00 00 00 00   movl   $0x0,(%esp)  
    18:    e8 fc ff ff ff                  call   19 <main+0x19>  
    1d:    83 c4 04                      add    $0x4,%esp  
    20:    31 c0                          xor    %eax,%eax  
    22:    59                               pop    %ecx  
    23:    5d                               pop    %ebp  
    24:    8d 61 fc                      lea    -0x4(%ecx),%esp  
    27:    c3                               ret    
    
  数得目标代码为40字节；  
  4.生成可执行文件：gcc  -o 1 1.c  
  5.启动GDB：  
  root@xyf-laptop:~/桌面# gdb  1  
  GNU gdb 6.8-debian  
  Copyright (C) 2008 Free Software Foundation, Inc.  
  License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
  This is free software: you are free to change and redistribute it.  
  There is NO WARRANTY, to the extent permitted by law.  Type "show copying"  
  and "show warranty" for details.  
  This GDB was configured as "i486-linux-gnu".  
  (gdb)   
  6.生成二进制代码：  
  (gdb) x/40xb main  
  0x80483c4 <main>:    0x8d    0x4c    0x24    0x04    0x83    0xe4    0xf0    0xff  
  0x80483cc <main+8>:    0x71    0xfc    0x55    0x89    0xe5    0x51    0x83    0xec  
  0x80483d4 <main+16>:    0x04    0xc7    0x04    0x24    0xb0    0x84    0x04    0x08  
  0x80483dc <main+24>:    0xe8    0x13    0xff    0xff    0xff    0xb8    0x00    0x00  
  0x80483e4 <main+32>:    0x00    0x00    0x83    0xc4    0x04    0x59    0x5d    0x8d  
  (gdb)   
  7.GDB命令手册  
  关于断点的命令：  
  awatch:用来为一个表达式设置观察点，在表达式的值发生改变时，或者当表达式的值被读取的时候，程序停止执行。  
  break:用于在程序中设置断点 break + 要设置断点的行号  
  clear:用于清除断点 clear + 要清除断点的行号  
  commands:用于为遇到断点之后执行特定的指令而设置的 command + 断点号；如果commands 之后没有断点号，gdb会将最后一个设置的断点编号作为默认值。  
  condition:在满足一定的条件时才在指定的行上设置断点 condition 断点编号 条件表达式   
  delete:用于清除断点和自动显示的表达式的命令。与clear的不同之处：clear要给出断点的行号，delete要给出断点的编号。用clear命令清除断点时gdb会给出提示，而用delete清除断点时gdb不会给出任何提示  
  disable:让所设断点暂时失效。如果要让多个编号处的断点失效可将编号之间用空格隔开  
  enable:与disable 相对  
  ignore:这条命令在一定范围内忽略用户设定的断点 ignore N CONT  
  tbreak:用于设置临时断点，所谓临时断点就是断点只在设置之后起作用一次。  
  watch:与awatch类似  
    
  关于数据的命令：  
  display:该命令用于显示表达式的值，使用了该命令后，每当程序运行到断点处都会显示表达式的值。 display 表达式 如:display 5\*j  
  info display:用于显示当前所有要显示值的表达式的有关情况  
  delete display:用于删除一个要显示值的表达式，调用这个命令删除一个表达式后，被删除的表达式将不被显示。 delete display 1  
  disable display:使一个要显示的表达式暂时无效。可有enable重新使之有效  
  enable display:反操作disable diplay  
  undisplay:用于结束某个表达式值的显示。同delete display功能相似  
  whatis:显示某个表达式的数据类型 如:whatis j  
  print:用于打印表达式的值。可以打印内存中从某个变量开始的一段区域的内容。$表示给定序号的前一个序号，
* 表示给定序号的向前第2个序号。例如给定序号是5，那么$代表4,
* 代表3.如果是print $表示 print 1  
  print还可用于对变量赋值如print j=8;  
  print打印内存中从某个部分开始的一块连续空间的内容。如print 开始表达式@要打印的连续空间的大小  
  ptype:用于给出类型的定义。疑问ptype与whatis的区别？  
  set:set variable 用于为变量赋值 类似print set variable j=9 print j=9  
    
  关于文件的命令  
  add-shared-symbol-files:该命令用于从动态的连接映射的共享目标文件中装入符号表。  
  add-symbol-file:该命令用于从已经动态装入的文件中装入负号表。使用格式是：add-symbol-file FILE ADDR其中FILE给出的是动态装入的文件的名称，ADDR是文件中正文的起始地址。  
  cd：改变当前工作目录。  
  core-file:使某个文件成为core dump  
  directory:用来向源文件搜索目录中添加一个目录  
  file:用来装入待调试程序的命令。  
  list:list的非空系数的几种形式：  
  LINENUM：当前文件的LINENUM行  
  FILE：LINENUM： 指定文件的LINENUM行  
  FUNCTION：当前文件的FUNCTION函数  
  FILE：FUNCTION  
  \*ADDRESS列出包含该地址的文件  
  list 起始行号 截止行号  
  forward:用来从列表当前行开始向后查找第一个匹配某个字符串的程序行 forward 匹配的字符串  
  load:该命令用于动态的向正在调试的程序中装入文件，并记录它的符号表，准备联接  
  path:该命令用于向目标文件的搜索路径中增加目录的，可以使用当前目录的$cwd  
  pwd:显示当前工作路径  
  reverse-search:从列表当前行开始向前查找第一个匹配的字符串，查找到的行号保存在$\_中  
  search:同forward  
    
  关于程序运行的命令：  
  cont:使程序继续执行  
  handle:对信号设置处理函数。如：handle 信号 信号处理  
  信号可以是符号，也可以是数字。如果用all来作参数，则意味着调试器接受除了（信号SIGTRAP和信号SIGINT）之外的所有信号  
    
  处理函数：  
  stop:如果信号发生重新进入调试器，并打印出提示信息；  
  print：如果信号发生就打印一条提示信息  
  pass:让程序能看到这个信号发生  
  jump:指定程序开始调试的指令和地址 jump 行号或是指令地址  
  kill:结束当前程序的调试。  
  next:继续程序的运行，越过子程序调用  
  nexti:用于单步执行一条语句中的一条指令，它会跟踪到子程序的内部，next用于单步执行一条语句，而nexti用于单步执行一条指令，一条语句可能有几条指令构成。  
  step:用于执行一条语句的，也不越过子过程的调用  
  stepi:用于执行一条指令的，它不越过子程序的调用，而是跟踪到子过程的内部  
    
  关于堆栈的命令：  
  backtrace:用来打印栈侦指针的。backtrace 要打印的栈侦指针的个数  
  frame: 该命令用于打印栈侦的。frame 要打印的编号  
  select-frame:指定要选择的栈侦的编号。一般与info frame连用。  
  info reg：查看寄存器使用情况。  
  info stack：查看栈。  
  where：查看当前位置。  
  up：跳到上一层函数。  
  down：与up相对。  
  attach <进程号>：把当前高度的程序粘在某个进程上，用于调试运行中的进程。  
  ptype <变量or数据类型>：查看变量或数据类型的定义。  
    
  注：参考《深入理解计算机系统》以及http://read.newbooks.com.cn/info/35369.html

### 权限修改

#### chmod1先设置user

##### 八进制形式 Chmod 777 700 644 1.txt

##### Ls一下观察

##### 然后设置一下

##### 再ls观察

##### 尝试执行

##### 再设置尝试写

##### 再设置尝试读

##### 翻着回来

##### 设置好读,然后读

##### 设置好写,然后写

##### 设置好执行,然后执行.

#### chmod2设置group

##### ls先观察组

##### 禁止other和user的任何操作

##### 把group的数值设置

##### 然后观察变化

##### 设置回来在观察

#### Ls观察文件所在的组,然后设置

##### 符号化形式 Chmod a=r,a-rwx,g+w,o-x 1.txt

##### A代表所有=user | group|other

##### 777. a+rwx rwxrwxrwx

##### 755. u+rwx ,g=rx,o=rx rwxr-wr-w

##### 644. u=rw,g=r,o=r. rw-r--r--

##### 来个表格

#### 可以看到文件的owner

#### find按照权限查找

### ftp服务器实战

#### 开启一个ftp服务器

#### chroot控制用户范围

## vim 中级用法

### 控制数字

#### System.out.println(“[debug]1=====”);

##### yyp

##### 1<ctrl+a> 就变成了debug2

#### .blog{margin:60px;}

##### yyp

##### ^lcwnews esc

##### 30<ctrl-a>

### reg的控制

光标移动到结束行，ma

光标移动到起始行,输入y'a

光标移动到需要复制的行，输入p,行前复制则输入大写P

#### :argdo %s/<\/code>check\\_.\{-}<\/code>//gce |update

#### \\_.\*可以包括回车符

#### \{-}表示非贪婪匹配

#### :%s/\[\(.\{-}\)\](\(.\{-}\))/<a href=\"\2\">\1<\/a>/c

#### s是替换一行

#### %s是全部替换

### 宏

 qq 开始录制名称为 q 的宏

 q 结束宏录制

 @q 播放名称为 q 的宏

### 格式设置

 :e ++ff=dos 强制使用dos格式重新打开当前文件（解决^M问题）

 :set ff=unix 将当前待保存内容设定为 unix 格式

 :e ++enc=gbk 或者 :e ++enc=utf-8 强制使用指定编码集重新打开当前文件。

### 自动保存撤销步骤

#### 理解缓冲buffer

##### 文件打开过程

###### Vi oeasy.txt

###### 目前oeasy.txt已经存在,把文件复制出一个副本.然后把这个副本放入vi工作区,这个工作区叫做工作缓冲区buffer.

###### 打开文件之后,首先显示的是文件开头的23行,在移动光标的过程中,vi可以显示当前工作缓冲区的其他部分,如图B

###### 图P76Fig4.8

##### 文件保存的过程

###### W命令,保存文件.

###### 把缓冲区里面的缓冲文件(拷贝出来的副本)存储为原文件

###### 然后继续在缓冲区编辑

##### 多vi 开同一文件

set swapfile

###### 观察隐藏文件

###### 设置专门文件夹

###### set directory=$HOME/.vim/swap

set noswapfile

##### 打开多文件

###### 打开一个网页

###### 可以再打开一个css

###### Open ./sdf/sdf

###### 或者open .

###### 打开当前文件夹进行浏览

###### 打开之后又一个新的缓冲buffer建立了

##### 缓冲切换

###### Ls

###### B2

###### Help buffer

##### 全部保存并推出

###### 修改两个文件

###### ZZ保存所有的buffer并且退出.

###### Help ZZ

##### 自动保存autosave

#### 撤销u与重做Ctrl+r

##### set undofile

###### 新建文件夹

###### 可以找到ls -a

###### 可以找到相关文件

##### 专门的撤销保存文件夹

###### mkdir ~/.vim/undodir

###### chmod 777 ~/.vim/undodir

###### 在~/.vimrc中添加

Set undofile

Set undodir=～/.vim/undodir

set noundofile

#### 设置备份文件

set backup

###### 设置备份文件文件夹

set backupdir=$Vim\backup

**set nobackup**

##### 修改文件3

###### 如果用两个terminal同时打开同一个文件会如何？

###### 提示

##### 总结

###### 打开过程是把当前的文件拷贝成一个缓冲buffer，然后在buffer中进行修改

### g全局命令

g/pattern/cmd  
这个g命令式在文中查找pattern，然后对找到的这些行执行cmd命令；注意：cmd应用的范围是找到行  
cmd命令有很多，比如d，删除命令；c，删除并且进入insert模式；s，替换命令，这个用的多，例如g/test/s/aaa/bbb/ 在文中查找包含test的行，并且把该行中的aaa替换成bbb,也可以把aaa省略，  
g/test/s//bbb/,这时候表示的是把test替换成bbb指令

#### https://blog.csdn.net/huangkangying/article/details/50506640

#### g~

### 窗口控制

#### https://coolshell.cn/articles/11312.html

## 提示符prompt

### Shell管理

#### https://www.freebsd.org/ports/shells.html

#### https://coolshell.cn/articles/8619.html

#### 什么是shell

##### 查看自己所用的shell

###### Echo $0

echo $SHELL

###### 当前的这个sell在什么位置

一个系统有内核(Kernel)自然就有外壳(Shell).

这里需要了解一下 *终端(terminal)*和 shell 的区别：  
在命令行中，shell 提供了访问操作系统内核功能的途径，比如说我们所熟悉的 bash、zsh，都是不同的 shell；而终端则为 shell 提供视觉界面（窗口），比如我们所熟悉的 iTerm2、Linux 桌面上的终端工具等。甚至于我们在 VSCode 中所使用的命令行，也是某种意义上的终端。

什么是终端?汽车可能就是一个终端,手机也是一个终端,电视、空调、都需要连接服务器端都是终端,甚至马桶也可能是一个终端.

终端可以有界面,比如gui cli,也可能没有,就是看见你是个男的,小便他就翻两层盖子,然后对你的尿液进行检验.  
我们在 Windows 下所使用的 CMD、Powershell 既然是一个终端，也是一个 Shell，还是同名的脚本系统。

Terminal登陆之后,就是加载硬盘上的shell到内存,然后就是一直在shell程序运行的状态,或者通过shell运行硬盘里的程序,最后退出的也是这个shell

##### 我系统中都有什么shell?

##### cat /etc/shells

##### 如何切换shell

##### 当我换了之后只对当前登陆有效 登出之后,还是默认的shell

#### 还有那些shell?

##### Bourne shell

##### 最早Ken Thompson设计Unix的时候，使用的是命令解释器，命令解释器接受用户的命令，然后解释他们并执行。

早年间的计算机是由ibm垄断的,操作非常复杂,是用打孔卡把计算机指令一条一条的输入进去,然后进行执行的,如果写错了,要重新从头同打孔.执行的时候也是每一条指令挨着执行,一路走下去.数据也是通过打孔卡进行录入的.非常麻烦. 当时IBM计算机生产的目的是不同的结构，目的是具体的，这些机器没有兼容性，机器之间不能通信，甚至软件没有连接，因为这样可以保证IBM能够在各个方面保持领先地位。

一位大哥出现了,肯奥尔森,1926年出生在康涅狄格州的布里奇波特,他的父亲是挪威移民,一名没有大学学历的机械工程师.有三个儿子,都成为了工程师.

中学毕业之后,奥尔森加入了海军,当时二战已经接近尾声,他在海军的军旅生活中学习了雷达广播声呐导航等技术.后来加入mit的林肯实验室,工作.他的领导要求他做一台小型的计算机,他带着15个工程师9个月完成了任务.当时的空军自动防御系统是ibm和林科实验室合作承包的,他被派去和ibm进行联络工作.这个过程中他充分接触了ibm的官僚主义作风.ibm把计算机锁在清洁的玻璃房子里,专业人员穿着白大褂去操作,去跟专业人员交流的感觉就像通过ibm的人去接受神谕一样.奥尔森想要做的是,把ibm的大型机从小范围解放出来,人人都可以操作计算机,使用计算机,用非常简单的方式.

1957年他和一个伙伴打算创业,当时正是朝鲜战争,不适合创业,不过他有真正的市场需要的技术,第一年就盈利了.他做的就是pdp程控数据处理机,用这个名字就是要掩人耳目,不被ibm注意到. ，从公司名称到产品命名都避免与“计算机”字样沾边，这样可使像IBM之类的巨头放松紧惕。

有大量的可以被机器替代的工作,比如机线员,当时打电话没有交换机路由器,都是靠接线员.国际电话电报公司来了一笔订单,15个pdp-1,用于把接线员换成程控交换机.ibm根本看不上这个市场,小型机这个领域dec就这样开拓出来了.

1964年低价的2.7万美元的pdp5大获成功,卖出了1000多台,.dec还发布了他的磁存储专利. 在八十年代,它是仅次于IBA得业内第二号巨头,最辉煌时期公司有员工12万人,年销售收入达到130亿美元,纯利润11亿美元,在《财富》杂志评选得世界500强企业排行榜上名列第 30位。

此时DEC已无法避免与IBM正面交锋，1976年春IBM推出了它的小型机Seriesl开始进攻被DEC垄断的市场。而DEC似乎已有准备，1975年DEC大将戈登•贝尔策划了令IBM近乎绝望的“VAX战略”，让DEC公司的所有小型机组合成局域网，之后的新产物既发挥了小型机的灵活便利，又不亚于大型机的计算能力，这一战略让DEC一举登上了仅次于IBM的上游巨头第二把交椅。

正是有了这个漏洞，奥尔森利用了VAX战略，第一款VAX 8600与其他软件兼容的DEC的年销售额开始成倍增长，在不到一年的时间里就进入了美国的前100名。与此同时，IBM也陷入了自己的困境，并开始了一场快速的衰退。到20世纪80年代末，DEC已经能够与IBM发出的价值20亿美元的订单相匹配，仅VAX一家。

DEC小型机还培育和影响了科技产业中的未来明星。微软联合创始人比尔·盖茨和保罗·艾伦(Paul Allen)使用PDP-10开发了首款BASIC编程语言。曾为DEC开发多款操作系统的戴夫·卡特勒(Dave Cutler)则为微软开发了Windows NT和Azure操作系统。

1977年pc开始发站起来,奥尔森曾在1977年预言：“没有理由让所有人在家里保有一台计算机。奥尔森拒绝了当时的苹果公司合作的请求

人们需要更简单、更便宜的桌面电脑，DEC却仍在面向技术市场生产复杂设备。”

DEC前高级副总裁文·亨德尔(Win Hindle)说：“奥尔森创立的公司鼓励创新，奖励有好创意的人。”此外，奥尔森痛恨裁员，即使在公司面临困境时，也坚持不让任何一个员工下岗。

但到了1990年代，DEC开始走下坡路。该公司未能意识到个人PC和企业工作站的价值，也不能接受从专利技术到开放系统的市场变革，包括使用英特尔微处理芯片的PC和搭载Unix操作系统的通用服务器。

然而，IBM进行了反击，直接开放了x86体系结构，并且随着计算机技术的发展，发布了越来越多的新体系结构，而DEC的开发周期太长了。创始人的优柔寡断和错失与苹果做生意的机会，除其他外，导致DEC的巨额预算赤字，最终于1998年被康柏(Compaq)收购(后来被惠普(HP)收购)。DEC带动了整个计算机的快速发展，但最终在融资上失败了，被称为最不幸的科技公司。

　　最后，值得一提的是，为了应对财务问题，DEC将廉价生产硬盘驱动器的存储单元卖给了Quentin(最终卖给了Seagate)，并以低价将其数据库软件开发部门出售给了甲骨文(Oracle)。后来，他们以低价将互联网产品部门卖给了美国互联网产品公司。收购DEC后，一些计算机开发技术人员进入AMD，这些部门或技术人才直接或间接地使这些公司成为了今天的五大巨头。

https://en.wikipedia.org/wiki/Ken\_Olsen

##### 后来出现了Bourne shell(通称为sh)，顾名思义，就是一个叫Bourne shell创建的。对，它就是现在我们机器上面的/bin/sh这个可执行文件。这个老哥创建的sh一直沿用至今，现在的UNIX操作系统都配置有sh，而且各种新的shell都会向后兼容sh的语法。

##### Bourne shell 带来了：

**脚本可以写在文件里被调用，比如sh a.sh可以执行a.sh里面的shell命令**

**可以交互或者非交互的方式调用**

**可以同步执行也可以异步执行**

**支持输入输出的pipeline，就是管道方式**

**支持输入输出的重定向，就是现在使用的> 和 >>**

**提供一系列内置命令**

**提供流程控制基本的函数和结构**

**弱类型变量，就是可以直接 a=1，不需要指定a为int**

**提供本地和全局的变量作用域**

**脚本执行前不需要编译**

**去掉goto功能**

**使用``进行命令执行替换**

**增加了for~do~done的循环**

**增加了case~in~esac的条件选择**

**文件描述符2>代表错误信息导出**

#### 为什么会有csh

##### apt-get 安装csh

##### 我们可以看到这里有个bsd,那么什么是bsd呢?

##### echo /etc/shells就有了

##### cat /bin/csh到底是什么

Bourne老爷子创造的sh非常强大，后来引入的争议是Unix系统是C写的，为什么你的shell的语法不像C呢？然后Bill Joy就编写了C Shell(csh)。它用最类似C的语法来编写shell。后来csh演化成了tchsh，但是csh后面的路途就比较坎坷了，最终未能流行起来。但是现在比如在Mac系统上还保留csh。

#### 为什么会有ksh

Korn Shell(ksh)是1983年出现的，它向后兼容Bourne shell。同时吸取了C shell的一些优点，比如job control。

#### 以上三种shell是最早的shell,在SVR4中都提供的shell

#### linux使用的主shell-bash

在1989年，现在最广泛使用的Bash出现了，它的全称叫做Bourne-Again shell。目的是为了与POSIX的标准保持一致，同时保持对sh的兼容。其实现在很多机器上的/bin/sh往往都链接到bash，我们以为我们是使用Bourne shell，实际上我们使用的是Bourne-Again shell。

现在的sh是bsh

而bash是新的

#### shell的设置和查找

我们可以为每个用户指定不同的默认shell，在/etc/passwd中设置就可以了

postgres:x:503:503::/home/postgres:/bin/bash

如何查看自己的默认shell

echo $SHELL

### csh bsd 相关历史

#### 1989年at&t,发布了SVR4 system V release 4

#### 伯克利也发布了CSEG computer system Research group加了很多自己的东西 ,发布了BSD,而且好多学校开始装这个发行版了

就像FreeBSD删除AT&T代码那样，删到最后连系统都起不来了，重新发明轮子吧，内核一夜变回20年前！

推出了一份以第六版为基础，加上一些改进和新功能而成的 Unix。这就是著名的“1 BSD（1st Berkeley Software Distribution）”，开创了Unix的另一个分支：BSD 系列。 同时期，AT&T成立USG（Unix Support Group），将 Unix变成商业化的产品。从此，BSD的 Unix 便和AT&T 的Unix 分庭抗礼，Unix就分为System IV和4.x BSD这两大主流，各自蓬勃发展。

    1979年发布的Unix 第七版被称为是“最后一个真正的Unix”，这个版本的Unix内核只有40K bytes。后来这个版本被移植到VAX机上（我在大学时学习C语言时用过这个VAX机，我还记得那时上VAX机最大的爱好就是使用talk命令和别人聊天，呵呵）。20世纪80年代相继发布的8、9、10版本只授权给了少数大学。

 1982年，AT&T基于版本7开发了UNIX System Ⅲ的第一个版本，这是一个商业版本仅供出售。为了解决混乱的UNIX版本情况，AT&T综合了其他大学和公司开发的各种UNIX，开发了UNIX System V Release 1。这个新的UNIX商业发布版本不再包含源代码，所以加州大学Berkeley分校继续开发BSD UNIX，作为UNIX System III和V的替代选择。BSD对UNIX最重要的贡献之一是TCP/IP。

这是一个AT&T妄图私有化的Unix的时代。为了私有化Unix，1986年IEEE指定了一个委员会制定了一个一个开放作业系统的标准,称为 POSIX (Portable Operating Systems Interface)。最后加上个X，不知道是为了好听，还是因为这本质上是UNIX的标准。当然，AT&T的Unix取得了这个标准制订战争的胜利，还取得了Unix这个注册商标。此时BSD的拥护者自喻为冷酷无情的公司帝国的反抗军。就销售量来说，AT&T UNIX始终赶不上BSD/Sun。到1990年，AT&T与BSD版本已难明显区分，因为彼此都有采用对方的新发明。

   Berkeley Software Design, Inc（BSDI）很快就与AT&T的UNIX Systems Laboratories（USL）附属公司产生了法律纠纷，USL是AT&T注册的公司。AT&T为了拥有System V版权，以及Unix商标，为了垄断Unix，1992年，USL正式对BSDI提起诉讼，说BSD剽窃他的源码。而最终了结了好评如潮的BSD系统。

   这场官司一直打到 AT&T将自己的Unix系统实验室卖掉，新接手的Novell公司采取了一种比较开明的做法，允许BSDI自由发布自己的BSD，但是前提是必须将来自于AT&T的代码完全删除，于是诞生了4.4 BSD Lite版，由于这个版本不存在法律问题，4.4BSD Lite成为了现代BSD系统的基础版本。

   这桩诉讼最终在1994年1月了结，更多地满足了BSDI的利益。伯克利套件的18,000个文件中，只有3个文件要求删除，另有70个文件要求修改，并显示USL的版权说明。这项调解另外要求，USL不得对4.4BSD提起诉讼，不管是用户还是BSDI代码的分发者。于是，BSD Unix走上了复兴的道路。BSD的开发也走向了几个不同的方向，并最终导致了FreeBSD、OpenBSD和NetBSD的出现。

   从AT&T意识到了Unix的商业价值，不再将Unix源码授权给学术机构以来，到以后的几十年，Unix仍在不断变化，其版权所有者不断变更，授权者的数量也在增加。Unix的版权曾经为AT&T所有，之后Novell拥有了Unix，再之后Novell又将版权出售给了SCO（这一事实双方尚存在争议）。有很多大公司在取得了Unix的授权之后，开发了自己的Unix产品。（几年前，据传闻微软为了限制Linux，微软让SCO到法院告Linux剽窃其源码）

   由于Unix是由C语言写的，所以修改和移植都很容易，因此，很多商业公司及学术机构均加入这个操作系统的研发，各个不同版本的Unix也开始蓬勃发展。这才产生了今天这么多的各式各样的Unix衍生产品。如AIX、Solaris、HP-UX、IRIX、OSF、Ultrix等等。（这些商业化的Unix基本上都是源于AT&T授权的Unix System V）

#### 标准制定 at&t受到反垄断限制

#### 设置了SVID system v interface definition

#### 后来很多公司加入,形成了POSIX

### 日期时间

#### 时间

watch -n1 "date '+%D%n%T'|figlet -k"

while true; do echo "$(date '+%D %T' | toilet -f term -F border --gay)"; sleep 1; done

#### 系统运行了多长时间

##### uptime 有的时候好几十年就没关过机器

#### unixtime时间戳

#### date

#### 日历

cal

man cal

cal -d 2010-11

cal 2020

#### 农历lunar

### Linux实现删除撤回的方法。

#### https://www.cnblogs.com/moegarn/p/10913652.html

## 存储管理

### 内存

#### 内存查看free

##### 内存到底有多大,购买服务器的时候可以看到

##### free -h

##### 如果运行我的世界服务器,1G一下就容易出问题,就需要扩容.

### 磁盘管理命令

#### df -h

#### du / -hd1

##### humanreadable

##### d1 一层

#### lshw查看硬件详细信息

#### 系统盘 数据盘 弹性云

### 微软的出现

Ibm在小型机时代被pdp+uxix干了一刀,眼看个人电脑就要兴起了,ibm如何应对呢?他要做的事统一战线,自己做标准,占据专利的核心位置 ,具体的脏活儿累活交给底下的人oem就可以.

◆**IBM PC和埃斯特利奇的传奇**

然而后来的故事却令人大跌眼镜，当DEC公司走过巅峰之后，这个年轻的巨人似乎并不具备它的对手IBM蓝色巨人的优良基因和素质，它开始愈发骄傲、蛮横自大，以至于错过了马上即将到来的个人机时代。

正当IBM在小型机市场节节失利之时，历史的天平再次向其倾斜，IBM史上最负盛名的神奇人物唐•埃斯特利奇登场了。对，就是那个波浪头！

起初埃斯特利奇在IBM只是个小人物，40多岁才混到中层干部，此人一向不喜循规蹈矩，在制度森严和充满绅士风的IBM公司里竟然崇尚起奔放狂野的牛仔风，一天到晚穿着牛仔马靴咣当晃悠不算，还沉迷摆弄竞争对手苹果的Apple II电脑。事实上，若非后来高层提及此人，他就要快被炒鱿鱼了。

小型机时代的洗礼过后，新兴的苹果公司推出的Apple II使得“个人电脑”概念在业内传播流行。此时IBM意识到个人机时代或将来临，必须抢在DEC还有苹果公司下一步行动之前出奇制胜。但是由于公司当时封闭的企业文化和过高的开发成本等原因，各高层经理都拿不出合适的个人机方案。经过激烈讨论，最后大家一致认为只有埃斯特利奇适合干这事。就这样，波浪头先生在1980年受命于IBM危难之际，去完成其一生最大的使命。

埃斯特利奇的小组一共13人连同研发中心安置在弗罗里达州博卡拉顿镇的一间仓库里，他们必须在接下来一年内做出属于IBM的个人电脑。照以往经验，公司的大型机所有组成元素包括微处理器、磁盘驱动器、操作系统和其他配件都是自己生产或买下主要供应商控制生产权，这是IBM几十年不变的战略模式。埃斯特利奇则认为应该利用市场上现成的配件来组装一台电脑，这样可以保证个人机的标准开放性，也就是后人所熟知的硬件生态。他们采购了英特尔的8086微处理器和其他芯片配件，在考虑操作系统时经过几番周折最终选择与微软合作。当盖茨和鲍尔默走进埃斯特利奇的破仓库时，他们被一群大人团团围住整整质问了一天，大致就是“我们需要怎样怎样的操作系统”、“你得多长时间给我做出来”、“钱不是问题”，吓得少年盖茨和鲍哥一宿没睡好。

硬件有了，操作系统也有了，第一台IBM PC终于在1981年8月大功告成。公司经理人中没人认为这一石头砸下去能激起多大水花，直到几个月之后大伙才知道，这简直是惊涛巨浪！IBM PC第一年就为公司带来近10亿美元收入，此后销售一路攀升，不到三年，PC部门已成为年入45亿、坐拥上万员工的全美第三大计算机组织（仅次于IBM自己和DEC）。

硬件架构和操作系统的开放性让许多供应商能够为IBM PC提供配件增值，整个PC生态如滚雪球般在这个荒蛮的时代迅速生长。个人电脑的成功让IBM在80年代进入最辉煌的时期，同时也让埃斯特利奇登上了人生巅峰，时代周刊则破天荒地将IBM PC列为“1982年风云人物”。令人悲伤的是，埃斯特利奇夫妇于1985年度假期间因飞机失事不幸丧生，一代英才就此陨落。但是历史永远记下了“PC之父”的名字，他所开创的个人电脑兼容、开放标准至今影响着业界和PC工业。

IBM搞兼容PC的时候抓来了intel生产CPU、微软做操作系统（MS-DOS），所以微软系统当然是运行在X86平台的啊。个人电脑大发展之后,微软控制了用户的电脑使用习惯路径,intel把intel inside这个概念放到每个人的脑子里.他们把ibm架空了,亢龙有悔吧ibm  
至于跨平台，“感觉技术上并不难”真的只是感觉而已，这开发成本能把当时的微软搞破产不止一次。windows的内核设计和linux是不一样的，跨平台并不是windows的开发目标，难度比linux大得多。即使是linux也不是“只需要改动底层硬件相关的代码”就能跨平台了，现在linux能兼容这么多平台是靠无数的开发者花费了成吨的时间才做到的。

因为那时候unix不能跑在x86上。  
dos最早推出是81年，8088处理器上只有它，没得选  
linux是91年的事  
386bsd是92年的事

当然dos和unix没得比，dos就一磁盘管理系统。

于是，它决定寻找合作伙伴，**最终选定了Intel做微处理器，微软做操作系统MS-DOS**，贴上IBM的商标，和IBM的外围设备一起卖，也就是后来推出的IBM-PC。高傲的IBM将这次行动命名为“象棋行动”，**然而，它自以为的两枚微不足道的棋子，在后来却将棋盘一掀，摇身变为叱咤风云的Wintel。**Intel和微软乘着IBM的东风，以惊人的速度成长起来，在各自的领域成为了说一不二的新的巨人。最直观的体现在于，1986年，Intel劝说IBM采用自己最新的386芯片，IBM没听，而他的竞争对手康柏用了，直接导致IBM的市场份额从80%掉到了40%以下。

Vax的os负责人加盟微软,开始nt系列的开发,这也是win7,win10的底层

印度裔高管

强制收购tiktok

## 历史记录

### History

**Unix开源组织**

   AT&T的这种商业态度，让当时许许多的Unix的爱好者和软件开发者们感到相当的痛心和忧虑，他们认为商业化的种种限制并不利于产生的发展，相反还能导制产品出现诸多的问题。随着商业化Unix的版本的种种限制和诸多问题，引起了大众的不满和反对。于是，大家开始有组织地结成“反叛联盟”以此对抗欺行罢市的AT&T等商业化行为。

   另一方面，关于“大教堂”（集权、封闭、受控、保密）和“集市”（分权、公开、精细的同僚复审）两种开发模式的对比成为了新思潮的中心思想。这个新思潮对IT业产生了非常深远影响。为整个计算机世界带来了革命性的价值观。

   此时，一个名叫Richard Stallman的领袖出现了，他认为Unix是一个相当好的操作系统，如果大家都能够将自己所学贡献出来，那么这个系统将会更加的优异！他倡导的Open Source的概念，就是针对Unix这一事实反对实验室里的产品商业化私有化。尽管Stallman既不是、也从来没有成为一个Unix程序员，但在后1980的大环境下，实现一个仿Unix操作系统成了他追求的明确战略目标。Richard Stallman早期的捐助者大都是新踏入Unix土地的老牌ARPANET黑客，他们对代码共享的使命感甚至比那些有更多Unix背景的人强烈。

   为了这个理想，Richard Stallman于1984年创业了GNU，计划开发一套与Unix相互兼容的的软件。1985 年 Richard Stallman 又创立了自由软件基金会（Free Software Foundation）来为 GNU 计划提供技术、法律以及财政支持。尽管 GNU 计划大部分时候是由个人自愿无偿贡献，但 FSF 有时还是会聘请程序员帮助编写。当 GNU 计划开始逐渐获得成功时，一些商业公司开始介入开发和技术支持。当中最著名的就是之后被 Red Hat 兼并的 Cygnus Solutions。

   GNU组织的建立，延续了当年Unix刚出现时的情形，并为这种情形建立了可靠的法律和财务保障。GNU 工程十几年以来, 已经成为一个对软件开发主要的影响力量， 创造了无数的重要的工具。例如：强健的编译器，有力的文本编辑器，甚至一个全功能的操作系统。从那时开始，许多程序员聚集起来开始开发一个自由的、高质量、易理解的软件，让这使得Unix社区生机勃勃，一派繁荣景象。

 GNU发布了GPL,你可以看我的源代码,改我的源代码,可以用这个源代码赚钱,但是你的源代码必须公开出来.

GNU发布了gcc工具链,还有各种应用的源代码,但是他缺少一个内核.一个基于x86的内核.

   自90年代发起这个计划以来，GNU 开始大量的产生或收集各种系统所必备的组件，像是——函数库（libraries）、编译器（compilers）、调式工具（debugs）、文本编辑器（text editors）、网站服务器（web server），以及一个Unix的使用者接口（Unix shell）等等，等等。但由于种种原因，GNU一直没有开发操作系统的kernel。正当Richard Stallman在为操作系统内核伤脑筋的时候，Linux出现了。

Unix与黑客文化  
   
黑客的文化和Unix的商业化存在着必然的联系。自从Unix出现，黑客文化就与之而来。  
   
1993初，一个悲观的观察家撰文指出，已经有理由认为Unix的传奇故事连同他带有黑客文明将一同破产。许多人预测，从那时起Unix将在六月内死亡。他们很清楚，十年的Unix商业化，使自由跨平台的Unix梦以失败告终。Unix允诺的跨平台可移植性，在一打大公司专有的Unix版本之间不停地斗嘴中丢失，一个完美的操作系统最终沦为多种版本的一团乱麻，这应该说是人类文明史上的一个重大悲剧。  
   
在专有软件社会中，只有像微软一样的“集权制，大教堂”生产方式才能成功。那个时代的人悲观地相信，技术世界的个人英雄主义时代已经结束，软件工业和发展中的互联网络将逐渐地由像微软一样的巨型企业支配，再也没有“佐罗”，世界是恺撒大帝的世界，计算机文明将进入黑暗的帝国时代。黑客已经死了，自由不付存在。  
   
自从Unix出现以来，第一代的Unix黑客似乎垂垂老矣，衣食不饱( Berkeley计算机科学研究组在1994丢失了自己基金)。这是一个抑压的时代。专有的商业Unix的结果证明那么沉重、那么盲目、那么不适当，以致微软能够用那次等技术的Windows抢走他们生存的空间，拿走他们的干粮。黑客世界的残余力量被逼到了世界上的角落里，苟延残喘。  
   
就在黑客文化日渐衰落之时，美国新闻周刊的资深记者Steven Levy完成了著名的《黑客列传》一书，书中着力介绍了一个人物：Richard M. Stallman的故事，他是麻省理工学院（MIT）人工智能实验室领袖人物，坚决反对实验室的研究成果商业化。他是商业软件社会中坚强的一员，决不随波逐流，建立了全新的黑客文化。  
   
Richard M. Stallman（他的登陆名RMS更为人熟知）早在1970年代晚期就已经证明他是当时最有能力的程序员之一。Emacs编辑器就是他众多发明中的一项。RMS的目标是将后1980的松散黑客社群变成一台有组织的社会化机器以达到一个单纯的革命目标。也许他未意识到，他的言行与当年卡尔·马克思号召产业无产阶级反抗工作的努力如出一辙。RMS宣言引发的争论至今仍存于黑客文化中。他的纲要远不止于维护一个代码库，已经暗含了废除软件知识产权主张的精髓。RMS通过“自由软件（free software）”让黑客文化更加有自我意识。当然，这个充满魅力又具争议的人物本身已经成为了一个黑客文化英雄。  
   
只有痴迷的“黑客”和具有创造力的怪人结成的反叛联盟才能把我们从愚蠢中拯救出来——他们接着教导我们，真正的专业和奉献精神，正是我们在屈服于世俗观念的“合理商业做法”之前的所作所为。 ——《The Art of Unix Programming》  
   
RMS让世界上所有的人都知道，入侵电脑系统只是低级不入流的黑客干的事，真正的黑客，是为了自由，为了软件的自由，为了挑战计算机世界中的霸权主义而斗争。他们不是街头小混混，他们更像是绿林好汉，更像是罗宾汉，更像是佐罗。就像渴望民主的人民同专制的政府斗争一样。RMS领导着许多的黑客通过互联网向专有软件发出宣战。  
   
X Windows是首批由服务于全球各地不同组织的许多个人以团队形式开发的大规模开源项目之一。电子邮件使创意得以在这个群体中快速传播，问题由此得以快速解决，而开发者可以人尽其才。软件更新可以在数小时之内发送到位，使得每个节点在整个开发过程中步调一致。网络改变了软件的开发模式。  
   
另一方面，RMS的理论体系有许多东西非常有争议，他的GPL被认为是一种“病毒式”的协议，BSD的fans和老牌Unix黑客们认为，他们编写Unix的年头都比GPL声明要长得多，GPL依然有太多的限制，而BSD协议则比GPL更加的自由。另一方面，RMS走向了另一个极端，他是完全反版权的，反商业化的。把软件产品从强制收费推向了强制免费、共享和开源，这也为他带来了许多许多的争议。  
   
在RMS组织黑客闹革命的年代里，没有多少黑客认同于RMS的理论体系，更多的他们参与GNU只是为了体现那种在互联网上协同工作，令人激动的工作模式。自从GNU设立以来，争议不断，而黑客文化却从未有统一在他的理想体系之下。  
   
自从Linux出现以后，一个新的黑客领袖出现了，Linus Torvalds的中庸态度网聚了世界上顶尖的黑客，其绕过了GPL和反GPL的派系之争，他使用GNU的工具从而以GPL的“传染性”保护了Linux，但他同时也不承认RMS的理论思想体系，他即开源，又支持商业化。虽然，他没有带给黑客们什么重要的思想体系或统一的价值观，但他整合了全世界黑客的阵营，让所有的黑客的行为都围绕着Linux这一事物进行。他以“用自由软件是因为它运行得更好”轻而易举地盖过了“用自由软件是因为所有软件都该是自由的”。  
   
1998年初，这种新思潮促使网景公司（Netscape Communications）公布了其Mozilla浏览器的源码。媒体对此事件的关注促成了Linux在华尔街的上市，推动了1999－2001年间科技股的繁荣。事实证明，此事无论对黑客文化的历史还是对Unix的历史都是一个转折点。  
   
Unix的历史教训  
下面的文字出自《The Art of Unix Programming》（Unix编程艺术）。令今天我们所有人所反思。  
   
在Unix历史中，最大的规律就是：  
距开源越近就越繁荣。任何将Unix专有化的企图，只能陷入停滞和衰败。  
   
回顾过去，我们早该认识到这一点。1984年至今，我们浪费了十年时间才学到这个教训。如果我们日后不思悔改，可能还得大吃苦头。  
   
虽然我们在软件设计这个重要但狭窄的领域比其他人聪明，但这不能使我们摆脱对技术与经济相互作用影响的茫然，而这些就发生在我们的眼皮底下。即使Unix社区中最具洞察力、最具远见卓识的思想家，他们的眼光终究有限。对今后的教训就是：过度依赖任何一种技术或者商业模式都是错误的——相反，保持软件及其设计传统的的灵活性才是长存之道。  
   
另一个教训是：别和低价而灵活的方案较劲。或者，换句话说，低档的硬件只要数量足够，就能爬上性能曲线而最终获胜。经济学家Clayton Christensen称之为“破坏性技术”，他在《创新者窘境》（The Innovator's Dilemma）[Christensen]一书中以磁盘驱动器、蒸汽挖土机和摩托车为例阐明了这种现象的发生。当小型机取代大型机、工作站和服务器取代小型机以及日用Intel机器又取代工作站和服务器时，我们也看到了这种现象。开源运动获得成功正是由于软件的大众化。Unix要繁荣，就必须继续采用吸纳低价而灵活的方案的诀窍，而不是去反对它们。  
   
最后，旧学派的Unix社区因采用了传统的公司组织、财务和市场等命令机制而最终未能实现“职业化”。只有痴迷的“黑客”和具有创造力的怪人结成的反叛联盟才能把我们从愚蠢中拯救出来——他们接着教导我们，真正的专业和奉献精神，正是我们在屈服于世俗观念的“合理商业做法”之前的所作所为。  
   
   
Unix族谱  
   
Unix的故事仍旧延续着……，许多网站也为这段历史留下记录。一个详细记录Unix历史的网站（[url]http://www.levenez.com/unix/[/url]），这个网站忠实记载着1969～2005 年Unix发展的大事，而且还有 PDF 档案可供下载，上面有一个庞大的UNIX家族版本树，让人叹为观止。网站的首页陈列每个时期Unix的历史，也代表着无数工程师的心血与努力。  
   
下面是一个简单的Unix的族谱：  
  
  
   
     |--AT&T (1969)-----\  
     |                  |  
     |              V6 (1976)  
     |                  |  
     |              V7 (1979)  
     |                  |  
     |   Novell owns AT&T's Unix (by 1994)  
     |     \_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     |     |       |      |        |         |  
     |    AIX    IRIX    SCO   HP-UX   Solaris 2.X  
     |   (IBM)   (SGI)          (HP)     (Sun)  
     |  
     |  
     |--Berkley (1977)-----\  
     |                     |  
     |                  1BSD (1977)  
UNIX-|                     |  
     |                4.4BSD (1993)  
     |                     |  
     |                   Net/2  
     |                     |  
     |               4.4BSD-Lite (by 1995)  
     |     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
     |     |       |          |         |          |            |  
     |   SunOS   Ultrix   NetBSD    OSF/1   NeXTSTEP   Mac OS X  
     |   (Sun)   (DEC)   (Various)  (DEC)    (NeXT)    (Apple)  
     |                   (FreeBSD)  
     |  
     |  
     |--Hybrids----\  
                   |  
                Linux (Various)  
                   |  
                   |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
                   |    |      |          |              |      |  
                   | RedHat  Debian  Mandrake   Slackware    S.u.S.E.  
                   |                          (Walnut Creek)  
                   |  
                   |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
                       |        |           |          |        |  
                    MkLinux  LinuxPPC  TurboLinux  OpenLinux  CorelLinux  
                    (Apple)                        (Caldera)   (Corel)  
   
  
  
Unix的特点  
   
现在的文献中提到Unix基本上是说，由Ken Thompson和Dennis Ritchie共同开发的。而通过历史我们也能发现，Unix的主要是由Ken Thompson写下的。但在学术界，Dennis Ritchie的名字往往被排在了Ken Thompson前面的。这就是因为，Dennis Ritchie不但发明了C语言，而且当时他设计Unix操作系统的设计思想，影响了整个世界，直到今天。  
   
当时，他们开发UNIX，没有正式立项，是Ken Thompson和Dennis Ritchie等少数几个人偷偷干的，如果一切都要从头从新设计，那几乎是不可能的。所以，Unix吸取与借鉴了Multics的经验，如内核，进程，层次式目录，面向流的I/O，把设备当作文件，……等等。但是Unix在继承中又有创新，比如Unix采用一种无格式的文件结构，文件由字节串加\0组成。这带来两大好处：一是在说明文件时不必加进许多无关的“填充物”，二是任何程序的输出可直接用作其他任何程序的输入，不必经过转换。后面这一点叫做“管道”(piping)，这就是Unix首创的。此外，像把设备当作文件，从而简化了设备管理这一操作系统设计中的难题，虽然不是UNIX的发明，但是实现上它采用了一些新方法，比Multics更高明一些。  
   
下面是Unix的特点：（30多年过去了，这些东西早已变成经典）  
   
l         Everything (including hardware) is a file  
所有的事物（甚至硬件本身）都是一个的文件。  
l         Configuration data stored in text  
以文件形式储存配置数据。  
l         Small, single-purpose program  
程序尽量朝向小而单一的目标设计  
l         Avoid captive user interfaces  
尽量避免令人困惑的用户接口  
l         Ability to chain program together to perform complex tasks  
将几个程序连结起来，处理大而复杂的工作。  
   
   
Unix的影响和哲学  
   
Unix是第三次工业革命中计算机软件领域最具代表性的产物。在这近40年中，由Unix造成的影响是最有深远意义的。就我看来，Unix为软件领域带来了至少以下有积极的东西，由这些东西所引发的直接或间接的事物更是举不胜数。  
   
1）  软件开发的若干哲学和思想。  
2）  全民参与推动软件，代码共享的模式。  
3）  开启了黑客文化和开源项目。  
4）  免费和商业的完美结合的Linux。  
5）  C语言，而后发展的C++，Java等等类C的语言和脚本。  
6）  TCP/IP，其的Socket编程已成为今天通用的网络编程主流。  
   
不能不说，AT&T虽然发展了Unix，但今天Unix的混乱的局面也和AT&T 有着直接原因。但反过来说，如果没有AT&T的反面教材，今天的GNU/Linux很有可能也不会出现。AT&T究竟是限制了Unix的发展，还是以反面示例促进了Unix社区，已不好评说。今天，软件是商业化好还是开源好的争论还在继续，纵观这几十年来Unix的历史，Linux的划时代地出现。相信你会得出自己的结论。不管怎么样，Unix的经历对计算机领域贡献的不单单是技术，他给我们提供了丰富而生动的教材。特别是Unix引发的哲学，让今天的我们依然受益不浅。  
   
说到Unix为我们所带来的软件开发的哲学，我必需要说一说。Unix遵循的原则是KISS（Keep it simple, stupid）。在[url]http://en.wikipedia.org/wiki/Unix\_philosophy[/url] 上有很多的基本上大同小异的Unix哲学，都是很经典的。  
   
Doug McIlroy 是认为UNIX的哲学是这样的：三条哲学，简明扼要，就是这三条哲学贯穿着整个Unix世界。尤其是第一条“do one thing and do it well”真是相当精彩！  
   
l         Write programs that do one thing and do it well.  
l         Write programs to work together.  
l         Write programs to handle text streams, because that is a universal interface.  
   
只要是Unix的程序员，他们会比别的程序员在任何时候都会不停地强调着这三条哲学。  
   
而《The Art of Unix Programming》总结了下面这些哲学，都是至理名言啊。  
Rule of Modularity: Write simple parts connected by clean interfaces.  
Rule of Clarity: Clarity is better than cleverness.  
Rule of Composition: Design programs to be connected to other programs.  
Rule of Separation: Separate policy from mechanism; separate interfaces from engines.  
Rule of Simplicity: Design for simplicity; add complexity only where you must.  
Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.  
Rule of Transparency: Design for visibility to make inspection and debugging easier.  
Rule of Robustness: Robustness is the child of transparency and simplicity.  
Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.  
Rule of Least Surprise: In interface design, always do the least surprising thing.  
Rule of Silence: When a program has nothing surprising to say, it should say nothing.  
Rule of Repair: When you must fail, fail noisily and as soon as possible.  
Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.  
Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.  
Rule of Optimization: Prototype before polishing. Get it working before you optimize it.  
Rule of Diversity: Distrust all claims for "one true way".  
Rule of Extensibility: Design for the future, because it will be here sooner than you think.  
X Windows 的设计者 Mike Gancarz 给出了下面九条哲学思想  
Small is beautiful.  
Make each program do one thing well.  
Build a prototype as soon as possible.  
Choose portability over efficiency.  
Store data in flat text files.  
Use software leverage to your advantage.  
Use shell scripts to increase leverage and portability.  
Avoid captive user interfaces.  
Make every program a filter.  
在今天，这种思想依然被传承着，在影响着世界上各个角落的每一个程序员。  
   
Unix痛恨者手册  
   
这里还需要值得一提的是一本叫《The Unix-Haters Handbook》，中文译做《Unix痛恨者手册》。可以在这里下载：[url]http://research.microsoft.com/~daniel/uhh-download.html[/url]。其中以调侃的语气声讨了Unix的种种不是。虽然这是十年前的一本书了，但还是值得一读。这本书指出了许多Unix的设计错误，指出了种种看起来很合理的设计走向了荒谬，还这样调侃了C语言——“如果说C语言给足了让你上吊的绳子，那么，C++在给了你足够的绳子把你的邻居全部捆起来之后，还给了你足够的绳子让你为一艘小帆船装上帆，最后你还有足够的绳子把自己吊死在帆船的桅杆上”。呵呵，相当的尖酸刻薄吧。里面有一句对操作系统的评价是这样的：“The fundamental difference between Unix and the Macintosh operating system is that Unix was designed to please programmers, whereas the Mac was designed to please users. (Windows, on the other hand, was designed to please accountants.”（Windows设计给会计人员？！连计算机用户都不是了，呵呵）  
   
不过，我可以感觉得到这本书的作者在书中对Unix的感情是比较复杂的，爱恨交加，在书的最后有这样一句话“would anyone have spent this much time and effort writing about how much they hated Unix if they didn't secretly love it? I'll leave that to the readers to judge, but in the end, it really doesn't matter: If this book doesn't kill Unix, nothing will”。是的，如果Unix能够存活这么长的时间，那么，不会有什么东西可以把他消灭了。  
   
从《Unix痛恨者手册》这本书，再加上Unix的历史，我们可以感到Unix的经历的风风雨雨，在Unix上面出现有种种教训，近40年的历程，Unix历经磨难，几近夭折，一路走来的确很不容易，让人由衷感叹。今天的Unix，今天的软件工业和以前相比已是不可同日而语。很大程度上，这些都要归功于这个充满苍桑的Unix。  
   
后记  
   
       在中国我们开始学习计算机的时候，我们被Microsoft所创造的文化所笼罩里。就在Unix出现革命性的转变，在Unix影响计算机世界文化的那几年里，科班出生专业开发人员学习的是MS-DOS和微软的文化，我们犹如一个井底之蛙一样，对外面的翻天覆地的变化无动于衷。微软创造的文化在我们这里尤其地根深蒂固，我们几乎忘记了另外一边的Unix。  
   
在那充满激情的Unix的岁月里，大伙为了科研目的或个人兴趣在Unix上进行各种开发，并且不计较金钱利益，将这些源码公开，互相共享。在那里，开发和自由成为主题，正因为如此，当今的世界才如此丰富多采。在近40年Unix文化和技术积淀的里面，蕴涵着比较纯正的计算机文化和思想。  
   
纵观整个Unix的历史过程中，许许多多的程序员、工程师前辈们在Unix中所摸爬滚打，他们的辛勤地、他们呕心沥血地跟随Unix，努力建立一个繁荣的计算机世界的文明。Unix不是一个简简单单的操作系统。有人说，Unix是程序员设计给程序员的，一点没错。Unix的近40年历史造就了它的博大精深，它给程序员们带来的绝不仅仅只是技术上的知识。它的失误，它的无奈，它的精神，它的荣耀，它从技术和思想上都启迪着我们。对于程序员来说，学习Unix就等同于向前辈程序学习。无论你是什么样的程序员，你都应该了解Unix，这是开发人员的根，前面的开发者造就了它，而它又在引领后面的开发人员，它是前辈程序员们交给我们的一份礼物，一个接力棒，它是开发人员赖以生存的土壤，是上一辈程序员留给我们这一代程序员开启未来的钥匙。Unix就像一个程序员教父一样，理当受到我们的尊敬和崇拜。

关于1979年的版权声明中，影响最大的当然就是学校教Unix核心原始码相关学问的教授了！ 想一想，如果没有核心原始码，那么如何教导学生认识Unix呢？这问题对于Andrew Tanenbaum(谭宁邦)教授来说，实在是很伤脑筋的！不过，学校的课程还是得继续啊！那怎么办？

既然1979年的Unix第七版可以在Intel的x86架构上面进行移植， 那么是否意味着可以将Unix改写并移植到x86上面了呢？在这个想法上， 谭宁邦教授于是乎自己动手写了Minix这个Unix Like的核心程序！ 在撰写的过程中，为了避免版权纠纷，谭宁邦完全不看Unix核心原始码！ 并且强调他的Minix必须能够与Unix兼容才行！谭宁邦在1984年开始撰写核心程序， 到了1986年终于完成，并于次年出版Minix相关书籍，同时与新闻组(BBS及News)相结合～

这个Minix版本比较有趣的地方是，他并不是完全免费的，无法在网络上提供下载！ 必须要透过磁盘/磁带购买才行！虽然真的很便宜～不过，毕竟因为没有在网络上流传， 所以Minix的传递速度并没有很快速！此外，购买时，随磁盘还会附上Minix的原始码！ 这意味着使用者可以学习Minix的核心程序设计概念喔！ (这个特色对于Linux的启始开发阶段，可是有很大的关系喔！)

此外，Minix操作系统的开发者仅有谭宁邦教授，因为学者很忙啊！加上谭宁邦始终认为Minix主要用在教育用途上面， 所以对于Minix是点到为止！没错，Minix是很受欢迎，不过，使用者的要求/需求的声音可能就比较没有办法上升到比较高的地方了！ 这样说，你明白吧？^\_^

* 查看一下你最常用的10个命令：

cat .bash\_history | sort | uniq -c | sort -rn | head -n 10 (or cat .zhistory | sort | uniq -c | sort -rn | head -n 10

## 编译内核

### 换内核

#### kernel.org上面的文件是c语言的文件,如果要编译的话,就会自动用本地gcc对应的指令集进行编译

#### 如果我的要编译的内核是mips的怎么办?

##### 就到mips的linux上面,用本机的gcc进行编译

##### 或者交叉编译工具链

###### https://mirrors.edge.kernel.org/pub/tools/crosstool/

* + 1. 比如说我在intel-64要编译arm64,就用左上角第一个

###### 或者别的工具链

* + 1. 建议用 musl.cc 的工具链

linux历史**Linux横空出世**

   1990年，Linus Torvalds还是芬兰赫尔辛基大学的一名学生，最初是用汇编语言写了一个在80386保护模式下处理多任务切换的程序，后来从Minix（Andy Tanenbaum教授所写的很小 的Unix操作系统,主要用于操作系统教学）得到灵感，进一步产生了自认为狂妄的想法——写一个比Minix更好的Minix，于是开始写了一些硬件的设备驱动程序，一个小的文件系统。这样0.0.1版本的Linux就出来了，但是它只具有操作系统内核的勉强的雏形，甚至不能运行，你必须在有Minix的机器上编译以后才能玩。这时候Linus已经完全着迷而不想停止，决定踢开Minix，于是在1991年10 月5号发布Linux 0.0.2版本,在这个版本中已经可以运行bash 和gcc。

   从一开始，Linus就决定自由扩散Linux，包括原代码，随即Linux引起黑客们（hacker）的注意，通过计算机网络加入了Linux的内核开发。Linux倾向于成为一个黑客的系统——直到今天，在Linux社区里内核的开发被认为是真正的编程。由于一批高水平黑客的加入，使Linux 发展迅猛，几乎一两个礼拜就有新版或修正版的出现，到1993年底94年初，Linux 1.0终于诞生了！Linux 1.0已经是一个功能完备的操作系统，而且内核写得紧凑高效，可以充分发挥硬件的性能，在4M内存的80386机器上也表现得非常好，至今人们还在津津乐道。时至今日，kernel的版本已经出到2.6。Linux的发展不像传统的软件工程，它完全是透过网络，集合世界各地的高手而成的一套操作系统，在这里我们也可以见识到网络快速传播的威力。Linux初次让整个世界感觉到了开源力量和网络力量的如此强大。（Linux 的标志和吉祥物是一只名字叫做 Tux 的 企鹅，标志的由来是因为Linus在澳洲时曾被一只动物园里的企鹅咬了一口，便选择了企鹅作为Linux的标志。）

   Linux 的历史是和GNU紧密联系在一起的。从1983年开始的GNU计划致力于开发一个自由并且完整的类Unix操作系统，包括软件开发工具和各种应用程序。到1991年 Linux 内核发布的时候，GNU已经几乎完成了除了系统内核之外的各种必备软件的开发。在 Linus Torvalds 和其它开发人员的努力下，GNU组件可以运行于Linux内核之上。整个内核是基于 GNU 通用公共许可，也就是GPL（GNU General Public License，GNU通用公共许可证）的，但是Linux内核并不是GNU 计划的一部分。1994年3月，Linux1.0版正式发布，Marc Ewing成立了 Red Hat 软件公司，成为最著名的 Linux 分销商之一。

   严格来讲，Linux这个词本身只表示Linux内核，但在实际上人们已经习惯了用Linux来形容整个基于Linux内核，并且使用GNU 工程各种工具和应用程序的操作系统(也被称为GNU/Linux)。基于这些组件的Linux软件被称为Linux发行版。一般来讲，一个Linux发行套件包含大量的软件，比如软件开发工具，数据库，Web服务器（例如Apache)，X Window，桌面环境（比如GNOME和KDE），办公套件（比如OpenOffice.org），等等。

   1991至1995年间，Linux从概念型的0.1版本内核原型，发展成为能够在性能和特性上均堪媲美专有Unix的操作系统，并且在连续正常工作时间等重要统计数据上打败了这些Unix中的绝大部分。1995年，Linux找到了自己的杀手级应用——开源的web服务器Apache。就像Linux，Apache出众地稳定和高效。很快，运行Apache的Linux机器成了全球ISP平台的首选。约60%的网站选用Apache，轻松击败了另两个主要的专有型竞争对手。今天的LAMP（Linux , Apache, MySQL, PHP）已经成为了架构Web服务器的主要首选。

   现如今的Linux不但可以装在几乎所有的主流服务器上，当然也包括桌面的X86系统中。其还常常被用于嵌入式系统，机顶盒、手机、交换机、游戏机、PDA、网络交换机、路由器、等等，都是因为Linux那精彩的内核。

   Linux的出现，不仅仅给世界带来了一个免费的操作系统，也不仅仅是对Unix自由、共享的文化的延续，它的出现带给了计算机世界自Unix、GNU以来更为成熟的思想和文化。

**Linux今天的领袖**

   Linux和GNU关系是比较微妙的。那时，自由软件基金会编写的用户软件工具包铺平了一条摆脱高成本专有软件开发工具的前进道路。意识服从经济，而不是领导：一些新手加入了RMS的革命运动，高举GPL大旗，另一些人则更认同整体意义上的Unix传统，加入了反对GPL的阵营，但其他大部分人置身事外，一心编码。

   Linus Torvalds巧妙地跨越了GPL和反GPL的派别之争。他利用GNU工具包搭起了自创的Linux内核，用GPL的传染性质保护它，但拒绝认同Richard Stallman的许可协议反映的思想体系计划。Linus Torvalds明确表示他认为自由软件一般情况下更好，但他偶尔也用专有软件。即使在他自己的事业中，他也拒绝成为狂热分子。这一点极大地吸引了大多数黑客，他们虽然早就反感Richard Stallman的言辞，但他们的怀疑论一直缺个有影响力或者令人信服的代言人。而Linus Torvalds正好充当了这一角色。

   Linus Torvalds令人愉快的实用主义及灵活而低调的行事风格，促使黑客文化在1993至1997年间取得了一连串令人惊奇的胜利，不仅仅在技术上的成功，还让围绕Linux操作系统的发行、服务和支持产业有了坚实的开端。结果，他的名望和影响也一飞冲天。Torvalds成为了互联网时代的英雄；到1995年为止，他只用了四年时间就在整个黑客文化界声名显赫，而Richard Stallman为此花了十五年，而且他还远远超过了Stallman向外界贩卖“自由软件”的记录。与Torvalds相比，Richard Stallman的言辞渐渐显得既刺耳又无力。

   今天，我们也说不清楚是GNU Linux还是Linux GNU。Linux既不排斥开源，也不排斥商业化，Linus认为好的软件是需要免费和商业化共同推进的。正是这种革命性的想法，造就了今天的Linux火红的局面。Linux就像一股清泉流入了所有人的心中，引发了很多的启迪和思考。

当1978年,intel开始了一个新的产品8086,这个产品性价比很高,甚至有可能制作个人版的计算机,以前的机器个人很难买的起,随着80286,80386,80486到奔腾cpup的进化,基于x86架构的操作系统也就呼之欲出了.

   由于最后判决悬而未决，这桩法律诉讼将BSD后裔的开发，特别是自由软件，延迟了两年，这导致没有法律问题的Linux内核获得了极大的支持。Linux跟386BSD的开发几乎同时起步，Linus说，当时如果有自由的基于386的Unix-like操作系统，他就可能不会创造Linux。尽管无法预料这给以后的软件业究竟造成了什么样的影响（如果没有这个法律纠纷，很有可能没有今天的革命性的Linux），但有一点可以肯定，Linux更加丰富了这块土壤。



到了1991年，芬兰的赫尔辛基大学的Linus Torvalds在BBS上面贴了一则消息， 宣称他以bash, gcc等工具写了一个小小的核心程序，这个核心程序可以在Intel的386机器上面运作， 让很多人很感兴趣！从此开始了Linux不平凡的路程！

## 文本操作

### grep

#### fortune-zh >>1.txt 追加八次

##### grep 白 1.txt

###### 找到李白等

##### grep -n 白 1.txt

#### 循环执行程序

for i in {1..10}; do echo "Hello, World"; done  
for I in 「1..100」;do fortune >> 2.txt; done

#### 控制大小写grep the 2.txt

##### grep -c 计数

##### 可以搜索红楼梦中出现雪芹两个字的情况

###### 当时有个哥们在实验室叫做lee mcmahon他想看看联邦文集The Federalist Papers到底是谁写的,这套文集有好多文档有85篇文章,作者到底什么情况呢?计算机可以帮我们统计吗?他中午吃饭的时候就和ken。Tompson聊到这个,第二天tompson就给做出来个grep

###### 第二天天

当时unix只有32k的内存,如果用当时的编辑器ed去做,不可能的,因为整套文档超过1M

###### 可以搜搜kernel里面是否有linus的名字

##### grep 大写the

##### grep 小写 The

##### grep -i 忽略大小写

##### 大写一个数量

##### 小写一个数量

##### -ic 不是和,为什么 观察 ,一行里面可能有两个

#### 控制不能是they怎么办

##### grep “the ” 4.txt

##### 排除所有的they

###### -v排除的意思

cat test.log | grep "login"|grep -v "deviceType"

#### grep -s substitute

#### 正则表达式

##### grep -E “[日月]” oeasy.txt

##### grep -E “[hjkl]” fortune.txt

##### grep -E “\w{6,}” fortune.txt

##### https://blog.robertelder.org/regular-expression-visualizer/

### sed

### awk

$ awk -F":" '{ print $1 }' /etc/passwd

$ awk -F":" '{ print $1 $3 }' /etc/passwd

$ awk -F":" '{ print $1 " " $3 }' /etc/passwd

$ awk -F":" '{ print "username: " $1 "\t\tuid:" $3" }' /etc/passwd

awk '{if(NR>=20 && NR<=30) print $1}' test.txt

只看20行到30行

[root@Gin scripts]# cat test.txt

I am Poe,my qq is 33794712

请从该文件中过滤出'Poe'字符串与33794712，最后输出的结果为：Poe 33794712

1

2

[root@Gin scripts]# awk -F '[ ,]+' '{print $3" "$7}' test.txt

Poe 33794712

[root@Gin scripts]# awk '{count++;print $0;} END{print "user count is ",count}' passwd

[root@Gin scripts]# awk 'BEGIN {count=0;print "[start] user count is ",count} {count=count+1;print $0} END{print "[end] user count is ",count}' passwd

[root@Gin scripts]# ll |awk 'BEGIN {size=0;} {size=size+$5;} END{print "[end]size is ",size}'

如果以M为单位显示:

1

2

[root@Gin scripts]# ll |awk 'BEGIN{size=0;} {size=size+$5;} END{print "[end]size is ",size/1024/1024,"M"}'

[end]size is 0.00142002 M

https://www.cnblogs.com/ginvip/p/6352157.html

### sort

#### sort -k2 按照第几列排序

#### sort -k2 -n按照数字方式排序

#### ls -la | sort -k7 -n

### rev

### tac

### tr

## 显示与计算

### 计算expr

#### Factor命令

##### Man factor

##### 选项

###### Factor –help

###### Factor –version

##### 参数

###### Factor 200

###### Factor 大小写是否影响

###### Factor 20000000000000000000000

###### Factor 0

###### Factor 0.111

###### Factor abc

###### Alias factor = fa

### 数学计算bc

#### Bc 直接用

#### Man bc

##### Bc 执行文件

#### Bc作为管道

##### pi=$(echo "scale=10; 4\*a(1)" | bc -l)

##### 定义函数e

##### 564,566 w 1.cal

##### 文件打开方式

### unix哲学,

#### do one thing at a time and do well

#### 少则得 多则惑

keep it simple, stupid

UNIX 的特点：

**Everything (including hardware) is a file.**所有的事物（甚至硬件本身）都是一个的文件。

**Configuration data stored in text.**以文本形式储存配置数据。

**Small, single-purpose program.**程序尽量朝向小而单一的目标设计。

##### 指甲刀就是指甲刀,小刀就是小刀,不要瑞士军刀

**Avoid captive user interfaces.**避免过于复杂花哨的界面。

**Ability to chain program together to perform complex tasks.**将几个程序连结起来，处理大而复杂的工作。

 流水线作业,而不是一体机

Doug McIlroy（UNIX 管道的发明人、UNIX 传统的奠基人之一） 认为 UNIX 的哲学是这样的：

**Write programs that do one thing and do it well.**

写一次只做一件事，并能把这件事做好的程序。

**Write programs to work together.**

写互相协作（调用）的程序。

**Write programs to handle text streams, because that is a universal interface.**

写处理文件流的程序。因为这（处理文件流）是一个通用接口。

《The Art of Unix Programming》 总结了下面这些哲学：

**Rule of Modularity: Write simple parts connected by clean interfaces.**

模块化原则：写简单的，能够用清晰的接口连接的代码。

**Rule of Clarity: Clarity is better than cleverness.**

清晰化原则：清晰的代码要好过“聪明”的代码。

**Rule of Composition: Design programs to be connected to other programs.**

组件化原则：设计可以互相关联（拆分）的程序。

**Rule of Separation: Separate policy from mechanism; separate interfaces from engines.**

隔离原则：策略和机制分离，接口和引擎分离。

**Rule of Simplicity: Design for simplicity; add complexity only where you must.**

简洁原则：设计力求简洁，直到无法更简洁。

**Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.**

小巧原则：不要写大的程序（模块、方法）。除非很明显的，没有别的办法可以完成。

**Rule of Transparency: Design for visibility to make inspection and debugging easier.**

透明原则：为可见性设计，使检查和调试更容易。

**Rule of Robustness: Robustness is the child of transparency and simplicity.**

健壮性原则：健壮性是透明和简单的孩子。

**Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.**

陈述性原则：将认知转化为数据。所以，程序的逻辑可以是愚蠢（简单易懂）的，健壮的。

**Rule of Least Surprise: In interface design, always do the least surprising thing.**

最少的惊讶原则：在界面设计中，少做令人惊讶的设计。（不要标新立异）

**Rule of Silence: When a program has nothing surprising to say, it should say nothing.**

沉默原则：如果一个程序没有什么特别的东西要说（输出），那就什么都别说。

**Rule of Repair: When you must fail, fail noisily and as soon as possible.**

修复原则：如果必须失败，那就尽早。

**Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.**

节约原则：程序员的时间是非常宝贵的。程序员的时间（编程时间）优于机器时间。

**Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.**

 生产原则：避免手工编程。如果可以的话，编写可以编写程序的代码。

**Rule of Optimization: Prototype before polishing. Get it working before you optimize it.**

优化原则：建立原型后再去修正。当它能正常工作后，再去优化它。

**Rule of Diversity: Distrust all claims for “one true way”.**

多样性原则：怀疑所有所谓的“不二法门”。

**Rule of Extensibility: Design for the future, because it will be here sooner than you think.**

扩展原则：为未来设计，因为未来来的比你想象的要早。

Mike Gancarz （X Windows 的设计者）给出了下面九条哲学思想：

**Small is beautiful.**

小即是美。

**Make each program do one thing well.**

让每个程序（方法）只做一件事情，并把它做好。

**Build a prototype as soon as possible.**

尽早建立原型。

**Choose portability over efficiency.**

注重可移植性，而非效率。

**Store data in flat text files.**

将数据存储在存文本文件中。

**Use software leverage to your advantage.**

利用软件来发挥你的优势。

**Use shell scripts to increase leverage and portability.**

使用Shell脚本提高编程的手段和程序的可移植性。

**Avoid captive user interfaces.**

避免过于复杂花哨的界面。

**Make every program a filter.**

使每个程序（方法）称为一个过滤器（筛选器）

## 变量的控制

### Export、set、env

### $PATH的设置

#### ls在哪,which ls

##### 如果把它从¥PATH中去除

##### ls还好用吗

#### java的JAVA\_HOME和¥PATH设置

#### .profile的设置

### Var

### 定一个$num变量 然后factor $num

## 进程管理

### Ps

## 系统编程

### 文件名里面可以最多有多少个字符?

### 输出一个ascii码表

### ssh权限控制可以用

#### 新root中应该有可以有自己的shell

#### https://www.cnblogs.com/LiuYanYGZ/p/12463926.html

#### https://blog.csdn.net/yanggd1987/article/details/54310686

## 需求

   普通用户通过ssh登陆到跳板机后，再通过ssh跳转到内网其他服务器。跳板机只提供ssh、ls等基本命令，禁止scp、sftp、管道等以防数据传输到本地；另外用户登陆后需要将其锁定在特定的目录下，防止用户浏览服务器数据。   
   通常情况下我们使用磁盘限额来限制用户传输数据，但是磁盘限额必须是针对独立的磁盘分区，而不能够是文件目录，因此用户家目录必须是独立挂载的，若是放在根目录下，配置磁盘限额效果不大。   
   ssh的chroot功能就可以很好的解决此类问题。根据sshd\_config的man中所述，实现chroot功能需要配置”ChrootDirectory”这个参数。   
   ChrootDirectory：定义了用户通过认证以后的chroot目录，此目录及其所有子目录的属主必须是root，且这些目录只有root帐号可以进行写操作，其他任何组和帐号都不可写。chroot以后，sshd会将用户的工作目录转到chroot目录中用户自己的主目录。如果ChrootDirectory定义的目录下没有相应的/home/username目录，则会直接转到chroot的/目录下

## 配置

本次实验将test用户锁定在/var/chroot下，test登陆后只能使用ssh、ls等基本命令。

Ssh连接的时候用-v观察细节

一、创建用户   
1.创建test用户(不创建其缺省家目录)

useradd -M test

passwd test

* 1
* 2

2.建立chroot目录中用户主目录

mkdir -p /var/chroot/home/test

chown -R test.test /var/chroot/home/test

chmod 700 /var/chroot/home/test

* 1
* 2
* 3

二、搭建基本的chroot环境

**注意：**一个最基本的chroot环境至少有一个shell(例如sh,bash)和一些必要的系统设备文件(例如/dev/null，/dev/zero)，如果要允许用户执行一些命令，那么还要准备相应的命令可执行文件和命令依赖的库文件

mkdir -p /var/chroot

cd /var/chroot

mkdir {bin,dev,lib64,etc,home}

mknod dev/null c 1 3

mknod dev/zero c 1 5

#ssh命令需要，如缺少会报告：PRNG is not seeded

mknod dev/random c 1 8

mknod dev/urandom c 1 9

#ssh命令需要，如缺少会报告：Host key verification failed

mknod dev/tty c 5 0

#修改/var/chroot及其子目录的属主，并修改权限

chown -R root.root /var/chroot

chmod -R 755 /var/chroot

#允许用户写这些设备文件，不可写会有些命令报错

chmod 0666 dev/{null,zero,tty}

#复制/etc/passwd和/etc/group文件到/var/chroot/etc中，并删除用户自己和root以外的所有帐号。如果没有这两个文件，用登录以后会报“I have no name!”

cp -p /etc/passwd /var/chroot/etc/

cp -p /etc/group /var/chroot/etc/

cat /var/chroot/etc/group

root:x:0:

test:x:516:

cat /var/chroot/etc/passwd

root:x:0:0:root:/root:/bin/bash

test:x:516:516::/home/test:/bin/bash

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24

三、配置ssh

vim /etc/ssh/sshd\_config

#在最后添加如下行，否则会报错

Match User test

ChrootDirectory /var/chroot

#重启ssh

service sshd restart

* 1
* 2
* 3
* 4
* 5
* 6
* 7

四、拷贝基础命令   
例如用户登陆后必须有一个可用的shell，因此用到/bin/bash，还有其他命令如ls、mkdir等。   
**注意：**/bin/ls命令和/usr/bin/ssh命令使用的类库文件目录不一样，因此我们需要在执行脚本前先建好相应的路径(此问题已经通过脚本解决)

ldd /bin/ls | awk '{ print $3 }' | grep "/lib" | sort | uniq

/lib64/libacl.so.1

/lib64/libattr.so.1

/lib64/libcap.so.2

/lib64/libc.so.6

/lib64/libdl.so.2

/lib64/libpthread.so.0

/lib64/librt.so.1

/lib64/libselinux.so.1

ldd /usr/bin/ssh | awk '{ print $3 }' | grep "/lib" | sort | uniq

/lib64/libcom\_err.so.2

/lib64/libcrypt.so.1

/lib64/libc.so.6

/lib64/libdl.so.2

/lib64/libfipscheck.so.1

/lib64/libfreebl3.so

/lib64/libgssapi\_krb5.so.2

/lib64/libk5crypto.so.3

/lib64/libkeyutils.so.1

/lib64/libkrb5.so.3

/lib64/libkrb5support.so.0

/lib64/libnsl.so.1

/lib64/libnspr4.so

/lib64/libplc4.so

/lib64/libplds4.so

/lib64/libpthread.so.0

/lib64/libresolv.so.2

/lib64/librt.so.1

/lib64/libselinux.so.1

/lib64/libutil.so.1

/lib64/libz.so.1

/usr/lib64/libcrypto.so.10

/usr/lib64/libnss3.so

/usr/lib64/libnssutil3.so

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35

在此我们使用脚本:

#!/bin/bash

#comment:用于ssh登陆chroot后，给用户添加命令

# 要允许执行的文件列表

cmdlist="/bin/bash /bin/ls /bin/cp /bin/mkdir /bin/mv /bin/rm /bin/rmdir /usr/bin/ssh /usr/bin/id"

# chroot路径

chroot\_path="/var/chroot"

# 判断依赖的库文件

lib\_1=`ldd $cmdlist | awk '{ print $1 }' | grep "/lib" | sort | uniq`

lib\_2=`ldd $cmdlist | awk '{ print $3 }' | grep "/lib" | sort | uniq`

# 复制命令文件

for i in $cmdlist

do

if [ ! -d `dirname ${chroot\_path}$i` ];then

mkdir -p `dirname ${chroot\_path}$i`

fi

cp -a $i ${chroot\_path}$i && echo "$i done"

done

# 复制依赖的库文件(因为是i386，所以是lib，如果是x86\_64，则是lib64,)

for j in $lib\_1

do

if [ ! -d `dirname ${chroot\_path}$j` ];then

mkdir -p `dirname ${chroot\_path}$j`

fi

cp -f $j ${chroot\_path}$j && echo "$j done"

done

for k in $lib\_2

do

if [ ! -d `dirname ${chroot\_path}$k` ];then

mkdir -p `dirname ${chroot\_path}$k`

fi

cp -f $k ${chroot\_path}$k && echo "$k done"

done

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33

至此，我们已经能够用test用户登陆了，并且用户被锁定在/var/chroot/home/test目录下，还可以使用ls mkdir等命令，但是此时ssh命令却无法使用。

ssh test@10.60.80.100

bash-4.1$ mkdir test

bash-4.1$ ls

a test

bash-4.1$ ssh root@10.60.80.101

You don't exist, go away!

* 1
* 2
* 3
* 4
* 5
* 6

报这个错意味着系统无法通过用户数据库(可能的类型有 /etcp/passwd, /etc/group, /etc/shadow,/etc/gshadow)校验用户名是否正确。   
通常的解决方法是复制对应的文件到chroot目录中。   
对于ssh来说，可以复制/lib64/libnss\_\* ,这些文件复制到chroot目录下的对应目录.   
解决方法:

cp /lib64/libnss\_\* lib64/.

* 1

另外通过以上我们可以看到test用户登陆后默认为”bash-4.1$”，这是由于环境配置文件导致，我们可以通过以下更改：

cp /etc/bashrc /var/chroot/etc/

cp /home/xxx/.bashrc /var/chroot/home/test/

cp /home/xxx/.bash\_profile /var/chroot/home/test/

* 1
* 2
* 3

## 测试

1.通过scp传输数据

scp login\_union.py test@10.60.80.100:/home/test

test@10.10.65.100's password:

/etc/bashrc: line 65: id: command not found

/etc/bashrc: line 65: id: command not found

bash: scp: command not found

lost connection

* 1
* 2
* 3
* 4
* 5
* 6

由于我们没有安装id命令，因此显示”/etc/bashrc: line 65: id: command not found”，在脚本中添加/usr/bin/id即可   
由于跳板机没有安装scp命令，因此从远程主机传输数据时提示”bash: scp: command not found”，若要解决此问题只需在脚本中添加”/usr/bin/scp”命令即可

2.通过sftp传输数据   
若需要允许使用sftp，只需修改以下即可

vim /etc/ssh/sshd\_config

Subsystem sftp internal-sftp

## shell

## 网络

### netstat

查看连接你服务器 top10 用户端的 IP 地址：

netstat -nat | awk '{print $5}' | awk -F ':' '{print $1}' | sort | uniq -c | sort -rn | head -n 10

## vim使用插件

### 查看

### vim源文件

#### 下载

git clone https://github.com/vim/vim.git

cd vim/src

make

make install

#### 如果缺少依赖库

Apt install make build-essential libncurses5-dev libncurses5w-dev –fix-missing

**配置层次结构**

我曾经错误地认为，Vim仅从〜/ .vimrc文件中读取其所有设置和脚本。阅读各种“dotfiles”的代码库更坚定了我的这一看法。通常人们觉得只通过一个.vimrc文件来控制编辑器的各个方面是一种危险的做法。这些庞大的配置文件有时被称为“vim发行版”。

实际上，Vim的结构非常整洁，.vimrc只是多个配置文件中的其中一个而已。其实，你可以让Vim告诉你究竟加载了哪些脚本。试试看：任意编辑计算机上的某个源代码文件。加载后，运行如下命令：

:names

花点时间读完整个清单。猜猜看这些脚本可能会做些什么，并记下它们所在的目录。

清单比你预期的要长吗？如果你安装了大量插件的话，那么编辑器需要做大量工作。你可以通过以下命令检查是什么导致编辑器的速度变慢，然后再看看它创建的start.log：

vim --startuptime start.log name- of-your- file

为了比较起见，下面我们看看如果没有这些配置，Vim的启动速度有多快：

vim --clean --startuptime clean.log name- of-your- file

为了确定启动时或加载缓冲区时会运行哪些脚本，Vim会遍历“runtimepath”。该设置是一组以逗号分隔的目录列表，各个目录的结构都是一致的。Vim会检查每个目录的结构，找到需要运行的脚本，并按照目录在列表中的顺序一一处理。

运行以下命令就可以检查系统上的runtimepath：

: setruntimepath

在我的系统上，runtimepath默认包含以下目录。并非所有这些都必须出现在文件系统中，但如果存在就会被使用。

* ~/.vim 主目录，保存个人偏好的文件。
* /usr/local/share/vim/vimfiles 系统范围的Vim目录，保存由系统管理员决定的文件。
* /usr/local/share/vim/vim81 即$VIMRUNTIME，保存与Vim一起分发的文件。
* /usr/local/share/vim/vimfiles/after 系统范围Vim目录中的“after”目录。系统管理员可以利用该目录来覆盖默认设置，或添加新的设置。
* ~/.vim/after 主目录中的“after”目录。可以利用该目录用个人偏好覆盖默认设置或系统设置，或添加新的设置。

这些目录会按照顺序处理，所以要说“after”目录有什么特别的话，那就是它位于列表末尾。实际上“after”并没有什么特别之处。

在处理每个目录时，Vim都会查找具有特定名称的子文件夹。如果想了解更多这方面的信息，请参阅:help runtimepath。下面我们只挑部分进行说明。

* plugin/ 编辑任何类型的文件都会自动加载的Vim脚本文件，称为“全局插件”。
* autoload/ （不要与“插件”相混淆。）自动加载中的脚本包含仅在其他脚本请求时加载的函数。
* ftdetect/ 用于检测文件类型的脚本。可以根据文件扩展名、位置或内部文件内容决定文件类型。
* ftplugin/ 编辑已知类型的文件时执行的脚本。
* compiler/ 定义如何运行各种编译器或格式化工具，以及如何解析其输出。可以在多个ftplugins之间共享。且不会自动执行，必须通过 :compiler 调用。
* pack/ Vim 8原生软件包的目录，它采用了“Pathogen”格式的包管理。原生的包管理系统不需要任何第三方代码。

最后，通用的编辑器设置都会放到~/.vimrc中。你可以通过它来设置用于覆盖特定文件类型的默认值。有关.vimrc设置的全面讲解，请运行 :options。

**第三方插件**

在Vim中，插件只是脚本，必须放在runtimepath中的正确位置才能执行。从概念上讲，插件的安装非常简单：只需下载文件。问题在于，很难删除或更新某些插件，因为它们的子目录加入到了runtimepath中，很难判断哪个插件负责哪些文件。

为了满足这种需求，网上出现了很多插件管理器。最早在2003年就出现了Vim.org插件仓库。然而，直到2008年左右，插件管理器的概念才真正流行起来。

这些工具在Vim的runtimepath中添加了单独的查检目录，并会为插件文档编译帮助标签。大多数插件管理器还可以从网上安装和更新插件代码，有的还支持并行更新，或者显示彩色的进度条。

以下是按时间顺序整理的插件管理器。我按照每个插件最早和最新版本进行了排序，如果找不到官方的发行版本，则根据最早和最后的提交日期排序。

查看信息

ip -a

邮件设置

iptables

调整防火墙

如果使用 iptables 过滤与系统的连接，则需要打开 HTTP (80) 和 HTTPS (443) 端口。

通过发出以下命令打开必要的端口：

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

/var/wwwroot可以访问

https://www.sundayle.com/s-nail/

不用自建的postfix,配置s-nail使用QQ邮箱服务器来发送邮件

安装

apt-get install s-nail

配置

vim /etc/s-nail.rc

set from="shaopengxx@qq.com"

set smtp="smtps://smtp.qq.com:465"

set smtp-auth-user="shaopengxx@qq.com"

set smtp-auth-password="lvyuzhxxxxbcab" #不是qq密码。而是授权码

set smtp-auth=login

yum install mailx

vim /etc/mail.rc

set from="shaopengxx@qq.com"

set smtp="smtps://smtp.qq.com:465"

set smtp-auth-user="shaopengxx@qq.com"

set smtp-auth-password="lvyuzhxxxxbcab" #不是qq密码。而是授权码

set smtp-auth=login

set ssl-verify=ignore

set nss-config-dir=/etc/pki/nssdb

测试

echo "this is test" | s-nail -s "test email" shaopengxx@qq.com