Linux For Personal Work

基准：

1，

1.1刷u-boot，装openwrt、android

1.2支持DD、VMDK格式的虚拟机软件。Qemu-img convert ，VBoxManage

2.LFS、BFS、发行版、rolling release

BLFS：mykali\_rolling功能全化的包[向centos看齐]、my\_kali liveCD化

Debian系：deepin、minit、kali  
装kali后首要安装包：bash-completion  
笔记本特有安装包：快捷键支持包、fancontrol

Fedora系：sonwbird NST Amahi;

Openwrt系：pand、LEDE

Automotive grade linux (AGL)、android、

FreeBSD10/macos、

3，协同

协同硬件

server：散热好、台式机元件、硬盘位多，dell M4800  
workstation：轻薄本、续航长、外壳金属，xps15、macpro（>=2014）  
phone/pad: M2，ios，Lineage OS + F-Droid App Store；phonenix OS ，remix OS

协同软件

1.支持miracast/airplay投屏

2.远程桌面

支持teamviewer远程桌面

数据库UI mysql workbench

VPN客户端anyconnect VPN公私皆可，思科。  
1）shadowsocks，goagent  
2）全局上网，PAC上网模式  
3）客户端——基于http的socks5——服务器端  
4）kcp-server；kcptunel

VPS

支持文件协同编辑Libreoffice 、Dia、

Nextclod开源云

移民自由的软件通用

《自由软件，自由社会》3th

Aria2 -c 断点续传，metalink3.0、bittorrent、sftp、https下载。

Transmission下载种子

firefox：Flash Video Download v16.0.12或……

标准

APUE CCNA

《LFS》

OSTC

LDAP统一认证相关课程

LFCS认证

Benchmark——深圳市金城保密技术有限公司2010王学峰

1. FAT文件名、路径、内容恢复。
2. 按正则式搜索字符来匹配：\*+！//
3. U盘序列号、芯片生产厂家、产品系列、型号、序列号
4. 硬件信息读取：OS及版本、CPU序列号、主板序列号、硬盘序列号、磁盘共享信息、网卡MAC、IP、网关
5. 易失/运行信息读取：进程名、进程路径、ID  
    端口、对应进程ID、进程名称、端口的协议  
    服务名、显示名、当前状态、启动类型
6. 软件信息：软件名、版本、安装路径、是否过期  
    文件管理器、打开文件的记录、文件全路径、访问时间  
    OUTLOOK邮箱软件的加密邮件读取  
    密码复杂性要求、密码长度最小值、密码最长留存期  
    本地安全策略、ActiveX安全策略、系统日志
7. 数据恢复技术：恢复上网记录、记录格式、存？  
    恢复使用USB设备的记录、记录格式、主机的USB使用记录存？  
    文件名称、路径、大小、恢复程度

参考

《鸟哥的私房菜 综合版》

《Unix and Linux System Administrator handbook 4th》或《Linux Administrator Handbook 4th》

Linux.Toy

CBT Nuggest 课程

目录

目录 I

第零章 编辑工具 1

0节 1

git 1

第一节 4

spacemacs的基本编辑用法之evil 4

spacemacs的配置 7

操作痕迹 12

第一篇 动态语言 13

Bash概述 13

0.1 bash脚本 13

0.2空格的说明： 14

0.3常用的重要shell可执行文件语句： 14

第一章 标识符 16

第一节 符号 保留字 16

1.1变量 16

1.2 关键字 18

第二节 指令组、子程序保留字 19

2.1函数、进程与文件 19

2.2 IO是返回值么 21

2.3 多进程信息IO(与同步) 21

第二章 正则表达式 22

2.1 grep 22

2.2 ？shell 的模式匹配 22

第三章 程序结构 24

第一节 代码块与数据传递 24

第二节 条件结构与流程控制 25

第三节 进程间信号通信 27

附录 程序功能分类与路径 27

服务程序路径 27

服务型程序特点 28

热加载程序功能 28

第四章 函数 29

第五章 调试 32

Shell 脚本的追踪与调试 32

除错的大概过程： 32

附录1 处理文本文件的idiom algorithms 32

第二篇 信息工具 33

第一节 文字编码、字体与图像 33

语言书写字符（文字）编码 33

颜色空间 34

图形图像编码 34

声音编码 36

第二节 文件访问控制 36

第三节 文件关联 38

例1 用于解决播放器问题 41

第四节 系统加固 42

一、多磁盘卷 42

二、镜像备份 文件压缩 技术 43

附2.智能设备ROM数据提取 45

附1.可启动硬盘镜像的脚本。 45

附3. PC机的两种启动方式 47

三、文件属性清洗 49

四、数据加密 验证 技术 50

附录1. 局域网部署Windows 系统 52

AWD环境下系统加固 54

Linux上网基础 54

一、Linux无线栈 54

二、Nmap 路由排查 55

DNS 56

第三篇 移动端生存 57

IM 57

取证 58

第零章 编辑工具

0节

git

1参《ProGit 2rd 中译本》

《Pro Git》《日常使用git的19个建议》

https://git.oschina.net/，https://bitbucket.org，

gitExtensions，tortoisegit；gitblit

Mysysgit，githug

分布式版本控制系统，与集中化版本控制系统(cvs,subversion,perforce)的概念意义不同。本地版本控制系统(e.g. Rcs of macos)。

配置文件。每次提交都要使用<用户名称，email地址>信息对，且该信息随该次提交保存到仓库中。Git config --system，全局配置/etc/.gitconfig；git config --global，用户全局配置~/.gitconfig；项目配置，工作目录/.git/config，git config --local。查看配置git config --list

本地仓库，即git目录。用git clone时为 .git/，用git clone --bare时选定目录即本地中心仓库。仓库repository，亦称主拷贝master copy，是软件版本及相关记录的存储地。非本地仓库，即远程仓库时，git支持git协议（git://）、超文本协议http(s)://、ssh协议（用户名@服务器名:/path/to/repo）

Git reset ，用仓库中的版本来覆盖：

A，git reset -hard {{某次提交的hash}}，工作目录回退到一个特定的历史版本，其之后提交却不暂存和识别（因无）。

B，git reset -soft {{某次提交的hash}},回滚工作目录，但结点后的更改放入暂存区。E.g.合并最近的三次提交。可视为综合版

C，git reset {{某次提交的hash}}，回滚，节点后的更改放入未暂存状态。即不改动工作目录，可视为重新提交的一种方法。

Git checkout {{某次提交的hash}} [文件名]，签出文件为暂存状态。

从标为已跟踪的文件或从工作目录中提取当前的快照，快照放入暂存目录或提交到本地仓库。已提交的快照表明其已正确地存入本地数据库中，故不会丢失；提交之后对工作目录的任何操作都可以回退为最近的快照。Git以提交数据的SHA-1校验和作为唯一标识索引。

修订已有提交，git commit --amend。把暂存区合并到最近一次的提交，并更新其commit，打开默认编辑器并编写提交信息。

Revert，用一个新的commit来完成回退功能。

每文件在git内有三种状态｛已修改modified，已暂存staged，已提交committed，｝。已修改的文件表该文件已被修改但未暂存或提交，已暂存表已修改且已归入下次的提交清单中，已提交表新快照已正确地存入到本地数据库中。

用户项目所在的目录称工作目录，快照暂存时放快照的目录称暂存目录。工作目录下的文件有两种状态｛已跟踪，未跟踪｝。所有文件缺省都为已跟踪的，可用.gitignore文件指定要忽略/不跟踪的文件(#为注释行，glob式正则，！表取非)。

git blame 某行——>其作者和提交ID，b B，b e。Git log，查看文件状态。

Git status查看工作目录下各文件当前的状态。不能显示已提交的文件，能查看modified、staged状态的文件。Git stash [list，pop]

Git log 查看已提交的文件。

squash, 合并两个commit为一个commit，提交到远程/push

Ssh login using keys，-P选项。Githug、mysysgit、gitlab

2源码依赖libiconv，zlib，openssl，curl，expat。

选项： -p

Branch，merge归并 把一个分支的版本合并到另一个分支上。基于分支的开发模式：1，长期分支 2，特性分支/短期分支 3，远程分支 remote clone，origin，用git fetch同步。跟踪分支。-p

查看本分支与另一分支的某文件差异： git diff 分支 某文件

Merge，次分支合并到目的分支，两分支的末端快照与两分支的共同祖先间执行三方合并，得一新快照，新快照是由多个父提交组成的。Merge时发现同一文件被不同分支修改则提示并用merge tool解决。次分支在合并后可以关闭并删除此分支。

查看尚未合并的变更：git log --no-merges mater.. ； git show --no-merges master..

查看已合并分支：git branch --merged

合并分支而不删除分支： git merge --no-ff ?分支

Pull,push,sync，update 。Check out ，check in。

Conflict，冲突，协同开发时，多个并发者对同一文件各自进行了改动，并check in到？内，以致提交时双方无包含关系。冲突由领导开发者手动消除。

Push，推送。本地分支只能显示与远程仓库同步。可把不愿分享的内容放到私人分支上，须与别人协作的内容push到公开分支且对远程仓库有写入权限。

Rebase。变更基线，衍合。只对自己已完成的代码，且尚未推送给别人的本地修改执行变基操作；从不对已推送的提交执行变基操作；若工作区中存在尚未提交到仓库的变更，则勿用rebase。

交互式使用变更基线git rebase -i {{某个提交序列号}}，其自动打开一编辑器。

Git pull -rebase，自己的更改尚未push到远程时，已有人先Push到远程仓库一份提交时，用该命令来先衍合本地代码到最新版的主仓库；然后排除错误，最后提交新的并更新远程主仓库。故一般以远程主仓库做基线进行变基线。

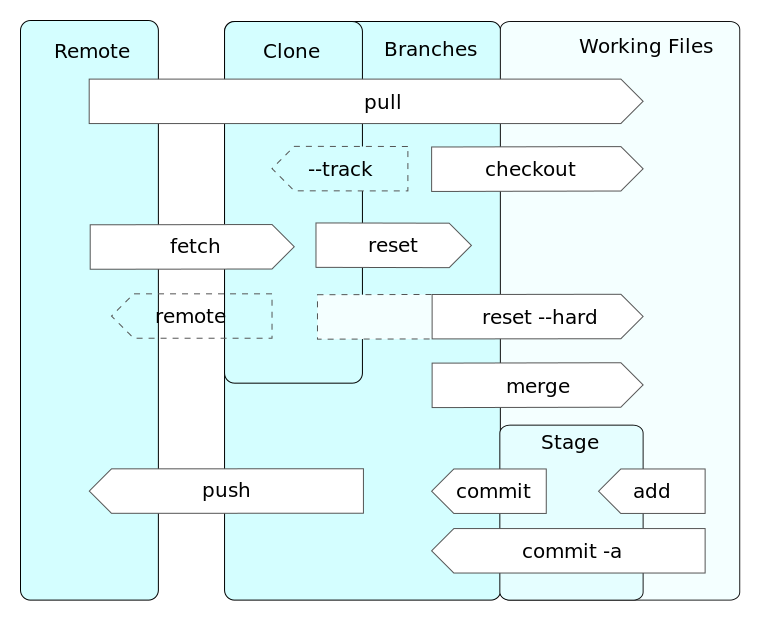
git bisect，二分查找找出错误的版本。

辨析：

1，clone和fetch

2，reset、merge、checkout

3，add、commit，push；remote和pull



magit

1，spc g i ,git init ,新建仓库，不清理内容；

2，spc g s ，git status，

3，spc s l ，

4，spc u s ， stage

5， c 、编辑提交注释、C-c 、C-c ，提交

例

1，magit插件

<spc>git s ;;大部分git命令

<spc> git init ;;

<spc> git status ;;

<spc> git commit ;;

<spc> git push ;;

<spc> crtl-c ;;提交

<spc> git checkout;;

<spc> git list log;;

当前buffer的时光轴timemachine

2，标签语言

doxygen代码文档系统。idea：利用doxygen的注释命令编写详细设计文档。

将JavaDoc风格（？具体）的源码注释文字转换为unixmanpage、HTML、CHM、pdf、RTF格式的文档。跨3平台。

注：doxygen的文档编码转码用iconv。 graphviz拓扑可视化？

1，编写文档元数据配置文件。包括输出设置，项目设置。doxygen -g myconf\_file生成配置文件，然后修改项目配置文件。

2编注释。包括行间注释和行内注释。

3，生成文档。  
doxygen my??\_file  
markdown

第一节

1，模式：与手机模式、驾驶模式等相同。模式表示的是系统当前的能力范围；

2，<project，布局，buffer>等整体，面向应用的概念，编辑时用户关心的功能范围；

3，python-mode、···，意在沟通模式与project的鸿沟，转通用为定制，服务当前用户。

spacemacs的基本编辑用法之evil

1，Vim风格模式对应与切换快捷键。Dotspacemacs-editing-style变量为[vim，emacs，hybrid]，缺省为hybrid即混合风格。

|  |  |  |  |
| --- | --- | --- | --- |
| 窗口标签颜色 | 对应状态 | 对应到vim | 功能 |
| 橙黄 | Normal | Vim命令状态 | Fd回到normal模式，丢弃buffer。SPC f [s S]保存/所有 |
| 青绿 | Insert | Vim插入状态 |  |
| 灰白 | Visual |  |  |
| 红 | 用iedit导航和替换 | 多光标编辑 <spc> / ;;  <spc> s e f/p insert |  |
| 紫  巧克力 | 移动，buffers内导航  替换 |  | gb，gf，hijk |
| 蓝 | Emacs或  混合风格 |  | 常规emacs或  插入状态+快捷键可用 |

显示行号，大小写setic、setnoic。

注：multiple curse。1，搜索词高亮，即选中文本；2，<spc> s edit，即进入iedit状态；3，限制范围为File；4，用编辑模式insert、append或Delete、undo。

Major-mode ;;emacs在当前缓冲区/文件的行为，

Minor-mode;; 一项功能

micro-mode ;;

2，文件建立保存丢弃。缺省300char或30S自动保存一次，缺省在cache dir内，即dotsapcemacs-autosave-file-location [original，cache] 原文件目录下或cache路径下。

<spc> buffer ;;

|  |  |  |  |
| --- | --- | --- | --- |
| leader | 一级 | 二级 | 三级 |
| <spc> | buffer | find |  |
| file |  |  |
| help |  |  |
| project |  |  |
| select |  |  |
|  | te**x**t |  |  |
|  | delete |  |  |

Switch move +hijk

save Saveall

rname Rname

Replacestring

define +key

Edit

NORMAL模式：

表格 1spacemacs混合风格与mac与word快捷键

｛ctrl, alt｝+｛u, d, k, f, b, a, e, v, <, >｝.ctrl、alt须hold住后再按键，Esc则按完再按键。M+{!， | }执行shell命令和在？区域执行shell命令。

Kill，可恢复，先mark再kill或copy；delete， 。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | spacemacs | Spacemacs-evil  Normal模式 | Spacemacs-emacs | mac | Word | Win |
|  |  |  | M | Command |  | <Alt> |
|  | <spc> | **:** | Crtl 或alt + |  |  |  |
| 定位 |  | crtl-g |  |  | crtl-g |  |
|  |  | r替换当前字符  R替换多个字符 |  |  |  |  |
| 文件内查找 |  | ？向前/向后搜 |  |  |  | F3,F6 |
|  |  |  |  |  |  |  |
| 查找问替换 |  | :1,$ s/word1/word2/gc  c表交互 |  |  | crtl-h |  |
| 保存 |  | :w [all]  :x == :wq |  |  | crtl-s |  |
| 另存为 |  | :saveas 文件名  :w 文件名 |  |  | crtl-a | F12 |
| 重命名 |  | :file 新名 |  |  | F2 | F2 |
| 选中 |  | v按字符,  V按行,  crtl-v按矩块;后  Shift-i 插入，esc,esc |  |  | shift+方向键 |  |
| 复制 |  | 次数+操作+单位向量  数字+d +w,e,$ | +dk |  |  |  |
| 剪切 |  | x字符  [c切后进插入模式,不推荐] |  |  | crtl-x |  |
| 撤销 |  | u、e， 反撤销U  e!全撤 |  |  | crtl-z |  |
| 粘贴 |  |  |  |  |  |  |
| 复制到粘贴板 |  | 粘贴板清除！ |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 打开 |  | :r 文件名 |  |  | crtl-o |  |
| 新建 |  |  |  |  | crtl-n |  |
| 单窗多文件跳转 |  | :n,N  :files 列出所有打开文件 |  | <spc> b m h/i/j/k |  |  |
| 分窗显示 |  | split |  |  |  |  |
| 多窗切换 |  | crtl-w |  |  | crtl-Tab |  |
| 加密、解密 |  | vim -x  :set key=|\_| 解除 |  |  |  |  |
| 右键菜单 |  | :help key-notation |  |  |  | Shift+F10 |

例：标签文档org-mode \*.org文件。

Han插件(chinese-pyim---->Han layer)。C-x RET c-\

注：

编码：< /etc/sysconfig/i18n，bash的$LANG，GNOME的编码设定，文件编码>。DOS断行符^M$即CR LF，UNIX断行符$即LF；可用dos2unix或unix2dos处理换行符。iconv -f 原编码 -t 新编码 输入文件 [-o 输出文件]。

emacs的编码插件unicad，

编辑模式：

Msword等文本编辑器可视为仅有编辑模式。源码编辑支持

表格 3spacemacs的源码编辑

|  |  |  |  |
| --- | --- | --- | --- |
|  | vim | emacs | VS |
| 风格切换和Leader键 |  |  |  |
| 补全 | crtl-n | ysn alt - / |  |
|  |  | <spc> c l ;;注释当前N行 | crtl-g定位 |
| Buffer ，丢弃 |  |  |  |
| 跳转项 |  |  |  |
| 在专案中找 | == |  |  |
|  |  |  | crtl-h查找替换 |
| 查找文件 |  |  | crtl-s |
| 返回前一缓冲 |  |  | crtl-a |
| 两类分窗，全屏 |  | SPC T F全屏 |  |
| 查找函数 |  |  |  |
| 编译 |  |  |  |
|  | :help vim-script-intro |  |  |
|  |  | <spc> ` shell |  |

1，一个伪装成编辑器的编辑语言，将文本编辑像代码封装到方法中。

2，其它编辑器中，默认开启的是插入模式。Vim编辑多数时间在普通模式，按键像命令一样，执行光标、剪切、粘贴、查找、替换、写宏；鼠标+命令；该模式使用一种语言来操作：有v、adj、n，e.g. caw即change a word，cas即change q statement，c5w即change 5 words，df:即删除直到遇到‘:’为止；<q，qw，@w>来录制宏，vim最佳命令<I，A>、<m，”，’>标签命令，<ctrl+] ctrl+t>查看函数定义和返回，=括号对齐，<ctrl+r，U>重做和撤销。命令模式“：命令”，只需记一部分命令。vimscript

3，vimperator火狐插件。

Vim插件管理器？：vundle插件管理器

?Key bindings为VS

CAD风格

表格 2 vim的CAD风格操作

|  |  |  |
| --- | --- | --- |
|  | vim | cad |
| 移动 | 次数+操作+单位向量  数字+d +w,e,$  x，y d |  |
| 复制 |  |  |
| 将 x至 y 行写入指定文件中 | x，y w 文件 |  |
|  | :!外shell命令。e.g. :!xxd %  :!xxd % -r；crtl-d退回 |  |
| shell中的!! | . |  |

注：vim当前文件名 %

Spacemacs的帮助

<spc> help [descriple，function，key，variable]

spacemacs的配置

1,emacs配置文件

内建程序：内建工具和游戏；m-x 内建程序名 启动，c-g （keyboard-quit）关闭并保存buffer，或c-x k关闭buffer。 内建工具有 dired，即diretory editor，支持先mark后操作；claendar，日历和日记，q退出；eww，网页浏览器，输入网址或关键字（搜索引擎为duckduckgo）。

帮助系统：示例c-h t、参考手册c-h i、辅助工具c-h[a/f，b/k/c/w，v]

用主模式、次模式的概念来更改工作环境。主模式与文件类型相对应，据后缀加载模式，缺省为fundamental-mode，每一buffer用一主模式，主模式下可修改键序；次/微模式与功能特征相对应。每主或次模式名与一函数——同名文件.el相对应。E.g. Text-mode——text-mode.el。

Text mode，用于自然语言、程序语言。标签文本模式 用于带标签的文本indented text mode

主模式有基本模式(fundamental-mode，prog-mode，text-mode，special-mode)、父模式、派生模式。如text-mode、html-mode、都派生自text-mode，c++-mode、c-mode、emacs-lisp-mode、sql-mode派生自prog-mode。Emacs的输出buffer用special-mode，用户勿用。可设定emacs启动时的主模式

C-X h全选 适用于输入多条命令的情况

执行shell：同命令buffer内执行（M-!或M-|）和新buffer内执行（M-x shell开启，c-x k关闭）。其中M-|用选中的区域做命令的stdin。C-X k关闭该shell buffer

region区域：即vim的block，由MARK和point标定。C-spc或c-@标定mark。光标的位置称“POINT”，位于当前字符与其左侧字符之间，标定POINT。C-x c-x使光标在mark和point二者间跳光标。C-w 为kill选定区域；c-x u或c-/或c-\_撤销。M-=统计区域的行数和字数。

mark：

擦除buffer文本：

1，有两种擦除buffer文本的命令：delete和kill。擦除空白（spc ret tab）、单字符的命令不会保存被擦除文本，这类命令称delete；擦除多于1个字符的命令会保存被擦除的文本，这类命令称kill命令。亦即kill时隐含着保存。E.g. M-c 删除空白，即合并两行；DEL/c-d，BS，删除字符；m-spc，m-\，c-x c-o。 M-BS，M-d，ESC 5 BS，ESC 18 DEL / ESC 18 c-d，即为kill。

2，emacs将kill ring存为一个kill-ring的语言变量，用c-h v kill-ring查看当前内容；KILL RING 缺省为60个项/记录/条目，条目为kill ring entry；所有的buffers共用一个kill ring。选定区域后，M-w拷贝到kill ring的一新项。

3，恢复文件，AUTO-SAVE FILE缺省在文件所在目录下，命名为“#原文件名#”，300次击键保存一次。保存文件时，emacs移除AUTO-SAVE FILE；加载文件时，先找回目录下的“# #”文件并提示恢复。

配置：

emacs的V25已集成MELPA格式包的管理模块。

1，排错配置文件   
emacs -nw -Q #nowindow不要GUI环境；Quickstart，裸emacs启动，不加载配置文件；  
--debug-init对应~/.emacs.d/init.el。  
C-x C-c退出emacs

2，查看变量、命令、快捷键的文档  
alt/meta-x+describe-variable=c-h v  
alt/meta-x+describe-function=c-h f  
alt/meta-x+describe-key=c-h k

3，版本号小于23.3的亦可用steve purcell的配置，包较新。Reddit，gooogle Plus论坛，《writting gnu emacs extensions》《emergercy elisp》

Emacs的远程、FTP、剪贴薄功能不太好。

Emacs ref card

<Esc> !、M-!输入shell命令。

Yasnippet插件使用、ultisnips

exwm，emacs作桌面管理器

术语：

delete删除，undo，redo

kill移除，undo或yank召回（现在的回滚/撤销）

Kill-ring 移除环

buffer：为多文件编辑平顺，emacs保持一个独立的存储空间，称buffer，每个编辑文件/任务一个。创建buffer来源有两个：人工所建、emacs自建。

key sequence：key combine

key binding：键序列  
  
功能辅助键：CTRL、^、C-，ALT/meta/option、M-、<Esc> ，对于同一功能的缀字符，二者表明相逆操作，比如C-v向前，M-v向后；C-x接key binding，M-x接命令串。

命令：一个用于交互的lisp函数，由连字符和小写词组成。执行没有key binding的命令——按M-x/Alt-x/ESC-x，然后输入命令名，最后键入回车，若回应no match则表该程序/模块/命令未安装。

input EVENT：

Rime，OpenCC中文繁简互搜和转换

2.Vim配置：

配置：~/.vimrc 。~/.vim/plugin/\*.vim，/usr/share/vim/?。

痕迹：~/.viminfo记录vim的操作信息！

临时文件： .doc的为.asd，.xls的为.asp，vim的为

Vim graphical cheat sheet

spf13\_vim2.0，spacevim0.1，space\_vim0.5

Vundle，插件管理器。neovundle、VAM

3.Spacemacs，插件集成且开箱即用。

自动探测更新包，手动决定或enter键更新，更新有错后可回滚。SPC h SPC。~/.spacemacs总优先加载，~/.spacemacs.d/initel其次，SPC f c R重新加载配置。

1，package/feature，包或功能，可提供一组功能的文件集称包。典型地elpa、melpa或第三方来源的，A，MELPA包。B，local包。C，在线quelpa包。。当某文件包含(provide ‘无后缀文件名)时表该文件提供某一功能。

2，包下载。以下是包下载/自动安装的前提，若不符合则缺省自动移除：A，用户启用的某个层Include了该包且该报没被其它层exclude；B该包在spcmacs至少存在一个已定义<layer>/init-<package>函数；C该包被直接加入’dotspacemacs-additional-packages’内。

3，包加载。其它文件用(require ‘无后缀文件名)直接加载/使用一个包文件或功能文件，加载后该文件的函数、宏、mode可为当前用；其路径则是由emacs的load-path决定，可用(push “/some/path” load-path)来增加路径，用SPC h d v load-path来查看路径。(Load-file “”)不做任何串扩展，亦不使用.elc文件。Load-path是立即加载的，若不存在？

4，包延迟加载。用时加载/延迟加载则用(auto-load ‘somefunc “无后缀文件名”)，义为~。若延迟加载失败则？

2，包间序。

（with-eval-after-load “应该先加载的文件” “文件加载后才能执行的代码”）用于指定某段代码只能在某文件auto-load或require后才能执行并执行。Spacemacs提供了use- package包来方便该功能，可指定某包加载前、加载、加载后的动作及其严格顺序。

3，层。

识别：一个位于spacemacs的层搜索路径内的一个文件夹，其常有layer.el、package.el、func.el、config.el、keybindings.el、[local/]等文件，分别放用到的其它层、本层用到的函、配置层的变量。

执行绪：Use-package函对包按 :init，:config顺序预设、加载和执行。  
层对某个加载的包按其pre-init、init、post-init顺序调用执行。通常的习惯是，拥有包的层才定义其init-<包名>函，其它引用包的层应只定义包的pre-init-<包名>、post-init-<包名>函。  
层自身按layer.el、package.el、func.el、config.el、keybinds.el、[local/]顺序执行。

设计理念：层是spacemacs中启用/禁用配置的单元/单位概念，一个层对应着一组配合紧密的功能，由包和胶水配置代码组成；  
可用configuration-layer/[layer，package]-usedp函数测某个层或包是否已调入当前。一个层可被其它层？使用？调入，即在某层的config.el文件中用configuration-layer/ declare-layer[s] 启用？调入？其它层。或在 .spacemacs文件中用dotspacemacs-cofiguration-layer内启用的层，其包才会存储和保留，见前。用户应考虑的是层。

例1：配置python。Jedi、ycmd在处理中文路径时的缺陷。

#pip install anaconda\_mode

编辑.sapcemacs，加入python层。

1，dotfile中添加对python的支持

2，按python layer [按spc h spc，输入python 回车]安装相关包：  
pip install --upgrade “jedi>=0.9.0”“json-rpc>=1.8.1”“service-factory>=0.1.5”  
jedi 0.9 存在莫名其妙的bug: lambda key error，can Anaconda-mode主页。

3，测试。在相关函数上安spc m g g，再按 spc b p返回

例2：配置基于ycmd的c-c++mode

0，下载并编译ycmd或YouCompleteMe。

1,配置ycmd层。先在.spacemacs内启用ycmd层，重新加载配置文件下载对应包。然后在.emacs.d/layers/+tools/ycmd/package.el内配ycmd的程序的参，参数种类即emacs-ycmd的参种类，主要有ycmd模块路径、自定义的ycmd配置文件路径、启动超时限制、模式关联。

2，检查company包配置。先在.spacemacs内启用auto-company层以下载该层和其包。然后检查其下company包的配置，该包正常工作需要的前置条件有？，；该包使用emacs-ycmd包？

3，检查auto-company层配置。拥有company包，故其内定义一个auto-completion/ init - company 函，其完成 ；该层使用emacs-ycmd包？

4，配置.spacemacs以启用用到的其它层，如syntax-check、。

项目类功能的设计思想和功能定制

打开文件，判断项目根目录——某个major模式，此时项目=buffer内容——insert时调某个minormode

Spacemacs默认使用flycheck检查（缺省仅在保存时执行一次 SPC e xx或 spc t s ）、gnuglobal后端引擎、

company-mode前端、yasnippet代码模版 alt+/、<spc> j =;;自动对齐

例1：

1，cmake代码配置系统Cmake-mode   
识别：C++ ,Java;  
输出：{dsp,sln,vcproj},Makefile  
 Graphviz可用的全局相依图；

autoscan/aclocal/autoconf/automake ：  
1,cd path-to-src/  
 autoscan-->configure.scan<=rename=>configure.in  
2,News,README,changlog,Authors,{Makefile.am}  
3,cp /usr/share/automake-1.x/{depcomp,compile} ./  
 makedepend  
4,aclocal;autoconf;automake -a;  
5, ./configure

编译命令 SPC c C，SPC c k，SPC c r重编译，SPC c d调试。cc-mode。

GUD：COMAND-x compile/gdb 且gdb-many-windows非nil，则启动gdb，可置gud-tooltip-echo-area非nil 且comand-x gud-tooltip-mode启动增强模式。

|  |  |
| --- | --- |
| 交互区 | 局变/寄存器区 |
| 源码区  =>当前执行行[改源码后不同步] | 该程序的I/O |
| 栈区 | 断点/线程区 |

？区： C-x C-a C-b ;;设置断点gud-break

``` c-s;;单步 gud-step，入函

```c-n;;单步gud-next，跳函

```c-i;;单指令gud-stepi

```c-p;;印出point的表达式gud-print

```c-r;;运行至下一断点/检查点

```c-d;;删除当前行断点gud-remove

```c-j;;跳转执行gud-jump

Project 专案，缺省将有 .git/或 .projectfile文件的目录等，是隐式定义的。SPC p xxx

Spacemacs布局和子布局(工作间)，一个布局对应一组buffers，一个布局对应？组子布局。辨析windows和buffer辨析。 缺省布局 SPC l xx，创建项目布局和自定义布局。SPC l w 号 创建一新工作间。

缺省NeoTree项目导航。

操作痕迹

错误诊断：1，屏幕输出；2，日志记录。

日志系统

1，后台支持，klogd记录内核。syslogd记录登陆系统、登陆终端、等登入服务的信息，/etc/syslog.conf。logrorate日志轮替。

2，文件约定 /var/log/[dmesg系统启动信息，message出错信息，lastlog最近一次帐号登陆，secure 输入u-p，cron定期任务]。

3，信息条目格式<时间，主机名，服务名或库名，内容>

手工分析：

自动分析：logwatch 部分分析，登录日志。

注：syslog.conf的七等级{info，notice，warn，error，crit，alert，emerg}逐渐加重。

Emacs的痕迹

1，dump文件，错误日志文件

2，临时文件命名法。emacs的AUTO-SAVE-FILE表自动保存临时文件。缺省的恢复文件命名为“#原文件名#”，每300次击键保存一次。当用户发出保存文件命令时，emacs移除AUTO-SAVE-FILE；当加载文件时，emacs先找同目录下的“#……#”文件并提示恢复。

3，历史记录存放位置

第一篇 动态语言

Zsh&Bash&javascript&QML&kivy

cheat.sh ？小抄工具

参考：BLFS

my\_kali-rolling功能全化的包（向centos系看齐）

my\_kali liveCD化

kali-rolling、centos的命令行补全包：bash-completion（debian缺省不装）

linux控制台补充包：zhcon—>fbterm

windows的linux环境

|  |  |  |  |
| --- | --- | --- | --- |
| msys2 | msys | shell |  |
| SDK | mingw32/mingw-64 |
|  | posix API | Cygwin |
|  | 应用管理 | pacman |
|  | 版本控制 | git |

Bash概述

先得会搜，搜了得会改，改了得会变。

一个已经发生过的操作称一个shell事件，亦即history中的一行<事件号，事件名>，存于$HISTFILE。history -c清空， -anrw。

0.1 bash脚本

Shell 脚本是包含命令的可执行文本文件。一个bash shell程序由unix/linux命令、bash shell命令、程序结构控制语句和注释组成。创建并编写shell脚本的顺序如下：

1."#!"幻数（音sharp bang），告知内核用哪个程序来翻译。必须位于第一行，对于bash，通常用“#!/bin/bash”  
#!/usr/bin/env bash

2.注释(comments)，行注释以#开头且脚本首行必以#!path-to-script开头而不能是注释。# 行注释，应写功能、版本、历史记录、用到的特殊指令、定义的变量、联系方式

3.编写shell脚本

4.使脚本可执行 chmod +x scriptname

通常情况是：用户在编辑器中创建一个脚本，保存文件后，打开脚本的执行权限，执行之。

ie, 装机后几件事：显卡驱动、网卡驱动、温测风控软件

#/bin/bash

#lm\_sensors V3.1.1包，侦测硬件温度的控制台小程序编写

sensors-detect

sensors

#fancontrolV3.4.0包，守护进程，转速自动；一般用于laptop，因其BIOS/ACPI中无温-转自调节功能。配置文件编写。

运行脚本的方式：

1. 用”.”命令，用脚本名做参数；
2. 赋给脚本文件可执行属性，直接打开文件名；

0.2空格的说明：

1.等号赋值两边不能有空格

2.命令与选项之间需要空格 ；命令“[” “]”各自的两边须空格；(尽管某些命令的参数能连一起，例如ps)。

3.管道两边可有可无空格

0.3常用的重要shell可执行文件语句：

命令 选项 &文件描述符与console脱绑定

命令 选项 文件描述符[0~9] 命令的I重定向符> | 命令的O重定向符<

文件描述符出现在重定向符右边时须用&n式引用

命令 选项 <<结束标记符，缺省时为文件尾

#文件描述符 出现在重定向符右边时须用&n式引用。文件描述符(?每进程一个表?)，即OS内概念，用一个唯一数字（通常为0-9）来表示一个文件。常用的文件描述符如下：  
用<或>时，等价0< 或 1>（下面会详细介绍）。

#文件描述符名称常用缩写默认值

#0 标准输入stdin键盘

#1标准输出stdout屏幕

#2标准错误输出stderr屏幕

#here document即时文件，即 <<结束标记符，<结束标记符，等。

例：

cmd >&n cmd的输出送到文件描述符n；cmd <&n 文件描述符n做cmd的输入；cmd m<&n m来自文件描述各个n  
cmd m>&n cmd的文件符m内信息重定向到文件描述符n

cmd >&- 关闭cmd的标准输出；cmd <&- 关闭标准输入  
cmd <&n- 移动输入文件描述符n而非复制它。（需要解释）？  
cmd >&n- 移动输出文件描述符 n而非复制它。（需要解释）？  
注意： >&实际上复制了文件描述符，这使得cmd > file 2>&1与cmd 2>&1 >file的效果不一样。？

流式IO重导向>，>>，<，<<，:>， &>，2&> ，2<>>&，>&2

cmd <> file：以读写模式把文件file重定向到输入，文件file不会被破坏。

cmd >| file：功能同>，但复盖file文件（即便设置了noclobber）。

目前仅在csh中仍沿用>!实现这一功能。

: > filename  清空"filename".# 若存在, 则创建一个0长度的文件(与'touch'的效果相同？文件不存在与空文件与空白文件).

即时文件

\* cmd << text

从命令行读取输入，直到一个与text相同的行结束。除非使用引号把输入括起来，此模式将对输入内容进行shell变量替换。如果使用<<- ，则会忽略接下来输入行首的tab，结束行也可以是一堆tab再加上一个与text相同的内容，可以参考後面的例子。

\* cmd <<< word

把word（而不是文件word）重定向为行输入。

| 管道 (pipeline)

UNIX 类系统基础观念。上个指令的标准输出引导至下个指令的标准输入。善用管道精简 script 。例：who | wc -l

xargs ,从stdin生成后面指令的输入。当有不支持pipe的指令出现时用作其前缀，ie，find /sbin -perm+7000 | xargs ls -l。

1读取用户输入：

read -p,-t,变量名；定义并读取。

Select

3， 文件查找命令：

locate,与命令相关的所有文件。find,which: 安装路径，任何文件，命令行程序。

type -tpa 命令名，分析该命令的类型{buildin ,file(即外部),alias}。

Disktype 查看镜像类型。

&，jobs -r,

3用于测试文件特定属性的选项：编写脚本的时候常常需要一些特定文件，这些文件需要拥有特定的权限，这时候就需要对文件的属性进行测试（详见附表操作符）。

shell中的常用选项：在脚本中指定命令行选项时，最好使其命名规则与LINUX一致。下面是一些选项及其含义的列表。

3.-a 扩展

4.-c 计数、拷贝

5.-d 目录、设备

6.-e 执行

7.-f 文件名、强制

8.-h 帮助 help或human

9.-i 忽略状态

10.-l 注册文件

11.-o 完整输出

多层目录： -p

递归： -R,-r

权限： -m

2,文件关联ln df ， find按属性找

第一章 标识符

1.1提示符：

主提示符 $

次提示符 > 第一条命令没有输入完整的情况下出现

1.2转义字符或续行符 \

一些保留字符，或者说bash的提示符，在编写命令的时候需要用到\ ，使得shell能够解释该提示符。而有的情况下反过来，可以防止shell解释来保护字符。语句以行为单位，结束符ENTER（CR）。

1.3变量的显示与设置

echo ${变量} 或者 echo $变量，串连接时${}用法可以消歧义。

变量的声明，用“=”将变量与内容连接起来即可，不能有空格。

eg: echo $mydoc

eydoc=here

echo $mydoc

注意：等号两边不能有空格符，变量名只能是英文字母与数字，而且开头字符不能是数字。

若需要使变量需要在其他子进程可用，则需要用export使变量变成(?拷贝还是引用)环境变量。

1.5 若干命令

linux kernel 构造系统内命令行频率排名

1,文件rm/touch mv gzip tar cpio file

3,3，特殊文件读取sh grep 链接器的配置：  
ld 配置文件目录/etc/ld.so.conf/\*.conf  
ldconfig

按周期调用程序并显示其结果：  
cv ,结果存入日志；  
watch,结果不存入日志；  
ie,watch cv -qw

4, more ,sync ,cp,link/unlink,sort

第一节 符号 保留字

1.1变量

变量是在程序运行时包含可更改的存储数据的单元，变量先定义后引用；shell 变量命名规定：

shell中变量的用途：暂存脚本的IO参数，e.g. read从stdin读入参数到其后变量/参数中；  
变量的值即字符串型，分两种串：直接写出的、执行命令结果输出的串；  
脚本定义的变量缺省仅能在本脚步内使用；脚本之间传变量可用export，在父子进程(?脚本，命令?)间导出变量的拷贝，即进程间传值；

1. 变量名不可包含shell的保留字，e.g. ! &space 等；
2. 变量名可为任何字符串；
3. 变量名不能包含space，即shell以空格划分符号名并解释；

注：shell 保留字符：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| space | \*？｛｝ | . | $前缀 | 后缀& | | |
|  | 产生文件名列表 | 当前目录 | 取变量值 | 后台执行命令 | 管道 |
| >< | ‘单引号’ | “” | ‘反引号’ |  | \ |
| 重定向IO |  |  |  |  | 转义符 |

1,交互式的环境变量与脚本式的环境变量集不全同；2，变量预设类型为”串”:，需其它类型则用declare定义。bash只支持整形运算（declare -i num或者 let）、逻辑运算，如果需要更复杂的运算，则得借助bc,awk ,nawk等工具来处理（具体见）。

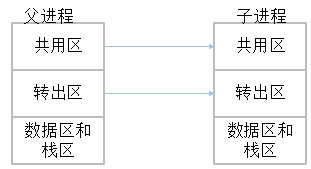
|  |  |  |
| --- | --- | --- |
| declare | -r | 只读变量 |
| +- x | 导出/关闭导出 |
| -i | 声明整型 |
| -a | 声明数组 |

declare -x 变量名=

export 变量1 变量2 变量3 #缺省时列出当前进程自己已导出的变量

#env列出当前进程和其祖先已导出的变量

通常，在命令行上输入的命令都是由相应的进程执行的，即创建子进程来完成命令的功能。然而一个进程在自己的环境中定义的变量是局部变量，仅限于自身范围，不能自动传给其子进程。就是说，子进程只能继承父进程的公共区和导出区中的数据，而数据区和栈区是私有的，不被继承。如图示。应为共享非拷贝



数组项赋值, 名[idx]=值，使用:${名[idx]}。

逻辑运算：运算符和执行路径同C语。

1，环境变量

Shell什么时候产生subshell,shell与subshell间默认传值还是引用？。

/.bash\_profile，source ~/.bash\_profile，

1,$BASH 当前 Shell 的路径字符串

2，当前 Shell 环境缺省路径串。链接：PATH=”$PATH”:/home/bin; ：为连接符，MS下则为 ;

3,，$HISTFILE $HISTSIZE，$PATH

set 查看、指定当前变量，unset取消当前环境中的变量。export 变量名，env 变量；将变量变成环境变量，环境变量即全局变量。

赋值符= 用于给变量赋值，两边禁用space，变量名=变量值。引用变量的值用$变量名，即取变量的值；对变量赋值时格式为？？

注：$IFS　　分割输入参数的分割字符，默认为空格。$HOME　当前用户的根目录路径串。$PS1　　表示第一个系统提示符。$PS2　　表示的二个系统提示符。$PWD　　表示当前工作路径。$EDITOR 表示系统的默认编辑器名称。

2，普通变量

3，可扩展串变量

约定：变量串——通配串——命令串，级别越来越高。则

变量即可扩展双引号"string"

被双引号用括住的内容，将被视为单一字串。它防止通配符扩展，但允许变量扩展。heyyou=homeecho "$heyyou" # We get home

命令才扩展，等同于$(command)

ie,fdv=`date +%F`echo "Today $fdv" 在倒引 号 内的 date +%F 会被视为指令，执行的结果会带入 fdv 变数中。ie, cd /lib/modules/`rname -r`/kernel 。ie, kill -HUO `lsof -t 某文件`杀掉正在使用某文件的所有进程。 Eval，令其参数字串作命令执行。

禁止扩展 单引号'string'

同类嵌套时内层正常须转义。被单引号用括住的内容，将被视为单一字串。在引号内的代表变数的$符号，没有作用，也就是说，他被视为一般符号处理，防止任何变量替换。heyyou=homeecho '$heyyou' # We get $heyyou

非空普通字符扩展 问号?

在文件名扩展(Filename expansion)上扮演的角色是匹配一个任意的字元，但不包含 null 字元。# ls a?a1善用她的特点，可以做比较精确的档名匹配。模式匹配扩展

1.2 关键字

逻辑运算执行路径同C。

第二章 正则表达式

2.1 grep

-E 识别egrep的选项，-i 忽略大小写，-n输出行号，-v 即输出不存在的；

LANG环境变量的设置不同，表明所用编码不同，可能导致匹配的结果的字符顺序不同。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| std::regrex | bash wildcard | grep/egrep | posix/extend | ECMA-262 | javascript |
| 数字符 | [0-9] /[^0-9]  -v [0-9] | [0-9] /[^0-9]  -v [0-9] | [[:digit:]] [[:xdigit:]] |  |  |
| 字母符 | [a-zA-Z]/[^a-zA-Z]/-v[a-zA-Z] | [a-zA-Z]/[^a-zA-Z]/-v[a-zA-Z]/  -i [a-z] | [[:alpha:]]  [[:lower:]]  [[:upper:]] | [::alpha:] |  |
| 非换行符 | 至多一次？ | . 或？ |  |  |  |
| 位于开头结尾的字符 | ^字符前缀  字符后缀$ |  |  |  |  |
| 转义符 | \ |  |  |  |  |
| 位于开头结尾的单词 |  | \<单词前缀  单词后缀\> |  |  |  |
| 字符幂集 |  | \{指数\} |  |  |  |
| 闭包 | \* | \* |  |  |  |
| 正闭包 | + | .\*或+ |  |  |  |
|  |  |  | [:print:] |  |  |
|  |  |  | [[:space:]]  [[:blank:]] |  |  |
|  |  |  | [[:cntrl:]] |  |  |

C++11：<regrex> std::regrex\_match(string\_in,smatch\_result,regrex\_str[,正则标准])  
《正则指引》余晟

或运算符 |， ！一次都没有，@重复限定1次。

2.2 ？shell 的模式匹配

正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。

匹配除换行符以外的任意字符

常用元字符

！表非，@表仅现一次，\*表0或多次，？表至多现一次，+表至少现一次。

|  |  |  |
| --- | --- | --- |
| 正义匹配元字符 | 反义匹配元字符 | 常用限定符 |
| \w 匹配字母或数字或下划线或汉字 | \W 匹配任意不是字母，数字，下划线，汉字的字符 | \* 重复零次或更多次  + 重复一次或更多次  ? 重复零次或一次  {n} 重复n次  {n,} 重复n次或更多次  {n,m} 重复n到m次 |
| \s 匹配任意的空白符 | \S 匹配任意不是空白符的字符 |
| \d 匹配数字 | \D 匹配任意非数字的字符 |
| \b 匹配单词的开始或结束 | \B 匹配不是单词开头或结束的位置 |
| ^ 匹配字符串的开始 | [^x] 匹配除了x以外的任意字符 |
| $ 匹配字符串的结束 |  |

[^aeiou] 匹配除了aeiou这几个字母以外的任意字符

1. 程序结构

第一节 代码块与数据传递

同C语言main函数参数编码方式。

1，脚本的IO:echo read >><<

在脚本程序文件中，”echo 变量名” 将变量值、字符串常量送到标准输出；”read 变量名”将从标准输入读入一行到变量；here运算符”<<”?

2, 脚本命令行参数--位置参数的使用：

Shell脚本同命令 可接受参数，脚本内引用参数格式为$i,作用域为本脚本，读写修饰为只读(read only likly constant)  
➀$0, $1, $2, $3, $4, $5, $6, $7, $8, $9, ${10}, ${11}.....  
个位数的，可直接使用数字，但两位数以上，则必须使用 {} 符号来括住。  
  
*➁*$\* 则是代表所有引用变量的符号。使用时，得视情况加上双引号。  
echo "$\*"。*$\* ，即[$1 $2 $3...],*引用script的执行引用变量，引用参数的算法与一般指令相同，指令本身为0，其后为1，然后依此类推。还有一个与 $\* 具有相同作用的符号，但效用与处理方式略为不同的符号@

$? 状态值 (status variable)

一般来说，UNIX(linux) 进程以执行系统调用exit()来结束的。这个回传值即status值。回传给父进程，用来检查子进程的执行状态。<exit N,$?>  
一般指令程序倘若执行成功，其回传值为 0；失败为 1。

ps: 正常时，系统会回收执行完毕的 PID，然后再次依需要分配使用。故 script 即使临时文件是使用动态档名的写法，如果script 执行完毕后仍不加以清除，会有名称重复的问题。  
~ 如果调用status动作,lsb脚本应该根据其状态返回下面的值:

· 0 程序正在运行,或者服务正常

· 1 程序已死,但/var/run/下的pid文件还存在(或者在别的位置)

2 程序已死,但/var/lock/下的锁文件还存在(或者在别的位置)，？怎么删

3 程序没有运行 ？怎么删

· 4 程序或者服务状态未知

i.e.

符号$$或许可以符合这种需求。它代表当前shell 的 PID。echo "$HOSTNAME, $USER, $MAIL"> ftmp.$$使用它来作为文件名的一部份，可以避免在同一时间，产生相同文件名的覆盖现象。

tar cvfz dfbackup.tar.gz /home/user > /dev/nullecho"$?"$$由于进程的ID是唯一的，所以在同一个时间，不可能有重复性的 PID?。有时，script会需要产生临时文件，用来存放必要的资料。而此script亦有可能在同一时间被使用者们使用。在这种情况下，固定文件名在写法上就显的不可靠。唯有产生动态文件名，才能符合需要。

3，shell函数

变量的时空作用域，函数调用与脚本调用的异同

3.1.4 IO是返回值么

0，read ，select 进行输入。Select 变量 in 列表

1，函数参数 脚本参数

argc==$#==count($\*)

C语言main的argv[argc]==$0+$\*

下标\*、@表的数字集相同；\*表整体引用，即作串引用，@表重划分引用，即同python的带yield的集合且分隔符取|\_|或””。

位置参量与命令行参数

$0 脚本名，按下标访问

$# 参数个数，参数列表码数

$\* $@：

$\* 所有位置参量 功能相当于ls $，不算脚本名的所有实参，即[$1 S2 $3...]  
“$\*” 即 ？？“$1”“$2”“$3”···

$@ 没有加双引号，跟$\*一样，  
 加了双引号,则成一个变量，如"$1 $2 $3"，就是说一整个字符串。而$\*加上引号则变成了一个个单个的变量。  
用set命令重置变量的值。

2，传参返回值与IO

上条命令退出状态码$?，即函数返回用的return [n]的值；

exit [n 缺省为前句的退出码] 退出当前脚本 ??

3.1.5指令组和子程序

函数{}，子进程();

命令块：命令组，管道，脚本；

函数、进程与文件

A，()和{}都是对一串命令进行执行，：

B, ()和{}都是把一串的命令放在括号里面，并且命令之间用;号隔开

C,()最后一个命令可以不用分号，()里的各命令不必和括号有空格，{}最后一个命令要用分号，{}的第一个命令和左括号之间必须要有一个空格

D,()使一串命令在子shell进程内执行，{}则使其在当前shell进程执行。

几个例子：

$ var=test

$ (echo $var;var=notest; echo $var) ###变量var值为notest，此是在子shell中有效

$ echo $var ###父shell中值仍为test

test

或：

$ { echo $var;var=notest; echo $var;} ###注意左括号和var之间要有一个空格

$ echo $var ###父shell中的var变量的值变为了notest

同进程内执行

source shell文件 #似python的exec指令。

当前shell代码块 **{  } 大括号 (Block of code)**

它在当前的 shell 执行，不会产生 subshell。同exec [arg]。大括号也被运用在 "函数" 的功能上。广义地说，单纯只使用大括号时，作用就像是个没有指定名称的函数一般。因此，这样写 script也是相当好的一件事。尤其对输出输入的重导向上，这个做法可精简 script 的复杂度。——即lamda

例1：

# cat ftmp-02

#!/bin/basha=fsh{a=inbc ; echo -e "\n $a \n"}echo $a

#./ftmp-02inbcinbc

例2：

$ { var1=test1;var2=test2;echo $var1>a;echo $var2;} ###输出test1被重定向到文件a中，

test2 ###而test2输出则仍输出到标准输出中。

$ cat a

test1

$ { var1=test1;var2=test2;echo $var1;echo $var2;}>a ###括号内命令的标准输出全部被重定向到文件a中

$ cat a ,test1 ,test2

注：此外，大括号还有另外的用法。1，串集合，如下{xx,yy,zz,...}  
这种大括号的组合，常用在字串的组合上，来看个例子  
mkdir {userA,userB,userC}-{home,bin,data}  
我们得到 userA-home, userA-bin, userA-data, userB-home, userB-bin,userB-data, userC-home, userC-bin,userC-data，这几个目录。这组符号在适用性上相当广泛。能加以善用的话，回报是精简与效率。像下面的例子  
chown root /usr/{ucb/{ex,edit},lib/{ex?.?\*,how\_ex}}  
嵌套用法  
2，${var\_name}==$var\_name，串连接时标准用法可消歧义。

新进程内执行.

sh shell文件==path to/shell文件#脚本在新子进程执行

指定为在子shell执行**(   ) 指令群组 (command group)**  
这种用法称为指令群组；除了上述的指令群组，括号也用在 array 变数的定义上；另外也应用在其他可能需要加上escape字元才能使用的场合，如运算式。指令群组特性，shell产生 subshell来执行这组指令，故在其中所定义的变量仅作用于指令群组本身。

例子1，(cd ~ ; vcgh=`pwd` ;echo $vcgh)，

例子2，# cat ftmp-01#!/bin/basha=fsh(a=incg ; echo -e "\n $a \n")echo $a#./ftmp-01incgfsh

**((  ))**  
与 let 指令等价，用于算数运算；bash内建，而let指令非内建。

#!/bin/bash(( a = 10 ))echo -e "inital value, a = $a\n"(( a++))echo "after a++, a = $a"

第二节 条件结构与流程控制

4，顺序结构

Null命令“：”,一个内置的shell命令，不做任何工作，只返回状态0。相当于一句空话，否则报错，==exit0？。示例如下：

name=tom

If grep "$name" databasefile >& /dev/null

Then

:

Else

Echo "$1 not found in databasefile"

Exit 1

同行多句时 用; 分号 (Command separator)  
在 shell 中，担任"连续指令"功能的符号就是"分号"。譬如以下的例子：cd ~/backup ; mkdir startup ;cp ~/.\* startup/.  
  
  
;; 连续分号 (Terminator)  
专用在 case 的选项，担任 Terminator 的角色。  
case "$fop" inhelp) echo "Usage: Command -help -version filename";;version) echo "version 0.1" ;;esac

1前置条件与返回状态值

shell 脚本的循环和条件两类控制结构，都须前值条件。

linux命令结束时必返回状态值(成功为0)，控制结构以测试命令的返回值做前置条件。而对于表达式、字符串、整数等不是命令的标识符来说不存在返回值一说，故用test命令判定两个表达式后返回一值，语法为：

test value1 -optvalue2

注：test的等价符  
[[     ]] -----？空格

与 [] 符号，作用相同，但她允许在其中直接使用 || 与&& 逻辑等符号。

#!/bin/bashread akif [[ $ak > 5 || $ak< 9 ]]thenecho $akfi  
[   ] 中括号

常出现在流程控制中，括住判断式。if [ "$?" != 0 ]thenecho "Executes error"exit1fi

另：这个符号在正则表达式中担任类似 "范围" 或 "集合" 的角色rm -r 200[1234]；上例，代表删除 2001, 2002, 2003, 2004 等目录的意思。

①test string1=/!=string2 字符串比较还有 -z 长度为0，即空则真，-n码不为0则真，②文件判定常用:无value1，选项有 -f file file存在且为普通文件则真，-d file file为目录文件则真，-e,文件存在；-h file file为符号链接则真，-c/b file file为字符设备/块设备文件则真，-s file file为空？则真，-r/w/x file file权限为只读/写/执行则真。  
③数值判定， -gt大于则真，-lt小于则真，-ge/le大于等于/小于等于则真，-eq/ne相等/不相等则真.

④用$? 查看上一条命令的执行状态

2.1,if分支结构

前置条件为命令、命令的&& || !表达式；

if 前置条件真 then

命令块

[else][elseif then]

fi

多分支结构case

Case变量 in

值1）

命令（命令组）

；；

值2）

命令（命令组）

；；

\*）

命令（命令组）

；；

esac

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

#!/bin/bash

#统计某串在文件中的频次

#file word\_frequency.sh

count=0

look4=$1

file=$2

for word in `cat $file`

do

if [ $word ==“$look4” ] ; then

((count++))

fi

done

echo $count

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

2.2,while/until for/select循环结构

前置结构分别为 while/until 前置条件 ,for值in 值列表；

循环主体为do

代码块

done  
结束循环可用continue继续下次、break跳出；

For变量in词表

Do

命令（命令组）

Done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

While命令

Do

命令（命令组）

Done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Until命令

Do

命令（命令组）

Done

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

break/continue

BASH 中同样有 break 语句和 continue 语句，作用和用法和 C 语言中相同，break 语句可以让程序流程从当前循环体中完全跳出，而 continue 语句可以跳过当次循环的剩余部分并直接进入下一次循环。

第三节 进程间信号通信

1，Trap 可执行文件做参数 信号代码；？发出者？，若接受到指定信号代码则执行 指定可执行文件。信号代码如下：0 义为发信号的程序正常终止，1为发信号的程序？挂起终端，2为用户按下CTRL+C或DELETE键，3 为用户按下CTRL+L或CTRL+\ 退出键,15结束为kill命令本程序发出信号，24为按下CTRL+Z发出 终止程序运行 不能设置trap？？，

2，fork bomb

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

windows 下

:s

start “” %0

goto s

linux 下

:() { :|:& }; :

#定义一函数“:”，

#函数体为管道命令“：|：&”，

#管道即fork出2个子进程，且递归调用。

#“:”为调用函数“:”

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

2，Trap内部实现即对应“访管指令”

trap [响应命令] [信号码列表] #信号码取0~16

3，#nohup，忽略发给其命令的SIGHUP信号(终止信号)。或#ssh -Nf

进程组ID，进程组一个ID；会话期首进程ID即会话期ID。控制端与控制进程。

在同一进程组中某进程退出时为确保不会有孤儿进程产生而规定的发SIGHUP、SIGCONT信号规则 导致了控制终端退出后后台程序也退出，且用“&”的进程也避免不了的现象。

多进程信息IO(与同步)

进程

$$所在进程的进程号。

$-当前shell选项

wait [pid缺省为所有子进程] #缺省表所有子进程

内存模型

附录 程序功能分类与路径

服务程序路径

服务程序分类、目录结构、安装包规则

idea

chkconfig管理以下两类。

windows下：Win+R msconfig。

super daemon类

负责启停一些daemon，即daemon daemon。

e.g. inetd——>xinetd负责启停一些daemon，配置文件为/etc/xinetd.conf

注：PID为1。分Sysv init->upstart或systemd两类。

stand\_alone类

注：能力较弱或开放权限较多的服务需要用super daemon管理

daemon脚本所在目录

启动脚本 /etc/init.d/\*，亦可用serivice程序，相等。

初始化配置文档 /etc/sysconfig/\*

托管给super daemon时？放于 /etc/xinet.d/\*

程序到TCP/UDP 端口、IP绑定 /etc/serivice

注：tcpwarp包的tcpd功能libwarp.so 由/etc/hosts.deny、/etc/hosts.allow、/etc/serivice组成。

e.g. /etc/init.d/sshd

/etc/init.d/haldaemon，udev。hal负责pnp硬件侦测，udev负责创建设备文件。

daemon exe所用的文件

自己配置文件位置 /etc/\*

自产生的数据库 /var/lib/\*

自己PID记录处 /var/run/\*

e.g. network sshd httpd vsftpd exe集+script的以script为总入点。

例

1，周期执行。非24时开机的系统用anacron，其基于时间戳；服务器用contab命令(其程序为crond)。

contab -e

# 分 时 日 月 周 身份 指令串

#\*为任意，同列多值时用 “，”隔开

#\*/N为每隔N个当列时间单位

：wq

contab -l #查看

排期执行 at

服务型程序特点

crtl-Z 和 后缀& 、bg的异同；fg。

热加载程序功能

第四章 函数

4.1 命名空间：

由于函数是在当前shell中执行的，所以变量对函数和shell都是可见的。  
①BASH 中要求函数的定义必须在函数使用之前。所有函数在使用前必须定义。这意味着必须将函数放在脚步开始部分，直至shell解释器首次发行它时，才可以使用。调用函数仅使用函数名即可。

执行函数时，它保留当前shell和内存信息。（此外如果执行或调用一个脚本文件中的另一段代码，将创建一个单独的shell，因而去除所有原脚本中定义的存在变量。）

②变量作用域；在函数中对环境中所作的任何变动也会对于shell环境生效。

4.2形式

为了方便结构化的设计，BASH 中也提供了函数定义的功能。BASH 中的函数定义形式：（亦可在函数名前加上关键字f u n c t i o n)：

function my\_funcname {

code block

}

或者

my\_funcname() {

code block

}

4.3 传入参数与位置参数的使用

函数内引用传入的参数同在一般脚本中使用位置参量 $1 $2...$9一样，函数取得所传参数后，将原始参数？传回shell脚步，因此最好先在函数内重新设置变量保存所传的函数。这样如果函数有一点错误，就可以通过已经本地化的变量名迅速加以跟踪。  
（？函数里调用参数（变量）的转换以下划线开始，后加变量名，如:\_FILENAME 或\_filename函数在当前环境中运行。，函数共享调用它的脚本中的变量，还允许以位置参量复制的方向想函数传递参数。？）  
可以使用local功能在函数内部创建局部变量。

4.4 从调用函数中返回

当函数完成处理或希望函数基于某一测试语句返回时，可以做两种处理：

1）让函数正常执行到函数末尾，然后返回脚步中调用函数的控制部分

2）使用return返回脚步总函数调用的下一个语句，可以带返回值。0为无错误，1为有错误

这里是可选的，与最后状态命令报表的例子极其类似。其格式为：

return 从函数中返回， 用最后状态命令决定返回值。

Return 0 无错误返回。

Return 1 有错误返回

4.5 调用关系

4.5.1 调用本文件内定义的函数。

要在脚本中调用函数，首先创建函数，并确保它位于调用之前。

以下脚本使用了两个函数。此脚本前面提到过，它用于测试目录是否存在

ie,1:

#!/bin/sh

#function file

#------------------------------

#函数1

is\_it\_a\_directory()

{

\_DIRECTORY\_NAME=$1

if [ $# -lt 1 ]; then

echo "is\_it\_a\_directory:I need a directory name to check"

return 1

fi

if [ ! -d $\_DIRECTORY\_NAME ];then

return 1

else

return 0

fi

}

#-------------------------------

#函数2

error\_msg()

{

echo -e "\007"

echo $@

echo -e "\007"

return 0

}

#===============================

echo -n "enter destination directory:"

read DIREC

#在脚步中直接调用上面定义的函数1

if is\_it\_a\_directory $DIREC then :

else

#在脚步中直接调用上面定义的函数

error\_msg "$DIREC does not exist...creating it now"

ie,2

mkdir $DIREC >/dev/null 2>&1

if [ $? != 0 ]

then

error\_msg "could not creat directory::check it out"

exit 1

else :

fi

fi

echo "extracting files..."

上述脚本中，两个函数定义于脚本开始部分，并在脚本主体中调用。所有函数都应该在任何脚本主体前定义。注意错误信息语句，这里使用函数error\_msg显示错误，反馈所有传递到该函数的参数，并加？？两声警报。

4.5.2调用其它脚本文件内的函数

前面讲述了怎样在命令行中调用函数，这类函数通常用于系统报表功能。

现在再次使用上面的函数，但是这次将之放入函数文件functions.sh里。

[root@localhost my\_shell]# vi functions.sh

#!/bin/sh

#------------------------------

is\_it\_a\_directory()

{

\_DIRECTORY\_NAME=$1

if [ $# -lt 1 ]; then

echo "is\_it\_a\_directory:I need a directory name to check"

return 1

fi

if [ ! -d $\_DIRECTORY\_NAME ];then

return 1

else

return 0

fi

}

#-------------------------------

error\_msg()

{

echo -e "\007"

echo $@

echo -e "\007"

return 0

}

#===============================

?调用

第五章 调试

Shell 脚本的追踪与调试

Shell提供了一些用于调试脚本的选项，如下所示：

-n 即dry run,读命令但不执行，检查语法

-v 将过的脚本命令打印到标准错误输出

-x 跟踪执行，将执行的每一条命令和结果依次打印出来

使用这些选项有三种方法，

①在命令行提供参数

$ sh -x ./script.sh

②在脚本内开头提供参数

#! /bin/sh -x#跟踪模式

③是在脚本内中间用set命令启用或禁用参数

#! /bin/sh

if [ -z "$1" ]; then #第2个元素，即下标为1的第一个参数应？；若长为零，即空

set -x

echo "ERROR: Insufficient Args."

exit 1

set +x #？

fi

set -x和set +x分别表示启用和禁用-x参数，这样可以只对脚本中的某一段进行跟踪调试。

除错的大概过程：

①首先使用“-n”选项检查语法错误，

②别忘了先定制PS4变量的值来增强“-x”选项的输出信息，至少应该令其输出行号信息(先执行export PS4='+[$LINENO]'。也可以利用trap,调试钩子等手段输出关键调试信息，快速缩小排查错误的范围，并在脚本中使用“set -x”及“set +x”对某些代码块进行重点跟踪。

③然后使用“-x”选项跟踪脚本的执行。

？Shell脚本怎样进行TDD

附录1 处理文本文件的idiom algorithms

1，DNF的包依赖推导算法。bicode的包依赖推导。

第二篇 信息工具

选用标准：高品质 常更新 很强大

包

util-linux——》fdisk,gdisk,mount 。raid-tools——》mdadm。iproute。Vim。Busybox。

gnupg2。lizp。selinux(chacl,getfacl,setfacl)。coreutils,shadow-utils,

lvm2。ntfs-3g包

dmidecode,硬件信息侦测包。biosdevname 将linux命名转为硬件物理平台名称,e.g. ethx——》物名

第一节 文字编码、字体与图像

|  |  |  |
| --- | --- | --- |
| 数据类型 | 较好的格式 | 较封闭的 |
| 文本文件 | ASCII pdf |  |
| 图像 | PNG TIFF JEPG | GIF BMP |
| 声音 | WAV MP3 AIFF | Au WM RealAudio |
| 视频 | MPEG | AVIQ Quicktime Realvedio |
| 数据库 | ASCII DBF | Excel Access Filemaker |

ODF格式:ISO/IEC26300.[有插件ODF Plugin-in for MSoffice]

OXML、UOF2.0 。

语言书写字符（文字）编码

1，glibc-common包的iconv

iconv -l #列出本机支持的编码集(输出集包含于glibc支持的集)

file -i 原文件名 #查原文件的编码格式；或-bi

iconv -f src\_encoding -t dst\_encoding 原文件名 -o输出文件

#from原编码格式 to 目的编码格式

#对windows下创建的文本文件，若file-mime显示为ISO-8859-1，则iconv中用GB18030/ windows-936/CP936/等 代替

e.g.

\*.lrc文件，VLC支持的字幕文件的编码。VLC支持的字幕文件后缀：(?格式、编码?) jubler subtitle workshop 。 .cdg .idx .srt .sub .utf .ass .ssa .aqt .jss .psb .rt .smi .txt .smil .stl .usf .dks .pjs .mpl2 .mks;

.srt文件中的中文在vlc下不显示问题，所有中文在vlc下不显示问题。

颜文字emoji、表情图：utf8mb4

注：

1，ASCII包含于ISO-8859，字母语言集编码，不含亚洲字符。

2，机内码

|  |  |
| --- | --- |
|  | Windows Code Page |
| GB2312-80 | CP936包含于GBK,win95 |
| GB18030 | CP54936，win NT |
| 颜文字emoji | ? |

mysql的utf8mb4编码集？

3，MS平台，或记事本中所称“ASCI”不是指ASCI/ASCII编码方式，而是指 当前windows系统locale的编码系统 的编码。比如安装的是中文版widowsNT就是指GB18030，安装的是繁体中文版则是BIG5编码。

4，GBK、Big5的编码与ASCII的故意重合，又与ISO-8859意外重复；故linux下file识别GBK、Big5、ISO-8859-1为三者之一时，认定为ISO-8859-1（即优先识别为某种西文，西方观念优先）。

参考：http://blog.csdn.net/haiross/article/details/45061181

输入码 ，输出码参第一卷

2，tr -d ‘\r’该DOS行尾,删’\r’而成unix行尾；col 处理tab->space,或expand -他命令；

windows下UTF8文本文件开头含Byte Order Mode，

Vim：set fileencoding = encoding-name。Python：缺省环境为ASCII，因与C一致，而C标识符取自ASCII集。(PEP213)可用指定的注释格式来指明解析源码文件编码、数据文件缺省的编码 标准。

mount -t ntfs时中文选项 -o iocharset=??

Convmv专用于处理文件名乱码。--notest [-f -t]

1，VFAT、NTFS文件名用utf-16码，建议用mount的iocharset选项或nls选项来解决；

2，http、ftp下载的中文乱码，常含有%时，用--unescape来转码%即可；百分号编码，亦即URL编码。

3，参网站？说明

颜色空间

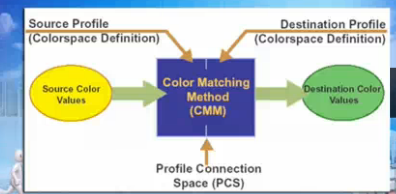
1. GPGPU驱动

二、

2.1 Xorg于2005.12发布x11R7.0，/etc/x11/xorg.conf之section “device”可配双屏显示驱程。

idea:xserver--/usr/bin/x---xinit.exe; xclient---xterm---gnome;startx.sh负责启动前服务器和客户端。初始化服务器的配置读取~/.xserverrc,/etc/X11/xinit/serverrc，色彩配置在/usr/share/X11/rgb/\*；初始化读取客户端的配置~/.xinitrc, /etc/X11/xinit/xinitrc；可查看starx.sh文件。

2.2计算机系统配色机制。



三、字体

startx.sh依赖于XFS程序，chkfontpath 或/etc/X11/xorg.conf文件检查图形系统的字体配置。

字体程序：XFS程序(X font server) /etc/init.d/xfs.sh, 配置文件为/etc/X11/fs/config, 字库文件为 /usr/share/X11/fonts/\* 。？命令mkfontscale，mkfontdir，fc-cache-fv ——————不适用

？路径fonts.dir

F25安装字体：

/usr/share/fonts/

fs-cache -f -v刷新字体缓存。

xlsofnts列出已装字体,

一些参考网址

图形图像编码

文件格式和用户数据的编码

文件格式~编码标准——容器和编码

1，多媒体文件格式编码表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文件格式音频编码 | 视频 | H.263 | H.264 | MPEG-1 | MPEG-2 | MPEG-4 | WMV | XVID |
| PCM |  |  |  |  |  |  |  |  |
| MP2 |  |  |  | VCD/.dat | SVCD/? |  |  |  |
| MP3 |  |  | .flv |  | 兼容DVD 720X480/MPG | AVI/.avi |  | 兼容AVI |
| AAC |  |  | 兼容MP4，主流 |  |  | 三星MP4 |  |  |
| WMA |  |  |  |  |  |  | WMV |  |
| AC3 |  |  |  |  | 兼容DVD/.vob |  |  |  |
| AMR\_NB |  | 3GP |  |  |  |  |  |  |

Swf——canvas格式转换。 监控视频.asf格式

1，图像小波编码：EBCOT最佳阶段嵌入码块编码算法及JPEG2000。

2，CIF(common intermediate format )视频格式，用于支持NTSC和PAL信号的视频会议系统，亦为ITUH.261标准一部分。规定每秒30帧，每帧288行，每行352像素。

3，Free Lossless Audio Codec，无损，压缩率50%且实时播放，8声道\*32bit/声道，655KHZ/道的PCM；win下可用Foobar2000。 Ape、tta、tak亦可。

4，H.265,High efficieny video coding，2013.03，支持到8192\*4320(即8K)，压缩比更高。

2，容器修复和操作

mencoder、

avidemux：libass字幕ASS/SSA

vob容器、 vobcopy客服CSS数据干扰系统

mkvtoo mix

直播

推流端/主播端：采集、预处理、编码、推流。

服务端/CDN：清晰度转换、录制、截图。

Player端：拉流、解码、绘制。

互动系统：聊天室、礼物系统、赞。

软硬编码的效率分界在720p，一台8核设备实时转10路流。

A，推流使用RTMP+Qos或flv可将延迟控制在1~3S内，RTMP是TCP(1935口)长连接，flv（flash video包括.flv .f4v swf格式文件）是http长连接，HLS（http live streaming包括.m3u(8) 文件）是小段http连接，也是一个系统(vlc H5 apple)必须支持的部分。

RTSP，基于TCP554口，点播控制协议，其可用RTP/RTCP来传流。自己仅传播放器命令，先传AVC sequence header以标识编码信息。CDN有成熟的加速服务。实例有 open broadcaster software、crtmpserver。

SIP 属于RTSP，是3GPP标准中双向电视会议协议，支持的视频编码有：MPEG-4、H.263，音频编码有AMR、GSMFR、G.723.1、G.11。SDP(会议描述协议)协商媒体类型，SIP传送控制信号。

H.323 为现用电话/视频网络使用的协议(尤其中国运行商系统)，SIP为未来因特网多媒体通信的事实标准。

B，RTP/RTCP，基于UDP的流传输协议，前者传数据，后者进行Qos。支持H.264、MP4。TS流即transport stream，无文件容器的流文件，常指MPEG2的流组成的文件。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RTSP | | | http seek | HLS |
| SIP | RTMP | RTP/RTCP |
| TCP | TCP实时 | UDP | TCP | |

real Vedio：.rm, .rvm, .rmvb, .rv;

real Audio: .ra ;

Open Broadcast System Studio，

avidemux、blender

声音编码

APE??

1，ffmpeg

2，camstudio、krut、webinaria录屏、simple screen recoder

3，darwin steaming server

视听取证工具

技术客观性、原始性好；

全称人、与其他证据相佐证；头部人物说明、时间、地点；

当事人自由自愿，非胁迫、威逼利诱、侵犯隐私，勿要挟语气、着重叙事实、承认或否认，勿言法律责任。

第二节 文件访问控制

系统对文件访问权限的控制分三层：

最外防火墙，

中间为用户对文件的访问权限DAC(chmod, chattr,setfacl)，目录所有者可对文件进行所有操作。

最内为操作系统MAC(selinux)。ls -l 的输出权限列：文件类型 attr rwx acl ，ls -Z冒号三字段列即文件/目标的安全性环境。

DAC自行访问控制

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| chmod | u  g  o  a | +  -  = | r  w  x | 目录名  文件名 |

文件系统内属性(特定于ext2/3),即隐藏属性（suid,sgid,sbit）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| chattr | +  -  = | a,i  s,u,S | 文件名 |  |

ie,用chattr改/var/log/syslog.conf的属性:

账户：bin,daemon,adm,nobldy,1~499;

ACL，访问控制列表。设定user,file,dir的rwx，需文件系统支持(ext2,JFS,XFS,RFS);mount -o acl ,启动文件系统的acl支持，可用dumpe2fs -h查看。msys2支持否？

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| setfacl  getfacl | -m,-x  -b,-k  -R,-d | ?::acl参 | 文件名 |  |

MAC强制

1，“domain-type”模型。资源(文件 端口 目录)在type内，称object目标；进程在domain内，称subject主体；用户在role内，，策略.pp文件。

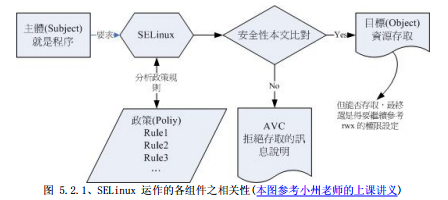
2，三种模式：策略强制执行Enforcing、不强制执行且记入日志permissive、完全禁用disabled。政策policy[targeted,strict],安全性环境security context==使用者与被用者的rwx匹配计算。策略有两种：strict所有进程受保护、targeted仅目标网络进程受保护，SELINUXTYPE=。

3，委任式存取控制。RH系selinux和DB系apparmor。Ls -Z或 --context选项对应selinux字段。mount的 context="system\_u:object\_r:samba\_share\_t:s0"选项意为赋给samba使用该设备的权限。

服务：sestatus -v测试设置，setenforce,获取模式getenforence[强制enforce,宽容permissive,关闭disable],或配置文件/etc/selinux/config中SELINUX=permissive\_mod新配置须重启os生效。

设置文件：chcon,restorecon -R,-v;查看 seinfo,sesearch;诊断：setroubleshoot,aduitzwhy。

注：所有的权限认证系统的功能基础都由操作系统的PAM模块实现。



注：以root权限运行的进程，在漏洞处植入代码，代码有root权限则可将su文件拷到/system/xbin目录(分区)，设置su文件的所有者为“root”，其setUID为“当前用户”，用adbshell。

第三节 文件关联

命令行选项规律：传递-h或-H给GNU或BSD工具：ls，du，df来以易读格式输出。ls -lh，df -h，df -k ，free -k -b -m -g

一、块设备、字符设备文件关联

fuser -u m v k i, --signal

lsof -aUu,td ;list opened files by

lsof -i4 -p 进程名 找进程的networkfiles ;lsof -i6 -p 进程名 找进程的networkfiles

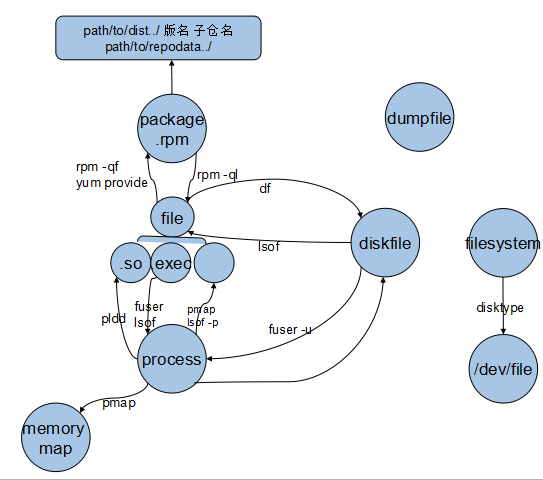
pidof -sx程序名

目录挂目录： mount --bind src dst ;按盘符挂载：e2label,mount -L 盘符名

镜像文件挂目录：mount -t fs\_type src\_file dst\_dir -loop??

sshfs 远程挂载

Scp远程复制。Lrzsz。secureCRT



{ldconfig,pmap,ldd,pldd,iconv}

二、其它设备文件关联

0.模块文件关联（与挂载设备类比）

1.[depmod，计算依赖并写成磁盘文件。]

/lib/modules/$(uname -r)/kernel/fs/\*挂载后进入/proc/filesystem/\*，ie,对raid功能lsmod|grep md , modprobe md,加载成功则有 cat /proc/mdstat/ |grep “md”。

2./etc/modprobe.conf设定预加载模块的别名和加载参数。

例1：更新/重编译现有系统的外置模块

Initramfs-tools包

1. update-initramfs:  
   内部会调用mkinitramfs;  
   对生成文件生成sha1sum校验
2. initramfs.conf

构建和安装

1.depmod –a -A，  
依赖/lib/module/`uname –r`/ \*.ko；  
生成/lib/modules/`uname-r`/modules.dep #类似fstab  
 和/lib/modules/`uname-r`/\*.map #用于热插拔

Modules.dep格式如下：  
#逆序排列  
模块名：其依赖的模块名1 其依赖的模块名2  
#modeprobe加载模块时，从右至左加载，卸载时从左至右

2.modprobe   
–l   
-a #用的相对路径，路径配置在modules.dep内，私mount  
-r #模块配置信息在/etc/module.d/\*, 以.conf后缀的文件中均可指定选项。/etc/modprobe.conf文件仍可使用，但将在未来遗弃。

Modprobe.conf书写格式：  
一行一命令；空行、注释行#无效；“\”为续行符。  
命令：1. alias，支持shell通配符，alias 别名 模块名 ，modinfo命令可查看模块自带的别名  
 2. options [模块名] [option] ，设置加载选项  
 3. install 起作用的模块名 shell命令语句 #在执行modprobe时执行  
 remove 起作用的模块名 shell命令语句 #在执行 –r选项时执行  
 4. Blacklist 起作用的模块名 #使该模块的内部名失效，即屏蔽该模块的内部别名。

1. 构建和安装Initramfs，mkinitrd -v --with=模块名 initrd名 #initrd损坏导致启动错误怎么办？
2. 停用旧模块，测试新模块
3. 编写/etc/modprobe.d/\*.conf关闭旧模块  
   update-initramfs –u
4. 安装新模块：prebuilt包（n卡官方.run包或仓库nvidia-drivert、nvidia-xconfig包）或手工构建包手工配置再编辑3.1、执行3.2.
5. 加载并测试固化后的新模块  
   5.1modprobe nvidia #官发  
   lsmod |grep –I nouveau #开源

Inotify-tools

例2．重编译kali内核

1，安装构建环境的依赖  
Apt install build-essential libncurse5-dev fakeroot unxz

2，安装源码  
apt install linux-source-4.9  
#/usr/src/linux-source-4.9.tar.xz确认内核存在

3，解压tar -xaf parh/to/linux-source-4.9.tar.xz

4，配置内核  
A，  
cp /boot/config-version path/to/linux源码目录下  
mv path/to/linux源码目录下/ .config #重命名  
B，  
make menuconfig

5，构建内核

Make clean

make deb-pkg LOCALVERSION=-custom KDEB\_PKGVERSION=$(make kernelverdion)-1

#生成linux-image-version.deb内核+内模块  
linux-headers-version.deb构建扩展模块所需的头文件  
linux-image-version-dbg.deb调试内核+内核模块的符号  
linux-libc-dev.deb用于编译用户空间应用程序的头文件  
linux-firmware-image-version某些驱程要用到的固件？外置设备驱程模块？

6，安装内核  
dpkg -i ./linux-image-version.deb  
reboot  
#启动异常时，先用grub引导旧正常的内核，再卸载掉（什么命令）新异常内核。

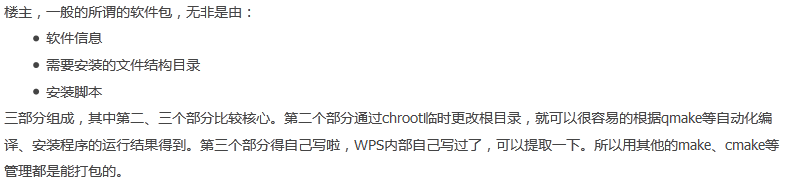
三、流设备关联

例1： 无线局域网模块栈，GPS栈，U盘



包管理

0,包制作



COPR OBS PPA仓库平台。

1，软件包状态查询或变更管理信息记录

2，软件包安装

centos系仓库配置：

默认情况下，一个已下载的rpm包放在/var/cache/yum/x86\_64/[centos/Fedora\_版号] / [base,updates,fedora] / packages/下。

定制放到的路径可用yum install --downloadonly --downloadder= path/to/your/package\_file

debian系仓库配置：

/etc/apt/sources.list文件内——e.g. file: path/to/debian Jessie main contrib

deb [option] uri/to/debian-distribution/../ stable/testing/版名 main/contrib/non-free

deb-src

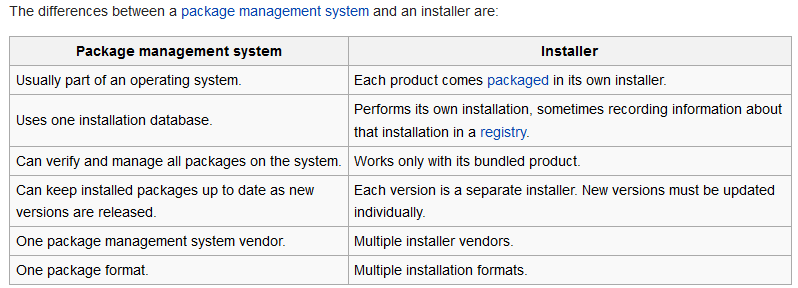
Apt.confd.d/下控制源的优先级和禁用关系；

Sources.dist配置源址；改后须apt-get update，否则不会自动更新；

pacman

win控制面板+register table。selinux+包工具+mandb。register table，树状DT，软件的全部配置信息、初始化信息、文件和程序的关联信息、硬件的信息。存储为system.dat+user.data。

3，软件包卸载



例1 用于解决播放器问题

xmms

1，安装xmms-mp3,xmms-wma;

2,安装xmms,xmms-dev;

3,图形界面中，

选”使用字体”为”使用X字体”；

播放清单-> -sony-\*-\*-``````16-\*``````-iso8859-1

使用x字体-> \*-\*-\*-``````-16-\*``````-gbk-0

<xlsfonts命令列出已安装字体>

标题->标题格式->%f

安装二进制版Mplayer

1,安装软件包livna,freshrpms

2，安装mplayer,mplayer-skins;

3,解压解码库all-20061022.tar.bz2;

默认须解压至/usr/lib/win32;编译时 ./configure的codecsdir = ... windows-all决定位置；

4,mplayer-gui,mplayerplug-in;

rhythmbox

gstreamer与ffmpeg问题?

所需插件（1,gstreamer1-plugins-bad-free,2, gstreamer1-plugins-ugly[mp3],

3,g\*1-vapi,g\*1-libav;4,g\*1-p\*-crystalhd）centos版的存于EPLE源、li.nux.ro源内。

注：centos系多媒体类二进制包源站点——li.nux.ro(较新的)、linuxtech、atrpms。pulseAudio。fedora中版权为第三方的库放在{rpm Fusion，livna}

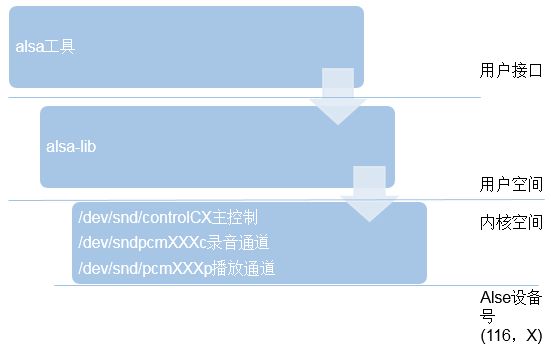
局域网服务：paprefs图形UI，pulseaudio-module-zeroconf声音服务程序；

局域网客户：paprefs ， DLNA/uPnP协议客户端或PulseAudio作客户端；

Linux声音系统

Advanced linux sound architecture，高级linux声音体系，由内核驱动程序和用户空间函数库(alsa-lib)组成。建议开发者使用函数库而非直接调用设备文件。Alsa-utils是测试套件。

Alsa提供的设备文件是字符类设备，主设备号是116，次设备号另单独定义。主要的设备文件有/dev/snd/controlCX主控制文件，/dev/snc/pcmXXXc捕获capture PCM通道的文件，/dev/snd/pcmXXXp 播放play PCM通道的文件。



老式的Open sound system中，主设备号为14，主要的设备文件有 /dev/mixer控制音源和音量，/devsndstats输出声卡驱动的信息，/dev/dsp、/dev/dspW、/dev/audio读写设备 read读时即录音、write写时即放音、区别在于编码方式不同。

声音采集和加工技术，SRS音效技术

第四节 系统加固

一、多磁盘卷

块文件的RW方法有分区、RAID、磁盘捆绑。

A，RAID 冗余阵列技术

|  |  |  |  |
| --- | --- | --- | --- |
| level |  |  |  |
| RAID0 | >=2 | 无备份 | 2倍速 |
| RAID1 | >=2 | 全备份 | 单速 |
| RAID10 | >=4 | 全备份 | 2倍速 |
| RAID2 | >=3 | 汉明校验 |  |
| RAID5 | >=3 | 折中0,1 |  |

ieda：1，重点关注数据的可靠性、读写效能；2，以partation或disk为磁盘单位，组向上模拟成一大硬盘；3，实现RAID 0/1/5 spare disk,向下支持热插拔。

软件：包名mdadm,模块名md;

操作：

0,底层每盘分区格式标为linux raid ,即0xFD;

1,组阵列盘，mdadm --create --auto=yes /dev/mdX阵列名 --level Y --raid-devices=Z --spare-device=盘号 /dev/sdxx{}分区集。若需持久化则须写/etc/mdadm.conf

2，分区或格式化阵列盘，mkfs.ext4 阵列名

3，挂载使用阵列盘上的分区，mdadm --detail 。mount

4,换底层坏盘。mdadm --detail。mdadm --manage 阵列名 --remove 换出底层盘名 --add 换入底层盘名

5,umount 阵列名；mdadm --stop阵列名。

B，跨区卷/简单磁盘捆绑技术

即Just bundle of disks，或diskspaning。例如lvm。

idea:1，关注一个文件按系统大小的动态可调，不关注底层数据可靠性、读写性能；2，一pv(0x8e)格式的分区为底层盘，先整合成一VG，再将VG分割成若干LV，将Lv格式化后使用；3，

操作:

win的内存交换空间称“虚拟内存”，其使用的是文件pagefile.sys，即交换文件。

二、镜像备份 文件压缩 技术

用以下单项技术或用colonezilla的drbl版。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 备份、逆删除工具 | 下层栈 |  |
| 工作在文件系统上  fsck； (win scandisk) | <dump,restore>;  tar | udev | 提供文件系统 |
|  |  |  |  |
| 工作在设备块上  gdisk,partimage | <testdisk,photorec>;  ddrescue | hal | 提供裸设备 |
|  | DFF |  |  |

注：fsck问题记入lost+found目录。

无损压缩

huffman各字符出现频率不均匀情况下实现压缩。Lz77，Lempel-Ziv-1977，字典压缩算法的鼻祖；先Lz77再进行Huffman第二遍压缩：Gzip .gz；Lz77-Lempel-ziv-markove，引入区间编码(Range Encoding)。

tar 选项 -C 目的目录 缺省为当前目录-f 指定的archive文件

debian下不使用-Cf选项

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| lzma2 | -J | -------c------->  <------x------- | .xz | xz |
| CRC | --lzip | .lz | lzip |
| 10:1 | -j | .bz2 | bzip2 |
| 2:1 | -z | gz | gzip |

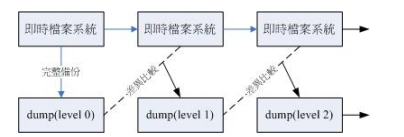
-t 查看； -c 即compress,-x即extract。

tar -P保持文件的属性不变

-p 保留文件的绝对路径 ，都是信息危险项。

文件系统上层数据备份

dump,restore



|  |  |
| --- | --- |
| dump | -S 列出所需空间,字节 |
| -level 0~9 |
| -f 备份文件名 |
| -j bzip2压缩 |
| -W 与/etc/fstab里备份选项核算 |

|  |  |
| --- | --- |
| restore | -t,-C 查看，比较 |
| -i 编辑备份 |
| -f ？文件名 |
| -r |

linux配置文件备份，Backupninja（部分，不完善）。

文件系统元数据说明：NTFS、UFS

任意文件复制

位复制时目标盘容量与源盘容量大小关系？

ddrescue,cpio,

dd二进制系

|  |  |  |
| --- | --- | --- |
| dc3dd(dd) op=value | ddrescue [op] ifile ofile [logfile] | cpio |
| if=输入文件 |  |  |
| of=输出文件 |  | -o 输出文件 |
| bs,ibs,obs读写单位x B/次 | -c 默认128块 | -C 字节 |
| count单位的个数 |  |  |
| skip从ifile开头跳过n个单位 | -i |  |
| seek从ofile开头跳过n个单位 |  |  |
| conv编码转换 | conv | -c 文件格式 |
| --log --hlog --mlog | 推荐使用日志模式 -l | -t 执行信息 |
|  | -f -b 2048 -r1 -d |  |

注：

ddrescue用日志时可随时中断并继续

1，ddrescue/dd\_rescue输出文件不变且指定多遍读时会将多遍读的结果合成一个文件。

2，lziprecover可将同一文件的每个拷贝综合作为输入来输出/解压出原文件包。ddrescue+lzip压缩备份，对备份多份长久存放缺损或photorec的多遍恢复集，用liziprecover恢复，适合永久存储。

lziprecover

e.g.1 -R#一份文件只有一字节出错时

-m#从多个备份恢复出一份完整的，可多字节纠错

e.g.2 -d \*.lz #解压

-tv \*.lz#校验

-m -v -o out.lz inbk.1.lz inbk.2.lz

注：数据恢复。testdisk包(photoRec+testdisk)；vim :%xxd,Ultraedit(win);(商业R-studio data recovery software)

1，分区表损坏分析与恢复。

2，分区损坏分析与恢复。

3，基于文件系统的数据恢复。

4，基于文件系统块的数据恢复。

5，mkfs.ntfs选项、shred 与 文件可恢复程度

3，Phototec怎与ddrescue结合？按名称判定还是按内容判定

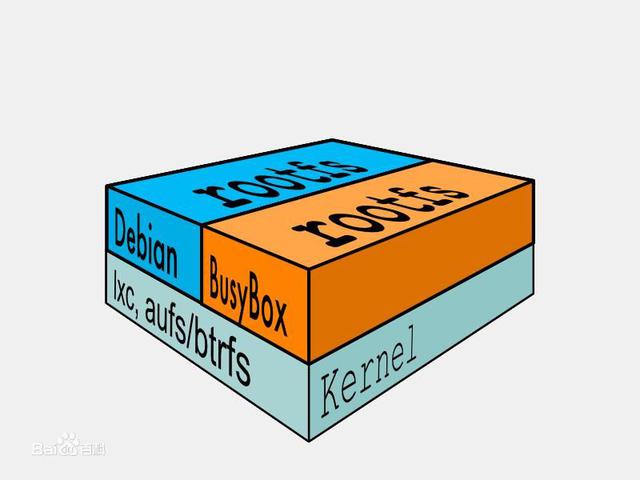
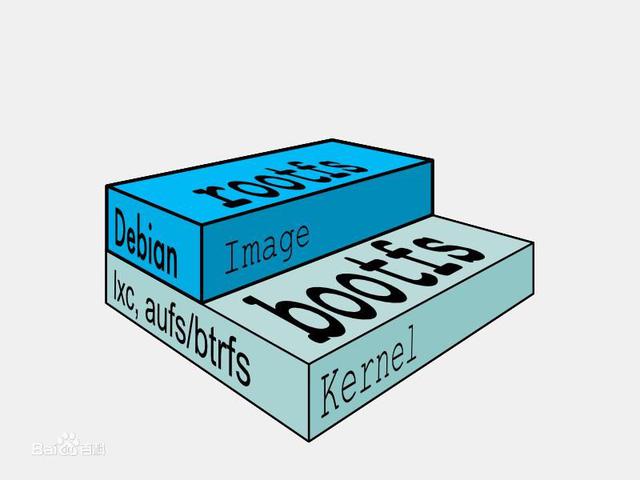
文件系统级数据恢复：

E2fsprogs包的ext4grep编辑inode

The Sleuth Kit

partimage等同windows Norton’s Ghost。clone与image的异同？

附1.可启动硬盘镜像的脚本。



winPE支持 .gho、.img、.iso格式镜像/备份。E01格式 .dd格式关系? .dd格式是否支持压缩。aff格式。

1，iso文件格式。

ie,gzip -d initrd-xxx.img initrd-xxx.img.gz;#file initrd-xxx.img; cpio -iduvc<initrd-xxx.img

ie,(iso9660,UDF).iso.img=.ima,.img---zip--->.imz，.dmg(apple disk image)；请编写光盘转换为可启动硬盘镜像的脚本。

2，DVD启动规范。

3，fedora系mkdiskimage脚本(dd属genisoimage属mkisofs)

Kali构建ISO文件

1，apt install curl git live-build-config.git  
git clone git://git.kali.org/live-build-config.git

2，cd live-build-config

1. ./build.sh --distribution kali-rolling --verbose#下载用于构建ISO的包  
   或  
   ./build.shi --distribution kali-rolling -variant[gnome，kde，xface，mate，e17，lxde，i3wm ] --verbose #debian live-build man page
2. 配置定制文件  
   kali-config/common/include.[chroot，binary，installer]/  
   kali-config/common/hooks/

3，运行构建工具

注：grub操作工具

A，grub

1）grub-install

2）mkinitramfs -o /boot/initrd.img $(uname -r) #版本号  
update-initramfs -u #何用？

3）update-grub #更新grub的配置文件/boot/……

B，efi boot

例1 引导工具包（win>6）

以下三列等价：

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

#文件/etc/grub.d/[00,20]

#文件/etc/default/grub

#命令update-grub

#文件/boot/grub2/grub.cfg

menuentry custom\_name{

[insmod part\_msdos ntfs ntldr]

set root=’hd0,msdos2’

chainloader+1

boot

}

#grub2启动时命令行

仅输入大括弧内内容

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

以下组等价

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

drivemap -s (hd0,5) ${root}

ntldr /bootmgr

boot

set root=(hd0,5)

Chainloader+1

boot

ntldr (hd0,5)/bootmgr

boot

注：root变量== / ==${root}；chainloader+1与 ntldr /bootmgr应等价。文件系统模块、menu.lst放在/boot/grub/下且于第二阶段执行。

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

引导iso文件

loopback custom\_dev\_name (hd0,5)/path/to/osimgfile.iso

linux (custom\_dev\_name)/path/to/kernel #vmlinuz vmlinuz.efi

initrd (custom\_dev\_name)/path/to/initrd.img # initrd.lz

配置 kdump 使用的内存大小。添加启动参数"crashkernel=Y@X"，这里，Y 是为 kdump 捕捉内核保留的内存，X 是保留部分内存的开始位置。对于 i386 和 x86\_64, 编辑 /etc/grub.conf, 在内核行的最后添加"crashkernel=128M" 。

dmidecode -t memory | grep Size

附3. PC机的两种启动方式

boot loader

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | grub | EFI | NT6.1 | XP |
| 程序 | grldr | EFI shell | bootmgr | NTLDR |
| 配置文件 | menu.lst | BootX64.EFI | C:\boot\bcd | Boot.ini |
| 编辑/命令 |  |  |  |  |

引导程序：rEFInd、Yosemite、Bootcamp[Apple制]

DOS分区体系、GPT分区体系、Apple分区体系、BSD分区体系。

A，EFI+GPT磁盘引导

EFI加载？驱程、？引导程序、？os

GPT头扇区与GPT备份扇区的内容不完全相同，不可以copy法恢复，须用？

B，BIOS+MBR

BIOS加载首扇区。

dos/mbr分区体系中，bootloader分配446B，分区表分配16B×4=64B，结束标志0x55AA占2B，共512B即一个扇区。  
bootloader的446B：引导代码分配440B，磁盘签名分配4B。磁盘签名：win会读签名至注册表中，若丢失则提示“磁盘未初始化”。  
结束标志：地址[0x1BC，0x1BD]处，若丢失则亦提示磁盘未初始化。  
分区表项：16B/个，每项由<分区类型标志，可启动标志，有分区起始地址>对应<fdisk的toggle功能，fdisk 0x80，小端序/小头位序>

嵌入式SOC的启动stepstone==BIOS。

例2 引导安全

kernel 参数加single可指定内核启动进入担任模式且不需要密码；

/etc/passwd放root密码，/etc/shadow放个人密码，/etc/group放分组信息。删除无用，须改

Ro为rw，且行末加init=/bin/bash，进入后passwd 新密。

bootloader

e.g. 引导MS：启动时F8或|\_|， \windows\system32\bcdedit.exe；启动时 chainloader path/to/bootmgr

例3 windows原始镜像文件

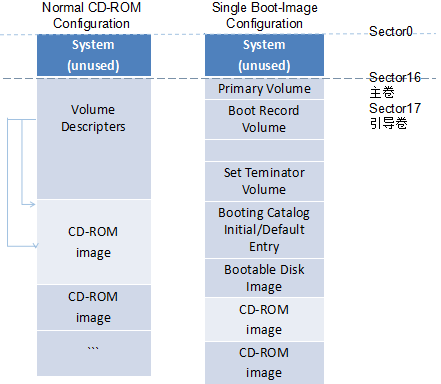
1，win7:grub/2不能正常引导其iso文件，必须用windows7 USB/DVD Download tool [烧制环境为64位系统时要64位机用的 Bootset.exe]，刻入U盘。否则，常出现碎片过多错误提示。

注：有解，见http://reboot.pro/topic/10442-grub4dos-and-windows-7iso/，http://vdisk.weibo.com/s/dBeg0N0j-yzZ?archive\_ref=F0JG1d0ph9Xoo&archive\_path=%2Ffiles%2Ftools%2FRMPrepUSB

1）ISO9660

2）即要在载入DOS之前就必须检测到CD-ROM，这在当时从软件上是无法实现的，唯一的解决方法就是修改电脑主板上的BIOS使之在硬件级而不是软件级首先识别CD-ROM，并自动加载民CD-ROM上的启动引导器。  
1995年1月末5日，可启动CD-ROM格式规范Bootable CD-ROM Format Specification1.0 即E1 Torito规范发布，定义了可启动CD-ROM的数据结构与映像数据的配置及光盘制作的一些详细说明。实际上，该规范也隐含地制定了能够读取可启动CD-ROM光盘的BIOS规范，使得符合E1Torito规范的可启动CD-ROM在电脑上能够正常启动。随后，BIOSBoot Specification1.01发布，该规范为BIOS厂家提供了制造支持可启动CD-ROM的BIOS标准。可启动CD-ROM既符合E1Torito标准又符合之前的ISO9660标准。

引导光盘工作原理。BIOS首先检查光盘的第17个扇区，查找其中的代码，若发现其中的启动记录卷描述表Boot Record Volume Descripter ，它就根据表中的地址继续查找启动目录Boot Catalog，找到启动目录后，再根据其中描述的启动入口Boot Entry找到相应的启动磁盘映象Bootable Disk Image或启动引导文件，找到启动磁盘映像后，读取其中的数据，并执行相应的启动操作。  
相对于单重启动CD-ROM而言，多重启动CD-ROM的启动目录中包含多个启动入口，指向多个启动磁盘映象。启动映像配置图如图：  
图中描述的多重启动配置是E1 Torito规范所描述的多重启动映像配置，但由于多重启动CD-ROM在实际工作中的应用较少，目前大多数主板的BIOS对此支持的不完善。在这类主板上，用遵循E1 Torito规范所制作的多重启动CD-ROM往往只能引导1个（默认启动出口所指的方向）启动映像。  
为了解决这个问题，又相继研究出了一些解决办法来实现CD-ROM的多重启动，目前最流行的办法是模拟（软件、硬盘）式BIOS模拟法。其工作原理与E1 Torito规范所描述单重启动映像配置原理基本相同，只是默认启动的不是软盘映像，而是1个启动引导文件，该启动引导文件引导光盘启动，再由它去查找其他的启动磁盘映象或引导文件，根据配置文件列出启动选项供用户选择。EasyBoot采用的就是这种方法。



1. Universal Disc Format将光盘模拟作硬盘式IO（对-R盘空间越来越小），切单文件可>2GB。
2. CD文件提取

5）SSD：1）物理扇区512B？。2）page，页，RW的单位，基本存贮单元，一般为物理扇区的整数倍。3）block，区块，擦除的基本单位，写前必须擦除；由多个page组成。4）place，区域，由多个block区块组成。

2，win10 ?

老毛桃引导程序。

？winPE为什么通用

三、文件属性清洗

三.1 磁盘数据清洗

文件时间信息

1， create modify access

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ctime | mtime上次改动时 | | | atime上次访问时 |  |
| touch |  | -m | | | -a | -t |
| ls -l -time= |  | mtime缺省即 | | | atime |  |
| find |  | -mtime | +  -  = | 天数 |  |  |

-ok 命令 {}\；#{}表得到的文件名

-exec 命令 {}\；# 即对得到的结果文件逐个执行 命令

2，cp -d -R -p -u -v -x

命令历史

hsitory -w ,文件~/.bash\_history；!命令行头，!历史命令号

擦除数据：

BleachBit痕迹清理；linus； python。

unix下The Sleutnkit autopsy可数据恢复,**shred**; win下eraser.

注：随机数。#$RANDOM #/dev/random #/dev/urandom #date+”%S%N”  
使用# head -100 /dev/urandom |cksum |cut -f1 -d “”

三.2，目录痕迹

定制信息

1. top命令

w 保存当前配置为磁盘文件；

f,o 调整显示的属性及其顺序；

<,>,F 选择要排序的属性；

注：ps -fax中[]表示的即为内核线程；

日志系统缺省配置

1，klogd记录内核，其配置，其输出在

2，syslogd记录登录信息（/etc/syslog.conf），其配置在，其输出在

Var/log/[dmesg，message，lastlog，secure]分别记录系统启动信息，出错信息，最近一次帐号登录。

win下以日志“事件日志服务启动”为开机标志。

目录结构

安装版本：centos，kali，drbl,minit

|  |  |  |  |
| --- | --- | --- | --- |
|  | Unix system resource | Centos系 | debian系 |
|  | /usr/lib | + <-/lib |  |
|  | /usr/lib64 | +<-/lib64 |  |
|  | /usr/bin | +<-/bin  +<-/sbin |  |
|  | /usr/include |  |  |
| 升级时不会动的资源 | /usr/local/lib  /usr/local/bin  /usr/local/include |  |  |
| 文件锁：程序相关，令脚本仅执行一次；有实例标志 | /var/run  /var/lock |  | /var/run侦听接口文件，属性为S，由服务程序创建 |
|  |  |  |  |
|  | /var  /tmp |  | 须定期清空 |
| 包文档 | /usr/share/doc |  |  |

注：Fedora->korora；Ubuntu->mint。

configure build时选项

prefix, lib,include,path;

CMAKE GMake NMake路径选项对应表。

V>=2.6的CMAKE的CMAKE\_INSTALL\_PREFIX()会更改目标可执行文件的RPATH(即链接ld -rpath选项)，即可执行文件在运行时使用的共享库的路径为其参。

四、数据加密 验证 技术

信息系统文档要在信息系统淘汰3~5年后才能销毁。

|  |  |  |
| --- | --- | --- |
| 保持年限 | 数据类型 | 备注 |
| 3年 | 成本计划 |  |
| 15年 | 月/季报表，明细总账，  原始凭证，记账凭证  日记账 |  |
| 25年 | 现金、银行存款日账 |  |
| 永久 | 外来账，对私账，改造账  合同，协议 |  |

A摘要算法

md5sum 文件 -c md5-hash.txt

sha1sum -b, 要检查的二进制文件

-c

-t 文本文件

MD5——> SHA-1——>SHA-2——>SHA-3 2015,keccak算法，便于硬件优化

e.g. 1，自构建可信源。2，请建立/etc/passwd，/etc/shadow，/etc/bin/passwd，/bin/login，/bin/ls，/bin/ps， /usr/bin/top、/etc/group、/sbin/portmap的指纹数据库并用shellscript定期运行检验。

A．摘要算法

B非对称密钥生成与使用

3，TOR

Tor Browser无痕浏览器。参考https://securityinabox.org/en/guide/torbrowser/windows#829

TrueCrypt：VeraCrypt，分区加密软件，跨平台，apache2.0。[VeraCrypt+BIOS加密可保系统安全]

注：PGP(RSA)、Tor、TrueCypt算法保密性好的工具。

4，用ssh生成RSA密钥，《Ssh logins using keys》

原理：术语注：Password较短的密码，一个字或单词组成。Passphrase，指较长的密码，由多个单词或字组成。

保护私钥的安全性有三种方法：A，用passphrase加密私钥，或私钥放在加密文件系统内。B，私钥放在usb keyring内，较安全写。key-chain/ring软件C，用ssh key agent代理软件，取出的私钥放在内存中；当代理软件停止时，内存的私钥丢失;ssh-agent做服务程序运行，ssh-add增加。

文件：Openssh格式和ppk格式，二者相同。Win下guttygen同ssh-keygen。Openssh缺省使用openssh格式的公私钥文件。其它格式的公私钥文件（ssh2，SECSH）用ssh-keygen -i来转为openssh格式。

步骤：Ssh加密连接三步走：

1，生成公私钥对。key=｛public key，private key｝ssh-keygen可用于生成密钥，公钥缺省文件名为id\_rsa.pub，私钥为id\_rsa。公钥为平文本格式。e.g. ssh-keygen -t rsa -b 2048生成密钥对文件。缺省密钥文件在~/.ssh/｛id\_rsa，id\_rsa.pub｝

2，在远程机上安装公钥。将分配给A用户的公钥追加到远程机A用户~/.ssh/authorized\_keys文件内，要保持一行；服务器/要登上的机上改该文件为 chmod 644。

3，测试连接。私钥文件不应为他人访问，仅存在于客户机上。公钥文件禁止为任何人所写入。登录时加-v，显示详细信息。配置在 /etc/ssh\_config，~/.ssh/config

6，逆向加密

彩虹表破解，暴力破解，掩码破解，字典破解

win2008缺省密码规则：总长>=8，必含大写字母、小写字母、数字；可在pwrshell运行gpedit.msc，在弹窗的计算机配置->windows设置->安全设置->账户策略->密码策略 中调“密码复杂性要求”项。

C

1，PKI：

客户端：USBkey（CA、私钥）、driver

服务端：签名服务器

eg. Let’s Encrypt

2，PGP做RSA签名验证

#解压文件与校验文件放同一目录

xz -cd linux-xxx.tar.xz | gpg -verify linux-xxx.sign#若无公钥则从报错信息中可找到公钥号；

gpg --keyserver hkp://deys.gnupg.net --recv-deys 公钥代号 #获取公钥

gpg --verify linux-xxx.sign

GPG的配置

附录1. 局域网部署Windows 系统

<http://openbenchmarking.org/suites/pts>

镜像：http://mirrors.zju.edu.cn/

http://mirrors.cqu.edu.cn/

http://mirrors.skyshe.cn/

http://mirrors.opencas.cn/

http://mirrors.aliyun.com/

http://centos.01link.hk/

安装时仓库目录为： http://b.mirrors.lanunion.org/CentOS/7.1.1503/os/x86\_64/

/usr/lib/mozilla/plugin/\*

注：Win+R,msconfig，配置启动服务项。linux的systemconfig、systemd

U盘用exFAT文件系统，避免出现4G单文件极限。

非最新版fdisk中，MS扩展分区ID标为Win95 Ext’d(LBA)时，mkfs格式化的FAT、NTFS才能使用。

mkfs.ntfs -L #1~5

-C#簇大小B

-S#块大小B

-T#创档时间归零。该时间怎样查？

-V

AWD环境下系统加固

Android安全攻防权威指南

1，快速修改自己密码

2，scp备份到隐蔽目录下

3，seay代码审计、用hash作文件名、检测文件上传时间

4，tail -f \*.log看日志

入侵检测系统

1.主动防御系统Instrusion Prevention system：2+3+日志+入侵检测(I Detection System)，联动+sandbox

2.Antivirus Programs：攻击代码的signature

3. Packet Filter、Application Gateway：

A， IP ：ports过滤

B，Decp Packet Inspection

OSSEC

杀毒软件 clamTK(Linux)，Clamwin(win)

Digital Forensics Framework

Linux上网基础

一、Linux无线栈

参archlinux.org的wireless network configuration，www.linuxwireless.org

1，支持较好的芯片：ath9k\_htc，carl9170，rt2800USB，rtl8192cu

2，查看设备位置、驱程位置。开源驱程默认在linux-firmwire包内。

#lsusb -v|grep -i wireless

#dmsg|grep usbcore或 lspci |grep -i wireless得芯片的地址和型号

#lsusb -nn -s芯片地址 #或lspci -vv -s 得正在使用的驱程模块

#modinfo 正在使用的驱程模块 #得驱程文件的位置、信息等。

3，用udev来配置驱程得其接口名称

Udev创建设备名，缺省时有线为en自动ID，无线为wl自动ID，可用配置文件/etc/udev/ rule.d/io-network.rule来定制设备名；用$ip link或$sysfs来查看网络设备名。Udev缺点：会对一个设备加载多个驱程，导致二者争用！可将不对的模块黑名单化(/etc/modprebe.d/中)

4，用networkmanager等应用包配置ip网络

自动network manager=手动：=ip+wpa\_suppliciant=ip+iwconfig=ip+iw，无线包受ipreoute2的ip命令控制。Wpa\_suppliciant包用于控制连接，一般先于udev安装。Iw包单一，wireless\_tools包较老。

发wifi信号，抓握手包，用web接口，启用模仿AP，启动MDK进程获取用户输入的密码。

Dsploit无线解密码。

kill wifi，网络剪刀手，防用网。

Perf、eeBPF、faceNiff

aircrack-ng

Airbase-ng 、airserv-ng TCP/IP服务，从AP发到客户端。

Airmon-ng\_\_\_\_airodump-ng\_\_\_\_\_\_\_aireplay-ng

Aircrack-ng破协议，airdecap-ng解密捕获的文件。airodump-ng捕获原始802.11帧。

二、Nmap 路由排查

缺省脚本。 -sC/--script=default依次按1.1~1.4发送4个报文，存在任何回复则证明目标机存在。 --script= ，--script-args-file/--script-args

NSE脚本=Lua脚本

/usr/share/nmap/scripts/\*.nse，eg. Telnet-brute，http-brute。/usr/share/nmap/nselib/data，放脚本所用的缺省字典。

1.1，发送回显报文，ICMP echo request，-sn -PE

1.2，发送TCP SYN包到443口，亦称半开放扫描[port 113]，-PS

1.3，发送TCP ACK包到80口，主机通时判防火墙，-PA

1.4，发送，ICMP timestamp request，-PP

1.5，SCTP INIT，-PU

1.6，ECHO，-PY

1.7，netmask，-PM

IP协议包，-PO

1.8，-sU：判断是关闭状态，stealthy、scanTCP FIN包

判断某UDP端口是关闭的，UDP,

ICMP

2，-sP

-sS

-sU

扫描

3，

不/总进行DNS解析：-n/R

追踪每个路由节点，同win tracert ：--traceroute

4，扫描端口配置

--top-ports个数

端口状态

1. Open/closed
2. filtered被IDS/IPS虑/屏掉  
   unfiltered未虑掉，不知是否开放
3. A|B——>o|f 或  
    c|f 或

6. OS版本侦测 -o

1. 通过OS的端口特征或系统指纹
2. 将探测生成的指纹与已知指纹nmap-os-db匹配相似度。

三、DNS

dig ：domain information groper

dig ……@某DNS的IP/URL

win 下nslookup

第三篇 移动端生存

1，VPS

IM

1，signal开源、移动设备打电话

2，jitsi meet，

C/S，——>Rocket.chat客户端

IP，server==FreeSWITCH，client==RIng

jabber

3，zoom付费

洋葱：onion(ios)==Orweb(android)==Orfox

Tor：Torchat——>Torchat2，2014年终止，JTorchat2014年终——>Ricochet，分布式

取证

一、ROM级数据提取

1,闪存

1. 名称,不规范的有”字库”——nandflash

Bootloader=CFE[?]

2. 16M，W25Q128的Pin8为R/W允许控制脚

32M，W25Q128的Pin16为R/W允许控制脚

1. openwrt最小16MB ROM/64MB RAM，建议32MBROM/256MB RAM
2. 编程器与JTAG
3. JTAG Forensics
4. 1，识别TAP脚、连接器模式
5. 2，JTAG仿真器
6. 3，配置CPU+memory，或选好配置，读取flash数据。
7. ——easyJTAG、USBPROG5.0、Jlink Pro/plus
8. JTAG

1。选件 JlinkV9

选软件openOCD、DNW

1. 原理

1，RAM\_init.bin至SOC

2，RAM\_bootloader.bin至RAM/SOC

3，ROM\_bootloader.bin至NAND。本步以前用TTL大法，现用JTAG更快；若无亦退回TTL大法。切记TTL传递亦是基于有bootloader 正常工作的前提（booting u-boot in RAM）。

注：TTL大法连线，TTLUSB的Rx对路由器的Tx，TTLUSB的Tx对路由器的Rx。

RAM version of u-boot，

RAM init sequence:

1，ROM code，Inital Program Loader （IPL），Primary Program Loader（PPL）

2，1st u-boot stage==init stage==secondary programer loader (SPL)

3，2st u-boot stage

1. 各加载文件的制作

1，

2，对于u-boot，配源码中

3，

1. Bootloader
2. hackPascal的breed；u-boot
3. 菜单。  
   Reset+poweron：路由器  
    ：Android

5.

1，openwrt ——>LEDE

openwrt——>中文化——>pandorabox

2，CPU  
AR 系列，5G：AR9344：TP4310。MW4530（水星）  
QCA系列，TEW823DRU，TL-WDR4900/ 7500V2 V3

3，网卡（不能用BCM系列，缺驱动）  
AR系列：AR8327N支持MIMO

二、OS数据提取

1，刷u-boot，装openwrt、android

参 lifehacker网

Android引导程序的命令行

fastboot 和adb包。SDK中包含adb和fastboot包，或minimal ADB and Fastboot Setup1.4。设备须用USB连接PC且已有驱程被PC操作系统识别。

Fastboot给bootloader发命令，该功能并非都使能。

Adb，发基本的Linux shell命令、特定的开发命令到phone，凡设备在启动完或recovery mode的设备上可用。

adb reboot recovery #重启并进入recovery模式

adb reboot bootloader #重启并进入bootloader，此时无adb shell.。即fastboot，只有可解锁的bootloader才可进行fastboot oem unlock和烧ROM。

#root与可备份的程度有关。Unlock与bootloader能不能接受新的OS有关。

Fastboot flash recovery #烧ROM

fastboot devices #查看设备连接信息

adb push S D # 电脑到phone

adb pull S D # phone到电脑

adb install SOURCE\_APK

adb shelll COMMAND #在phone上的终端上运行一个命令

adb back #非保护数据的备份

adb restore #非保护数据的

adb sideload #烧、取ROM或zip

Recovery mode：nandroid，backup，install。

术语：

Android中ROM指一个android修改版，其中stock指原声版、原厂版。

Flash指安装包文件，格式应为.zip。

Bootloader分为已解锁和未解锁。

Hard bricked，soft bricked，broke。

三、文件级存取

附2.智能设备ROM数据提取

HCD——>libmtp——>jmtpfs——>gvfs-mtp

#IOS交互

#libimobiledevice[-utils]

连idevice pair pair#配对

接ifuse /……/目标目录 #存取ios设备的fuse文件系统攻击。fuse的一个特化，配置亦在/etc/fuse.conf

卸fusermount –u /挂载目录

7~10天

桌面HERE WeGo地图，移动段MAPS.ME。openstreetmap功能不足

startpage、DuckDuckGo搜索

wordpress 博客

四、

B，非对称密钥生成与应用

TLS1.3 Fizz C++14 TLS函数库

“FCN一键接入”私有网络的工具