UNIX TOOLBOX - 中文版

这是一份收集Unix/Linux/BSD命令和任务的文档，它有助于高级用户或IT工作。它是一份简明扼要的实用指南，当然读者应该知道他/她在干什么。

1 系统

硬件 | 状态信息 | 用户 | 限制 | 运行级别 | root 密码 | 编译内核

正在运行的内核和系统信息

```
# uname -a                         # 获取内核版本（和BSD版本）
# lsb_release -a                   # 显示任何 LSB 发行版版本信息
# cat /etc/SuSE-release            # 获取 SuSE 版本
# cat /etc/debian_version          # 获取 Debian 版本
```

使用 /etc/DISTR-release 其中DISTR(发行代号)= lsb (Ubuntu), redhat, gentoo, mandrake, sun (Solaris), 等等。

```
# uptime                           # 显示系统开机运行到现在经过的时间
# hostname                         # 显示系统主机名
# hostname -i                      # 显示主机的 IP 地址
# man hier                         # 描述文件系统目录结构
# last reboot                      # 显示系统最后重启的历史记录
```

1.1 硬件信息

内核检测到的硬件信息

```
# dmesg                            # 检测到的硬件和启动的消息
# lsdev                            # 关于已安装硬件的信息译注：许多 Linux 发行版需要自行安装，如：apt-get install procinfo
# dd if=/dev/mem bs=1k skip=768 count=256 2>/dev/null | strings -n 8 # 读取 BIOS 信息
```

Linux

```
# cat /proc/cpuinfo                # CPU 讯息
# cat /proc/meminfo                # 内存信息
# grep MemTotal /proc/meminfo      # 显示物理内存大小
# watch -n1 'cat /proc/interrupts' # 监控内核处理的所有中断
# free -m                          # 显示已用和空闲的内存信息 (-m 为 MB)译注：包括 SWAP 分区
# cat /proc/devices                # 显示当前核心配置的设备
# lspci -tv                        # 显示 PCI 设备
# lsusb -tv                        # 显示 USB 设备
# lshal                            # 显示所有设备属性列表
# dmidecode                        # 显示从 BIOS 中获取的硬件信息
```

FreeBSD

```
# sysctl hw.model                  # CPU 讯息
```

```
# sysctl hw                          # 得到很多硬件信息
# sysctl vm                          # 虚拟内存使用情况
# dmesg | grep "real mem"            # 物理内存
# sysctl -a | grep mem               # 内核内存的设置和信息
# sysctl dev                         # 显示当前核心配置的设备
# pciconf -l -cv                     # 显示 PCI 设备
# usbdevs -v                         # 显示 USB 设备
# atacontrol list                    # 显示 ATA 设备
```

1.2 显示状态信息

以下的命令有助于找出正在系统中运行着的程序。

```
# top                                # 显示和更新使用 cpu 最多的进程
# mpstat 1                           # 显示进程相关的信息
# vmstat 2                           # 显示虚拟内存的状态信息
# iostat 2                           # 显示 I/O 状态信息(2 秒 间隙)
# systat -vmstat 1                   # 显示 BSD 系统状态信息(1 秒 间隙)
# systat -tcp 1                      # 显示 BSD TCP 连接信息(也可以试试 -ip)
# systat -netstat 1                  # 显示 BSD 当前网络连接信息
# systat -ifstat 1                   # 显示 BSD 当前网卡带宽信息
# systat -iostat 1                   # 显示 BSD CPU 和磁盘使用情况
# tail -n 500 /var/log/messages      # 显示最新500条内核/系统日志的信息
# tail /var/log/warn                 # 显示系统警告信息(看syslog.conf)
```

1.3 用户

```
# id                                 # 显示当前用户和用户组的 ID
# last                               # 列出目前与过去登入系统的用户相关信息译注：单独执行 last 指令，它会读取位于 /var/log 目录下，名称为 wtmp 的文件，并把该给文件的内容记
# who                                # 显示目前登入系统的用户信息
# groupadd admin                     # 建立新组"admin"和添加新用户 colin 并加入 admin 用户组(Linux/Solaris)
# useradd -c "Colin Barschel" -g admin -m colin
# userdel colin                      # 删除用户 colin(Linux/Solaris)
# adduser joe                        # FreeBSD 添加用户 joe(交互式)
# rmuser joe                         # FreeBSD 删除用户 joe(交互式)
# pw groupadd admin                  # 在 FreeBSD 上使用 pw
# pw groupmod admin -m newmember     # 添加新用户到一个组
# pw useradd colin -c "Colin Barschel" -g admin -m -s /bin/tcsh
# pw userdel colin; pw groupdel admin
```

加密过的密码存储在 /etc/shadow (Linux and Solaris) 或 /etc/master.passwd (FreeBSD) 中. 如果手动修改了 master.passwd，需要运行 `# pwd_mkdb -p master.passwd` 来重建数据库。

使用 nologin 来临时阻止所有用户登录(root除外)。用户登录时将会显示 nologin 中的信息。

```
# echo "Sorry no login now" > /etc/nologin       # (Linux)
# echo "Sorry no login now" > /var/run/nologin   # (FreeBSD)
```

1.4 限制

某些应用程序需要设置可打开最大文件和 socket 数量(像代理服务器，数据库)。 默认限制通常很低。

Linux

每 shell/脚本

shell 的限制是受 `ulimit` 支配的。使用 `ulimit -a` 可查看其状态信息。 举个例子，改变可打开最大文件数从 1024 到 10240，可以这么做：

```
# ulimit -n 10240                    # 这只在shell中有用
```

`ulimit` 命令可以使用在脚本中来更改对此脚本的限制。

每 用户/进程

登录用户和应用程序的限制可以在 `/etc/security/limits.conf` 中配置。举个例子：

```
# cat /etc/security/limits.conf
*   hard    nproc   250              # 限制所有用户进程数
asterisk hard nofile 409600          # 限制应用程序可打开最大文件数
```

系统级

用sysctl来设置内核限制。要使其永久，可以在 `/etc/sysctl.conf` 中进行配置。

```
# sysctl -a                          # 显示所有系统限制
# sysctl fs.file-max                 # 显示系统最大文件打开数
# sysctl fs.file-max=102400          # 更改系统最大文件打开数
# cat /etc/sysctl.conf
fs.file-max=102400                   # 在 sysctl.conf 中的永久项
# cat /proc/sys/fs/file-nr           # 在使用的文件句柄数
```

FreeBSD

每 shell/脚本

在 csh 或 tcsh 中使用 `limits` 命令，在 sh 或 bash 中使用 `ulimit` 命令。

每 用户/进程

在 `/etc/login.conf` 中配置登录后的默认限制。未作限制的值为系统最大限制值。

系统级

内核限制同样使用 sysctl 来设置。永久配置，在 `/etc/sysctl.conf` 或 `/boot/loader.conf` 中。其语法与 Linux 相同，只是键值不同。

```
# sysctl -a                          # 显示所有系统限制
# sysctl kern.maxfiles=XXXX          # 最大文件描述符数
kern.ipc.nmbclusters=32768           # 在 /etc/sysctl.conf 中的永久项
kern.maxfiles=65536                  # Squid译注：代理服务器 通常用这个值
kern.maxfilesperproc=32768
kern.ipc.somaxconn=8192              # TCP 列队。apache/sendmail 最好用这个值
# sysctl kern.openfiles              # 在使用的文件描述符数
# sysctl kern.ipc.numopensockets     # 已经开启的 socket 数目
```

详情请看 FreeBSD 手册 11章http://www.freebsd.org/handbook/configtuning-kernel-limits.html。

Solaris

在 /etc/system 中的下列设置，会提高每个进程可以打开最大文件描述符的数量：

```
set rlim_fd_max = 4096          # 一个进程可以打开文件描述符的"硬"限制
set rlim_fd_cur = 1024          # 一个进程可以打开文件描述符的"软"限制
```

1.5 运行级别

Linux

一旦内核加载完成，内核会启动 init 进程，然后运行 rc译注：/etc/rc.d/rc 脚本，之后运行所有属于其运行级别的命令脚本。这些脚本都储存在 /etc/rc.d/rcN.d 中(N代表运行级别)，并且都建立着到 /etc/init.d 子目录中命令脚本程序的符号链接。
默认运行级别配置在 /etc/inittab 中。它通常为 3 或 5：

```
# grep default: /etc/inittab
id:3:initdefault:
```

可以使用 init 来改变当前运行级别。举个例子：

```
# init 5                        # 进入运行级别 5
```

运行级别列表如下：

0 系统停止
1 进入单用户模式(也可以是 S)
2 没有 NFS 特性的多用户模式
3 完全多用户模式(正常操作模式)
4 未使用
5 类似于级别3，但提供 XWindow 系统登录环境
6 重新启动系统

使用 chkconfig 工具控制程序在一个运行级别启动和停止。

```
# chkconfig --list              # 列出所有 init 脚本
# chkconfig --list sshd         # 查看 sshd 在各个运行级别中的启动配置
# chkconfig sshd --level 35 on  # 对 sshd 在级别 3 和 5 下创建启动项
# chkconfig sshd off            # 在所有的运行级别下禁用 sshd
```

Debian 和基于Debian 发行版像 Ubuntu 或 Knoppix 使用命令 update-rc.d 来管理运行级别脚本。默认启动为 2,3,4 和 5，停止为 0,1 和 6。

```
# update-rc.d sshd defaults     # 设置 sshd 为默认启动级别
# update-rc.d sshd start 20 2 3 4 5 . stop 20 0 1 6 .  # 用显示参数
# update-rc.d -f sshd remove    # 在所有的运行级别下禁用 sshd
# shutdown -h now (或者 # poweroff)  # 关闭停止系统
```

FreeBSD

BSD 启动步骤不同于 SysV, 她没有运行级别。她的启动状态(单用户，有或没有 XWindow)被配置在 /etc/ttys中。所有的系统脚本都位于 /etc/rc.d/中，第三方应用程序位于 /usr/local/etc/rc.d/中。service 的启动顺序被配置在 /etc/rc.conf 和/etc/rc.conf.local中。默认行为可在 /etc/defaults/rc.conf 中进行配置。 这些脚本至少响应 start|stop|status.

```
# /etc/rc.d/sshd status
sshd is running as pid 552.
# shutdown now                  # 进入单用户模式
# exit                          # 返回到多用户模式
# shutdown -p now               # 关闭停止系统
# shutdown -r now               # 重新启动系统
```

同样可以使用进程 init 进入下列状态级别。举个例子： # init 6 为重启。

0 停止系统并关闭电源 (信号 USR2)
1 进入单用户模式 (信号 TERM)
6 重新启动 (信号 INT)
c 阻止进一步登录 (信号 TSTP)
q 重新检查 ttys(5) 文件 (信号 HUP)

1.6 重设 root 密码

Linux 方法 1

在引导加载器(lilo 或 grub)中，键入如下启选项：

```
init=/bin/sh
```

内核会挂载 root 分区，进程 init 会启动 bourne shell 而不是 rc，然后是运行级别。使用命令 passwd 设置密码然后重启。别忘了需要在单用户模式下做这些动作。
如果重启后 root 分区被挂载为只读，重新挂在它为读写：

```
# mount -o remount,rw /
# passwd                        # 或者删除 root 密码 (/etc/shadow)
# sync; mount -o remount,ro /   # sync 在重新挂在为只读之前 sync 一下
# reboot
```

FreeBSD 和 Linux 方法 2

FreeBSD 不会让你这么做。解决方案是用其他操作系统(像系统紧急修复光盘)挂载 root 分区，然后更改密码。

- 用 live cd 或安装盘启动进入修复模式后，会得到一个 shell。
- 用 fdisk 查找 root 分区。比如：fdisk /dev/sda
- 挂载它并使用 chroot 命令:

```
# mount -o rw /dev/ad4s3a /mnt
# chroot /mnt                   # 改变程序执行时所参考的根目录位置为 /mnt
# passwd
# reboot
```

1.7 内核模块

Linux

```
# lsmod                         # 列出所有已载入内核的模块
# modprobe isdn                 # 载入 isdn 模块
```

FreeBSD

```
# kldstat                       # 列出所有已载入内核的模块
# kldload crypto                # 载入 crypto 模块
```

1.8 编译内核

Linux

```
# cd /usr/src/linux
# make mrproper                 # 清除所有东西，包括配置文件
# make oldconfig                # 从当前内核配置文件的基础上创建一个新的配置文件
# make menuconfig               # 或者 xconfig (Qt) 或者 gconfig (GTK)
# make                          # 创建一个已压缩的内核映像文件
# make modules                  # 编译模块
# make modules_install          # 安装模块
# make install                  # 安装内核
# reboot
```

FreeBSD

要改变和重建内核，需要拷贝源配置文件然后编辑它。当然也可以直接编辑 GENERIC 文件。

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYKERNEL
# cd /usr/src
# make buildkernel KERNCONF=MYKERNEL
# make installkernel KERNCONF=MYKERNEL
```

要重建完全的操作系统：

```
# make buildworld               # 构建完全的系统，但不是内核
# make buildkernel              # 使用 KERNCONF 配置文件编译内核
# make installkernel
# reboot
# mergemaster -p                # 建立临时根环境并比对系统配置文件
# make installworld
# mergemaster                   # 升级所有配置和其他文件
# reboot
```

对于源的一些小改动，有时候简单的命令就足够了：

```
# make kernel world             # 编译并安装内核和系统
# mergemaster
# reboot
```

2 进程

列表 | 优先级 | 后台/前台 | Top | Kill

2.1 进程列表

PID是每个进程唯一号码。使用 ps 获取所有正在运行的进程列表。

```
# ps -auxefw                    # 所有正在运行进程的详尽列表
```

然而，更典型的用法是使用管道或者 pgrep:

```
# ps axww | grep cron
  586  ??  Is     0:01.48 /usr/sbin/cron -s
# ps aux | grep 'ss[h]'         # Find all ssh pids without the grep pid
# pgrep -l sshd                 # 查找所有进程名中有sshd的进程ID
# echo $$                       # The PID of your shell
# fuser -va 22/tcp              # 列出使用端口22的进程
# fuser -va /home               # 列出访问 /home 分区的进程
# strace df                     # 跟踪系统调用和信号
# truss df                      # 同上(FreeBSD/Solaris/类Unix)
# history | tail -50            # 显示最后50个使用过的命令
```

2.2 优先级

用 renice 更改正在运行进程的优先级。负值是更高的优先级，最小为-20，其正值与 "nice" 值的意义相同译注：进程的优先级通常被称作它的 nice 值。用户只能对自己所有的进程使用renice命令，root用户可以在任何进程上使用renice命令，只有root用户才能提高进程的优先级。

```
# renice -5 586                 # 更强的优先级
586: old priority 0, new priority -5
```

使用 nice 命令启动一个已定义优先级的进程。 正值为低优先级，负值为高优先级。确定你知道 /usr/bin/nice 或者使用 shell 内置命令译注：要查看所有 shell 内置命令，可运行 # info bash builtin(# which nice)。

```
# nice -n -5 top                # 更高优先级(/usr/bin/nice)
# nice -n 5 top                 # 更低优先级(/usr/bin/nice)
# nice +5 top                   # tcsh 内置 nice 命令(同上)
```

nice 可以影响 CPU 的调度，另一个实用命令 ionice译注：此命令仅可工作在2.6.13及以上内核版本上，并且采用了CFQ 的 IO 调度方式。通过 #cat /sys/block/[sh]d[a-z]*/queue/scheduler 命令可以得知你的系统采用了什么样的调度算法 可以调度磁盘 IO。This is very useful for intensive IO application which can bring a machine to its knees while still in a lower priority. 此命令仅可在 Linux (AFAIK) 上使用。你可以选择一个类型(idle - best effort - real time)，它的 man 页很短并有很好的解释。

```
# ionice c3 -p123               # 给 pid 123 设置为 idle 类型
# ionice -c2 -n0 firefox        # 用 best effort 类型运行 firefox 并且设为高优先级
# ionice -c3 -p$$               # 将当前的进程(shell)的磁盘 IO 调度设置为 idle 类型
```

例中最后一条命令对于编译(或调试)一个大型项目会非常有用。每一个运行于此 shell 的命令都会有一个较低的优先级，但并不妨碍这个系统。$$ 是你 shell 的 pid (试试 echo $$)。

2.3 前台/后台

当一个进程在 shell 中已运行，可以使用 [Ctrl]-[Z] (^Z), bg 和 fg 来 调入调出前后台译注：在命令后面加 & 可直接使其在后台运行。。举个例子：启动 2 个进程，调入后台。使用 jobs 列出后台列表，然后再调入一个进程到前台。

```
# ping cb.vu > ping.log
^Z                              # ping 使用 [Ctrl]-[Z] 来暂停(停止)
# bg                            # 调入后台继续运行
# jobs -l                       # 后台进程列表
```

```
[1]  - 36232 Running                 ping cb.vu > ping.log
[2]  + 36233 Suspended (tty output)  top
# fg %2                             # 让进程 2 返回到前台运行
```

使用 nohup 开启一个持续运行的进程直到 shell 被关闭(避免挂断)。

```
# nohup ping -i 60 > ping.log &
```

2.4 Top

top 程序用来实时显示系统中各个进程的运行信息。

```
# top
```

当 top 在运行的时候，按下 h译注：也可以是？键会显示帮助画面。常用键如下：

- **u [用户名]** 只显示属于此用户的进程。使用 + 或者空白可以查看所有用户
- **k [PID]** 结束 PID 进程
- **1译注：数字** 显示所有进程状态信息(只有Linux)
- **R** 将当前排序倒转

2.5 Kill命令与信号

使用 kill 或 killall 终止或发送一个信号给进程。

```
# ping -i 60 cb.vu > ping.log &
[1] 4712
# kill -s TERM 4712                 # 同 kill -15 4712
# killall -1 httpd                  # 发送 HUP 信号终止进程 httpd
# pkill -9 http                     # 发送 TERM 信号终止包含 http 的进程
# pkill -TERM -u www                # 发送 TERM 信号终止 www 所有者进程
# fuser -k -TERM -m /home           # 终止所有访问 /home 的进程(卸载该分区前)
```

下面是一些重要的信号：

1	HUP (挂起)
2	INT (中断)
3	QUIT (退出)
9	KILL (KILL 信号不能被捕捉，不能被忽略。)
15	TERM (软件终止信号)

3 文件系统

磁盘信息 | Boot | 磁盘使用情况 | 已打开的文件 | 挂载/重挂 | 挂载 SMB | 挂载映像文件 | Burn ISO | Create image | Memory disk | Disk performance

3.1 权限

用 chmod 和 chown 更改访问权限和所有权。对于所有用户的默认掩码(umask)可以在 /etc/profile (Linux) 或 /etc/login.conf (FreeBSD) 中修改。其默认掩码(umask)通常为 022。掩码可以和777做减法，从而得到755的权限。

```
1 --x 执行                          # Mode 764 = 执行/读/写 | 读/写 | 读
2 -w- 写                            # |---所有者|---用户组|---其他用户|
4 r-- 读
  ugo=a                             u=所有者, g=用户组, o=其他用户, a=所有用户

# chmod [OPTION] MODE[,MODE] FILE   # MODE 可以是 [ugoa]*([-+=]([rwxXst]))
# chmod 640 /var/log/maillog        # 更改 maillog 访问权限为 -rw-r-----
# chmod u=rw,g=r,o= /var/log/maillog # 同上
# chmod -R o-r /home/*              # 递归去除所有其他用户的可读权限
# chmod u+s /path/to/prog           # 在可执行位设置 SUID (知道你在干什么!当执行一个具有 setuid 权限的文件时，文件的执行过程将具有文件所有者的特权(比如root)。所以，应尽量i
# find / -perm -u+s -print          # 查找所有设置过 SUID 位的程序
# chown user:group /path/to/file    # 改变文件的所有者和文件关联的组
# chgrp group /path/to/file         # 改变文件关联的组
# chmod 640 `find ./ -type f -print` # Change permissions to 640 for all files
# chmod 751 `find ./ -type d -print` # Change permissions to 751 for all directories
```

3.2 磁盘信息

```
# diskinfo -v /dev/ad2              # 显示磁盘信息(扇区/大小) (FreeBSD)
# hdparm -I /dev/sda                # 显示 IDE/ATA 磁盘信息 (Linux)
# fdisk /dev/ad2                    # 显示和修改磁盘分区表
# smartctl -a /dev/ad2              # 显示磁盘检测信息
```

3.3 Boot

FreeBSD

如果新内核不能引导，要引导一个旧内核，停止启动倒计时，做如下动作：

```
# unload
# load kernel.old
# boot
```

3.4 系统挂载点/磁盘使用情况

```
# mount | column -t                 # 显示系统已挂载分区情况
# df                                # 显示磁盘剩余空间和挂载的设备
# cat /proc/partitions              # 显示所有设备的所有分区(Linux)
```

磁盘使用情况

```
# du -sh *                          # 列出当前目录下所有文件夹大小
# du -csh                           # 当前目录下所有目录大小总数
# du -ks * | sort -n -r             # 由大到小排序显示目录大小
# ls -lSr                           # 由小到大显示文件列表
```

3.5 谁打开了那些文件

对于找出哪些文件阻止卸载分区并给出有代表性的错误是有帮助的：

```
# umount /home/
umount: unmount of /home             # 不能卸载，因为有一个文件锁定了 home
   failed: Device busy
```

FreeBSD 和大多数 Unix

```
# fstat -f /home                     # 对于一个挂载点
# fstat -p PID                       # 对于一个应用程序进程 ID
# fstat -u user                      # 对于一个用户
```

查找已打开日志文件(或其他已打开文件)，比如 Xorg：

```
# ps ax | grep Xorg | awk '{print $1}'
1252
# fstat -p 1252
USER     CMD          PID   FD MOUNT      INUM MODE         SZ|DV R/W
root     Xorg        1252 root /             2 drwxr-xr-x     512  r
root     Xorg        1252 text /usr     216016 -rws--x--x  1679848 r
root     Xorg        1252    0 /var     212042 -rw-r--r--    56987  w
```

在 /var 中的只有一个 inum 为 212042 的文件：

```
# find -x /var -inum 212042
/var/log/Xorg.0.log
```

Linux

使用 fuser 或 lsof 在一个挂载点中查找已打开的文件：

```
# fuser -m /home                     # 列出访问 /home 的进程
# lsof /home
COMMAND   PID    USER   FD   TYPE DEVICE    SIZE     NODE NAME
tcsh    29029 eedcoba  cwd    DIR   0,18   12288  1048587 /home/eedcoba (guam:/home)
lsof    29140 eedcoba  cwd    DIR   0,18   12288  1048587 /home/eedcoba (guam:/home)
```

关于一个应用程序：

```
ps ax | grep Xorg | awk '{print $1}'
3324
# lsof -p 3324
COMMAND   PID    USER   FD   TYPE DEVICE    SIZE    NODE NAME
Xorg    3324 root    0w   REG        8,6   56296      12492 /var/log/Xorg.0.log
```

关于单个文件：

```
# lsof /var/log/Xorg.0.log
COMMAND  PID USER   FD   TYPE DEVICE  SIZE  NODE NAME
Xorg    3324 root    0w   REG    8,6 56296 12492 /var/log/Xorg.0.log
```

3.6 挂载/重挂载一个文件系统

举个 cdrom 的例子。如果已经列于 /etc/fstab 中：

```
# mount /cdrom
```

或在 /dev/ 中查找设备，亦或使用 dmesg 命令

FreeBSD

```
# mount -v -t cd9660 /dev/cd0c /mnt  # cdrom
# mount_cd9660 /dev/wcd0c /cdrom     # 另外一个方法
# mount -v -t msdos /dev/fd0c /mnt   # 软驱
```

/etc/fstab 中的一条：

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/acd0               /cdrom          cd9660  ro,noauto       0       0
```

要允许用户做这些，可以这么做：

```
# sysctl vfs.usermount=1  # 或者在 /etc/sysctl.conf 中插入一条 "vfs.usermount=1"
```

Linux

```
# mount -t auto /dev/cdrom /mnt/cdrom   # 典型的 cdrom 挂载命令
# mount /dev/hdc -t iso9660 -r /cdrom   # IDE
# mount /dev/sdc0 -t iso9660 -r /cdrom  # SCSI
```

/etc/fstab 中的条目：

```
/dev/cdrom   /media/cdrom  subfs noauto,fs=cdfss,ro,procuid,nosuid,nodev,exec 0 0
```

用 Linux 挂载一个 FreeBSD 分区

用 fdisk 查找分区号，这通常是 root 分区，但也可能是其他 BSD slice。如果 FreeBSD 有许多 slice，他们不列于同一个 fdisk 分区表中，但可见于 /dev/sda* 或 /dev/hda* 中。

```
# fdisk /dev/sda                     # 查找 FreeBSD 分区
/dev/sda3   *        5357        7905    20474842+  a5  FreeBSD
# mount -t ufs -o ufstype=ufs2,ro /dev/sda3 /mnt
/dev/sda10 = /tmp; /dev/sda11 /usr   # 其他 slice
```

重挂载

不用卸载一个设备来重挂载。 对 fsck 来说是必须的。举个例子：

```
# mount -o remount,ro /              # Linux
# mount -o ro /                      # FreeBSD
```

从 cdrom 拷贝原始数据进一个 iso 映像文件：

```
# dd if=/dev/cd0c of=file.iso
```

3.7 给即时烧录(on-the-fly)添加 swap

假设你需要很多的 swap (即刻)，如一个 2GB 文件 /swap2gb (只限 Linux)。

```
# dd if=/dev/zero of=/swap2gb bs=1024k count=2000
# mkswap /swap2gb                    # 创建交换区
# swapon /swap2gb                    # 激活这个 swap。现在可以使用了
# swapoff /swap2gb                   # 当使用完毕，释放这个 swap
# rm /swap2gb
```

3.8 挂载一个 SMB译注：SMB (Server Message Block,服务器信息块)，又称 CIFS (Common Internet File System,通用Internet文件系统) 共享

假设我们要访问计算机 smbserver 上的名叫 myshare 的 SMB 共享，在 window PC 上键入的地址是 \\smbserver\myshare\。我挂载到 /mnt/smbshare 上。注意 cifs 必须是 IP 或 DNS 名，不是 Windows 名字。

Linux

```
# smbclient -U user -I 192.168.16.229 -L //smbshare/    # 列出共享
# mount -t smbfs -o username=winuser //smbserver/myshare /mnt/smbshare
# mount -t cifs -o username=winuser,password=winpwd //192.168.16.229/myshare /mnt/share
```

此外，mount.cifs 软件包可以存储认证到一个文件中。例如，/home/user/.smb:

```
username=winuser
password=winpwd
```

现在可以像下面那样挂载：

```
# mount -t cifs -o credentials=/home/user/.smb //192.168.16.229/myshare /mnt/smbshare
```

FreeBSD

使用 -I 来获取 IP (或 DNS 名) ；smbserver 是 Windows 名。

```
# smbutil view -I 192.168.16.229 //winuser@smbserver    # 列出共享
# mount_smbfs -I 192.168.16.229 //winuser@smbserver/myshare /mnt/smbshare
```

3.9 挂载镜像文件

Linux loop-back

```
# mount -t iso9660 -o loop file.iso /mnt               # 挂载 CD 镜像文件
# mount -t ext3 -o loop file.img /mnt                  # 用 ext3 文件系统挂载镜像文件
```

FreeBSD

用于存储设备 (如果需要做 # kldload md.ko 动作)：

```
# mdconfig -a -t vnode -f file.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
# umount /mnt; mdconfig -d -u 0                        # 清除 md 设备
```

用于虚拟节点：

```
# vnconfig /dev/vn0c file.iso; mount -t cd9660 /dev/vn0c /mnt
# umount /mnt; vnconfig -u /dev/vn0c                   # 清除 vn 设备
```

Solaris and FreeBSD

用于 loop-back 文件接口或 lofi：

```
# lofiadm -a file.iso
# mount -F hsfs -o ro /dev/lofi/1 /mnt
# umount /mnt; lofiadm -d /dev/lofi/1                  # 清除 lofi 设备
```

3.10 创建并刻录 ISO 镜像文件

这将会拷贝 CD 或者 DVD 的扇区。当不用 conv=notrunc，镜像文件会等于 CD 内容大小而非 CD 容量大小。看下面和 dd 例子。

```
# dd if=/dev/hdc of=/tmp/mycd.iso bs=2048 conv=notrunc
```

使用 mkisofs 把目录中所有文件创建成 CD/DVD 镜像文件。克服文件名限制：-r 开启 Rock Ridge 扩展用于 Unix 系统，-J 开启 Joliet 扩展用于微软系统。-L 允许 ISO9660 文件名第一个字符为句点。

```
# mkisofs -J -L -r -V TITLE -o imagefile.iso /path/to/dir
```

对于 FreeBSD，mkisofs 可以到 port 的 sysutils/cdrtools 中找到。

刻录 ISO 镜像文件

FreeBSD

FreeBSD 默认情况下没有在 ATAPI 驱动上启用 DMA。DMA 可用 sysctl 命令启用，其参数如下，或者在 /boot/loader.conf 中添加如下条目：

```
hw.ata.ata_dma="1"
hw.ata.atapi_dma="1"
```

burncd 用于 ATAPI 驱动(burncd 为基本系统的一部分)，cdrecord (在 sysutils/cdrtools 中)用于 SCSI 驱动。

```
# burncd -f /dev/acd0 data imagefile.iso fixate      # ATAPI 驱动
# cdrecord -scanbus                  # 查找 burner 设备描述符(如 1,0,0)
# cdrecord dev=1,0,0 imagefile.iso
```

Linux

对于 Linux，同样使用 cdrecord 如上文所述。此外，它还可以使用本地 ATAPI 接口查找设备描述符：

```
# cdrecord dev=ATAPI -scanbus
```

然后同上面一样烧录 CD/DVD。

dvd+rw-tools

dvd+rw-toolshttp://fy.chalmers.se/~appro/linux/DVD+RW/ 工具包(FreeBSD: ports/sysutils/dvd+rw-tools)可以做上面的一切，其还包括 growisofs 工具来刻录 CD 或 DVD。本实例所引用的 DVD 设备 /dev/dvd 可能是指向 /dev/scd0 (Linux)的符号连接，或者 /dev/cd0 (FreeBSD)，或者 /dev/rcd0c (NetBSD/OpenBSD)，或者 /dev/rdsk/c0t1d0s2 (Solaris)。对于本实例 FreeBSD 手册 18.7 章http://www.freebsd.org/handbook/creating-dvds.html 上有一份很好的文档。

```
          # -dvd-compat 选项将完结光盘，光盘便不可再附加数据
# growisofs -dvd-compat -Z /dev/dvd=imagefile.iso     # 刻录已存在的 iso 镜像文件
# growisofs -dvd-compat -Z /dev/dvd -J -R /p/to/data  # 直接刻录
```

转换 Nero .nrg 文件成 .iso

Nero 简单的添加了 300KB 的头到一个常规的 iso 镜像文件中。我们可用 dd 工具来去除它。

```
# dd bs=1k if=imagefile.nrg of=imagefile.iso skip=300
```

转换 bin/cue 镜像成 .iso

bchunk 程序http://freshmeat.net/projects/bchunk/可以做到这一点。在 FreeBSD 中，它在 port 的 sysutils/bchunk 中。

```
# bchunk imagefile.bin imagefile.cue imagefile.iso
```

3.11 创建基于文件的镜像文件

举个例子，一个使用文件 /usr/vdisk.img 的 1GB 分区。这里我们使用 vnode 0,但也可为 1。

FreeBSD

```
# dd if=/dev/random of=/usr/vdisk.img bs=1K count=1M
# mdconfig -a -t vnode -f /usr/vdisk.img -u 0         # 创建设备 /dev/md1
# bsdlabel -w /dev/md0
# newfs /dev/md0c
# mount /dev/md0c /mnt
# umount /mnt; mdconfig -d -u 0; rm /usr/vdisk.img    # 清除 md 设备
```

这个基于文件的镜像文件可以在 /etc/rc.conf 和 /etc/fstab 中配置成启动期间自动挂载。可用 `# /etc/rc.d/mdconfig start` (先用 `# mdconfig -d -u 0` 命令删除 md0 设备) 测试你的设置。
需要注意的是，那个自动设置仅工作于这个基于文件的镜像文件不在 root 分区中。原因是 /etc/rc.d/mdconfig 脚本早于启动就执行了，并且 root 分区仍然是只读的。脚本 /etc/rc.d/mdconfig2 之后，镜像文件将位于 root 分区外挂载。
/boot/loader.conf:

```
md_load="YES"
```

/etc/rc.conf:

```
# mdconfig_md0="-t vnode -f /usr/vdisk.img"          # /usr 不在 root 分区中
```

/etc/fstab: (行后的两个 0 0 很重要，它告诉 fsck 忽略这个设备,现在还不存在。)

```
/dev/md0                /usr/vdisk      ufs     rw              0       0
```

也可能在增加镜像文件的大小之后，如增大到 300MB。

```
# umount /mnt; mdconfig -d -u 0
# dd if=/dev/zero bs=1m count=300 >> /usr/vdisk.img
# mdconfig -a -t vnode -f /usr/vdisk.img -u 0
# growfs /dev/md0
# mount /dev/md0c /mnt                              # 文件分区现在为 300MB
```

Linux

```
# dd if=/dev/zero of=/usr/vdisk.img bs=1024k count=1024
# mkfs.ext3 /usr/vdisk.img
# mount -o loop /usr/vdisk.img /mnt
# umount /mnt; rm /usr/vdisk.img                    # 清楚
```

Linux with losetup

/dev/zero 比 urandom 更快，但对于加密来说却不够安全。

```
# dd if=/dev/urandom of=/usr/vdisk.img bs=1024k count=1024
# losetup /dev/loop0 /usr/vdisk.img                 # 创建并联结 /dev/loop0
# mkfs.ext3 /dev/loop0
# mount /dev/loop0 /mnt
# losetup -a                                        # 查看已经挂载的 loop 设备
# umount /mnt
# losetup -d /dev/loop0                             # Detach
# rm /usr/vdisk.img
```

3.12 创建基于内存的文件系统

基于内存的文件系统对于重量级 IO 应用程序来说非常快。怎样创建一个挂载到 /memdisk 的 64M 分区：

FreeBSD

```
# mount_mfs -o rw -s 64M md /memdisk
# umount /memdisk; mdconfig -d -u 0                 # 清除该 md 设备
md      /memdisk      mfs      rw,-s64M    0    0         # /etc/fstab 条目
```

Linux

```
# mount -t tmpfs -osize=64m tmpfs /memdisk
```

3.13 磁盘性能

在 ad4s3c (/home) 分区上读写一个 1GB 的文件。

```
# time dd if=/dev/ad4s3c of=/dev/null bs=1024k count=1000
# time dd if=/dev/zero bs=1024k count=1000 of=/home/1Gb.file
# hdparm -tT /dev/hda      # 仅限 Linux
```

4 网络

路由 | 额外 IP | 更改 MAC 地址 | 端口 | 防火墙 | IP 转发 | NAT | DNS | DHCP | 通信量 | QoS | NIS

4.1 调试（也可看流量分析）

Linux

```
# ethtool eth0              # 显示以太网状态(replaces mii-diag)
# ethtool -s eth0 speed 100 duplex full # 把网卡 eth0 速度改为 100兆/秒，采用全双工
# ethtool -s eth0 autoneg off # 禁用自动协商模式
# ethtool -p eth1           # 闪烁网络接口 LED 灯 - 如果支持的话，非常实用
# ip link show              # 在 Linux 上显示所有网络接口(同 ifconfig 类似)
# ip link set eth0 up       # 使设备激活(或Down掉)。同 "ifconfig eth0 up"
# ip addr show              # 在 Linux 上显示所有 IP 地址(与 ifconfig 类似)
# ip neigh show             # 与 arp -a 类似
```

其他系统

```
# ifconfig fxp0             # 查看 "media" 字段(FreeBSD)
```

```
# arp -a                    # 查看路由(或主机) ARP 条目(所有系统)
# ping cb.vu                # 第一个要试的事情...
# traceroute cb.vu          # 列印到目的地的路由路径
# ifconfig fxp0 media 100baseTX mediaopt full-duplex # 100兆/秒 全双工(FreeBSD)
# netstat -s                # 对每个网络协议做系统级分析
```

另一些命令，虽然不总是默认安装，但很好找：

```
# arping 192.168.16.254     # 在网络层上 Ping
# tcptraceroute -f 5 cb.vu  # 使用 tcp 替换 icmp 来跟踪，透过防火墙
```

4.2 路由

列印路由表

```
# route -n                  # Linux 或使用 "ip route"
# netstat -rn               # Linux, BSD 和 UNIX
# route print               # Windows
```

添加删除路由

FreeBSD

```
# route add 212.117.0.0/16 192.168.1.1
# route delete 212.117.0.0/16
# route add default 192.168.1.1
```

永久的添加路由可在 /etc/rc.conf 配置文件中设置

```
static_routes="myroute"
route_myroute="-net 212.117.0.0/16 192.168.1.1"
```

Linux

```
# route add -net 192.168.20.0 netmask 255.255.255.0 gw 192.168.16.254
# ip route add 192.168.20.0/24 via 192.168.16.254       # 等同于上面命令
# route add -net 192.168.20.0 netmask 255.255.255.0 dev eth0
# route add default gw 192.168.51.254
# ip route add default via 192.168.51.254 dev eth0      # 等同于上面命令
# route delete -net 192.168.20.0 netmask 255.255.255.0
```

Solaris

```
# route add -net 192.168.20.0 -netmask 255.255.255.0 192.168.16.254
# route add default 192.168.51.254 1                    # 1 = 通过此路由跳译注：数据包生存周期依赖于 IP 头中的生存周期(Time-to-Live，简称 TTL)。根据 RFC 的定义，这个域值由f
# route change default 192.168.50.254 1
```

永久条目配置在 /etc/defaultrouter 中。

Windows

```
# Route add 192.168.50.0 mask 255.255.255.0 192.168.51.253
# Route add 0.0.0.0 mask 0.0.0.0 192.168.51.254
```

使用 add -p 来是路由设置永久有效。

4.3 配置额外的 IP 地址

Linux

```
# ifconfig eth0 192.168.50.254 netmask 255.255.255.0       # 第一个 IP
# ifconfig eth0:0 192.168.51.254 netmask 255.255.255.0     # 第二个 IP
# ip addr add 192.168.50.254/24 dev eth0                   # 等价命令
# ip addr add 192.168.51.254/24 dev eth0 label eth0:1
```

FreeBSD

```
# ifconfig fxp0 inet 192.168.50.254/24                     # 第一个 IP
# ifconfig fxp0 alias 192.168.51.254 netmask 255.255.255.0 # 第二个 IP
```

永久条目设置在 /etc/rc.conf 中

```
ifconfig_fxp0="inet 192.168.50.254  netmask 255.255.255.0"
ifconfig_fxp0_alias0="192.168.51.254 netmask 255.255.255.0"
```

Solaris

用 ifconfig -a 命令检查设置

```
# ifconfig hme0 plumb                                      # 启用网卡
# ifconfig hme0 192.168.50.254 netmask 255.255.255.0 up    # 第一个 IP
# ifconfig hme0:1 192.168.51.254 netmask 255.255.255.0 up  # 第二个 IP
```

4.4 更改 MAC 地址

通常在你更改之前先停下网络接口。不要告诉我为什么你想改变 MAC 地址.....

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:05      # Linux
# ifconfig fxp0 link 00:01:02:03:04:05          # FreeBSD
# ifconfig hme0 ether 00:01:02:03:04:05         # Solaris
# sudo ifconfig en0 ether 00:01:02:03:04:05     # Mac OS X Tiger
# sudo ifconfig en0 lladdr 00:01:02:03:04:05    # Mac OS X Leopard
```

对于 Windows 已经有许多工具了。像 etherchangehttp://ntsecurity.nu/toolbox/etherchange。或者看看 "Mac Makeup", "smac"。

4.5 使用中的端口

监听打开的端口：

```
# netstat -an | grep LISTEN
# lsof -i                                       # 列出所有因特网连接(Linux)
# socklist                                      # 列出打开的 socket (Linux)
# sockstat -4                                   # 使用 socket 的应用程序列表(FreeBSD)
# netstat -anp --udp --tcp | grep LISTEN        # Linux
# netstat -tup                                  # 列出活跃的连接(Linux)
# netstat -tupl                                 # 列出系统中正在监听的端口(Linux)
# netstat -ano                                  # Windows
```

4.6 防火墙

检查正在运行的防火墙(只是典型配置)：

Linux

```
# iptables -L -n -v                 # 状态信息
Open the iptables firewall
# iptables -P INPUT       ACCEPT    # 打开所有
# iptables -P FORWARD     ACCEPT
# iptables -P OUTPUT      ACCEPT
# iptables -Z                       # 把所有链的包及字节的计数器清空
# iptables -F                       # 清空所有链
# iptables -X                       # 删除所有链译注：链必须没有被引用
```

FreeBSD

```
# ipfw show                         # 状态信息
# ipfw list 65535 # 如果显示 "65535 deny ip from any to any", 那防火墙已被禁用
# sysctl net.inet.ip.fw.enable=0    # 禁用
# sysctl net.inet.ip.fw.enable=1    # 启用
```

4.7 路由 IP 转发

Linux

查看然后启用 IP 转发：

```
# cat /proc/sys/net/ipv4/ip_forward     # 查看 IP 转发 0=禁用, 1=启用
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

或者编辑 /etc/sysctl.conf：

```
net.ipv4.ip_forward = 1
```

FreeBSD

查看并启用：

```
# sysctl net.inet.ip.forwarding          # 查看 IP 转发 0=禁用, 1=启用
# sysctl net.inet.ip.forwarding=1
# sysctl net.inet.ip.fastforwarding=1    # 专用路由器或防火墙
Permanent with entry in /etc/rc.conf:
gateway_enable="YES"                     # 如果主机是网关则设置为 YES。
```

Solaris

```
# ndd -set /dev/ip ip_forwarding 1       # 查看 IP 转发 0=禁用, 1=启用
```

4.8 NAT - 网络地址转换

Linux

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE  # 激活 NAT
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 20022 -j DNAT \
--to 192.168.16.44:22           # 转发端口 20022 到内部 IP 端口(ssh)
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 993:995 -j DNAT \
--to 192.168.16.254:993:995     # 转发 993-995 范围端口
# ip route flush cache
# iptables -L -t nat            # 查看 NAT 状态信息
```

使用 -D 替换 -A 来删除端口转发。

FreeBSD

```
# natd -s -m -u -dynamic -f /etc/natd.conf -n fxp0
Or edit /etc/rc.conf with:
firewall_enable="YES"           # 设置 YES 来启用防火墙功能
firewall_type="open"            # 防火墙类型(看 /etc/rc.firewall)
natd_enable="YES"               # 启用 natd (如果 firewall_enable == YES)。
natd_interface="tun0"           # 公共的网络接口或要使用的 IP 地址。
natd_flags="-s -m -u -dynamic -f /etc/natd.conf"
```

端口转发：

```
# cat /etc/natd.conf
same_ports yes
use_sockets yes
unregistered_only
# redirect_port tcp insideIP:2300-2399 3300-3399  # 端口范围
redirect_port udp 192.168.51.103:7777 7777
```

4.9 DNS

在 unix 上，对于所有的网络接口的 DNS 条目都存储在 /etc/resolv.conf 文件中。主机域也储存在这个文件中。最小化配置如下：

```
nameserver 78.31.70.238
search sleepyowl.net intern.lab
domain sleepyowl.net
```

检查系统域名：

```
# hostname -d                       # 等同于 dnsdomainname
```

Windows

在 Windows 上，DNS 配置于每个网络接口。要显示配置的 DNS 和清空 DNS 缓存可是使用：

```
# ipconfig /?                       # 显示帮助
# ipconfig /all                     # 显示所有信息包括 DNS
# ipconfig /flushdns                # 清除 DNS 缓存
```

转发查询

Dig 是你测试 DNS 设置的好朋友。举个例子，用于测试的 DNS 服务器为 213.133.105.2 ns.second-ns.de。查看哪个服务器客户端接收应答(简单应答).

```
# dig sleepyowl.net
sleepyowl.net.          600     IN      A       78.31.70.238
```

```
;; SERVER: 192.168.51.254#53(192.168.51.254)
```

路由器 192.168.51.254 应答了，并返回了一条 A 条目(记录)。任何条目都可查询，DNS 服务器可用 @ 来选定：

```
# dig MX google.com
# dig @127.0.0.1 NS sun.com          # 测试本地服务器
# dig @204.97.212.10 NS MX heise.de  # 查询外部
# dig AXFR @ns1.xname.org cb.vu      # 查看区传送(zone transfer)
```

程式 host 也很强大。

```
# host -t MX cb.vu                   # 获取邮件 MX 记录
# host -t NS -T sun.com              # 通过 TCP 连接获取 NS 记录
# host -a sleepyowl.net              # 获取所有
```

反向查询

查找属于一个 IP 地址(in-addr.arpa.)的域名。可用 `dig`, `host` 和 `nslookup` 命令查询：

```
# dig -x 78.31.70.238
# host 78.31.70.238
# nslookup 78.31.70.238
```

/etc/hosts

单个主机可以配置于文件 /etc/hosts 来代替本地正在运行的 `named` 反向域名查询。格式很简单，举个例子：

```
78.31.70.238    sleepyowl.net    sleepyowl
```

对于 hosts 文件和 DNS 查询之间的优先级，可在 `/etc/nsswitch.conf` 和 `/etc/host.conf` 中配置 order 名称解析。这个文件同样存在于 Windows 上，通常在：

```
C:\WINDOWS\SYSTEM32\DRIVERS\ETC
```

4.10 DHCP

Linux

一些发行版(SuSE)使用 dhcpcd 作为客户端。默认网络接口是 eth0。

```
# dhcpcd -n eth0           # 触发更新(并不总是可以工作)
# dhcpcd -k eth0           # 释放并关闭
```

租约(lease)的全部信息存储在：

```
/var/lib/dhcpcd/dhcpcd-eth0.info
```

FreeBSD

FreeBSD (和 Debian) 使用 `dhclient`。要配置一个网络接口(如：bge0)运行：

```
# dhclient bge0
```

租约(lease)的全部信息存储在：

```
/var/db/dhclient.leases.bge0
```

使用

```
/etc/dhclient.conf
```

设置 prepend 选项或强制不同的选项：

```
# cat /etc/dhclient.conf
interface "rl0" {
    prepend domain-name-servers 127.0.0.1;
    default domain-name "sleepyowl.net";
    supersede domain-name "sleepyowl.net";
}
```

Windows

dhcp 租约(lease)使用 `ipconfig` 来更新：

```
# ipconfig /renew          # 更新所有适配器
# ipconfig /renew LAN      # 更新名叫 "LAN" 的适配器
# ipconfig /release WLAN   # 释放名叫 "WLAN" 的适配器
```

是的，这是一个使用简单名称重新命名你的适配器的好主意！

4.11 通信量分析(Traffic analysis)

Bmonhttp://people.suug.ch/~tgr/bmon/ 是一个小的流量监控控制台，而且可以显示不同的网络接口的流量。

用 tcpdump 嗅探(sniff)

```
# tcpdump -nl -i bge0 not port ssh and src \(192.168.16.121 or 192.168.16.54\)
# tcpdump -l > dump && tail -f dump              # 缓冲输出
# tcpdump -i rl0 -w traffic.rl0                  # 把数据报文写入二进制文件
# tcpdump -r traffic.rl0                         # 从文件读取数据报文(也可以使用 ethereal)
# tcpdump port 80                                # 两个经典命令
# tcpdump host google.com
# tcpdump -i eth0 -X port \(110 or 143\)         # 查看端口 110(POP) 或 143(IMAP)的数据报文
# tcpdump -n -i eth0 icmp                        # 只捕获 ping
# tcpdump -i eth0 -s 0 -A port 80 | grep GET     # -s 0 为全部包, -A 为 ASCII
```

另一些重要选项：

- `-A` 显示每个包清晰文本(除了报头)
- `-X` 显示包的 ASCII 文本
- `-l` 使标准输出变为缓冲行形式
- `-D` 显示所有可用网络接口

对于 Windows 可以使用 www.winpcap.org。使用 windump -D 来列出网络接口。

用 nmap 扫描

Nmaphttp://insecure.org/nmap/ 是一个用于 OS 探测的端口扫描工具，她通常在许多发行版上有安装，并且同样可用于 Windows。如果你不扫描你的服务器，骇客们会为你做这些...

```
# nmap cb.vu               # 扫描主机上所有保留的 TCP 端口
```

```
# nmap -sP 192.168.16.0/24 # 找出在 0/24 上主机所使用的 IP译注：通过使用 "-sP" 参数，进行 ping 扫描。缺省情况下，Nmap给每个扫描到的主机发送一个 ICMP echo 和一个 TCP ACK，主
# nmap -sS -sV -O cb.vu    # 做秘密 SYN 扫描来探测系统和系统服务的版本信息
PORT      STATE  SERVICE             VERSION
22/tcp    open   ssh                 OpenSSH 3.8.1p1 FreeBSD-20060930 (protocol 2.0)
25/tcp    open   smtp                Sendmail smtpd 8.13.6/8.13.6
80/tcp    open   http                Apache httpd 2.0.59 ((FreeBSD) DAV/2 PHP/4.
[...]
Running: FreeBSD 5.X
Uptime 33.120 days (since Fri Aug 31 11:41:04 2007)
```

其他非标准但好用的工具有 `hping` (www.hping.org)，她是一个 IP 分组组装/分析器，和 `fping` (fping.sourceforge.net)。fping 可以在一个循环队列(round-robin fashion)中扫描多种主机。

4.12 流量控制(QoS)

流量控制管理着一个网络的队列、流量监控、调度以及其他流量设置(traffic parameters)。以下简单实用的示例使用 Linux 和 FreeBSD 的能力来更好的利用带宽。

上传限制

DSL 或有线调制解调器有一个很长的列队来提高上传吞吐量(upload throughput)。然而用一个快速的设备(如以太网)填充这个列队将大大减少交互性。这就是限制设备上传速度有用的原因，以匹配调制解调器的实际能 力，这可以有效提高交互性。设置大约为 modem 最大速度的 90%。

Linux

给 512K 上传速度的 modem。

```
# tc qdisc add dev eth0 root tbf rate 480kbit latency 50ms burst 1540
# tc -s qdisc ls dev eth0                           # 状态
# tc qdisc del dev eth0 root                        # 删除队列
# tc qdisc change dev eth0 root tbf rate 220kbit latency 50ms burst 1540
```

FreeBSD

FreeBSD 使用 `dummynet` 来控制带宽，其配置工具为 ipfw。Pipe 用来设置限制带宽的单位[K|M]{比特/秒|字节/秒}，0 意味着没有限制。使用同样的 pipe 数字可重新配置它。举个例子，限制上传带宽为 500K。

```
# kldload dummynet                                  # 如有必要加载这个模块
# ipfw pipe 1 config bw 500Kbit/s                   # 创建一个带宽限制的 pipe
# ipfw add pipe 1 ip from me to any                 # 转移所有上传进入这个 pipe
```

服务质量 (Quality of service)

Linux

使用 `tc` 的优先级队列来优化 VoIP。在 voip-info.org 或 www.howtoforge.com 上可以看到完整的例子。假设 VoIP 使用 UDP 端口 10000:11024 并且使用 eth0 设备(也可为 ppp0 或 so)。下列命令定义了三个队列，并且用 QoS `0x1e`(设置所有位) 强制 VOIP 流量到队列 1。默认流量流入队列 3，Qos Minimize-Delay 流入队列 2。

```
# tc qdisc add dev eth0 root handle 1: prio priomap 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 0
# tc qdisc add dev eth0 parent 1:1 handle 10: sfq
# tc qdisc add dev eth0 parent 1:2 handle 20: sfq
# tc qdisc add dev eth0 parent 1:3 handle 30: sfq
# tc filter add dev eth0 protocol ip parent 1: prio 1 u32 \
  match ip dport 10000 0x3C00 flowid 1:1            # 使用服务端端口范围
  match ip dst 123.23.0.1 flowid 1:1                # 或/和使用服务器 IP
```

状态和移除：

```
# tc -s qdisc ls dev eth0                           # queue status
# tc qdisc del dev eth0 root                        # delete all QoS
```

计算端口范围和掩码 (mask)

用你所计算的端口掩码来定义 tc 过滤器的端口范围。查询 2^N 端口范围结尾，推断范围并转换成十六进制。这就是你的掩码 (mask)。例如 10000 -> 11024，它的范围是 1024。

```
# 2^13 (8192) < 10000 < 2^14 (16384)                # 结尾是 2^14 = 16384
# echo "obase=16;(2^14)-1024" | bc                  # 掩码是 0x3C00
```

FreeBSD

假设最大连接带宽为 500Kbit/s，我们使用优先级 100:10:1 定义 3 个队列给 VoIP:ssh:剩余所有。

```
# ipfw pipe 1 config bw 500Kbit/s
# ipfw queue 1 config pipe 1 weight 100
# ipfw queue 2 config pipe 1 weight 10
# ipfw queue 3 config pipe 1 weight 1
# ipfw add 10 queue 1 proto udp dst-port 10000-11024
# ipfw add 11 queue 1 proto udp dst-ip 123.23.0.1 # 或/和使用服务器 IP
# ipfw add 20 queue 2 dsp-port ssh
# ipfw add 30 queue 3 from me to any                # 剩余所有
```

状态和移除：

```
# ipfw list                                         # 规则信息
# ipfw pipe list                                    # 管道信息
# ipfw flush                                        # 删除除默认外所有规则
```

4.13 NIS 调试

一些可工作在已配置好的 NIS 客户端上的命令：

```
# ypwhich                 # 获取提供 NIS 服务的服务器名
# domainname              # 已配置的 NIS 域名
# ypcat group             # 列印 NIS 映射 group
# cd /var/yp && make      # 重建 yp 数据库
```

ypbind 正在运行吗？

```
# ps auxww | grep ypbind
/usr/sbin/ypbind -s -m -S servername1,servername2       # FreeBSD
/usr/sbin/ypbind          # Linux
# yppoll passwd.byname
Map passwd.byname has order number 1190635041. Mon Sep 24 13:57:21 2007
The master server is servername.domain.net.
```

Linux

```
# cat /etc/yp.conf
ypserver servername
domain domain.net broadcast
```

5 SSH SCP

公钥认证 | 指纹 | SCP | 隧道(Tunneling)

5.1 Public key authentication

使用公钥认证而不是密码连接主机。方法是附加你的公钥文件到远程主机。本例中我们用客户端产生的 key **从 host-client 连接到 host-server**。

- 使用 ssh-keygen 生成密钥对。私钥放在 ~/.ssh/id_dsa，公钥在 ~/.ssh/id_dsa.pub。
- 拷贝你的公钥到服务器的 ~/.ssh/authorized_keys2。

```
# ssh-keygen -t dsa -N ''
# cat ~/.ssh/id_dsa.pub | ssh you@host-server "cat - >> ~/.ssh/authorized_keys2"
```

使用来自 ssh.com 的 Windows 客户端

ssh.com 的非商业性版本的客户端可下载自它主 FTP 站点：ftp.ssh.com/pub/ssh/。 用 ssh.com 客户端产生的密钥需要在 OpenSSH 服务器上进行转换。可以使用 ssh-keygen 命令来完成。

- 使用 ssh.com 客户端创建一对密钥：Settings - User Authentication - Generate New....
- 我使用 DSA 密钥类型；密钥长度为 2048。
- 拷贝 ssh.com 客户端产生的公钥到服务器的 ~/.ssh 目录。
- 她的密钥对在 C:\Documents and Settings\%USERNAME%\Application Data\SSH\UserKeys。
- 在服务器上使用 ssh-keygen 转换公钥：

```
# cd ~/.ssh
# ssh-keygen -i -f keyfilename.pub >> authorized_keys2
```

注意： 我们使用 DSA 密钥，使用 RSA 密钥也是可以的。这个密钥不受密码保护。

在 Windows 上使用 Putty

Puttyhttp://www.chiark.greenend.org.uk/~sgtatham/putty/download.html 是一个简单并且自由的(MIT许可)译注：free 不单单是免费 ssh Windows 客户端。

- 使用 puTTYgen 程序创建密钥对。
- 保存密钥对(比如：C:\Documents and Settings\%USERNAME%\.ssh).
- 拷贝公钥到服务器的 ~/.ssh 目录：

```
# scp .ssh/puttykey.pub root@192.168.51.254:.ssh/
```

- 使用 ssh-keygen 在 OpenSSH 服务器上转换这个公钥：

```
# cd ~/.ssh
# ssh-keygen -i -f puttykey.pub >> authorized_keys2
```

- 在 Putty 中设置指向私钥的位置：Connection - SSH - Auth

5.2 检查指纹

在首次连接时，SSH 会请求保存不知道的主机指纹。要避免中间人(man-in-the-middle)攻击，服务器的管理员可以发送密钥指纹给客户端，来让其在首次登陆时验证服务器的真实性。使用 ssh-keygen -l 获取服务器的指纹：

```
# ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub      # RSA 密钥
2048 61:33:be:9b:ae:6c:36:31:fd:83:98:b7:99:2d:9f:cd /etc/ssh/ssh_host_rsa_key.pub
# ssh-keygen -l -f /etc/ssh/ssh_host_dsa_key.pub      # DSA 密钥(默认)
2048 14:4a:aa:d9:73:25:46:6d:0a:48:35:c7:f4:16:d4:ee /etc/ssh/ssh_host_dsa_key.pub
```

现在客户端在连接到服务器时可验证其服务器的真实性：

```
# ssh linda
The authenticity of host 'linda (192.168.16.54)' can't be established.
DSA key fingerprint is 14:4a:aa:d9:73:25:46:6d:0a:48:35:c7:f4:16:d4:ee.
Are you sure you want to continue connecting (yes/no)? yes
```

5.3 安全文件传输

一些简单的命令：

```
# scp file.txt host-two:/tmp
# scp joe@host-two:/www/*.html /www/tmp
# scp -r joe@host-two:/www /www/tmp
```

在 Konqueror 或 Midnight 控制台中，用地址 **fish://user@gate** 来访问远程文件系统是可行的，就是比较慢而已。
此外，也可以用基于 SCP 文件系统客户端的 **sshfs** 来挂载一个远程目录。看 fuse sshfshttp://fuse.sourceforge.net/sshfs.html.

5.4 隧道(Tunneling)

SSH 隧道可以让你通过 SSH 连接进行端口转发(转发/反向隧道)，从而确保了传输及端口访问的安全。它只能工作在 TCP 协议上。通常端口转发命令如下(也可看 ssh 和 NAT 实例)：

```
# ssh -L localport:desthost:destport user@gate  # gate 为目标主机网关
# ssh -R destport:desthost:localport user@gate  # 转发你的 localport 到目标端口
# ssh -X user@gate    # 转发 X 程序
```

这将会连接到 gate 并转发端口到目标主机 desthost:destport。注意 desthost 为 gate 中的目标主机名。因此，如果连接到了 gate，那么 desthost 就是 localhost。也可以做更多的端口转发。

在 gate 上直接转发

假设我们想访问在 gate 上运行的 CVS(2401端口) 和 HTTP(80端口)。下面是个简单的例子，desthost 就是 localhost，我们使用本的端口 8080 代替 80 端口，所以我们不需要 root 权限。一旦 ssh session 打开，二个服务就都可在本地端口访问。

```
# ssh -L 2401:localhost:2401 -L 8080:localhost:80 user@gate
```

转发 Netbios 和远程桌面到第二个服务器

假设有一台在 gate 后面没有运行 ssh 的 Winodws SMB 服务器。我们需要访问 SMB 共享和远程桌面。

```
# ssh -L 139:smbserver:139 -L 3388:smbserver:3389 user@gate
```

现在这个 SMB 共享可以使用 \\127.0.0.1\ 访问，但只能在本地共享关闭的情况下，因为本的共享也是在 139 端口监听的。
保持本的共享也是可行的，因此我们需要为这个通道使用新 IP 地址来新建一个虚拟设备，SMB 共享将会使用此地址连接。此外，本地 RDP 已经在 3389 端口监听了，所以我们选择端口 3388。对于这个例子，让我们使用一个虚拟 IP 地址 10.1.1.1。

- 对于 Putty 上使用源端口=10.1.1.1:139。它可以创建多重回路(multiple loop)设备和通道。在 Windows 2000 上，只有 Putty 为我工作。
- 对于 ssh.com 的客户端，要禁用 "Allow local connections only"。因为 ssh.com 客户端绑定了所有地址，所以只能连接单个共享。

现在用 IP 地址 10.1.1.1 创建回路(loopback)接口：

- # 系统->控制面板->添加硬件 # 是，我已经连接了此硬件(Y) # 添加新的硬件设备(在列表最下面)。
- # 安装我手动选择的硬件 # 网络适配器 # Microsoft，Microsoft Loopback Adapter。
- 配置这个假设备的 IP 地址为 10.1.1.1，掩码 255.255.255.0，没有网关。
- 高级->WINS，开启 LMHOSTS 查询；禁用 TCP/IP 上的 NetBIOS。
- # 启用 Microsoft 网络客户端。# 禁用 Microsoft 网络文件和打印机共享

做完这些之后我有重启。现在用 \\10.1.1.1 连接 SMB 共享和用 10.1.1.1:3388 连接远程桌面。

调试

如果不能工作：

- 端口有没有转发：运行控制台运行 netstat -an 命令并查看有没有 0.0.0.0:139 或者 10.1.1.1:139
- 有没有 telnet 到 10.1.1.1 139？
- 你需要打开 "本地端口接受其他主机连接"。
- "Microsoft 网络文件和打印机共享" 有没有被禁用？

在 NAT 后面连接两个客户端

假设两个客户端在一个 NAT 网关后面，cliadmin 客户端要连接到 cliuser 客户端(目的地)，两者都可用 ssh 登录到正在运行 sshd 的 gate 上。你不需要 root 权限，只要端口大于 1024 即可。我们在 gate 上使用 2022 端口。而且，由于 gate 使用与本地，所以网关端口不是必须的。
开启 cliuser 客户端(从目标到 gate)：

```
# ssh -R 2022:localhost:22 user@gate            # 转发客户端 22 端口到 gate:2022 端口
```

开启 cliadmin 客户端(从主机到 gate)：

```
# ssh -L 3022:localhost:2022 admin@gate         # 转发客户端 3022 端口到 gate:2022 端口
```

现在 admin 可以直接连接 cliuser 客户端：

```
# ssh -p 3022 admin@localhost                   # local:3022 -> gate:2022 -> client:22
```

在 NAT 后面的 VNC 连接

假设一个在 NAT 后面，监听在端口 5900 上可被访问的 Windows VNC 客户端。
开启 cliwin 客户端到 gate：

```
# ssh -R 15900:localhost:5900 user@gate
```

开启 cliadmin 客户端(从主机到 gate)：

```
# ssh -L 5900:localhost:15900 admin@gate
```

现在 admin 直接连接到 VNC 客户端：

```
# vncconnect -display :0 localhost
```

6 使用 SSH 建立 VPN

自 4.3 版开始，OpenSSH 可以使用 tun/tap译注：tun 为虚拟点对点设备，tap 为虚拟以太网设备。 设备来加密一个隧道。其非常类似于基于 TLS 的 VPN 解决方案(像 OpenVPN)。对于 SSH 的一个优势是，她不需要安装和配置额外的软件。另外隧道使用 SSH 认证(像共享密钥)。 其缺点是，对于一个缓慢的连接， 其传输效率较低。并且这个隧道依赖于单个(易断的) TCP 链接。这个技术对于快速设置一个基于 IP 的 VPN 来说非常有用。她对于用单个 TCP 端口转发没有限制，并且在所有 3/4 层 协议像 ICMP、TCP/UDP 等上都可用。不管怎么样，下面这些选择在 sshd_conf 文件中是必须的：

```
PermitRootLogin yes
PermitTunnel yes
```

6.1 单个 P2P 连接

这里，我们用点对点隧道连接 hclient 和 hserver 两个主机。这个连接是从 hclient 开始到 hserver 的，并且是用 root 来做。这个通道的连接点是 10.0.1.1(服务端)和 10.0.1.2(客户端)，然后我们创建设备 tun5(当然也可以是其它数字)。这个过程非常简单：

- 使用 SSH 的通道选项 -w 来连接
- 设置隧道的 IP 地址。服务端和客户端各一次。

连接到服务端

连接始于客户端,然后再服务端执行命令。

Linux上的服务端

```
cli>#  ssh -w5:5 root@hserver
srv>#  ifconfig tun5 10.0.1.1 netmask 255.255.255.252   # 在服务端 shell 上执行
```

FreeBSD上的服务端

```
cli>#  ssh -w5:5 root@hserver
srv>#  ifconfig tun5 10.0.1.1 10.0.1.2                  # 在服务端 shell 上执行
```

连接到客户端

在客户端上执行命令：

```
cli>#  ifconfig tun5 10.0.1.2 netmask 255.255.255.252   # Linux上的客户端
cli>#  ifconfig tun5 10.0.1.2 10.0.1.1                  # FreeBSD上的客户端
```

现在两个主机都连上了，并且可以在任何 3/4 层协议上使用此通道 IP 地址透明的通讯。

6.2 连接两个网络

除上面的 p2p 设置外，一个更有用的是SSH VPN 用两个 gate 连接两个私有网络。假设有这样一个例子，netA 为 192.168.51.0/24 还有 netB 为 192.168.16.0/24。设置过程同上面相似，我们只需要添加 routing。如果 gate 不同于默认网关，那在私有网络接口上必须开启 NAT。
192.168.51.0/24 (netA)|gateA <-> gateB|192.168.16.0/24 (netB)

- 使用隧道选项 -w 连接 SSH。
- 配置隧道的 IP 地址。服务端和客户端各一次。
- 为两个网络添加 routing。
- 如果需要，在 gate 的私有网络接口上开启 NAT。

设置是从 netA 中的 gasteA 开始的.

连接 gateA 到 gateB

连接从 gateA 开始，命令执行于 gateB。

Linux 上的 gateB

```
gateA>$ ssh -w5:5 root@gateB
gateB># ifconfig tun5 10.0.1.1 netmask 255.255.255.252 # 在 gateB 的 shell 中执行
gateB># route add -net 192.168.51.0 netmask 255.255.255.0 dev tun5
gateB># echo 1 > /proc/sys/net/ipv4/ip_forward          # 如果不是默认网关
gateB># iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

FreeBSD 上的 gateB

```
gateA>$ ssh -w5:5 root@gateB                          # 创建 tun5 设备
gateB># ifconfig tun5 10.0.1.1 10.0.1.2               # 在 gateB 的 shell 中执行
gateB># route add 192.168.51.0/24 10.0.1.2
gateB># sysctl net.inet.ip.forwarding=1               # 如果不是默认网关
gateB># natd -s -m -u -dynamic -n fxp0                # 看 NAT
gateA># sysctl net.inet.ip.fw.enable=1
```

配置 gateA

在 gateA 上执行命令：

Linux 上的 gateA

```
gateA># ifconfig tun5 10.0.1.2 netmask 255.255.255.252
gateA># route add -net 192.168.16.0 netmask 255.255.255.0 dev tun5
gateA># echo 1 > /proc/sys/net/ipv4/ip_forward
gateA># iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

FreeBSD 上的 gateA

```
gateA># ifconfig tun5 10.0.1.2 10.0.1.1
gateA># route add 192.168.16.0/24 10.0.1.2
gateA># sysctl net.inet.ip.forwarding=1
gateA># natd -s -m -u -dynamic -n fxp0                # 看 NAT
gateA># sysctl net.inet.ip.fw.enable=1
```

现在两个私有网络都可以通过 SSH VPN 来透明的连接。如果 gate 不是默认网关，那么 IP 转发和 NAT 设置都是必须的。在这种情况下，客户端将不知道在哪里转发响应(response)，并且 NAT 必须是开启的。

7 RSYNC

Rsync 差不多可以代替 cp 和 scp，此外，断点续传是重启有效的。尾部的斜杠也有着不同的意思，请阅读 man 页面.....这里有一些例子：
拷贝目录中所有内容：

```
# rsync -a /home/colin/ /backup/colin/
# rsync -a /var/ /var_bak/
# rsync -aR --delete-during /home/user/ /backup/      # 使用相对路径(看下面)
```

同之前一样，但使用了压缩和网络。Rsync 使用 SSH 进行传输，并且使用 SSH 密钥，如果设置的话。和 SCP 一样使用 ":"。一个典型的拷贝：

```
# rsync -axSRzv /home/user/ user@server:/backup/user/
```

排除在 /home/user/ 中任何 tmp 目录，并且保持相对目录层次结构，远程目录的结构就是 /backup/home/user/。典型的用于备份的命令。

```
# rsync -azR --exclude /tmp/ /home/user/ user@server:/backup/
```

SSH 连接使用端口 20022：

```
# rsync -az -e 'ssh -p 20022' /home/colin/ user@server:/backup/colin/
```

使用 rsync 守护进程(使用"::")是很快的，但没有透过 SSH 加密。 位置 /backup 定义在了配置文件 /etc/rsyncd.conf 中。变量 RSYNC_PASSWORD 可以设置用来免除手动输入密码。

```
# rsync -axSRz /home/ ruser@hostname::rmodule/backup/
# rsync -axSRz ruser@hostname::rmodule/backup/ /home/    # 回拷贝
```

一些重要选项：

-a, --archive	归档模式，等于 -rlptgoD (非 -H)
-r, --recursive	对子目录以递归模式处理
-R, --relative	使用相对路径名
-H, --hard-links	保留硬链结
-S, --sparse	对稀疏文件进行特殊处理以节省DST的空间
-x, --one-file-system	不跨越文件系统边界
--exclude=PATTERN	指定排除不需要传输的文件模式
--delete-during	传输期间删除
--delete-after	传输结束以后再删除

7.1 在 Windows 上使用 Rsync

可以通过 cygwin 或 独立打包的 cwrsyncht tp://sourceforge.net/projects/sereds 来在 Windows 上运行 rsync。这对于自动备份来说非常方便。只装其中一个(不是两个)， 然后添加路径到 Windows 系统变量中：# 控制面板 -> 系统 -> 高级标签，环境变量按钮。编辑 "Path" 添加 rsync 的安装路径，比如：C:\Program Files\cwRsync\bin 或者 C:\cygwin\bin。这可以让 `rsync` 和 `ssh` 可用于 Windows 命令窗口中。

公钥认证

Rsync 是自动使用 SSH 隧道的，因此在服务端使用 SSH 认证。自动备份可免受用户的影响，rsync 命令对于使用 SSH 公钥认证可以不需要密码。
下面所有的命令都可在 windows 控制台中执行。在控制台(开始 -> 运行 -> cmd)中像在 SSH 中描述的那样创建和上传密钥，根据你的情况改变 "user" 和 "server"。如果文件 authorized_keys2 不存在，拷贝 id_dsa.pub 成 authorized_keys2 并上传它。

```
# ssh-keygen -t dsa -N ''                    # 创建密钥对
# rsync user@server:.ssh/authorized_keys2 .  # 从服务器拷贝本地文件
# cat id_dsa.pub >> authorized_keys2         # 或者使用编辑器添加这个公钥
# rsync authorized_keys2 user@server:.ssh/   # 拷贝文件回服务器
# del authorized_keys2                       # 删除本地拷贝
```

现在测试一下(在同一行里面):

```
rsync -rv "/cygdrive/c/Documents and Settings/%USERNAME%/My Documents/" \
'user@server:My\ Documents/'
```

自动备份

使用批处理文件自动备份并添加到任务计划(程序 -> 附件 -> 系统工具 -> 任务计划)。举个例子，创建批处理文件 backup.bat 取代 user@server。

```
@ECHO OFF
REM rsync the directory My Documents
SETLOCAL
SET CWRSYNCHOME=C:\PROGRAM FILES\CWRSYNC
SET CYGWIN=nontsec
SET CWOLDPATH=%PATH%
REM uncomment the next line when using cygwin
SET PATH=%CWRSYNCHOME%\BIN;%PATH%
echo Press Control-C to abort
rsync -av "/cygdrive/c/Documents and Settings/%USERNAME%/My Documents/" \
'user@server:My\ Documents/'
pause
```

8 SUDO

Sudo 可以给用户一些超级用户的权限而不需要 root 密码。Sudo 对于一个服务器和工作站混合的多用户环境来说非常有用。使用 sudo 运行命令：

```
# sudo /etc/init.d/dhcpd restart           # 用 root 权限运行 rc 脚本
# sudo -u sysadmin whoami                  # 使用其他用户运行命令
```

8.1 配置

Sudo 的配置在 /etc/sudoers 中，并且只能用 visudo 编辑译注：并不是说不能用其他编辑器编辑，而是因为 visudo 会对其语法进行严格检查，避免给系统带来严重后果。。其基本语法是(列表是以逗号分隔的)：

```
user hosts = (runas) commands          # 在 /etc/sudoers 中
```

users 一个或多个用户或是%用户组(像 %wheel) 来获得权限
hosts 主机列表(或 ALL)
runas 列出用户以何种身份(或 ALL)来执行命令，放在 ( ) 内！
commands 列出可被 users 以 runas 或 root 权限运行的命令(或 ALL)

另外一些关键字可以定义别名，他们是 User_Alias, Host_Alias, Runas_Alias 和 Cmnd_Alias。这对于一些较大的设置比较有用。下面是 sudoers 例子：

```
# cat /etc/sudoers
# 主机别名
Host_Alias   DMZ     = 212.118.81.40/28
Host_Alias   DESKTOP = work1, work2

# 用户别名 和 runas 别名
User_Alias   ADMINS  = colin, luca, admin
User_Alias   DEVEL   = joe, jack, julia
Runas_Alias  DBA     = oracle,pgsql

# 命令别名，其值为全路径命令
Cmnd_Alias   SYSTEM  = /sbin/reboot,/usr/bin/kill,/sbin/halt,/sbin/shutdown,/etc/init.d/
Cmnd_Alias   PW      = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root # Not root pwd!
Cmnd_Alias   DEBUG   = /usr/sbin/tcpdump,/usr/bin/wireshark,/usr/bin/nmap

# 一个真实的规则
root,ADMINS  ALL     = (ALL) NOPASSWD: ALL    # ADMINS 别名中的用户可做任何事情不需要密码
DEVEL        DESKTOP = (ALL) NOPASSWD: ALL    # 开发人员可在 DESKTOP 别名的主机上做任何事情
DEVEL        DMZ     = (ALL) NOPASSWD: DEBUG  # 开发人员可以在 DMZ 别名的主机上使用 DEBUG 别名中的命令

# 用户 sysadmin 可以在 DMZ 服务器上执行一些命令
sysadmin     DMZ     = (ALL) NOPASSWD: SYSTEM,PW,DEBUG
sysadmin     ALL,!DMZ = (ALL) NOPASSWD: ALL   # 可以在非 DMZ 主机上做任何事情
%dba         ALL     = (DBA) ALL              # 用户组 dba 可以运行 DBA 别名中用户权限的所有命令

# 所有用户可以在 DESKTOP 别名的主机上 挂载/卸载 CD-ROM
ALL          DESKTOP = NOPASSWD: /sbin/mount /cdrom,/sbin/umount /cdrom
```

9 文件加密

9.1 单个文件

加密和解密：

```
# openssl des -salt -in file -out file.des
# openssl des -d -salt -in file.des -out file
```

那个 file 可以是归档文件(tar archive)。

9.2 归档并加密整个目录

```
# tar -cf - directory | openssl des -salt -out directory.tar.des      # 加密
# openssl des -d -salt -in directory.tar.des | tar -x                 # 解密
```

9.3 压缩归档并加密整个目录

```
# tar -zcf - directory | openssl des -salt -out directory.tar.gz.des  # 加密
# openssl des -d -salt -in directory.tar.gz.des | tar -xz             # 解密
```

- 在使用-k mysecretpassword后，des会取消交互式的密码请求。不过，这非常不安全。
- 使用des3代替des来获得更强的加密(Triple-DES Cipher)。这同样会消耗更多的CPU。

9.4 GPG

GnuPG 是众所周知的对邮件或任何数据进行加密和签名的软件。此外，gpg 还提供高级密钥管理系统。此章节只涵盖了文件加密，没有邮件加密、签名或者信任网络(Web-Of-Trust)。
单纯的加密是一个对称式的加密算法(symmetric cipher)。在本例中，文件是用一个秘密来加密的，任何人知道了这个密码都可以对其进行解密，因此就不需要密钥。Gpg 添加后缀 ".gpg" 到已加密的文件名。

```
# gpg -c file                       # 使用密码加密文件
# gpg file.gpg                      # 文件解密(选项 -o 其他文件)
```

使用密钥

对于更详细的请看 GPG 快速上手 http://www.madboa.com/geek/gpg-quickstart 和 GPG/PGP 基础 http://aplawrence.com/Basics/gpg.html，特别是 gnupg 文档http://gnupg.org/documentation。
密钥对(私钥，公钥)为非对称加密技术。 要点如下：

- 你的公钥是用来给别人加密文件的并且只有你作为接收者才可以解密(甚至不是一个人加密的文件也可以解密)。公钥是公开的也就意味着可以分发。
- 用你的密码加密的私钥用来解密用你的公钥加密的文件。这个密钥必须保持安全。因为如果遗失了私钥或者密码，那么所有的文件都是使用你的公钥加密的。
- 多个密钥文件被称为密钥环(keyrings)，她可以包含一个以上的密钥。

首先生成密钥对。使用默认就行，但你至少要输入你的全名、邮件地址和可选注释。该注释对于创建相同的名字和邮件地址的多个密钥来说非常有用。此外，你应该使用"口令(passphrase)"，而不是简单的密码。

```
# gpg --gen-key                     # 这需要一些时间
```

在 Unix 上密钥存储在 ~/.gnupg/ 中，在 Windows 上通常存储在
C:/Documents and Settings/%USERNAME%/Application Data/gnupg/ 中。

```
~/.gnupg/pubring.gpg                # 包含你的公钥和所有其他导入的信息
~/.gnupg/secring.gpg                # 可包含多个私钥
```

常用选项的简短描述：

-e 加密数据
-d 解密数据
-r 为某个收件者加密('全名' 或者 'email@domain')
-a 输出经过 ascii 封装的密钥
-o 指定输出文件

本实例使用'Your Name' 和 'Alice' 作为密钥的 email 或 全名 或 部分名字的参考。举个例子，我可以使用 'Colin' or 'c@cb.vu' 给我的密钥 [Colin Barschel (cb.vu) <c@cb.vu>]。

只用于个人的加密

不需要导出/导入任何密钥，因为你都已经有了。

```
# gpg -e -r 'Your Name' file                # 使用你的公钥加密
# gpg -o file -d file.gpg                   # 解密。使用 -o 指定输出文件
```

用密钥加密-解密

首先你需要导出给别人使用的公钥。并且你需要导入来自 Alice 她所加密文件的公钥。你可以用简单的 ascii 文档或者使用公钥服务器来保存这些密钥。
举个例子，Alice 导出她的公钥，然后你导入它，之后你就可以加密一个文件给她。这个加密文件只有 Alice 可以解密。

```
# gpg -a -o alicekey.asc --export 'Alice'               # Alice 导出她的公钥到 ascii 文件中
# gpg --send-keys --keyserver subkeys.pgp.net KEYID     # Alice 把她的公钥放入一个服务器
# gpg --import alicekey.asc                             # 你导入她的密钥到你的公钥环(pubring)中
# gpg --search-keys --keyserver subkeys.pgp.net 'Alice' # 或者从一个服务器中获取他的公钥
```

一旦这些公钥导入后，加密或解密一个文件会非常简单：

```
# gpg -e -r 'Alice' file                    # 给 Alice 加密文件
# gpg -d file.gpg -o file                   # 解密 Alice 给你的加密文件
```

密钥管理

```
# gpg --list-keys                           # 列出所有公钥并查看其 KEYID
    KEYID 跟在 '/' 后面 比如：pub   1024D/D12B77CE 它的 KEYID 是 D12B77CE
# gpg --gen-revoke 'Your Name'              # 产生一份撤销密钥证书
# gpg --list-secret-keys                    # 列出所有私钥
# gpg --delete-keys NAME                    # 从本的密钥环中删除一个公钥
# gpg --delete-secret-key NAME              # 从本的密钥环中删除一个私钥
# gpg --fingerprint KEYID                   # 显示 KIYID 这个密钥的指纹
# gpg --edit-key KEYID                      # 编辑密钥(比如签名或者添加/删除 email)
```

10 分区加密

Linux with LUKS | Linux dm-crypt only | FreeBSD GELI | FreeBSD 只使用密码

有(许多)其他替代方法来加密磁盘，我只呈现我所知道和使用的方法。请记住，安全只是系统还未经过实际考验而已。入侵者可以轻易通过键盘事件记录密码。此外，当已经加载了分区，其数据是可以自由访问的，并不会阻止入侵者去访问它。

10.1 Linux

这部分我们使用可用于 2.6 内核的 Linux dm-crypt (device-mapper)。在这个实例中，让我们加密 `/dev/sdc1` 分区，它可为任何其他分区、磁盘、USB或者用 `losetup` 创建的基于文件的分区。对于基于文件的分区，我们使用 `/dev/loop0`。看 镜像文件分区。Device mapper 利用标签来标识一个分区。我们使用 `sdc1` 作为此标签，但可以为任何字符串。

dm-crypt with LUKS

LUKS 和 dm-crypt 是较好的加密技术，并且可为同一个分区设置多个口令，更改密码也很方便。可简单输入 `# cryptsetup --help` 来测试 LUKS 是否可用。如果没有显示任何关于 LUKS 的信息，可看下面Without LUKS 的介绍。第一步如果需要的话创建一个分区：`fdisk /dev/sdc`。

创建加密分区

```
# dd if=/dev/urandom of=/dev/sdc1          # 可选
# cryptsetup -y luksFormat /dev/sdc1       # 这破坏了在 sdc1 上的数据
# cryptsetup luksOpen /dev/sdc1 sdc1
# mkfs.ext3 /dev/mapper/sdc1               # 创建 ext3 文件系统
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt
# cryptsetup luksClose sdc1                # Detach 已加密的分区
```

Attach

```
# cryptsetup luksOpen /dev/sdc1 sdc1
# mount -t ext3 /dev/mapper/sdc1 /mnt
```

Detach

```
# umount /mnt
# cryptsetup luksClose sdc1
```

dm-crypt without LUKS

```
# cryptsetup -y create sdc1 /dev/sdc1      # 或任何其他分区像 /dev/loop0
# dmsetup ls                               # 检查一下，将显示：sdc1 (254, 0)
# mkfs.ext3 /dev/mapper/sdc1               # 只有第一次要这么做！
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt/
# cryptsetup remove sdc1                   # Detach 已加密的分区
```

这样做等同于(非 mkfs 部分) re-attach 分区。如果密码不正确，mount 命令将会失败。对于这个例子，只要简单的移除 sdc1 (`cryptsetup remove sdc1`)并重建即可。

10.2 FreeBSD

两个流行的 FreeBSD 磁盘加密模块为 `gbde` 和 `geli`。我现在使用 `geli` 原因是它够快并且它使用加解密硬件加速设备。详情可看 FreeBSD 使用手册 18.6http://www.freebsd.org/handbook/disks-encrypting.html。`geli` 模块必须已编译或加载进内核：

```
options GEOM_ELI
device crypto                                # 内核配置文件中加入这两行
# echo 'geom_eli_load="YES"' >> /boot/loader.conf   # 也可以在系统引导时加载或者做：kldload geom_eli
```

使用密码和密钥

我为一个典型的磁盘加密使用这些设置，其使用了一个口令和一个加密主密钥(master key)的密钥。这意味着你需要密码和生产的密钥 `/root/ad1.key` 来 attach 分区。主密钥存储在这个加密分区中并且不可见。看下面为 USB 或 映像文件的加密设置。

创建加密分区

```
# dd if=/dev/random of=/root/ad1.key bs=64 count=1  # 加密主密钥的密钥
# geli init -s 4096 -K /root/ad1.key /dev/ad1       # 对于磁盘也可用 -s 8192
# geli attach -k /root/ad1.key /dev/ad1             # 将 /dev/ad1 与所生成的密钥 /root/ad1.key 关联
# dd if=/dev/random of=/dev/ad1.eli bs=1m           # 可选，需要很长时间
# newfs /dev/ad1.eli                                # 创建文件系统
# mount /dev/ad1.eli /mnt
```

Attach

```
# geli attach -k /root/ad1.key /dev/ad1
# fsck -ny -t ffs /dev/ad1.eli                      # 检查文件系统
# mount /dev/ad1.eli /mnt
```

Detach

Detach 步骤会在关机时自动完成。

```
# umount /mnt
# geli detach /dev/ad1.eli
```

/etc/fstab

加密分区在 /etc/fstab 中配置成自动加载。系统启动时会询问加密分区的密码。对于本例下列设置是必须的：

```
# grep geli /etc/rc.conf
geli_devices="ad1"
geli_ad1_flags="-k /root/ad1.key"
# grep geli /etc/fstab
/dev/ad1.eli         /home/private             ufs             rw      0       0
```

仅用密码

加密一个 USB stick 或者映像文件使用密码而不是密钥来得更方便。这种情况下，没有必要随身携带额外的密钥文件。所做步骤同上面非常相似，只是不需要密钥文件。让我们来加密一个 1 GB 的映像文件`/cryptedfile`。

```
# dd if=/dev/zero of=/cryptedfile bs=1M count=1000  # 1 GB 文件
# mdconfig -at vnode -f /cryptedfile
# geli init /dev/md0                                # 仅用密码加密
# geli attach /dev/md0
# newfs -U -m 0 /dev/md0.eli
# mount /dev/md0.eli /mnt
# umount /dev/md0.eli
# geli detach md0.eli
```

现在可以把这个映像文件加载成仅需密码的文件系统。

```
# mdconfig -at vnode -f /cryptedfile
# geli attach /dev/md0
# mount /dev/md0.eli /mnt
```

11 SSL 认证

所谓的 SSL/TLS 认证是加密的公钥认证，它由一个公用密钥和私用密钥组成。证书用来认证终端和加密数据的。例如，用在 web 服务器(https)或者邮件服务器(imaps)。

11.1 步骤

- 我们需要一个证书颁发机构来签署我们的证书。这一步通常由供应商提供，如 Thawte、Verisign等。不过，我们也可以创建我们自己的。

- 创建一个证书签发申请(signing request)。这个申请需要一个已经包含所有必需的信息的未签署证书(公共部分)。该证书申请通常发送到认证供应商去签署。这一步同样也在本地机器上创建了私钥。
- 证书颁发机构签署证书。
- 如果有需要，加入证书和密钥到单个文件来给应用程序使用(web 服务器、邮件服务器等) 。

11.2 配置 OpenSSL

我们使用 /usr/local/certs 作为这个例子的目录或者根据你的设置相应的编辑 /etc/ssl/openssl.cnf 文件，因此你知道文件将创建在哪里。以下是 openssl.cnf 的相关部分：

```
[ CA_default ]
dir             = /usr/local/certs/CA       # 保存所有信息的文件夹
certs           = $dir/certs                # 已生成证书的默认保存目录
crl_dir         = $dir/crl                  # 生成的证书撤销列表(CRL)的默认保存目录
database        = $dir/index.txt            # 保存已签发证书的文本数据库文件
```

确保所有目录已经创建

```
# mkdir -p /usr/local/certs/CA
# cd /usr/local/certs/CA
# mkdir certs crl newcerts private
# echo "01" > serial                        # 仅当 serial 不存在时
# touch index.txt
```

11.3 创建一个认证授权

如果你没有来自供应商的认证授权，你必须创建你自己的。如果打算去供应商签署申请，那么这个步骤不是必须的。创建认证授权 (CA)：

```
# openssl req -new -x509 -days 730 -config /etc/ssl/openssl.cnf \
-keyout CA/private/cakey.pem -out CA/cacert.pem
```

11.4 创建证书签发申请

要创建一个新证书(比如给邮件服务器或 web 服务器)，首先用其私钥创建证书申请。如果你的应用程序不支持加密的私钥(比如 UW-IMAP 就不支持)，那么就用 -nodes 来禁用加密。

```
# openssl req -new -keyout newkey.pem -out newreq.pem \
-config /etc/ssl/openssl.cnf
# openssl req -nodes -new -keyout newkey.pem -out newreq.pem \
-config /etc/ssl/openssl.cnf                # 不对这个密钥加密
```

11.5 签署证书

该证书申请由 CA 签发确认，这个步骤通常由供应商完成。注意：在下面命令中替换 "servername" 成你的服务器名称。

```
# cat newreq.pem newkey.pem > new.pem
# openssl ca -policy policy_anything -out servernamecert.pem \
-config /etc/ssl/openssl.cnf -infiles new.pem
# mv newkey.pem servernamekey.pem
```

现在，servernamekey.pem 就是私钥，servernamecert.pem 就为服务器的证书。

11.6 创建联合认证(united certificate)

IMAP 服务器想要私钥和服务器证书在同一个文件中。通常，这还是比较容易处理的，但是该文件要保证安全! Apache 也可以处理好它。创建一个包含证书和密钥的文件 servername.pem 。

- 用文本编辑器打开私钥文件(servernamekey.pem)，并拷贝私钥到 "servername.pem" 文件中去。
- 服务器证书(servernamecert.pem)也做同样的动作。

最后 servername.pem 文件应该看起来像这样：

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDutWy+o/XZ/[...]qK5LqQgT3c9dU6fcR+WuSs6aejdEDDqBRQ
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIERzCCA7CgAwIBAgIBBDANB[...]iG9w0BAQQFADCBxTELMAkGA1UEBhMCREUx
-----END CERTIFICATE-----
```

现在我们的 /usr/local/certs/ 目录中有了这些；

CA/private/cakey.pem (CA 服务器私钥)
CA/cacert.pem (CA 服务器公钥)
certs/servernamekey.pem (服务器私钥)
certs/servernamecert.pem (服务器已签署的证书)
certs/servername.pem (私钥和服务器证书)

要保证私钥的安全!

11.7 查看证书信息

要查看证书信息，只要这么做：

```
# openssl x509 -text -in servernamecert.pem     # 显示证书信息
# openssl req -noout -text -in server.csr       # 显示申请信息
# openssl s_client -connect cb.vu:443           # 检查 web 服务器认证信息
```

12 CVS

服务器设置 | CVS 测试 | SSH 隧道 | CVS 使用

12.1 服务器设置

CVS 环境初始化

决定主 repository 将要创建和重置的 cvs 根目录。比如 /usr/local/cvs (根)：

```
# mkdir -p /usr/local/cvs
# setenv CVSROOT /usr/local/cvs      # 设置 CVSROOT 环境变量(本地)
# cvs init                           # 创建所有初始化 CVS 配置文件
# cd /root
# cvs checkout CVSROOT               # 签出配置文件来修改他们
# cd CVSROOT
edit config ( fine as it is)
# cvs commit config
cat >> writers                       # 创建 writers 文件 (也可为 readers)
colin
^D                                   # 使用 [Control][D] 退出编辑
# cvs add writers                    # 添加文件 writers 进 repository
# cvs edit checkoutlist
# cat >> checkoutlist
writers
^D                                   # 使用 [Control][D] 退出编辑
# cvs commit                         # 提交所有配置更改
```

添加一个 **readers** 文件，如果你要区分读写权限的话。注意： 不要在主 cvs 中直接编辑文件，而应该签出要编辑的文件，修改完成后再签入。我们所做的文件 **writers** 用来定义可写权限。
下面有三种流行的方式去访问 CVS。前两个不需要任何进一步的配置。看 CVSROOT 部分的实例了解如何使用它们：

- 直接本的访问文件系统。用户需要有足够的权限来直接访问 CVS，除了要登录到操作系统，没有进一步的验证。然而这仅对本地 repository 才有用。
- 使用 ext 协议通过 ssh 来远程访问。任何有 ssh shell 账户和在 CVS 服务器上可读写权限的都可直接使用 ext 协议通过 ssh 来访问 CVS，而不需要任何额外的隧道。没有服务器来处理运行在 CVS 上的验证工作。ssh 登录会去验证。
- 用 pserver 来远程访问。这是对于有较大用户量的首选方法，用户由 CVS 的 pserver 通过一个专门的密码数据库来验证，因此不需要本地用户帐户。这种设置在下面会有说明。

用 inetd 设置网络

如果不需要网络访问，CVS 可以运行于本地。对于远程访问，在 /etc/inetd.conf (Suse 为 /etc/xinetd.d/cvs)中配置如下行，可让守护进程 inetd 启动 pserver：

```
cvspserver      stream  tcp  nowait  cvs  /usr/bin/cvs  cvs \
--allow-root=/usr/local/cvs pserver
```

这是个用来阻断从 internet 访问 cvs 端口的好方法，可使用 ssh 隧道来远程的访问 repository。

单独认证

CVS 用户可能不是操作系统的一部分(即不是本地用户)。这其实可从安全的角度去看。简单的添加一个叫 **passwd** (in the CVSROOT directory) 的文件，其包含 crypt 格式的用户登录名和密码。这也可以使用 apache 的 htpasswd 工具来完成。
注意：这个 passwd 文件仅仅是文件，可以在 CVSROOT 中直接编辑。它不能被签出。更多信息请用 htpasswd --help

```
# htpasswd -cb passwd user1 password1  # -c 创建文件
# htpasswd -b passwd user2 password2
```

现在添加 `:cvs` 到每行的结尾处，用来告诉 cvs 服务器更改用户到 cvs (或任何你正在运行的 cvs 服务器下)。它看起来像这样：

```
# cat passwd
user1:xsFjhU22u8Fuo:cvs
user2:vnefJOsnnvToM:cvs
```

12.2 测试它

测试作为一般用户登录(比如我)

```
# cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs login
Logging in to :pserver:colin@192.168.50.254:2401/usr/local/cvs
CVS password:
```

CVSROOT 变量

这是个环境变量用来指定 repository 的位置。对于本地使用，该变量只需设置成 repository 的目录。对于通过网络使用，传输协议必须指定。使用 `setenv CVSROOT string` (csh, tcsh shell) 或者 `export CVSROOT=string` (sh, bash shell) 设置 CVSROOT 环境变量。

```
# setenv CVSROOT :pserver:<username>@<host>:/cvsdirectory
For example:
# setenv CVSROOT /usr/local/cvs                          # 仅限本的使用
# setenv CVSROOT :local:/usr/local/cvs                   # 同上
# setenv CVSROOT :ext:user@cvsserver:/usr/local/cvs      # 通过 SSH 直接访问
# setenv CVS_RSH ssh                                     # ext 协议访问
# setenv CVSROOT :pserver:user@cvsserver.254:/usr/local/cvs   # 通过 pserver 网络访问
```

一旦登录成功就可导入一个新项目进 repository：**cd 进入**你的项目根目录

```
cvs import <module name> <vendor tag> <initial tag>
cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs import MyProject MyCompany START
```

在 repository 中有个名叫 MyProject 新项目(之后用来签出)。CVS 会导入当前目录的内容进新项目。

签出：

```
# cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs checkout MyProject
或者
# setenv CVSROOT :pserver:colin@192.168.50.254:/usr/local/cvs
# cvs checkout MyProject
```

12.3 通过 SSH 隧道访问 CVS

我们需要两个 shell 来做这个。在第一个 shell 中，我们连接到 cvs 服务器并对 cvs 连接进行端口转发(port-forward)。在第二个 shell 中，我们就像在本地一样使用 cvs。
在 shell 1:

```
# ssh -L2401:localhost:2401 colin@cvs_server   # 直接连接到 cvs 服务器。或：
# ssh -L2401:cvs_server:2401 colin@gateway     # 使用一个网关间接连接到 cvs 服务器
```

在 shell 2:

```
# setenv CVSROOT :pserver:colin@localhost:/usr/local/cvs
# cvs login
Logging in to :pserver:colin@localhost:2401/usr/local/cvs
CVS password:
# cvs checkout MyProject/src
```

12.4 CVS 命令及其使用

导入

该 import 命令用来添加整个目录，它必须运行于要导入的目录中。比如，目录 /devel/ 包含的所有文件和子目录要导入。该目录名在 CVS 中(模块)将会称为 "myapp"。

```
# cvs import [options] directory-name vendor-tag release-tag
# cd /devel                          # 必须在该目录中来导入
# cvs import myapp Company R1_0      # 修订(release)标签可以为任何单个单词
```

在添加了一个新目录 "/devel/tools/" 后，也可这么导入。

```
# cd /devel/tools
# cvs import myapp/tools Company R1_0
```

签出、更新和提交

```
# cvs co myapp/tools                 # 仅会签出 tools 目录
# cvs co -r R1_1 myapp               # 签出修订版本为 R1_1 的 myapp (sticky)
# cvs -q -d update -P                # 典型的 CVS 更新
# cvs update -A                      # 重置所有 sticky 标签(或日期、选项)
# cvs add newfile                    # 添加一个新文件
# cvs add -kb newfile                # 添加一个二进制文件
# cvs commit file1 file2             # 仅提交这两个文件
# cvs commit -m "message"            # 提交所有更改并为这个更改添加日志消息
```

创建一个 patch

It is best to create and apply a patch from the working development directory related to the project, or from within the source directory.

```
# cd /devel/project
# diff -Naur olddir newdir > patchfile # Create a patch from a directory or a file
# diff -Naur oldfile newfile > patchfile
```

应用一个 patch

Sometimes it is necessary to strip a directory level from the patch, depending how it was created. In case of difficulties, simply look at the first lines of the patch and try -p0, -p1 or -p2.

```
# cd /devel/project
# patch --dry-run -p0 < patchfile    # Test the path without applying it
# patch -p0 < patchfile
# patch -p1 < patchfile              # strip off the 1st level from the path
```

13 SVN

Server setup | SVN+SSH | SVN over http | SVN usage

Subversion (SVN)http://subversion.tigris.org/ is a version control system designed to be the successor of CVS (Concurrent Versions System). The concept is similar to CVS, but many shortcomings where improved. See also the SVN bookhttp://svnbook.red-bean.com/en/1.4/.

13.1 Server setup

The initiation of the repository is fairly simple (here for example `/home/svn/` must exist):

```
# svnadmin create --fs-type fsfs /home/svn/project1
```

Now the access to the repository is made possible with:

- `file://` Direct file system access with the svn client with. This requires local permissions on the file system.
- `svn://` or `svn+ssh://` Remote access with the svnserve server (also over SSH). This requires local permissions on the file system.
- `http://` Remote access with webdav using apache. No local users are necessary for this method.

Using the local file system, it is now possible to import and then check out an existing project. Unlike with CVS it is not necessary to cd into the project directory, simply give the full path:

```
# svn import /project1/ file:///home/svn/project1/trunk -m 'Initial import'
# svn checkout file:///home/svn/project1
```

The new directory "trunk" is only a convention, this is not required.

Remote access with ssh

No special setup is required to access the repository via ssh, simply replace `file://` with `svn+ssh/hostname`. For example:

```
# svn checkout svn+ssh://hostname/home/svn/project1
```

As with the local file access, every user needs an ssh access to the server (with a local account) and also read/write access. This method might be suitable for a small group. All users could belong to a subversion group which owns the repository, for example:

```
# groupadd subversion
# groupmod -A user1 subversion
# chown -R root:subversion /home/svn
# chmod -R 770 /home/svn
```

Remote access with http (apache)

Remote access over http (https) is the only good solution for a larger user group. This method uses the apache authentication, not the local accounts. This is a typical but small apache configuration:

```
LoadModule dav_module         modules/mod_dav.so
LoadModule dav_svn_module     modules/mod_dav_svn.so
LoadModule authz_svn_module   modules/mod_authz_svn.so    # Only for access control

<Location /svn>
  DAV svn
  # any "/svn/foo" URL will map to a repository /home/svn/foo
  SVNParentPath /home/svn
  AuthType Basic
  AuthName "Subversion repository"
  AuthzSVNAccessFile /etc/apache2/svn.acl
  AuthUserFile /etc/apache2/svn-passwd
  Require valid-user
</Location>
```

The apache server needs full access to the repository:

```
# chown -R www:www /home/svn
```

Create a user with htpasswd2:

```
# htpasswd -c /etc/svn-passwd user1  # -c creates the file
```

Access control svn.acl example

```
# Default it read access. "* =" would be default no access
[/]
* = r
[groups]
project1-developers = joe, jack, jane
# Give write access to the developers
[project1:]
@project1-developers = rw
```

13.2 SVN commands and usage

See also the Subversion Quick Reference Cardhttp://www.cs.put.poznan.pl/csobaniec/Papers/svn-refcard.pdf. Tortoise SVNhttp://tortoisesvn.tigris.org is a nice Windows interface.

Import

A new project, that is a directory with some files, is imported into the repository with the `import` command. Import is also used to add a directory with its content to an existing project.

```
# svn help import                              # Get help for any command
    # Add a new directory (with content) into the src dir on project1
# svn import /project1/newdir http://host.url/svn/project1/trunk/src -m 'add newdir'
```

Typical SVN commands

```
# svn co http://host.url/svn/project1/trunk      # Checkout the most recent version
    # Tags and branches are created by copying
# svn mkdir http://host.url/svn/project1/tags/   # Create the tags directory
# svn copy -m "Tag rc1 rel." http://host.url/svn/project1/trunk \
                             http://host.url/svn/project1/tags/1.0rc1
# svn status [--verbose]                         # Check files status into working dir
# svn add src/file.h src/file.cpp                # Add two files
# svn commit -m 'Added new class file'           # Commit the changes with a message
# svn ls http://host.url/svn/project1/tags/      # List all tags
# svn move foo.c bar.c                           # Move (rename) files
# svn delete some_old_file                       # Delete files
```

14 实用命令

less | vi | mail | tar | dd | screen | find | 混杂的

14.1 less

`less` 命令用来在控制台中分屏显示文本文档。它在许多发行版中可用。

```
# less unixtoolbox.xhtml
```

一些重要指令(^N 代表 [control]-[N])：

h H 显示指令的汇总列表
f ^F ^V SPACE 向前滚动一屏(或者 N 行)
b ^B ESC-v 向后滚动一屏(或者 N 行)
F 向前滚动；类似于"tail -f"
/pattern 向前搜索匹配该模式的行
?pattern 向后搜索匹配该模式的行
n 重复之前的搜索
N 反方向重复之前的搜索
q 退出

14.2 vi

Vi 在任何 Linux/Unix 发行安装版(gentoo 没有?)上都存在。因此，我们有必要了解一些基本的命令。Vi 有两个模式：命令模式和插入模式。使用 **[ESC]** 键可进入命令模式，使用 **i** 键可进入插入模式。如果你迷失了，可在命令模式下键入 `: help`。
编辑器 `nano` 和 `pico` 通常也都可用，而且更容易(IMHO)使用。

Quit

:w newfilename 保存文件为 newfilename
:wq or :x 保存并退出
:q! 退出但不保存

移动和查找

/string 向前查找 string
?string 向后查找 string
n 同方向重复上一次搜索命令
N 反方向重复上一次搜索命令
{ 光标移至段落结尾
} 光标移至段落开头
1G 光标移至文件的第一行首
nG 光标移至文件的第 n 行首
G 光标移至文件的最后一行首
:%s/OLD/NEW/g 替换所有查找到的 OLD 为 NEW

删除文本

dd 删除当前行
D 删除光标到当前行末尾的字符
dw 删除单词
x 删除字符
u 回复上一次操作
U 回复所有此行的更改

14.3 mail

mail 命令是一个读取和发送邮件的应用程序，她通常已安装。要发送一封邮件，可以简单的输入 "mail user@domain"。其第一行为主题，然后是邮件内容。在一个新行中使用单个点(.)来结束并发送邮件。例子：

```
# mail c@cb.vu
Subject: Your text is full of typos
"For a moment, nothing happened. Then, after a second or so,
nothing continued to happen."
.
EOT
#
```

这同样可用于管道：

```
# echo "This is the mail body" | mail c@cb.vu
```

也是测试邮件服务器的简单方法。

14.4 tar

命令 tar (磁带存档) 可以为文件和目录创建档案。归档文件 .tar 是未压缩的，一个压缩过的归档文件的后缀是 .tgz 或 .tar.gz (zip) 或者 .tbz (bzip2)。不要使用绝对路径建立一个归档文件，你可能要解开这个归档文件到某个地方。一些常用命令如下：

创建

```
# cd /
# tar -cf home.tar home/        # 归档整个 /home 目录(c 为创建)
# tar -czf home.tgz home/       # 等同于 zip 压缩
# tar -cjf home.tbz home/       # 等同于 bzip2 压缩
```

从一个目录树中只包含一个(或2个)目录，并保持相对目录结构。举个例子，/usr/local/etc 和 /usr/local/www，它们在归档文件中的第一层目录是 local/。

```
# tar -C /usr -czf local.tgz local/etc local/www
# tar -C /usr -xzf local.tgz    # 释放 local 目录到 /usr
# cd /usr; tar -xzf local.tgz   # 同上面一样
```

释放(Extract)

```
# tar -tzf home.tgz             # 列出归档文件中的所有文件，并不释放
# tar -xf home.tar              # 释放归档文件(x 为释放)
# tar -xzf home.tgz             # 等同于 zip 压缩
# tar -xjf home.tgz             # 等同于 bzip2 压缩
# tar -xjf home.tgz home/colin/file.txt    # 释放单个文件
```

更高级的

```
# tar c dir/ | gzip | ssh user@remote 'dd of=dir.tgz' # 归档压缩 dir/ 目录并存储到远程主机上
# tar cvf - `find . -print` > backup.tar              # 归档当前目录
# tar -cf - -C /etc . | tar xpf - -C /backup/etc     # 拷贝目录
# tar -cf - -C /etc . | ssh user@remote tar xpf - -C /backup/etc      # 远程拷贝
# tar -czf home.tgz --exclude '*.o' --exclude 'tmp/' home/
```

14.5 dd

程序 dd (磁盘备份(disk dump) 或 destroy disk，也可看 dd 的含义) 用来拷贝分区、磁盘或者其它拷贝。通常这么用：

```
# dd if=<source> of=<target> bs=<byte size> conv=<conversion>
```

重要的 conv 选项：

notrunc 不截短输出文件
noerror 出错时不停止处理(e.g. 坏扇区)
sync 把每个输入块填充到ibs个字节，不足部分用空(NUL)字符补齐

默认字节大小为 512 (一个扇区)。MBR 处于磁盘的第一个扇区，之后的 63 个扇区是空的。较大的字节大小可以加快拷贝速度但也需要更多的内存。

备份和恢复

```
# dd if=/dev/hda of=/dev/hdc bs=16065b                # 拷贝磁盘到磁盘(相同大小)
# dd if=/dev/sda7 of /home/root.img bs=4096 conv=notrunc,noerror # 备份 /
# dd if /home/root.img of=/dev/sda7 bs=4096 conv=notrunc,noerror # 恢复 /
# dd bs=1M if=/dev/ad4s3e | gzip -c > ad4s3e.gz                 # 压缩备份
# gunzip -dc ad4s3e.gz | dd of=/dev/ad0s3e bs=1M                # 解压恢复
# dd bs=1M if=/dev/ad4s3e | gzip | ssh eedcoba@fry 'dd of=ad4s3e.gz' # 也可为远程的
# gunzip -dc ad4s3e.gz | ssh eedcoba@host 'dd of=/dev/ad0s3e bs=1M'
# dd if=/dev/ad0 of=/dev/ad2 skip=1 seek=1 bs=4k conv=noerror    # 忽略 MBR
    # 如果目标(ad2)比较小，这是必须的。
```

恢复

该 dd 命令会读取分区的每一个区块，即所有区块。对于有问题的区块，最好使用 conv=sync,noerror 选项，dd 将会跳过坏的区块并入 0。因此，这就是设置块大小等于或小于磁盘块大小的重要性。1k 大小似乎安全，用 bs=1k 来设置它。假如一个磁盘有坏扇区并且有个分区的数据要恢复，那么用 dd 工具创建一个镜像文件，挂载这个镜像文件，然后拷贝内容到新的磁盘中。如果用了 noerror 选项，dd 会跳过坏扇区并写入 0，也即坏扇区中的内容会丢失。

```
# dd if=/dev/hda of=/dev/null bs=1m                   # 检查坏扇区
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc | gzip | ssh \ # 发送到远程
root@fry 'dd of=hda1.gz bs=1k'
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc of=hda1.img    # 存储为一个映像文件
# mount -o loop /hda1.img /mnt                        # 挂载这个映像文件
# rsync -ax /mnt/ /newdisk/                           # 拷贝到一个新磁盘
# dd if=/dev/hda of=/dev/hda                          # 刷新磁状态
  # 上面的命令对于刷新磁盘(refresh disk)很有用。这绝对安全，但必须先卸载磁盘。
```

删除

```
# dd if=/dev/zero of=/dev/hdc                    # 删除全部数据
# dd if=/dev/urandom of=/dev/hdc                 # 更好的删除全部数据译注：/dev/urandom 设备文件提供了一种比单独使用$RANDOM更好的,能产生更"随机"的随机数的方法。
# kill -USR1 PID                                 # 查看 dd 进度(仅Linux!)
```

MBR 技巧

MBR 包含了引导程序和分区表，它的大小为 512 字节。前 446 字节为引导程序，446 到 512 字节为分区表。

```
# dd if=/dev/sda of=/mbr_sda.bak bs=512 count=1                 # 完全备份 MBR
# dd if=/dev/zero of=/dev/sda bs=512 count=1                    # 删除 MBR 和分区表
# dd if=/mbr_sda.bak of=/dev/sda bs=512 count=1                 # 完全恢复MBR
# dd if=/mbr_sda.bak of=/dev/sda bs=446 count=1                 # 仅回复引导程序
# dd if=/mbr_sda.bak of=/dev/sda bs=1 count=64 skip=446 seek=446 # 恢复分区表
```

14.6 screen

Screen 提供了两个主要功能：

- 在一个终端内运行多个终端会话(terminal session)。
- 一个已启动的程序与运行它的真实终端分离的，因此可运行于后台。真实的终端可以被关闭，还可以在稍后再重新接上(reattached)。

简短实例

开启 screen：

```
# screen
```

在 screen 会话中，我们可以开启一个长时间运行的程序(如 top)。Detach 这个终端，之后可以从其他机器 reattach 这个相同的终端(比如通过 ssh)。

```
# top
```

现在用 **Ctrl-a Ctrl-d** 来 detach。Reattach 终端：

```
# screen -r
```

或更好的：

```
# screen -R -D
```

现在 attach 到这里。具体意思是：先试图恢复离线的 screen 会话。若找不到离线的 screen 会话，即建立新的 screen 会话给用户。

Screen 命令 (在 screen 中)

所有命令都以 **Ctrl-a** 开始。

- **Ctrl-a ?** 各功能的帮助摘要
- **Ctrl-a c** 创建一个新的 window (终端)
- **Ctrl-a Ctrl-n 和 Ctrl-a Ctrl-p** 切换到下一个或前一个 window
- **Ctrl-a Ctrl-N** N 为 0 到 9 的数字，用来切换到相对应的 window
- **Ctrl-a "** 获取所有正在运行的 window 的可导航的列表
- **Ctrl-a a** 清楚错误的 Ctrl-a
- **Ctrl-a Ctrl-d** 断开所有会话，会话中所有任务运行于后台
- **Ctrl-a x** 用密码锁柱 screen 终端

当程序内部运行终端关闭并且你登出该终端时，该 screen 会话就会被终止。

14.7 Find

一些重要选项：

```
-x (BSD) -xdev (Linux)      留于同一文件系统 (fstab 中的 dev)
-exec cmd {} \;             执行命令并用全路径替换 {}
-iname                      同 -name 一样，但不区分大小写
-ls                         显示关于文件的信息(同 ls -la)
-size n                     n 为 +-n (k M G T P)
-cmin n                     查找系统中最后 n 分钟改变文件状态的文件
```

```
# find . -type f ! -perm -444        # 寻找所有无法读取的文件
# find . -type d ! -perm -111        # 寻找所有无法访问的目录
# find /home/user/ -cmin 10 -print   # 寻找最后 10 分钟创建或修改的文件
# find . -name '*.[ch]' | xargs grep -E 'expr' # 在当前目录及子目录搜索 'expr' 表达式
# find / -name "*.core" | xargs rm   # 寻找 core 垃圾并删除它们(也可试试 core.*)
# find / -name "*.core" -print -exec rm {} \;  # 另一种语法
      # 寻找图像文件并创建一个归档文件，iname 为不区分大小写。-r 为附加
# find . \( -iname "*.png" -o -iname "*.jpg" \) -print -exec tar -rf images.tar {} \;
# find . -type f -name "*.txt" ! -name README.txt -print  # 除 README.txt 的文件
# find /var/ -size +10M -exec ls -lh {} \;     # 查找 > 10 MB 的文件
# find /var/ -size +10M -ls           # 这个更简单
# find . -size +10M -size -50M -print
# find /usr/ports/ -name work -type d -print -exec rm -rf {} \;  # 清理 port
      # 以 SUID 查找文件；这些文件很脆弱，必须保持安全。
# find / -type f -user root -perm -4000 -exec ls -l {} \;
```

小心 xarg 或 exec，因为当文件或目录中包含空格时可能会返回错误的结果。在有疑惑时用 "-print0 | xargs -0" 代替 "| xargs"。选项 -print0 必须在 find 命令的最后。看这个不错的 find 迷你教程http://www.hccfl.edu/pollock/Unix/FindCmd.htm.

```
# find . -type f | xargs ls -l       # 不能工作于有空格的名字
# find . -type f -print0 | xargs -0 ls -l  # 可工作于有空格的名字
# find . -type f -exec ls -l '{}' \; # 或使用用于 -exec 的引用 '{}'
```

14.8 混杂的

```
# which command                      # 显示命令的全路径名
# time command                       # 显示一个命令执行完成所用的时间
# time cat                           # 使用 time 作为秒表，用 Ctrl-c 来停止
# set | grep $USER                   # 列显当前环境变量
# cal -3                             # 显示三个月日历
# date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
# date 10022155                      # 设置日期和时间
# whatis grep                        # 显示命令的简短信息
# whereis java                       # 查询命令的的路径和标准目录
```

```
# setenv varname value                # 设置环境变量，设置变量 varname 的值为 value (csh/tcsh)
# export varname="value"              # 设置环境变量，设置变量 varname 的值为 value  (sh/ksh/bash)
# pwd                                 # 显示当前工作目录
# mkdir -p /path/to/dir               # 如果存在不显示错误，建立所需的上级目录
# mkdir -p project/{bin,src,obj,doc/{html,man,pdf},debug/some/more/dirs}
# rmdir /path/to/dir                  # 移除目录
# rm -rf /path/to/dir                 # 移除目录和其内容(强制)
# cp -la /dir1 /dir2                  # 存档、硬连接目录所有文件，用来替代拷贝
# cp -lpR /dir1 /dir2                 # 同上 (FreeBSD)
# cp unixtoolbox.xhtml{,.bak}         # 拷贝文件成新扩展名的快速方法
# mv /dir1 /dir2                      # 修改目录名
```

15 软件安装

15.1 列出已安装过的软件包

```
# rpm -qa                             # 列出已安装过的软件包(RH, SuSE, 基于 RPM 的)
# dpkg -l                             # Debian, Ubuntu
# pkg_info                            # 列出所有已安装过的软件包(FreeBSD)
# pkg_info -W smbd                    # 查看 smbd 安装了那些软件包(FreeBSD)
# pkginfo                             # Solaris
```

15.2 添加/删除软件

前端界面：SuSE 为 yast2/yast，Red Hat 为 redhat-config-packages。

```
# rpm -i pkgname.rpm                  # 安装软件包(RH, SuSE, 基于 RPM 的)
# rpm -e pkgname                      # 删除软件包
```

Debian

```
# apt-get update                      # 更新源列表
# apt-get install emacs               # 安装 emacs 软件包
# dpkg --remove emacs                 # 删除 emacs 软件包
# dpkg -S file                        # 查找拥有该 file 的软件包
```

Gentoo

Gentoo 使用 emerge 作为 "Portage" 软件包管理系统的核心。

```
# emerge --sync                       # 同步更新本地 protage 树
# emerge -u packagename               # 安装或更新一个软件包
# emerge -C packagename               # 删除软件包
# revdep-rebuild                      # 修复依赖关系的缺失
```

Solaris

<cdrom> 路径通常为 /cdrom/cdrom0.

```
# pkgadd -d <cdrom>/Solaris_9/Product SUNWgtar
# pkgadd -d SUNWgtar                  # 添加下载的软件包(先要 bunzip2)
# pkgrm SUNWgtar                      # 删除软件包
```

FreeBSD

```
# pkg_add -r rsync                    # 获取并安装 rsync
# pkg_delete /var/db/pkg/rsync-xx     # 删除 rsync 软件包
```

可使用 PACKAGESITE 环境变量来设置哪里可以获取软件包。举个例子：

```
# export PACKAGESITE=ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages/Latest/
# or ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-6-stable/Latest/
```

FreeBSD ports

Port 树 /usr/ports/ 是一个准备编译和安装的软件集。可用 portsnap 工具来跟新 port。

```
# portsnap fetch extract              # 当第一次运行这个命令，会创建 port 树
# portsnap fetch update               # 跟新 port 树
# cd /usr/ports/net/rsync/            # 选择软件安装目录
# make install distclean              # 安装并清理(也可看 man ports)
# make package                        # Make 一个二进制软件包
```

15.3 库路径

由于复杂的依赖关系和运行时链接，程序难于分发或拷贝到其他系统。不过对于较少依赖关系的小程序，缺失的库可被拷贝过去。运行时库(即缺失的库)可用 ldd 和 ldconfig 来检查和管理。

```
# ldd /usr/bin/rsync                  # 列出所有所需的运行时库
# ldconfig -n /path/to/libs/          # 添加一个路径到共享库目录Add a path to the shared libraries directories
# ldconfig -m /path/to/libs/          # FreeBSD
# LD_LIBRARY_PATH                     # 设置连接库路径的环境变量
```

16 媒体转换

有时候需要转换一个视频、音频文件或者文档成其他格式。

16.1 文本编码

文本编码可以得到完全错误的，特别是当语言需要某些特殊字符像 àäç。命令 iconv 可以从一个编码转换成另一个编码。

```
# iconv -f <from_encoding> -t <to_encoding> <input_file>
# iconv -f ISO8859-1 -t UTF-8 -o file.input > file_utf8
# iconv -l                            # 列显系统所支持的字符编码
```

若文档显示良好，通常都可不使用 -f 选项，iconv 会使用本地字符集(char-set)。

16.2 Unix - DOS 新行

在 Unix Shell 中转换 DOS (CR/LF) 到 Unix (LF) 新行格式。也可使用 dos2unix 和 unix2dos 工具，如果你有它们的话。

```
# sed 's/.$//' dosfile.txt > unixfile.txt
```

在 Windows 环境中转换 Unix 到 Dos 新行格式。需要在 mingw 或 cygwin 中使用 sed。

```
# sed -n p unixfile.txt > dosfile.txt
```

16.3 PDF 转换成 Jpeg 和 连接一串 PDF 文件

用 gs (GhostScript) 工具转换 PDF 文档的每一页成 jpeg (或 png)图像。也可以使用更短的 convert (来自 ImageMagick 或 GraphicsMagick 工具) 命令。

```
# gs -dBATCH -dNOPAUSE -sDEVICE=jpeg -r150 -dTextAlphaBits=4 -dGraphicsAlphaBits=4 \
 -dMaxStripSize=8192 -sOutputFile=unixtoolbox_%d.jpg unixtoolbox.pdf
# convert unixtoolbox.pdf unixtoolbox-%03d.png
# convert *.jpeg images.pdf          # 把所有图片转换成一份简单的 PDF 文档
```

Ghostscript 同样可连接多个 pdf 文件成一份 PDF 文档。这仅可工作于这些 PDF 文件都 "呈现一致(well behaved)" 的情况下。

```
# gs -q -sPAPERSIZE=a4 -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=all.pdf \
file1.pdf file2.pdf ...              # 在 Windows 上使用 '#' 代替 '='
```

16.4 视频转换

使用 mpeg4 编码压缩佳能数码相机视频并修复无用音质。

```
# mencoder -o videoout.avi -oac mp3lame -ovc lavc -srate 11025 \
-channels 1 -af-adv force=1 -lameopts preset=medium -lavcopts \
vcodec=msmpeg4v2:vbitrate=600 -mc 0 vidoein.AVI
```

对于声音的处理可看 sox。

16.5 拷贝音频光盘

程序 cdparanoiahttp://xiph.org/paranoia/ 可以保存音轨(FreeBSD port 在 audio/cdparanoia/)，oggenc 可编码 Ogg Vorbis 格式，lame 可转换成 mp3。

```
# cdparanoia -B                      # 拷贝音轨成 wav 文件到当前目录列表(dir)
# lame -b 256 in.wav out.mp3         # 编码成 256 kb/s 的 mp3
# for i in *.wav; do lame -b 256 $i `basename $i .wav`.mp3; done
# oggenc in.wav -b 256 out.ogg       # 编码成 256 kb/s 的 Ogg Vorbis
```

17 打印

17.1 打印命令 lpr

```
# lpr unixtoolbox.ps                 # 用默认打印机打印
# export PRINTER=hp4600              # 更改默认打印机
# lpr -Php4500 #2 unixtoolbox.ps     # 指定打印机 hp4500 并打印 2 份
# lpr -o Duplex=DuplexNoTumble ...   # 启用双面打印
# lpr -o PageSize=A4,Duplex=DuplexNoTumble ...
# lpq                                # 查看默认打印机的队列
# lpq -l -Php4500                    # 详细显示打印机队列信息
# lprm -                             # 删除所有打印机内的用户打印作业
# lprm -Php4500 3186                 # 删除作业 3186。可使用 lpq 查看作业号
# lpc status                         # 列印所有可用打印机
# lpc status hp4500                  # 如果打印机在线，查看其状态和列队长度
```

当要打印 PDF 文件时，一些打印设备不具备处理 postscript 的能力。可以这样解决译注：此例事实上利用管道(pipe)方式将 PDF 的转换结果利用 -sOutputFile 选项导入给 lpr 打印。：

```
# gs -dSAFER -dNOPAUSE -sDEVICE=deskjet -sOutputFile=\|lpr file.pdf
```

18 数据库

18.1 PostgreSQL

更改 root 用户或其它用户的密码

```
# psql -d template1 -U pgsql
> alter user pgsql with password 'pgsql_password';  # pgsql 为需要更改密码的用户名
```

创建用户和数据库

命令 createuser, dropuser, createdb 和 dropdb 等同于 SQL 命令译注：其实是一个 Shell 脚本的快捷方式。我们创建一个新用户叫 bob 和一个数据库叫 bobdb；使用数据库的超级用户 pgsql 来创建：

```
# createuser -U pgsql -P bob         # -P 会请求一个秘密
# createdb -U pgsql -O bob bobdb     # 新数据库 bobdn 的所有者是 bob译注：通常，执行这个命令的数据库用户成为新数据库的所有者。不过，如果执行用户拥有合适的权限，那么他可以通
# dropdb bobdb                       # 删除数据库 bobdb
# dropuser bob                       # 删除用户 bob
```

一般数据库认证机制配置在 pg_hba.conf 文件中。

允许远程访问

文件 $PGSQL_DATA_D/postgresql.conf 可指定绑定地址。对于 Postgres 8.x 通常为 listen_addresses = '*'。
文件 $PGSQL_DATA_D/pg_hba.conf 定义了访问控制。举例：

```
# TYPE  DATABASE    USER        IP-ADDRESS        IP-MASK           METHOD
host    bobdb       bob         212.117.81.42     255.255.255.255   password
host    all         all         0.0.0.0/0                           password
```

备份和恢复

使用 pgsql 或 postgres 用户来完成备份与恢复。下面是备份和恢复单个数据库：

```
# pg_dump --clean dbname > dbname_sql.dump
# psql dbname < dbname_sql.dump
```

备份和恢复所有数据库(包括用户)：

```
# pg_dumpall --clean > full.dump
# psql -f full.dump postgres
```

在这个例子中，你可以声明任意现有的数据库进行连接，但是如果你是向一个空的数据库集群装载，那么 postgres 应该是比较好的选择。

18.2 MySQL

更改 mysql root 用户或其它用户的密码

方法 1

```
# /etc/init.d/mysql stop
or
# killall mysqld
# mysqld --skip-grant-tables
# mysqladmin -u root password 'newpasswd'
# /etc/init.d/mysql start
```

方法 2

```
# mysql -u root mysql
mysql> UPDATE USER SET PASSWORD=PASSWORD("newpassword") where user='root';
mysql> FLUSH PRIVILEGES;                          # 使用用户名替代"root"
mysql> quit
```

创建用户和数据库

```
# mysql -u root mysql
mysql> CREATE DATABASE bobdb;
mysql> GRANT ALL ON *.* TO 'bob'@'%' IDENTIFIED BY 'pwd';
                          # 使用 localhost 替代 % 来限制网络访问
mysql> DROP DATABASE bobdb;                       # 删除数据库 bobdb
mysql> DROP USER bob;                             # 删除用户 bob
mysql> DELETE FROM mysql.user WHERE user='bob and host='hostname';
                          # 删除 mysql 数据库 user 表中 user=bob,host=hostname 的记录
mysql> FLUSH PRIVILEGES;
```

允许远程访问

远程访问通常允许单个数据库，而不是所有的数据库。文件 `/etc/my.cnf` 包含约定的 IP 地址。通常为 `bind-address =` 绑定地址。

```
# mysql -u root mysql
mysql> GRANT ALL ON bobdb.* TO bob@'xxx.xxx.xxx.xxx' IDENTIFIED BY 'PASSWORD';
mysql> REVOKE GRANT OPTION ON foo.* FROM bar@'xxx.xxx.xxx.xxx';
mysql> FLUSH PRIVILEGES;                  # 使用 'hostname' 也可为 '%' 来完全访问
```

备份和恢复

备份和恢复单个数据库：

```
# mysqldump -u root -psecret --add-drop-database dbname > dbname_sql.dump
# mysql -u root -psecret -D dbname < dbname_sql.dump
```

备份和恢复所有的数据库：

```
# mysqldump -u root -psecret --add-drop-database --all-databases > full.dump
# mysql -u root -psecret < full.dump
```

这里 mysql root 的密码为 "secret"，-p 选项后面没有空格。当单独使用 -p 选项(不跟密码)，命令行提示符后会要求输入密码。

18.3 SQLite

SQLitehttp://www.sqlite.org 是一个小而强大的、独立的(self-contained)、无服务器的(serverless)、零配置的(zero-configuration) SQL 数据库。

备份和恢复

实用备份和恢复 SQLite 数据库命令。举个例子，你可以编辑备份文件来修改字段的属性和类型，然后再恢复这个数据库。这比使用 SQL 命令来得容易。对于 3.x 数据库可使用 `sqlite3`。

```
# sqlite database.db .dump > dump.sql             # 备份
# sqlite database.db < dump.sql                   # 恢复
```

转换 2.x 到 3.x 数据库

```
sqlite database_v2.db .dump | sqlite3 database_v3.db
```

19 磁盘限额

磁盘限额用来限制磁盘空间大小和/或用户(或用户组)可用的文件数。The quotas are allocated on a per-file system basis and are enforced by the kernel.

19.1 Linux 设置

Quota 工具包通常已安装，其包含一些命令行工具。
在 fstab 中激活用户配额并重新挂载分区。如果分区正在使用，关闭所有使用的文件，或者重启系统。添加 `usrquota` 到 fstab 的挂载类型中，举个例子：

```
/dev/sda2     /home     reiserfs     rw,acl,user_xattr,usrquota 1 1
# mount -o remount /home
# mount                                 # 检查 usrquota 已经激活，否则重启
```

用 `quotacheck` 初始化 quota.user 文件。

```
# quotacheck -vum /home
# chmod 644 /home/aquota.user         # 让用户检查自己的配额
```

用脚本(e.g. SuSE 的 /etc/init.d/quotad)或 `quotaon` 来启用限额：

```
quotaon -vu /home
```

检查配额启用情况：

```
quota -v
```

19.2 FreeBSD 设置

Quota 工具是 FreeBSD 基本系统的一部分，然而内核需要 quota 选项。如果不存在，新增它并重新编译内核。

```
options QUOTA
```

与 Linux 一样，添加限额到 fstab 选项(是 userquota，而不是 usrquota)中：

```
/dev/ad0s1d    /home    ufs      rw,noatime,userquota    2  2
# mount /home                          # 重新挂载分区
```

在 /etc/rc.conf 中启用磁盘限额并开启 quota 服务。

```
# grep quotas /etc/rc.conf
enable_quotas="YES"                 # 在启动时打开限额(或者 "NO")
check_quotas="YES"                  # 在启动时检查限额(或者 "NO")
# /etc/rc.d/quota start
```

19.3 分配限额

磁盘限额默认并不限制(设置为0)。用 `edquota` 来对单用户进行限制。一个 quota 也可给许多用户复用。虽然 quota 实现之间的文件结构不同，但其原理是相同的：限制节点(inodes)数量以及使用者可以取用的磁盘区块数量。Only change the values of soft and hard. 如果未指定，默认区块大小为 1k。使用 `edquota -t` 设置 grace 时间。举个例子：

```
# edquota -u colin
```

Linux

```
Disk quotas for user colin (uid 1007):
  Filesystem         blocks       soft       hard     inodes     soft     hard
  /dev/sda8             108       1000       2000          1        0        0
```

FreeBSD

```
Quotas for user colin:
/home: kbytes in use: 504184, limits (soft = 700000, hard = 800000)
   inodes in use: 1792, limits (soft = 0, hard = 0)
```

给许多用户分配限额

命令 `edquota -p` 用来复用一个 quota 给其他用户。举个例子，复用所指用户的限额给所有用户：

```
# edquota -p refuser `awk -F: $3 > 499 {print $1}' /etc/passwd`
# edquota -p refuser user1 user2     # 复用给 2 个用户
```

检查

用户只需输入 `quota` (文件 quota.user 必须可读) 来可以检查他们的限额。Root 可以查看所有用户的限额。

```
# quota -u colin                    # 查看用户的限额
# repquota /home                    # 所有用户在这个分区上的限额情况
```

20 SHELLS

许多 Linux 发行版使用 BASH Shell，BSD 使用的是 tcsh，Bourne Shell 仅用于脚本。过滤器(Filter)非常有用并且可用于管道(pipe)：

`grep`	模式匹配
`sed`	查找并替换字符串或字符
`cut`	从一个标记开始打印所指定列数据
`sort`	按字母或数字排序
`uniq`	删除一个文件中重复行

举个例子，一次使用所有命令：

```
# ifconfig | sed 's/  / /g' | cut -d" " -f1 | uniq | grep -E "[a-z0-9]+" | sort -r
# ifconfig | sed '/.*inet addr:/!d;s///;s/ .*//'|sort -t. -k1,1n -k2,2n -k3,3n -k4,4n
```

sed 的模式字符串中的第一个字符是一个 tab。要在命令控制台中输入 tab，可以使用 ctrl-v ctrl-tab。

20.1 bash

Bash、sh 的重定向译注：当执行一个程序时，运行该程序的进程打开了3个文件描述符，分别是：0(标准输入)、 1(标准输出)和2(标准错误输出)。重定向输出符号(>)是 1>的简写，它通知 shell 重定向标准输出。类似的，<是 0<的简写，表示重定向标准输入。符号 2>将重定向标准错误输出。和管道：

```
# cmd 1> file                       # 重定向标准输出到 file。
# cmd 2> file                       # 重定向标准错误输出到 file。
# cmd 1>> file                      # 重定向标准输出并追加到 file。
# cmd &> file                       # 重定向标准输出和标准错误输出到 file。
# cmd >file 2>&1                    # 重定向标准错误输出到标准输出然后重定向到 file。
# cmd1 | cmd2                       # cmd1 的输出通过管道连接到 cmd2 的输入
# cmd1 2>&1 | cmd2                  # cmd1 的输出和错误输出通过管道连接到 cmd2 的输入
```

修改你的配置文件 ~/.bashrc (也可以是 ~/.bash_profile)。下列条目非常有用，使用". .bashrc"重新加载。

```
# in .bashrc
bind '"\e[A":history-search-backward # 使用上下键查找
bind '"\e[B":history-search-forward  # 历史命令。无价之宝!
set -o emacs                        # Set emacs mode in bash (看下面)
set bell-style visible              # Do not beep, inverse colors
    # 设置一个漂亮的提示符像 [user@host]/path/todir>
PS1="\[\033[1;30m\][\[\033[1;34m\]\u\[\033[1;30m\]"
PS1="$PS1@\[\033[0;33m\]\h\[\033[1;30m\]]\[\033[0;37m\]"
PS1="$PS1\w\[\033[1;30m\]>\[\033[0m\]"

# 要检查当前可用别名(alias)，只需简单输入命令 alias
alias  ls='ls -aF'                  # 添加指示符(*/=>@| 其中之一)
alias  ll='ls -aFls'                # 清单
alias  la='ls -all'
alias ..='cd ..'
alias ...='cd ../..'
export HISTFILESIZE=5000            # 巨大的历史记录
export CLICOLOR=1                   # 使用颜色(如果可用)
export LSCOLORS=ExGxFxdxCxDxDxBxBxExEx
```

20.2 tcsh

Tcsh、csh 的重定向和管道(> 和 >> 同 sh 中一样)：

```
# cmd >& file                       # 重定向标准输出和标准错误输出到 file。
# cmd >>& file                      # 追加标准输出和标准错误输出到 file。
# cmd1 | cmd2                       # cmd1 的输出通过管道连接到 cmd2 的输入
# cmd1 |& cmd2                      # cmd1 的输出和错误输出通过管道连接到 cmd2 的输入
```

Csh/tcsh 的设置在 `~/.cshrc` 中，使用"source .cshrc"来重新加载。例子：

```
# in .cshrc
alias  ls      'ls -aF'
alias  ll      'ls -aFls'
alias  la      'ls -all'
alias  ..      'cd ..'
alias  ...     'cd ../..'
set   prompt    = "%B%n%b@%B%m%b%/> " # 像 user@host/path/todir>
set   history   =  5000
set   savehist  = ( 6000 merge )
set   autolist                        # 控制命令补全和变量补全
set   visiblebell                     # 使用闪动屏幕的方式来取代蜂鸣器鸣叫

# Bindkey 和颜色
bindkey -e     Select Emacs bindings  # 将命令行编辑器切换到emacs模式
bindkey -k up history-search-backward # 使用上下键来搜索
bindkey -k down history-search-forward
setenv CLICOLOR 1                     # 使用颜色(可能的话)
setenv LSCOLORS ExGxFxdxCxDxDxBxBxExEx
```

该 emacs 模式将使用 emacs 快捷键来修改命令提示行。这是非常有用的(不单为 Emacs 用户)。最常用的命令如下：

C-a	移动光标到行头
C-e	移动光标到行尾
M-b	移动光标到前一个单词
M-f	移动光标到后一个单词
M-d	剪切下一个单词
C-w	剪切最后一个单词
C-u	剪切光标前所有字符
C-k	剪切光标后所有字符
C-y	粘帖最后剪切的字符(简易的粘帖)
C-_	重做

注意: C- = 按住 control 键，M- = 按住 meta (它通常为 alt 或者 escape)键。

21 脚本

基础 | 脚本实例 | sed/实用命令

Bourne shell译注：Shell 存在很多种，如 bash(Bourne Again Shell),csh(C Shell),tcsh(TC Shell),zsh(Z Shell) 等。通过 ps 命令可识别出正在运行的是哪种 Shell。 (/bin/sh) 存在于所有的 Unix 系统上，并且用她写的脚本是(完全)可移植的； `man 1 sh` 是一个好的参考。

21.1 基础

变量和参数

使用 `variable=value` 的命令格式设置变量，其中 variable 是变量名称，value是打算赋给该变量的值。使用 $variable 获取变量值。

```
MESSAGE="Hello World"                 # 赋予一个字符串
PI=3.1415                             # 赋予一个十进制小数
N=8
TWON=`expr $N * 2`                    # 算术表达式(只限整数)
TWON=$(($N * 2))                      # 另一种语法
TWOPI=`echo "$PI * 2" | bc -l`        # 使用 bc 进行浮点运算
ZERO=`echo "c($PI/4)-sqrt(2)/2" | bc -l`
```

命令行参数：

```
$0, $1, $2, ...                       # $0 命令本身
$#                                    # 命令参数个数
$*                                    # 所有参数(也可以是 $@)
```

一些特殊的变量

```
$$                                    # 当前进程 ID
$?                                    # 最后命令退出状态码
  command
  if [ $? != 0 ]; then
    echo "command failed"
  fi
mypath=`pwd`
mypath=${mypath}/file.txt
echo ${mypath##*/}                    # 只显示文件名
echo ${mypath%%.*}                    # 除了扩展名的全路径
var2=${var:=string}                   # 如果var没有被赋值，则string值先赋值给var，
                                      # 然后再赋值给var2
```

结构控制

```
for file in `ls`
do
    echo $file
done

count=0
while [ $count -lt 5 ]; do
    echo $count
    sleep 1
    count=$(($count + 1))
done

myfunction() {
    find . -type f -name "*.$1" -print       # $1 为方法的第一个参数
}
myfunction "txt"
```

产生一个文件

```
MYHOME=/home/colin
cat > testhome.sh << _EOF
# 所有_EOF前的代码都会进入到 testhome.sh 文件中去
if [ -d "$MYHOME" ] ; then
    echo $MYHOME exists
else
```

```
    echo $MYHOME does not exist
fi
_EOF
sh testhome.sh
```

21.2 Bourne 脚本实例

来一个小实例，此脚本从本 xhtml 文档创建一个 PDF 小册子：

```
#!/bin/sh
# 此脚本可以创建一份供双面打印机打印的 PDF 格式的书
if [ $# -ne 1 ]; then                          # 检查参数是否等于 1
  echo 1>&2 "Usage: $0 HtmlFile"
  exit 1                                       # 如果不等于1，非0退出
fi

file=$1                                        # 文件变量
fname=${file%.*}                               # 文件名变量
fext=${file#*.}                                # 文件扩展名变量

prince $file -o $fname.pdf                     # www.princexml.com
pdftops -paper A4 -noshrink $fname.pdf $fname.ps # 创建 postscript 小册子
cat $fname.ps |psbook|psnup -Pa4 -2 |pstops -b "2:0,1U(21cm,29.7cm)" > $fname.book.ps

ps2pdf13 -sPAPERSIZE=a4 -sAutoRotatePages=None $fname.book.ps $fname.book.pdf
                                               # 在 Windows 上使用 #a4 和 #None!
exit 0                                         # exit 0 意为成功
```

21.3 一些 sed 命令

这里是单行 sed 命令的金矿http://student.northpark.edu/pemente/sed/sed1line.txt。还有一个很好的 sed 介绍及教程http://www.grymoire.com/Unix/Sed.html。

```
sed 's/string1/string2/g'                      # 替换 string1 为 string2
sed -i 's/wroong/wrong/g' *.txt                # 用 g 替换所有返回的单词
sed 's/\(.*\)1/\12/g'                          # 修改 anystring1 为 anystring2
sed '/<p>/,/<\/p>/d' t.xhtml                   # 删除以 <p> 开始，以 </p> 结尾的行
sed '/ *#/d; /^ *$/d'                          # 删除注释和空行
sed 's/[ \t]*$//'                              # 删除行尾空格 (使用 tab 代替 \t)
sed 's/^[ \t]*//;s/[ \t]*$//'                  # 删除行头尾空格
sed 's/[^*]/[&]/'                              # 括住首字符 [] top -> [t]op
sed = file | sed 'N;s/\n/\t/' > file.num       # 为文件添加行号
```

21.4 正则表达式

一些基本的正则表达式同样可用于 sed。作为一个良好的启蒙，可看 基本正则语法http://www.regular-expressions.info/reference.html。

```
[\^$.|?*+()                                    # 特殊字符，其他字符将匹配自身
\                                              # 转义特殊字符，当成普通字符对待
*                                              # 重复前项 0 次或多次
.                                              # 单个字符除换行符
.*                                             # 匹配 0 个或多个字符
^                                              # 匹配字符串行开始处
$                                              # 匹配字符串行结尾处
.$                                             # 匹配字符串行最后一个字符
^ $                                            # 匹配单个空格的行
[^A-Z]                                         # 匹配任何以 A-Z 字符开始的行
```

21.5 一些实用命令

下列命令对于包含于一个脚本或者单行命令来说很有用。

```
sort -t. -k1,1n -k2,2n -k3,3n -k4,4n           # 排序 IPv4 格式的 IP 地址
echo 'Test' | tr '[:lower:]' '[:upper:]'       # 转换成大写
echo foo.bar | cut -d . -f 1                   # 返回 foo
PID=$(ps | grep script.sh | grep bin | awk '{print $1}')          # 正在运行名为 script 脚本的 PID
PID=$(ps axww | grep [p]ing | awk '{print $1}')                   # ping 的 PID (w/o grep pid)
IP=$(ifconfig $INTERFACE | sed '/.*inet addr:/!d;s///;s/ .*//')   # Linux
IP=$(ifconfig $INTERFACE | sed '/.*inet /!d;s///;s/ .*//')        # FreeBSD
if [ `diff file1 file2 | wc -l` != 0 ]; then [...] fi            # 文件改变了?
cat /etc/master.passwd | grep -v root | grep -v \*: | awk -F":" \ # 创建 http passwd
'{ printf("%s:%s\n", $1, $2) }' > /usr/local/etc/apache2/passwd

testuser=$(cat /usr/local/etc/apache2/passwd | grep -v \    # 查看 passwd 中的用户
root | grep -v \*: | awk -F":" '{ printf("%s\n", $1) }' | grep ^user$)
:(){ :|:& };:                                  # bash fork 炸弹。会干掉你的机器译注：http://forum.ubuntu.org.cn/viewtopic.php?t=92074
tail +2 file > file2                           # 删除文件的第一行
```

我使用一种小伎俩来一次更改许多文件的扩展名。举个例子，从 .cxx 到 .cpp。排除最后的 | sh 先测试一下。你同样可以使用命令 rename 来做这些，如果安装了的话。或者使用 bash 内建命令。

```
# ls *.cxx | awk -F. '{print "mv "$0" "$1".cpp"}' | sh
# ls *.c | sed "s/.*/cp & &.$(date "+%Y%m%d")/" | sh # 如 拷贝 *.c 成 *.c.20080401
# rename .cxx .cpp *.cxx                       # 重命名所有 .cxx 成 .cpp
# for i in *.cxx; do mv $i ${i%%.cxx}.cpp; done      # bash 内建的
```

22 编程

22.1 C 基础

```
strcpy(newstr,str)                             /* 拷贝 str 到 newstr */
expr1 ? expr2 : expr3                          /* if (expr1) expr2 else expr3 */
x = (y > z) ? y : z;                           /* if (y > z) x = y; else x = z; */
int a[]={0,1,2};                               /* 初始化数组 (或者 a[3]={0,1,2}; */
int a[2][3]={{1,2,3},{4,5,6}};                 /* 初始化二维数组 */
int i = 12345;                                 /* 从 i 转换成 char str */
char str[10];
sprintf(str, "%d", i);
```

22.2 C 实例

一个最小化的 C 程式 simple.c：

```
#include <stdio.h>
main() {
    int number=42;
    printf("The answer is %i\n", number);
}
```

编译：

```
# gcc simple.c -o simple
# ./simple
The answer is 42
```

22.3 C++ 基础

```
*pointer                                // 指向对象的指针
&obj                                    // 对象 obj 的地址
obj.x                                   // 类(对象) obj 成员 x
pobj->x                                 // 指针 pobj 指向类(对象)成员 x
                                        // (*pobj).x 同 pobj->x
```

22.4 C++ 实例

来一个稍微现实一点的 C++ 程序，我们在一个头文件(IPv4.h)中创建一个类并且实现它(IPv4.cpp)，然后创建一个程式来使用其功能。这个类的成员方法实现了 IP 地址从一串整数转换成我们熟知的点分格式。这是一个最小化的 C++ 程式和多源文件(multi-source)的编译。

IPv4 class

IPv4.h:

```
#ifndef IPV4_H
#define IPV4_H
#include <string>

namespace GenericUtils {                        // 创建 namespace
class IPv4 {                                    // 类定义
public:
    IPv4();
    ~IPv4();
    std::string IPint_to_IPquad(unsigned long ip);// 成员方法接口
};
} //namespace GenericUtils
#endif // IPV4_H
```

IPv4.cpp:

```
#include "IPv4.h"
#include <string>
#include <sstream>
using namespace std;                            // 使用 namespace
using namespace GenericUtils;

IPv4::IPv4() {}                                 // 默认构造/析构函数
IPv4::~IPv4() {}
string IPv4::IPint_to_IPquad(unsigned long ip) {  // 成员方法实现
    ostringstream ipstr;                        // 使用字符串流
    ipstr << ((ip &0xff000000) >> 24)           // 位右移
          << "." << ((ip &0x00ff0000) >> 16)
          << "." << ((ip &0x0000ff00) >> 8)
          << "." << ((ip &0x000000ff));
    return ipstr.str();
}
```

程序 simplecpp.cpp

```
#include "IPv4.h"
#include <iostream>
#include <string>
using namespace std;

int main (int argc, char* argv[]) {
    string ipstr;                               // 定义变量
    unsigned long ipint = 1347861486;           // 数字形式的 IP
    GenericUtils::IPv4 iputils;                 // 创建一个类的对象
    ipstr = iputils.IPint_to_IPquad(ipint);     // 调研类的成员方法
    cout << ipint << " = " << ipstr << endl;    // 输出结果

    return 0;
}
```

编译和执行：

```
# g++ -c IPv4.cpp simplecpp.cpp               # 编译成目标文件
# g++ IPv4.o simplecpp.o -o simplecpp.exe     # 连接目标代码，生成可执行文件
# ./simplecpp.exe
1347861486 = 80.86.187.238
```

使用 ldd 脚本检查并列出可执行程序所依赖的共享库文件。这个命令同样可以用来检查共享库的丢失。

```
# ldd /sbin/ifconfig
```

22.5 简单的 Makefile

相应的最小化多源文件(multi-source)编译 Makefile 显示如下。每一个命令行必须以 tab 开始！可以将一个较长行使用反斜线"\"来分解为多行。

```
CC = g++
CFLAGS = -O
OBJS = IPv4.o simplecpp.o

simplecpp: ${OBJS}
        ${CC} -o simplecpp ${CFLAGS} ${OBJS}
clean:
        rm -f ${TARGET} ${OBJS}
```

23 在线帮助

23.1 文档

Linux 文档	en.tldp.org
Linux Man Pages	www.linuxmanpages.com
Linux 命令目录	www.oreillynet.com/linux/cmd
Linux doc man howtos	linux.die.net
FreeBSD 手册	www.freebsd.org/handbook
FreeBSD Man Pages	www.freebsd.org/cgi/man.cgi
FreeBSD 用户 wiki	www.freebsdwiki.net
Solaris Man Pages	docs.sun.com/app/docs/coll/40.10

23.2 其他 Unix/Linux 参考

Rosetta Stone for Unix	bhami.com/rosetta.html (a Unix command translator)
Unix guide cross reference	unixguide.net/unixguide.shtml
Linux 命令行列表	www.linuxguide.it/commands_list.php
Short Linux reference	www.pixelbeat.org/cmdline.html

That's all folks!