chmod 4755，这个4表示其他用户执行文件时，具有与所有者相当的权限。

如/user/bin/passwd与/etc/passwd的文件权限。/etc/passwd文件只有root用户拥有写的权限，但普通用户却可以直接调用/user/bin/passwd命令来修改自己的密码，即更改/etc/passwd文件，这是因为/usr/bin/passwd文件权限为rwsr-xr-x，这里的s使得其他用户执行/usr/bin/passwd命令时具有与所有者root相当的权限。

有t特权的目录下的文件或目录，只有拥有者和root才能删除和修改。其他用户即使有写的权限也不能删除。

在看access系统调用时，real UID、GID和effective IDs：

real UID是标记谁调用了该可执行文件，effective UID表示该可执行程序所具有的权限的用户。

默认情况下real UID和effective UID是一样的，但是当使用了setUID之后，两个可能不同。

例如passwd程序，当你执行这个程序的时候，它的real UID就是调用passwd的用户，这个用户可能是root，也可能是任何普通用户，但是它的effective UID是root，只有个这样，passwd程序才能修改/etc/passwd文件。

FTP服务器的搭建：

安装ftp客户端和vsftp服务器端。

配置文件在/etc/vsftpd/

ftpusers中配置的用户是不允许登录ftp服务器的。

rpm命令相关：

rpm –ql ftp 查看ftp软件包安装了什么文件。

rpm -qf 文件路径/文件 查看文件由什么软件包安装。

chattr命令和lsattr命令：

这两个命令用来查看和改变文件目录属性的。

chattr命令的用法：chattr [ -RVf ] [ -v version ] [ mode ] files…

最关键的是在[mode]部分，[mode]部分是由+-=和[ASacDdIijsTtu]这些字符组合的，这部分是用来控制文件的属性。

+ ：在原有参数设定基础上，追加参数。

- ：在原有参数设定基础上，移除参数。

= ：更新为指定参数设定。

A：文件或目录的 atime (access time)不可被修改(modified), 可以有效预防例如手提电脑磁盘I/O错误的发生。

S：硬盘I/O同步选项，功能类似sync。

a：即append，设定该参数后，只能向文件中添加数据，而不能删除，多用于服务器日志文件安全，只有root才能设定这个属性。

c：即compresse，设定文件是否经压缩后再存储。读取时需要经过自动解压操作。

d：即no dump，设定文件不能成为dump程序的备份目标。

i：设定文件不能被删除、改名、设定链接关系，同时不能写入或新增内容。i参数对于文件 系统的安全设置有很大帮助。

j：即journal，设定此参数使得当通过mount参数：data=ordered 或者 data=writeback 挂 载的文件系统，文件在写入时会先被记录(在journal中)。如果filesystem被设定参数为 data=journal，则该参数自动失效。

s：保密性地删除文件或目录，即硬盘空间被全部收回。

u：与s相反，当设定为u时，数据内容其实还存在磁盘中，可以用于undeletion。

各参数选项中常用到的是a和i。a选项强制只可添加不可删除，多用于日志系统的安全设定。而i是更为严格的安全设定，只有superuser (root) 或具有CAP\_LINUX\_IMMUTABLE处理能力（标识）的进程能够施加该选项。

sed使用：

s命令：替换字符串：

例：OSNAME=NFS Desktop

echo $OSNAME | sed ‘s/ //’

结果为NFSDesktop。

-i参数：直接在原文件上进行操作。

例：sed -i “s/printa/printb” b.c //将b.c文件例第一个匹配的printa替换为printb。

kdump：

kdump是在系统崩溃、死锁或者死机的时候用来转储内存运行参数的一个工具和服务，打个比方，如果系统一旦崩溃那么正常的内核就没有办法工作了，在这个时候将由kdump产生一个用于capture当前运行信息的内核，该内核会将此时的内存中的所有运行状态和数据信息收集到一个dump core文件中以便于Red Hat工程师分析崩溃原因，一旦内存信息收集完成，系统将自动重启。这和以前的diskdump，netdump是同样道理。只不过kdump是RHEL6特有的。

shell脚本中的local：

Shell脚本中定义的变量是global的，其作用域从被定义的地方开始，到shell结束或被显式删除的地方为止。

Shell函数定义的变量默认是global的，其作用域从“函数被调用时执行变量定义的地方”开始，到shell结束或被显示删除处为止。函数定义的变量可以被显示定义成local的，其作用域局限于函数内。但请注意，函数的参数是local的。

如果同名，Shell函数定义的local变量会屏蔽脚本定义的global变量。

ssh免密码登陆：

在客户机上执行ssh-keygen -t rsa -P ‘ ’

将在当前用户家目录～/.ssh/id\_rsa.pub复制到需要远程登陆的用户的家目录中，并将文件内容cat id\_rsa.pub > ~/.ssh/authorized\_keys。可能会出现Agent admitted failure to sign using the key错误。在客户机上，当前用户执行ssh-add即可。

计算整个目录下所有文件的MD5值：

find . -print | xargs md5sum > file

shell，在一个脚本中引入另外一个脚本：

实例：a.sh:

#!/bin/bash

echo “a.sh”

. b.sh

b.sh:

#!/bin/bash

echo “b.sh”

执行sh a.sh结果为：

a.sh

b.sh

Debian系Linux更改密码：

在grub引导时，选择需要进入的条目，键入e来进入编辑模式。

在linux那一行后添加 “init=/bin/bash”

ctrx+x 启动系统，进入单用户模式。

挂在系统为读写模式：

mount -no remount,rw /

passwd 进行密码更改。

sync

mount -no remount,ro /

reboot

debootstrap命令：

debootstrap命令是debian/ubuntu下的一个工具，用来构建一套基本的系统（跟文件系统）。生成的目录符合Linux文件系统标准（FHS），即包含到了/boot、/etc、/bin、/usr等目录，但它比发行版本的Linux体积小很多，当然功能也没有那么强大，因此，只能说是“基本的系统”。

awk基本用法：

awk ‘{print $n}’ file //n表示要取的某一行的第n个元素。元素之间使用空格分开。file表示要处理的文件。

# /etc/fstab文件详解：

 /etc/fstab是用来存放文件系统的静态信息的文件。当系统启动的时候，系统会自动地从这个文件读取信息，并且会自动将此文件中指定的文件系统挂载到指定的目录。

## 使用内核名称标识磁盘：

# <file system> <dir> <type> <options> <dump> <pass>

tmpfs /tmp tmpfs nodev,nosuid 0 0

/dev/sda1 / ext4 defaults,noatime 0 1

/dev/sda2 none swap defaults 0 0

/dev/sda3 /home ext4 defaults,noatime 0 2

## 字段定义

/etc/fstab 文件包含了如下字段，通过空格或 Tab 分隔：

<file system> <dir> <type> <options> <dump> <pass>

* **<file systems>** - 要挂载的分区或存储设备.
* **<dir>** - <file systems>的挂载位置。
* **<type>** - 要挂载设备或是分区的文件系统类型，支持许多种不同的文件系统：ext2, ext3, ext4, reiserfs, xfs, jfs, smbfs, iso9660, vfat, ntfs, swap 及 auto。 设置成auto类型，mount 命令会猜测使用的文件系统类型，对 CDROM 和 DVD 等移动设备是非常有用的。
* **<options>** - 挂载时使用的参数，注意有些mount 参数是特定文件系统才有的。一些比较常用的参数有：
* auto - 在启动时或键入了 mount -a 命令时自动挂载。
* noauto - 只在你的命令下被挂载。
* exec - 允许执行此分区的二进制文件。
* noexec - 不允许执行此文件系统上的二进制文件。
* ro - 以只读模式挂载文件系统。
* rw - 以读写模式挂载文件系统。
* user - 允许任意用户挂载此文件系统，若无显示定义，隐含启用 noexec, nosuid, nodev 参数。
* users - 允许所有 users 组中的用户挂载文件系统.
* nouser - 只能被 root 挂载。
* owner - 允许设备所有者挂载.
* sync - I/O 同步进行。
* async - I/O 异步进行。
* dev - 解析文件系统上的块特殊设备。
* nodev - 不解析文件系统上的块特殊设备。
* suid - 允许 suid 操作和设定 sgid 位。这一参数通常用于一些特殊任务，使一般用户运行程序时临时提升权限。
* nosuid - 禁止 suid 操作和设定 sgid 位。
* noatime - 不更新文件系统上 inode 访问记录，可以提升性能(参见 atime 参数)。
* nodiratime - 不更新文件系统上的目录 inode 访问记录，可以提升性能(参见 atime 参数)。
* relatime - 实时更新 inode access 记录。只有在记录中的访问时间早于当前访问才会被更新。（与 noatime 相似，但不会打断如 mutt 或其它程序探测文件在上次访问后是否被修改的进程。），可以提升性能(参见 atime 参数)。
* flush - vfat 的选项，更频繁的刷新数据，复制对话框或进度条在全部数据都写入后才消失。
* defaults - 使用文件系统的默认挂载参数，例如 ext4 的默认参数为:rw, suid, dev, exec, auto, nouser, async.
* **<dump>** dump 工具通过它决定何时作备份. dump 会检查其内容，并用数字来决定是否对这个文件系统进行备份。 允许的数字是 0 和 1 。0 表示忽略， 1 则进行备份。大部分的用户是没有安装 dump 的 ，对他们而言 <dump> 应设为 0。
* **<pass>** fsck 读取 <pass> 的数值来决定需要检查的文件系统的检查顺序。允许的数字是0, 1, 和2。 根目录应当获得最高的优先权 1, 其它所有需要被检查的设备设置为 2. 0 表示设备不会被 fsck 所检查。

## 文件系统标识

在 /etc/fstab配置文件中你可以以三种不同的方法表示文件系统：内核名称、UUID 或者 label。使用 UUID 或是 label 的好处在于它们与磁盘顺序无关。如果你在 BIOS 中改变了你的存储设备顺序，或是重新拔插了存储设备，或是因为一些 BIOS 可能会随机地改变存储设备的顺序，那么用 UUID 或是 label 来表示将更有效。

要显示分区的基本信息请运行：

$ lsblk -f

NAME FSTYPE LABEL UUID MOUNTPOINT

sda

├─sda1 ext4 Arch\_Linux 978e3e81-8048-4ae1-8a06-aa727458e8ff /

├─sda2 ntfs Windows 6C1093E61093B594

└─sda3 ext4 Storage f838b24e-3a66-4d02-86f4-a2e73e454336 /media/Storage

sdb

├─sdb1 ntfs Games 9E68F00568EFD9D3

└─sdb2 ext4 Backup 14d50a6c-e083-42f2-b9c4-bc8bae38d274 /media/Backup

sdc

└─sdc1 vfat Camera 47FA-4071 /media/Camera

## 内核名称

你可以使用 fdisk -l 来获得内核名称，前缀是 dev.

设备 启动 Start 末尾 扇区 Size Id 类型

/dev/sda1 \* 2048 7999487 7997440 3.8G 82 Linux 交换 / Solaris

/dev/sda2 8001534 209713151 201711618 96.2G 5 扩展

/dev/sda5 8001536 209713151 201711616 96.2G 83 Linux

## 标签:

**注意:**使用这一方法，每一个标签必须是唯一的.

要显示所有设备的标签，可以使用 lsblk -f 命令。在 /etc/fstab 中使用 LABEL= 作为设备名的开头 :

# <file system> <dir> <type> <options> <dump> <pass>

tmpfs /tmp tmpfs nodev,nosuid 0 0

LABEL=Arch\_Linux / ext4 defaults,noatime 0 1

LABEL=Arch\_Swap none swap defaults 0 0

## UUID:

所有分区和设备都有唯一的 UUID。它们由文件系统生成工具 (mkfs.\*) 在创建文件系统时生成。

lsblk -f 命令将显示所有设备的 UUID 值。/etc/fstab 中使用 UUID= 前缀:

# <file system> <dir> <type> <options> <dump> <pass>

tmpfs /tmp tmpfs nodev,nosuid 0 0

UUID=24f28fc6-717e-4bcd-a5f7-32b959024e26 / ext4 defaults,noatime 0 1

UUID=03ec5dd3-45c0-4f95-a363-61ff321a09ff /home ext4 defaults,noatime 0 2

UUID=4209c845-f495-4c43-8a03-5363dd433153 none swap defaults 0 0

## 提示和技巧

### 自动挂载

* 如果 /home 分区较大，可以让不依赖 /home 分区的服务先启动。把下面的参数添加到 /etc/fstab 文件中/home 项目的参数部分即可：

noauto,x-systemd.automount

这样 /home 分区只有需要访问时才会被挂载。内核会缓存所有的文件操作，直到 /home 分区准备完成。

**注意:**这样做会使 /home 的文件系统类型被识别为 autofs，造成 mlocate 查询时忽略该目录。实际加速效果因配置而异，所以请自己权衡是否需要。

* 挂载远程文件系统也是同理。如果你仅想在需要的时候才挂载，也可以添加 noauto,x-systemd.automount参数。另外，可以设置 x-systemd.device-timeout=# 参数，设置超时时间，以防止网络资源不能访问的时候浪费时间。
* 如果你的加密文件系统需要密钥，则需要添加 noauto 参数到 /etc/crypttab 文件中的对应位置。systemd 开机的时候就不会打开这个加密设备，会一直等待到设备被访问时再使用密钥文件挂载。比如在使用加密RAID设备的时候可以节省一定的时间，因为 systemd 不必等到设备可用后才能访问。例如：

/etc/crypttab

data /dev/md0 /root/key noauto

**交换分区 UUID**

如果交换分区没有 UUID，可以手动加入。如果使用 lsblk -f 命令没有列出交换分区的 UUID 就说明发生了这种情况。下面是为交换分区指定 UUID 的步骤：

确定交换分区：

# swapon -s

禁用交换分区：

# swapoff /dev/sda7

用新 UUID 重新创建交换分区：

# mkswap -U random /dev/sda7

激活交换分区:

# swapon /dev/sda7

**路径名有空格**

如果挂载的路径中有空格，可以使用 "\040" 转义字符来表示空格（以三位八进制数来进行表示）

/etc/fstab

UUID=47FA-4071 /home/username/Camera\040Pictures vfat defaults,noatime 0 2

/dev/sda7 /media/100\040GB\040(Storage) ext4 defaults,noatime,user 0 0

.....</nowiki>}}

**外部设备**

外部设备在插入时挂载，在未插入时忽略。这需要 nofail 选项，可以在启动时若设备不存在直接忽略它而不报错.

/dev/sdg1 /media/backup jfs defaults,nofail 0 2

**atime 参数**

使用 noatime, nodiratime 或 relatime 可以提升 ext2， ext3 及 ext4 格式磁盘的性能。 Linux 在默认情况下使用atime选项，每次在磁盘上读取（或写入）数据时都会产生一个记录。这是为服务器设计的，在桌面使用中意义不大。默认的 atime 选项最大的问题在于即使从页面缓存读取文件(从内存而不是磁盘读取)，也会产生磁盘写操作！

使用 noatime 选项阻止了读文件时的写操作。大部分应用程序都能很好工作。只有少数程序如 Mutt 需要这些信息。Mutt 的用户应该使用 relatime 选项。使用 relatime 选项后，只有文件被修改时才会产生文件访问时间写操作。nodiratime 选项仅对目录禁用了文件访问时间。relatime 是比较好的折衷，Mutt 等程序还能工作，但是仍然能够通过减少访问时间更新提升系统性能。

**注意:**noatime 已经包含了 nodiratime。不需要同时指定。

**tmpfs**

tmpfs 是一个临时文件系统，驻留于你的交换分区或是内存中（取决于你的使用情况）。使用它可以提高文件访问速度，并能保证重启时会自动清除这些文件。

经常使用 tmpfs 的目录有 /tmp, /var/lock and /var/run. 不要将之使用于 /var/tmp, 因为这一目录中的临时文件在重启过程中需要被保留。使用 tmpfs /run 目录，/var/run 和 /var/lock 是为了兼容老版本建立的链接。默认 /etc/fstab中的的/tmp也是 tmpfs.

默认情况下， tmpfs 分区被设置为你总的内存的一半，当然你可以自由设定这一值。注意实际中内存和交换分区的使用情况取决于你的使用情况，而 tmpfs 分区在其真正使用前是不会占用存储空间的。

要将 /tmp 放到 tmpfs，将下行加入 /etc/fstab：

tmpfs /tmp tmpfs nodev,nosuid 0 0

可以指定大小，但不要修改 mode 选项，以保证文件具有正确的访问权限(1777)。在上例中 /tmp 将最多使用一半内存，要指定最大空间，使用 size 挂载选项：

tmpfs /tmp tmpfs nodev,nosuid,size=2G 0 0

这里有一个更高级的例子，展示如何为用户添加 tmpfs 挂载。这对于网站、mysql 临时文件, ~/.vim/, 和其他情况很有用。尝试并获得理想的挂载选项来完成目标是很重要的。目标是尽量采用安全的策略来防止滥用。限制大小，同时指定 uid 和 gid 加上 mode 是非常安全的。更多信息.

tmpfs /www/cache tmpfs rw,size=1G,nr\_inodes=5k,noexec,nodev,nosuid,uid=648,gid=648,mode=1700 0 0

参阅 **mount** 命令 man 手册以获得更多的内容。

重启后方能生效。注意不要直接执行 mount -a 命令，因为可能造成无法访问当前目录中的文件（比如你应该保证 lockfiles 的正常存在）。然而，如果它们都是空的，那么就可以直接执行 mount -a 而不必重启电脑。

应用更改后，可以通过 findmnt 检查是否生效：

$ findmnt --target /tmp

TARGET SOURCE FSTYPE OPTIONS

/tmp tmpfs tmpfs rw,nosuid,nodev,relatime

**使用**

一般需要大量读写操作的程序在使用 tmpfs 时都会提升性能。有些程序把共享内存放到 tmpfs 上时性能会大幅提升，例如将 Firefox Profile 文件夹放到内存后，Firefox 性能大幅提升。

**Note:**tmpfs 目录(/tmp) 挂载时需要去掉 noexec 参数，否则有些编译程序无法执行，此外，tmpfs 的默认大小是内存的一般，可能会产生空间不够的问题。

下面命令可以让makepkg在tmpfs目录进行编辑，也可以在在/etc/makepkg.conf中进行设置:

$ BUILDDIR=/tmp/makepkg makepkg

**普通用户读写 FAT32**

为了取得对 FAT32 分区的写权限，你必须修改/etc/fstab文件。

/etc/fstab

/dev/sdxY /mnt/some\_folder vfat user,rw,umask=000 0 0

“users”标签的意思是任何用户（甚至非 root 用户）都可以挂载或卸载分区 '/dev/sdX'。“rw”标签则分配读写的使用权。但我不知道“umask”标签的意义（umask 是权限掩码命令 umask=000 指任何人没有特权，且权限为777，即所有人都可以读、写、执行）。我曾试图在“man mount”中查询，但是没有什么结果。

比如你的 FAT32 分区在 '/dev/sda9'，你想将其挂载到 '/mnt/fat32'，那么你需要输入并运行

/etc/fstab

/dev/sda9 /mnt/fat32 vfat user,rw,umask=111,dmask=000 0 0

debian系统更改root用户密码：

开机时在内核启动参数后添加init=/bin/bash

进入命令行执行：

mount -no remount,rw /

passwd root

sync

mount -no remount,ro /

reboot

如果系统使用的是systemd，则在内核启动参数添加：

systemd.unit=emergency.target

systemd：

systemd.unit=rescue.target 是一个设置基本系统和救援 shell 的特殊 target *unit* (与运行级1相似)；systemd.unit=emergency.target 与传递保留参数的init=/bin/sh 给系统使系统从该状态启动相似；systemd.unit=multi-user.target 设置一个非图形化的多用户系统； systemd.unit=graphical.target 设置一个图形化的登录界面。

关于特殊的 systemd 启动 *unit* 的详细内容，请参看 systemd.special 手册页。

systemd的的服务管理程序：systemctl

systemctl 是最主要的工具。它融合 service 和 chkconfig 的功能于一体。你可以使用它永久性或只在当前会话中启用/禁用服务。

查看正在运行的服务：

systemctl

运行一个服务：

systemctl start ssh.service

关闭一个服务：

systemctl stop ssh.service

重新启动一个服务：

systemctl restar ssh.service

显示一个服务的运行状态：

systemctl status ssh.service

在开机时启用一个服务：

systemctl enable ssh.service

在关机时禁用一个服务：

systemctl disable ssh.service

检查一个服务是否是开机启动：

systemctl is-eabled ssh.service

dpkg-deb命令：

-c：显示软件包中的文件列表；

-e：将主控信息解压；

-f：把字段内容打印到标准输出；

-x：将软件包中的文件释放到指定目录下；

-X：将软件包中的文件释放到指定目录下，并显示释放文件的详细过程；

-w：显示软件包的信息；

-l：显示软件包的详细信息；

-R：提取控制信息和存档的清单文件；

-b：创建debian软件包。

dpkg命令的使用：  
dpkg -l 查看当前系统中已经安装的软件包的信息  
dpkg -L （软件包名称）查看系统中已经安装的软件文件的详细列表  
dpkg -s 查看已经安装的指定软件包的详细信息  
dpkg -S 查看系统中的某个文件属于那个软件包;  
dpkg -i \*.deb文件的安装  
dpkg -r \*.deb文件的卸载;  
dpkg -P 彻底的卸载 包括软件的配置文件等等  
查看没有安装的deb包命令  
dpkg -c 查询deb包文件中所包含的文件  
dpkg -I 查询deb包的详细信息

kill和kill -9：

kill就是给某个进程id发送了一个信号。默认发送的信号是SIGTERM，让程序自行退出，可以清理资源什么的。而kill -9发送的信号是SIGKILL，即exit，直接杀死进程。exit信号不会被系统阻塞，所以kill -9能顺利杀掉进程。当然你也可以使用kill发送其他信号给进程。

在sbuild执行中，出现错误如下：

There are problems and -y was used without --force-yes

解决方法：

**Create a file in /etc/apt/apt.conf.d**

# cat /etc/apt/apt.conf.d/02allow-unsigned

APT::Get::AllowUnauthenticated 1;

apt-get update时出现公钥找不到时：

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 6AF0E1940624A220 #此处6AF0E1940624A220需要是错误提示的key

在编译比较大的软件时，若是出现collect2: error: ld terminated with signal 9 [Killed]等错误，即某进程被杀死，则可能是因为内存不足，OOM。/var/log/kern.log中可以看到：



**strace命令：**

strace常用来跟踪进程执行时的系统调用和所接收的信号。

参数：

-c 统计每一系统调用的所执行的时间,次数和出错的次数等.

-d 输出strace关于标准错误的调试信息.

-f 跟踪由fork调用所产生的子进程.

-ff 如果提供-o filename,则所有进程的跟踪结果输出到相应的filename.pid中,pid是各进程的进程号.

-F 尝试跟踪vfork调用.在-f时,vfork不被跟踪.

-h 输出简要的帮助信息.

-i 输出系统调用的入口指针.

-q 禁止输出关于脱离的消息.

-r 打印出相对时间关于,,每一个系统调用.

-t 在输出中的每一行前加上时间信息.

-tt 在输出中的每一行前加上时间信息,微秒级.

-ttt 微秒级输出,以秒了表示时间.

-T 显示每一调用所耗的时间.

-v 输出所有的系统调用.一些调用关于环境变量,状态,输入输出等调用由于使用频繁,默认不输出.

-V 输出strace的版本信息.

-x 以十六进制形式输出非标准字符串

-xx 所有字符串以十六进制形式输出.

-a column

设置返回值的输出位置.默认 为40.

-e expr

指定一个表达式,用来控制如何跟踪.格式如下:

[qualifier=][!]value1[,value2]...

qualifier只能是 trace,abbrev,verbose,raw,signal,read,write其中之一.value是用来限定的符号或数字.默认的 qualifier是 trace.感叹号是否定符号.例如:

-eopen等价于 -e trace=open,表示只跟踪open调用.而-etrace!=open表示跟踪除了open以外的其他调用.有两个特殊的符号 all 和 none.

注意有些shell使用!来执行历史记录里的命令,所以要使用\\.

-e trace=set

只跟踪指定的系统 调用.例如:-e trace=open,close,rean,write表示只跟踪这四个系统调用.默认的为set=all.

-e trace=file

只跟踪有关文件操作的系统调用.

-e trace=process

只跟踪有关进程控制的系统调用.

-e trace=network

跟踪与网络有关的所有系统调用.

-e strace=signal

跟踪所有与系统信号有关的 系统调用

-e trace=ipc

跟踪所有与进程通讯有关的系统调用

-e abbrev=set

设定 strace输出的系统调用的结果集.-v 等与 abbrev=none.默认为abbrev=all.

-e raw=set

将指 定的系统调用的参数以十六进制显示.

-e signal=set

指定跟踪的系统信号.默认为all.如 signal=!SIGIO(或者signal=!io),表示不跟踪SIGIO信号.

-e read=set

输出从指定文件中读出 的数据.例如:

-e read=3,5

-e write=set

输出写入到指定文件中的数据.

-o filename

将strace的输出写入文件filename

-p pid

跟踪指定的进程pid.

-s strsize

指定输出的字符串的最大长度.默认为32.文件名一直全部输出.

-u username

以username 的UID和GID执行被跟踪的命令

**meld比较两个文件夹的不同。**

vim cstag提示找到更多而不显示具体的，可以在vim中执行这个命令:tselect

**nm命令：**

参数：

**僵尸进程和孤儿进程及其危害：**

　 孤儿进程：一个父进程退出，而它的一个或多个子进程还在运行，那么那些子进程将成为孤儿进程。孤儿进程将被init进程(进程号为1)所收养，并由init进程对它们完成状态收集工作。

僵尸进程：一个进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait或waitpid获取子进程的状态信息，那么子进程的进程描述符仍然保存在系统中。这种进程称之为僵死进程。

**问题及危害**

unix提供了一种机制可以保证只要父进程想知道子进程结束时的状态信息， 就可以得到。这种机制就是: 在每个进程退出的时候,内核释放该进程所有的资源,包括打开的文件,占用的内存等。但是仍然为其保留一定的信息(包括进程号the process ID,退出状态the termination status of the process,运行时间the amount of CPU time taken by the process等)。直到父进程通过wait / waitpid来取时才释放。 但这样就导致了问题，如果进程不调用wait / waitpid的话，那么保留的那段信息就不会释放，其进程号就会一直被占用，但是系统所能使用的进程号是有限的，如果大量的产生僵死进程，将因为没有可用的进程号而导致系统不能产生新的进程. 此即为僵尸进程的危害，应当避免。

孤儿进程是没有父进程的进程，孤儿进程这个重任就落到了init进程身上，init进程就好像是一个民政局，专门负责处理孤儿进程的善后工作。每当出现一个孤儿进程的时候，内核就把孤儿进程的父进程设置为init，而init进程会循环地wait()它的已经退出的子进程。这样，当一个孤儿进程凄凉地结束了其生命周期的时候，init进程就会代表党和政府出面处理它的一切善后工作。因此孤儿进程并不会有什么危害。

任何一个子进程(init除外)在exit()之后，并非马上就消失掉，而是留下一个称为僵尸进程(Zombie)的数据结构，等待父进程处理。这是每个子进程在结束时都要经过的阶段。如果子进程在exit()之后，父进程没有来得及处理，这时用ps命令就能看到子进程的状态是“Z”。如果父进程能及时处理，可能用ps命令就来不及看到子进程的僵尸状态，但这并不等于子进程不经过僵尸状态。如果父进程在子进程结束之前退出，则子进程将由init接管。init将会以父进程的身份对僵尸状态的子进程进行处理。

僵尸进程危害场景：

例如有个进程，它定期的产生一个子进程，这个子进程需要做的事情很少，做完它该做的事情之后就退出了，因此这个子进程的生命周期很短，但是，父进程只管生成新的子进程，至于子进程退出之后的事情，则一概不闻不问，这样，系统运行上一段时间之后，系统中就会存在很多的僵死进程，倘若用ps命令查看的话，就会看到很多状态为Z的进程。严格地来说，僵死进程并不是问题的根源，罪魁祸首是产生出大量僵死进程的那个父进程。因此，当我们寻求如何消灭系统中大量的僵死进程时，答案就是把产生大量僵死进程的那个元凶枪毙掉（也就是通过kill发送SIGTERM或者SIGKILL信号啦）。枪毙了元凶进程之后，它产生的僵死进程就变成了孤儿进程，这些孤儿进程会被init进程接管，init进程会wait()这些孤儿进程，释放它们占用的系统进程表中的资源，这样，这些已经僵死的孤儿进程就能瞑目而去了。

**进程优先级以及nice值：**

进程cpu资源分配就是指进程的优先权(priority)。优先权高的进程有优先执行的权利。

使用ps -l查看进程：



PRI：代表这个进程可被执行的优先级，其值越小越早被执行。

NI：代表这个进程的nice值。

nice值表示进程可被执行的优先级的修正数，如前面所说，PRI值越小越早被执行。加入nice值后，将会使得PRI变为：PRI(new)=PRI(old)+nice。这样，当nice值为负数时，优先级将提高。

nice值并不是进程的优先级，但是他会影响到进程的优先级变化。

**修改nice值的方法**：nice renice

进程开始时就设定nice值：

如：

nice –n -5 /usr/local/mysql/bin/mysql\_safe &

语　　法：nice [-n <优先等级>][--help][--version][执行指令]

补充说明：nice指令可以改变程序执行的优先权等级。

参　　数：-n<优先等级>或-<优先等级>或-adjustment=<优先等级> 设置欲执行的指令的优先权等级。等级的范围从-20-19，其中-20最高，19最低，只有系统管理者可以设置负数的等级。

调整已经存在进程的nice值：

如：

renice -5 -p 1200

#PID为1200的进程nice值设为-5

linux renice 命令详解

语　　法：renice [优先等级][-g <程序群组名称>...][-p <程序识别码>...][-u <用户名称>...]

补充说明：renice指令可重新调整程序执行的优先权等级。预设是以程序识别码指定程序调整其优先权，您亦可以指定程序群组或用户名称调整优先权等级，并修改所有隶属于该程序群组或用户的程序的优先权。等级范围从-20–19，只有系统管理者可以改变其他用户程序的优先权，也仅有系统管理者可以设置负数等级。

参　　数：

-g <程序群组名称> 　使用程序群组名称，修改所有隶属于该程序群组的程序的优先权。

-p <程序识别码> 　改变该程序的优先权等级，此参数为预设值。

-u <用户名称> 　指定用户名称，修改所有隶属于该用户的程序的优先权

也可以使用top命令更改已经存在进程的nice值：

#进入top后按“r”–>输入进程PID–>输入nice值

**Linux修改可同时运行的程序的最大值**：

临时：更改/proc/sys/kernel/pid\_max的值。

永久：更改/etc/sysctl.conf中更改kernel.pid\_max = 所需要的值。

**iostat和vmstat命令：**

iostat主要用于监控系统设备的IO负载情况，iostat首次运行时显示自系统启动开始的各项统计信息，之后运行iostat将显示自上次运行该命令以后的统计信息，用户可以通过指定统计的次数和时间来获取所需的统计信息。

vmstat命令可以展现给定时间间隔的服务器状态值，包括服务器的cpu使用率、内存使用，虚拟内存交换情况，IO读写情况。

**strace命令：追踪程序的系统调用**

-o file：将内容保存到file文件中。

-e trace=系统调用：只追踪指定系统调用

-c：统计系统调用信息。可以用来查看系统调用的大致情况。

**linux free命令中显示的buff/cache和shared的含义：**

#free -m

total used free shared buff/cache available

Mem: 1989 678 17 105 1293 1155

Swap: 2040 88 1952

shared表示被多个进程共享的内存。

buff：A buffer is something that has yet to be "written" to disk.

cache: A cache is something that has ben "read" from the disk and stored for later use.

available:启动一个新的应用程序不使用swap时可用的内存。

也就是说，buffer是用于存放要输出到disk（块设备）的数据的。而cache是存放从disk读出的数据。这二者都是为了提高IO的性能，并由OS管理。

释放cache：

echo 0 > /proc/sys/vm/drop\_caches表示不释放

echo 1 > /proc/sys/vm/drop\_caches清空最近访问过的文件页面缓存

echo 2 > /proc/sys/vm/drop\_caches清空文件节点缓存和目录项缓存

echo 3 > /proc/sys/vm/drop\_caches清空所有

swap分区：Linux内核为了提高读写效率与速度，会将文件在内存中进行缓存，这部分内存就是Cache Memory(缓存内存)。即使你的程序运行结束后，Cache Memory也不会自动释放。这就会导致你在Linux系统中程序频繁读写文件后，你会发现可用物理内存变少。当系统的物理内存不够用的时候，就需要将物理内存中的一部分空间释放出来，以供当前运行的程序使用。那些被释放的空间可能来自一些很长时间没有什么操作的程序，这些被释放的空间被临时保存到Swap空间中，等到那些程序要运行时，再从Swap分区中恢复保存的数据到内存中。这样，系统总是在物理内存不够时，才进行Swap交换。

**文件系统管理工具：**

通用：mkfs，blkid，fsck

ext系列文件系统专用工具：dumpe2fs、tune2fs、e2label、e2fsck

**内建命令exec：**

环境变量（传给子进程的变量，遗传性是本地变量和环境变量的根本区别）只能单向从父进程传给子进程。不管子进程的环境变量如何变化，都不会影响父进程的环境变量。

有两种方法执行shell脚本，一种是新产生一个shell，然后执行相应的shell脚本；一种是在当前shell下执行，不再启用其他的shell。

新产生一个shell然后再执行的脚本的方法是在脚本文件的开头加入#!/bin/sh，这种方法先启用一个新的sub-shell然后在其下执行命令。

另一种方法是通过source命令或者（.），不再产生新的shell，而是在当前的shell执行一切命令。

exec命令在执行时，会把当前的shell进程关闭，然后换到后面的命令继续执行。比如在当前shell窗口中直接执行exec ls，则当前的shell窗口将会关闭，如果在当前shell窗口中先执行bash（在当前的shell中打开一个子进程），然后再执行exec ls，则当前shell窗口不会关闭，但是当前shell中打开的子进程则会退出。

系统调用exec是以新的进程去代替原来的进程，但是进程的PID保持不变。因此，可以这样认为，exec系统调用并没有创建新的进程，只是替换了原来进程上下文的内容。原进程的代码段、数据段、堆栈段被新的进程所代替。

exec关于文件描述符的操作。

# Objdump:

Objdump命令是查看目标文件或者可执行文件的目标文件的构成的gcc工具。

参数：

--archive-headers | -a：显示档案库的成员信息，类似ls -l，将lib\*.a的信息列出。

实例：objdump -a /usr/bin/zipz

输出：/usr/bin/zip: file format elf64-x86-64 /usr/bin/zip

--target=bfdname|-b bfdname：指定目标代码格式，这不是必须的，objdump可以自动识别许多格式。objdump -i可以列出目标码格式列表。

--demangle|-C：将底层的符号名解码成用户级名字，除了去掉所开头的下划线之外，还使得C++函数名以可理解的方式显示出来。

--debugging|-g：显示调试信息。企图解析保存在文件中的调试信息并以C语言的语法显示出来。仅支持某些类型的调试信息。有些其他格式被readelf -w支持。

--debugging-tags|-e：类似-g选项，但是生成的信息是和ctags工具相兼容的格式。

--disassemble|-d：从objfile中反汇编那些特定指令机器码的section。

--disassemble-all|-D：与-d类似，但反汇编所有section。

--prefix-addresses：反汇编的时候，显示每一行的完整地址。这是一种比较老的反汇编格式。

-EB|-EL|--endian={big|little}：指定目标文件的小端。这个项将影响反汇编出来的指令。在反汇编的文件没描述小端信息的时候用。例如S-records。

-f|--file-headers：显示objfile中每个文件的整体头部摘要信息。

-h|--section-headers|--heders：显示目标文件各个section的头部摘要信息。

-H|--help：简短的帮助信息。

-i|--info：显示对于-b或者-m选项可用的架构和目标格式列表。

-j|--section=name：仅仅显示指定名称为name的section的信息。

-l|--line-numbers：用文件名和行号标注相应的目标代码，仅仅和-d、-D或者-r一起使用。使用-ld和使用-d的区别不是很大，在源码级调试的时候有用，要求编译时使用了-g之类的调试编译选项。

-m machine|--architecture=machine：指定反汇编目标文件时使用的架构，当待反汇编文件本身没描述架构信息的时候（比如S-records），这个选项很有用。可以用-i选项列出这里能够指定的架构。

-r|--reloc：显示文件的重定位入口。如果和-d或者-D一起使用，重定位部分以反汇编后的格式显示出来。

-R|--dynamic-reloc：显示文件的动态重载定位入口，仅仅对与动态目标文件有意义，比如某些共享库。

-s|--full-contents：显示指定section的完整内容。默认所有非空section都会被显示。

-S|--source：尽可能反汇编出源代码尤其是当编译的时候指定了-g这种调试参数时，效果比较明显。隐含了-d参数。

--show-raw-insn：反汇编时，不显示汇编指令的机器码，如不指定--prefix-addresses，这将是缺省选项。

--no-show-raw-insn 反汇编时，不显示汇编指令的机器码，如不指定--prefix-addresses，这将是缺省选项。

--start-address=address 从指定地址开始显示数据，该选项影响-d、-r和-s选项的输出。

--stop-address=address 显示数据直到指定地址为止，该项影响-d、-r和-s选项的输出。

-t|--syms 显示文件的符号表入口。类似于nm -s提供的信息

-T|--dynamic-syms 显示文件的动态符号表入口，仅仅对动态目标文件意义，比如某些共享库。它显示的信息类似于 nm

-D|--dynamic 显示的信息。

-V|--version 版本信息

--all-headers|-x 显示所可用的头信息，包括符号表、重定位入口。-x 等价于-a -f -h -r -t 同时指定。

-z|--disassemble-zeroes 一般反汇编输出将省略大块的零，该选项使得这些零块也被反汇编。

@file 可以将选项集中到一个文件中，然后使用这个@file选项载入。

通过阅读dpkg-shlibdeps源码，其中有介绍：

Output format of objdump -w -T

