

Red Flag Asianux Server 3

系统管理手册

北京中科红旗软件技术有限公司

地址：中国北京海淀区万泉河路 68 号紫金大厦 6 层

Red Flag Software Co., Ltd.

<http://www.redflag-linux.com>

声明:

本软件受相应版权法保护,并在 GNU GPL 约束其使用、拷贝、发布及反编译的授权下发布。在未经红旗软件公司事先书面授权的情况下,文档的任何部分都不得以任何形式和途径进行复制、修改及分发。本手册在编写过程中由于已考虑了各种可能的预防措施,红旗软件公司对可能出现的内容错误及缺失不承担责任。

此出版物仅以其原有的存在形式提供,不含任何种类的明示或默示,包括但不限于那些隐含的用于商业目的的、为某种特定目的而定制的、或无特定目的的担保。此出版物可能会出现技术上的失误或印刷上的错误。其更正将不断添加于此,并合并到此出版物的最新版本中。

红旗软件公司保留在任何时刻对此出版物介绍的产品和/或程序进行添加和/或修改的权利。

本文档的最终解释权归属于红旗软件公司。

©2007, 版权所有: 北京中科红旗软件技术有限公司。

本产品使用了如下字库:

东文字库, 版权所有©长沙东文软件有限公司。

本产品使用了如下输入法:

智能通用输入法平台 - SCIM, 版权所有©苏哲。

目 录

序.....	1
本书的适用对象	1
印刷惯例	1
提示与警告	2
第 1 章 命令行操作	1
1.1 基础知识	1
1.1.1 文件命名	1
1.1.2 路径	1
1.1.3 文件类型	1
1.1.4 目录结构	2
1.1.5 Shell 简介	3
1.1.6 系统帮助	4
1.2 目录操作命令	5
1.2.1 查看目录	5
1.2.2 改变工作目录	5
1.2.3 创建目录	6
1.2.4 删除目录	6
1.2.5 显示当前目录	6
1.3 文件操作命令	7
1.3.1 显示文本文件	7
1.3.2 创建新文件	8
1.3.3 拷贝文件	8
1.3.4 移动和重命名文件	9
1.3.5 删除文件	9
1.3.6 文件链接	9
1.3.7 文件内容比较	10
1.3.8 查找文件	11
1.3.9 在文件中查找正文	12
1.4 文件权限操作	12
1.4.1 改变文件主	13
1.4.2 改变用户组	13

1.4.3	文件权限设置	13
1.4.4	改变文件权限	14
1.4.5	默认权限	16
1.5	定向和管道	16
1.5.1	输入重定向	17
1.5.2	输出重定向	17
1.5.3	管道	19
1.6	进程和作业控制命令	19
1.7	基本网络命令	22
1.7.1	telnet 命令	22
1.7.2	ftp 命令	23
1.7.3	ping 命令	24
1.7.4	finger 命令	24
第 2 章	文件系统管理	25
2.1	文件系统	25
2.1.1	Red Flag Asianux Server 3 支持的文件系统类型	25
2.1.2	文件系统的创建、加载与卸载	25
2.1.3	维护文件系统	28
2.1.4	常用文件系统管理命令	28
2.1.5	使用设备	28
2.2	磁盘分区管理	30
2.2.1	查看分区表	31
2.2.2	创建分区	31
2.2.3	删除分区	32
2.2.4	重新划分分区大小	32
2.3	EXT3 文件系统	33
2.3.1	ext3 的特性	33
2.3.2	创建一个ext3 文件系统	33
2.3.3	转换到ext3 文件系统	34
2.3.4	还原到ext2 文件系统	34
2.4	交换空间	34
2.4.1	使用交换分区	35
2.4.2	使用交换文件	35
2.5	RAID设备	36

2.5.1	RAID的种类.....	36
2.5.2	RAID的级别.....	36
2.5.3	使用RAID.....	37
2.5.4	配置软件RAID.....	37
2.6	LVM逻辑卷管理.....	44
2.6.1	LVM的优点.....	44
2.6.2	LVM相关概念和术语.....	45
2.6.3	LVM结构图和工作原理.....	46
2.6.4	LVM的一般操作.....	47
2.6.5	磁盘分区问题.....	51
2.6.6	建立LVM用例.....	52
2.6.7	使用snapshot做备份.....	54
2.6.8	更换卷组硬盘.....	55
2.6.9	迁移卷组到其它系统.....	56
2.6.10	分割卷组.....	57
2.6.11	转变根文件系统为LVM.....	59
2.6.12	共享LVM卷.....	61
2.7	配置磁盘限额.....	61
2.7.1	配置文件系统的磁盘限额支持.....	61
2.7.2	设置用户限额.....	62
2.7.3	设置用户组限额.....	63
2.7.4	设置软限制过渡期.....	63
2.7.5	管理磁盘配额.....	63
第3章	高级文件系统指南.....	65
3.1	日志系统（JOURNALING）.....	65
3.1.1	元数据（Meta-data）.....	65
3.1.2	fsck.....	65
3.1.3	日志（Journal）.....	65
3.1.4	不同的日志文件系统.....	66
3.2	EXT3.....	66
3.2.1	日志的记录方式.....	66
3.2.2	JBD的日志记录方法.....	67
3.2.3	数据保护.....	67

3.2.4	ext3 工具.....	68
第 4 章	软件包管理	69
4.1	使用RPM命令	69
4.1.1	安装、升级和更新.....	69
4.1.2	删除.....	69
4.1.3	查询.....	70
4.1.4	验证.....	70
4.2	安装TAR格式的软件包.....	70
第 5 章	使用VIM编辑器	72
5.1	VIM的工作模式.....	72
5.1.1	命令模式.....	72
5.1.2	插入模式.....	72
5.1.3	末行模式.....	72
5.2	VIM编辑文件的基本过程	72
5.2.1	光标的移动.....	72
5.2.2	基本编辑指令.....	74
5.2.3	离开.....	76
第 6 章	系统安全	77
6.1	系统安全概要.....	77
6.1.1	安全管理组成.....	77
6.1.2	常见安全问题及对策.....	78
6.2	系统备份	80
6.2.1	备份前的准备.....	80
6.2.2	常用备份命令.....	80
6.3	加密措施	83
6.3.1	SSH.....	83
6.3.2	PGP.....	87
6.3.3	OPENSSL.....	87
第 7 章	网络安全	88
7.1	XINETD	88
7.1.1	简介.....	88
7.1.2	Xinetd的配置.....	88
7.2	可插入验证模块（PAM）	92
7.2.1	PAM 的优越性.....	92

7.2.2	PAM配置文件.....	93
7.2.3	PAM配置文件的格式.....	93
7.2.4	创建 PAM 模块.....	95
7.2.5	采用防火墙的优势.....	95
7.2.6	防火墙安全控制基本原则.....	95
7.2.7	防火墙基本类型.....	95
7.2.8	Iptables.....	96
7.3	SSH协议.....	98
7.3.1	SSH的特性.....	98
7.3.2	为何使用SSH.....	99
7.3.3	OpenSSH配置文件.....	99
7.3.4	更安全的Shell.....	100
7.4	NMAP扫描工具.....	101
7.4.1	Nmap 简介.....	102
7.4.2	Nmap 使用说明.....	102
7.4.3	Nmap 使用示例.....	113
7.5	流量控制.....	114
7.5.1	简介.....	114
7.5.2	配置.....	115
附 录	121
附录A	常见问题.....	121
附录B	LINUX和DOS常用命令对照表.....	123
附录C	文件类型和文件名.....	124
附录D	术语表.....	126

序

欢迎使用 Red Flag Asianux Server 3 操作系统！

《Red Flag Asianux Server 3 系统管理手册》广泛地讨论了进行系统管理所需的基础知识及相关系统管理主题，是帮助您顺利执行系统管理任务并配置和管理一个高效、安全、稳定的服务器系统的理想选择。主要包括：

- 命令行界面操作
- 文件系统管理
- 配置任务计划
- 查看和维护系统日志
- 系统性能监视
- 系统信息查看
- 系统服务管理
- 系统启动服务配置
- 软件包管理
- 用户和用户组管理
- 使用 vi 编辑器

本书的适用对象

本手册适用于已掌握 Red Flag Asianux Server 3 系统的基础知识，并希望进一步学习系统管理技术，掌握操作系统定制的中级使用者。

事实上，Red Flag Asianux Server 3 提供了多个图形化系统管理工具，利用这些工具能够简便、快捷地实现系统管理和配置任务，无论是资深管理员，还是 Linux 新手，使用起来都能得心应手。

印刷惯例

《Red Flag Asianux Server 3 系统管理手册》用不同的字体、大小和风格代表文件名、命令、菜单项和其它特殊元素，具体如下：

格 式	含 义	示 例
command、filename、output message	系统命令、文件名或目录名、计算机的屏幕输出信息。	使用 <code>ls -a</code> 命令来查看当前工作目录中的所有文件； 编辑文件/etc/fstab； [root@localhost /root]#
application	表示一个应用程序或实用工具的名称。	使用 Kedit 编辑文本文件。
<key> <key1+key2>	表示键盘上的按键和组合按键。	使用<Tab>键进行命令补全； 按<Ctrl+Shift>切换输入法类型。

“Menu Item”	界面上引用的文本、按钮和菜单项。	确认后，请单击“下一步”按钮继续。
→	连续菜单选择之间的分隔符。	“新建→用户”表示打开“新建”菜单，选择其中的“用户”子菜单项。
user input	用户在命令行或文本框中输入的内容。	在 boot: 提示下键入 linux expert 命令，进入专家安装方式。

提示与警告

为了强调《Red Flag Asianux Server 3 系统管理手册》中的某些重要的信息，我们使用下面两种方式加以重点说明：



一些有用的额外信息、使用中的提示和帮助用户更加顺利完成工作的小技巧等。



看到这一标记时应特别注意，它表示一些重要的警告和错误提示信息。

第1章 命令行操作

熟悉在命令行界面下工作对使用和管理 Linux 操作系统意义重大，本章将介绍在 Red Flag Asianux Server 3 系统中进行 shell 操作的知识。

1.1 基础知识

以下关于 Linux shell 及文件和目录的知识是学习本章的基础。

1.1.1 文件命名

Linux 下文件名的最大长度可以是 256 个字符，通常由字母、数字、“.”（点号）、“_”（下划线）和“-”（减号）组成。文件名中不能含有“/”符号，因为“/”在 Linux 目录树中表示根目录或路径中的分隔符（如同 DOS 中的“\”）。

Linux 系统支持文件名中的通配符，具体如下：

星号 (*)：匹配零个或多个字符；

问号 (?)：匹配任何一个字符；

[ab1 A-F]：匹配任何一个列举在集合中的字符。本例中，该集合是 a、b、1 或任何一个从 A 到 F 的大写字符；

1.1.2 路径

操作系统查找文件所经过的路径称为路径名。使用当前目录下的文件时可以直接引用文件名；如果要使用其他目录下的文件，就必须指明该文件在哪个目录之中。

按查找文件的起点不同可以分为两种路径：**绝对路径**和**相对路径**。从根目录开始的路径称为绝对路径，从当前所在目录开始的路径称为相对路径，相对路径是随着用户工作目录的变化而改变的。

与 DOS 相同，每个目录下都有代表当前目录的“.”文件和代表当前目录父目录的“..”文件，相对路径名一般就是从“..”开始的。



在 Linux 目录树中，表示根目录或是路径中的分隔符是“/”。

1.1.3 文件类型

Red Flag Asianux Server 3 系统支持以下文件类型：普通文件、目录文件、设备文件以及符号链接文件。

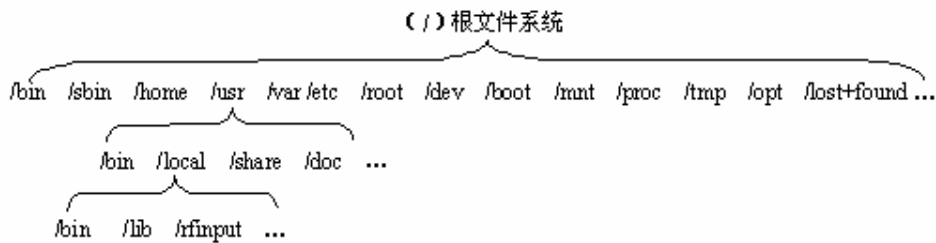
普通文件： 包括文本文件、数据文件、可执行的二进制程序等。

- 目录文件：** 简称目录，Linux 中把目录看作一种特殊文件，利用它构成文件系统的分层树型结构。每个目录文件至少包括两个文件，“..”表示上一级目录，“.”表示该目录本身。
- 设备文件：** 一种特别文件，Linux 系统利用它们来标识各个设备驱动器，核心使用它们与硬件设备通信。有两类特别设备文件：字符设备和块设备。
- 符号链接：** 一种特殊文件，它们存放的数据是文件系统中通向某个文件的路径。当使用符号链接文件时，系统自动地访问所保存的这个路径。

1.1.4 目录结构

通过对系统目录组织结构的了解，可以在进行文件操作和系统管理时方便地知道所要的东西在什么地方。

Red Flag Asianux Server 3 文件系统采用分层的树形目录结构。即在一个根目录（通常用 “/” 表示）中，包含多个下级子目录或文件；子目录中又可含有更下级的子目录或文件信息，这样逐层地延伸下去，构成一棵倒置的树。树中的“根”与“杈”代表的是目录或称为文件夹，而“叶子”则是每个文件，如下图所示。



Linux 树型目录结构

下面列出了主要的系统目录及其简单描述：

- /bin:** 存放普通用户可以使用的命令文件。目录/usr/bin 也可用来贮存用户命令。
- /sbin:** 一般存放非普通用户使用的命令（有时普通用户也可能会用到）。目录/usr/sbin 中也包括了许多系统命令。
- /etc:** 系统的配置文件。
- /root:** 系统管理员（root 或超级用户）的主目录。
- /usr:** 包括与系统用户直接相关的文件和目录，一些主要应用程序也保存在该目录下。
- /home:** 用户主目录的位置，保存了用户文件（用户自己的配置文件、文档、数据等）。
- /dev:** 设备文件。在 Linux 中设备以文件形式表现，从而可以按照操作文件的方式简便地对设备进行操作。

<code>/media:</code>	文件系统挂载点。一般用于安装移动介质、其它文件系统（如 DOS）的分区、网络共享文件系统或任何可安装文件系统。
<code>/lib:</code>	包含许多由/bin 和/sbin 中的程序使用的共享库文件。目录/usr/lib/中含有更多用于用户程序的库文件。
<code>/boot:</code>	包括内核和其它系统启动时使用的文件。
<code>/var:</code>	包含一些经常改变的文件。例如假脱机（spool）目录、文件日志目录、锁文件、临时文件等。
<code>/proc:</code>	操作系统的内存映像文件系统，是一个虚拟的文件系统（没有占用磁盘空间）。当您查看它们时，看到的是内存里的信息，这些文件有助于了解系统内部信息。
<code>/initrd:</code>	在计算机启动时挂载 initrd.img 映像文件的目录以及载入所需设备模块的目录。
<code>/opt:</code>	存放可选择安装的文件和程序。主要由第三方开发者用于安装和卸装他们的软件包。
<code>/tmp:</code>	用户和程序的临时目录，该目录中的文件被系统自动清空。
<code>/lost+found:</code>	在系统修复过程中恢复的文件。

1.1.5 Shell简介

用户在命令行下工作时，不是直接同操作系统内核打交道，而是由命令解释器接受命令，分析后再传给相关的程序。进入 Red Flag Asianux Server 3 环境时系统将自动启动相应的 shell，shell 是一种命令行解释程序，它提供用户与操作系统之间的接口。Red Flag Asianux Server 3 下默认的 shell 是 bash。

bash 命令的基本格式如下：

命令名 **[选项]** **[参数 1]** **[参数 2]...**

其中，方括号括起的部分表明该项对命令而言是可选的。

[选项]: 对命令有特别定义，一般以“-”开始，多个选项可用一个“-”连起来，如 `ls -l -a` 与 `ls -la` 相同。

[参数]: 提供命令运行的信息，或是命令执行过程中所使用的文件名。



输入用户名、口令与文件名、命令名时，一定要区分大小写，因为大小写字母在 Linux 系统中代表不同的含义。



在命令、选项和参数之间要用空格隔开。连续的空格会被 shell 解释为单个空格。

➤ 键入命令

在 shell 提示符下输入相应的命令，然后按回车键确认，shell 将会读取该命令并执行。如果系统找不到输入的命令，会显示：“Command not Found”。这时，需检查键入命令的拼写及大小写是否正确。

使用分号（;）可以将两个命令隔开，这样可以实现在一行中输入多个命令。命令的执行顺序和输入的顺序相同。

➤ 命令补齐

当要输入的命令目录很深或命令中的文件名很长时，只要按一下<TAB>键，系统会在可能的命令或文件名中找到相匹配的项，自动帮您补齐。如果有一个以上的文件符合输入的字符串，不能补齐时，可以按两下<TAB>键，系统将把所有符合的文件名列出来。

➤ 历史记录

shell 会把过去输入过的命令记忆下来，只要按上下方向键，就可以选择以前输入过的命令了。

有了以上基础，可以运行下面列出的几个简单命令来实际使用一下：

clear: 刷新屏幕；

date: 在屏幕上显示日期和时间；

echo: 将命令行中的内容回显到标准输出上。

cal: 显示月份和日历。

1.1.6 系统帮助

Red Flag Asianux Server 3 具有强大的系统和网络功能，数量众多的实用工具软件和大量复杂的操作命令。为了帮助用户随时查询，顺利进行操作，系统提供了多种多样的联机帮助信息。

➤ 联机手册

通过 **man** 命令使用联机用户手册，系统可以显示任何命令的联机帮助信息。它将命令名称作为参数，该命令的语法格式为：

man command

常用的 Linux 系统帮助手册按章节分类，位于 `/usr/man` 目录下。

例如，下面的命令行将显示 **cal** 命令的手册页：

\$ man cal

使用命令“**man man**”会显示出 **man** 命令本身的使用方法。

- 在所查询的命令后加`--help`参数的方式，也可以显示出命令的参考信息。
- 用 `help command` 可列出许多内部命令的帮助。
- `whatis` 命令可查找简要的帮助信息，命令语法为：`whatis keyword`。

1.2 目录操作命令

1.2.1 查看目录

查看目录内容的命令是 `ls`，它默认显示当前目录的内容，可以在命令行参数的位置给出一个或多个目录名，从而可以查看这些目录。命令的语法格式为：

ls [选项]...[文件名]...

`ls` 命令有多个命令行选项，如：

- a: 列出所有文件，包括那些以“.”开头的文件；
- d: 如果后面接的是一个目录，那么使用该参数只输出该目录的名称；
- l: 使用长格式显示文件条目，包括连接数目、所有者、大小、最后修改时间、权限等；
- t: 按文件修改时间进行排序，而不是使用文件名排序；
- C: 按列纵向对文件名排序；
- F: 在文件名后加上一个符号来表示文件类型；
- Cx: 按行跨页对文件名排序；
- CF: 按列列出目录中的文件名，该命令在文件名之后附加一个字符用来区分目录和文件的类型；
 - 目录文件名之后附加一个斜线 (/)
 - 可执行文件名之后附加一个星号 (*)
 - 符号链接文件之后附加一个 @ 字符
 - 普通文件名之后不添加任何字符
- CR: 按多栏格式显示当前目录中的所有文件以及沿目录树向下各个子目录的所有文件，也称作递归列表。该命令可以区分目录和可执行的文件，即在文件名之后附加一个字符。

1.2.2 改变工作目录

进入一个目录，或者说改变当前工作目录使用 `cd` 命令，其命令的语法格式为：

cd 目录名

`cd` 命令带有唯一的一个参数，即表示目标目录的路径名（相对路径名或绝对路径名）。

利用点点 (..) 把工作目录向上移动一级目录：**cd ..**

为了从系统中的任何地方返回到用户主目录，可以使用不带任何参数的 **cd** 命令。

1.2.3 创建目录

使用 **mkdir** 命令创建一个目录或多个目录，以便有效地组织自己的文件。其命令的语法格式为：

```
mkdir [选项] 目录名 [目录名...]
```

同一子目录应包含类似的文件。例如，应建立一个子目录，包含所有的数据库文件；另一个子目录包含电子表格文件；还有一个子目录用来保存某项目相关文件。

-p 选项：同时创建目录及其子目录。

```
mkdir -p 目录名/子目录名
```

1.2.4 删除目录

当目录不再被使用，或磁盘空间已达到使用限定值时，就需要从文件系统中删除失去使用价值的目录。

利用 **rmdir** 命令可从目录中删除一个或多个空的子目录，其语法格式如下：

```
rmdir [选项] 目录名 [目录名...]
```

子目录被删除之前应该是空目录。即该目录中的所有文件必须已被清空。如果该目录中仍有其它文件，则不能用 **rmdir** 命令将其删除。

当前的工作目录必须在被删除目录之上，不能是被删除目录本身，也不能是被删除目录的子目录。

-p 选项：递归地删除指定的目录及其子目录。即：如果指定的目录有子目录，就先删除其子目录，再删除该目录。

-r 选项：递归地删除目录中的所有文件和该目录本身。详见 1.3.5 节有关删除文件命令的介绍。

1.2.5 显示当前目录

在具体操作时，很可能会记不清自己当前所在的目录，命令 **pwd** 用来显示用户当前所在目录树中的位置。如：

```
# pwd
```

```
# /usr/local/rfinput/bin
```

系统给出的信息表示用户当前所在的目录是 `/usr/local/rfinput/bin`。

1.3 文件操作命令

1.3.1 显示文本文件

文本文件是由可打印字符和控制字符（如制表符和换行符）组成的。有几个命令可以显示文本文件。

➤ cat 命令

cat 命令的一般语法是：

cat [选项] 文件名 [文件名...]

该命令运行后，指定文件的内容将在标准输出（通常是屏幕）上显示出来。如果文件内容很长，在一个屏幕中显示不够完整，便会出现屏幕滚动。为控制滚屏，可以按<Ctrl+S>组合键，停止滚屏；按<Ctrl+Q>即可恢复滚屏。

其中选项及其意义如下：

- v: 用一种特殊的形式显示控制字符，除去 LFO 与 TAB
- n: 显示输出行的编号
- b: 显示非空输出行的编号

➤ head 命令

其命令语法如下：

head [显示行数] 文件名 [文件名...]

head 命令在屏幕上显示指定文件最前面的若干行，行数由“显示行数”确定，默认值是 10。

➤ tail 命令

其命令语法如下：

tail [显示行数] 文件名 [文件名...]

tail [+n] 文件名 [文件名...]

在屏幕上显示指定文件末尾的若干行，行数由“显示行数”确定；或从指定行号开始显示，直至该文件的末尾。

➤ more 命令

`more` 命令显示文件内容，每次显示一屏。其语法是：

`more [选项] 文件名 [文件名]`

可在每个屏幕的底部出现一个提示信息，给出至今已显示的该文件的百分比。

可以用几种不同的方法对提示做出回答：

——按 **<Space>** 键，显示文本的下一屏内容。

——按 **<Enter>** 键，只显示文本的下一行内容。

——按斜线符 (**/**)，接着输入一个模式，可以在文本中寻找下一个相匹配的模式。

——按 **h** 键，显示帮助屏，该屏上有相关的帮助信息。

——按 **b** 键，显示上一屏内容。

——按 **q** 键，退出 `more` 命令。

1.3.2 创建新文件

可以利用命令和实用程序来创建文件，如文本编辑器，专门用于把有用的数据放入文件中。然而，有时可能只需要仅有文件名的文件，即空文件。

Linux 系统提供 `touch` 命令来创建空文件。其语法如下：

`touch 文件名 [文件名...]`

不存在的文件名被当作空文件创建。已存在文件的时间标签会更新为当前的时间（默认方式）；它们的数据将原封不动地被保留下来。

1.3.3 拷贝文件

使用 `cp` 命令可以做文件的备份，或者做其他用户文件的个人拷贝。

可以使用 `cp` 命令把一个源文件拷贝到一个目标文件，或者把一系列文件拷贝到一个目标目录中。其语法是：

`cp 源文件 目标文件`

`cp 源文件1 [源文件2...] 目标文件`

在第一种语法格式中，源文件被拷贝到目标文件。

——如果目标文件是目录文件，那么把源文件拷贝到这个目录中，而文件名保持不变；

——如果目标文件不是目录文件，那么源文件就拷贝到该目标文件中，后者原有的内容将被破坏，但文件名不变；

在第二种语法格式中，所有的源文件都被拷贝到目标文件——该目标文件必须是目录文件，所

有源文件的名字均不改变。



cp 命令复制一个文件，而原文件保持不变。如果把一个文件拷贝到一个已经存在的目标文件中，那么，原目标文件的内容将被破坏。

1.3.4 移动和重命名文件

mv 命令用来移动文件或对文件重命名。该命令的语法为：

```
mv 源文件 目标文件
```

```
mv 源文件1 [源文件2...] 目标文件
```

在第一种用法中，源文件被移至目标文件后有两种不同的结果：

- 如果目标文件是某一目录文件的路径，源文件会被移到此目录下，且文件名不变；
- 如果目标文件不是目录文件，则源文件名会变为此目标文件名，并覆盖已存在的同名文件。

在第二种用法中，所有的源文件都会被移至目标文件，此处的目标文件必须是目录文件。所有移到该目录下的文件都将保留以前的文件名。



如果将一个文件移到一个已经存在的目标文件，则目标文件的内容将被覆盖。

如果源文件和目标文件在同一个目录下，mv 的作用就是重命名文件，例如：

```
mv oldname newname
```

1.3.5 删除文件

用 rm 命令删除不需要的文件和目录。该命令的语法为：

```
rm [选项] 文件名1 [文件名2...]
```

在删除文件之前，最好再看一下文件的内容，确定是否要真正删除。

-i 选项：该选项在使用文件扩展名字符删除多个文件时特别有用。此选项会要求用户逐一确定是否要删除文件，必须输入 y 或 Y，按<Enter>键后才能删除文件。如果仅按<Enter>键或其他字符，文件不会被删除。

-r 选项：可以删除目录。当一个目录被删除时，所有文件和子目录将均被删除。**注意，它是一个非常危险的命令选项。**

1.3.6 文件链接

红旗 Linux 具有为一个文件起多个名字的功能，称为链接。这样，只需对一个文件进行修改，

即可完成对所有目录下相应链接文件的修改。

`ln` 命令用来创建链接，语法为：

`ln 源文件 目标文件`

`ln 源文件1 [源文件2...] 目标文件`

在第一种语法格式中，如果目标文件是到某一目录文件的路径，源文件会链接到此目录下，文件名不变；如果目标文件不是到某一目录文件的路径，源文件会链接到此目标文件，并覆盖已经存在的同名文件。

在第二种语法格式中，所有的源文件都被链接到目标文件——该目标文件必须是目录文件。所有源文件的名字均不被改变。

文件链接有两种形式，即硬链接和符号链接。

➤ 硬链接

默认情况下，`ln` 命令创建硬链接。

一个文件的硬链接数可以在目录的长列表格式的第二列中看到，无额外链接的文件链接数为 1。`ln` 命令会增加链接数，`rm` 命令会减少链接数。一个文件除非链接数为 0，否则不会物理地从文件系统中被删除。

对硬链接有如下限制：不能对目录文件作硬链接；不能在不同的文件系统之间作硬链接。

➤ 符号链接

符号链接也称软链接，是将一个路径名链接到某个文件。事实上，它只是一个文本文件，其中包含它提供链接的另一个文件的路径名。另一个文件是实际包含所有数据的文件。所有读写文件内容的命令，当它们被用于符号链接时，将沿着链接方向前进去访问实际的文件。

如果给 `ln` 命令加上 `s` 选项，则建立符号链接。例如：

`ln -s source destination`

符号链接没有硬链接的限制，可以对目录文件作符号链接，也可以在不同文件系统之间作符号链接。

1.3.7 文件内容比较

➤ 比较文本文件

`diff` 命令用于比较文本文件，并显示两文件间的不同。其一般格式是：

`diff 文件1 文件2...`

如果两个文件完全一样，则不显示任何输出。如果有区别，即会分段显示两文件的区别。

➤ 比较数据文件

`cmp` 命令用于比较任何两个包含正文或数据的普通文件。其一般语法为：

```
cmp file1 file2
```

由于二进制数据不能显示到屏幕上，`cmp` 命令只是简单地报告从哪一个字节开始出现不同。

1.3.8 查找文件

➤ `find` 命令

`find` 命令用来查找文件和目录的位置。该命令的语法为：

```
find 路径名 [选项]
```

其中，常用选项有：

-print: 显示输出查找到的结果。如果未指定任何选项，系统默认为 `-print`。如 `find` 命令的最基本的用法是列出指定目录下的所有文件和子目录：

```
# find /usr -print
```

-name: 按文件名查找。

-size: 按文件大小查找。

例如，下面的命令将查找 `/usr` 目录下超过 100k 的文件：

```
# find /usr -size 100k
```

-user: 按文件主查找。

-type: 按文件类型查找。常见的类型有：

类 型	说 明
b	块特别文件
c	字符特别文件
f	普通文件
l	符号链接文件
d	目录文件

➤ locate命令

locate 是一个使用方便且查询速度极快的文件和目录查找命令。该命令的语法为：

locate 文件名 [选项]

使用 locate 命令的前提是要先创建一个用于定位文件或目录位置的 slocate 数据库，而且该数据库应时时更新，这样才能保证 locate 查找结果的准确性。

以下命令用于从/开始创建 slocate 数据库：

locate -u

数据库创建后就可以查找文件了。例如，要查找所有关于 **telnet** 命令的文件，可使用：

locate telnet

locate 命令将在其数据库中检查所有与 telnet 匹配的文件和目录并在屏幕上显示结果。

更新 slocate 数据库的命令是 updatedb，需要以 root 用户身份执行此命令。

一般情况下，系统管理员会设置由 cron 程序在夜间自动更新数据库。cron 是一个后台守护进程，它定期执行计划好的任务。关于设置任务计划的详情，请参阅本手册第 4 章相关内容。

1.3.9 在文件中查找正文

grep 命令用来在文本文件中查找指定模式的词或短语，并在标准输出上显示包括给定字符串的所有行。grep 命令的语法为：

grep [选项] 查找模式 文件名 [文件名...]

默认情况下，grep 在查找模式时是区分大小写的；如果不想区别大小写，可以用选项 **-i**。

例如，下面的命令将在/etc 目录及其子目录下的所有文件中查找字符串“hello world”出现的次数：

grep 'hello world' /etc/*/*

查找模式可能是唯一的参数，如果在模式中使用 shell 元字符，通常要使单引号（'）把它括起来。

1.4 文件权限操作

在多用户操作系统中，出于安全性的考虑，需要给每个文件和目录加上访问权限，严格地规定每个用户的权限。同时，用户可以为自己的文件赋予适当的权限，以保证他人不能修改和访问。

1.4.1 改变文件主

Linux 为每个文件都分配了一个文件所有者，称为文件主，对文件的控制取决于文件主或超级用户（root）。文件或目录的创建者对创建的文件或目录拥有特别使用权。

文件的所有关系是可以改变的，`chown` 命令用来更改某个文件或目录的所有权。`chown` 命令的语法格式是：

`chown` [选项] 用户或组 文件1 [文件2...]

用户可以是用户名或用户 ID。文件是以空格分开的要改变权限的文件列表，可用通配符表示文件名。

如果改变了文件或目录的所有权，原文件主将不再拥有该文件或目录的权限。

系统管理员经常使用 `chown` 命令，在将文件拷贝到另一个用户的目录下以后，让用户拥有使用该文件的权限。

1.4.2 改变用户组

在 Linux 下，每个文件又同时属于一个用户组。当创建一个文件或目录，系统会赋予它一个用户组关系，用户组的所有成员都可以使用此文件或目录。

文件用户组关系的标志是 GID。文件的 GID 只能由文件主或超级用户（root）来修改。`chgrp` 命令可以改变文件的 GID，其语法格式为：

`chgrp` [选项] group 文件名

其中，`group` 是用户组 ID。文件名是以空格分开的要改变属组的文件列表，它支持通配符。

1.4.3 文件权限设置

Linux 系统中的每个文件和目录都有访问许可权限，用它来确定谁可以通过何种方式对文件、目录进行访问与操作。

访问权限规定三种不同类型的用户：

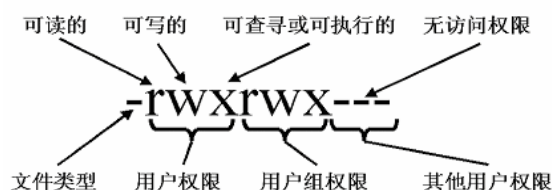
- 文件主（owner）
- 同组用户（group）
- 可以访问系统的其他用户（others）

访问权限规定三种访问文件或目录的方式：

- 读（r）
- 写（w）

➤ 可执行或查找 (x)

当用 `ls -l` 命令或 `l` 命令显示文件或目录的详细信息时，最左边的一列为文件的访问权限。其中各位的含义如下：



➤ 文件访问权限

读权限 (r)： 只允许指定用户读其内容，而禁止对其做任何更改操作。将所访问的文件内容作为输入命令都需要有读的权限。例如：`cat`、`more` 等。

写权限 (w)： 允许指定用户打开并修改文件。例如命令 `vi`、`cp` 等。

执行权限 (x)： 指定用户将该文件作为一个程序执行。

➤ 目录访问权限

读权限 (r)： 可以列出存储在该目录下的文件，即读目录内容列表。这一权限允许 `shell` 使用文件扩展名字符列出相匹配的文件名。

写权限 (w)： 允许从目录中删除或添加新的文件，通常只有目录主才有写权限。

执行权限 (x)： 允许在目录中查找，并能用 `cd` 命令将工作目录改到该目录。

1.4.4 改变文件权限

1.4.4.1 以符号模式改变权限

`chmod` 用于改变文件或目录的访问权限。用户可以用它控制文件或目录的访问权限。只有文件主或超级用户 `root` 才有权用 `chmod` 改变文件或目录的访问权限。

`chmod` 命令的语法为：

chmod key 文件名

key 由以下各项组成：

[who] [操作符号] [mode]

其中，操作对象 `who` 可以是下述字母中的任一个或组合：

u	user, 表示用户, 即文件或目录的所有者。
g	group, 表示同组用户, 即与文件属主有相同组 ID 的所有用户。
o	others, 表示其他用户。
a	all, 表示所有用户, 它是系统默认值。

操作符号可以是:

+	添加某个权限
-	取消某个权限
=	赋予给定权限并取消其他所有权限 (如果有的话)

mode 所表示的权限可用下述字母的任意组合:

r	可读
w	可写
x	可执行
s	在文件执行时把进程的属主或组 ID 置为该文件的文件属主
t	保存程序的文本到交换设备上
u	与文件属主拥有相同的权限
g	与和文件属主同组的用户拥有相同的权限
o	与其他用户拥有相同的权限

这三部分必须按顺序输入。可以用多个 key, 但必须以逗号间隔。

1.4.4.2 以绝对方式改变权限

通常也可以用 `chmod` 命令配以不同类型的 key 直接设置权限。此时以数字代表不同的权限。这里 key 可以包括三个 (或三个以上) 的数字, 每个数字表示对不同类型用户的权限。

数字表示的属性的含义:

0 表示禁止该权限, 1 表示可执行权限, 2 表示可写权限, 4 表示可读权限, 然后将其相加。所以数字属性的格式应为 3 个从 0 到 7 的八进制数, 其顺序是 (u) (g) (o)。

通常, key 是以三位八进制数字出现的, 第一位表示用户权限、第二位表示组权限, 第三位表示其他用户权限。

例如, 要使文件 `myfile` 的文件主和同组用户具有读写权限, 但其他用户只可读, 可以用以下命令指定权限:

```
chmod 664 myfile
```

1.4.5 默认权限

默认情况下，系统将创建的普通文件的权限设置为 `-rw-r-r--`，即文件主对该文件可读可写（`rw`），而同组用户和其他用户都只可读；同样，在默认配置中，将每一个用户主目录的权限都设置为 `drwx-----`，即只有文件主对该目录可读、写和可查询（`rwx`），即用户不能读其他用户目录中的内容。

用户可以修改新建文件的默认存取权限，如使用如下命令：

```
umask u=rwx,g=,o=
```

它会在创建新文件时给文件主以全部权限，而同组用户及其他用户没有任何权限。

1.5 定向和管道

执行一个 Shell 命令行通常会自动打开三个标准文件，即标准输入文件（`stdin`），通常对应终端的键盘；标准输出文件（`stdout`）和标准错误输出文件（`stderr`），这两个文件都对应终端的屏幕。进程从标准输入文件中得到数据，将正常输出数据输出到标准输出文件，而将错误信息送到标准错误文件中。

下面以 `cat` 命令为例，`cat` 命令的功能是从命令行给出的文件中读取数据，并将这些数据直接送到标准输出。例如，使用以下命令：

```
$ cat config
```

将会把文件 `config` 的内容依次显示到屏幕上。但是，如果 `cat` 的命令行中没有参数，它就会从标准输入中读取数据，并将其送到标准输出。例如：

```
$ cat
```

```
Hello world
```

```
Hello world
```

```
$
```

直接使用标准输入/输出文件存在以下问题：

- 1) 数据从标准终端输入时，输入的数据只能用一次，下次再想用这些数据时则需要重新输入；而且在终端上输入时，若输入有误修改起来较为困难。
- 2) 输出到终端屏幕上的信息只能看不能修改。用户无法将输出的内容进行更多处理，如将输出作为另一命令的输入做进一步地处理等。

为了解决上述问题，Linux 系统为输入、输出的传送引入了另外两种机制，即输入/输出重定向和管道。

1.5.1 输入重定向

输入重定向是指把命令（或可执行程序）的标准输入重定向到指定的文件中。即输入可以不是来自于键盘，而来自某个指定的文件。

例如，命令 `wc` 统计指定文件包含的行数、单词数和字符数。如果仅在命令行上键入：

```
$ wc
```

`wc` 将等待用户的输入，这时 Shell 好像宕掉一样，从键盘键入的所有文本都出现在屏幕上，但并没有任何结果，直至按下 `<Ctrl+D>`，`wc` 才将命令结果写在屏幕上。

如果给出一个文件名作为 `wc` 命令的参数，`wc` 将返回该文件所包含的行数、单词数和字符数。

另一种把 `/etc/passwd` 文件内容传给 `wc` 命令的方法是重定向 `wc` 的输入。输入重定向的一般形式为：

命令<文件名

可以用下面的命令把 `wc` 命令的输入重定向为 `/etc/passwd` 文件：

```
$ wc < /etc/passwd
```

```
20 23 726
```

大多数命令都以参数的形式在命令行指定输入文件的文件名，所以输入重定向并不经常使用。尽管如此，当使用一个不接受文件名作为输入参数的命令，或需要的输入内容存在于一个文件里时，就能利用输入重定向来解决问题了。

1.5.2 输出重定向

输出重定向是指把命令（或可执行程序）的标准输出或标准错误输出重新定向到指定文件中。这样，该命令的输出就不会显示在屏幕上，而是写入到指定的文件中。

输出重定向比输入重定向更常用。例如，如果某个命令的输出很多，在屏幕上不能完全显示，那么将输出重定向到某个文件中，然后再用文本编辑器打开该文件，即可查看输出信息；如果想保存一个命令的输出，也可以使用此方法。还有，输出重定向可以用于把一个命令的输出当作另一个命令的输入。

输出重定向的一般形式为：

命令>文件名

例如：

```
$ ls > directory.out
```

```
$ cat directory.out
```

```
ch1.doc ch2.doc ch3.doc chimp config mail/ test/
```

将 ls 命令的输出保存为一个名为 directory.out 的文件。



如果“>”符号后边的文件已存在，那么这个文件将被覆盖。

为避免输出重定向中指定文件只能存放当前命令的输出重定向的内容，Shell 提供了输出重定向的一种追加手段。

输出追加重定向与输出重定向非常相似，区别仅在于输出追加重定向的功能是把命令（或可执行程序）的输出结果追加到指定文件的最后，而该文件原有内容不被破坏。

如果要某条命令的输出结果追加到指定文件的后面，可以使用追加重定向操作符“>>”。形式为：

命令>>文件名

例如：

```
$ ls *.doc>>directory.out
```

```
$ cat directory.out
```

```
ch1.doc ch2.doc ch3.doc chimp config mail/ test/
```

```
ch1.doc ch2.doc ch3.doc
```

和程序的标准输出重定向一样，程序的错误输出也可以重新定向。使用符号 2>（或追加符号 2>>）表示对错误输出设备重定向。例如下面的命令：

```
$ ls /usr/tmp 2> err.file
```

可在屏幕上看到程序的正常输出结果，但又将程序的任何错误信息送到文件 err.file 中，以备将来检查用。

还可以使用另一个输出重定向操作符(&>)将标准输出和错误输出同时送到同一文件中。例如：

```
$ ls /usr/tmp &> output.file
```

利用重定向将命令组合在一起，可实现系统单个命令不能提供的新功能。例如使用下面的命令序列，即统计了/usr/bin 目录下的文件个数。

```
$ ls /usr/bin > /tmp/dir
```

```
$ wc -w < /tmp/dir
```

```
459
```

1.5.3 管道

将一个程序或命令的输出作为另一个程序或命令的输入有两种方法，一种是通过一个临时文件将两个命令或程序结合在一起，例如上节例子中的/tmp/dir 文件将 ls 和 wc 命令联在一起；另一种是 Linux 所提供的管道功能，这种方法比前一种方法更为方便。

管道可以把一系列命令连接起来，这意味着第一个命令的输出会作为第二个命令的输入通过管道传给第二个命令，第二个命令的输出又会作为第三个命令的输入，以此类推。显示在屏幕上的是管道行中最后一个命令的输出。

通过使用管道符 “|” 来建立一个管道行。用管道重写上面的例子：

```
$ ls /usr/bin|wc -w  
1789
```

再如：

```
$ cat sample.txt|grep “High” |wc -l
```

管道将 cat 命令的输出送给 grep 命令。grep 命令在输入里查找单词 High，grep 命令的输出则是所有包含单词 High 的行，这个输出又被送给 wc 命令，wc 命令统计出输入中的行数。假设 sample.txt 文件的内容如下：

```
Things to do today:  
  
Low:Go grocery shopping  
  
High:Return movie  
  
High:Clear level 3 in Alien vs. Predator
```

那么该管道行的结果是 2。

1.6 进程和作业控制命令

简单地说，进程是一个程序或任务的执行过程。在 Linux 系统中，执行任何一个命令都会创建一个或多个进程。即命令是通过进程实现的。

从进程的角度可以更好地理解 Linux 操作系统的多任务概念。对于系统管理员来说，管理系统进程是日常管理的重要部分。

➤ 用ps查看系统中的进程状态

可以通过 ps 命令观察进程状态，它会把当前瞬间进程的状态显示出来。可根据显示的信息确定哪个进程正在运行，哪个进程是被挂起，还是遇到了某些困难，进程已运行了多久，进程正在使用的资源，进程的相对优先级，及进程的标识号（PID）。这些信息对用户很有用，对于系统管理员来说

更为重要。

ps 命令的一般用法是：

ps [option] [arguments]...

ps 命令有以下几个主要的参数：

- a: 显示包括系统中所有用户进程的状态
- f: 显示进程和子进程的树形目录
- l: 以长列表形式显示进程信息
- r: 只显示正在运行的进程
- u: 以用户格式显示进程信息，给出用户名和起始时间
- pids: 显示指定 ID 的进程信息

如果不带任何选项，ps 命令会列出每个与您的当前 shell 有关的进程的 PID。结果如下：

```
PID TTY TIME CMD
596 pts/0    00:00:00 bash
627 pts/0    00:00:00 vi
628 pts/0    00:00:00 ps
```

其中，各字段的含义如下：

- PID: 进程标识号
- TTY: 开始该进程的终端号
- TIME: 报告进程累计使用的 CPU 时间
- CMD: 正在执行的进程名

要获得一个完整的进程信息列表，可使用带有下列选项的 ps 命令：

ps -aux

除列出以上字段外，它还列出了 CPU 使用率（%CPU）、内存使用率（%MEM）、虚拟映像大小（SIZE）、驻留数据集大小（RSS）、终端号（TTY）、状态（STAT）等。

➤ top命令

top 命令用于读入计算机系统的信息，这些信息包括当前的系统数据和进程的状态等。输入 top 命令后，屏幕输出如下：

```

5:23pm up 1:39, 4 users, load average: 0.46, 0.35, 0.33
66 processes: 64 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 0.5% user, 1.7% system, 0.0% nice, 97.6% idle
Mem: 126432K av, 118336K used, 8096K free, 144K shrd, 3668K buff
Swap: 265032K av, 85644K used, 179388K free

```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	COMMAND
834	root	17	0	58552	6364	6144	S	0.7	5.0	3:00	X
1470	root	17	0	1156	1112	884	R	0.7	0.8	0:00	top
1015	root	19	0	3756	3284	3072	S	0.5	2.5	0:25	ksnapshot
1	root	8	0	108	64	64	S	0.0	0.0	0:07	init
2	root	9	0	0	0	0	SW	0.0	0.0	0:00	keventd
3	root	9	0	0	0	0	SW	0.0	0.0	0:00	kpm-idled
4	root	19	19	0	0	0	SWN	0.0	0.0	0:00	ksoftirqd_CPU0
5	root	9	0	0	0	0	SW	0.0	0.0	0:01	kswapd
6	root	9	0	0	0	0	SW	0.0	0.0	0:00	kreclaimd
7	root	9	0	0	0	0	SW	0.0	0.0	0:00	bdfush
8	root	9	0	0	0	0	SW	0.0	0.0	0:00	kupdated
9	root	-1	-20	0	0	0	SW<	0.0	0.0	0:00	mdrecoveryd
65	root	9	0	864	36	36	S	0.0	0.0	0:00	minilogd
73	root	9	0	0	0	0	SW	0.0	0.0	0:00	khud
555	root	8	0	64	4	4	S	0.0	0.0	0:00	apmd
623	root	9	0	108	4	4	S	0.0	0.0	0:00	automount
679	root	9	0	88	4	4	S	0.0	0.0	0:00	lpd
738	root	9	0	104	60	60	S	0.0	0.0	0:00	gpm

图中的每一列显示了系统的详细信息，图中开头几行的信息含义如下：

- Uptime:** 显示当前时间、自上次启动系统开始过去的时间、激活用户的数目以及在过去 1、5 和 15 分钟之内的 CPU 平均占用情况。
- Process:** 显示了系统所有的进程，并把进程按挂起、运行、创建和停止分类。
- CPU states:** 统计被用户和系统占用的当前 CPU 的状态。
- Mem:** 统计当前内存的占用状态。
- Swap:** 统计了 swap 区域的占用情况。

在 top 命令中显示了进程的列表，其中包括的内容有：PID、用户、优先级、nice 参数、所需的内存信息（SIZE、RSS、SHARE）、状态（STAT）、CPU 占用的百分比、占用的内存信息、已用的计算机时间和各自的程序调用（COMMAND）等。关于 top 命令的详细使用信息，可查看其在线帮助。

➤ 用 kill 命令终止进程

运行过程中，可能在某一时刻，系统中有的进程出现了问题，不能正常运行，但也不能正常退出。这时可以使用 kill 命令终止进程的执行，释放这些进程占用的系统资源，常用的 kill 命令的格式为：

kill [-s signal] pid

kill -l [signal]

命令的选项和参数的意义如下：

pid 给出了需要结束的进程的 PID，可以通过命令 ps 获得进程的 PID。在命令 kill 中可以一次列出许多的进程 PID。

-s signal 是一个可选参数，用来给出发给进程的信号。默认情况下，命令 **kill** 给进程发 **TERM** 信号，该信号将通知进程退出。如果进程不接收该信号，可以通过参数 **-9** 强制结束进程。

-l 该参数要求 **kill** 命令列出它可以发给进程的所有信号。

➤ 用 **at** 安排任务

at 命令用于实现在指定的时间运行您所安排的作业。**at** 命令的一般用法如下：

at [选项] 时间 [日期]

at hh:mm:	用指定的小时（hh）和分钟（mm）（24 小时制）安排作业；
at hh:mm month day year:	用指定的年（year）、月（month）、日（day）、小时（hh）和分钟（mm）安排作业；
at -l:	列出已安排的作业；
at now +count time_units:	作业运行的时间安排在现在的时间加上 count 个时间单位。时间单位（time_units）可以是分钟、小时、天或星期；
at -d job_id:	取消作业号与 job_id 相同的作业。

由 **at** 命令调度的命令是在 **at** 命令行后输入的命令列表。**at** 的命令列表可以从标准输入（stdin）得到。如果标准输入来自键盘，您应该在输入完命令之后键入 **<Ctrl+D>**，表明输入结束。

1.7 基本网络命令

Red Flag Asianux Server 3 具有强大的网络功能，提供了丰富的网络应用程序，完全支持 TCP/IP 协议。在网络环境下，可以进行远程注册、远程命令调用、传送文件等操作。本节介绍了几个基本的网络操作命令。

1.7.1 telnet 命令

telnet 命令是 Linux 下的远程登录工具，只要拥有合法的注册名和口令，就能像使用本地机器一样访问远程计算机了。**telnet** 也允许用户通过输入注册名和口令从远程网点登录到自己的计算机上，从而通过网络或电话线完成检查电子邮件、编辑文件和运行程序等操作。但是 **telnet** 只能在字符终端方式下工作，不支持图形用户界面。

telnet 的基本用法是：

telnet [选项] IP地址/主机名

命令键入后，**telnet** 即会启动一个远程会话，本命令可使用的选项参数主要有：

-d 启动调试功能

-a	自动注册
-n tracefile	打开跟踪程序，把跟踪程序数据保存在 tracefile 中
-e escape_char	将会话的转义字符设置为 escape_char
-l user	把用户名发送给远程系统，以便自动注册。本参数自动包括 -a 参数
port	指出与远程系统连接的端口号。如不指定，将连接到缺省端口

成功地连接到远程计算机上后，telnet 即可显示登录信息，并提示用户输入注册名与口令。注册成功后，便可以开始工作了。

在使用 telnet 后需要退出注册回到本地的 shell 命令提示符下。

1.7.2 ftp命令

FTP（文件传输协议）是在 TCP/IP 网络计算机之间传输文件的简单而有效的方法。ftp 命令的功能是在本地机和远程机之间传送文件。它允许用户传输 ASCII 文件和二进制文件。在 ftp 会话过程中，用户可以通过使用 ftp 客户程序连接到另一台计算机上。用户可以在目录中上下移动、列出目录内容、把文件从远程机拷贝到本地机上、把文件从本地机传输到远程系统中。前提是必须在本地和远程文件系统中具有进行这些操作的权限。

ftp 命令的基本格式如下：

ftp [选项] IP地址/主机名

可以用 help 命令取得可供使用的命令清单，也可以在 help 命令后面指定具体的命令名称，获得该命令的说明。

ls	列出远程机的当前目录
cd	在远程机上改变工作目录
lcd	在本地机上改变工作目录
ascii	设置文件传输方式为 ASCII 模式
binary	设置文件传输方式为二进制模式
close	终止当前的 ftp 会话
hash	每次传输完数据缓冲区中的数据后就显示一个#号
get (mget)	从远程机传送指定文件到本地机
put (mput)	从本地机传送指定文件到远程机
open	连接远程 ftp 站点
quit	断开与远程机的连接并退出 ftp
?	显示本地帮助信息

!
转到 Shell 中

随着 Internet 的迅速发展，提供信息资源的网站往往无法为每个要使用 FTP 的用户开设帐号，因此出现了一种匿名 FTP 机制：可以使用 anonymous 用户名，用自己的电子邮件地址作为口令来访问大多数共享信息资源。但是因为安全的原因，匿名 FTP 的可访问资源是有限的，而且有些网站也不提供此服务。

1.7.3 ping命令

ping 命令用来确定网络上的主机是否可到达和到达速率。ping 命令的格式为：

ping [选项] IP地址/主机名

ping 命令将大小固定的数据包发送给对方，并要求对方返回。当终止 ping 命令时，会显示一些统计数据。通过数据判断是否返回以及返回时间，用户可以确定对方是否可到达，是否开机，以及网络延时时间。如果要退出请按<Ctrl+C>中断。

1.7.4 finger命令

使用 finger 命令来查询系统用户的信息，该命令基本格式为：

finger [选项] 用户名@主机名

运行 finger 命令后会显示系统中某个用户的用户名、主目录、停滞时间、登录时间、登录 shell 等信息，查询远程机上的用户信息时，就需要使用在用户名后面加上“@主机名”的方式。

第2章 文件系统管理

对于任何一个成熟的操作系统而言，文件系统管理都是一个十分重要的部分。文件系统管理的好坏会直接影响到操作系统的性能和安全。

2.1 文件系统

文件系统是操作系统在硬盘或分区上保存文件信息的方法和数据结构，也就是文件在硬盘或分区上的组织方式。

作为一种类 UNIX 操作系统，大部分 Linux 文件系统具有类似的通用结构，其关键概念有超级块（superblock）、索引节点（inode）、数据块（data block）、目录块（directory block）。

超级块中包含了关于该硬盘或分区上的文件系统的整体信息，如文件系统的大小等；超级块后面的数据结构是索引节点，它几乎包含了针对某个具体文件的全部信息，如文件的存取权限、拥有者、文件大小、建立时间及对应的目录块和数据块等；数据块是真正存储文件内容的位置。但是索引节点中不包括文件的名字，文件名是放在目录块里的。目录块里包含有文件的名字及此文件的索引节点编号。

2.1.1 Red Flag Asianux Server 3 支持的文件系统类型

Red Flag Asianux Server 3 系统的重要特征之一是支持多种文件系统。这样它更为灵活并且可以与许多其他种类的操作系统交换数据，其中最常用的是以下几种：

- **ext3**: ext2 的升级版，是 Red Flag Asianux Server 3 默认的文件系统类型，其主要优点是在 ext2 的基础上加入了记录数据的日志功能。可方便地从 ext2 迁移至 ext3，且支持异步的日志。
- **ext2**: 支持标准 Unix 文件类型，可用于多种存储介质，向上兼容性好，支持长达 255 个字符的文件名。
- **reiserfs**: 一种新型的文件系统，通过完全平衡树结构来容纳数据，包括文件数据，文件名以及日志支持。ReiserFS 还可以支持海量磁盘和磁盘阵列，并能上面继续保持很快的搜索速度和很高的效率。
- **vfat**: Windows 9X/2000 及 NT 操作系统使用的扩展 DOS 文件系统，提供了对长文件名的支持。
- **xf**s: SGI 的 xfs 是非常好的 64 位高性能日志文件系统，它为 Linux 社区提供了一种健壮、优秀、功能丰富的文件系统，它具有的可伸缩性能够满足最苛刻的存储需求。
- **iso9660**: 标准的 CD-ROM 文件系统。其中的 Rock Ridge 扩展允许长文件名的自动支持。
- **NFS**: 允许在多台计算机之间共享文件系统的网络文件系统。

此外，还支持一些古老的文件系统类型，如 MINIX、Msdos、Hpfs、sysv 等。

2.1.2 文件系统的创建、加载与卸载

2.1.2.1 建立文件系统

一个分区或磁盘在作为文件系统被使用之前，先要初始化将记录数据的结构写入磁盘，这个过程叫做建立文件系统。

命令 **mkfs** 用于创建文件系统，它可以在任何指定的块设备上建立不同类型的文件系统。

mkfs 命令的语法格式如下：

mkfs [-v] [-t fs-type] [fs-options] device [size]

mkfs 命令中各项参数的意义如下：

-v：强迫产生长格式输出；

-t fs-type：选择文件系统的类型；

fs-option：将要建立的文件系统选项，它可以是以下选项：

选 项	说 明 和 描 述
-c	查找坏块，并初始化坏块列表
-l filename	从文件 filename 中读初始的坏块表
-v	让文件系统程序产生长格式输出

device：将创建文件系统所在设备的设备号；

size：文件系统的大小；

例如：要在软盘上创建一个 ext2 的文件系统，用以下命令：

```
# mkfs -t ext2 /dev/fd0
```

2.1.2.2 加载文件系统

成功地建立了文件系统后，还需要将文件系统加载或称安装（mount）到 Linux 目录树的某个位置才能使用。文件系统所连接到的目录被称为加载点或安装点。对于系统硬件设备，Linux 并不使用设备号或驱动器号来访问，而是将它们对应为/dev 目录下的一个（也可能是多个）文件。

文件系统的安装，可以在系统引导过程中自动安装，也可以使用命令手工装载。

加载文件系统的命令为 mount，该命令的语法格式如下：

mount [-t fs-type] device mountpoint

其中：device 代表文件系统的存储设备；mountpoint 代表文件系统将要被放置在目录系统中的位置，即载入点。

mount 命令常用如下几个选项：

选 项	说 明 和 描 述
-a:	加载符合要求的所有文件系统，如果不加其他参数，将加载 /etc/fstab 文件中列出的所有文件系统。
-o:	用于确定文件系统的读/写限制，ro（只读）、rw（读写）等。
-f:	完成操作步骤，并不真正安装文件系统。

例如：把/dev/hda8 上类型为 ext2 的文件系统加载到目录/mnt/tmp 下，并使之按只读方式被安装。

```
# mount -t ext2 -o ro /dev/hda8 /mnt/tmp
```



文件系统的加载位置 *mountpoint* 必须是系统中已存在的目录；否则，需要在加载前创建此目录。

2.1.2.3 卸载文件系统

除了根文件系统之外，其他文件系统都是可以拆卸的。卸载文件系统的命令是 `umount`，其格式如下：

`umount device mountpoint`

该命令使用设备名和安装点为参数，用于卸下与设备名或是安装点对应的文件系统。



不能卸载当前正在使用的文件系统，否则系统会报出错。正确的方法是待完全退出安装点所在目录后，再执行卸载命令。

2.1.2.4 用fstab文件配置文件系统

一般来说，用户经常使用的文件系统是比较固定的，如果每次使用时都进行加载是很麻烦的，而且有时候需要一次安装很多的文件系统，可以考虑定义一个在系统引导时自动安装文件系统的方法。

Linux 中使用 `/etc/fstab` 文件能够完成这一功能，`fstab` 文件中列出了引导时需安装的文件系统的类型、加载点及可选参数。`fstab` 文件是一个文本文件，可以方便地通过编辑工具进行修改。



请在修改前备份原来的 `/etc/fstab` 文件，以防修改出错导致下次系统引导时文件系统不能正确加载。

以下给出的是一个实际的 `/etc/fstab` 文件（您的系统不一定与之相同）

```
# cat /etc/fstab
/dev/hda1      /              ext3          defaults      1      1
/dev/hda5      /home          ext3          defaults      1      2
/dev/hda6      swap           swap          defaults      0      0
/dev/cdrom /mnt/cdromiso9660 noauto,owner,ro 0      0
/dev/fd0       /mnt/floppy    ext3          noauto,owner  0      0
none          /proc          proc          defaults      0      0
none          /dev/pts       devpts        gid=5,mode=620 0      0      0
```

`/etc/fstab` 文件也称为文件系统安装表，它的每一行代表一个需要安装的文件系统，其格式如下：

`device mountpoint fstype options dump passno`

其中：

device: 指定要加载的文件系统设备
mountpoint: 指定文件系统的加载点
fstype: 指定安装文件系统的类型
options: 使用逗号隔开的安装参数列表
dump: 确定文件系统两次备份之间的时间

passno: 指定系统引导时检查文件系统的顺序，根文件系统为 1，其余值为 2，如果没有指定，表示引导时文件系统不被检查。

2.1.3 维护文件系统

文件系统是系统数据和资源的存储位置，所以应定期检查文件系统以发现损坏的文件并及时加以修补。

对文件系统进行检查可以通过使用 fsck 工具来完成，该命令的格式为：

```
fsck [options] filesystem ...
```

关于 fsck 的详细用法，请参考它的 man page。



用 fsck 检查文件系统时，最好先卸下该文件系统，这样可以防止在检查过程中有其它程序正在操作该文件系统。

2.1.4 常用文件系统管理命令

➤ df命令

使用 df 命令可检查文件系统的磁盘空间占用情况，提供所有映射文件系统的空闲空间信息，其命令的语法格式为：

```
df [选项]... [文件系统] ...
```

该工具默认以 KB 为单位显示已用与可用的磁盘空间，查看以 MB 和 GB 为单位的信息，使用 df -h 命令。

➤ du命令

使用 du 命令可以显示被目录占用的空间信息，此命令可以进入被统计目录的子目录中，并显示出子目录的统计信息，常用的选项如下：

-a: 同时显示出目录和文件的磁盘使用情况；

-s: 只显示磁盘的总体使用情况；

使用不加目录名的 du 命令将会显示出当前目录下的所有信息。

2.1.5 使用设备

在 Red Flag Asianux Server 3 中，可以方便地使用各种驱动器、文件系统和网络设备，包括 Linux 分区、MS DOS 和 Windows 分区、U 盘以及 CDROM 文件系统。

➤ 使用CDROM

将光盘放入光盘驱动器中，在 shell 提示符下键入以下命令：

```
# mount /dev/cdrom /mnt/cdrom
```

它通知操作系统自动探测文件系统并安装它，被安装的设备为/dev/cdrom，安装点为/mnt/cdrom。如果命令生效，光盘中的内容将出现在目录/mnt/cdrom 下。

如果安装命令失败，首先要确认/dev/cdrom 设备存在。如果使用的是 IDE CDROM，对应设备文件名可能是/dev/hdb、/dev/hdc 等；如果使用 SCSI CDROM，对应设备文件名可能为/dev/sda1，/dev/sda2...

假设/dev/cdrom 不存在，而 CDROM 设备文件名为/dev/hdb，可以使用如下命令创建一个到/dev/cdrom 的符号链接。

```
# ln -s /dev/hdb /dev/cdrom
```

如果系统提示“设备已经安装（mounted）或目录忙”，可能是由于当前目录是挂载点/mnt/cdrom 造成的，必须切换到其它目录下才能进行。

执行完对光盘的操作后，在 shell 提示符下键入以下命令将其卸载。

```
# umount /mnt/cdrom
```

➤ 使用软盘

使用软盘之前，需要先挂载它。将软盘插入软盘驱动器，执行如下命令：

```
# mount /dev/fd0 /mnt/floppy
```

在加载时，首先要确认/mnt/floppy 目录存在并为空，并且/mnt/floppy 不能是系统中任何用户的当前目录。

成功安装后，软盘的文件出现在/mnt/floppy 目录下，这些文件对所有用户可读，但只有 root 才可以修改、删除这些文件。

当执行完对软盘的操作，将其从软驱中取出之前，需要先卸载软盘。命令如下：

```
# umount /mnt/floppy
```

➤ 关于mtools工具

安装了系统中提供的 mtools 工具后，就可以使用 m 系列命令实现对 DOS/Windows 格式软盘的快速访问了。该系列命令包括：

mtools 系列命令

命 令	说 明 和 描 述
mcd	进入 DOS 子目录
mcopy	拷贝 DOS 文件
mdel	删除 DOS 文件
mdir	察看 DOS 目录内容
mformat	格式化 DOS 磁盘
mmd	创建 DOS 目录
mmove	移动 DOS 下的文件
mren	将 DOS 下的文件改名

➤ 使用Zip驱动器

安装 SCSI 仿真并口扩展 Zip 驱动器的命令如下：

```
# mount -t vfat /dev/sda4 /mnt/zipdrive
```

使用“-t vfat”选项是因为 Zip 驱动器事先已经格式化为 vfat 文件系统，这一文件系统属于 Windows 文件系统，支持长文件名。

卸载 Zip 驱动器的命令为 umount。如果不卸载，盘片将无法弹出。

如果 Zip 驱动器安装不成功。则在安装结束后，应该使用以下命令把并行端口 zip 驱动器模块加载到内核中：


```
# /sbin/insmod parport
# /sbin/insmod ppa
```

为了让上面的指令在每次系统启动时自动执行，应该将其加在文件/etc/rc.d/rc.local 的结尾。如果仍然不能识别，应编辑文件/etc/modules.conf，下面是文件/etc/modules.conf 的示例行。


```
alias parport_lowlevel parport_pc
```

2.2磁盘分区管理

利用 parted 程序可以方便地进行磁盘分区的管理和定制，如查看现存的分区表、改变分区的大小、删除分区、创建新分区等。



改变系统的硬盘分区是一件非常危险的事情，即使对于有丰富经验的系统管理员，我们仍建议在改变磁盘分区前进行必要的备份。



关于磁盘分区的命名设计与分区方法和原则的详细信息，敬请参阅：《Red Flag Asianux Server 3 安装手册》中的相关内容。

在 shell 提示符下以超级用户身份键入如下命令（/dev/hdb 表示要定制的设备名）。

```
# parted /dev/hdb
```

启动 parted 后，在(parted)提示下键入 help 将显示可用命令的列表。下表列出的是用户最常用的 parted 命令。

parted 命令

命 令 和 参 数	说 明 和 描 述
check minor-num	对文件系统进行简单检查
cp from to	把文件系统从一个分区复制到另一个分区；from 和 to 表示分区设备名中的数字
help	显示可用的命令列表
mklabel label	为分区表创建磁盘标签
mkfs minor-num fs-type	创建类型为 fs-type 的文件系统
mkpart part-type fs-type start-mb end-mb	制作分区，不创建新文件系统

命令和参数	说明和描述
mkpartfs part-type fs-type start-mb end-mb	制作分区并创建指定的文件系统
move minor-num start-mb end-mb	移动分区
print	显示分区表
quit	退出 parted 程序
resize minor-num start-mb end-mb	重新划分分区大小，从 start-mb 到 end-mb
rm minor-num	删除分区
select device	选择另一个设备来定制，不需重新启动 parted
set minor-num flag state	在分区上设置标志；state 可以是 on 或 off



要新建、删除分区或重新划分分区大小，分区所在设备不能正在被使用，即分区不能被挂载，且交换空间不能被启用。如果分区中不包含正在被使用的文件，可以用 `umount` 命令来卸载分区，使用 `swapoff` 命令来关闭交换空间。

2.2.1 查看分区表

启动 parted 后，键入 `print` 命令来查看分区表，屏幕将输出如下信息。

```
(parted) print
Disk geometry for /dev/hda: 0.000-78533.437 megabytes
Disk label type: msdos
Minor   Start      End      Type      Filesystem  Flags
1         0.031    6997.060  primary   ext3         boot
2      6997.061    7506.936  primary   linux-swaps
3      7506.936   11507.497  primary   ext3
4     11507.498   78528.669  extended
5     11507.559   15508.059  logical   ext3
6     15508.090   18512.402  logical   reiserfs
(parted) █
```

第一行显示了磁盘的大小，第二行显示磁盘标签类型，后面部分为分区表，其中：

Minor 域表示分区设备名中的数字，例如数字 1 代表 `/dev/hda1`；**Start** 和 **End** 分别表示对应分区在硬盘上的起止位置，以 MB 为单位；**Type** 代表分区类型，可以是 `primary`、`extended` 和 `logical` 之一；**Filesystem** 是文件系统的类型，可以是 `ext2`、`ext3`、`FAT`、`linux-swaps` 等；**Flags** 域列出了分区被设置的标准，可用的标志有：`boot`、`root`、`swap`、`hidden`、`raid`、`lvm` 等。

2.2.2 创建分区

如果我们要在 `/dev/hdb` 上创建一个新分区，那么输入以下命令启动 parted：

```
# parted /dev/hdb
```

然后用 **print** 命令查看当前的分区表，确认磁盘上是否有足够的空闲空间可用于新分区。



不要试图在正在被使用的设备上创建分区。

根据分区表决定新分区的起止点和分区类型，每个硬盘上只能有四个主分区，如果想拥有四个以上的分区，则必须先划出一个扩展分区，再在扩展分区内创建多个逻辑分区。

例如，要在/dev/hda1 上从 18000MB 到 21000MB 创建一个文件系统为 ext2 的主分区，键入以下命令：

```
mkpart primary ext2 18000 21000
```

如果使用 **mkpartfs** 命令，分区创建后文件系统也会被创建。只要一按下<Enter>键，对分区的改变就会生效，因此在执行前请仔细检查一下命令。

创建了分区后，使用 **print** 命令来确认所建分区已加入分区表中，并具有正确的分区类型、文件系统类型和大小。记住新分区的设备名中的数字以方便后续操作。

使用 **mkpart** 划分的分区还没有格式化，用下面的命令为分区创建文件系统：

```
# mkfs -t ext2 /dev/hdb3
```

接下来，可以为新分区注明标签、在目录树中为它创建加载点，使用 **mount** 命令加载并使用它。还可以把它的信息添加到/etc/fstab 文件中。



parted 尚不支持创建 ext3 文件系统。因此，如果想创建一个 ext3 文件系统，先使用 part 划分分区，然后使用 mkfs 命令来创建，参见本章 2.3.2 节。

2.2.3 删除分区

如果要删除/dev/hdb 上的一个分区，首先输入如下命令启动 parted：

```
# parted /dev/hdb
```

然后用 **print** 命令查看当前的分区表，确认要删除的分区。

使用 **rm** 来删除分区，例如，要删除分区设备名为 hdb3，则在（parted）提示符下键入：**rm 3**。

只要按下<Enter>键，分区就会被删除，请在命令执行前仔细检查一下。

分区删除后，使用 **print** 命令可以看到分区已经被从分区表删除。最后要把它从/etc/fstab 文件中删除，找到与被删除的分区相应的行，从文件中删除它。



不要试图删除正在使用的设备上的分区。

2.2.4 重新划分分区大小

如果要重新划分/dev/hdb 上某一个分区的大小，首先启动 parted：

```
# parted /dev/hdb
```

然后用 **print** 命令查看当前的分区表，查看磁盘上要重划大小的分区的设备名及其起止点。

使用 **resize** 命令重新划分分区大小，然后跟随分区的设备名中的数字和以 MB 为单位的起始点和终止点。格式如下：

(parted) **resize minor-num start-mb end-mb**

分区被重新划分大小后, 使用 **print** 命令来确认分区已被正确地重新划分了大小, 并且具备正确的分区类型和文件系统类型。

重新引导了系统后, 使用 **df** 命令来确定分区已被挂载, 并且它的新大小也已被识别。



不要试图重新划分正在被使用的设备上的分区的大小。



parted 是一个复杂且功能强大的工具, 需要足够地相关知识才可以保证安全地使用。建议您从它的 man page 等帮助文档中学习更多的信息。

2.3 ext3 文件系统

ext3 是日志文件系统, 它被设计为 ext2 的升级版本, ext3 在 ext2 的基础上加入了日志记录功能。Red Flag Asianux Server 3 中默认的文件系统类型就是 ext3。

2.3.1 ext3 的特性

➤ 可用性

当发生异常断电或系统崩溃时, ext2 很容易造成文件系统处于不一致的状态。重新引导时就需要用 e2fsck 程序对每个挂载在系统上的 ext2 文件系统检查其一致性, 这不仅会耽误大量时间, 还可能出现严重的后果。

由于 ext3 文件系统提供了日志记录文件, 一致性检查只在某些罕见的硬件失效情况下才发生, 系统断电或异常崩溃后, ext3 文件系统的恢复时间一般只需要几秒钟到几分钟。

➤ 数据完好性

ext3 文件系统提供更强健的数据完好性, 它允许您选择数据接受的保护类型和级别。

➤ 速度

ext3 的日志记录机制优化了硬盘驱动器的头运动, 所以尽管 ext3 不止把数据写入一次, 它的总处理能力在多数情况下仍比 ext2 系统要高。

➤ 兼容性

向下兼容 ext2, ext2 文件系统可以被很容易地转换为 ext3 系统, 从而获得健壮的日志文件系统的优越性。

2.3.2 创建一个 ext3 文件系统

创建 ext3 文件系统的步骤如下:

- 1) 用 parted 或 fdisk 分区工具创建分区;
- 2) 用 mkfs 将分区格式化为 ext3 文件系统;
- 3) 在系统目录树中创建挂载点;
- 4) 把分区信息添加到/etc/fstab 文件中。



以上步骤的详细解释，敬请参阅本章 2.1.2 节：文件系统的创建、加载与卸载。

2.3.3 转换到 ext3 文件系统

tune2fs 程序能够在不改变已有数据的条件下给现存的 ext2 文件系统添加一个日志记录文件，把 ext2 文件系统转换成 ext3 格式。

要把 ext2 文件系统转换成 ext3，需要用户具有超级用户权限，命令如下：

```
# tune2fs -j /dev/hdb1
```

以上命令中，假设/dev/hdb1 为要转换的文件系统所在设备名。



如果要转换的是根文件系统，还需要创建一个用于引导系统的 initrd 映像文件。使用 mkinitrd 程序创建 initrd 文件。关于 mkinitrd 命令的信息，敬请参考其 man page。

2.3.4 还原到 ext2 文件系统

如果要将 ext3 文件系统还原为 ext2 格式，必须首先卸载分区。输入命令：

```
# umount /dev/hdb1
```

以上命令中，假设/dev/hdb1 为要转换的文件系统所在设备名。

下一步，把文件系统类型改回 ext2。

```
# tune2fs -O ^has_journal /dev/hdb1
```

以超级用户的身份键入以下命令来检查分区错误：

```
# e2fsck -y /dev/hdb1
```

然后，通过键入以下命令来把分区重新挂载为 ext2 文件系统：

```
# mount -t ext2 /dev/hdb1 /mount/point
```

下一步，删除根目录下的 .journal 文件。方法是转换到分区的挂载目录中，然后键入：

```
# rm -f journal
```

以上步骤完成后，文件系统被还原成为 ext2。如果要永久地把该分区改换成 ext2 文件系统，请记住更新/etc/fstab 文件。

2.4 交换空间

交换空间是系统从硬盘中划分的一部分空间，当物理内存（RAM）被充满时，内存中不活跃的页就会被移到交换空间。交换空间的大小一般设为物理 RAM 的 1~2 倍，但不能超过 2048MB。

交换空间可以是一个或多个专用的交换分区（推荐的方式），也可以是一个或多个交换文件，或者是两者的组合。

安装 Red Flag Asianux Server 3 系统时已经创建了一个 swap 分区，即用来支持虚拟内存的交换空间。

Red Flag Asianux Server 3 系统允许用户在安装完成后重新调整交换空间的大小。

2.4.1 使用交换分区

➤ 创建和启用交换分区

Red Flag Asianux Server 3 系统允许包含多个交换分区，每个最大可达 124M。这些 swap 分区可以根据需要随时创建并激活，其步骤如下：

- 1) 用 parted 或 fdisk 划分一个分区，并给予恰当的大小。

- 2) 格式化该分区，并检查坏块：

```
# mkswap -c /dev/hda4
```

用实际的分区设备名代替/dev/hda4，这里没有指明分区大小，系统会自动检测。

- 1) 使用如下命令激活交换分区：

```
# swapon /dev/hda4
```

- 2) 如果要在系统启动时自动激活此交换分区，应该在/etc/fstab 中加入如下一行：

```
/dev/hda4 swap swap defaults 0 0
```

添加交换分区并启用它后，用 cat /proc/swaps 或 free 命令查看交换分区是否被成功启用。

➤ 关闭交换分区

要关闭交换分区，可以执行如下命令：

```
# swapoff /dev/hda4
```

➤ 删除交换分区

- 1) 确认将要删除的交换分区已被关闭；
- 2) 删除/etc/fstab 文件中对应的行；
- 3) 用 parted 或 fdisk 命令删除对应的分区。

2.4.2 使用交换文件

➤ 创建和启用交换文件

交换文件是在临时需要交换空间时的一种补救方法。系统中最多可以有 8 个交换文件，每一个最大为 16M，下面是创建交换文件的步骤。

- 1) 创建一个具有您所希望大小的文件

```
# dd if=/dev/zero of=/swapfile bs=1024 count=8192
```

该命令将物理地创建一个交换文件/swapfile，每一个文件块大小为 1024 字节，一共有 8192 块，文件大小为 8M（因为交换文件必须是连续的，所以是不能用 cp 命令创建交换文件的）。

- 2) 创建交换文件

```
# mkswap /swapfile
```

- 3) 激活该交换文件，激活后的文件被立即启用。

```
# swapon /swapfile
```

- 4) 想要在系统启动时自动激活此交换文件，应该在/etc/fstab 中加入如下一行：

```
/swapfile swap swap defaults 0 0
```

添加交换文件并启用它后，用 cat /proc/swaps 或 free 命令查看交换文件是否被成功启用。

➤ 关闭交换文件

在使用交换文件后，您可以关闭它。

```
# swapoff /swapfile
```

➤ 删除交换文件

```
# rm /swapfile
```

2.5 RAID设备

RAID（Redundant Array of Inexpensive Disks）称为独立磁盘冗余阵列。

RAID 的基本想法就是把多个便宜的小磁盘组合到一起，成为磁盘组，使性能达到或超过一个容量巨大、价格昂贵的磁盘。这样的磁盘组对于计算机来说，就像一个单独的逻辑存储单元或磁盘。

利用如磁盘条带化（disk striping, RAID0）、磁盘镜像（disk mirroring, RAID1）和带奇偶校验的磁盘条带（disk striping with parity, RAID5）等技术，把数据分布到各个磁盘上，以达到冗余性、低延迟、读写的高带宽以及硬盘毁坏后的最大可恢复性。

采用 RAID 的主要优势在于：

- 加快了速度
- 扩充了存储能力
- 可高效恢复磁盘

2.5.1 RAID的种类

通常有两种可以实现 RAID 的方法：硬件 RAID 和软件 RAID。

➤ 硬件RAID

建立在硬件基础之上，与系统和主机无关，管理着 RAID 子系统。对于主机来说，每个 RAID 组只是一个单独的硬盘。例如，硬件 RAID 设备通常关联到一个 SCSI 控制器，RAID 组看起来就是一个 SCSI 驱动器。外置的 RAID 磁盘柜把所有的 RAID 智能化地统一到外置磁盘子系统的控制器中。全部的子系统通过一个普通的 SCSI 控制器连到主机上。对主机而言，如同一个单独的硬盘。

➤ 软件RAID

通过核心磁盘（块设备）代码来实现不同的 RAID 级别，提供了最廉价的解决方案。不仅不再需要昂贵的磁盘控制器卡和热交换底盘，而且软件 RAID 在便宜的 IDE 硬盘上工作效果与在 SCSI 硬盘上一样好。加上现在的高速 CPU，软件 RAID 的性能已超越了硬件 RAID。

2.5.2 RAID的级别

Linux 内核中的 MD 驱动程序是完全独立于硬件的 RAID 解决方案的一个例子。软件 RAID 的性能现在很大程度上依赖于服务器 CPU 的性能和负载。Linux 内核提供的 RAID 级别包括：

➤ RAID级别0

通常称为“带区”。它利用的是带区数据映射技巧的特定性能。即当数据写入磁盘组时，被分成带区，交错写入磁盘组的磁盘中。这带来了高 I/O 性能、低开销，但不提供任何冗余。磁盘组的存储量等于总的磁盘容量之和。

➤ RAID级别1

即“镜像”，与 RAID 其他的各级别相比，这个级别使用的时间较长。1 级通过把同样的数据写到磁盘组的每个磁盘上，将镜像复制到每个磁盘上，来提供数据冗余。1 级在读数据操作时，并行处理 2 个或更多的磁盘，因此数据传输速率高，但其他操作时无法提供高速的 I/O 传输速率。1 级提供了非常好的数据的高可信度，并且改善了读数据操作的性能，但是耗费很大。如果组成磁盘组的各磁盘规格相同，磁盘组的容量只等于一块磁盘的容量。

➤ RAID级别4

采用奇偶校验，但不提供冗余。它的数据分布在各个磁盘上，有一块盘作为奇偶校验盘。它更适用于 I/O 传输，而不是大文件传输。因为提供奇偶校验的磁盘常成为瓶颈，所以在没有相应技术的情况下，如回写高速缓存技术，不常使用 4 级。如果组成磁盘组的各磁盘规格相同，磁盘组容量等于构成磁盘组的磁盘的总容量，减去一块磁盘的容量。

➤ RAID级别5

最常用到的 RAID 类型。通过把奇偶校验分布到磁盘组中的一些或所有磁盘上，消除了 4 级在写数据上的瓶颈。在 4 级中，读的性能超过了写，性能是不对称的。5 级常使用缓冲技术来降低性能的不对称性。如果组成磁盘组的各磁盘规格相同，磁盘组容量等于磁盘的总容量，减去一块磁盘的容量。由于 RAID5 和 RAID4 的容量相同，而性能优异，所以一般都使用 RAID5。

➤ 线性RAID

线性 RAID 是为创建一个大型虚拟驱动设备建立的简单的磁盘分组。在线性 RAID 中，磁盘上的带区是连续分配的，当第一块磁盘分配满后，跳转到下一块磁盘上，依次类推。这个磁盘组未做性能方面的改进，各种 I/O 操作都被划分到各磁盘上是不可靠的。线性 RAID 无冗余，并且可靠性降低。如果磁盘组中的任何一块磁盘出现故障，全部的阵列就都无法使用了。其容量是所有组成磁盘组的磁盘容量之和。

2.5.3 使用RAID

安装 Red Flag Asianux Server 3 时，在“设置分区”步骤中可以方便地为系统划分 RAID 分区，创建 RAID 设备并将其格式化。已创建好并激活的 RAID 设备可以挂载到系统中，就像一个普通硬盘分区一样使用。

对于已创建的 RAID 设备，也可以从系统中卸载、停用和删除。如果在安装时没有创建 RAID 设备，也可以在系统的使用过程中再添加或修改磁盘分区，创建需要的软件 RAID 设备。

与此相关的内容属于系统管理的高级课题，建议由经验丰富的系统管理员执行，具体操作请参阅 [下一节：配置软件 RAID](#)；相关的命令包括：mkraid、mkfs、raidstart、raidstop 等，主要的配置文件为：/etc/raidtab。

2.5.4 配置软件RAID

RAID (Redundant Array of Inexpensive Disks) 称为独立磁盘冗余阵列。RAID 的基本想法是把多个便宜的小磁盘组合到一起，成为一个磁盘组，使其性能达到或超过一个容量巨大、价格昂贵的磁盘。

软件 RAID 使您不必购买昂贵的硬件 RAID 控制器和附件就能极大地增强 Linux 磁盘的 IO 性能和可

靠性。由于 Linux RAID 是用软件实现的，所以它灵活、速度快。使用软件 RAID，可以实现将几个物理磁盘合并成一个更大的虚拟设备，达到性能改进和数据冗余的目的。



《Red Flag Asianux Server 3 安装手册》中介绍了在安装过程中创建软件 RAID 设备的方法和步骤。

2.5.4.1 软件RAID级别

目前用于 Linux 2.4/2.6 内核的软件 RAID 支持以下级别：线性模式，RAID0，RAID1，RAID4 和 RAID5。

➤ 线性模式

将两个或更多的磁盘组合到一个物理设备中，**磁盘不必具有相同的大小**。因为磁盘彼此之间是附加在一起的，所以写入 RAID 设备时将首先填满磁盘 0，然后是磁盘 1，以此类推。

该级别中没有冗余。如果一块磁盘出现故障，那么很可能会丢失所有数据。不过，因为文件系统只是丢失一个大的连续数据组块，所以可以非常幸运地恢复一些数据。

对于单独的读和写，读取和写入性能不会提高。但是如果几个用户同时使用这一设备，并且一个用户实际使用第一块磁盘，而另一个用户正访问刚好位于第二块磁盘上的文件。如果发生这种情况，那么将会带来性能的提高。

➤ RAID 0

也称为分带（stripe）模式。它与线性模式类似，只不过读取和写入是在设备上并行完成的。设备的大小应该大致相等。因为所有访问都是并行完成的，所以设备都是同等填充的。如果一个设备比其他设备大很多，那么在 RAID 设备中仍将使用额外的空间，但在写入 RAID 设备的高端部分时，将只能访问那个更大的磁盘。这当然会降低性能。

与线性模式一样，RAID0 也没有冗余。与线性模式不同的是，如果驱动器出现故障，那么将无法恢复任何数据。如果从 RAID0 中取出一个驱动器，那么 RAID 设备将不仅丢失一个连续的数据块，而是整个设备上都将充满小的空洞。

因为读取和写入是在设备上并行完成的，读取和写入性能将会增加。这通常是运行 RAID0 的主要原因。如果磁盘总线足够快时，可以非常接近 $N \times P \text{ MB/S}$ 。

➤ RAID 1

一个真正具有冗余的模式。RAID1 可以用于两个或多个磁盘，拥有 0 块或多块备用磁盘。这种模式在其他一些磁盘上保留一块磁盘信息的准确镜像。当然，这些磁盘的大小必须相等。如果一块磁盘大于其他磁盘，那么您的 RAID 设备将具有最小磁盘的大小。

如果最多取出了 $N-1$ 块磁盘（或出现故障），那么所有数据仍然保持不变。如果有备用的磁盘，而且系统（例如，SCSI 驱动程序或 IDE 芯片组等）在故障过程中没有被破坏，那么在检测到驱动器故障之后，会立即在一块备用磁盘上开始重建镜像。

RAID1 的写入性能比在一个单独的设备上稍差一些，这是因为写入数据的相同副本必须发送到阵列中的每一块磁盘上。读取性能通常更为糟糕，但在 2.4 内核中已经得到很大改进。

➤ RAID 4

这个 RAID 级别很少使用。它可以用于三块或更多的磁盘上。它在一个驱动器上保存奇偶校验信息，并以这种方式将数据写入 RAID0 中的其他磁盘，而不是完全镜像信息。因为一块磁盘是为奇偶校验信息保留的，所以阵列的大小是 $(N-1) \times S$ ，其中， S 是阵列中最小驱动器的大小。就像在 RAID-1 中那样，磁盘的大小应该相等，否则就必须接受上面的公式。 $(N-1) \times S$ 中的 S 是阵列中最小驱动器的大小。

如果一个驱动器出现故障，可以通过奇偶校验信息来重建所有数据。如果两个驱动器出现故障，那么所有数据都将丢失。

不经常使用这个级别的原因是奇偶校验信息存储在一个驱动器上。每次写入其他磁盘时，都必须更新这些信息。因此，如果奇偶校验磁盘并不比其他磁盘快很多，那么它将成为一个瓶颈。

➤ RAID 5

在希望结合大量物理磁盘并且仍然保留一些冗余时，RAID5 可能是最有用的 RAID 模式。RAID5 可以用在三块或更多的磁盘上，并使用 0 块或更多的备用磁盘。就像 RAID4 一样，得到的 RAID5 设备的大小是 $(N-1) * S$ 。

RAID5 与 RAID4 之间最大的区别就是奇偶校验信息均匀分布在各个驱动器上，这就避免了 RAID 4 中出现的瓶颈问题。如果其中一块磁盘出现故障，但由于使用奇偶校验信息，那么所有数据仍然可保持不变。如果可以使用备用磁盘，那么在设备故障之后，将立即开始重建数据。如果两块磁盘同时出现故障，那么所有数据都会丢失。RAID5 可以经受一块磁盘故障，但不能经受两块或多块磁盘故障。

读取和写入性能通常会提高，但很难预测其提高程度。

➤ 何为备用磁盘

备用磁盘是在一块活动磁盘出现故障之前不参与 RAID 的磁盘。在检测出设备故障时，该设备将被标记为“错误”，并立即开始在第一块备用磁盘上将其重建。

因此，备用磁盘向 RAID5 添加了极好的额外安全措施，而这本来是很难实现的（在物理上）。因为通过备用磁盘实现了所有冗余，所以这允许系统在某个设备出现故障的情况下运行一段时间。

2.5.4.2 RAID设置

2.5.4.2.1 使用mdadm创建RAID

Mdadm 使用的是 md 驱动，由于其拥有多种模式，而且不依赖任何配置文件，是替代 raidtools 的理想工具。其基本语法为：mdadm [mode] [options]

其中[mode] 有 7 种模式：

模 式	描 述
Assemble	将以前定义的某个阵列加入当前在用阵列。
Build	创建一个 legacy array，每个 device 没有 superblocks。
Create	创建一个新的阵列，每个 device 具有 superblocks。
Manage	管理阵列，比如 add 或 remove。
Misc	允许单独对阵列中的某个 device 做操作，比如抹去 superblocks 或 终止在用的阵列。
Follow or Monitor	监控 raid 1, 4, 5, 6 和 multipath 的状态。
Grow	改变 raid 容量或 阵列中的 device 数目。

可用的 [options] 为:

操 作	描 述
-A, --assemble	加入一个以前定义的阵列
-B, --build	创建一个没有 superblock 的 legacy array
-C, --create	创建一个新的阵列
-Q, --query	查看一个 device, 判断它为一个 md device 或是 一个 md 阵列的一部分
-D, --detail	打印一个或多个 md device 的详细信息
-E, --examine	打印 device 上的 md superblock 的内容
-F, --follow, --monitor	选择 Monitor 模式
-G, --grow	改变在用阵列的大小或形态
-h, --help	帮助信息, 用在以上选项后, 则显示该选项信息
-V, --version	查看版本
-v, --verbose	显示细节
-b, --brief	较少的细节。用于 --detail 和 --examine 选项
-f, --force	强行执行操作
-c, --config=	指定配置文件, 缺省为 /etc/mdadm/mdadm.conf
-s, --scan	扫描配置文件或 /proc/mdstat 以搜寻丢失的信息。配置文件 /etc/mdadm/mdadm.conf create 或 build 使用的选项。
-c, --chunk=	修改 kibibytes 的块大小. 缺省为 64.
--rounding=	修改 linear array (==块大小)
-l, --level=	设定 raid 级别
--create 可用	linear, raid0, 0, stripe, raid1, 1, mirror, raid4, 4, raid5, 5, raid6, 6, multipath, mp.
--build 可用	linear, raid0, 0, stripe

操 作	描 述
-p, --parity=	设定 raid5 的奇偶校验规则:left-asymmetric, left-symmetric, right-asymmetric, right-symmetric, la, ra, ls, rs.缺省为 left-symmetric
--layout=	类似于--parity
-n, --raid-devices=	指定阵列中可用 device 数目, 这个数目只能由 --grow 修改
-x, --spare-devices=	指定初始阵列的富余 device 数目
-z, --size=	组建 RAID1/4/5/6 后从每个 device 获取的空间总数
--assume-clean	目前仅用于 --build 选项
-R, --run	阵列中的某一部分出现在其他阵列或文件系统中时, mdadm 会确认该阵列。此选项将不作确认。
-f, --force	通常 mdadm 不允许只用一个 device 创建阵列, 而且创建 raid5 时会使用一个 device 作为 missing drive。此选项正相反。
-a, --auto	{=no, yes, md, mdp, part, p} {NN}

➤ 线性模式

系统中有两个或更多分区, 这些分区的大小不一定相等 (当然也可以相等), 需要将它们彼此连接。执行如下命令:

```
mdadm -C /dev/md0 --level=linear -n2 /dev/sda10 /dev/sda11
```

输出结果如下:

```
mdadm:chunk size defaults to 64K
mdadm:array /dev/md0 started
```

在此不支持备用磁盘。如果一块磁盘出现故障, 那么阵列也会出现故障。这里没有添加备用磁盘的信息。

可以通过检查 `proc/mdstat` 文件或执行命令 `mdadm -D /dev/md0`, 确认 RAID 阵列正在运行。

`cat /proc/mdstat` 输出结果如下:

```
Personalities : [linear] [multipath]
Event : 1
Md0 : active linear sda11[1] sda10[0]
5630528 blocks 64k rounding
```

现在, 可以在设备 `/dev/md0` 上创建一个文件系统了, 并像对待其他任何设备一样装载它了。

➤ RAID 0

系统中有两个或更多设备，它们的大小大致相等，可以通过并行访问它们来结合其存储容量并提高其性能。

执行如下命令：

```
mdadm -C /dev/md0 -l0 -n2 /dev/sda10 /dev/sda11
```

与线性模式类似，这里也不支持备用磁盘。RAID0 没有冗余，因此当一块磁盘出现故障时，阵列也会随之出现故障。

通过 `cat /proc/mdstat` 文件或执行命令 `mdadm -D /dev/md0`，可以查看到设备正在运行。

至此，即可格式化、装载和使用 `/dev/md0` 设备了。

➤ RAID 1

系统中有两个大小大致相同的设备，要使这两个设备彼此镜像，或想在多个磁盘 of 设置一个备用磁盘。如果一个活动设备出现故障，那么它就会自动成为镜像的一部分。

执行如下命令：

```
mdadm -C /dev/md0 -l1 -n2 /dev/sda10 /dev/sda11
```

使用 `mdadm -D /dev/md0` 命令以及 `cat /proc/mdstat` 命令可以查看 RAID 设备的状态：

```
[root@rac1 /]# mdadm --detail /dev/md0
/dev/md0:
Version : 00.90.01
Creation Time : Thu Oct 25 10:56:41 2007
Raid Level : raid1
Array Size : 524096 (511.81 MiB 536.67 MB)
Device Size : 524096 (511.81 MiB 536.67 MB)
Raid Devices : 2
Total Devices : 2
Preferred Minor : 0
Persistence : Superblock is persistent
Update Time : Thu Oct 25 13:08:15 2007
State : clean
Active Devices : 2
Working Devices : 2
Failed Devices : 0
Spare Devices : 0
Number   Major   Minor   RaidDevice State
0        8       17      0         active sync  /dev/sda10
```

```
1      8      33      1      active sync  /dev/sda11
UUID : d58a1c24:4c7e19d4:1f80bde6:cc4f6f8a
Events : 0.16
```

至此，即可格式化、装载和使用/dev/md0 设备了。

➤ RAID 5

系统中有三个或更多大小大致相同的设备，想要将它们结合为一个较大的设备，但为了数据的安全性，仍然需要保留一定程度的冗余。最终您将有许多用作备用磁盘的设备，在其他设备出现故障之前，这些设备不会参与到阵列中。

如使用了 N 个设备，其中最小设备的大小是 S ，那么整个阵列的大小是 $(N-1) * S$ 。丢失的空间用于奇偶校验（冗余）信息。因此，如果任何一块磁盘出现故障，那么所有数据都将保持不变。但如果两块磁盘出现故障，那么所有数据都将丢失。

执行如下命令：

```
mdadm -C /dev/md0 -l5 -n2 /dev/sda10 /dev/sda11 -x1 /dev/sda12
```

如果成功地创建了设备，那么重建过程就已经开始了。在重建完成之前，阵列是不一致的。但阵列是完全可以正常工作的（当然除了处理设备故障之外），您可以格式化它，即使在重建时也可以使用它。

2.5.4.2.2 启动和停止RAID设备

启动 RAID 设备，具体命令如下：

```
mdadm -A /dev/md0
```

在运行 RAID 设备时，可以使用如下命令停止它：

```
mdadm -S /dev/md0
```

2.5.4.2.3 几个有用的概念

➤ 持久的超级块

最早时，raidtools 先要读取/etc/raidtab 文件，然后初始化阵列。但是，这要求装载/etc/raidtab 所在的文件系统。如果想要在 RAID 上启动，那么是很不适宜的。

此外，在 RAID 设备上装载文件系统时，原来的方法会使问题复杂化。它们不能像往常那样放入/etc/fstab 文件，而是必须从初始化脚本装载。

持久的超级块解决了这些问题。如果在/etc/raidtab 文件中使用 persist-superblock 选项，初始化阵列时，一个特殊的超级块被写入参与阵列的所有磁盘的开始位置。这允许内核从所涉及的磁盘上直接读取 RAID 设备的配置，而不是从某些配置文件中读取。

但是，仍然应该维护一个一致的/etc/raidtab 文件，因为可能会在以后重建阵列时需要该文件。

➤ 数据块大小（chunk-size）

需要选择某个合适的块大小，它是指可以写入设备的最小基本数据量。块大小为 4KB 时，写入 16KB 记录将导致第一个和第三个 4KB 的块写入第一块磁盘，第二个和第四个块写入第二块磁盘（使用两块磁盘的 RAID0）。因此对于大型写入来说，使用较大的块大小可以减少开销，而主要包含小文件的阵列使用较小的块大小会更有利。

必须为所有 RAID 级别（包括线性模式）指定块大小。但是，块大小对于线性模式来说没有任何区别。

要想得到最佳性能，应该尝试这个值以及 RAID 阵列上的文件系统的块大小。



/etc/raidtab 中 chunk-size 的参数以 KB 为单位指定组块的大小。因此，4 表示 4KB。

2.5.4.3 RAID不能做什么

RAID 的容错功能设计用于避免由偶发的驱动器故障所产生的负面影响。这种设计非常好。但是，对于各种各样的可靠性问题，RAID 并非总是理想的解决方案。在生产环境中，在实现具有容错功能的 RAID（1、4、5）之前，准确了解 RAID 能做什么及不能做什么至关重要。当处于依赖 RAID 的境况中时，我们不希望对它的作用抱有错误的认识。我们首先要澄清对 RAID 1、4 和 5 的一些常见错误认识。

许多人认为，如果将所有重要数据保存在 RAID 1、4 或 5 卷上，就没有必要再对这些数据执行定期的备份。这是完全错误的，理由如下：RAID 1/4/5 有助于避免由偶然的驱动器故障引起的意外停机；但是，它并不能防止意外或恶意的数据损坏。如果读者在 RAID 卷上以 root 身份键入“cd /; rm -rf *”，那么顷刻之间将丢失大量重要的数据，对于这种情况，就算拥有一个包含 10 个驱动器的 RAID 5 配置也无济于事。同样，如果服务器失窃，或建筑物失火，那么 RAID 也帮不上忙。毫无疑问，如果没有实施备份策略，就不会拥有历史数据的档案文件。假如某位同事删除了一批重要文件，也无法将其恢复。仅此一点就应该足以让您相信，在大多数情况下，即使是在考虑采用 RAID 1、4 和 5 时，也应该规划并实施一种备份策略。

在由劣质硬件组成的系统上实施软件 RAID 是另一种错误认识。如果正在装配一台要承担重要任务的服务器，那么在预算许可的范围之内购买质量最好的硬件是合理的。如果系统不稳定或散热不良，那么将陷入一种 RAID 无能为力的困境。与此类似，如果停电，RAID 显然也不能提供更长的正常运行时间。如果服务器计划担负任何比较重要的任务，请确保已为它配备了不间断电源（UPS）。

文件系统存在于软件 RAID 卷之上，使用软件 RAID 并不能避开文件系统问题。例如，恰好在一种非日志文件系统或定期整理碎片的文件系统，则可能存在耗时且易出问题的文件系统检查。因此，软件 RAID 不会提高 ext2 文件系统的可靠性；这就是为什么在 Linux 阵营中仍然强调保留 ReiserFS、JFS 和 XFS 的原因。软件 RAID 和可靠的日志文件系统是一种理想的组合。

2.6 LVM逻辑卷管理

LVM 是 Logical Volume Manager 的简写，它为计算机提供了更高层次的磁盘存储解决方案，使系统管理员可以更为方便、灵活地分配存储空间。

2.6.1 LVM的优点

LVM 通常用于装备有大量磁盘的系统，但它同样适于仅有一、两块硬盘的小系统。

2.6.1.1 小系统使用LVM的益处

传统的文件系统是基于分区的，一个文件系统对应一个分区。这种方式比较直观，但不易改变：

- 不同的分区相对独立，各分区空间经常利用不平衡，空间不能充分利用；
- 当一个文件系统/分区已满时，无法对其扩充，只能重新分区，或把分区中的数据移到另一个更大的分区中，非常麻烦；
- 要把硬盘上的多个分区合并在一起使用，只能采用重新分区的方式，需要数据的备份与恢复。

当采用 LVM 时，情况会有所不同：

- 硬盘的多个分区由 LVM 统一为卷组管理，可以方便地加入或移走分区以扩大、减小卷组的可用容量，硬盘空间被充分利用；
- 文件系统建立在逻辑卷上，而逻辑卷可在卷组容量范围内根据需要改变大小；
- 文件系统建立在 LVM 上，可跨分区，使用方便。

2.6.1.2 大系统使用LVM的益处

在使用很多硬盘的大系统中，使用 LVM 主要是方便管理、增加系统的扩展性。

在一个有很多不同容量硬盘的大型系统中，为不同用户分配空间是一个技巧性的工作，要在用户需求与实际可用空间中寻求平衡。

用户/用户组的空间建立在 LVM 上，可以随时根据使用情况对各逻辑卷进行调整。当系统空间不足而加入新的硬盘时，不必把用户的数据从原硬盘迁移到新硬盘，而只须把新分区加入卷组并扩充逻辑卷即可。同样，使用 LVM 可以在不停止服务的情况下，把用户数据从旧硬盘转移到新硬盘的空间上。

2.6.2 LVM相关概念和术语

➤ LVM (Logical Volume Manager)

LVM (逻辑卷管理) 是操作系统的一个磁盘管理子系统，是与传统的静态分区完全不同的一种磁盘管理方法。如果下定义的话，就是一系列用于建立和控制逻辑卷区域的操作系统命令、库函数和其它工具的集合。

LVM 把实际的物理磁盘数据映射到一个简单而灵活的虚拟逻辑存储视图上，藉以控制磁盘资源；也就是重新考虑了管理文件系统和卷的方法，在文件系统管理中增加了一个额外的抽象层，可以实现虚拟分区或动态建立一个逻辑卷及更改卷的大小，允许文件系统跨越磁盘等功能。

➤ 物理存储介质 (The Physical Media)

系统的存储设备：硬盘或硬盘上的分区，如：/dev/sda、/dev/hda1 等，是存储系统底层的存储单元。

➤ 物理卷 (PV): Physical Volume

硬盘分区或从逻辑上与硬盘分区具有同样功能的设备 (如 RAID)，是 LVM 的基本存储逻辑块，和基本的物理存储介质 (如分区、磁盘等) 不同的是，其中包含有 LVM 管理参数。

➤ 卷组 (VG): Volume Group

LVM 中的最高抽象层，由一个或多个物理卷组成。可以在卷组上创建一个或多个逻辑卷。

➤ 逻辑卷 (LV): Logical Volume

逻辑卷 (LV) 在卷组上建立，相当于非 LVM 系统中的分区，可以在其上创建文件系统，如/home 或 /var 等。

➤ 物理块 (PE): Physical Extent

每个物理卷被划分为大小相等的称为 PE (Physical Extents) 的基本单元，具有唯一编号的 PE 是可以被 LVM 寻址的最小单元。PE 的大小是可配置的，默认为 4MB。

➤ 逻辑块 (LE): Logical Extent

逻辑卷也被划分为被称为 LE (Logical Extents) 的可被寻址的基本单位。在同一个卷组中，LE 的大

小和 PE 是相同的，并且一一对应。

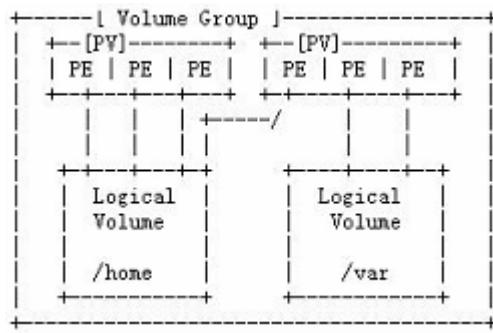
➤ **VGDA（卷组描述符区域）**

和非 LVM 系统将包含分区信息的元数据保存在位于分区起始位置的分区表中一样，逻辑卷以及卷组相关的元数据被保存在位于物理卷起始处的 VGDA 中。VGDA 包括以下内容：PV 描述符、VG 描述符、LV 描述符及一些 PE 描述符。

系统启动 LVM 时激活 VG，并将 VGDA 加载至内存，来识别 LV 的实际物理存储位置。当系统进行 I/O 操作时，就会根据 VGDA 建立的映射机制来访问实际的物理位置。

2.6.3 LVM结构图和工作原理

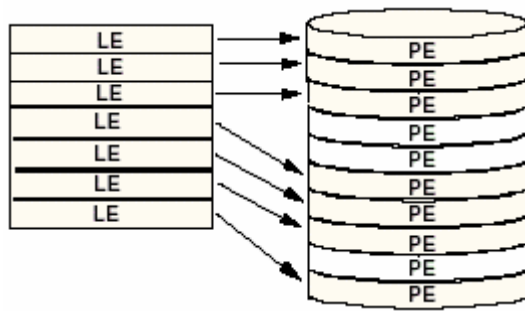
下面是一个 LVM 结构示意图。



LVM 结构图

从图中可以看出：物理卷（PV）由大小相同的基本单元 PE 组成。一个卷组（VG）由一个或多个物理卷组成。PE 和 LE 有着一一对应的关系。逻辑卷建立在卷组上，文件系统建立在逻辑卷上。

那么，LVM 系统如何知道在向某个 LV 中存放数据时，到底存放到哪个（些）实际硬盘呢？在 LVM 系统里，PE 和 LE 之间是一一对应的关系，这种对应关系被存储在 VGDA 中的一个被称为“PE/LE 对应表”的表中。Translation Table 存放在 LVM 磁盘上，当 VG 被激活时才装载到内存中。PE 是在创建卷组时指定的，默认为 4M；在同一个 VG 中的所有 PV 的 PE 大小是相同的，不管实际硬盘的大小和型号是否相同。当 LV 创建时，LVM 系统创建 LE 并自动维护 PE/LE 对应表，使得每一个 LV 里面的 LE 都可以找到与之对应的 PE，从而知道数据该往哪个硬盘上写。



PE/LE 对应图



可以在一个卷组上创建多个逻辑卷，但是一个物理卷只能属于一个卷组。

2.6.4 LVM的一般操作

2.6.4.1 建立PV

为把一个磁盘或分区作为 PV，首先应使用 `pvccreate` 对其初始化，如对 IDE 硬盘 `/dev/hdb`。

➤ 使用整个磁盘

```
# pvccreate /dev/hdb
```

这将在磁盘上建立 VG 的描述符。

➤ 使用磁盘分区，如 `/dev/hdb1`。

使用 `fdisk` 的 `t` 命令把 `/dev/hda1` 的分区类型设为 `0x8e`，然后运行：

```
# pvccreate /dev/hdb1
```

这将在分区 `/dev/hda1` 上建立 VG 的描述符。

PV 初始化命令 `pvccreate` 的一般用法为：

```
pvccreate PV1 [ PV2 ... ]
```

它的参数可以是整个磁盘、分区，也可以是一个 loop 设备。

2.6.4.2 建立VG

在使用 `pvccreate` 建立了 PV 后，可以用 `vgcreate` 建立卷组，如 PV1、PV2 分别是 `/dev/hda1` 与 `/dev/hdb1`，使用如下命令将建立一个名为 `testvg` 的卷组，它由两个 PV：`/dev/hda1` 与 `/dev/hdb1` 组成。

```
# vgcreate testvg /dev/hda1 /dev/hdb1
```

`vgcreate` 的一般用法为：

```
# vgcreate [options] VG_name PV1 [PV2 ...]
```

其中，可选项包括设置 VG 最大支持的 LV 数、PE 大小（缺省为 4MB）等。

注意：当使用 `devfs` 系统时，应使用设备的全名而不能是 *Symbol Link*，如对于上例，应为：

```
# vgcreate testvg /dev/ide/host0/bus0/target0/lun0/part1 \
                /dev/ide/host0/bus0/target1/lun0/part1
```

2.6.4.3 激活VG

在被激活之前，VG 与 LV 是无法访问的，这时可以使用以下命令激活所要使用的卷组。

```
# vgchange -a y testvg
```

当不再使用 VG 时，可用如下命令使之不再可用。

```
# vgchange -a n testvg
```

`vgchange` 可用来设置 VG 的一些参数，如是否可用（`-a [y|n]`选项）、支持最大逻辑卷数等。

2.6.4.4 移除VG

在移除一卷组前应确认卷组中不再有逻辑卷，首先休眠卷组：

```
# vgchange -a n testvg
```

然后可用 `vgremove` 移除该卷组：

```
# vgrremove testvg
```

2.6.4.5 为VG增加新PV

当卷组空间不足时，可以加入新的物理卷来扩大容量，这时可用命令 `vgextend`，如：

```
# vgextend testvg /dev/hdc1
```

其中 `/dev/hdc1` 是新的 PV，当然在这之前，它应使用 `pvcreeate` 初始化。

2.6.4.6 从VG移除PV

在移除 PV 之前，应确认该 PV 没用被 LV 使用，这可用命令 `pvdplay` 查看，如：

```
# pvdplay /dev/hda1
```

```
--- Physical volume ---
```

```
PV Name          /dev/hda1
VG Name          testvg
PV Size          1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#              1
PV Status        available
Allocatable      yes (but full)
Cur LV          1
PE Size (KByte)  4096
Total PE         499
Free PE          0
Allocated PE     499
PV UUID          Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

如这个 PV 仍在被使用，则应把数据传移到其它 PV 上。在确认它未被使用后，可用命令 `vgreduce` 将其从 VG 中删除，如：

```
# vgreduce testvg /dev/hda1
```

2.6.4.7 创建LV

在创建逻辑卷前，应决定 LV 使用哪些 PV，这可用命令 `vgdisplay` 与 `pvdplay` 查看当前卷组与 PV 的使用情况。在已有的卷组上创建逻辑卷使用命令 `lvcreate`，如：

```
# lvcreate -L1500 -n testlv testvg
```

将在卷组 `testvg` 上建立一个 1500MB 的线性 LV，命名为 `testlv`，对应的块设备为 `/dev/testvg/testlv`。

```
# lvcreate -i2 -I4 -l100 -n anothertestlv testvg
```

将在卷组 testvg 上建立名为 anothertestlv 的 LV，其大小为 100LE，采用交错方式存放，交错值为 2，块大小为 4KB。

如果需要 LV 使用整个 VG，可首先用 vgdisplay 查找 Total PE 值，然后在运行 lvcreate 时指定，如：

```
# vgdisplay testvg | grep "Total PE"
```

```
Total PE      10230
```

```
# lvcreate -l 10230 testvg -n mylv
```

将使用卷组 testvg 的全部空间创建逻辑卷 mylv。

在创建逻辑卷后，就可在其上创建文件系统并使用它。

命令 lvcreate 的常用方法：

```
lvcreate [options] -n 逻辑卷名 卷组名 [PV1 ...]
```

其中的常用可选项有：

- i Stripes:** 采用交错 (striped) 方式创建 LV，其中 Stripes 指卷组中 PV 的数量。
- I Stripe_size:** 采用交错方式时采用的块大小 (单位为 KB)，Stripe_size 必须为 2 的指数：2^N，N=2,3...9。
- l LEs:** 指定 LV 的逻辑块数。
- L size:** 指定 LV 的大小，其后可以用 K、M、G 表示 KB、MB、GB。
- s:** 创建一已存在 LV 的 snapshot 卷。
- n name:** 为 LV 指定名称。

2.6.4.8 删除LV

为删除一个逻辑卷，必须首先从系统卸载其上的文件系统，然后可用 lvremove 删除，如：

```
# umount /dev/testvg/testlv
```

```
# lvremove /dev/testvg/testlv
```

```
lvremove -- do you really want to remove "/dev/testvg/testlv"? [y/n]: y
```

```
lvremove -- doing automatic backup of volume group "testvg"
```

```
lvremove -- logical volume "/dev/testvg/testlv" successfully removed
```

2.6.4.9 扩展LV

为逻辑卷增加容量可用使用 lvextend，既可以指定要增加的尺寸，也可以指定扩容后的尺寸，例如，扩大逻辑卷 testlv 的容量为 12GB：

```
# lvextend -L12G /dev/testvg/testlv
```

```
lvextend -- extending logical volume "/dev/testvg/testlv" to 12 GB
```

```
lvextend -- doing automatic backup of volume group "testvg"
```

```
lvextend -- logical volume "/dev/testvg/testlv" successfully extended
```

为 LV testlv 再增大容量 1GB 至 13GB:

```
# lvextend -L+1G /dev/testvg/testlv
```

```
lvextend -- extending logical volume "/dev/testvg/testlv" to 13 GB
```

```
lvextend -- doing automatic backup of volume group "testvg"
```

```
lvextend -- logical volume "/dev/testvg/testlv" successfully extended
```

为 LV 扩容的一个前提是: LV 所在的 VG 有足够的空闲存储空间可用。

在为 LV 扩容之后, 应同时为 LV 之上的文件系统扩容, 使二者相匹配。对不同的文件系统有相对应的扩容方法。

➤ ext2/ext3

除非内核已有 ext2online 补丁, 否则在改变 ext2/ext3 文件系统的大小时应卸载它:

```
# umount /dev/testvg/testlv
```

```
# resize2fs /dev/testvg/testlv
```

```
# mount /dev/testvg/testlv /home
```

这里假设 testlv 安装点为/home。在 es2fsprogs-1.19 或以上版本中包含 resize2fs 命令。

在 LVM 发行包中有一个称为 e2fsadm 的工具, 它同时包含了 lvextend 与 resize2fs 的功能, 如:

```
# e2fsadm -L+1G /dev/testvg/testlv
```

等价于下面两条命令:

```
# lvextend -L+1G /dev/testvg/testlv
```

```
# resize2fs /dev/testvg/testlv
```

但用户仍需首先卸载文件系统。

➤ reiserfs

与 ext2 不同, Reiserfs 不必卸载文件系统, 如:

```
# resize_reiserfs -f /dev/testvg/testlv
```

➤ xfs

SGI XFS 文件系统必须在安装的情况下才可改变大小, 并且要使用安装点而不是块设备, 如:

```
# xfs_growfs /home
```

2.6.4.10 缩小LV

逻辑卷可以扩展同样也可以缩小, 但应在缩小 LV 之前首先减小文件系统, 否则将可能导致数据丢失。

➤ ext2/ext3

可以使用 LVM 的工具 e2fsadm 操作, 如:

```
# umount /home
```

```
# e2fsadm -L-1G /dev/testvg/testlv
```

```
# mount /home
```

如采用 `resize2fs`，就必须知道缩小后卷的块数：

```
# umount /home
```

```
# resize2fs /dev/testvg/testvl 524288
```

```
# lvreduce -L-1G /dev/testvg/testvl
```

```
# mount /home
```

➤ reiserfs

在缩小 `reiserfs` 时，应首先卸载它，如：

```
# umount /home
```

```
# resize_reiserfs -s-1G /dev/testvg/testvl
```

```
# lvreduce -L-1G /dev/testvg/testvl
```

```
# mount -treiserfs /dev/testvg/testvl /home
```

➤ xfs

无法实现。

2.6.4.11 在PV间转移数据

若要把一个 PV 从 VG 中移除，应首先把其上所有活动 PE 中的数据转移到其它 PV 上，而新的 PV 必须是本 VG 的一部分，有足够的空间。如要把 `PV1:/dev/hda1` 上的数据移到 `PV2:/dev/sda1` 上可用命令：

```
# pvmove /dev/hdb1 /dev/sdg1
```

如果在该 PV 之上的 LV 采用交错方式存放，则这个转移过程不能被打断。

建议在转移数据之前备份 LV 中的数据。

2.6.4.12 系统启动和关闭

为使系统启动时可自动激活并使用 LVM，可将以下几行添加到启动 `rc` 脚本中：

```
/sbin/vgscan
```

```
/sbin/vgchange -a y
```

这些行将浏览所有可用的卷组并激活它们。要注意的是，它们应在安装卷组上的文件系统操作之前被执行，否则将无法安装文件系统。

在系统关机时，要关闭 LVM，可以将以下这行添加到关机 `rc` 脚本中，并确保它在卸装了所有文件系统后执行：

```
/sbin/vgchange -a n
```

2.6.5 磁盘分区问题

LVM 允许 PV 建立在几乎所有块设备上，如整个硬盘、硬盘分区、Soft RAID：

```
# pvcreate /dev/sda1
```

```
# pvcreate /dev/sdf
```

```
# pvcreate /dev/hda8
# pvcreate /dev/hda6
# pvcreate /dev/md1
```

所以在一块硬盘上可以有多个 PV 或分区，但一般建议一块硬盘上只有一个 PV：

- 便于管理，易于处理错误。
- 避免交错方式中性能下降。LVM 不能辨别两个 PV 是否在同一硬盘上，所以当采用交错方式时，会导致性能更差。

但在某些情况下可采用：

- 把已存在的系统合并到 LVM 中。在一个只有少数硬盘的系统中，转换为 LVM 时需在各分区之间转移数据。
- 把一个大硬盘分给不同的 VG 使用。

当一个 VG 的有不同的 PV 在同一硬盘时，创建交错方式的 LV 时应注意使用哪一个 PV。

2.6.6 建立LVM用例

在本节中，将在 3 块 SCSI 硬盘：/dev/sda，/dev/sdb，/dev/sdc 上按步骤建立 LVM。

2.6.6.1 准备分区

首先要做的是初始化硬盘，建立 PV。*这将会删除硬盘上的原有数据*。在此，使用整个硬盘为 PV：

```
# pvcreate /dev/sda
# pvcreate /dev/sdb
# pvcreate /dev/sdc
```

pvcreate 在每个硬盘的起始端建立卷组描述区（volume group descriptor area，VGDA）。

2.6.6.2 创建卷组

利用上面三个 PV 建立卷组：

```
# vgcreate test_vg /dev/sda /dev/sdb /dev/sdc/
```

然后可用 vgdisplay 查看和验证卷组的信息：

```
# vgdisplay
--- Volume Group ---
VG Name          test_vg
VG Access        read/write
VG Status        available/resizable
VG #             1
MAX LV           256
Cur LV          0
Open LV          0
```

MAX LV Size	255.99 GB
Max PV	256
Cur PV	3
Act PV	3
VG Size	1.45 GB
PE Size	4 MB
Total PE	372
Alloc PE / Size	0 / 0
Free PE / Size	372/ 1.45 GB
VG UUID	nP2PY5-5TOS-hLx0-FDu0-2a6N-f37x-0BME0Y

其中，最重要的前三条要正确，且 VS size 是以上三个硬盘容量之和。

2.6.6.3 建立LV

在确定卷组 test_vg 正确后，就可在其上创建 LV。LV 的大小可在 VG 大小范围内任意选择，如同在硬盘上分区。

➤ 建立线性方式LV

在 test_vg 上建立一个大小为 1GB 的线性方式 LV：

```
# lvcreate -L1G -ntest_lv test_vg
```

```
lvcreate -- doing automatic backup of "test_vg"
```

```
lvcreate -- logical volume "/dev/test_vg/test_lv" successfully created
```

➤ 建立交错方式LV

在 test_vg 上建立一个大小为 1GB 的交错方式 LV，交错参数为 4KB：

```
# lvcreate -i3 -l4 -L1G -ntest_lv test_vg
```

```
lvcreate -- rounding 1048576 KB to stripe boundary size 1056768 KB / 258 PE
```

```
lvcreate -- doing automatic backup of "test_vg"
```

```
lvcreate -- logical volume "/dev/test_vg/test_lv" successfully created
```



如果使用 `-i2` 参数，则 LV 将仅使用 test_vg 中的两块硬盘。

2.6.6.4 建立文件系统

在 LV test_lv 创建后，就可在其上建立文件系统。

如，ext2/ext3 系统：

```
# mke2fs /dev/test_vg/test_lv
```

如，reiserfs：

```
# mkreiserfs /dev/test_vg/test_lv
```

2.6.6.5 测试文件系统

安装 LV:

```
# mount /dev/test_vg/test_lv /mnt
```

```
# df
```

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda1	1311552	628824	616104	51%	/
/dev/test_vg/test_lv	1040132	20	987276	0%	/mnt

则可以通过/mnt 访问 LV。

2.6.7 使用snapshot做备份

例如我们要对卷组“test_vg”每晚进行数据库备份，就要采用 snapshot 类型的卷组。这种卷组是其它卷组的一个只读拷贝，它含有在创建 snapshot 卷组时原卷组的所有数据，这意味可以备份这个卷组而不用担心在备份过程中数据会改变，也不需要暂时关闭数据库卷以备份。

2.6.7.1 建立snapshot卷

一个 snapshot 卷可大可小，但必须有足够的空间存放所有在本 snapshot 卷生存期间改变的数据，一般最大要求是原卷组的 1.1 倍。如空间不够，snapshot 卷将不能使用。

```
# lvcreate -L592M -s -n dbbackup /dev/test_vg/databases
```

```
lvcreate -- WARNING: the snapshot must be disabled if it gets full
```

```
lvcreate -- INFO: using default snapshot chunk size of 64 KB for "/dev/test_vg/dbbackup"
```

```
lvcreate -- doing automatic backup of "test_vg"
```

```
lvcreate -- logical volume "/dev/test_vg/dbbackup" successfully created
```

2.6.7.2 安装snapshot卷

现在可以安装该卷:

```
# mkdir /mnt/test_vg/dbbackup
```

```
# mount /dev/test_vg/dbbackup /mnt/test_vg/dbbackup
```

```
mount: block device /dev/test_vg/dbbackup is write-protected, mounting read-only
```

从上面可以看出，snapshot 卷是只读的。

当使用 XFS 文件系统时，mount 命令要使用 nouuid 与 norecovery 选项:

```
# mount /dev/test_vg/dbbackup /mnt/test_vg/dbbackup -o nouuid,norecovery,ro
```

2.6.7.3 备份数据

如采用 tar 向磁带备份:

```
# tar -cf /dev/rmt0 /mnt/test_vg/dbbackup
```


2.6.7.4 删除snapshot卷

在完成备份后，就可卸载并删除 snapshot 卷。

```
# umount /mnt/test_vg/dbbackup
# lvremove /dev/test_vg/dbbackup
lvremove -- do you really want to remove "/dev/test_vg/dbbackup"? [y/n]: y
lvremove -- doing automatic backup of volume group "test_vg"
lvremove -- logical volume "/dev/test_vg/dbbackup" successfully removed
```

2.6.8 更换卷组硬盘

由于某种原因，需要用新的硬盘替代卷组中的旧硬盘，如用一个 SCSI 硬盘替换 IDE 硬盘，其步骤如下。

2.6.8.1 准备和初始化新硬盘

首先用 pvcreate 命令初始化新的硬盘，如使用整个硬盘：

```
# pvcreate /dev/sdf
pvcreate -- physical volume "/dev/sdf" successfully created
```

2.6.8.2 加入卷组

把新硬盘加入卷组：

```
# vgextend test_vg /dev/sdf
vgextend -- INFO: maximum logical volume size is 255.99 Gigabyte
vgextend -- doing automatic backup of volume group "test_vg"
vgextend -- volume group "test_vg" successfully extended
```

2.6.8.3 数据搬家

在移除旧硬盘前，要把其上的数据转移到新硬盘上。在转移数据时，不要求卸载文件系统，但建议在数据转移前进行备份，以防转移过程中出现意外导致数据丢失。

pvmove 用来实现数据转移，根据数据量的多少，可能要使用大量的时间，并可能降低逻辑卷的性能，因此要在系统不太忙时执行操作。

```
# pvmove /dev/hdb /dev/sdf
pvmove -- moving physical extents in active volume group "test_vg"
pvmove -- WARNING: moving of active logical volumes may cause data loss!
pvmove -- do you want to continue? [y/n] y
pvmove -- 249 extents of physical volume "/dev/hdb" successfully moved
```

2.6.8.4 移除未用硬盘

当数据被转移到其它硬盘后，就可以从卷组中删除这块不再使用的硬盘了。

```
# vgreduce dev /dev/hdb
```

```

vgreduce -- doing automatic backup of volume group "test_vg"
vgreduce -- volume group "test_vg" successfully reduced by physical volume:
vgreduce -- /dev/hdb

```

从此，卷组 test_vg 不再使用 IDE 硬盘/dev/hdb，这块硬盘可以从机器中拆下或用作其它用途。

2.6.9 迁移卷组到其它系统

把一个卷组转移到其它系统是很容易的（如更换服务器），这要用命令 `vgexport` 与 `vgimport`。

2.6.9.1 卸载文件系统

为整体搬迁卷组，应首先把它从文件系统中卸载，如：

```
# umount /mnt/design/users
```

2.6.9.2 设置卷组为非活动状态

把卷组从内核中卸载，以避免任何对它可能的操作：

```

# vgchange -an test_vg
vgchange -- volume group "test_vg" successfully deactivated

```

2.6.9.3 Export卷组

这个操作不是必须的，但它可以防止系统对卷组的访问：

```

# vgexport test_vg
vgexport -- volume group "test_vg" successfully exported

```

当机器关机后，构成卷组的硬盘就可被转移到新的服务器上。

2.6.9.4 Import卷组

在新的服务器上，可用 `pvscan` 查看卷组情况，如在这台计算机上，新的硬盘设备为/dev/sdb，使用 `pvscan` 可以：

```

# pvscan
pvscan -- reading all physical volumes (this may take a while...)
pvscan -- inactive PV "/dev/sdb1" is in EXPORTED VG "test_vg" [996 MB / 996 MB free]
pvscan -- inactive PV "/dev/sdb2" is in EXPORTED VG "test_vg" [996 MB / 244 MB free]
pvscan -- total: 2 [1.95 GB] / in use: 2 [1.95 GB] / in no VG: 0 [0]

```

现在可以 import 卷组 test_vg（同时也激活它）以安装其上的文件系统。

```

# vgimport test_vg /dev/sdb1 /dev/sdb2
vgimport -- doing automatic backup of volume group "test_vg"
vgimport -- volume group "test_vg" successfully imported and activated

```

2.6.9.5 安装文件系统

```
# mkdir -p /mnt/design/users
```

```
# mount /dev/test_vg/users /mnt/design/users
```

在完成以上操作后，原卷组在新的服务器上就可使用了。

2.6.10 分割卷组

这种情况是：需要在系统中加入新的卷组，但没有其它可用新硬盘，而已有的卷组中还有大量空间可用。如向系统加入一个“design”卷组。

2.6.10.1 检查可用空间

```
# pvscan
```

```
pvscan -- reading all physical volumes (this may take a while...)
pvscan -- ACTIVE   PV "/dev/sda"   of VG "dev"    [1.95 GB / 0 free]
pvscan -- ACTIVE   PV "/dev/sdb"   of VG "sales"  [1.95 GB / 1.27 GB free]
pvscan -- ACTIVE   PV "/dev/sdc"   of VG "ops"    [1.95 GB / 564 MB free]
pvscan -- ACTIVE   PV "/dev/sdd"   of VG "dev"    [1.95 GB / 0 free]
pvscan -- ACTIVE   PV "/dev/sde"   of VG "ops"    [1.95 GB / 1.9 GB free]
pvscan -- ACTIVE   PV "/dev/sdf"   of VG "dev"    [1.95 GB / 1.33 GB free]
pvscan -- ACTIVE   PV "/dev/sdg1"  of VG "ops"    [996 MB / 432 MB free]
pvscan -- ACTIVE   PV "/dev/sdg2"  f VG "dev"    [996 MB / 632 MB free]
pvscan -- total: 8 [13.67 GB] / in use: 8 [13.67 GB] / in no VG: 0 [0]
```

我们决定把/dev/sdg1 与/dev/sdg2 分配组 design，但首先要将其上的物理块移到其它卷的空闲空间中（如把卷组 dev 移到/dev/sdf，卷组 ops 移到/dev/sde）。

2.6.10.2 从选定硬盘移出数据

由于硬盘上的逻辑卷仍在使用的，所以首先要转移它们的数据。

把所有在使用的物理块从/dev/sdg1 上转移到/dev/sde，从/dev/sdg2 转移到/dev/sdf。

```
# pvmove /dev/sdg1 /dev/sde
```

```
pvmove -- moving physical extents in active volume group "ops"
pvmove -- WARNING: moving of active logical volumes may cause data loss!
pvmove -- do you want to continue? [y/n] y
pvmove -- doing automatic backup of volume group "ops"
pvmove -- 141 extents of physical volume "/dev/sdg1" successfully moved
```

```
# pvmove /dev/sdg2 /dev/sdf
```

```
pvmove -- moving physical extents in active volume group "dev"
pvmove -- WARNING: moving of active logical volumes may cause data loss!
pvmove -- do you want to continue? [y/n] y
```

```
pvmove -- doing automatic backup of volume group "dev"
pvmove -- 91 extents of physical volume "/dev/sdg2" successfully moved
```

2.6.10.3 创建新卷组

现在把/dev/sdg2 从卷组 dev 中分割出来，并加入到新卷组 design 中。我们可用 vgreduce 与 vgcreate 完成工作，但 vgsplit 此时更方便：

```
# vgsplit dev design /dev/sdg2
vgsplit -- doing automatic backup of volume group "dev"
vgsplit -- doing automatic backup of volume group "design"
vgsplit -- volume group "dev" successfully split into "dev" and "design"
```

2.6.10.4 移除剩余的卷

接下来的工作是把/dev/sdg1 从卷组 ops 中分出来，并加入到卷组 design：

```
# vgreduce ops /dev/sdg1
vgreduce -- doing automatic backup of volume group "ops"
vgreduce -- volume group "ops" successfully reduced by physical volume:
vgreduce -- /dev/sdg1
```

```
# vgextend design /dev/sdg1
vgextend -- INFO: maximum logical volume size is 255.99 Gigabyte
vgextend -- doing automatic backup of volume group "design"
vgextend -- volume group "design" successfully extended
```

2.6.10.5 建立新逻辑卷及文件系统

在卷组 design 上建立逻辑卷，为今后方便考虑，现只使用一部分空间：

```
# lvcreate -L750M -n users design
lvcreate -- rounding up size to physical extent boundary "752 MB"
lvcreate -- doing automatic backup of "design"
lvcreate -- logical volume "/dev/design/users" successfully created
```

```
# mke2fs /dev/design/users
mke2fs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
96384 inodes, 192512 blocks
```

```

9625 blocks (5.00<!-- ) reserved for the super user
First data block=0
6 block groups
32768 blocks per group, 32768 fragments per group
16064 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840

```

Writing inode tables: done

Writing superblocks and filesystem accounting information: done

```
# mkdir -p /mnt/design/users
```

```
# mount /dev/design/users /mnt/design/users/
```

现在就可以使用卷组 design。为方便使用，可把下面一行加入文件/etc/fstab 中。

```
/dev/design/user /mnt/design/users ext2 defaults 1 2
```

2.6.11 转变根文件系统为LVM



强烈要求在进行下面的操作前对系统进行备份。另外，把文件系统建立在 LVM 上会导致系统升级很复杂。

在下面的例子中，系统除了/boot 外都安装在同一个分区中，文件系统的情况为：

```
/dev/hda1 /boot
```

```
/dev/hda2 swap
```

```
/dev/hda3 /
```

进行转换的一个必要条件是硬盘上还有足够的空间给分区/dev/hda4 创立 LVM 并把分区的内容都复制到 LVM 上，否则：

- 1) /分区还有至少一半空间空闲，可以缩减/分区，并把分出的空间划分到分区/dev/hda4；
- 2) 如果硬盘上已经没有足够的空间，就必须使用第二块硬盘，如/dev/hdb。

在完成以上准备及备份系统后，可继续以下步骤：

- 1) 确认使用的 Linux 内核支持 LVM，并且在编译时设置了 CONFIG_BLK_DEV_RAM 与 CONFIG_BLK_DEV_INITRD。
- 2) 设置/dev/hda4 分区类型为 LVM (8e)

```
# fdisk /dev/hda
```

```
Command (m for help): t
```

```
Partition number (1-4): 4
```

```
Hex code (type L to list codes): 8e
```

Changed system type of partition 4 to 8e (Unknown)

Command (m for help): w

1) 设置 LVM

- 初始化 LVM (vgscan)

```
# vgscan
```

- 转变分区为 PV:

```
# pvcreate /dev/hda4
```

- 建立卷组:

```
# vgcreate vg /dev/hda4
```

- 建立逻辑卷用以存放根系统: (这里假设空间为250MB)

```
# lvcreate -L250M root vg
```

2) 在逻辑卷上建立文件系统并把系统复制到其上:

```
# mke2fs /dev/vg/root
```

```
# mount /dev/vg/root /mnt/
```

```
# find / -xdev | cpio -pvmd /mnt
```

3) 修改新系统的 fstab 文件/mnt/etc/fstab, 使/安装到/dev/vg/root:

```
/dev/hda3 / ext2 defaults 1 1
```

改变为:

```
/dev/vg/root / ext2 defaults 1 1
```

4) 创建 LVM 初始化 RAM 盘。

```
# lvmcreate_initrd
```

此处要确认为 lvmcreate_init 给出正确的 initrd image 文件名, 它应在/boot/目录下。

5) 在/etc/grub.conf 中为 LVM 加入新入口项, 其形式如下:

```
title lvm
    root (hd0,0)
    kernel /boot/KERNEL_IMAGE_NAME root=/dev/vg/root
    initrd /boot/INITRD_IMAGE_NAME
    ramdisk = 8192
```

此处的 KERNEL IMAGE NAME 支持 LVM 的内核, INITRD IMAGE NAME 指由 lvmcreate_initrd 建立的 initrd image。如果 LVM 的配置很多, 可以把 ramdisk 设置得大一些: 此处为 8192, 缺省为 4096。在 lvmcreate_initrd 的输出中有如下行:

```
lvmcreate_initrd -- making loopback file (6189 kB)
```

其中括号中的数值为实际所需大小。

6) 重启计算机, 在 GRUB 提示符处输入 “lvm”, 启动计算机。此时系统的根文件系统是新建立的

逻辑卷。

如果系统未能正常启动，可能的原因是内核不支持 LVM、initrd image 不正确等。

7) 在正常启动后，就可把硬盘其它分区：/dev/hda3 加入 LVM。

➤ 首先设置分区类型为 8e (LVM)

```
# fdisk /dev/hda
Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 8e
Changed system type of partition 3 to 8e (Unknown)
Command (m for help): w
```

➤ 把它初始化为 PV，并加入卷组中。

```
# pvcreate /dev/hda3
# vgextend vg /dev/hda3
```

2.6.12 共享 LVM 卷

在使用 fibre-channel 或 shared-SCSI 的环境中，多台计算机以物理方式直接访问一组硬盘，于是可以使用 LVM 把这些硬盘分为不同的逻辑卷。如果需要共享数据，则应使用 GFS。



LVM 不支持物理共享访问，这会导致数据的丢失。

2.7 配置磁盘限额

为了防止某个用户或用户组占用过多的磁盘存储空间，需要对用户或用户组的可用存储空间进行限制。磁盘限额的意义是强制使用者在大部分时间内保持他们对系统磁盘的占用在限额之下，取消其无限制地使用磁盘空间的能力。

在 Linux 系统中，限额是对文件系统设定的，设定之前需要启动文件系统的配额设置支持。磁盘限额服务目前可以支持 ext2 和 ext3 两种文件格式。

Red Flag Asianux Server 3 系统中提供了一个图形化的磁盘限额管理工具——rfquota。本节先讲述如何使用命令行管理磁盘限额，再介绍 rfquota 的使用。

2.7.1 配置文件系统的磁盘限额支持

首先，请确定系统中的哪块分区要进行磁盘限额，在/etc/fstab 文件中为其对应的文件系统中添加 usrquota 及 grpquota 选项。

假如要为/home 分区同时设置用户级和用户组级的磁盘限额，在/etc/fstab 文件中找到/home 对应的一行：

```
/dev/hda5 /home ext2 defaults 1 2
```

然后对该行做如下修改：

```
/dev/hda5 /home ext2 defaults,usrquota,grpquota 1 2
```

对每个需要设置磁盘配额的分区分进行下面的操作。添加了 `userquota` 和 `grpquota` 选项后，重新挂载每个被修改的文件系统。如果文件系统没有被任何进程使用，使用 `umount` 命令后再用 `mount` 命令重新挂载这个文件系统；否则可以重新启动系统。

启用配额的文件系统被重新挂载后，先要运行 `quotacheck` 命令在文件系统上创建磁盘配额文件。

```
# quotacheck -cug /home
```

`-c` 选项为启用了配额的文件系统创建配额文件（`aquota.user` 和 `aquota.group`），`-u` 选项代表检查用户配额，`-g` 选项代表检查组配额。系统在运行配额检查时，会在配额文件中创建磁盘使用信息。

接下来，运行以下命令来生成启用了配额的文件系统的当前磁盘用量表。

```
# quotacheck -vug
```

`-v` 选项表示在检查配额过程中显示详细的状态信息。

2.7.2 设置用户限额

对磁盘配额的限制一般是从一个用户占用磁盘大小和所有文件的数量两个方面来进行的。在具体操作之前，先了解一下磁盘配额的两个基本概念：软限制和硬限制。

➤ **软限制**：用户/用户组在文件系统可拥有的最大磁盘空间和最多文件数量，在某个宽限期——称为过渡期（`grace period`）内可以暂时超过这个限制。

➤ **硬限制**：用户/用户组可拥有的磁盘空间或文件的绝对数量，绝对不允许超过这个限制。

要为用户配置配额，以超级用户身份执行以下命令：（此处设置对象为 `tester` 用户）

```
# edquota tester
```

这个命令将启动默认文本编辑器 `vi`，其内容如下所示：

```
Disk quotas for user tester (uid 503):
Filesystem      blocks    soft    hard    inodes    soft    hard
/dev/hda7        52         0      0       46         0      0
```

表示 `tester` 用户在 `/dev/hda7` 分区中迄今使用了 52 个数据块，没有设置磁盘空间大小限制，同样也没有限制文件节点数量。

对用户进行磁盘容量的限制时，只需要修改 `blocks` 列后面的 `soft` 和 `hard` 列的数值即可，单位是 `KB`。请根据需要为用户设置空间限额，下面是修改后的文件。

```
Disk quotas for user tester (uid 503):
Filesystem      blocks    soft    hard    inodes    soft    hard
/dev/hda7        52    50000  55000     46         0      0
```

同样，如果对文件/目录的数量限制可以相应地修改 `inodes` 列后面的 `soft` 和 `hard` 列的数值。可以同时这两项都做出限制。

要查看或验证用户的配额设置，使用以下命令：

```
# quota tester
```



记住保存修改以使设置生效。磁盘限额生效后，该用户的磁盘使用就不能超过硬限制。如果用户试图超过这个限制，操作将被取消，然后得到一个错误信息。

edquota 命令的 **-p** 参数可以对已有的用户设置进行拷贝。如果我们要对某些用户进行和 **tester** 一样的限额配置，请通过以下命令完成：

```
# edquota -p tester -u xiaoming chengding huahua tutu
```

这样，这些用户即被赋予了和 **tester** 一样的磁盘配额。

2.7.3 设置用户组限额

可以为用户组分配磁盘限额。例如，要为系统中的 **test** 组设置用户组配额，使用以下命令：

```
# edquota -g test
```

与设置用户限额相似，在 **vi** 编辑器中修改限额值后保存文件即可。

要查看或验证用户组的配额设置，请使用以下命令：

```
# quota test
```

2.7.4 设置软限制过渡期

使用 **edquota** 命令的 **-t** 选项可以修改软限制的过渡期，过渡期可以用秒、分钟、小时、天、周、或月表示。运行下面的命令：

```
# edquota -t
```

命令将打开缺省的编辑器 **vi**，显示如下内容：

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem      Block grace period   Inode grace period
/dev/hda7        7days                7days
```

系统默认的过渡期是 7 天，可以使用天、小时、分、秒为单位来修改这一期限。

2.7.5 管理磁盘配额

➤ 报告磁盘配额

使用 **repquota** 工具创建磁盘用量报告。例如，**repquota /home** 命令的输出如下：

```
[root@localhost xyzhou]# repquota /home
*** Report for user quotas on device /dev/hda7
Block grace time: 7days; Inode grace time: 7days
```

User		used	Block limits			grace	File limits		
			soft	hard			used	soft	hard
root	--	234664	0	0		3503	0	0	
tester	--	52	50000	55000		46	0	0	
xyzhou	+-	55000	50000	55000	6days	93	0	0	
zys	--	52	0	0		46	0	0	
hp	--	52	100	200		46	0	0	
#501	--	52	0	0		46	0	0	
#502	--	52	100	200		46	0	0	

每一行用户名后面的“--”用于判断该用户是否超出其块限制或文件节点限制。如果任何一个软限被超出，相应的“-”列就会被“+”代替，第一个“-”代表块限制，第二个代表文件节点限制。

grace 列通常是空白。如果某个软限额被超出，这一列会显示过渡期中的剩余时间。如果过渡期已超过，将显示 **none**。

要查看所有启用了磁盘配额的文件系统的磁盘用量，使用以下命令：

```
# repquota -a
```

➤ **保持配额的正确性**

定期运行 `quotacheck` 命令以保持磁盘配额的正确性：**`quotacheck -vug`**。

系统管理员可以把 `quotacheck` 脚本添加到系统的任务计划中定期执行，参考本手册第四章相关内容以获得设置任务计划的详情。

➤ **启用和禁用**

要启用配额，使用 `quotaon` 命令，如下命令将为所有已设置磁盘配额的文件系统启用用户和用户组限额：

```
# quotaon -vug
```

要为指定的文件系统（如/home）启用配额，在上面的命令后加上/home 即可。

要关闭用户和用户组限额，使用以下命令：

```
# quotaoff -vug
```

第3章 高级文件系统指南

Red Flag Asianux Server 3 支持多种最新的日志文件系统，包括 XFS、REISERFS、EXT3 等。本章将向您介绍这些日志文件系统的特性及简单的使用方法，以便您尽可能轻松、愉快地使用最新的文件系统技术。

在此之前，先介绍一些必要的基本知识，以帮助更好地理解。

3.1 日志系统（Journaling）

日志是一项非常重要的技术，在 ReiserFS、XFS、ext3 等文件系统中都会用到它，用以达到快速检查文件系统一致性的目的。

3.1.1 元数据（Meta-data）

文件系统的存在允许用户储存、检索和操作数据，为此文件系统需要保持一个内在的数据结构使得数据有组织并且便于访问。这一内部的数据结构（确切地说就是“关于数据的数据”）被称为元数据，它为文件系统提供了其特定的身份和性能特征。

通常，我们并不直接和文件系统的元数据打交道。而是一个特别的 Linux 文件系统驱动程序为我们完成相应的工作。Linux 文件系统驱动程序是专门用来操作复杂的元数据的。然而，为了使得文件系统驱动程序正常工作，有一个很重要的必要条件，它需要在某种合理的、一致的和没有干扰的状态下找到元数据。否则，文件系统驱动程序将不能理解和操作元数据，即不能存取文件。

3.1.2 fsck

当 linux 系统关闭时，内核驱动会把所有的缓冲区数据转送到磁盘，并确保文件系统被彻底卸载，这样文件系统的元数据将会处于可用的状态，并可以在下次启动时被正常装载及使用。但当一些意外发生时，文件系统没有被彻底卸载，元数据可能是错误的，这时就需要用 fsck 对文件系统进行全面地检查，修正找到的任何错误，使元数据恢复一致。

fsck 的工作就是确保要装载的文件系统的元数据处于可使用的状态。典型方式是，fsck 扫描那些将被装载的文件系统，确定它们已被彻底卸载，并做出合理的假设——所有的元数据都没有问题；当 fsck 检测到没有被彻底卸载的文件系统时，就会彻底的扫描并且全面地检查该文件系统的元数据，修正这一过程中找到的任何错误；一旦 fsck 完成工作，文件系统就可以使用了。

尽管意想不到的电源故障或系统挂起可能造成最近修改的数据丢失，但是由于元数据现在是一致的，文件系统可以被装载和使用。

使用 fsck 可以确保文件系统的一致性，但却不是最佳的解决方案。使用 fsck 所面临的问题是：fsck 必须扫描文件系统的全部元数据，才能确保文件系统的一致性。

对文件系统所有的元数据做彻底的一致性检查是一项极为费时的的工作，文件系统越大，完成扫描所花费的时间就越长。当 fsck 运行时，系统实际上是被中断了而不能工作，如果有一个庞大的文件系统，可能会花上半个小时或更长时间来执行 fsck，这通常是难以接受的。

3.1.3 日志（Journal）

日志文件系统通过增加一个叫做日志的新数据结构来解决 fsck 问题。该日志位于磁盘上的结构。在对元数据做任何改变之前，文件系统驱动程序会向日志中写入一个条目，该条目描述了它将要做什么，然后继续并修改元数据。通过这种方法，日志文件系统就拥有了近期元数据被修改的历史记录，当检查到没

有彻底卸载的文件系统的一致性问题时，这个记录就大有用处了。

可以这样来看待日志文件系统——除了存储数据和元数据以外，它们还有一个日志——可以称之为元元数据。

fsck 如何处理日志文件系统呢？实际上，通常它什么都不做，只是忽略文件系统并允许它被装载。在快速地恢复文件系统到达一致性状态的背后，真正起作用的在于 Linux 文件系统驱动程序中。

当文件系统被装载时，Linux 文件系统驱动程序查看文件系统是否完好。如果由于某些原因出了问题，那么就需要对元数据进行修复，但不是执行对元数据的彻底扫描（就像 fsck 那样），而是查看日志。由于日志中包含了按时间顺序排列的近期的元数据修改记录，它就简单地查看最近被修改的那部分元数据。因而，它能够在几秒钟时间内将文件系统恢复到一致性状态。并且与 fsck 所采用的传统方法不同，这个日志重放过程在大型的文件系统上并不需要花更多的时间。有了日志，数百 G 的文件系统元数据几乎能在瞬间恢复到一致性的状态。

3.1.4 不同的日志文件系统

哪种 Linux 日志记录文件系统是“最好的”？没有一个对每个应用程序都“合适的”文件系统。每个文件系统都有自身的长处。所以，理解每种文件系统的长处和弱点，以便对使用哪种文件系统做出一个有根据的选择，远远优于选出一个绝对的“最好的”文件系统，并将它用于所有可能的应用程序。

3.2 Ext3

ext3 是一个非常可靠、健壮的高质量日志文件系统。与 resierFS、XFS 相比，虽然 ext3 的可伸缩性具有局限性，但已被证明在大多数服务器和 workstation 所执行的典型文件系统操作中，使用 ext3 不仅速度很快而且很容易调整。

Ext3 是一个非常全面的文件系统。ext3 很像 ext2，不会提供 ReiserFS 那样特别快的小文件性能，但也不会带来意外的性能或功能瓶颈。因为 ext3 基于 ext2 的代码，所以它的磁盘格式和 ext2 的相同。这就意味着，一个干净卸装的 ext3 文件系统可以作为 ext2 文件系统毫无问题地重新挂装。

由于都使用相同的元数据，可以执行 ext2 到 ext3 文件系统的现场升级。通过升级一些关键系统实用程序、安装新的 2.4 内核，并在每个文件系统中输入单条 tune2fs 命令，就可以把现有的 ext2 服务器转换成日志记录 ext3 系统，甚至可以在 ext2 文件系统已安装的情况下进行这些操作。这种转换是安全的、可逆的与简单的，并且不会导致任何意外的性能急剧下降。与转换到 XFS 或 ReiserFS 不同，这种转换不需数据备份和从头创建文件系统。

除了与 ext2 兼容之外，ext3 还通过共享 ext2 的元数据格式继承了 ext2 的其它优点。譬如，ext3 用户可以使用一个稳固的 fsck 工具。相反，ReiserFS 的 fsck 还很幼稚，当脆弱的元数据真的出现时，对脆弱元数据的修复过程将是困难和危险的。

3.2.1 日志的记录方式

Ext3 处理日志记录的方式与 ReiserFS 和其它日志记录文件系统所用的方式迥异。

3.2.1.1 仅记录元数据日志

典型的日志记录文件系统（譬如 ReiserFS、XFS）对元数据有特别处理，但对数据不够重视。使用 ReiserFS 和 XFS 时，文件系统驱动程序记录元数据，但不提供数据日志记录。使用仅元数据日志记录，文件系统元数据将会异常稳固，因而可能永远不需要执行彻底 fsck。然而，意外的重新引导和系统锁定可能导致最近修改数据的明显毁坏。

举例来说，假设正在修改名为/tmp/myfile.txt 的文件时，机器意外锁定，被迫需要重新引导。如果使

用的是仅元数据日志记录文件系统，文件系统元数据将容易地修复，但是存在一种明显的可能性：在将/tmp/myfile.txt 文件装入到文本编辑器时，文件不仅仅丢失最近的更改，而且还包含许多乱码甚至可能是完全不可读的信息。

3.2.1.2 ext3 的日志记录方法

在 ext3 中，日志记录代码使用一个特殊的称为“日志记录块设备”层或 JBD 的 API。JBD 被设计成在任何块设备上实现日志的特殊目的。Ext3 通过“钩入 (hooking_in)” JBD API 来实现其日志记录。例如，ext3 文件系统代码将正在执行的修改告知 JBD，并且还会在修改磁盘上包含的特定数据之前请求 JBD 的许可。通过执行这些操作，给予了 JBD 代表 ext3 文件系统驱动程序管理日志的适当机会。这是很好的安排，因为 JBD 是作为一个单独的、一般实体而开发的，将来它可以用于向其它文件系统添加日志记录能力。

关于 JBD 管理的 ext3 日志有一些巧妙的特性。其中，ext3 的日志存储在一个索引节点中——基本上是个文件。能否看到这个位于/.journal 的文件，取决于如何在文件系统上“启用 ext3”的。当然，通过将日志存储在索引节点中，ext3 可以向文件系统添加必要的日志，而无需对 ext2 元数据进行不兼容扩展。这是 ext3 文件系统保持对 ext2 元数据，及 ext2 文件系统驱动程序的向后兼容性的关键方式之一。

3.2.2 JBD 的日志记录方法

有许多方法用于实现日志。例如，文件系统开发者可能会设计出一种日志，该日志存储在主机文件系统上需要修改的字节范围。这种方法的好处在于，日志能够以一种非常高效的方式存储许多对文件系统的微小修改，这是因为它只记录需要修改的个别字节，而不记录除此以外的任何信息。

JBD 使用另外一种（从某种意义上说是更好的）方法。JBD 存储完整的被修改的文件系统块本身，而不是记录必定会被更改的字节范围。ext3 文件系统驱动程序也使用这种方法，存储内存中被修改的块（大小为 1K、2K 或 4K）的完整副本，以跟踪暂挂的 IO 操作。开始，这样做看起来有点浪费；毕竟，包含已修改数据的完整块中还可能包含未修改的（已经在磁盘上）数据。

JBD 所使用的方法称为物理日志记录，这就意味着 JBD 使用完整的物理块，作为实现日志的主要媒介。相反，只存储已修改的字节范围而非完整块的方法称为逻辑日志记录，这是 XFS 所使用的方法。因为 ext3 使用物理日志记录，所以 ext3 日志将具有比其它文件系统日志（例如，XFS 日志）更大的相对磁盘占用。但是，因为 ext3 在文件系统内部和日志中使用完整块，ext3 处理的复杂度比实现逻辑日志记录的要小。另外，完整块的使用允许 ext3 执行一些额外的优化。譬如，将多个暂挂的 IO 操作“压扁”到同一内存数据结构的单个块中。接下来，这种优化允许 ext3 将这多个更改在一次写操作中写到磁盘上，而不需要多次写操作。此外，因为文字块数据存储在内存中，这些内存数据在写到磁盘之前，不必或只需作很少更改，这大大地减少了 CPU 开销。

3.2.3 数据保护

ext3 有两种确保数据和元数据完整性的方法。

最初，ext3 被设计用来执行完整数据和元数据日志记录。在这种方式下（称之为“data=journal”方式），JBD 将所有对数据和元数据的更改都记录到文件系统中。因为数据和元数据都被记录，JBD 可以使用日志将元数据和数据恢复到一致状态。完整数据日志记录的缺点是可能会比较慢，但可以通过设置相对较大日志来减少性能损失。

最近，ext3 添加了一种新的日志记录方式，该方式提供了完整日志记录的好处而不会带来严重的性能损失，它只对元数据进行日志记录。但是，ext3 文件系统驱动程序保持对与每个元数据更新对应的特殊数据块的跟踪，将它们分组到一个称为事务的实体中。当事务应用于适当的文件系统时，数据块首先被写到磁盘。一旦写入数据块，元数据将随后写入日志。通过使用这种技术（称为“data=ordered”），即使只有

元数据更改被记录到日志中，ext3 也能提供数据和元数据的一致性。ext3 缺省使用这种方式。

3.2.4 ext3 工具

由于 ext3 向下兼容 ext2，可以使用 e2fsprogs 中的工具，包括 mke2fs、e2fsck、resize2fs、tune2fs 等。在较新的 e2fsprogs 中，增加了 mke3fs、e3fsck 两个工具，创建 ext3 文件系统可以直接使用 mke3fs 或 mkfs -t ext3，也可以先把文件系统创建为 ext2 再升级为 ext3。



若 ext2 文件系统已安装在系统中，则升级后该分区的根目录中会出现 *journal* 文件。



相关概念请参阅 [2.3 节 ext3 文件系统](#)。

第4章 软件包管理

通常 Linux 下的应用软件包有以下三种类型：

➤ **tar包**

由 Unix 系统的打包工具 tar 制作，如 example-1.2.3-1.tar.gz。

➤ **rpm包**

RedHat 公司提供的一种软件包封装格式，如 example-1.2.3-1.i386.rpm。

➤ **dpkg包**

Debian Linux 提供的一种包封装格式，如 example-1.2.3-1.i386.deb。

通常用 tar 打包的都是源程序，用 rpm、dpkg 打包的则是可执行程序。一般一个软件会提供多种打包格式的安裝程序，用户可以根据情况选择使用。自己编译安装源程序具有更大的灵活性，但初级用户可能会遇到一些困难；而可执行程序包能够更容易地完成安装。

本章先简单介绍如何使用 shell 命令安装和管理系统中的应用程序和软件包，之后详细说明如何使用图形化的软件包管理工具在桌面环境下安装和管理 rpm 软件包。

4.1 使用rpm命令

rpm 是一个功能十分强大的软件包管理系统，它使得 Linux 下安装、升级和删除软件包的工作变得简单容易，并且具有查询、验证软件包的功能。与图形化工具相比，使用命令行可以获得更大的灵活性。

以下的例子都以 example-1.2.3-1.i386.rpm 代表对象软件包的名称。

4.1.1 安装、升级和更新

应用下面三个参数为系统安装软件包：

rpm -i 安装一个新的软件包

rpm -U 升级一个软件包，如果系统中原来不存在，就进行安装

rpm -F 更新一个软件包，如果系统中原来不存在，就不进行安装

经常和这几个参数配合使用的参数包括：

-v 查看安装过程中的各种信息

-h 在安装过程中显示进度条

一个常用的命令形式如下：

rpm -Uvh example-1.2.3-1.i386.rpm

这个命令将升级或安装软件包，同时显示安装信息和进度条。

4.1.2 删除

删除一个软件包的命令是：

rpm -e example



删除时使用的是软件名，而不是软件包的全称。

4.1.3 查询

➤ 列出用户已经安装的RPM包清单

如果想查询系统中所有已经安装的 RPM 包，使用 **rpm -qa** 即可输出所有已安装 RPM 包的列表；

如果是查看某个已经安装的软件包，则使用 **rpm -q example** 命令。

➤ 查看一个RPM包中包括的文件

想要查看某个软件包中包含的文件清单，有下面两种方法：

a) 如果是未安装的软件包，则使用

```
rpm -qlp example-1.2.3-1.i386.rpm
```

b) 如果是已安装的软件包，请使用

```
rpm -ql example
```

➤ 确定某个文件属于哪个RPM包

如果遇到某个不认识的文件，要找出它属于哪个软件包，则首先记录这个文件的完整路径(绝对路径)，然后输入以下命令：

```
rpm -qf filename
```

➤ 查询RPM包的用途

用户可以在安装或使用查询每个软件包的用途、版本及其它信息，使用如下的命令完成查询：

```
rpm -qip example-1.2.3-1.i386.rpm
```

4.1.4 验证

验证一个软件包，就是比较原始包和已安装软件包中文件的信息。具体来说，这些信息包括每个文件的大小、MD5 校验和、访问许可权、类型以及所属的用户和组等。

使用命令 **rpm -V** 可以验证一个包，下面是常用的几种情况：

➤ 验证包含某个特殊文件的软件包

```
rpm -Vf filename
```

➤ 验证所有已安装的软件包

```
rpm -Va
```

上面介绍了几个常用RPM命令，关于RPM工具的更多资源，请参看相关的man手册页；还可以在以下的网址<http://www.rpm.org>获得RPM的最新资源。

4.2 安装tar格式的软件包

*.tar.gz 形式的二进制软件包是用 **tar** 工具来打包、用 **gzip** 程序压缩的，安装时需要先解开压缩包，其安装过程分为如下几个步骤：

- 1) 获得应用软件：可以通过网络下载、光盘或其它渠道得来；
- 2) 解压缩文件；
一般的 tar 包，都会再做一次压缩，常见的是 gzip 压缩，用 “**tar -xvzf *.tar.gz**”，就可以完成解压和解包工作；
- 3) 阅读附带的 INSTALL 和 README 文件；
- 4) 执行 “**./configure**” 命令为编译做好准备；
- 5) 通过后，将生成用于编译的 makefile 文件，运行 “**make**” 命令开始进行编译；编译的过程视软件的规模和计算机性能的不同，所耗费的时间也不同；
- 6) 执行 “**make install**” 命令完成安装；
- 7) 执行 “**make clean**” 命令删除安装时产生的临时文件。

怎样运行安装后的应用软件呢？一般来说，Linux 下的应用软件可执行文件存放在 /usr/local/bin 目录下，但这也不是绝对的，最好的方法是查看该软件所附的 INSTALL 和 README 文件，其中会有明确说明。



与安装 RPM 软件包相比，用户自己编译安装源程序虽然具有灵活的可配置性，但编译过程中可能会遇到很多问题，它适合于有一定开发经验的用户，一般不推荐初学者使用。

第5章 使用Vim编辑器

Vim 自产生以来，历经不断革新，现在最新版的 Vim 已经具有很强大的功能，使用户能够更加轻松、便捷地使用它。

5.1 Vim的工作模式

5.1.1 命令模式

开始进入 Vim 时处于命令模式，如果已经处于插入模式或末行模式，按<ESC>键可回到命令模式。在这种模式下，只能用按键指令，不能输入文字。

5.1.2 插入模式

插入模式就是要把文本插入到要编辑的文件，插入位置根据所用的命令不同而不同。从命令模式进入插入模式需要键入 i、a、o、r 及 I、A、O、R 等命令。在完成文本的输入后，必须用<ESC>键返回命令模式。

5.1.3 末行模式

末行模式因命令出现在屏幕的最底部一行而得名。在命令方式下，键入某些特殊字符，如/、?、: 等，光标跳到屏幕末行并显示键入的末行字符，此时键入命令后回车，Vim 会根据需要在末行显示出一定的响应信息，同时将自动回到命令状态。

5.2 Vim编辑文件的基本过程

在命令行键入 **Vim testfile**，其中 testfile 代表要打开的文件名，如果文件不存在，Vim 将自动新建一个名为 testfile 文件。

进入 Vim 后，按<i>键进入插入模式，即可进行文件的编写工作。光标可以由方向键来移动。<BackSpace>键可以删去前一个字符。

写好文件后按<ESC>键可回到命令模式，然后用:w 存档（注意，是冒号命令），这时还不会离开 Vim，要离开可按:q，也可以合起来用:wq，代表保存后离开。

5.2.1 光标的移动

注意：本节所述都是在命令模式下的操作。

5.2.1.1 基本的光标移动

左	h	Backspace 或左方向键
下	j	Enter 或+或下方向键
上	k	-或上方向键
右	l	Space 或右方向键

向下翻页 Ctrl+f PageDown

向上翻页 Ctrl+b PageUp

5.2.1.2 复杂光标移动

- 0 移至行首，或是<Home>键
- ~ 移至第一个非空白字符
- \$ 移至行尾，或<End>键
- G 移至文件尾（最后一行的第一个非空白字符处）
- gg 移至文件首（第一行第一个非空白字符处）
- w 移至下一个字首
- W 同上，但会忽略一些标点符号
- e 移至前一个字字尾
- E 同上，但会忽略一些标点符号
- b 移至前一个字字首
- B 同上，但会忽略一些标点符号
- H 移至屏幕顶部第一个非空白字符
- M 移至屏幕中间第一个非空白字符
- L 移至屏幕底第一个非空白字符
- n| 移至第 n 个字符处
- :n 或 n G 移至第 n 行行首，**注：n 表示具体数字，如 1, 2, 3……**
-) 移至下一个句首
- (移至上一个句首
- } 移至下一个段落首
- { 移至上一个段落首

5.2.2 基本编辑指令

5.2.2.1 进入插入模式指令

- i 在光标所在字符前开始输入文字（insert）
- a 在光标所在字符后开始输入文字（append）
- o 在光标所在行下开一新行来输入文字（open）
- I 在行首开始输入文字
- A 在行尾开始输入文字
- O 在光标所在行上开一新行来输入文字
- J 将下一行整行连接到本行（joint）

5.2.2.2 删除指令

- x 删除光标所在处的字符。也可用键。
- X 删除光标所在位置前的字符。
- dd 删除一整行。
- dw 删除一个字（delete word）。
- dG 删至文件尾。
- D 删至行尾，或 d\$（含光标所在处字符）。

5.2.2.3 取代及还原

- r 取代光标所在处的字符。
- R 取代字符直至按<Esc>为止。
- cc 取代整行内容。或大写 S 亦可。
- cw 替换一个英文字。
- ~ 光标所在处之大小写转换。
- C 取代至行尾，即光标所在处以后的字都会被替换。或 c\$。
- c0 取代至行首，或 c~。

u 撤销前面的操作，即 undo，撤销的次数是没有限制的。

U 在光标没离开本行之前，回复所有编辑动作。

5.2.2.4 复制

Yy 复制光标所在行整行。或一个大写 Y。

2yy 或 y2y 复制两行。

y~ 复制至行首，或 y0，不含光标所在处字符。

y\$ 复制至行尾。含光标所在处字符。

Yw 复制一个字。

Yg 复制至文件尾。

y1G 复制至文件首。

5.2.2.5 查找与替换

➤ 查找

/ 在命令模式的情形下，按/会在左下角出现一个/，键入要查找的字串，按回车开始查找。

? 和/相同，只是/是向前（下）找，?则是向后（上）找。

n 继续查找。

N 继续寻找（反向）。

* 寻找光标所在处的字（要完全符合）。

同上，但*是向前（下）找，#则是向后（上）找。

g* 同*，但部分符合即可。

g# 同#，但部分符合即可。

➤ 替换

:[range]s/pattern/string/[c,e,g,i]

用 string 替代 pattern。Range 指的是范围，1,7 指从第一行至第七行，1,\$指从第一行至最后一行，也就是整篇文章，也可以%代表；c 每次替换前会询问；e 不显示 error；g 不询问，整行替换；i 不分大小写。

5.2.3 离开

- :q 如文件有修改而没保存，会警告，且无法离开。
- :q! 放弃所有修改，强迫离开。
- :wq 保存文件后离开，即使文件没有修改也会再保存一次。
- :x 保存文件后离开，但如果文件没有修改，则不会做保存的动作。
- :ZZ 和:x 完全一样。
- :w 另存，不加文件名就是写入原文件。

第6章 系统安全

随着现代通信技术的迅速发展，Internet 使用范围不断扩大、用户人数也在不断增加，而 Internet 上任何一台计算机都可能成为网络黑客试图攻击的对象。对于企业和关键应用领域的服务器系统来说，安全问题就显得更为重要。本章主要介绍 Red Flag Asianux Server 3 的系统安全管理策略。

6.1 系统安全概要

网络服务器作为 Internet/Intranet 上的关键设备，往往储存了大量的重要信息，或是向大量用户提供重要服务，一旦遭到破坏，后果将会很严重。所以网络建设者和管理员应认真对待安全方面的问题，以保证服务器的正常运行。

6.1.1 安全管理组成

Linux 系统安全包括 3 个要素：物理安全管理、普通用户安全管理和超级用户安全管理。

➤ 物理安全

- 保证放置计算机的机房的安全，必要时需加报警系统，同时应提供软件备份方案，把备份好的软件放到安全的地点；
- 保证所有的通信设施（包括有线通讯线、电话线、局域网、远程网等）都不会被非法人员监听；
- 钥匙或信用卡识别设备、用户口令和钥匙分配、文件保护、备份或恢复方案等关键文档资料要保存在安全的位置。

➤ 普通用户安全管理

- 系统管理员有责任发现并报告系统的安全问题，当普通用户登录时，其 Shell 在给出提示前先执行/etc/profile 文件，要确保该文件中的 Path 指定最后搜索当前工作目录；
- 系统管理员可定期抽取一个用户，将该用户安全检查结果（用户的登录情况简报、SUID/SGID 文件列表等）发送到其部门及相关人员；
- 注意提高安全管理意识。系统管理员应强化安全规则，用户必须遵守个人安全标准，在权限允许范围内进行操作，也可使用一些提高安全性的工具。

➤ 超级用户安全管理

- 在日常使用中最好不要使用 root 帐号，以普通用户身份进入系统可以防止对系统进行破坏性的操作，以 root 身份工作时应保证输入的每个命令的正确性；
- 超级用户不要运行其他用户的程序，如有需要，就选用 su 命令进入普通用户帐号；
- 经常改变 root 的用户口令；
- 设置用户口令的时效；
- 不要把当前工作目录排在 PATH 路径表的前边，以免特洛伊木马的侵入，键入/bin/su 来执行 su 命令；
- 不要未注销帐户就离开终端，特别是作为 root 用户更不能这样；
- 可以将登录名 root 改成别的名称，使破坏者不能在 root 用户登录名下猜测各种可能的用户口令从而非法进入 root 帐户；

- 查出不寻常的系统使用情况，如大量地占用磁盘、CPU 时间、进程，大量地使用 `su` 的企图，大量的无效登录与到某一系统的网络传输，以及可疑的 `uucp` 请求；
- 保持系统文件安全的完整性，检查所有系统文件的存取许可，要特别注意设备文件的存取许可，任何具有 `SUID` 许可的程序都可能是黑客攻击的对象；
- 将磁盘的备份存放在安全的地方；
- 查出久未使用的登录帐户，并取消此帐户；
- 确保没有无用户口令的登录帐户；
- 启动系统记帐、加密、RSA 等安全机制；
- 当安装来源不可靠的软件时，要检查源代码和 `makefile` 文件，查看特殊的子程序调用或命令；
- 如认为系统已泄密，就设法查出责任人与事故原因，并及时进行补救。

6.1.2 常见安全问题及对策

➤ 系统安装时的考虑

在系统安装的分区步骤中，不要只图简单便把所有的空间都留给根分区，应该把不同的部分放在不同的分区。建议把“`/var`”和“`/tmp`”放在不同的分区，如果服务器有较多的用户访问，这几乎是必须的。

另外，最好将“`/home`”和“`/usr`”也放在不同的分区，这样可以避免由于日志或用户的原因使硬盘被占满而导致服务器的性能降低。

对分区的最后一点建议是如果要提供一种或多种服务，最好要把这个服务有关的内容放在单独一个分区，例如：建立一台 `WWW` 服务器。在分区的时候，可以留一个单独的分区 `/www`，将来可以用 `chroot` 提高这种服务的安全性。

另外，出于安全和性能的考虑，应该在安装时或安装后进行软件包的选择，有些软件对系统提供的服务没有用，就不必安装。对服务器系统完成所有的配置工作后，可以将相应的软件包，编译程序包等从系统中删除或保存在活动介质上。

➤ 取消不必要的服务

一般来说，除了 `http`、`smtp`、`ssh` 之外，其他服务都可以取消，诸如网络邮件存储及接收所用的 `imap`/`ipop`、寻找和搜索资料用的 `gopher` 及用于时间同步的 `daytime` 和 `time` 等；

还有一些系统状态报告服务，如 `finger`、`systat`、`netstat` 等，虽然对系统查错和寻找用户非常有用，但也给黑客提供了方便之门。因此，也应考虑全部或部分取消，以增强系统的安全性。

➤ 帐户口令安全

`Linux` 采用了将系统管理员和一般用户分开的策略，这种策略保证了系统的健壮性，同时也使 `Linux` 下的病毒难以编写（用户编写的程序仅对自己的目录有写权限，而与操作系统的其它部分是隔离开的）。

首先应该设置 `BIOS` 口令，其次可以通过修改 `/etc/lilo.conf` 文件为 `Lilo` 的单用户模式设置口令限制，另外就是系统中用户的口令。

脆弱的口令是系统不安全的最主要原因，建议用下面的规则选择有效的口令：

- 至少要有 6 个字符，最好包含一个以上的数字或特殊字符；
- 口令不能太简单，所谓简单就是很容易被猜出来，避免用自己的名字、电话号码、生日、职业

或者其它个人信息作为口令；

- 不要把口令写在日历上或计算机旁边等别人能看到的地方；
- 应该设置口令的有效期，在一段时间之后就要更换口令；
- 如果发现有人试图猜测您的口令，而且已经试过多次了，就必须重新设定口令。

Red Flag Linux 中采用影子口令文件/etc/shadow 加强了户帐户的安全管理，并且提供了 PAM 身份验证机制，可以动态地改变身份验证方法和要求；而且通过设定 PAM，还能实现许多安全功能，大大提高系统的安全性。

➤ 对服务器进行设置

可以采用以下两种防范措施：

- 设置机器的 BIOS 选项，将从硬盘以外的启动设为不允许状态，并在 BIOS 上加设口令，使无权用户无法改变其设定；
- 为 GRUB 或 Lilo 设置口令，只有输入正确的口令时，才能启动机器；
- 只允许超级用户有使用<Ctrl+Alt+Del>组合键的权利，或禁止任何用户使用<Ctrl+Alt+Del>组合键的权利。这样，可以有效地阻止用户在本地控制台登录后，使用<Ctrl+Alt+Del>来关闭或重启系统，导致服务的中断。

➤ 文件系统安全

系统中的文件权限通过设置十位的权限标志位实现。所以对一些关键系统文件的属性设置要十分小心，以免导致不可挽回的损失。

带有文件的附加权限位 SUID 与 SGID 的程序运行时会引起很大的安全漏洞，应该尽量减少使用机会。系统管理员应该经常使用 find 命令来浏览自己的文件系统以检查新的 SUID 程序。

此外，可以用 chattr 命令来改变文件的属性，这里主要注意的是两个属性。

a 只可添加属性； i 不可改变属性；

对于系统的配置文件，最好设置为不可改变属性，而对于一些日志文件，可设置为只可添加属性。参见下面的示例：

```
chattr +i /etc/xinetd.conf
```

```
chattr +a /var/log/secure
```

如果要去掉这些属性，将上面命令中的“+”号改为“-”号即可。

➤ 保持最新的系统核心

Kernel 是 Linux 系统的核心，它常驻内存，实现基本的操作系统功能，它的安全性对整个系统的安全至关重要。

系统管理员可以通过编译核心，只选择必要的功能，这样既可节约系统资源，又防止了可能的系统漏洞。

在 Internet 上或是来自软件发行商经常会有最新的安全修补程序发布，系统管理员也应及时查阅新的修补程序，维护系统的安全运行。

定期对服务器进行备份、正确使用系统的加密设定、架设网络防火墙与采用一些系统监视工具、入侵检测工具等都是保护系统和网络安全的重要手段，我们将在本手册后面的章节中分别讨论如何实现这些功

能。

6.2 系统备份

世界上没有百分之百的安全，为了防止不可预料的网络攻击、系统硬件故障或用户的非法操作而发生数据丢失，系统管理员必须制订一个备份计划，并定期对系统进行安全备份，以便在系统万一崩溃时，可以及时将系统恢复到最佳状态。

6.2.1 备份前的准备

➤ 选择备份介质

有很多介质可以用来数据备份，目前比较常用的备份介质有软盘、磁带机、光盘和硬盘。对于少量数据的存储，软盘即可解决问题，但在大量数据的情况下，就需要光盘等介质了。用户需要根据自己系统备份计划的实际情况，从可靠性、速度、可用性、易用性和费用成本几个方面考虑来进行备份介质的选择。

➤ 进行备份的时机

进行系统备份要定期执行，备份通常应该选择在系统比较空闲时进行，以免影响系统处理正常任务，如可以选择在 0:00 之后进行。

进行系统备份也不可能完全依赖于管理员的手工操作，可以使用 Red Flag Asianux Server 3 系统的任务计划功能方便地完成，具体参见 [《Red Flag Asianux Server 3 用户手册》3.4.6 节：任务计划](#)。

➤ 备份策略的选择

- 完全备份

每隔一定时间对系统进行一次全面备份的方法，是最基本的备份方案。但这样做工作量很大，又需要过多的备份介质，因此不能频繁地进行全面备份，要隔一段较长时间，如一个月，进行一次完整备份。但这样一旦发生数据丢失，就只能恢复到上次备份的数据。

- 增量备份

先进行一次完全备份，然后每隔一个较短时间进行一次备份，仅备份在这个期间更改的内容。当经过一个较长时间的积累后再进行一次完全备份。这样每次备份的工作量小，能够频繁操作，而且也比较经济。

- 更新备份

与增量备份方式有些相似。首先每月进行一次完全备份，然后每天进行一次更新数据的备份。不同之处是：增量备份是备份该天更改的数据，而更新备份是备份从上次进行完全备份后更改的全部数据文件。一旦发生数据丢失，可以使用前一个完全备份恢复到前一个月的状态，再使用前一个更新备份恢复到前一天的情况。

增量备份和更新备份都能以较为经济的方式实现，在不同备份策略之间进行选择不但与系统数据更新的方式有关，也依赖于管理员的习惯。

➤ 备份工具的选择

有许多工具可用于制作备份。Red Flag Asianux Server 3 中提供了传统的 tar、bzip2、gzip、cpio 等工具，当然也可以使用其他第三方的软件包。

6.2.2 常用备份命令

有时候，我们需要把一组文件贮存成一个文件以便备份或传输到另一个目录甚至另一台计算机上。我

们还需要把一组文件压缩成一个文件，因而它占用少量的磁盘空间并能更快地通过网络上下载。

下面将介绍 Linux 下最常用的归档压缩工具 tar、bzip2、gzip 和 zip。

6.2.2.1 tar命令

利用 tar 可以将文件和目录归档，也可以在档案中改变文件，或向档案中加入新的文件。tar 最初被用来在磁带上创建档案，现在则可以在任何设备上使用。tar 命令实现把一大堆文件和目录全部打成一个包的功能，这对于备份或将几个文件组合成一个文件以便网络传输是非常有用的。

tar 命令的语法格式为：

tar [主选项+辅助选项] 文件或目录

使用时，主选项是必须的，辅助选项可以选用。主选项主要包括：

- c: 创建新的档案文件。如果用户想备份一个目录或一些文件，就选择此选项。
- r: 把要存档的文件追加到档案文件末。如用户已完成备份文件，又发现还有一部分文件或目录忘记了，就可以使用此选项。
- t: 列出档案文件的内容，查看已经备份了哪些文件。
- u: 更新文件。即用新增的文件取代原备份文件，如果在备份文件中找不到要更新的文件，则把它追加到备份文件的最后。
- x: 从档案文件中释放文件。

辅助选项主要有：

- b: 为磁带机而设定，其后跟一数字，用来说明区块的大小。
- f: 使用档案文件或设备，此选项通常为必选项。
- k: 保存已存在的文件，使用户在还原文件中，遇到相同的文件不会进行覆盖。
- m: 在还原文件时，把所有文件的修改时间设定为现在。
- M: 创建多卷的档案文件，以便在几个磁盘中存放。
- v: 详细报告 tar 处理的文件信息。
- w: 每一步都要求确认。
- z: 用 gzip 来压缩/解压缩文件，加上此选项后可以将档案文件进行压缩，还原时一定要该选项才能进行解压缩。

例 1: 把/home 目录包括其子目录全部做成备份文件 home.tar。

tar cvf home.tar /home

例 2: 把/home 目录包括其子目录全部备份并进行压缩，生成文件名为 home.tar.gz。

tar czvf home.tar.gz /home

例 3: 把 home.tar.gz 文件还原并解压缩。

tar xzvf home.tar.gz

例 4: 查看 home.tar 文件的内容，并以分屏方式显示在屏幕上。

```
# tar tvf home.tar |more
```

例 5：在软盘/dev/fd0 中创建一个备份文件，将/tmp 目录中所有的文件都拷贝进来。

```
# tar cf /dev/fd0 /tmp
```

要恢复设备磁盘中的文件，则可使用 xf 选项。

当需要备份的文件大小超过设备的可用存储空间时，可以创建一个多卷的 tar 文件，使用 M 选项向一个软盘存储过程中，系统在一张软盘已满时会提示放入新的软盘，以实现把 tar 档案存入多张磁盘中。如

```
# tar cMf /dev/fd0 /home
```

6.2.2.2 bzip2 和bunzip2

要使用 bzip2 来压缩文件，在 shell 提示符下键入以下命令：

```
bzip2 filename
```

该文件就会被压缩，并被保存为 filename.bz2。

要解开被压缩的文件，键入以下命令：

```
bunzip2 filename.bz2
```

filename.bz2 文件会被删除，而代之以 filename 文件。

可以使用 bzip2 命令同时处理多个文件和目录，方法是将它们逐一列出，并用空格隔开。例如：

```
bzip2 filename.bz2 file1 file2 file3 /usr/local/rfinput
```

上面的命令把 file1、file2、file3 以及/usr/local/rfinput 目录的内容压缩起来，存放到 filename.bz2 文件中。

6.2.2.3 gzip和gunzip命令

gzip 是一个经常使用的文件压缩和解压缩命令。该命令的语法格式为：

```
gzip [选项] 压缩/解压缩的文件名
```

常用的选项参数如下：

- c: 将输出写到标准输出上，并保留原有文件。
- d: 将压缩文件解压缩。
- l: 对每个压缩文件显示其大小、未压缩文件的大小、压缩比和名称等。
- r: 递归式地查找指定目录并压缩其中的所有文件或是解压缩。
- t: 测试、检查压缩文件是否完整。
- v: 对每一个压缩和解压缩文件，显示文件名和压缩比。
- num: 用指定的数字来调整压缩的速度。

现在假设在目录/home 下有文件 aa.txt、bb.txt、cc.txt，把它们压缩成.gz 文件的命令如下：

```
gzip /home/*
```

```
ls
```

```
aa.txt.gz bb.txt.gz cc.txt.gz
```

要将上例中的文件解压，并列出详细的信息，使用命令

```
gzip -dv /home/*
```

要解开被压缩的文件，也可以使用以下命令：

```
gunzip filename.gz
```

filename.gz 会被删除，而代之以 filename。



要获得这两个命令的详细信息，可以在 shell 提示下键入 `man gzip` 和 `man gunzip` 来阅读 `gzip` 和 `gunzip` 命令的说明书页。

6.2.2.4 zip和unzip

要使用 `zip` 命令压缩文件，在 shell 提示符下键入下面的命令：

```
zip -r filename.zip filesdir
```

在上例中，filename.zip 表示要创建的压缩文件，filesdir 表示要压缩的文件目录。`-r` 选项表示递归地压缩所有包括在 filesdir 目录中的文件。

要解压缩 filename.zip 文件，键入以下命令：

```
unzip filename.zip
```

可以使用 `zip` 命令同时处理多个文件和目录，方法是将它们逐一列出，并用空格隔开：

```
zip -r filename.zip file1 file2 file3 /usr/local/rfinput
```

上面的命令把 file1、file2、file3 以及 /usr/local/rfinput 目录的内容压缩起来，存放到 filename.zip 文件中。



要获得这两个命令的详细信息，请参阅 `zip` 和 `unzip` 命令的说明书页。

6.3 加密措施

Red Flag Asianux Server 3 在设计中充分考虑了安全因素，使用了多种有代表性的加密程序来保护系统与用户的安全。

6.3.1 SSH

SSH (Secure Shell) 是一个用来登录远程服务器并在远程服务器上执行命令的程序，在缺少安全防护的网络上，能为两台互不信任的主机间提供安全可靠的加密通信。

OpenSSH 是 SSH 协议的免费开源实现。

6.3.1.1 SSH的特点

使用 OpenSSH 工具能够增强系统的安全性。OpenSSH 加密所有的通信（包括口令），有效地防止了窃取和网络攻击。除此之外，OpenSSH 还提供了多种安全认证方法。而 telnet、rlogin、ftp 等连接工具使用纯文本口令，并被明文发送，这些信息可能会被截取，未经授权的人员可能会使用截取的口令登录系统

并造成危害。

OpenSSH 包括：ssh（替代了 rlogin 和 telnet）、scp（替代了 rcp）、sftp（替代了 ftp）和服务端端的 sshd。其他的基本工具包括 ssh-add、ssh-agent、ssh-keygen、ssh-keyscan 等。

OpenSSH 的一个很有用的功能是：它自动把 DISPLAY 变量转发给客户端。也就是说，如果在本地主机上运行 X 窗口系统，并且使用 ssh 命令登录到了远程机器上，当在远程机器上执行一个需要 X 的程序时，它会显示在本地主机上。这会使您的工作更加方便。

6.3.1.2 配置OpenSSH服务器

远程服务器只需运行 sshd 服务器守护进程即可。

要启动 OpenSSH 服务，使用/etc/rc.d/init.d/sshd start 命令；要停止 OpenSSH 服务，使用/etc/rc.d/init.d/sshd stop 命令。

OpenSSH 的服务器端配置文件是/etc/ssh/sshd_config，默认的配置文件的多数情况下，可以胜任。如果需要进行更细致的定制，请阅读 sshd 的手册页。

6.3.1.3 配置OpenSSH客户端

客户端需要安装有 openssh-clients 和 openssh 软件包，才可以连接到 OpenSSH 服务器上。

OpenSSH 客户端的配置文件是/etc/ssh/ssh_config，允许通过设置不同的选项来改变客户端程序的运行方式。有兴趣的用户可以查看 ssh 的相关手册页。

6.3.1.4 使用ssh命令

ssh 命令是 rlogin、rsh 和 telnet 命令的安全替换。它允许用户登录远程机器并在其上执行命令。使用 ssh 登录远程机器和使用 telnet 相似。

例如：登录到一个名为 example.test.com 的远程主机，在 shell 提示下键入如下命令：

```
ssh example.test.com
```

第一次使用 ssh 登录远程机器时，会看到和下面相仿的消息：

```
The authenticity of host 'example.test.com' can't be established.
```

```
RSA key fingerprint is 94:68:3a:3a:bc:f3:9a:9b:01:5d:b3:07:38:e2:11:0c.
```

```
Are you sure you want to continue connecting (yes/no)?
```

键入 **yes** 继续，把该服务器添加到您的已知主机列表中。下一步，便会询问用户在远程主机上的口令。输入正确口令后，即在远程主机的 shell 提示符下了。

如登录时未指定用户名，本地客户机登录远程机器时用的用户名会被传递给远程机器。如果想指定不同的用户名，请使用以下命令：

```
ssh username@example.test.com 或 ssh -l username example.test.com
```



使用 ssh 后，需要用“exit”命令退出登录。

ssh 命令还可以不经登录而在远程机器上执行命令。它的语法格式是：ssh hostname command。例如，查看远程主机 example.test.com 上/usr/share/apps 目录下的内容，就在 shell 提示下键入命令：

```
ssh example.test.com ls /usr/share/apps
```

输入正确的口令后，远程机器/usr/share/apps 目录下的内容将会被显示出来，然后返回到本地 shell 提示下。

6.3.1.5 使用scp命令

scp 命令可以通过安全、加密的连接在机器间传输文件。它的使用与 rcp 相似。

把本地文件传输给远程机器的一般语法是：

```
scp localfile username@remotehostname:/remotefile
```

localfile 指定本地源文件，username@remotehostname:/remotefile 指定远程目标文件。

要把本地文件 file1 传送到用户在 example.test.com 上的主目录中，在 shell 提示下键入：

```
scp file1 username@example.test.com:/home/username
```

把远程文件传输给本地系统的一般语法是：

```
scp username@remotehostname:/remotefile /localfile
```

remotefile 指定远程源文件，localfile 指定本地目标文件。

源文件可以由多个文件组成。例如，要把目录/downloads 的内容传输到远程机器 example.test.com 上现存的 uploads 目录，键入下列命令：

```
scp /downloads/* username@example.test.com:/uploads/
```

6.3.1.6 使用sftp命令

sftp 命令是 ftp 命令的安全替换，用来打开一次安全的 FTP 交互会话。它的使用与 ftp 相似，但它使用的是安全、加密的连接。

sftp 一般语法是：sftp username@hostname。一旦通过验证，就可以使用一组和 FTP 相似的命令。



请参阅sftp的手册页，以获取该命令的列表。要阅读手册页，请在shell提示符下执行man sftp命令。

6.3.1.7 使用RSA/DSA认证

OpenSSH 不仅是安全的而且是加密的。OpenSSH 的一个更加吸引人的特性是其功能组件——RSA/DSA 密钥认证系统，它可以代替 OpenSSH 缺省使用的标准安全密码认证系统。

RSA 和 DSA 认证协议基于一对专门生成的密钥（公钥和私钥）来认证用户。经过适当的配置，能够不必提供密码即可同远程机器建立安全的连接。

RSA 和 DSA 认证需要一些初始配置。要设置 RSA 和 DSA 认证，首先需要生成一对密钥，即一把私钥和一把公钥。公钥用于对消息进行加密，只有拥有私钥的人才能对该消息进行解密。公钥只能用于加密，而私钥只能用于解密由匹配的公钥编码的消息。

Red Flag Asianux Server 3 默认使用 SSH 协议 2 和 RSA 密钥。密钥必须单独为每个用户生成。要为某用户生成密钥，用将要连接到远程机器的用户身份来执行下面的步骤。如果以 root 身份执行下列步骤，就只有 root 用户才能使用这对密钥。

➤ 生成RSA密钥对

- 1) 要生成 RSA 密钥对，先在 shell 提示符下键入下列命令：

ssh-keygen -t rsa

- 2) 当要求输入存放密钥的位置时，按<Enter>键接受~/.ssh/id_rsa 的默认位置。接下来输入一个与用户帐号口令不同的口令，再输入一次以确认。
- 3) 命令完成后，公钥被写入~/.ssh/id_rsa.pub；私钥被写入~/.ssh/id_rsa。注意：一定不要把私钥出示给任何人。
- 4) 使用 `chmod 755 ~/.ssh` 命令改变用户主目录下.ssh 目录的许可权限。
- 5) 把公钥~/.ssh/id_rsa.pub 的内容复制到想要连接的远程机器上的~/.ssh/authorized_keys 文件中。如果文件~/.ssh/authorized_keys 不存在，可以把~/.ssh/id_rsa.pub 文件复制为远程机器的~/.ssh/authorized_keys 文件。

➤ 生成DSA密钥对

- 1) 要生成 DSA 密匙对，先在 shell 提示符下键入下面的命令：

ssh-keygen -t dsa

- 2) 当要求输入存放密钥的位置时，接受~/.ssh/id_dsa 的默认位置。接下来输入一个与用户帐号口令不同的口令，再输入一次以确认。
- 3) 命令完成后，公钥被写入~/.ssh/id_dsa.pub；私钥被写入~/.ssh/id_dsa。注意：一定不要把私钥出示给任何人。
- 4) 使用 `chmod 755 ~/.ssh` 命令改变用户主目录下的.ssh 目录的许可权限。
- 5) 把公钥~/.ssh/id_dsa.pub 的内容复制到想要连接的远程机器中的~/.ssh/authorized_keys2 文件中。如果文件~/.ssh/authorized_keys2 不存在，可以把~/.ssh/id_dsa.pub 文件复制为远程机器上的~/.ssh/authorized_keys2 文件。

➤ 配置ssh-agent

ssh-agent 是一个用于保存私钥的授权代理。只要使用 `ssh-add` 命令把私钥添加到 ssh-agent 的高速缓存中，ssh 将从 ssh-agent 获取您的私钥，而不会提示要密码了

- 1) 在 shell 提示符下，键入下面的命令：

exec /usr/bin/ssh-agent \$SHELL

- 2) 然后，键入下面的命令：

ssh-add

- 3) 接着，输入你的密钥口令。如果配置了不止一个密钥对，会被提示输入每个口令。
- 4) 当用户注销后，口令就会被忘记。必须在每次登录到虚拟控制台或打开终端窗口时都执行这两条命令。



默认情况下，系统禁止 root 用户通过 ssh 远程登录，只能以普通用户身份登录。



可以访问OpenSSH的官方站点：<http://www.openssh.com>获得更多详细的信息。

6.3.2 PGP

PGP—Pretty Good Privacy，是一个基于 RSA 公匙加密体系的邮件加密软件。可以用它对您的邮件加密以防止非授权者阅读，还能在邮件中加上数字签名从而使收信人可以确信邮件的来源。它让用户可以安全地和从未见过的人们通讯，事先并不需要任何保密的渠道用来传递密匙。它采用了审慎的密匙管理，这种 RSA 和传统加密的杂合算法，用于数字签名的邮件文摘算法，加密前压缩等。

PGP 的创始人是美国的 Phil Zimmermann。它的创造性在于把 RSA 公匙体系的方便和传统加密体系的高速度结合起来，并且在数字签名和密匙认证管理机制上有巧妙的设计。



如果需要了解关于PGP的详细知识，请访问以下的参考站点：<http://www.pgpi.org>。

6.3.3 OPENSsl

Openssl 是一个协议独立的加密方案，在网络信息包的应用层和传输层之间提供了安全的通道。

一些服务器软件，例如 IMAP、POP、Samba、FTP、Apache 等，在提供服务时需要用户对用户进行认证，只有认证通过后服务才会被许可。然而对于 server/client 方式的服务，客户端和服务端之间通讯都是以明文方式进行的，Openssl 正是提供了对传输的数据的一种加密方式。

Openssl 可以安装在 Linux 服务器上，它需要一些第三方提供的应用程序来为服务提供加密。简单地说，就是 HTML 或 CGI 经过幕后的服务器进行了加密处理，然而对 HTML 和 CGI 的作者来说是透明的。

Openssl 软件包提供了 SSL (Secure Sockets Layer) 及 TLS (Transport Layer Security) 协议的加密保护，而且提供了 apache 方式的许可证，从而强化了 HTTP 服务器的安全性。



可以访问Openssl的官方网址：<http://www.openssl.org/>以获得更多、更新的信息。

第7章 网络安全

安全性是网络服务器可靠运行的基础。随着通信技术和 Internet 的广泛应用，服务器被攻击的情况可能经常发生，来自网络上的安全威胁是 Linux 服务器安全问题的主要来源。

本章介绍如何在 Red Flag Asianux Server 3 构建的服务器平台上，利用系统提供的安全工具，达到有效保护系统安全、减少成功入侵数量、检测和追踪入侵日志、降低危害程度并快速从攻击中恢复的策略。



有关物理和文件系统安全的相关知识请参考第6章 系统安全。

7.1 Xinetd

7.1.1 简介

xinetd 提供了访问控制，改进的日志功能和资源管理，是 Red Flag Asianux Server 3 系统中的 Internet 标准超级守护进程。

Inetd 被称作超级服务器，用来实现对主机网络连接的控制。当一个请求到达由 Inetd 管理的服务端口，Inetd 将该请求转发给名为 tcpd 的程序。tcpd 根据配置文件/etc/hosts.allow 和/etc/hosts.deny 来判断是否允许服务响应该请求。如果请求被允许则相应的服务器程序（如：telnetd）将被启动。这个机制也被称作 tcp_wrapper。

xinetd (eXtended InterNET services daemon) 提供类似于 inetd + tcp_wrapper 的功能，但是更加强大和安全。它具有以下特色：

- 支持对 tcp、udp、RPC 服务（但是当前对 RPC 的支持不够稳定，可以启动 protmap 与 xinetd 共存来解决这个问题）。
- 基于时间段的访问控制。
- 功能完备的 log 功能，即可以记录连接成功也可以记录连接失败的行为。
- 能有效的防止 DoS 攻击（Denial of Services）。
- 能限制同时运行的同一类型的服务器数目。
- 能限制启动的所有服务器数目。
- 能限制 log 文件大小。
- 将某个服务绑定在特定的系统接口上，从而能实现只允许私有网络访问某项服务。
- 能实现作为其他系统的代理。如果和 IP 伪装结合，可以实现对内部私有网络的访问。

7.1.2 Xinetd的配置

Xinetd 的配置文件是/etc/xinetd.conf 与/etc 目录下的一个名为 xinetd.d 的网络连接配置文件目录。配置文件中的语法和/etc/inetd.conf 完全不同且不兼容。它本质上是/etc/inetd.conf 和/etc/hosts.allow，/etc/hosts.deny 功能的组合。

以下所示为/etc/xinetd.d 目录下的文件

```
[root@localhost root]# ls /etc/xinetd.d
amanda      cups-lpd    finger      ntalk       rsh         swat
amandaidx   daytime     imap        pop3s       rsync       talk
amidxtape   daytime-udp imaps        proftpd     servers     telnet
chargen     echo        ipop2        rexec       services    time
chargen-udp echo-udp     ipop3        rlogin      sgi_fam     time-udp
```

其中每个文件代表一种网络服务器程序，文件中一般具有下列形式。

```
service service-name
{
.....
.....
}
```

其中 service 是必需的关键字，每一项都定义了由 service-name 定义的服务。例如下面给出的是文件 /etc/xinetd.d/telnet 的内容：

```
[root@localhost xinetd.d]# cat telnet
# default: on
# description: The telnet server serves telnet sessions; it uses \
#      unencrypted username/password pairs for authentication.
service telnet
{
    flags          = REUSE
    socket_type    = stream
    wait           = no
    user           = root
    server          = /usr/sbin/in.telnetd
    log_on_failure += USERID
    disable        = yes
}
```

Service-name 是任意的，但通常是标准网络服务名，也可以增加其他非标准的服务，只要它们能通过网络请求激活，包括 localhost 自身发出的网络请求。

操作符可以是=，+=，或-=。所有属性都可以使用=，其作用是分配一个或多个值，某些属性可以使用+=或-=形式，其作用分别是将该值增加到某个现存的值表中，或将其从现存值表中删除。

➤ 可使用的属性

Socket_type

使用的 TCP/IP socket 类型，值可能为 stream (TCP)，dgram (UDP)，raw 和 seqpacket (可靠的有序数据报)。

protocol

指定该服务使用的协议，其值必须是在/etc/protocols 中定义的。如果不指定，使用该项服务的缺省协议。

Server

要激活的进程，必须指定完整路径。

Server_args

指定传送给该进程的参数，但是不包括服务程序名。

Port

定义该项服务相关的端口号。如果该服务在/etc/services 中列出，它们必须匹配。

Wait

这个属性有两个可能的值。如果是 yes，那么 xinetd 会启动新的进程并停止处理该项服务的请求直到该进程终止。这是个单线程服务。如果是 no，那 xinetd 会为每个请求启动的一个进程，而不管先前启动的进程的状态。这是个多线程服务。

User

设置服务进程的 UID。若 xinetd 的有效 UID 不是 0，该属性无效。

Group

设置进程的 GID。若 xinetd 的有效 UID 不是 0，这个属性无效。

Nice

指定进程的优先级。

Id

该属性被用来唯一地指定一项服务。因为有些服务的区别仅仅在于使用不同的协议，因此需要使用该属性加以区别。默认情况下服务 id 和服务名相同。如 echo 同时支持 dgram 和 streama 服务。设置 id=echo_dgram 和 id=echo_streams 来分别唯一标识两个服务。

Type 可以是下列一个或多个值：RPC（对 RPC 服务），INTERNAL（由 xinetd 自身提供的服务，如 echo），UNLISTED（没有列在标准系统文件如/etc/rpc 或/etc/service 中的服务）。

Access_time

设置服务可用时的时间间隔。格式是 hh: mm_hh: mm；如 08:00-18:00 意味着从 8A.M 到 6P.M.可使用这项服务。

Banner

无论该连接是否被允许，当建立连接时就将该文件显示给客户机。

Flags

可以是以下一个或多个选项的任意组合：

REUSE:	设置 TCP/IP socket 可重用。也就是在该服务 socket 中设置 SO_REUSEADDR 标志。当中断时重新启动 xinetd。
INTERCEPT:	截获数据报进行访问检查，以确定它是来自于允许进行连接的位置。
NORETRY:	如果 fork 失败，不重试。
IDONLY:	只有在远程端识别远程用户时才接受该连接（也就是远程系统必须运行 ident 服务器），该标记只适用于面向连接的服务。若没有使用 USERID 记录选项则该标记无效 log_on_success 和/或 log_on_failure 属性设置 USERID 值以使该值生效。仅用于多线程的流服务。
NAMEINARGS:	允许 server_args 属性中的第一个参数是进程的完全合格路径，以允许使用 TCP_Wrappers。

NODELAY:	若服务为 tcp 服务，并且 NODELAY 标记被设置，则 TCP_NODELAY 标记将被设置。若服务不是 tcp 服务则该标记无效。
-----------------	---

Rpc_version

指定 RPC 版本号或服务号。版本号可以是一个单值或者一个范围中。

rpc_number

如果 RPC 程序号不在/etc/rpc 中，就指定它。

Env

用空格分开的 VAR=VALUE 表，其中 VAR 是一个 shell 环境变量且 VALUE 是其设置值。这些值以及 xinetd 的环境都在激活时传送给服务程序。这个属性支持=和+=操作符。

Passenv

用空格分开的 xinetd 环境中的环境变量表，该表在激活时传递给服务程序。设置 no 就不传送任何变量。该属性支持所有操作符。

Only_from

用空格分开的允许访问服务的客户机表。例如：only_from=172.16.84.2。如果不为该属性指定一个值，就拒绝访问这项服务。该属性支持所有操作符。

No_access

用空格分开的拒绝访问服务的客户机表。例如：no_access =172.16.84.3。该属性支持所有操作符。

Instances

接受一个大于或等于 1 的整数或 UNLIMITED。设置可同时运行的最大进程数。UNLIMITED 意味着 xinetd 对该数没有限制。

Log_type

指定服务 log 记录方式，可以为：

SYSLOG facility[level]:	设置该工具为 daemon, auth, user 或 local0-7。设置 level 是可选的，可以的 level 值为 emerg, alert, crit, err, warning, notice, info, debug, 默认值为 info;
file[soft[hard]]:	指定 file 用于记录 log，而不是 syslog。限度 soft 和 hard 用 KB 指定（可选）。一旦达到 soft 限，xinetd 就登记一条消息。一旦达到 hard 限，xinetd 停止登记使用该文件的所有服务。如果不指定 hard 限，它成为 soft 加 1%，但缺省时不超过 20MB。缺省 soft 限是 5MB；

Redirect

该属性语法为 redirect=Ipaddress port。它把 TCP 服务重定向到另一个系统。如果使用该属性，就忽略 server 属性。

Bind

把一项服务绑定到一个特定端口。语法是 bind=Ipaddress。这样有多个接口（物理的或逻辑的）的主机允许某个接口但不是其他接口上的特定服务（或端口）。

Log_on_success

指定成功时登记的信息。可能值是：

ID:	进程的 PID。如果一个新进程没被分叉，PID 设置为 0。
HOST:	客户机主机 IP 地址。
USERID:	通过 RFC1413 高用捕获客户机用户的 UID。只可用于多线程流服务。
EXIT:	登记进程终止和状态。
DURATION:	登记会话持续期。

缺省时不登记任何信息。该属性支持所有操作符

Log_on_failure

指定失败时登记的信息。总是登记表明错误性质的消息。可能值是 **ATTEMPT**：记录一次失败的尝试。所有其他值隐含为这个值。

ATTEMPT:	记录一次失败的尝试。所有其他值隐含为这个值。
HOST:	客户机主机 IP 地址。
USERID:	通过 RFC1413 调用捕获客户机用户的 UID。只可用于多线程流服务
RECORD:	记录附加的客户机信息如本地用户，远程用户和终端的类型。缺省时不登记任何信息。该属性支持所有操作符。

Disabled

只可用于 defaults 项，指定被关闭的服务列表，是用空格分开的不可用服务列表来表示的。它和在 /etc/xinetd.conf 文件中注释掉该服务项有相同的效果。

➤ 使配置生效

完成对相关网络服务程序配置文件的修改后，可以使用如下命令重新启动 xinetd 守护进程使配置生效。

```
/etc/rc.d/init.d/xinetd restart
```

7.2 可插入验证模块（PAM）

使用身份验证的方法来验证用户身份从而为用户分配对系统访问权限的程序（即确定用户的身份）。PAM 使用一个可插入式的、模块化的结构。它为系统管理员设置用户身份验证政策提供了很大的灵活性。

在多数情况下，一个支持 PAM 的应用程序使用默认的 PAM 配置文件就可满足一般的要求。但在某些情况下，还需要对 PAM 配置文件进行特殊的配置。因为如果对 PAM 进行了错误的配置就可能会导致系统的安全性被破坏，所以在对这些文件进行配置前必须对其结构有一定的了解。

7.2.1 PAM 的优越性

PAM 提供了以下优越性：

- 可以被多种应用程序使用的常用用户身份验证系统。
- 为系统管理员和应用程序开发者在进行用户身份验证处理时提供了极大的灵活性。
- 提供了一个详细的用户接口，使开发者们不需要再开发自己的用户身份验证系统。

7.2.2 PAM配置文件

/etc/pam.d/目录中包括所有支持 PAM 应用程序的 PAM 配置文件。以前的 PAM 版本使用/etc/pam.conf 文件，但现在这个文件已经过时，只在/etc/pam.d/目录不存在时才被使用。

7.2.2.1 PAM服务文件

每个支持 PAM 的应用程序或服务都在/etc/pam.d/目录中有一个相应的文件。而该目录中的每个文件的名称都与其所控制访问的服务名称相同。

支持 PAM 的应用程序需要定义它们的服务名并在/etc/pam.d/目录中安装它们自己的 PAM 配置文件。例如，login 程序定义了它的服务名为 login 并安装/etc/pam.d/login PAM 配置文件。

7.2.3 PAM配置文件的格式

每个 PAM 配置文件都包括一组有以下格式的命令：

<模块接口> <控制标识> <模块名称> <模块参数>

这些选项将在以下一节中陆续介绍。

7.2.3.1 模块接口 (Module Interface)

当前有四类可用的 PAM 模块接口。每类接口分别代表了用户身份验证的不同方面：

- **auth** — 这个模块接口用作验证。例如，它需要一个密码并验证密码是否正确。带有此接口的模块也可以设置许可证，如组群成员或 Kerberos 令牌。
- **account** — 这个模块接口用来验证所允许的访问。例如，检查一个用户帐号是否已经过期；或一个用户是否允许在一天的某个特定的时间进行登录。
- **password** — 这个模块接口用来改变用户的密码。
- **session** — 这个模块接口用来配置并管理用户会话。带有这个接口的模块也可以进行允许访问所需的额外任务，如挂载用户的主目录，使用户的邮箱有效。



一个单独的模块可以包括一个或多个模块接口。例如，pam_unix.so 包括了所有四个模块接口。

在一个 PAM 配置文件中，模块接口是第一个定义的项。一般的配置文件都会有与以下形式类似的行：

```
auth      required pam_unix.so
```

它告诉 PAM 使用 pam_unix.so 模块的 auth 接口。

7.2.3.2 模块接口堆积

模块界面命令可以堆积 (stacked) 或在一个上面堆放另一个，这样多个模块可以一起使用来实现一个目的。那么模块排列的顺序对用户身份验证的过程就非常重要。

系统管理员要求先有特定的条件存在才允许用户进行身份验证，堆积命令可以使这个过程更容易。例如，reboot 命令通常使用一组堆积模块，请看下面的 PAM 配置文件：

auth	required	pam_nologin.so
auth	required	pam_securetty.so
auth	required	pam_env.so
auth	sufficient	pam_rhosts_auth.so
auth	required	pam_stack.so service=system-auth

在允许某人使用 `rlogin` 以前，PAM 会验证 `/etc/nologin` 是否存在，如不存在，PAM 会确认该用户不想以 `root` 身份去登录远程访问网络连接，而且所有的环境变量将被装载。然后，如果一个 `rhosts` 验证被成功执行，那么就允许连接。如果 `rhosts` 验证失败，那么就要执行标准的密码认证过程。

7.2.3.3 控制标志 (Control Flag)

所有的 PAM 模块在调用后都会产生一个成功或失败的结果。控制标志会指示 PAM 根据其结果下一步应如何做。模块可以以特定的顺序进行堆积排列，控制标志决定一个特定模块对用户验证成功或失败的整个过程的重要性。

有四个预先定义好的控制标志：

- **required** —要使验证过程继续，这个模块的结果必须是成功。如果这一步的测试失败，用户不会马上被通知，而在所有模块的测试都完成后才会被通知。
- **requisite** —要使验证可以继续进行，这个模块的结果必须是成功。然而，如果这一步测试失败，用户会被马上通知，而通知的信息中包括了第一个失败的 **required** 或 **requisite** 模块测试。
- **sufficient** —如果模块验证失败，其结果就不会被理会。但是，如果一个标识为 **sufficient** 的模块成功，并且前面没有任何标识为 **required** 的模块验证失败，那么用户就可以通过验证使用这个服务，无需其它任何验证结果。
- **optional** —模块结果不被理会。标识为 **optional** 的模块只有在这个界面中没有引用其它模块时，才成为验证成功所必须的条件。



required 模块调用顺序并不重要。只有控制标志为 sufficient 和 requisite 的模块的调用顺序才变得重要。

PAM 现在可使用一个更新的、可以进行更准确控制的控制标志语法。PAM 的帮助文件在 `/usr/share/doc/pam-<version-number>/` 目录下，其中的 `<version-number>` 是用户系统上的 PAM 的版本号，在此有崭新的语法描述。

7.2.3.4 模块名

模块名称提供了包含特定的模块接口的 PAM 可插入的模块的名字。在较老版本的服务器产品中，PAM 配置文件包括模块的完全路径。但是，自从使用了 `multilib` 系统（它就把 64 位的 PAM 模块存储在 `/lib64/security/` 中了），目录名就可以省略，应用程序会连接到适当的 `libpam` 版本，它可以定位这个模块的正确版本。

7.2.3.5 模块参数

在为一些模块进行验证时，PAM 使用参数来把信息传递给一个可插入模块。

例如，`pam_userdb.so` 模块使用存储在一个 Berkeley DB 文件中的信息来验证用户。Berkeley DB 是一个开源的数据库系统，内置于许多应用程序中。这个模块使用一个 `db` 参数，从而使 Berkeley DB 知道请求的服务使用的是哪个数据库。

以下是在一个 PAM 配置中一个典型的 `pam_userdb.so` 行。


```
auth        required pam_userdb.so db=<path-to-file>
```

<path-to-file>是到 Berkeley DB 数据库文件的完全路径。

无效的参数通常会被忽略，它不会影响到 PAM 模块验证的结果。但在一些模块中，无效的参数可能会导致模块验证的失败。多数模块都会把错误写入 /var/log/secure 文件。

7.2.4 创建 PAM 模块

用户可以随时为支持 PAM 的应用程序创建或添加新的 PAM 模块。例如，程序开发人员可以创建一个一次性密码的产生方法，并编写一个 PAM 模块来支持它。支持 PAM 的应用程序可以马上使用这个新模块而不用重新进行编译。这可让程序开发人员和系统管理员在不需要重新编译的情况下，为不同的应用程序测试各种用户身份验证方法。

编写模块的文档包括在 /usr/share/doc/pam-<version-number>/ 目录中。其中，<version-number> 是用户系统上的 PAM 版本号。

7.2.5 采用防火墙的优势

防火墙能提高主机与子网的安全性，红旗 Linux 防火墙体系的主要优点包括：

- 保证对主机和应用安全访问；
- 保证多种客户机和服务器的安全性；
- 保护关键部门不受来自内部、外部的攻击，为通过 Internet 与远程访问的雇员、客户、供应商提供安全通道。

7.2.6 防火墙安全控制基本原则

安全策略是防火墙的灵魂和基础。在建立防火墙之前要在安全现状、风险评估和商业需求的基础上提出一个完备的总体安全策略，这是配制防火墙的关键。防火墙在安全控制方面有两个基本准则：

- 准许访问除明确拒绝以外的全部访问——所有未被禁止的都允许访问
- 拒绝访问除明确准许的全部访问——所有未被允许的都禁止访问

可以看出，后一逻辑限制性大，前一逻辑比较宽松。

7.2.7 防火墙基本类型

防火墙的类型包括很多种，但大体上可以分为两类：一类基于 packet filter（包过滤型），另一类是基于 proxy service（代理服务）。它们的区别在于基于 packet filter 的 Firewall 通常直接转发报文，它对用户完全透明，速度较快。而基于 proxy 的 Firewall 则不是这样，它通过 proxy service 建立连接，可以有较强的身份验证和注册功能。

➤ 包过滤防火墙

包过滤（Packet filter）是在网络层中对数据包实施有选择的通过，依据系统事先设定好的过滤逻辑，检查数据流中的每个数据包，根据数据包的源地址、目标地址、以及包使用端口以确定是否允许该类数据包通过。包过滤式的防火墙会检查所有通过信息包里的 IP 地址，并按照系统管理员所给定的过滤规则过滤信息包。如果防火墙设定某一 IP 为危险的话，从这个地址而来的所有信息都会被防火墙屏蔽掉。

Packet filter 通常安装在路由器上，当然用来充当路由器的 PC 机上也可以安装 Packet filter，而且可能

会有更强的功能，如 Red Flag Asianux Server 3 中的 iptables。

IP 地址和端口号是网络层和传输层的特性，但 Packet filter 也同样可以在应用层工作。Internet 的应用程序通常有约定俗成的专用端口号。例如，telnet 服务总是在 TCP 端口 23 上运行。因此，可以设置一个 Firewall，来阻止向内部节点发送 telnet 请求。

使用 Packet filter 模式的 Firewall 的好处在于，在原有网络上几乎不需要增加任何额外费用就可以实现防火墙功能，因为差不多所有的路由器都可以对通过的封包进行过滤。目前，已安装的 Firewall 80% 都是包过滤模式的，它们都是在连接内部网络与 Internet 的路由器上设置了一些过滤原则。

包过滤防火墙不需要用户名和密码来登录，用户可能不会察觉到它的存在。这种防火墙速度快而且易于维护，具有更高的网络性能和更好的应用程序透明性。当然，它无法有效地区分同一 IP 地址的不同用户，安全性相对较低。

➤ 代理（Proxy Server）

代理服务器通常也称作应用级防火墙。所谓代理服务，即防火墙内外的计算机系统应用层的链接是通过两个终止于代理服务的链接来实现的，这样便成功地实现了防火墙内外计算机系统的隔离。

应用级网关使用软件来转发和过滤特定的应用服务，如 TELNET、FTP 等服务的连接。这是一种代理服务。它只允许有代理的服务通过，也就是说只有那些被认为“**可信赖的**”服务才被允许通过防火墙。另外代理服务还可以过滤协议，如过滤 FTP 连接、拒绝使用 FTP 放置命令等。应用级网关具有登记、日记、统计和报告功能，有很好的审计功能。还可以具有严格的用户认证功能。

代理服务器通常拥有高速缓存，在应用中可以节约时间与网络资源。代理服务具有信息隐蔽、保证有效的认证和登录、简化过滤规则等优点。网络地址转换服务（NAT Network Address Translation）可以屏蔽内部网络的 IP 地址，使网络结构对外部来讲是不可见的。

7.2.8 Iptables

Red Flag Asianux Server 3 中提供的 iptables 是建立在 netfilter 架构基础上的一个包过滤管理工具，最主要的作用是用来做防火墙或透明代理。

Iptables 从 ipchains 发展而来，它的功能更为强大。Iptables 提供以下三种功能：包过滤、NAT（网络地址转换）和通用的 pre-route packet mangling。

包过滤：用来过滤包，但是不修改包的内容。Iptables 在包过滤方面相对于 ipchains 的主要优点是速度更快，使用更方便。

NAT：NAT 可以分为源地址 NAT 和目的地址 NAT。

Iptables 可以追加、插入或删除包过滤规则。实际上真正执行这些过滤规则的是 netfilter 及其相关模块（如 iptables 模块和 nat 模块）。Netfilter 是 Linux 核心中一个通用架构，它提供了一系列的“表”（tables），每个表由若干“链”（chains）组成，而每条链中可以有一条或数条“规则”（rule）组成。

系统缺省的表为“filter”，该表中包含了 INPUT、FORWARD 和 OUTPUT 3 个链。每一条链中可以有一条或数条规则，每一条规则都是这样定义的：**如果数据包头符合这样的条件，就这样处理这个数据包**。当一个数据包到达一个链时，系统就会从第一条规则开始检查，看是否符合该规则所定义的条件：**如果满足，系统将根据该条规则所定义的方法处理该数据包；如果不满足则继续检查下一条规则**。最后，如果该数据包不符合该链中任一条规则的话，系统就会根据该链预先定义的策略来处理该数据包。

7.2.8.1 使用iptables

用户通过/sbin/iptables 命令来管理 iptables，和 route 命令相同，iptables 命令的效果在重新启动以后就不再有效。

可以使用/etc/rc.d/init.d/iptables save 将当前 iptables 规则写到/etc/sysconfig/iptables 文件中，那么每次开机时/etc/rc.d/init.d/iptables start 命令会使/etc/sysconfig/iptables 中的规则生效。

➤ table, chain, rule

iptables 可以操纵 3 个表：filter 表，nat 表，mangle 表。NAT 和一般的 mangle 用 -t 参数指定要操作哪个表。filter 是默认的表，如果没有 -t 参数，就默认对 filter 表操作。

Rule 规则：过滤规则，端口转发规则等，例如：禁止任何机器 ping 我们的服务器，可以在服务器上设置一条规则：

```
bash> iptables -A INPUT -s ! 127.0.0.1 -p icmp -j DROP
```

从 -s 开始即是一条规则，-j 前面是规则的条件，-j 开始是规则的行为（目的）。整条命令解释为，在 filter 表中的 INPUT 规则链中插入一条规则，所有源地址不为 127.0.0.1 的 icmp 包都被抛弃。

Chain 规则链：由一系列规则组成，每个包顺序经过 chain 中的每一条规则。chain 又分为系统 chain 和用户创建的 chain。下面先叙述系统 chain。

filter 表的系统 chain	INPUT, FORWARD, OUTPUT
nat 表的系统 chain	PREROUTING, POSTROUTING, OUTPUT
mangle 表的系统 chain	PREROUTING, OUTPUT

每条系统 chain 在确定的位置被检查。比如在包过滤中，所有的目的地址为本地的包，则会进入 INPUT 规则链，而从本地出去的包会进入 OUTPUT 规则链。

所有的 table 和 chain 开机时都为空，设置 iptables 的方法就是在合适的 table 和系统 chain 中添加相应的规则。

用户可以创建新的 chain。用户 chain 只有作为某个系统 chain 的目的才有作用。比如创建一个名叫 AAA 的规则链，想让 icmp 包通过它的检验。

```
bash> iptables -A INPUT -p icmp -j AAA
```

上述命令将用户创建的规则链 AAA 作为了一条系统 chain（INPUT）中的规则“-p icmp”的目的。

➤ policy: 默认策略

对每条 chain 有一个默认策略，也就是对包的默认的行为。可以设为抛弃（DROP）或接受（ACCEPT）。系统启动的时候所有的默认策略都是 ACCEPT。当包通过了 chain 所有的规则（不符合所有的规则的条件）的时候，系统按默认策略处理这个包。

7.2.8.2 iptables命令

➤ 针对表的操作

查看：iptables -t table_name -L

刷新：

- 1、清除所有的规则和用户创建的 chain

```
iptables -t table_name -F
```

- 2、清除所有的记数（符合规则的包的数目）

```
iptables -t table_name -Z
```

➤ 针对链的操作（注意以下都省略了 -t table_name）

查看: `iptables -L chain_name`

刷新: `iptables -F chain_name`

清除记数: `iptables -Z chain_name`

创建新链: `iptables -N chain_name`

删除链: `iptables -X chain_name`

重命名: `iptables -E chain_old_name chain_new_name`

设置策略: `iptables -P chain_name policy`

➤ 针对规则的操作

添加一条规则: `iptables -A chain_name rule-spec`

插入一条规则: `iptables -I chain_name 规则号 new_rule_spec`（插入后的规则号为命令中指定的号，原来存在的规则号顺延）。

删除一条规则: 有两种方法删除规则: 通过指定规则号删除一条规则或通过指定规则的内容来删除一条规则。

每个规则链中的规则号从 1 开始记数。

```
iptables -D chain_name 规则号
```

```
iptables -D chain_name 规则内容
```

修改一条规则: `iptables -R chain_name 规则号 new_rule_spec`



要想有效的使用 `iptables`, `Packet-filting-HOWTO` 和 `NAT-HOWTO` 是很重要也是非常好的文档。这两个 `HOWTO` 都有中文版本可供使用者参考。

7.3 SSH协议

SSH™ (Secure SHell 的缩写) 是一种用在基于客户机/服务器方式的两种系统间完成通讯, 并允许用户登录到服务器主机上进行远程控制的协议。与 `ftp`、`telnet` 等其他远程通信协议不同的是, `ssh` 对登录会话进行加密, 使入侵者无法收集加密的口令。

SSH 是用来取代原来的那些古老的、不安全的用来登录远程主机的安全终端应用程序, 例如 `telnet`、`rsh`。一个被称作 `SCP` 的程序取代了用来在两台主机间进行文件拷贝的旧程序, 如 `rccp`。因为这些旧的应用程序在客户端和服务端间未采用加密的传输方式, 而使用安全的方法登录到远程系统, 降低了客户系统和远程主机间的风险。

7.3.1 SSH的特性

SSH 协议提供了以下四种安全特性:

- 与某台服务器连接后, 利用 SSH 客户端可以确认在此后的连接都是与同一台服务器进行连接。
- 客户机使用强大的 128 位的加密传输方式, 向服务器端传输其认证信息。

- 收发所有数据都是使用 128 位加密方式，即使传输内容被截获也很难被解密和阅读。
- 客户端可从服务器转发 X11 应用程序。这一技术被称为 x11 转发，为图形化的网络应用提供了安全的方式。



X11 是指 X11R6.7 窗口显示系统，传统上被称为 X Window 系统或包含 xfree86 和开源的 X Window System 的一个开源 X Window 系统。

因为 ssh 协议为所有发送和接收的数据都进行加密，这样可以保证安全传输。利用一种被称为端口转发的技术，ssh 服务器可以变成一个管道，以确保其安全，就像 POP 协议，从而增强系统和数据整体安全性能。

7.3.2 为何使用SSH

计算机入侵者有各种各样的能让他们扰乱、拦截，并努力重建网络路由从而成功访问系统的工具。总的来讲，这些威胁可归纳如下：

- 截获两个系统间的通讯——在这种情况下，攻击者可以在网络通信实体的某处，在它们之间拷贝任何信息。攻击者可以截取信息，或者改变信息内容再发送给接收端。

这种攻击可以通过一种普通的网络小具——包嗅探器来实现。

- 冒充某一主机——使用这个策略，攻击者的系统将被配置成伪装的传输接收者。如果这一策略成功实现了，那么发送端是不会意识到此通信的错误所在。

这种攻击可以通过像 DNS 欺骗或 IP 欺骗等为人所知的手段来实现。

这两种技术用来拦截潜在的敏感信息，如果是恶意地截获，那么其结果也将是灾难性的。

如果使用 SSH 进行远程 shell 登录和文件拷贝，这些安全威胁将被大大减少。这是因为 ssh 客户端和服务端是采用数字签名的方式来核实其身份的。此外，所有在客户机和服务器系统间的沟通都是加密的。企图欺骗身份的通信方式将无法实行，因为每个数据包都进行加密传输，使用的密钥只有本地和远程系统才知道。



当入侵者攻击 dns 服务器时发生域名服务器中毒，客户端系统会被恶意地重复攻击。



当入侵者发送一些看似来自可信任的主机上的网络数据包时，ip 欺骗就发生了。

7.3.3 OpenSSH配置文件

OpenSSH 有两个不同的配置文件：一个为客户端程序（ssh，scp 及 sftp），另一个为服务器守护进程（sshd）。

System-wide SSH 配置信息存储在在 /etc/ssh/ 目录中：

- Moduli——包含用于 Diffie-Hellman 密钥交换的 Diffie-Hellman 组，其关键是构建一个安全的传输层。当建立 ssh 会话并进行密钥交换时，密钥不能单独地被任何一方建立。这个密钥用来为连接提供主机认证。

- ssh_config——全系统默认的 ssh 客户配置文件。
- sshd_config——sshd 守护进程的配置文件。
- ssh_host_dsa_key——sshd 守护进程的 DSA 私钥。
- ssh_host_dsa_key.pub——sshd 守护进程的 DSA 公钥。

- `ssh_host_key`——ssh 协议 1.0 的 `sshd` 守护进程的 RSA 私钥。
- `ssh_host_key.pub`——ssh 协议 1.0 的 `sshd` 守护进程的 RSA 公钥。
- `ssh_host_rsa_key`——ssh 协议 2.0 的 `sshd` 守护进程的 RSA 私钥。
- `ssh_host_rsa_key.pub`——ssh 协议 2.0 的 `sshd` 守护进程的 RSA 公钥。

User-specific SSH 配置信息被存放在用户的 `~/.ssh/` 目录下：

- `authorized_keys`——文件存放的是一个服务器的公钥认证名单。当客户端连接到服务器时，服务器用这个文件中所存放的签名公钥进行客户端的认证。
- `id_dsa`——包含用户的 SDA 私钥。
- `id_dsa.pub`——包含用户的 SDA 公钥。
- `id_rsa`——ssh 协议 2.0 的 ssh 的 RSA 私钥。
- `id_rsa.pub`——ssh 协议 2.0 的 ssh 的 RSA 公钥。
- `identity`——ssh 协议 1.0 的 ssh 的 RSA 私钥。
- `identity.pub`——ssh 协议 1.0 的 ssh 的 RSA 公钥。
- `known_hosts`——文件包含用于访问 SSH 服务器的 DSA 主机密码。这个文件对确保 ssh 客户端连接到正确的 ssh 服务器非常重要。



如果 ssh 服务器的主机密钥发生了变化，客户端将通知用户直到使用文本编辑器将服务器的主机密钥从 `known_hosts` 文件中删除后，才能进行连接。在此以前，请联系 SSH 服务器的系统管理员核实服务器未受攻击。

7.3.4 更安全的Shell

通过 SSH 命令可以实现与其他主机的图形会话，或进行网络传输，通过对端口的修改可避免不安全的网络连接。

7.3.4.1 Forwarding 转发

通过一个已建立的 SSH 连接打开图形会话就像在本地运行一个 x 程序一样容易。从安全 shell 提示符下运行一个 X 程序时，ssh 客户端与服务器间将建立一个新的安全通道，然后 x 程序的数据将利用该安全通道向客户机透明地传输。

7.3.4.2 Port Forwarding 端口转发

SSH 能够使 TCP/IP 协议端口转发变得安全。使用这一技术，对 SSH 客户端而言，SSH 服务器将变成一个的加密管道。

端口转发是将客户机本地端口映射到远程服务器端口上。SSH 可以从服务器上的任何端口映射到客户端的任意端口上，其端口号不必一致。

创建一个用来监听连接本地机的 TCP/IP 端口转发通道，使用以下命令：

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```



在 1024 以下的端口设置用于监听的转发端口，需要以 root 身份访问。

通过加密连接使用 pop3 协议到邮件服务器 `mail.example.com` 上检查电子邮件，可使用下列命令：

```
ssh -L 1100:mail.example.com:110 mail.example.com
```

一旦端口转发通道在客户机和邮件服务器间建立起来,直接在主机上利用 1100 端口在 POP3 邮件客户端上查收新邮件。发送的任何请求都将通过客户端系统的 1100 端口,直接安全地发送到 mail.example.com 服务器上。

如果 mail.example.com 没有运行 ssh 服务器,但同一网络上的另一台机器运行了 ssh 服务器,ssh 仍能确保其部分连接的安全。不过,在命令的使用上稍有不同:

```
ssh -L 1100:mail.example.com:110 other.example.com
```

在这个例子中,POP3 会从客户机的 1100 端口通过 SSH 连接,转发到 SSH 服务器 other.example.com 的 22 端口上。然后,other.example.com 连接到 mail.example.com 的 110 端口上检查新邮件。请注意:当采用这种技术时,只有客户端系统和 other.example.com 上 ssh 服务器之间的连接是安全的。

端口转发也可以用来穿网络防火墙,从而获得可靠的信息。如果防火墙被配置成允许通过标准端口(22)但屏蔽其他端口,通过已建立的 SSH 连接,在两台主机间仍能使用屏蔽了的端口重新连接。



利用端口转发技术来转发连接,这种方式允许客户端系统上的任何用户连接到服务器上。如果客户端系统泄露,攻击者也能访问转发到服务器。

系统管理员可以将端口转发功能禁用掉,在服务器的/etc/ssh/sshd_config 文件中的 AllowTcpForwarding 一行去掉其参数,然后重启 sshd 服务即可。

7.3.4.3 基于ssh的远程连接

为了使 SSH 真正安全,应禁止使用如 telnet、ftp 等的不安全连接协议。否则,受 SSH 连接保护的用户密码,在使用 telnet 时,马上就会被捕获到。

被禁用的一些服务包括:

- telnet
- rsh
- rlogin
- vsftpd

在系统中要禁止不安全的连接,请在命令行使用 chkconfig 命令,基于 ncurses 的应用程序 ntsysv,或运行图形应用程序 Services Configuration Tool。使用这些工具都要以 root 用户身份访问。

7.4 Nmap扫描工具

Red Flag Asianux Server 3 中提供了一个网络探索工具和安全扫描器——Nmap。

扫描器是一种自动检测远程或本地主机安全弱点的程序。使用扫描器,可以获得远程服务器的大量信息,通过这些信息,可以了解到远程主机存在的安全问题,从而能够及时修补系统的安全隐患。同时,扫描器也能够为攻击者提供很大的方便,可以大大简化他们的工作。

扫描器一般会先向远程 TCP/IP 端口发出请求,记录目标给予的回答,然后对应答信息进行分析。通过这种方法,可以搜集到目标主机的各种有用信息,比如,端口是否开放、提供的服务以及它们的软件版本、能否匿名登录等。



扫描器一般不是一个直接的网络漏洞攻击程序,它仅仅能帮助我们发现目标机器的某些内在的弱点。

7.4.1 Nmap简介

Nmap 是一个端口扫描工具，它提供了多种扫描技术以及多种高级功能。

Nmap 可以扫描一台主机，也可以扫描一个网段。要扫描一台主机，只要指定主机名或 IP 地址即可。要扫描网段，可以有几种表示方法，它们都是等效的：192.168.1.1/24，192.168.1.* 或 192.168.1.0-254。

用户可以选择 Nmap 工具扫描的网络类型，Nmap 可以扫描 TCP、UDP、TCP 同步，可以用 ping 命令检查主机是否开放等。Nmap 还可以扫描特定的一个或多个端口，例如，扫描端口号设为：20-30、80、6000- 可以扫描端口号为 20-30、80 及 6000 以上的端口。

Nmap 的运行结果是扫描的主机（主机组）的端口列表，通常给出端口号，服务名，状态等几项。

本文全方位地介绍 Nmap 的使用方法，网络管理员可以借助它了解自己的站点，发现网站的安全漏洞，并逐步完善自己的系统。Nmap 是网络安全管理必不可少的工具。



如果想了解更多关于 Nmap 的信息和高级功能，请访问它的主页 <http://www.insecure.org/nmap/>。

7.4.2 Nmap使用说明

Nmap 命令的语法相当简单。Nmap 的不同选项和 -s 标志组成了不同的扫描类型，比如：一个 Ping-scan 命令就是“-sP”。在确定了目标主机和网络之后，即可进行扫描。如果以 root 权限来运行 Nmap，它的功能会大大的增强，因为超级用户可以创建便于 Nmap 利用的定制数据包。

Nmap 的命令的语法格式如下：

nmap [扫描类型] [选项] 主机或网络

7.4.2.1 描述

Nmap 的设计初衷在于允许系统管理员查看一个大的网络系统有哪些主机以及其上运行何种服务。它支持多种协议的扫描，例如：UDP、TCP connect ()、TCP SYN (half open)、ftp proxy (bounce attack)、Reverse-ident、ICMP (ping sweep)、FIN、ACK sweep、Xmas Tree、SYN sweep 和 Null 扫描。

Nmap 还提供了一些高级功能，例如通过 TCP/IP 来甄别操作系统类型、秘密扫描、动态延迟和重发、平行扫描、通过并行的 Ping 侦测下属的主机、欺骗扫描、端口过滤探测、直接的 RPC 扫描、分布扫描、灵活的目标选择以及端口的描述。

运行 Nmap 后通常会得到一个关于被扫描主机的实用端口列表。Nmap 总是显示该服务的服务名称，端口号，状态以及协议。状态有 open, filtered 和 unfiltered 三种。Open 指的是目标机器将会在该端口接受您的连接请求；filtered 指的是有防火墙、过滤装置或者其它的网络障碍物在这个端口阻挡了 Nmap 进一步查明端口是否开放的动作；unfiltered 则只有在大多数的扫描端口都处在 filtered 状态下才会出现。

在利用一些参数和选项后，Nmap 还可以报告远程主机的如下特性：使用的系统、TCP 连续性、在各端口上使用的应用程序用户的用户名、DNS 名称、主机是否是一个 smurf 地址以及一些其它功能。

7.4.2.2 选项和参数

以下选项通常都是可以组合使用的。使用参数能够精确地定义一个扫描模式。Nmap 会对不规范的参数组合做出提示。



如果急于开始, 可以跳到 [第 7.5.3 节: Nmap 使用示例](#), 那里提供了 Nmap 最基本的使用方法范例。

➤ 扫描类型

-sT

TCP connect () 扫描: 这是对 TCP 的最基本的侦测形式。该 connect () 对目标主机上相应的端口进行试探, 如果端口正在监听, 则连接成功, 否则这个端口无法到达。该技术的优势在于您无须任何特殊权限, 大多数 UNIX 系统下这个命令可以被任何用户自由地使用。

这种形式的探测很容易被目标主机察觉并记录下来。它将会向 accept () 连接的服务显示一组连接及错误消息, 并使其立刻被关闭。

-sS

TCP SYN 扫描: 这种技术不需要打开一个完整 TCP 连接, 通常被认为是一种“半开”式的扫描。您发送一个 SYN 信息包, 就像要打开一个真正的连接而且正在等待对方的回应。一个 SYN|ACK (应答) 会表明该端口是开放监听的, 而一个 RST 则代表该端口未被监听。如果 SYN|ACK 回应返回, 则会马上发送一个 RST 包来中断这个连接。这种扫描的最大好处是只有极少的站点会对它做出记录, 但是需要有 root 权限来定制这些 SYN 包。

-sF -sX -sN

Stealth FIN、Xmas Tree 或者 Null 扫描模式: 有些时候, 即使是 SYN 扫描也不够隐蔽。一些防火墙和包过滤器会在重要端口守护, 将 SYN 被截获。一些应用软件如 Synlogger 以及 Courtney 对侦测此类扫描都是行家。所以, 要有更进一步的扫描能在不遇到麻烦的情况下通过它们。

新的想法是关闭的端口会对您发送的探测信息包返回一个 RST, 而打开的端口则对其忽略不理 (参阅 RFC 793pp64)。FIN 扫描使用一个空的 FIN 信息包作为探测器; Xmas tree 扫描则使用 FIN、URG 和 PUSH 标记; Null 扫描不使用任何标记。但不幸的是, Microsoft 决定完全忽略这一标准, 这样一来, 这一扫描类型在 WINDOWS 9X 以及 NT 下不能工作。

从积极的一面看, 这其实也是一个很好的区分两种平台的办法。如果扫描发现了打开的端口, 那就能知道这台机器不是运行 WINDOWS。如果 -sF, -sX, -sN 的扫描显示所有端口都是关闭的, 但一个 SYN (-sS) 扫描却显示有打开端口, 那就能大致推断它是 WINDOWS 平台。

这只是一个简单应用, 现在 Nmap 已经有了更彻底的操作系统判别方法 (当然它的原理类似上面所提到的), 这些平台包括 Cisco、BSDI、HP/UX、MVS 和 IRIX。

-sP

Ping 扫描: 有时只是想知道网络上有哪些主机是开放的, Nmap 可以通过向指定的 IP 地址发送 ICMP echo request 信息包做到这一点, 有回应的主机就是开放的。

但是, 一些站点例如 microsoft.com 对 echo request 包设置了障碍。这样的话, Nmap 还可以发送一个 TCPack 包到 80 端口, 如果获得了 RST 返回, 表示机器是开放的; 第三个方法是发送一个 SYN 信息包并等待 RST 或 SYN/ACK 响应了。这是非 root 的用户可以使用的。

对于 root 用户来说，默认的 Nmap 同时使用 ICMP 和 ACK 方法扫描。当然也可以更改 -P 选项。最好先 ping 一下主机，只有有回应的主机才有必要扫描，你可以在做实际深入扫描前先大面积地搜索一下活动的主机。

-sU

UDP 扫描：这一方法用来确定主机上哪个 UDP（User Datagram Protocol，RFC 768）端口是打开的。该技术将发送 0 字节的 UDP 包到目标计算机的各个端口，如果我们收到一个 ICMP 端口不可到达的回应，那么该端口就是关闭的，否则我们认为它是打开的。

但是，防火墙经常阻碍端口不可到达消息，从而导致端口看起来像打开的。有时，一个 ISP 只阻碍几个特殊的危险端口，例如 31337（后门）和 139（Windows NetBIOS），使这些易受攻击的端口看起来仿佛是打开的。不幸的是，区分真正打开的 UDP 端口和这些被主动过滤的并不总是很容易的。

有些人认为 UDP 扫描是没有意义的。我总是提醒他们记住最近的 Solaris rcpbind 漏洞。Rcpbind 会隐藏在某个 32770 之上的非正式 UDP 端口中，因此对 111 进行防火墙过滤是无关紧要的。但是您是否能发现过 30,000 以上的端口正处于监听状态？用 UDP 扫描就能轻松地做到这一点！再想想 cDc 出品的 Back Orifice 木马，它可以在 Windows 计算机中配置一个 UDP 端口，更不用说如此众多可以利用 UDP 的、易受攻击的服务了，例如 snmp、tftp、NFS 等等。

但不得不提及的是，由于大部分主机按照 RFC 1812 执行一个限制 ICMP 错误信息率的建议，UDP 扫描有时候会非常慢。例如：Linux 核心程序（在 net/ipv4/icmp.h）限制了每 4 秒产生 80 次的无法到达信息（每次产生 1/4 秒的延迟）；Solaris 有更严格的限制条件（大约每秒两次就会延迟），所以会耗费相当长的扫描时间。Nmap 会探测到这一限制并相应减缓速度，而不是以无用的将会被目标计算机忽略的大量信息包来填充网络。

Microsoft 忽视 RFC 的建议，而且看起来没有在 Win95 和 NT 计算机上做任何比率限制。这样，我们就把 Windows 计算机上多达 65K 的端口以极高的速度扫描完毕！

-sO

IP 协议扫描：该方法被用来确定一个主机支持的 IP 协议。这种技术将 raw IP 包（不包含任何协议头）发送到目标计算机的每一个指定协议。如果我们收到了一个 ICMP 协议不可到达消息，就表示该协议没有被使用。否则，我们假定它是打开的。请注意：有一些主机（AIX，HP-UX，Digital UNIX）和防火墙也许不发送协议不可到达消息。这将会导致所有协议看起来都是“打开的”。

因为被执行的技术非常类似于 UDP 端口扫描，ICMP 比率限制也许会适用。但是 IP 协议域只有 8 位，因此在大部分 256 协议可以在合理的时间内被探测到。

-sI <zombie host[:probeport]>

Idlescan：这种高级扫描方法允许对目标计算机上一个真正隐蔽的 TCP 端口进行扫描（这意味着没有包从您的真正 IP 地址发送到目标主机）。相反，一个唯一的边信道通路攻击会在 zombie 主机上使用可预言的“IP 分段 ID”序列搜集关于目标机器开放端口的信息。IDS 系统将会显示扫描仿佛来自于指定的 zombie 计算机（必须是性能良好的并且满足某种特定标准）。

除格外隐蔽外，该扫描类型允许映射两台机器之间以 IP 为基础的信任关系。端口列表从 zombie 主机图中显示打开的端口。因此，可以尝试通过使用你认为可以被信任的不同 zombies 扫描一个目标计算机（通过路由器/包过滤器规则）。显然当区分攻击目标时，这些信息很重要。否则，您的探测也许不得不耗费

相当多的资源来“拥有”一个“中间的”系统，结果只是发现其 IP 甚至不能被您最终跟随的目标主机/网络信任。

如果希望在 zombie 主机上为 IPID 更改探测一个特殊的端口，可以在一个端口数字后添加一个冒号。否则，Nmap 会使用“tcp ping”的默认端口。

-sA

ACK 扫描: 这种先进的方法通常被用来映射防火墙规则集。特别是它能帮助确定防火墙是静态的或只是一个阻碍 SYN 包进入的简单包过滤器。

该扫描类型发送一个 ACK 包到指定端口。如果一个 RST 返回，该端口就被分类为“unfiltered”；如果没有任何返回，或一个 ICMP 不可到达信息被返回，端口就被分类为“filtered”。请注意：Nmap 通常不打印“unfiltered”端口，因此在输出中得到无端口显示通常意味着所有探测器已经通过（并且返回 RST）。很明显，该扫描将永远不会以“open”状态显示端口。

-sW

Window 扫描: 除了有时可以检测打开端口，以及由于在 TCP 窗口被一些操作系统报告出现一个异常而导致的 filtered/nonfiltered，这种先进的扫描在其它方面非常类似于 ACK 扫描。易受攻击的系统至少包括 AIX, Amiga, BeOS/BSDI, Cray, Tru64 UNIX, DG/UX, OpenVMS, Digital UNIX, FreeBSD, HP-UX, OS/2, IRIX, MacOS, NetBSD, OpenBSD, OpenStep, QNX, Rhapsody, SunOS 4, Ultrix, VAX, 和 VxWorks 的一些版本。

-sR

RPC 扫描: 这种方法是结合 nmap 多种扫描的一种模式。它取得所有 TCP/UDP 开放端口，并且用 SunRPC 程序 NULL 命令将其填满，目的是试图确定它们是否是 RPC 端口，如果是，其上运行什么程序，何种版本。这样，即使目标主机躲在防火墙后或者由 TCP wrappers 防护着，它都能够获得效果近似于‘rpcinfo -p’的信息。但 Decoys 现在还不能正常工作在 RPC 扫描下，以后也许会在 UDP RPC 扫描中加入 decoy 支持。

-sL

列表扫描: 这种方法简单生成并打印一个 IP/名称 列表，没有执行实际的 ping 或端口扫描。除非使用 -n，否则 DNS 名称解析不会被执行。

-b <ftp relay host>

FTP bounce 攻击: FTP 协议的一个有趣的特点是它支持代理 FTP 连接（RFC 959）。换句话说，我可以从 evil.com 连接到一个 FTP 服务器 target.com 并且要求目标主机发送文件到 Internet 的*任何地方*！1985 年这一 RFC 被写下来后，这一特性便渐渐施行。但是在今天的网络中，我们不允许人们随意“劫持”ftp 服务器并任意攻击他人。Hobbit 于 1995 年写下有关这一协议的缺陷时说到：“它可以用来从许多站台上发出事实上难以追查的信件、新闻以及攻击性行为；填充你的硬盘，试图使防火墙失效或者更多烦人而

无意义的骚扰”。我们使用它是为了从一个代理 FTP 服务器扫描 TCP 端口，这样就可以连上一个防火墙后的 FTP 服务器，然后扫描有可能被阻碍的端口（139 是很好的例子）。如果 FTP 服务器允许读取并且写入一些目录（例如 `/incoming`），就可以将任意数据发送到你发现被打开的端口（尽管 `nmap` 不会这样做）。

通过“b”选项来使主机成为你的代理时，标准的 URL 格式是：`username:password@server:port`，除服务器之外的任何内容都是可选的。

➤ 一般选项

这些选项并非必需的，但是其中一些会非常实用。

-P0

在扫描前不尝试 PING 主机，用来扫描那些不允许 ICMP echo 请求（或应答）的主机或网络。`microsoft.com` 就是一个例子，必须使用 `-P0` 或者 `-PT80` 来扫描 `microsoft.com` 的端口。

-PT [portlist]

用 TCP 的 ping 来确定主机是否打开，而不是发送 ICMP echo 请求包并等待回应的方式，我们可以向目标网络（或者单机）大量发送 TCP ACK 包并一点点地等待它的回应，打开的主机会返回一个 RST。该选项可以让你在 ping 信息包阻塞时仍能高效率地扫描一个网络/主机。对于非 root 用户，我们用 `connect`（），以如下格式设置目标探测器：`-PT<port1>[,port2][...]`，默认端口是 80。这是由于该端口通常不被过滤。请注意，该选项可以接受多个由逗号分隔的端口数字。

-PS [portlist]

这一选项由 root 用户使用，用 SYN（连接请求）包代替 ACK 包。打开的主机会有一个 RST（或者 SYN|ACK，但比较少见）应答。您能够以和上述 `-PT` 一样的方式设置目的端口。

-PU [portlist]

该选项向指定的主机发送 UDP 探测。如果主机运行状态良好，就会得到一个 ICMP 端口不可到达信息包（或者如果端口是打开的，也可能是一个 UDP 应答）。由于许多 UDP 服务不会对一个空包做应答，最好认为是将这个空包发送给了预期关闭的端口，而不是打开的端口。

-PE

该选项使用一个真正 ping（ICMP echo 请求）包。它找到开放的主机并且将该子网中的广播地址全数搜寻——该广播地址是能够到达并能正确解析 IP 包的。如果发现它们允许大量拒绝的服务攻击（Smurf 是最常见的），就应该被删除。

-PP

使用一个 ICMP timestamp 请求（编码 13）查找监听主机。

-PM

除了使用网络掩码请求（ICMP 编码 17）外，其它方面同 **-PE** 和 **-PP** 类似。

-PB

默认 ping 类型。它使用 ACK（**-PT**）和 ICMP echo 请求（**-PE**）并行攻击。这种方式可以通过包过滤或防火墙。TCP 探测器目的端口能够以和上述 **-PT** 模式一样的方式被设置。请注意，因为 pingtype 标记现在能够结合起来使用，该标记被反对。因此，您应该使用“**PE**”和“**PT**”来获得相同的效果。

-O

该选项通过 TCP/IP 指纹识别判别远程主机的 OS 类型。换句话说，就是用一连串的信息包探测出所扫描的主机位于操作系统有关堆栈信息并区分其精细差别，以此判别操作系统。它使用搜集到的信息建立一个“指纹”，该指纹用来同已知的 OS 指纹识别（nmap-os-fingerprints 文件）数据库进行比较，这样判定操作系统就有了依据。

如果 Nmap 不能推测计算机的 OS，但有良好的条件（例如至少有一个端口是打开的），这时如果您确切知道 OS 在计算机上运行，Nmap 会提供一个您可以使用的 URL 来递交指纹识别。这样做可以为 Nmap 知晓的操作系统识别库提供有用的信息，这样一来，该功能就会更准确。请注意，如果您在表格中留下了一个 IP 地址，当我们加上指纹识别（以确认其在运行）时，计算机也许会被扫描。

-O 选项还能启用几种其它的测试。一种测试是“正常运行时间”测量，该测试使用 TCP timestamp 选项（RFC 1323）推测一台计算机最后一次重新启动是什么时候。该信息仅向提供该信息的计算机报告。

另外一种被 **-O** 启用的测试是 TCP 序列可预言性分类。这种测量近似的描绘建立对抗远程主机的一个伪造 TCP 连接如何困难。这对于以源 IP 为基础的信任关系（远程登录命令、防火墙过滤器等等）或是隐藏一个攻击源是很有用的。

当冗杂模式（**-v**）同 **-O** 一起开启时，IPID 序列也会被报告。大部分计算机在“incremental”类中，意思是它们在 IP 头中为其发送的每一个包增加“ID”域。这会使它们易受到几种先进的信息搜集和哄骗的攻击。

-6

该选项启用 IPv6 支持。如果使用该选项，所有目标主机都必须是 IPv6，而且它们可以通过常规 DNS 名称（AAAA 记录）被指定，或者也可以作为一个字面上的 IP 地址，例如 3ffe:501:4819:2000:210:f3ff:fe03:4d0 被指定。现在，connect（）TCP 扫描和 TCP connect（）Ping 扫描被支持。

-I

这会开启 TCP 反向 ident 扫描。正如 Dave Goldsmith 在 1996 年的 Bugtraq 所言，ident 协议（rfc 1413）允许通过 TCP 连接得到拥有进程的用户名——即使这个连接不是由该进程发起的。因此你可以连接到 http 端口，然后使用 identd 确认服务器是否在以 root 权限运行。这只能通过一个到目标端口的完整 TCP

连接完成（例如，-sT 扫描选项）。使用-I 选项，远程主机的 `identd` 在开放的端口接受连接查询，很显然，如果主机没有运行 `identd`，它将无法正常工作。

-f

该选项配置以细小的 IP 碎片包实现 SYN, FIN, XMAS 或 NULL 扫描请求。这一想法是将 TCP 头分别放在几个不同的信息包中，使包过滤器和入侵检测难于动作，而后就可以闯入系统做你想做的事了。但要注意，有些程序处理这些微小包有困难。比如我最喜欢的嗅探器分段在收到第一个 36 字节的信息碎片时就出现故障。之后，又来了一个 24 字节的！当包过滤器和能将 IP 碎片排列的防火墙没有获得此顺序时（就象 linux 内核中的 `CON-FIG_IP_ALWAYS_DEFRAG` 选项），一些网络系统就不能反映出找到目标，并且放弃用。

记住这个参数不一定能很好地工作在任何系统上，它在我的 Linux、FreeBSD 以及 OpenBSD 下是正常的，当然也有一些人说它能在部分不同的 *NIX 环境下工作。

-v

详细模式。这是被强烈推荐选项。它可以公布正在进行的操作的更多信息。你可以重复使用它以获得更大效果。如果你需要大量翻动屏幕请使用-d 命令两次。

-h

一个快捷的帮助选项，可以在屏幕上显示 `nmap` 的使用方法。正如您注意到的，这个 man page 并不是一个“快速入门参考”。

-oN <logfilename>

该选项用来指定一个记录扫描结果的文件，这个结果是便于阅读的。

-oX <logfilename>

该选项会以 XML 格式将扫描结果记录到指定的文件中，这会允许程序方便地捕捉并对 Nmap 结果进行说明。您可以给出变量“-”将输出发送到 stdout（shell 管道等等）。这种情况下常规输出会被抑制。如果使用该选项，请密切注意错误消息。同时也请注意“-v”也许会导致一些额外的消息被打印。定义 XML 输出结构的 DTD 在 <http://www.insecure.org/nmap/nmap.dtd> 中可以得到。

oG <logfilename>

该选项以 `grepable` 的格式将扫描结果记录到指定的文件中。这种简单的格式在行中提供了所有信息（这样就可以容易的用 `grep` 命令查找端口或是 OS 的信息，并且看到所有 IP）。这通常是和 Nmap 相互作用的程序首选的机制，但是现在我们推荐使用 XML 输出（-oX）。这一简单的格式包含的信息也许并不像其它格式那么多。您可以给出变量“-”将输出发送到 stdout 中（shell 管道等等）。在这种情况下常规

输出将会被抑制。如果使用该选项，请密切注意错误消息。同时也请注意“-v”也许会导致一些额外的消息被打印。

-oA <basefilename>

该选项会告诉 Nmap 记录所有主要格式（常规、grepable 和 XML）。可以为文件名给出一个 base，输出文件就会是 base.nmap，base.gnmap 和 base.xml。

--resume <logfilename>

如果网络扫描由于 control-C 或网络中断等情况而被取消，可以用此选项继续。失败扫描的记录文件必须是一个常规（-oN）记录或 grepable（-oG）记录。不会给出其它选项（它们同失败的扫描一样）。Nmap 会从日志文件中最后一个被成功扫描的记录开始启动扫描。

--append_output

将 Nmap 扫描结果附加到任何指定的输出文件中，而不是重写那些文件。

-iL <inputfilename>

从指定的文件中而不是从命令行中读取数据。该文件可以存放一个主机或网络的列表，中间用空格、TAB 键或者换行来分隔。如果希望从标准输入设备（文件）读取——比如在管道符的末端，你要将连字号（-）用于文件名。请参阅 [目标规范](#) 部分得到有关使用表达式填充的文件的更多信息。

-iR <num hosts>

该选项告诉 Nmap 通过简单的采集任意数字生成自己的主机进行扫描。在给定的 IP 被扫描后——对一个从不终止的扫描使用 0 后，该操作永远不会结束。该选项对于 Internet 估计各种事情的统计样本是很有用的。如果感到厌倦，请试试 `nmap -sS -PS80 -iR 0 -p 80` 寻找一些网络服务器察看。

-p < 端口范围 >

指定你希望扫描的端口。例如，“-p 23”只会对目标主机上的 23 端口进行探测。“-p 20-30,139,60000-”会扫描 20 到 30、139 以及所有大于 60000 的端口。默认扫描的是从 1 到 1024 的端口，或者是 nmap 的 services file 里列出的端口。对于 IP 协议扫描（-sO）来说，该选项会指定您希望扫描的协议数字（0-255）。当扫描 TCP 和 UDP 端口时，可以通过“T:”或“U:”前述端口数字来指定一个特定协议。限定符会持续到指定了另一个限定符为止。例如，变量“-p U:53,111,137,T:21-25,80,139,8080”将会扫描 UDP 端口 53, 111 和 137，以及列出的 TCP 端口。请注意，为了对 UDP 和 TCP 进行扫描，需要指定-sU 和至少一个 TCP 扫描类型（例如 -sS, -sF 或 -sT）。如果没有指定协议限制符，端口数字会被加入所有协议列表中。

-F 快速扫描模式

指定只希望对 nmap 中提供的 services file（或者 -sO 的协议文件）中列出的端口进行扫描。这明显会比扫描所有 65535 个端口快很多。

-D <decoy1 [,decoy2][,ME],...>

这是一种带有诱骗模式的扫描，在远程主机的连接记录里会记下所有你所指定的诱骗性的地址。这样的话他们的数据存储器会显示有一些端口扫描从某个 IP 发起，然而他们无法辨别哪个是真正的 IP 而哪个是用来作为掩护的，这可以击败一些通过路由进行跟踪的行为，所以它是一项隐藏你的 IP 的很实用的技术。

使用逗号分隔各个欺骗地址，可以随意地将“ME”放进任意一个你希望显示真实 IP 的地方，如果你将“ME”放在第六位甚至最后，有些端口扫描记录器（比如 Solar Designer's excellent scanlogd）可能根本就不会显示你的 IP，如果你不用“ME”的话，nmap 将会将它随机放置。

记住你用来诱骗的主机必须是开放的或者你可以半开扫描一下你的目标。因为要从一堆实际上没有用的 IP 地址里判别出哪个是真正的入侵者是相当容易的。你还可能要用 IP 地址来代替名字，这样诱骗主机的 name server logs 里才不会记录你。

还要记得有些（愚蠢的）“端口扫描探测器”会拒绝到达主机的端口扫描尝试。这样无意中就会导致你扫描的主机与“诱骗主机”连接的丢失，可能会带来一个很大的问题是——如果这个“诱骗主机”是一个网上的网关或者甚至就是其本地的机子，其连接一样会断开！所以大家最好小心使用这个参数。这种诱骗可以用在最初的 ping 扫描（用 ICMP、SYN、ACK 或其它）与实际的端口状态扫描中，它还可以用于远程 OS 的判别（-O）。

当然如果你写入太多的诱骗地址也是没什么用处的，那只能减缓扫描速度以及降低一些精确度。而且一些指令处理系统还可能会过滤掉你的欺骗包，虽然多数（几乎是全部了）不会对欺骗包做出任何限制。

-S <IP_Address>

在某些环境下 nmap 可能无法确定你的源地址——这种情况下 nmap 会有提示，这时你就要用 -S 带 IP 地址来标注。

另一种使用的可能性是用来欺骗目标使它认为某人在扫描它。设想一下，某个公司发现被竞争者持续不断的扫描，这是一个不被支持的用法，或者说不是主要目的。我只是用它来提醒人们在发现一个端口扫描者时别光顾责难，可能他是无辜的呢。-e 能够说明这个参数的一般用法。

-e <interface>

告诉 nmap 用哪个接口要发送或接收包。nmap 能够自动探测它，如果无法做到，会有提示信息。

-g <portnumber>

设置在扫描中使用的源端口号。许多“天真的”防火墙和包过滤器除了它们建立的允许 DNS（53）或 FTP-DATA（20）的包进来建立连接之外，其余一概过滤，显然这是很轻率的做法，因为入侵者能够轻易地编辑一个来自 FTP 或 DNS 的源端口。比如说，你如果无法从一个主机的 host:port 通过 TCP ISN 取

得信息，那么通过用 `-g` 命令，`nmap` 会改变源端口再次尝试。

请注意，这只是一个请求——`nmap` 只会当或只在能够这样做的时候允许这样做。例如，您不能进行所有从一个 `host:port` 到一个 `host:port` 的 TCP ISN 取样。因此，即使是使用了 `-g`，`nmap` 还是会更改源端口。需要了解的是，使用这个选项可能会有小小的延迟，因为有时需要在这些源端口号中存储一些有用的信息。

--data_length <number>

通常，`Nmap` 会发送一个仅有标题头的最低要求包。因此，它的 TCP 包通常是 40 字节，ICMP echo 请求则是 28 字节。该选项告诉 `Nmap` 向其发送的大部分包附加一个给定数字的 0 填满位。OS 探测 (`-O`) 包不会被影响，但是大部分 ping 和端口扫描包会被影响。这会导致速度变慢，但是并不十分明显。

-n

告诉 `Nmap` 从不要在其发现的活动 IP 地址中执行反向 DNS 解析。由于 DNS 通常比较慢，这会有助于速度的加快。

-R

告诉 `Nmap` 总是要在目标 IP 地址中执行反向 DNS 解析。通常，只有当一台计算机被发现是活动的时候才会这样做。

-r

告诉 `Nmap` 不要将被扫描的端口顺序任意排列。

-ttl <value>

在将包发送到给定的值中为活动域设置 IPv4 时间。

--randomize_hosts

告诉 `Nmap` 在开始扫描之前，将每个组搅乱到 2048 主机。这会使扫描在不同的网络监控系统中看起来不是非常明显，特别是将其同 `slow timing` 选项合并使用时。

-M <max sockets>

设定用来并行进行 TCP connect () 扫描的最大默认 sockets 数目。这会将扫描适当减缓从而避免使远程主机崩溃。另一个途径是用 `-sS`，这对于计算机来说通常比较容易处理。

--packet_trace

告诉 Nmap 以类似 tcpdump 的格式显示所有发送和接收的信息包。该选项对于调试来说是非常有用的，而且也是一个很好的学习工具。

--datadir [directoryname]

Nmap 运行时，可以从名称为 nmap-services, map-protocols, nmap-rpc 和 nmap-os-fingerprints 的文件中获得一些特殊数据。Nmap 会首先在目录选项中搜索这些文件。任何未在这里发现的文件会在由 NMAPDIR 环境变量指定的目录中被搜索；接下来是 ~/nmap，然后是程序被编译的位置，如 /usr/share/nmap；最后的方法是让 Nmap 在当前目录中搜索。

➤ 时间选项

虽然 Nmap 在一般情况下都能够很好地在运行时间内尽可能迅速地完成任务，但偶尔还是会有一些主机/端口无法侦测，这可能是 Nmap 默认的时间策略和你的目标不太一致，下面是对扫描的时间进行控制一些选项：

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane>

这是一个可以用来便利地表达 Nmap 时间策略优先权的参数设置。Paranoid 模式用极慢的速度来扫描以避免被数字记录系统监测，它使扫描连续而不是并发而且通常等待至少五分钟才发送一个信息包。Sneaky 也是类似的，只是它是每 15 秒发送一个信息包。Polite 模式是用来减轻网络负载以减少当机的可能性，它是连续发送探针并在两个包的间隙等待 0.4 秒。Normal 是 Nmap 的常规用法，是尽其所能地快速扫描——除非主机或端口连接丢失。Aggressive 模式是对每台主机设定了五分钟的 timeout，并且等待每个探针应答不超过 1.25 秒。Insane 模式是适应非常快的网络或者你不在乎丢失一些信息，它的 timeout 设定于 75 秒并且只等待回应 0.3 秒，它允许对一个很快的网络系统进行“扫荡”。

你也可以用数字（0-5）来代表参数，例如：“-T0”表示 Paranoid 模式，而“-T5”代表 Insane 模式。

--host_timeout <milliseconds>

具体指定 Nmap 对某个 IP 的扫描时间总量，超过则作放弃处理，默认是不做设定。

--max_rtt_timeout <milliseconds>

指定 Nmap 对一个探测器从远程目标返回回应的最大时间，默认是 9000。

--min_rtt_timeout <milliseconds>

当目标主机很快开始建立一个应答模式，Nmap 将会缩短给予每个探测器的时间量。这会加速扫描的速度，但是在一个应答花费比一般时间更多时，会导致包丢失。使用这个参数，可以确保 Nmap 在一个探测器上放弃之前，将至少等待给定的时间量。

--initial_rtt_timeout <milliseconds>

指定最初探测器的 timeout 时间，通常在用 -P0 扫描有防火墙保护的主机时很有效，Nmap 会通过 ping 以及最初的探测器信息得到一个很好的 RTT 评估。默认值为 6000。

--max_parallelism <number>

指定 nmap 允许的最大并行扫描数目，设定为 1 表明 Nmap 每次只扫描一个端口，它同样会对其它扫描如 ping sweep，RPC 等产生影响。

--min_parallelism <number>

告诉 Nmap 至少对以一个平行方式给定的端口数目进行扫描。这可以通过一个量值命令加速攻破具有某种特定防火墙主机的扫描的速度，但是请小心，如果您让它进行的太快，结果会变得不可靠。

--scan_delay <milliseconds>

指定 Nmap 必须等待的两个探测间的最小时间。这是减少网络负载以及使扫描在综合数据存储的记录下不那么显眼的有效方法。

➤ 目标规范

所有不带参数的选项都会被视为是 nmap 的目标主机描述。最简单的例子是仅仅在命令行列出单一的主机名或 IP 地址。如果你希望扫描一个子网，可以在主机名或 IP 地址中加入 “/mask”。Mask 必需是在 0（扫描整个网络）和 32（特定的单一主机）之间。用/24 扫描一个 C 类地址，而/16 则是扫描 B 类地址。

Nmap 还有一些更有用的符号用于指定各类网络地址。比如要扫描某 B 类网址，则可以用“128.210.*.*”或“128.210.0-255.0-255”甚至是“128.210.1-50,51-255.1,2,3,4,5-255”来表示。当然也可以 mask 来表示：“128.210.0.0/16”，这些都是等价的。

另一个有趣的用法是可以将整个网络“分割”，比如可以用“*.5.6-7”来扫描所有以.5.6 或.5.7 结束的 IP 地址。要得到更多信息，请参看下一节：[Nmap 使用示例](#)。

7.4.3 Nmap使用示例

由于 Nmap 的功能非常强大，选项繁多，想熟练使用需要一定的经验。下面，我们由浅入深地举几个例子说明如何使用 Nmap：

```
nmap -v target.example.com
```

扫描主机 target.example.com 上所有 TCP 端口。-v 表示使用详细模式。默认情况下，nmap 会使用 -sT 扫描方式。

```
nmap -sS -O target.example.com/24
```

开始一次 SYN 的半开扫描，针对的目标是 target.example.com 所在的 C 类子网，它还试图确定在其上运行的是什么系统。因为用到了半开扫描以及系统侦测，所以需要 root 权限。

```
nmap -sX -p 22,53,110,143,4564 198.116.*.1-127
```

发送一个 Xmas tree 扫描到 198.116 所在 B 类子网的一半范围内，我们将检测系统是否运行 sshd、DNS、pop3d、imapd 或者端口 4564。要注意，由于微软 TCP 堆栈的不完善，Xmas 扫描将不能在其平台上运行成功，同样的问题可能存在于 CISCO、IRIX、HP/UX 和 BSDI。

```
nmap -v - -randomize_hosts -p 80 '*.*.2.3-5'
```

这是定位一个网域（将整个网络分隔成许多小部份）再进行扫描的方式，这里扫描的是所有以 .2.3、.2.4 或 .2.5 结束的 IP 地址。如果是 root 身份的话也可以用 -sS。

```
host -l company.com | cut -d' ' -f 4 | ./nmap -v -iL -
```

用一个 DNS 域转换来寻找 company.com 中的主机并且将 IP 地址送至 nmap。

7.5 流量控制

Red Flag Asianux Server 3 系统中有一个成熟的带宽供给系统，称为 Traffic Control（流量控制），简称为 TC。流量控制支持以多种方式分类、排序、共享和限制出入流量。

7.5.1 简介

在 Linux 操作系统中，流量控制器（TC）主要是在输出端口处建立一个队列进行流量控制，控制的方式是基于路由，亦即基于目的 IP 地址或目的子网的网络号的流量控制。

TC 的基本功能模块为队列、类和过滤器。Linux 内核中支持的队列有：Class Based Queue, Token Bucket Flow, CSZ, First In First Out, Priority, TEQL, SFQ, ATM, RED。

这里我们讨论的队列与类都是基于 CBQ（Class Based Queue）的，而过滤器是基于路由（Route）的。

为了更好的使用 TC，下面说明 TC 中常用单位的规定。

➤ 带宽或者流速单位：

kpbs: 千字节 / 秒

mpps: 兆字节 / 秒

kbit: kbits / 秒

mbit: mbits / 秒

bps 或一个无单位数字: 字节 / 秒

➤ 数据的数量单位：

kb 或 k: 千字节

mb 或 m: 兆字节

mbit: 兆 bit

kbit: 千 bit

b 或一个无单位数字: 字节

➤ 时间的计量单位：

s、sec 或 secs: 秒

ms、msec 或 msec: 分钟

us、usec、usecs 或一个无单位数字：微秒

7.5.2 配置

配置和使用流量控制器 TC，主要分为以下几个方面：建立队列、建立类、建立过滤器和建立路由，另外还需要对现有的队列、类、过滤器和路由进行监视。

其基本使用步骤为：

- 1、 针对网络物理设备（如以太网卡 eth0）绑定一个 CBQ 队列；
- 2、 在该队列上建立类；
- 3、 为每一个类建立一个基于路由的过滤器；
- 4、 最后与过滤器相配合，建立特定的路由表。

下面以一个简单的环境为例进行说明，参见下图：

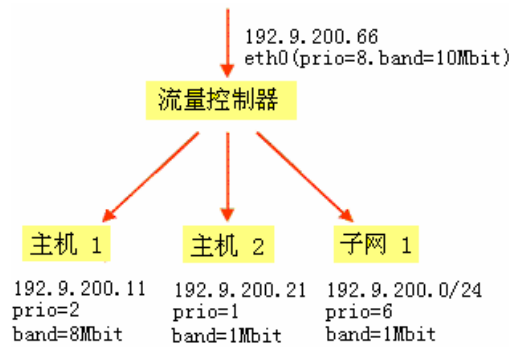


图1-1 流量控制示意图

流量控制器上的以太网卡（eth0）的 IP 地址为 192.9.200.66，在其上建立一个 CBQ 队列。假设包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，可接收冲突的发送最长包数目为 20 字节。

假如有三种类型的流量需要控制：

- 发往主机 1 的，其 IP 地址为 192.9.200.11。其流量带宽控制在 8Mbit，优先级为 2；
- 发往主机 2 的，其 IP 地址为 192.9.200.21。其流量带宽控制在 1Mbit，优先级为 1；
- 发往子网 1 的，其子网号为 192.9.200.0/24，子网掩码为 255.255.255.0。流量带宽控制在 1Mbit，优先级为 6。

7.5.2.1 建立队列

一般情况下，针对一个网卡只需建立一个队列。

将一个 cbq 队列绑定到网络物理接口 eth0 上，其编号为 1:0；网络物理接口 eth0 的实际带宽为 10 Mbit，包的平均大小为 1000 字节；包间隔发送单元的大小为 8 字节，最小传输包大小为 64 字节。

```
# tc qdisc add dev eth0 root handle 0:1 cbq bandwidth 10Mbit avpkt 1000 cell 8 mpu 64
```

7.5.2.2 建立类

类建立在队列之上。一般情况下，针对一个队列需建立一个根分类，然后在其上建立子类。类按其编

号顺序起作用，编号小的优先；一旦符合某个类匹配规则，则通过该类发送数据包，其后的类不再起作用。

- 1) 创建根分类 1:1；分配带宽为 10 Mbit，优先级别为 8。

```
# tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 10Mbit rate 10Mbit maxburst 20 allot 1514 prio 8 avpkt 1000 cell 8 weight 1Mbit
```

该队列的最大可用带宽为 10 Mbit，实际分配的带宽为 10 Mbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 8，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 1 Mbit。

- 2) 创建类 1:2，其父类为 1:1，分配带宽为 8 Mbit，优先级别为 2。

```
# tc class add dev eth0 parent 1:1 classid 1:2 cbq bandwidth 10Mbit rate 8Mbit maxburst 20 allot 1514 prio 2 avpkt 1000 cell 8 weight 800Kbit split 1:0 bounded
```

该队列的最大可用带宽为 10 Mbit，实际分配的带宽为 8Mbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 1，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 800 Kbit，分类的分离点为 1:0，且不可借用未使用带宽。

- 3) 创建类 1:3，其父类为 1:1，分配带宽为 1 Mbit，优先级别为 1。

```
# tc class add dev eth0 parent 1:1 classid 1:3 cbq bandwidth 10Mbit rate 1Mbit maxburst 20 allot 1514 prio 1 avpkt 1000 cell 8 weight 100Kbit split 1:0
```

该队列的最大可用带宽为 10 Mbit，实际分配的带宽为 1 Mbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 2，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 100 Kbit，分类的分离点为 1:0。

- 4) 创建类 1:4，其父类为 1:1，分配带宽为 1 Mbit，优先级别为 6。

```
# tc class add dev eth0 parent 1:1 classid 1:4 cbq bandwidth 10Mbit rate 1Mbit maxburst 20 allot 1514 prio 6 avpkt 1000 cell 8 weight 100Kbit split 1:0
```

该队列的最大可用带宽为 10 Mbit，实际分配的带宽为 64 Kbit，可接收冲突的发送最长包数目为 20 字节；最大传输单元加 MAC 头的大小为 1514 字节，优先级别为 1，包的平均大小为 1000 字节，包间隔发送单元的大小为 8 字节，相应于实际带宽的加权速率为 100 Kbit，分类的分离点为 1:0。

7.5.2.3 建立过滤器

过滤器主要服务于类。一般只需针对根分类提供一个过滤器，然后为每个子类提供路由映射。

- 1、应用路由分类器到 cbq 队列的根，父类编号为 1:0；过滤协议为 ip，优先级别为 100，过滤器为基于路由表。

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 100 route
```

- 2、建立路由映射类 1:2，1:3，1:4。

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 2 flowid 1:2
```

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 3 flowid 1:3
```

```
# tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 4 flowid 1:4
```

7.5.2.4 建立路由

该路由是与前面所建立的路由映射一一对应。

- 1、 发往主机 192.9.200.11 的数据包通过类 2 转发（类 2 的速率 8 Mbit）。
ip route add 192.9.200.11 dev eth0 via 192.9.200.66 realm 2
- 2、 发往主机 192.9.200.21 的数据包通过类 3 转发（类 3 的速率 1 Mbit）。
ip route add 192.9.200.21 dev eth0 via 192.9.200.66 realm 3
- 3、 发往子网 192.9.200.0/24 的数据包通过类 4 转发（类 4 的速率 1 Mbit）。
ip route add 192.9.200.0/24 dev eth0 via 192.9.200.66 realm 4



对流量控制器直接连接的网段建议使用 IP 主机地址流量控制限制，不要使用子网流量控制。如果必须对直接连接子网使用子网流量控制限制，则在建立该子网的路由映射前，需将原先由系统建立的路由删除，才可完成相应步骤。

7.5.2.5 监视

主要包括对现有队列、类、过滤器和路由的状况进行监视。

➤ 显示队列的状况

- 简单显示指定设备（eth0）的队列状况
tc qdisc ls dev eth0
qdisc cbq 1: rate 10Mbit (bounded,isolated) prio no-transmit
- 详细显示指定设备（eth0）的队列状况
tc -s qdisc ls dev eth0
qdisc cbq 1: rate 10Mbit (bounded,isolated) prio no-transmit
Sent 7646731 bytes 13232 pkts (dropped 0, overlimits 0)
borrowed 0 overactions 0 avgidle 31 undertime 0

这里主要显示了通过该队列发送了 13232 个数据包，数据流量为 7646731 个字节，丢弃的包数目为 0，超过速率限制的包数目为 0。

➤ 显示分类的状况

- 简单显示指定设备（eth0）的分类状况
tc class ls dev eth0
class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit
class cbq 1:1 parent 1: rate 10Mbit prio no-transmit #no-transmit 表示优先级为 8
class cbq 1:2 parent 1:1 rate 8Mbit prio 2
class cbq 1:3 parent 1:1 rate 1Mbit prio 1
CLASS CBQ 1:4 PARENT 1:1 RATE 1MBIT PRIO 6
- 详细显示指定设备（这里为 eth0）的分类状况
tc -s class ls dev eth0
class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit

```

Sent 17725304 bytes 32088 pkts (dropped 0, overlimits 0)
borrowed 0 overactions 0 avgidle 31 undertime 0
class cbq 1:1 parent 1: rate 10Mbit prio no-transmit
Sent 16627774 bytes 28884 pkts (dropped 0, overlimits 0)
borrowed 16163 overactions 0 avgidle 587 undertime 0
class cbq 1:2 parent 1:1 rate 8Mbit prio 2
Sent 628829 bytes 3130 pkts (dropped 0, overlimits 0)
borrowed 0 overactions 0 avgidle 4137 undertime 0
class cbq 1:3 parent 1:1 rate 1Mbit prio 1
Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
borrowed 0 overactions 0 avgidle 159654 undertime 0
class cbq 1:4 parent 1:1 rate 1Mbit prio 6
Sent 5552879 bytes 8076 pkts (dropped 0, overlimits 0)
borrowed 3797 overactions 0 avgidle 159557 undertime 0

```

这里主要显示了通过不同分类发送的数据包，数据流量，丢弃的包数目，超过速率限制的包数目等等。其中根分类（class cbq 1:0）的状况应与队列的状况类似。

例如，类 class cbq 1:4 发送了 8076 个数据包，数据流量为 5552879 个字节，丢弃的包数目为 0，超过速率限制的包数目为 0。

- 显示过滤器的状况

```
# tc -s filter ls dev eth0
```

```

filter parent 1: protocol ip pref 100 route
filter parent 1: protocol ip pref 100 route fh 0xffff0002 flowid 1:2 to 2
filter parent 1: protocol ip pref 100 route fh 0xffff0003 flowid 1:3 to 3
filter parent 1: protocol ip pref 100 route fh 0xffff0004 flowid 1:4 to 4

```

这里 flowid 1:2 代表类 class cbq 1:2，to 2 代表通过路由 2 发送。

- 显示现有路由的状况

```
# ip route
```

```

192.9.200.66 dev eth0 scope link
192.9.200.11 via 192.9.200.66 dev eth0 realm 2
202.102.24.216 dev ppp0 proto kernel scope link src 202.102.76.5
192.9.200.21 via 192.9.200.66 dev eth0 realm 3
192.9.200.0/24 via 192.9.200.66 dev eth0 realm 4
192.9.200.0/24 dev eth0 proto kernel scope link src 192.9.200.66
172.16.1.0/24 via 192.9.200.66 dev eth0 scope link
127.0.0.0/8 dev lo scope link

```



```
default via 202.102.24.216 dev ppp0
```

```
default via 192.9.200.254 dev eth0
```

如上所示，结尾包含有 `realm` 的显示行是起作用的路由过滤器。

7.5.2.6 维护

维护主要包括对队列、类、过滤器和路由的增添、修改和删除。

增添动作一般按照“队列→类→过滤器→路由”的顺序进行；修改动作没有什么要求；删除则按照“路由→过滤器→类→队列”的顺序进行。

➤ 队列的维护

对于一台流量控制器来说，出厂时通常针对每个以太网卡均已经配置好一个队列了，一般情况下，对队列无需进行增添、修改和删除动作了。

➤ 类的维护

● 增添

增添动作通过 `tc class add` 命令实现，如前面所示。

● 修改

修改动作通过 `tc class change` 命令实现：

```
# tc class change dev eth0 parent 1:1 classid 1:2 cbq bandwidth 10Mbit rate 7Mbit maxburst 20 allot
1514 prio 2 avpkt 1000 cell 8 weight 700Kbit split 1:0 bounded
```

对于 `bounded` 命令应慎用，一旦添加后就进行修改，只可通过删除后再添加来实现。

● 删除

删除动作只在该类没有工作前才可进行，一旦通过该类发送过数据，则无法删除它了。因此，需要通过 `shell` 方式来修改，通过重新启动来完成删除动作。

➤ 过滤器的维护

● 增添

增添动作通过 `tc filter add` 命令实现，如前面所述。

● 修改

修改动作通过 `tc filter change` 命令实现：

```
# tc filter change dev eth0 parent 1:0 protocol ip prio 100 route to 10 flowid 1:8
```

● 删除

删除动作通过 `tc filter del` 命令实现：

```
# tc filter del dev eth0 parent 1:0 protocol ip prio 100 route to 10
```

➤ 与过滤器——映射路由的维护

● 增添

增添动作通过 `ip route add` 命令实现，如前面所述。

- 修改

修改动作通过 `ip route change` 命令实现:

```
# ip route change 192.9.200.21 dev eth0 via 192.9.200.66 realm 8
```

- 删除

删除动作通过 `ip route del` 命令实现:

```
# ip route del 192.9.200.21 dev eth0 via 192.9.200.66 realm 8
```

```
# ip route del 192.9.200.0/24 dev eth0 via 192.9.200.66 realm 4
```



事实上, 构建于 2.6.18 核心之上的 Red Flag Asianux Server 3 完全能够像那些高端的专用带宽管理系统一样来管理带宽, 甚至比帧中继和 ATM 还要优秀。

现在, 很多人都没有用到这些高级功能。由于本手册并不是一本专门讲述该课题的教材, 有兴趣的用户可以访问 <http://lartc.org>。

附录

附录A 常见问题

本部分包括了一些在 Red Flag Asianux Server 3 系统管理过程中常见的问题，并给出它们的解决办法。

➤ 如何使ls不显示颜色

在/etc/bashrc 文件中删除 alias ls="ls --color"的那些语句，把用户目录下的.bashrc 文件也做此处理。

➤ 当键入ls后，大量的信息从屏幕上卷过，如何才能清晰地阅读输出

要防止 ls 命令的输出过快地从屏幕上卷过，可利用管道的方法。

也可以使用 less 来阅读/etc 的内容，在 shell 提示下键入下列命令：

```
ls -al /etc | less
```

使用另一个分页工具 more 来达到同样的效果。

➤ 怎样才能使root用户通过ssh远程登录

如须允许 root 用户直接登录 ssh，可通过修改 OpenSSH 的服务器端配置文件/etc/ssh/sshd_config 来实现。即将“PermitRootLogin no”一行中的 no 改为 yes 后，重启 sshd 服务即可。

➤ 为何基于Apache的httpd服务或Sendmail在启动时会被挂起

如果启动基于 Apache 的 httpd 服务或 Sendmail 时遇到问题，请确定/etc/hosts 文件中包括下面一行：

```
127.0.0.1 localhost.localdomain localhost
```

➤ 安装新应用程序时，当在shell下键入其名称，会得到“command not found”。为何不能启动应用程序

如果在 shell 下启动某个应用程序却无效，可以在应用程序的可执行名称前添加一个“./”。

假设您下载了 abcdef 客户应用程序，并在用户主目录中创建了一个 abc/的子目录。现在，除了使用该可执行文件的全路径来启动这个应用程序：

```
/home/xyz/abc/abcdef
```

还可以在/home/xyz/abc/下运行命令：**./abcdef**

之所以要使用全路径名来启动程序，是因为可执行文件的路径没有被放置在用户 shell 环境所知的目录中（如/usr/local/bin）。

➤ 如何在运行Linux时访问Windows分区

首先，需要知道 Windows 分区的位置，即明确将访问的 Windows 分区在哪个硬盘的哪个分区上。

然后，以超级用户身份执行以下命令：

```
mkdir /mnt/windows
```

 创建用于挂载 Windows 分区的目录

```
mount -t vfat /dev/hda1 /mnt/windows
```

 将它挂载到创建的目录中

```
cd /mnt/windows
```

 进入加载了 Windows 分区的目录

如果要在每次引导系统时自动挂载 Windows 分区，则须修改/etc/fstab 文件。在/etc/fstab 文件中添加如下一行：

/dev/hda1 /mnt/windows vfat auto,umask=0 0 0

在系统重新引导时，/etc/fstab 文件会被读取，Windows 分区会被自动挂载到目录/mnt/windows 中。

附录B Linux和DOS常用命令对照表

功 能	Linux	MS-DOS
复制文件	cp	copy
移动文件	mv	move
列举文件	ls	dir
清除屏幕	clear	cls
删除文件	rm	del
创建目录	mkdir	mkdir
查看文件	less	more
文件重命名	mv	ren
比较文件内容	diff	fc
查看当前路径	pwd	chdir
把输出回显到屏幕	echo	echo
在文件中寻找字符串	grep	find
显示命令帮助	man	命令/?
关闭和退出	exit	exit
显示或设置日期	date	date
显示时间	date	time
显示已被使用的内存	free	mem
格式化软盘	mke2fs 或 mformat	format

附录C 文件类型和文件名

在 Linux 系统中，文件名通常与它的文件类型相关，下面列出了常见文件名和它们所属文件类型的对应关系：

压缩和归档文件

.bz2	使用 bzip2 程序压缩的文件
.gz	使用 gzip 程序压缩的文件
.tar	使用 tar 程序压缩的文件，又称 tar 文件
.tbz	用 tar 和 bzip 压缩的文件
.tgz	用 tar 和 gzip 压缩的文件
.zip	使用 ZIP 压缩的文件，常见于 MS-DOS 应用程序中。

系统文件

.conf	一种配置文件，有时也用.cfg。
.lock	锁（lock）文件，用来判定程序或设备是否正在被使用。
.rpm	一种来自于 Red Hat 的软件包格式文件

编程和脚本文件

.c	C 语言的源码文件
.cpp	C++语言的源码文件
.h	C 或 C++语言的头文件
.o	程序的对象文件
.pl	Perl 脚本
.py	Python 脚本
.so	库文件
.sh	shell 脚本
.tcl	TCL 脚本

其它文件格式

.au	音频文件
.gif	GIF 图像文件
.jpg	JPEG 图像文件

.pdf	Portable Document Format, 可移植文档格式。
.png	PNG 图像文件 (Portable Network Graphic 的简写, 可移植网络图形)
.ps	PostScript 文件, 为打印而格式化过的文件。
.txt	纯 ASCII 文本文件
.wav	一种音频文件
.html/.htm	HTML 文件



如果某个文件的命名未反映出其文件类型, 可以用 `file` 命令帮助您判断, 具体操作请参见 `file` 命令的手册页。

附录D 术语表

account

在 Unix 系统中，指允许个人连接到系统的登录名称、个人目录、密码以及 shell 的组合。

alias

别名。在 shell 中为了能在执行命令时将某一字符串替换成另一个的一种机制。在提示符中键入 alias 可了解当前所定义的全部别名。

ARP

Address Resolution Protocol（地址解析协议）。该网际网络协议用于将网际网络地址动态地对应到局域网网络的硬件地址上。

ATAPI

AT Attachment Packet Interface，AT 附件包装接口。最为人们所熟知的是 IDE；它提供了额外的指令来控制 CDROM 以及磁带装置。而具有延伸功能的 IDE 控制器通常被称为 EIDE（Enhanced IDE，加强型 IDE 控制器）。

batch

批处理。将工作按顺序送到处理器，处理器一个接一个执行直到最后一个完成并准备好接受另一组处理清单的一种处理模式。

boot

引导。即发生在按下计算机的电源开关，机器开始检测接口设备的状态，并把操作系统加载到内存中的整个过程。

bootdisk

引导盘。包含来自硬盘（有时也可从其本身）加载操作系统的必要程序代码的可开机软磁盘。

BSD

Berkeley Software Distribution（伯克利软件发行套件）。一套由美国伯克利大学信息相关科系所发展的 Unix 分支。

buffer

缓冲区。指内存中固定容量一个小区域，其中的内容可以加载区域模式文件，系统分区表，以及执行中的进程等等。所有缓冲区的连贯性都是由缓冲区内存来维护的。

buffer cache

缓冲区存取。这是操作系统核心中甚为重要的一部份，负责让所有的缓冲区保持在最新的状态，在必要时可以缩小内存空间，清除不需要的缓冲区。

CHAP

Challenge-Handshake Authentication Protocol（询问交互式身份验证协议）：ISP 验证其客户端所采用的通信协议。它与 PAP 的不同处在于：进行最初的判别后，每隔固定的时间周期它将会重新再验证一次。

client

客户端。是指能够短暂地连接到其它程序或计算机上并对其下达命令或要求信息的一个程序或一部计算机。它是**服务器/客户端系统**组件的一部分。

client/server system

服务器/客户端系统。由一个 **server**（服务器端）与一个或多个 **client**（客户端）所组成的系统架构或通信协议。

compilation

编译。指把人们读得懂的以某种程序语言（例如 C 语言）书写的程序源代码转换成机器可读的二进制文件的一种过程。

completion

自动补齐。只要系统内有能与之配合对象，**shell** 将自动把一个不完全的子字符串，延展扩大成一个已存在的文件名、用户名或其它种种的能力。

compression

压缩。这是一种在通信连接的传送过程中缩小文件或减少字符数目的方法。压缩程序通常包含有 **compress**, **zip**, **gzip** 及 **bzip2**。

console

控制台。也就是人们一般使用并称为终端的概念。它们是连接到一部巨型中央计算机的使用者操作的机器。对 PC 而言，实际的终端就是指键盘与屏幕。

cookies

由远程 **web** 服务器写入到本地硬盘的临时文件。它让服务器可以在使用者再次连上网站的时候可以知道其个人偏好。

DHCP

Dynamic Host Configuration Protocol（动态主机配置协议）。一种以局域网络机器为设计基础，能从 **DHCP** 服务器动态取得 IP 地址的通信协议。

DMA

Direct Memory Access（直接内存存取）。一种运用在 PC 架构上的技术，它允许接口设备可以从主存储器存取或读写资料而无须通过 **CPU** 联系。

DNS

Domain Name System（网络域名系统）。用来负责分配名称/地址的机制。它可以将机器名称对应到 IP 地址。同样 **DNS** 也允许反向搜寻，也就是说可以从 IP 地址得知其机器名称。

DPMS

Display Power Management System（显示器电源管理系统）。用于所有现今生产的显示器以管理其电源使之能够延长使用年限的协议。

editor

编辑器。一般而言是指编辑文本文件所使用的程序（也就是文字编辑器）。最为人所熟知的 **GNU/Linux** 编辑器有 **Emacs** 以及 **VIM**。

email

电子邮件。是处于相同网络里的人们互相传送电子信息的一种方式。与定期邮件相同，**email** 需要收件人以及寄件人地址以便正确地传送信息。

environment variables

环境变量。可以直接通过 shell 查看环境变量。

ext2

Extended 2 filesystem 的简称。是 GNU/Linux 原有的文件系统并且有任何 Unix 文件系统的特色：支持特殊文件（字符设备，符号链结.....），文件的权限与所有权等等。

FAT

File Allocation Table（文件配置表）。使用于 DOS 以及 Windows 操作系统上的文件系统。

FDDI

Fiber Distributed Digital Interface（光纤分配式数字接口）。一种用于光纤通信的高速网络物理层。

FIFO

First In, First Out（先进先出）。一种内容项目被取出是依据其放入顺序的数据结构或硬件缓冲区。管道是 FIFO 概念在实践中最为普遍的一个例子。

Filesystem

文件系统。为使文件储存在实际介质（硬盘、磁盘）上时能够保持其资料的一致性所做的一种规划方式。

firewall

防火墙。在局域网络的拓扑中，负有与外界网络联系节点责任的机器或专用设备；同时也负有过滤或控制某些通信端口的活动以及确定哪些特定接口能够予以存取等多重任务。

framebuffer

视频缓冲区。将显示卡上的 RAM 对应到机器内存地址空间的一种技术。它允许应用程序存取显示卡上的 RAM 而无须与之直接沟通。

FTP

File Transfer Protocol（文件传输协议）。这是用于机器间彼此传输文件的标准网际网络通信协议。

gateway

网关。用来连接两个 IP 网段之间的网络设备。

GIF

Graphics Interchange Format（图形交换格式）。一种广泛用于 web 的影像文件格式，GIF 影像资料可被压缩或者存入动态画面。

GNU

GNU's Not Unix 的缩写。GNU 计划由 Richard Stallman 发起于 80 年代初期，其目标是要发展出一套 free 的操作系统（“free”代表“自由”而非免费）。

GPL

General Public License（通用公共许可证）。其理念与所有的商业软件授权大不相同：对于软件本身的复制、修改以及重新散布没有任何的限制，用户可以取得源代码，唯一的限制是将它散布给他人时，对方也将因相同的权利而获益。

GUI

Graphical User Interface（图形用户接口）。使用菜单，按钮，以及图标等等组成窗口外观的一种计算

机操作界面。

host

主机，计算机的一种称呼。一般而言对连接到网络上的计算机时才会使用这个名词。

HTTP

HyperText Transfer Protocol（超文本传输协议）。此种通信协议让您得以连上缤纷多彩的网站并取回 HTML 文件或档案。

HTML

HyperText Markup Language（超文本标记语言）。这种语言可以用来书写 web 网页文件。

inode

在 Unix 类的文件系统中用来指向文件内容的进入点。每个 inode 皆可由这种独特的方式作为识别，且同时包含着关于其所指向档案的相关信息，如存取时间、类型、文件大小。

Internet

际网络。这是一个连接世界上众多计算机的巨大网络。

IP address

IP 地址。一组在 Internet 上用来确认计算机的由四组数字组成的地址表示法，IP 地址看起来像是 192.168.0.1 这种样子。而机器本身的地址有二种类型：静态或动态。静态 IP 地址不会变动；而动态 IP 地址则是指每次重新连上网络时，IP 地址都会有所不同。

IP masquerading

IP 伪装。当使用防火墙时隐藏计算机真实 IP 地址以防止为外界所窥知的一种方法。传统上任何越过防火墙而来的外界网络连结所取得的是防火墙的 IP 地址。

IRC

Internet Relay Chat（网际网络接力聊天室）。一种网络上用来实时交谈的标准。它允许建立一个频道（channel）进行私人秘密会谈，还可以传输文件。

ISA

Industry Standard Architecture（工业标准结构）。用于个人计算机上非常早期的总线规格，它正慢慢地被 PCI 总线所取代。

ISDN

Integrated Services Digital Network（综合服务数字网络）。一组允许以单一线缆或光纤传送声音、数字网络服务及影像的通信标准。

ISO

International Standards Organization（国际标准化组织）。

ISP

Internet Service Provider（网络服务提供者）。是指对其顾客提供网络存取而不论其介质是采用电话还是专用线路的公司。

kernel

核心。这是操作系统的关键所在。核心负责分配资源并区分各个使用者的进程。它处理着允许程序与

计算机硬件直接沟通的所有动作，包含管理缓冲区快速存取等等。

LAN

Local Area Network（本地端局域网）。一般而言是指当机器以相同实体线缆连接时所构成的网络系统。

LDP

Linux Documentation Project（Linux 文件计划）。一个维护 GNU/Linux 文件的非营利组织。其最著名的成果为各式各样的 HOWTO 文件，除此之外它也维护着 FAQ，甚至是一些书籍。

loopback

一个机器连接到其本身的虚拟网络接口，它允许执行中的程序不必去考虑两个网络实体事实上都位于相同机器的这种特殊状况。

manual page

参考手册。包含指令及其用法定义，可以 man 这个指令查阅的小型文件。

MBR

Master Boot Record（主引导记录）。指可引导硬盘的第一扇区所使用的名称。MBR 中包含用来将操作系统加载到内存或开机加载程序（例如 LILO）的执行码，以及该硬盘的分区表。

MIME

Multipurpose Internet Mail Extensions（多用途网际网络邮件延伸格式）。在电子邮件里，以型态/子型态（type/subtype）形式描述其包含文件内容的一段字符串。

MPEG

Moving Pictures Experts Group（运动图像专家组）。一个制订影音压缩标准的 ISO 委员会；同时 MPEG 也是他们的算法名称。

NCP

NetWare Core Protocol（NetWare 核心协议）。由 Novell 公司定义的用以存取 Novell NetWare 系统的文件及打印服务的通信协议。

newsgroups

新闻群组。能由新闻或 USENET 客户端程序加以存取以便让人阅读或写入信息到某新闻群组的特定主题讨论区或新闻区。

NFS

Network FileSystem（网络文件系统）。提供通过网络来共享文件的网络文件系统。

NIC

Network Interface Controller（网络接口控制器）。安装到计算机上并提供对网络实体连接所使用的转接器，如 Ethernet 网卡。

NIS

Network Information Service（网络信息服务），NIS 的目的在于分享跨越 NIS 网域的共有信息，该 NIS 网域涵盖了整个局域网、部分的局域网或是数个局域网。它能够输出密码数据库，服务数据库，以及群组信息等。

PAP

Password Authentication Protocol（密码认证程序）。一种许多 ISP 用来认证客户端的协议，在这一设计中，客户端会送出一组未经编码的 ID 和密码给 server。

patch

补丁。包含有需发布的源代码的修订列表，目的是为了增加新功能，修改 bug 或按某些实际需要去修正。

path

指定文件或目录在文件系统的位置。在 GNU/Linux 中有两种不同的路径：**相对路径**指的是文件或目录相对于当前目录的位置；**绝对路径**指的是文件或目录相对于根目录的位置。

open source

开放源代码。其理念在于一旦允许广大的程序设计师可以共同使用及修改原始程序代码，最终将会产生出对所有人而言最有用的产品。一些受欢迎的开放源码程序包括 Apache，sendmail 以及 GNU/Linux。

PAP

Password Authentication Protocol（密码认证程序）。一种许多 ISP 用来认证客户端的协议，在这一设计中，客户端会送出一组未经编码的 ID 和密码给服务器。

PCI

Peripheral Components Interconnect。由 Intel 制定的总线规格，现在已成为 PC 架构中的总线标准。它是 ISA 的继承者，而且提供了许多服务：装置、设定信息、IRQ 分享、总线控制及其它更多的功能。

PCMCIA

Personal Computer Memory Card International Association（个人计算机存储卡国际协会）通常被简称为“PC Card”，是便携式计算机外接口的标准，如：调制解调器，硬盘，存储卡，以太网卡等。

pipe

一种特别的 Unix 文件形式。一个程序将资料写入 pipe，而另一个程序由 pipe 读出资料直到结束。管道采用 FIFO（先进先出），因此资料被另一个程序读入直到顺序结束。

pixmap

“pixel map”的缩写。是 bitmapped 影像的一种。

PNG

Portable Network Graphics（可移植网络图像文件）。该文件格式主要是给 web 使用，它被设计成无专利的，以取代具有专利权的 GIF，而且也有一些附加的功能。

PNP

Plug'N'Play（随插即用）。首先被用于 ISA 装置以便新增设定的信息，如今更广泛地用于所有装置以便回显设定参数。正如我们所知，所有的 PCI 装置都是即插即用的。

POP

Post Office Protocol（邮局协议）。这种常见的通信协议用于从 ISP 下载电子邮件。

PPP

Point to Point Protocol（点对点通信协议）。是一种通过序列信号线来传送资料的通信协议。通常被用于传送 IP 封包到网际网络，也可以和其它的通信协议一起使用，如 Novell 的 IPX 协议。

preprocessors

前置处理器。指示编译器取代在源代码中特定资料或程序片段，例如 C 的前置处理器为 `#include`，`#define` 等。

process

进程。在操作系统中，一个进程是伴随着一个程序的执行产生的。

prompt

提示符号。在 shell 中，它是在光标前的字符串。在其后输入字符命令。

Protocol

通信协议是指不同的机器经由网络通信的方式，不管是用软件或硬件，它们定义了数据传输时的格式。有许多的有名的通信协议，如 HTTP，FTP，TCP，和 UDP 等。

proxy

代理服务器。一台位于某一网络和网际网络间的机器，主要任务是加速多数被广泛使用的通信协议（如 HTTP、FTP）。它包含了一个预置的快速存取，可以降低重复资料被再次要求的成本。

quota

配额限制是限制使用者对于磁盘空间使用的一种方法。在某些文件系统上，管理者可以对各个使用者的目录做不同的大小限制。

RAID

Redundant Array of Independent Disks。始于伯克利大学资料系的一个计划，目的是让储存的资料分散于同一数组但不同的磁盘上。

RAM

Random Access Memory（随机存取内存）。是指计算机的主存储器“Random”也指内存的任何一部分都能被直接存取。

read-only mode

只读模式。表示不能写入文件，只能读取内容，当然也不能修改或删除文件。

read-write mode

读写模式。表示文件是可以被写入的，可以读取或修改文件内容，如果拥有这一权限，也可以删除文件。

root

root 是任何 Unix 系统上的超级使用者。Root 负责管理并维护整个 Unix 系统。

RFC

Request For Comments（计算机与通信技术文件）。RFC 是官方的 Internet 标准文件，由 IETF（Internet Engineering Task Force）所发行。他们描述所有使用或被要求使用的协议，如果想知道某一种通信协议是如何运作的，就可以去找对应的 RFC 文件来读。

RPM

Redhat Package Manager（红帽子软件包管理器）。一种为了产生软件套件而由 Red Hat 开发的软件包格式。它被用于许多 GNU/Linux 发行版本上，包括红旗 Linux。

run level

运行级别。是一项关于只允许某些被选定的进程存在的系统设定。在文件/etc/inittab 中清楚地定义每个运行级别有那些进程是被允许的。

SCSI

Small Computers System Interface (小型计算机系统接口)，一种高效且允许多种不同外设都能使用的总线规格。不同于 IDE，SCSI 总线的效能并不会受限于外围能接受指令的速度。只有高阶的机器才会在主板上内建 SCSI 总线，一般的 PC 用另外插卡的方式。

server

服务器。为程序或计算机提供功能或服务让客户端可以连接进来执行命令或是取得其所需的信息。

shadow passwords

影子密码。Unix 中的一种密码管理方式，系统中某个不是所有人都能读取的档案中存放着加过密的密码，是现在很常用的一种密码系统。它也提供了密码时间限制的功能。

shell

shell 是操作系统核心的基本接口，它提供命令行让使用者输入指令以便执行程序或系统命令。所有 shell 都有提供命令行的功能以便自动执行任务或是常用但复杂的任务。这些 shell 命令类似于 DOS 操作系统中的批处理文件，但是更为强大。常见的 shells 有 Bash，sh，和 tcsh 等。

SMB

Server Message Block 是 Windows (9x/2000 或 NT) 所使用的通信协议，用于通过网络共享文件或打印机。

SMTP

Simple Mail Transfer Protocol (简单邮件传输协议)，是一种用来传送电子邮件的协议。邮件传送代理者如 sendmail 或 postfix 都使用 SMTP，他们有时也会被称为 SMTP 服务器。

socket

一种符合于任何网络连结的文件形态。

TCP

Transmission Control Protocol (传输控制协议)。这是所有使用 IP 来传送网络封包中最可靠的通信协议。TCP 加入了必要的检查，在 IP 中来确保封包被传送。和 UDP 相反，TCP 在连接模式下运行，即在交换信息前，两端的机器就要先建立连接。

telnet

开启一个连接到远程主机，telnet 是进行远程登录最常用的方式，也有更好更安全的方式，如 ssh。

URL

Uniform Resource Locator (统一资源定位器)。一种统一并且特殊格式的字符串用以分辨在网络上的资源。这个资源可能是一个文件，一个服务器或是其它。

virtual desktops

虚拟桌面。在 X 窗口系统中，可以提供多个桌面。这一功能可以使您灵活安排工作窗口，避免让大量的程序都挤在同一桌面上。

WAN

Wide Area Network (广域网络)。

window manager

窗口管理器。一个负责图形环境“看起来的感觉”的程序。主要负责处理窗口的标题栏，框架，按钮，主菜单和一些快捷键方式。