第10章 Linux日常管理和维护

人民邮电出版社
POSTS & TELECOM PRESS

10.1 RPM软件包管理

10.2 tar包管理

10.3 进程管理

10.4 任务计划

10.5 Linux系统启动

10.6 维护GRUB

10.7 查看系统信息

人民邮电出版社
POSTS & TELECOM PRESS

在Linux系统中，最常用的软件包是RPM包和tar包。对于系统管理员来说，经常需要定时执行某个程序，则可以使用cron和at实现该功能。在Fedora 17系统中，使用的启动引导器是GRUB，其配置文件是/boot/grub/grub.conf。

10.1 RPM软件包管理

目前在众多的Linux系统上多采用RPM软件包，这种软件包格式在安装、升级以及卸载上非常方便，不需要进行编译。本节主要讲述RPM软件包的使用和管理。

10.1.1 RPM软件包简介

RPM软件包管理器（简称RPM）是一种开放的软件包管理系统，按照GPL条款发行，可以运行在Fedora 17以及其他Linux系统上。

1. 什么是RPM软件包

对于终端用户来说，RPM简化了系统安装、卸装、更新和升级的过程，只需要使用简短的命令就可完成。RPM维护一个已安装软件包和它们的文件的数据库，因此，可以在系统上使用查询和校验软件包功能。

对于开发者来说，RPM允许把软件编码包装成源码包和程序包，然后提供给终端用户，这个过程非常简单，这种对用户的纯净源码、补丁和建构指令的清晰描述减轻了发行软件新版本所带来的维护负担。Fedora 17系统上的所有软件都分成可被安装、升级或卸载的RPM软件包。

2. RPM软件包的设计目标

在使用RPM软件包之前，应该先来了解RPM的设计目标。

（1）可升级性

（2）强大的查询功能

（3）系统校验

（4）纯净源码

3. RPM包管理的用途

RPM软件包管理的用途如下：

（1）可以安装、删除、升级和管理软件；

（2）通过RPM软件包管理能知道软件包包含哪些文件，也能知道系统中的某个文件属于哪个软件包；

（3）可以查询系统中的软件包是否安装并查询其版本；

（4）开发者可以把自己的程序打包为RPM软件包发布；

（5）软件包签名GPG和MD5的导入、验证和签名发布；

（6）依赖性的检查，查看是否有软件包由于不兼容而扰乱了系统。

10.1.2 RPM软件包命令的使用

RPM主要有5种基本操作模式：安装、卸装、刷新、升级及查询。

1. 安装软件包

命令语法：

rpm -ivh [RPM包文件名称]

【例10.1】 安装bind-chroot-9.9.0-4.fc17.i686.rpm软件包。

```
[root@PC-LINUX ~]# cd /media/Fedora\ 17\ i386/Packages/b/
[root@PC-LINUX b]# pwd
/media/Fedora 17 i386/Packages/b
//进入Linux系统软件目录下
[root@PC-LINUX b]# rpm -ivh bind-chroot-9.9.0-4.fc17.i686.rpm
warning: bind-chroot-9.9.0-4.fc17.i686.rpm: Header V3 RSA/SHA256 Signature,
key                                          ID 1aca3465: NOKEY
Preparing...          ########################################### [100%]
   1:bind-chroot      ###########################################
[100%]
```

【例10.2】 在软件包bind-chroot-9.9.0-4.fc17.i686.rpm已安装的情况下仍旧安装该软件包。
如果在软件包已安装的情况下仍打算安装同一版本的软件包，可以使用“--replacepkgs”选项忽略错误。

```
[root@PC-LINUX b]# rpm -ivh --replacepkgs bind-chroot-9.9.0-4.fc17.i686.rpm
warning: bind-chroot-9.9.0-4.fc17.i686.rpm: Header V3 RSA/SHA256 Signature,
key ID 1aca3465: NOKEY
Preparing...                ########################################### [100%]
   1:bind-chroot            ###########################################
[100%]
```

2. 卸载软件包

命令语法：

rpm -e [RPM包名称]

【例10.3】 卸载bind-chroot软件包。

```
[root@PC-LINUX b]# rpm -e bind-chroot
```

3. 升级软件包

命令语法：

rpm -Uvh [RPM包文件名称]

【例10.4】 升级bind-9.9.0-4.fc17.i686.rpm软件包，升级软件包实际上是删除和安装的组合。

```
[root@PC-LINUX b]# rpm -Uvh bind-9.9.0-4.fc17.i686.rpm
warning: bind-9.9.0-4.fc17.i686.rpm: Header V3 RSA/SHA256 Signature, key ID
1aca3465: NOKEY
Preparing...                ########################################### [100%]
        package bind-32:9.9.0-4.fc17.i686 is already installed
```

【例10.5】 强制升级bind-9.5.0-16.a6.fc8.i386.rpm软件包。

```
[root@PC-LINUX b]# rpm -Uvh --oldpackage bind-9.9.0-4.fc17.i686.rpm
warning: bind-9.9.0-4.fc17.i686.rpm: Header V3 RSA/SHA256 Signature, key ID
1aca3465: NOKEY
Preparing...                ##########################################################
[100%]
        package bind-32:9.9.0-4.fc17.i686 is already installed
```

4. 刷新软件包

刷新软件包和升级软件包相似，刷新RPM软件包的基本命令语法如下。

rpm -Fvh [RPM包文件名称]

【例10.6】 刷新bind-chroot-9.9.0-4.fc17.i686.rpm软件包。

```
[root@PC-LINUX b]# rpm -Fvh bind-chroot-9.9.0-4.fc17.i686.rpm
warning: bind-chroot-9.9.0-4.fc17.i686.rpm: Header V3 RSA/SHA256 Signature,
key ID 1aca3465: NOKEY
```

5. 查询软件包

使用“rpm -q”命令可以查询软件包安装的相关信息。

（1）查询指定软件包的详细信息

命令语法：

rpm -q [RPM包名称]

【例10.7】 查询bind软件包是否安装

```
[root@PC-LINUX ~]# rpm -q bind
package bind is not installed
//查询到bind软件包没有安装
```

【例10.8】 查询bind软件包是否安装。

```
[root@PC-LINUX ~]# rpm -q bind
bind-9.9.0-4.fc17.i686
//查询到bind软件包已经安装
```

（2）查询系统中所有已安装的RPM软件包

命令语法：

```
rpm -qa
```

【例10.9】 查询系统内所有已安装的RPM软件包。

```
[root@PC-LINUX ~]# rpm -qa
libquvi-scripts-0.4.4-1.fc17.noarch
lxappearance-0.5.1-1.fc17.i686
un-extra-vada-fonts-1.0.2-0.14.080608.fc17.noarch
lzop-1.03-4.fc17.i686
libreport-gtk-2.0.10-3.fc17.i686
ar9170-firmware-2009.05.28-4.fc17.noarch
libsigc++20-2.2.10-2.fc17.i686
icedax-1.1.11-10.fc17.i686
poppler-qt-0.18.4-3.fc17.i686
gnome-packagekit-3.4.0-1.fc17.i686
setup-2.8.48-1.fc17.noarch
……
```

（3）查询指定已安装软件包的描述信息

命令语法：

rpm -qi [RPM包名称]

【例10.10】 查询bind软件包的描述信息。

```
[root@PC-LINUX ~]# rpm -qi bind
Name        : bind
Epoch       : 32
Version     : 9.9.0
Release     : 4.fc17
Architecture: i686
Install Date: 2012年06月03日 星期日 00时35分57秒
Group       : System Environment/Daemons
Size        : 5383347
License     : ISC
Signature   : RSA/SHA256, 2012年04月25日 星期三 09时52分10秒, Key ID 50e94c991aca3465
Source RPM  : bind-9.9.0-4.fc17.src.rpm
Build Date  : 2012年04月24日 星期二 21时12分00秒
Build Host  : buildvm-02.phx2.fedoraproject.org
Relocations : (not relocatable)
Packager    : Fedora Project
Vendor      : Fedora Project
URL         : http://www.isc.org/products/BIND/
Summary     : The Berkeley Internet Name Domain (BIND) DNS (Domain Name System) server
Description :
BIND (Berkeley Internet Name Domain) is an implementation of the DNS
(Domain Name System) protocols. BIND includes a DNS server (named),
which resolves host names to IP addresses; a resolver library
(routines for applications to use when interfacing with DNS); and
tools for verifying that the DNS server is operating properly.
```

（4）查询某已安装软件包所含的文件列表
命令语法：
rpm －ql [RPM包名称]

【例10.11】 查询bind软件包所包含的文件列表。

```
[root@PC-LINUX ~]# rpm -ql bind
/etc/NetworkManager/dispatcher.d/13-named
/etc/logrotate.d/named
/etc/named
/etc/named.conf
/etc/named.iscdlv.key
/etc/named.rfc1912.zones
/etc/named.root.key
/etc/rndc.conf
/etc/rndc.key
/etc/sysconfig/named
……..
```

（5）查询软件包的依赖要求

命令语法：

rpm -qR [RPM包名称]

【例10.12】 查询bind软件包的依赖关系。

```
[root@PC-LINUX /]# rpm -qR bind
/bin/bash
/bin/sh
/bin/sh
/bin/sh
/bin/sh
/bin/sh
bind-libs = 32:9.9.0-4.fc17
config(bind) = 32:9.9.0-4.fc17
coreutils
grep
libbind9.so.90
……
```

（6）查询系统中指定文件属于哪个软件包

命令语法：

rpm -qf [文件名]

【例10.13】 查询/etc/logrotate.d/named文件属于哪个软件包。

```
[root@PC-LINUX ~]# rpm -qf /etc/logrotate.d/named
bind-9.9.0-4.fc17.i686
//当指定文件时，必须指定文件的完整路径（如/etc/logrotate.d/named）
```

10.2 tar包管理

使用tar命令可以将文件和目录进行打包或压缩以做备份用。本节主要讲述tar包的使用和管理。

10.2.1 tar包简介

在Windows系统下最常见的压缩文件是zip和rar，Linux系统就不同了，它有.gz、.tar.gz、.tgz、.bz2、.Z、.tar等众多的压缩文件名，此外Windows系统下的.zip和.rar也可以在Linux下使用，本节主要讲解如何管理这些软件包。

在具体讲述压缩文件之前需要了解打包和压缩的概念。打包是指将许多文件和目录变成一个总的文件，压缩则是将一个大的文件通过一些压缩算法变成一个小文件。Linux系统中的很多压缩程序只能针对一个文件进行压缩，这样当需要压缩一大堆文件时，就得先借助其他的工具将这一大堆文件先打成一个包，然后再用原来的压缩程序进行压缩。

Linux系统下最常用的打包程序是tar，使用tar程序打出来的包称为tar包，tar包文件的命令通常都是以.tar结尾的。生成tar包后，就可以用其他的程序来进行压缩了。

tar可以为文件和目录创建备份。利用tar，用户可以为某一特定文件创建备份，也可以在备份中改变文件，或者向备份中加入新的文件。

tar最初被用来在磁带上创建备份，现在，用户可以在任何设备上创建备份，如软盘。利用tar命令可以把一大堆的文件和目录打包成一个文件，这对于备份文件或将几个文件组合成为一个文件进行网络传输是非常有用的。

10.2.2 tar包使用和管理

命令语法：

tar [主选项+辅选项][文件或者目录]

【例10.14】 备份/root/abc目录及其子目录下的全部文件，备份文件名为abc.tar。

```
[root@PC-LINUX ~]# touch /root/abc/a /root/abc/b /root/abc/c
//在/root/abc目录中创建/root/abc/a、/root/abc/b和/root/abc/c文件
[root@PC-LINUX ~]# tar cvf abc.tar /root/abc
tar: 从成员名中删除开头的“/”
/root/abc/
/root/abc/c
/root/abc/b
/root/abc/a
[root@PC-LINUX ~]# ls -l
总用量 284
drwxr-xr-x. 2 root root   4096 6月   3 05:38 abc
-rw-r--r--. 1 root root  10240 6月   3 05:39 abc.tar
-rw-------. 1 root root  10670 6月   3 01:17 anaconda-ks.cfg
-rw-r--r--. 1 root root 155641 6月   3 01:16 install.log
-rw-r--r--. 1 root root  65450 6月   3 01:15 install.log.syslog
//可以看到abc.tar就是/root/abc目录打包后生成的文件
```

【例10.15】 查看abc.tar备份文件的内容，并显示在显示器上。

```
[root@PC-LINUX ~]# tar tvf abc.tar
drwxr-xr-x root/root         0 2012-06-03 05:38 root/abc/
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/c
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/b
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/a
//可以看到该打包文件由一个目录和该目录下的3个文件打包而成
```

【例10.16】 将打包文件abc.tar解包出来。

```
[root@PC-LINUX ~]# tar xvf abc.tar
root/abc/
root/abc/c
root/abc/b
root/abc/a
```

【例10.17】 将文件/root/abc/d添加到abc.tar包里面去。

```
[root@PC-LINUX ~]# touch /root/abc/d
[root@PC-LINUX ~]# tar rvf abc.tar /root/abc/d
tar: 从成员名中删除开头的“/”
/root/abc/d
[root@PC-LINUX ~]# tar tvf abc.tar
drwxr-xr-x root/root         0 2012-06-03 05:38 root/abc/
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/c
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/b
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/a
-rw-r--r-- root/root         0 2012-06-03 05:41 root/abc/d
//查看abc.tar包内容，可以看到文件/root/abc/d已经添加进去了
```

【例10.18】 更新原来tar包abc.tar中的文件/root/abc/d。

```
[root@PC-LINUX ~]# tar uvf abc.tar /root/abc/d
tar: 从成员名中删除开头的“/”
/root/abc/d
[root@PC-LINUX ~]# tar tvf abc.tar
drwxr-xr-x root/root         0 2012-06-03 05:38 root/abc/
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/c
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/b
-rw-r--r-- root/root         0 2012-06-03 05:38 root/abc/a
-rw-r--r-- root/root         0 2012-06-03 05:41 root/abc/d
-rw-r--r-- root/root         0 2012-06-03 05:41 root/abc/d
```

10.2.3 tar包的特殊使用

tar可以在打包或解包的同时调用其他的压缩程序，比如调用gzip，bzip2等。

1. tar调用gzip

gzip是GNU组织开发的一个压缩程序，以.gz结尾的文件就是gzip压缩的结果。与gzip相对应的解压程序是gunzip，tar中使用参数“z”来调用gzip，下面举例说明。

【例10.19】 把/root/abc目录包括其子目录全部做备份文件，并进行压缩，文件名为abc.tar.gz。

```
[root@PC-LINUX ~]# tar zcvf abc.tar.gz /root/abc
tar: 从成员名中删除开头的“/”
/root/abc/
/root/abc/c
/root/abc/b
/root/abc/a
[root@PC-LINUX ~]# ls -l
总用量 276
drwxr-xr-x. 2 root root   4096 6月   3 05:48 abc
-rw-r--r--. 1 root root    161 6月   3 05:48 abc.tar.gz
-rw-------. 1 root root  10670 6月   3 01:17 anaconda-ks.cfg
-rw-r--r--. 1 root root 155641 6月   3 01:16 install.log
-rw-r--r--. 1 root root  65450 6月   3 01:15 install.log.syslog
//可以看到abc.tar.gz文件就是/root/abc目录压缩后的文件
```

【例10.20】 查看压缩文件abc.tar.gz的内容，并显示在显示器上。

```
[root@PC-LINUX ~]# tar ztvf abc.tar.gz
drwxr-xr-x root/root         0 2012-06-03 05:48 root/abc/
-rw-r--r-- root/root         0 2012-06-03 05:44 root/abc/c
-rw-r--r-- root/root         0 2012-06-03 05:44 root/abc/b
-rw-r--r-- root/root         0 2012-06-03 05:44 root/abc/a
//可以看到该压缩文件由一个目录和该目录下的3个文件压缩而成
```

【例10.21】 将压缩文件abc.tar.gz解压缩出来。

```
[root@PC-LINUX ~]# tar zxvf abc.tar.gz
root/abc/
root/abc/c
root/abc/b
root/abc/a
```

2. tar调用bzip2

bzip2是一个压缩能力更强的压缩程序，以.bz2结尾的文件就是bzip2压缩的结果。与bzip2相对应的解压程序是bunzip2。tar中使用参数“j”来调用bzip2，下面举例说明。

【例10.22】 将目录/root/abc及该目录所有文件压缩成abc.tar.bz2文件。

```
[root@PC-LINUX ~]# tar cjf abc.tar.bz2 /root/abc
tar: 从成员名中删除开头的“/”
[root@PC-LINUX ~]# ls -l
总用量 284
drwxr-xr-x. 2 root root   4096 6月   3 05:48 abc
-rw-r--r--. 1 root root    159 6月   3 05:50 abc.tar.bz2
-rw-------. 1 root root  10670 6月   3 01:17 anaconda-ks.cfg
-rw-r--r--. 1 root root 155641 6月   3 01:16 install.log
-rw-r--r--. 1 root root  65450 6月   3 01:15 install.log.syslog
```

【例10.23】 查看压缩文件abc.tar.bz2的内容，并显示在显示器上。

```
[root@PC-LINUX ~]# tar tjf abc.tar.bz2
root/abc/
root/abc/c
root/abc/b
root/abc/a
```

【例10.24】 将abc.tar.bz2文件解压缩。

```
[root@PC-LINUX ~]# tar xjf abc.tar.bz2
```

10.3 进程管理

Linux是一个多任务的操作系统，在系统中可以同时运行多个进程，正在执行的一个或多个相关进程称为一个作业。用户可以同时运行多个作业，并在需要时可以在作业之间进行切换。

10.3.1 进程概念

Linux系统上所有运行的内容都可以称为进程。

进程是在自身的虚拟地址空间运行的一个单独的程序。

和进程相比较，作业是一系列按一定顺序执行的命令。

Linux系统有以下3种进程，每一种进程都有其特性。

（1）交互式进程：一个由shell启动并控制的进程，交互式进程既可在前台运行，也可在后台运行。

（2）批处理进程：与终端无关，安排在指定时刻完成的一系列进程。

（3）守护进程：在引导系统时起动，以执行即时的操作系统任务，如lpd，inetd及named等。

10.3.2 查看系统进程信息

要查看Linux系统中的进程信息可以使用ps和top这两个命令。

1. ps命令

要对进程进行监测和控制，首先必须要了解当前进程的情况，也就是需要查看当前进程，ps命令是最基本同时也是非常强大的进程查看命令。使用该命令可以确定有哪些进程正在运行以及进程运行的状态、进程是否结束，进程有没有僵死，哪些进程占用了过多的资源等。

使用ps命令可以用于监控后台进程的工作情况。

命令语法：

ps [选项][/b]

表10-1　　ps命令输出字段的含义

字段	含　义
USER	进程所有者的用户名
PID	进程号，可以唯一标识该进程
%CPU	进程自最近一次刷新以来所占用的CPU时间和总时间的百分比
%MEM	进程使用内存的百分比
VSZ	进程使用的虚拟内存大小，以KB为单位
RSS	进程占用的物理内存的总数量，以KB为单位
TTY	进程相关的终端名
STAT	进程状态，R表示运行或准备运行，S表示睡眠状态，I表示空闲；Z表示冻结，D表示不间断睡眠，W表示进程没有驻留页，T表示停止或跟踪
START	进程开始运行时间
TIME	进程使用的总CPU时间
COMMAND	被执行的命令行

【例10.25】 显示所有不带控制台终端的进程，并显示用户名和进程的起始时间。

```
[root@PC-LINUX ~]# ps -aux
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3   5776  3512 ?        Ss   08:27   0:02 /usr/lib/system
root         2  0.0  0.0      0     0 ?        S    08:27   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    08:27   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    08:27   0:00 [kworker/u:0]
root         6  0.0  0.0      0     0 ?        S    08:27   0:00 [migration/0]
root         7  0.0  0.0      0     0 ?        S    08:27   0:00 [watchdog/0]
root         8  0.0  0.0      0     0 ?        S<   08:27   0:00 [cpuset]
root         9  0.0  0.0      0     0 ?        S<   08:27   0:00 [khelper]
root        10  0.0  0.0      0     0 ?        S    08:27   0:00 [kdevtmpfs]
root        11  0.0  0.0      0     0 ?        S<   08:27   0:00 [netns]
root        12  0.0  0.0      0     0 ?        S    08:27   0:00 [sync_supers]
root        13  0.0  0.0      0     0 ?        S    08:27   0:00 [bdi-default]
root        14  0.0  0.0      0     0 ?        S<   08:27   0:00 [kintegrityd]
root        15  0.0  0.0      0     0 ?        S<   08:27   0:00 [kblockd]
root        16  0.0  0.0      0     0 ?        S<   08:27   0:00 [ata_sff]
root        17  0.0  0.0      0     0 ?        S    08:27   0:00 [khubd]
root        18  0.0  0.0      0     0 ?        S<   08:27   0:00 [md]
root        20  0.0  0.0      0     0 ?        S    08:28   0:00 [kswapd0]
root        21  0.0  0.0      0     0 ?        SN   08:28   0:00 [ksmd]
root        22  0.0  0.0      0     0 ?        SN   08:28   0:00 [khugepaged]
root        23  0.0  0.0      0     0 ?        S    08:28   0:00 [fsnotify_mark]
root        24  0.0  0.0      0     0 ?        S<   08:28   0:00 [crypto]
……
```

【例10.26】 查看less进程是否在运行。

```
[root@PC-LINUX ～]# ps ax|grep less
 6366 pts/1       T          0:00 less
 6369 pts/1       T          0:00 less
 6374 pts/1       R+         0:00 grep --color=auto less
```

【例10.27】 显示系统进程。

```
[root@PC-LINUX ~]# ps
  PID TTY          TIME CMD
  890 pts/0    00:00:01 bash
 3858 pts/0    00:00:00 ps
```

【例10.28】 显示用户名和进程的起始时间。

```
[root@PC-LINUX ~]# ps -u
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
root       681  0.0  2.2  56720 22620 tty1     Ss+  00:26   0:11 /usr/bin/Xorg :
root       890  0.0  0.2   8360  2880 pts/0    Ss   00:26   0:01 -bash
root      3890  0.0  0.1   6592  1072 pts/0    R+   04:19   0:00 ps -u
```

【例10.29】 显示用户root的进程。

```
[root@PC-LINUX ~]# ps -u root|more
 PID TTY          TIME CMD
   1 ?        00:00:04 systemd
   2 ?        00:00:00 kthreadd
   3 ?        00:00:00 ksoftirqd/0
   5 ?        00:00:01 kworker/u:0
   6 ?        00:00:00 migration/0
   7 ?        00:00:03 watchdog/0
   8 ?        00:00:00 cpuset
   9 ?        00:00:00 khelper
  10 ?        00:00:00 kdevtmpfs
  11 ?        00:00:00 netns
  12 ?        00:00:00 sync_supers
  13 ?        00:00:00 bdi-default
  14 ?        00:00:00 kintegrityd
  15 ?        00:00:00 kblockd
  16 ?        00:00:00 ata_sff
  17 ?        00:00:00 khubd
  18 ?        00:00:00 md
  20 ?        00:00:08 kswapd0
  21 ?        00:00:00 ksmd
  22 ?        00:00:00 khugepaged
  23 ?        00:00:00 fsnotify_mark
  24 ?        00:00:00 crypto
```

2. top命令

使用top命令可以显示当前正运行的进程以及关于它们的重要信息，包括它们的内存和CPU用量。

命令语法：

top [bciqsS][d <间隔秒数>][n <执行次数>]

【例10.30】 使用top命令动态显示进程信息。

```
 [root@PC-LINUX ~]# top
top - 09:27:04 up 59 min,  2 users,  load average: 0.01, 0.08, 0.11
Tasks: 140 total,   1 running, 139 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.8%us,  0.8%sy,  0.0%ni, 98.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1027848k total,   917296k used,   110552k free,   212508k buffers
Swap:  2097148k total,        0k used,  2097148k free,   406880k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2863 root      20   0  2884 1084  820 R  1.5  0.1   0:00.03 top
    1 root      20   0  5776 3512 1940 S  0.0  0.3   0:02.97 systemd
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
    5 root      20   0     0    0    0 S  0.0  0.0   0:00.08 kworker/u:0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root      RT   0     0    0    0 S  0.0  0.0   0:00.21 watchdog/0
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    9 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kdevtmpfs
```

10.3.3 结束进程

要关闭某个应用程序可以通过结束其进程的方式实现，如果进程一时无法结束，可以将其强制结束。

1. 结束进程

有时，用户需要终止一个进程，要终止一个进程可能有以下原因。

（1）该进程使用CPU时间过多。

（2）该进程运行了很长时间，而没有产生期望的输出。

（3）该进程产生到屏幕或文件中的输出太多。

（4）该进程锁住了一个终端或其他的会话过程。

（5）由于操作或编程的错误，该进程正在使用错误的输入和输出文件。

如果要终止一个前台的命令，按[Ctrl+C]键就可以终止该进程。如果被终止的是一个后台进程，那么可以使用kill命令去杀死这个进程。

当命令不在前台时，按[Ctrl+C]键不能使之终止，这是因为后台进程不在终端控制下，所以任何键盘输入都被忽略。这时必须使用kill命令。

在使用kill命令之前，需要得到要被杀死的进程的ID号。用户可以用ps命令获得进程的ID号，然后用进程的ID号作为kill的参数。

【例10.31】 结束进程号为6388的进程。

```
[root@PC-LINUX ~]# ps -u
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       2368  0.0  0.1   1756   380 tty1     Ss+  06:22   0:00 /sbin/mingetty
root       2369  0.0  0.1   1760   408 tty2     Ss+  06:22   0:00 /sbin/mingetty
root       2370  0.0  0.1   1760   380 tty3     Ss+  06:22   0:00 /sbin/mingetty
root       2371  0.0  0.1   1756   380 tty4     Ss+  06:22   0:00 /sbin/mingetty
root       2372  0.0  0.1   1760   380 tty5     Ss+  06:22   0:00 /sbin/mingetty
root       2373  0.0  0.1   1760   380 tty6     Ss+  06:22   0:00 /sbin/mingetty
root       2966  2.0  3.5  23260 13024 tty7     Ss+  06:31   6:17 /usr/bin/X :0 -
root       3348  0.1  1.4  16812  5480 tty10    Ss+  06:52   0:19 /usr/bin/X :21
root       5235  0.0  0.4   5628  1764 pts/1    Ss   09:59   0:03 -bash
root       5975  0.0  0.2   4684   756 pts/1    T    10:35   0:00 more
root       6366  0.0  0.2   5020   740 pts/1    T    11:27   0:00 less
root       6369  0.0  0.2   5020   740 pts/1    T    11:27   0:00 less
root       6371  0.0  0.2   4684   760 pts/1    T    11:27   0:00 more
root       6386  0.0  0.2   4684   756 pts/1    T    11:31   0:00 more
root       6388  0.6  0.2   2416  1056 pts/1    T    11:31   0:02 top
root       6420  8.0  0.2   5160   900 pts/1    R+   11:37   0:00 ps –u
//查看进程信息，发现进程名为top的进程号是6388
[root@PC-LINUX ～]# kill -9 6388
```

2. 结束子进程

因为很多进程都会产生不同ID号的子进程，所以当用户要终止一个进程时，还必须终止其所有的子进程。当一个父进程产生子进程时，最好观察该进程几分钟，以确保所有的子进程都正常工作。在终止进程时，使用“ps -l”命令能检查出哪一个是父进程并终止它。

root用户如果要终止一个进程，必须确保输入了正确的PID号，否则可能会终止其他有用的进程。此外，仔细检查PID号，以免误删除系统的进程，除非知道它们为什么需要被终止。

如果用kill命令不能终止一个进程，就需要使用更有效的kill -9命令。

【例10.32】 强制杀死进程号是5975的进程。

```
[root@PC-LINUX ～]# kill -9 5975
[1]    已杀死                  rpm -ql bind | more    (wd: /media/Fedora 17 i386 DVD/Packages)
(wd now: ～)
```

10.3.4 进程的启动方式

输入需要运行的程序的程序名，执行一个程序，其实也就是启动了一个进程。在Linux系统中每个进程都具有一个进程号，用于系统识别和调度进程。启动一个进程有两种主要途径：手工启动和调度启动，后者是事先进行设置，根据用户要求自行启动。

1. 手工启动

（1）前台启动

【例10.33】查看前台启动进程。

```
[root@PC-LINUX ~]# find / -name fox.jpg
^Z
[1]+  已停止               find / -name fox.jpg
//输入命令按“回车”键后，按[Ctrl+Z]键挂起该命令
[root@PC-LINUX ~]# ps
  PID TTY          TIME CMD
 2548 pts/0    00:00:00 bash
 2870 pts/0    00:00:00 find
 2890 pts/0    00:00:00 ps
//查看进程信息，可以看到刚才执行的命令进程号是2870
```

（2）后台启动

直接从后台手工启动一个进程用得比较少一些，除非是该进程甚为耗时，且用户也不急着需要结果的时候。假设用户要启动一个需要长时间运行的格式化文本文件的进程。为了不使整个shell在格式化过程中都处于“瘫痪”状态，从后台启动这个进程是明智的选择。

从后台启动进程其实就是在命令结尾加上一个“&”号。

2. 调度启动

在 Linux系统中，任务可以被配置在指定的时间、指定的日期或系统平均负载量低于指定的数量时自动运行。Fedora 17预配置了重要系统任务的运行，以便使系统能够时时被更新。系统管理员可使用自动化的任务来执行定期备份、监控系统和运行定制脚本等。

10.3.5 进程的挂起及恢复

作业控制允许将进程挂起并可以在需要时恢复进程的运行，被挂起的作业恢复后将从中止处开始继续运行。只要在键盘上按[Ctrl+Z]键，即可挂起当前的前台作业。使用jobs命令可以显示shell的作业清单，包括具体的作业、作业号以及作业当前所处的状态。

恢复进程执行时，有两种选择：用fg命令将挂起的作业放回到前台执行，用bg命令将挂起的作业放到后台执行。

【例10.34】 挂起cat >/root/a作业后将作业放到前台执行。

```
[root@PC-LINUX media]# cat >/root/a
^Z
[1]+  已停止               cat > /root/a
//输入命令后，在按[Ctrl+Z]键挂起该命令
[root@PC-LINUX media]# jobs
[1]+  已停止               cat > /root/a
//查看Shell作业清单信息，可以看到一个挂起的作业
[root@PC-LINUX ～]# fg
cat >/root/a
//将挂起的作业放到前台执行
[root@PC-LINUX ～]# jobs
//再次查看Shell作业清单信息，已经没有挂起作业
```

【例10.35】用户正在使用vi编辑文件，挂起后将作业放在前台执行。

```
[root@PC-LINUX ~]# vi /root/it
[1]+  已停止               vi /root/it
//使用该命令将打开vi编辑器编辑文件/root/it，然后按[Ctrl+Z]键挂起该命令，接着可以执行其他命令
[root@PC-LINUX ~]# jobs
[1]+  已停止               vi /root/it
//查看Shell作业清单信息，可以看到一个挂起的作业
[root@PC-LINUX ~]# ps
  PID TTY          TIME CMD
 5251 pts/0    00:00:01 bash
 5567 pts/0    00:00:00 vi
 5582 pts/0    00:00:00 ps
//查看进程信息，可以看到刚才执行的命令进程号是5567
[root@PC-LINUX ～]# fg vi /root/it
//使用fg命令将刚才挂起的命令放到前台执行，继续使用vi编辑器编辑该文件，完毕并保存文件
[root@PC-LINUX ~]# ps
  PID TTY          TIME CMD
 5251 pts/0    00:00:01 bash
 5622 pts/0    00:00:00 ps
//查看进程信息，已经没有执行该命令的进程了
[root@PC-LINUX ～]# jobs
//查看Shell作业清单信息，可以看到没有挂起的作业
```

【例10.36】 挂起作业后放到后台执行。

```
 [root@PC-LINUX ~]# command find / -name "test" > find.out
^Z
[1]+  已停止               find / -name test
//输入完该命令后挂起命令
[root@PC-LINUX ~]# ps
  PID TTY          TIME CMD
 5251 pts/0    00:00:01 bash
 5663 pts/0    00:00:01 find
 5684 pts/0    00:00:00 ps
[root@PC-LINUX ~]# jobs
[1]+  已停止               find / -name test
[root@PC-LINUX ～]# bg
[1]+ find / -name test &
//将挂起的作业放到后台运行
[root@PC-LINUX ～]# jobs
[1]+   运行中                      find / -name test &
 //作业执行完毕
```

【例10.37】 挂起作业后指定作业号恢复作业。

```
[root@PC-LINUX ～]# jobs
[1]     已停止                  cat > /root/a
[2]     已停止                  vi /root/a
[3]-    已停止                  vi /root/b
[4]+    已停止                  vi /root/c
//使用jobs命令可以看到当前系统挂起的进程共4个
[root@PC-LINUX ～]# fg 4
//将号码是4的挂起作业放到前台执行，接着继续用vi编辑器编辑该文件
[root@PC-LINUX ～]# jobs
[1]     已停止                  cat > /root/a
[2]-    已停止                  vi /root/a
[3]+    已停止                  vi /root/b
//可以看到号码是4的作业已不在作业里面
```

10.4 任务计划

如果要在固定的时间上触发某个作业，就需要创建任务计划，按时执行该作业，在Linux系统中常用cron和at实现该功能。

10.4.1 配置cron实现自动化

使用cron实现任务自动化可以通过修改/etc/crontab文件以及使用crontab命令实现，其结果是一样的。

1. /etc/crontab文件实现自动化

cron守护进程可以在无需人工干预的情况下根据时间、日期、月份和星期的组合来调度执行重复任务的守护进程。

crontab文件格式：

<minute> <hour> <day> <month> <dayofweek> <commands>

第1列表示分钟0～59，每分钟用*表示，每3分钟用*/3表示

第2列表示小时0～23

第3列表示日期1～31

第4列表示月份1～12

第5列表示星期0～7，0或7表示星期天

第6列表示要执行的命令

2. crontab命令实现自动化

（1）crontab命令简介

用户可以使用crontab命令配置cron任务。所有用户定义的crontab都被保存在/var/spool/cron/<username>文件中，并使用创建它们的用户身份来执行。

crontab命令的语法是：

crontab [-u 用户名] -e -l -r

（2）控制对cron的使用

/etc/cron.allow和/etc/cron.deny文件被用来限制对cron的使用。

（3）启动和停止服务

启动cron服务，需要使用“systemctl start crond.service”命令。

停止cron服务，需要使用“systemctl stop crond.service”命令。

（4）创建crontab文件

root用户可以使用如下命令创建crontab文件：

crontab [-u 用户名] -e

普通用户在登录时使用如下命令创建crontab文件：

crontab -e

【例10.38】 以普通账号zhangsan登录系统，创建用户crontab条目。

```
[root@PC-LINUX ～]# su - zhangsan
//以普通用户zhangsan登录系统
[zhangsan@PC-LINUX root]$ crontab -e
//使用“crontab –e”命令打开vi编辑器,编辑用户zhangsan的crontab条目
8 * * * * cp /home/zhangsan/aa /home/zhangsan/bb
//在vi编辑器内输入以上crontab条目
[zhangsan@PC-LINUX root]$ su - root
口令：
//切换为root用户登录
[root@PC-LINUX ～]# cat /var/spool/cron/zhangsan
8 * * * * cp /home/zhangsan/aa /home/zhangsan/bb
//可以看到/var/spool/cron/zhangsan的内容就是刚才用“crontab -e”编辑的内容。
```

记住：普通用户没有权限打开该文件

（5）编辑crontab文件

root用户可以使用如下命令编辑crontab文件：

crontab [-u 用户名] -e

也可以使用vi直接编辑/var/spool/cron/<username>文件。

普通用户在登录时使用如下命令编辑crontab文件：

crontab -e

（6）列出crontab文件

为了列出crontab文件，root用户可以使用如下命令：

crontab [-u 用户名] -l

普通用户在登录时查看的命令如下所示：

crontab -l

（7）删除crontab文件

root用户可以使用如下命令删除crontab文件：

crontab [-u 用户名] -r

普通用户在登录时使用如下命令删除crontab文件：

crontab -r

（8）恢复丢失的crontab文件

如果不小心删除了crontab文件，可用crontab <filename>命令将备份的文件恢复到/var/spool/cron/中。

【例10.39】 以zhangsan用户登录恢复丢失的crontab文件。

```
[zhangsan@PC-LINUX ~]]$ crontab -r
[zhangsan@PC-LINUX ~]]$ crontab -l
no crontab for zhangsan
//删除crontab文件
[zhangsan@PC-LINUX ~]]$ crontab /home/zhangsan/zhangsancron
//恢复丢失的crontab文件
[zhangsan@PC-LINUX ~]]$ crontab -l
8 * * * * cp /home/zhangsan/aa /home/zhangsan/bb
//恢复以后可以看到丢失的crontab文件条目
```

10.4.2 使用at实现自动化

cron命令被用来调度重复的任务，而at命令被用来在指定时间内调度一次性的任务。

1. at简介

作业被提交时，at命令将会保留所有当前的环境变量，包括路径，该作业的所有输出都将以电子邮件的形式发送给用户。

和crontab一样，root用户可以通过/etc目录下的at.allow和at.deny文件来控制哪些用户可以使用at命令，在使用at命令之前必须安装at的RPM软件包，并且atd服务必须在运行。

2. 配置at作业

要在某一指定时间内调度一项一次性作业，可以输入“at [时间]”命令，必须先指定时间，接着指定日期。

命令语法：

at [-f script] [-m -l -r] [time] [date]

【例10.40】 在5天之后的此时此刻将/root/a文件复制到/home目录下。

```
[root@PC-LINUX ~]# at now +5 days
at> cp /root/a /home
at> <EOT>  //按[Ctrl+D]键退出
```

3. 使用at命令提交命令或脚本

使用at命令提交作业有几种不同的形式，可以通过命令行方式，也可以使用at命令提示符。一般来说在提交若干行的系统命令时，使用at命令提示符方式，而在提交shell脚本时，使用命令行方式。

如果想提交若干行的命令，可以在at命令后面跟上日期/时间并回车。然后就进入了at命令提示符，这时只需逐条输入相应的命令，然后按[Ctrl+D]键退出。

如果希望向at命令提交一个shell脚本，使用其命令行方式即可。在提交脚本时使用“-f”选项。

【例10.41】 使用echo命令向at命令提交作业。

```
[root@PC-LINUX ~]# echo hello>/root/a|at now
job 2 at Thu May 30 14:27:00 2013
[root@PC-LINUX ~]# ls -l /root
总用量 328
-rw-r--r--  1 root root      6 5月  30 14:27 a
-rw-------. 1 root root  10670 5月  30 07:10 anaconda-ks.cfg
-rw-r--r--. 1 root root 155641 5月  30 07:10 install.log
-rw-r--r--. 1 root root 119827 5月  30 07:09 install.log.syslog
//可以看到已经存在文件a了
[root@PC-LINUX ~]# cat /root/a
hello
//文件a的内容是hello
```

4. 列出所提交的作业

一个作业被提交后，可以使用“at -l”命令来列出所有的作业。

【例10.42】 使用“at –l”列出所有作业。

```
[root@PC-LINUX ～]# at –l
1       Tue Jun  4 14:27:00 2013 a root
```

【例10.43】 使用atq列出所有作业。

```
[root@PC-LINUX ～]# atq
1       Tue Jun  4 14:27:00 2013 a root
[root@PC-LINUX ～]# ls -l /var/spool/at
总用量 8
-rwx------  1 root   root   3113 5月  30 14:27 a00003015c7e03
drwx------. 2 daemon daemon 4096 5月  30 14:27 spool
//可以看到作业被复制到/var/spool/at目录下
```

5. 清除at作业

要清除某个作业，首先要执行“at -l”命令查看相应的作业标识，然后再清除作业。

命令语法：

atrm [作业表示号]

at -r [作业表示号]

【例10.44】使用命令atrm删除作业。

```
[root@PC-LINUX ～]# at -l
1       Tue Jun  4 14:27:00 2013 a root
//可以看到存在一个标识号为1的作业
[root@PC-LINUX ～]# atrm 1
[root@PC-LINUX ～]# at -l
//可以看到该作业已经删除
```

10.5 Linux系统启动

Linux系统的启动其实是一个复杂的过程，本节介绍Linux的启动过程以及systemd进程。

10.5.1 Linux系统启动过程

Linux系统的启动是从计算机开机通电自检开始直到登录系统需要经过的多个过程。下面详细讲述它的启动过程。

1. BIOS自检

计算机在接通电源之后首先由BIOS进行POST自检，然后依据BIOS内设置的引导顺序从硬盘、软盘或CDROM中读入引导块。

2. 启动GRUB 2

GRUB 2是引导加载程序，引导加载程序是用于引导操作系统启动。当机器引导它的操作系统时，BIOS会读取引导介质上最前面的512字节（主引导记录）。

3. 加载内核

接下来的步骤就是加载内核映像到内存中，内核映像并不是一个可执行的内核，而是一个压缩过的内核映像。

4. 执行systemd进程

systemd进程是系统所有进程的起点，内核在完成核内引导以后，即在本进程空间内加载systemd程序，它的进程号是1。

5. 执行/bin/login程序

login程序会提示使用者输入账号及密码，接着编码并确认密码的正确性，如果账号与密码相符，则为使用者初始化环境，并将控制权交给shell，即等待用户登录。

10.5.2 systemd进程简介

systemd是Linux系统执行的第一个进程，进程ID为1，是系统所有进程的起点，主要用来执行一些开机初始化脚本和监视进程。

Linux系统在完成核内引导以后就开始运行systemd程序。

1. Systemd简介

systemd是Linux下的一种init软件，在LGPL（GNU宽通用公共许可证）2.1及其后续版本许可证下开源发布，其开发目标是提供更优秀的框架以表示系统服务间的依赖关系，并依次实现系统初始化时服务的并行启动，同时达到降低Shell的系统开销的效果，最终代替现在常用的System V与BSD风格init程序。

systemd开启和监督整个系统是基于unit（单元）的概念。Unit是由一个与配置文件对应的名字和类型组成的（例如：httpd.sevice unit有一个具有相同名字的配置文件，是守护进程httpd的一个封装单元）。

Unit有以下几种类型：

（1）service：守护进程的启动、停止、重启和重新加载是此类unit中最为明显的几个类型；

（2）socket：此类unit封装系统和互联网中的一个套接字；

（3）device：此类unit封装一个存在于Linux设备树中的设备；

（4）mount：此类unit封装系统结构层次中的一个挂载点；

（5）automount：此类unit封装系统结构层次中的一个自挂载点；

（6）target：此类unit为其他unit进行逻辑分组；

（7）snapshot：与target unit相似，快照本身不做什么，唯一的目的就是引用其他unit。

2. 修改默认开机运行级别

在以前Fedora系统中都是通过修改/etc/inittab文件中的id:x:initdefault来改变默认运行级别。Fedora17放弃了在/etc/inittab里控制启动状态，使用systemd创建符号链接的方式指向默认的运行级别。

在systemd的管理体系里面，以前的运行级别（runlevel）的概念被新的运行目标（target）所取代。target的命名类似于multi-user.target等这种形式，如原来的运行级别3（runlevel 3）就对应新的多用户目标（ multi-user.target ），runlevel5就相当于（graphical.target）。

在/lib/systemd/system/下面定义runlevelX.target文件目的是为了能够兼容以前的运行级别的管理方法。

```
runlevel0.target -> poweroff.target
runlevel1.target -> rescue.target
runlevel2.target -> multi-user.target
runlevel3.target -> multi-user.target
runlevel4.target -> multi-user.target
runlevel5.target -> graphical.target
runlevel6.target -> reboot.target
```

修改运行级别时只需对它们建立到/etc/systemd/system/default.target的软链接即可。比如定义运行级别为5：

（1）首先删除原来的文件。

```
rm /etc/systemd/system/default.target
```

（2）建立运行级别链接。

```
ln -s /lib/systemd/system/runlevel5.target /etc/systemd/system/default.target
```

或者执行命令：systemctl enable graphical.target

（3）重启计算机。

```
init 6
```

3. 单元控制

在systemd管理体系里，称呼需要管理的守护进程为单元（unit），对单元（unit）的管理是通过命令systemctl来进行控制的。

下面举例说明在multi-user.target环境下启动与关闭httpd.service：

（1）开机自动启动httpd.service服务执行下面命令：

```
systemctl enable httpd.service
```

命令执行后会在/etc/systemd/system/multi-user.target.wants/下创建httpd.service软链接；

（2）开机自动关闭httpd.service服务执行下面命令：

```
systemctl disable httpd.service
```

命令执行后会删除/etc/systemd/system/multi-user.target.wants下的httpd.service软链接；

10.6 维护GRUB 2

使用引导加载程序可以引导操作系统的启动，一般情况下引导加载程序都安装在MBR（主引导扇区）中，本节主要讲述Linux系统中GRUB的维护。

10.6.1 引导加载程序和GRUB

在Linux系统中使用的引导装载程序主要是GRUB和LILO，但是目前主流的Linux发行版本多数采用GRUB。

1. 引导加载程序简介

当计算机要引导操作系统时，BIOS会读取引导介质上最前面的MBR记录。在单一的MBR中只能存储一个操作系统的引导记录，当需要多个操作系统时就会出现问题，所以需要更灵活的引导加载程序。

主引导记录本身要包含两类内容：引导加载程序和分区表。

2. GRUB概述

从Red Hat Linux 7.2起，GRUB取代LILO成为默认的启动加载程序。

GNU GRUB（GRand Unified Bootloader）是一个将引导加载程序安装到主引导记录的程序，主引导记录是位于一个硬盘开始的扇区。

GRUB支持直接和链式加载的引导方法。

3. GRUB特性

GRUB包含许多特性，这使得GRUB比其他可用的引导加载程序更加优越，下面列出一些比较重要的特性：

（1）GRUB在x86机器上提供一个真正基于命令行的、先于操作系统启动的环境。

（2）GRUB支持逻辑块寻址（LBA）方式。

（3）GRUB的配置能在每次系统引导时被读取。

4. GRUB 2新功能

GRUB2是GRUB的升级版，它实现了一些GRUB中所不具备的功能。

（1）图形接口；
（2）使用了模块机制；
（3）支持脚本语言；
（4）支持救援模式；
（5）国际化语言；
（6）有一个灵活的命令行接口；
（7）针对文件系统、文件、设备、驱动、终端、命令、分区表、os loader的模块化、层次化、基于对象的框架；
（8）支持多种文件系统格式；
（9）可访问已经安装的设备上的数据；
（10）支持自动解压。

5. 设备在GRUB 2中的命名

GRUB 2中设备fd表示软盘，hd表示硬盘（包含IDE和SCSI硬盘），设备是从0开始编号，分区是从1开始，主分区从1-4，逻辑分区从5开始。

（fd0）：表示整个软盘。

（hd0，1）：表示BIOS中的第1个硬盘的第1个分区。

（hd0，5）/boot/vmlinuz：表示BIOS中的第1个硬盘的第1个逻辑分区下的boot目录下的vmlinuz文件。

Grub 2的主要配置文件是/boot/grub2/grub.cfg，不要直接编辑此文件。而使用grub2-mkconfig工具生成该配置文件。

grub2-mkconfig将/etc/default/grub文件（包含默认菜单超时、内核命令行选项等）和/etc/grub.d/目录中的一组脚本生成配置文件。

```
[root@PC-LINUX ～]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

10.7 查看系统信息

本节主要讲述如何使用命令查看内存信息、查看磁盘空间占用情况以及显示目录或文件的大小。

10.7.1 查看内存信息

可以使用free命令显示系统的物理内存和交换分区的总量，以及已使用的、空闲的、共享的、在内核缓冲内的和被缓存的内存数量。使用free命令可以显示计算机系统的内存容量。

命令语法：

free [选项][-s <间隔秒数>]

【例10.45】查看系统的物理内存和交换分区的总量。

```
[root@PC-LINUX ~]# free
             total       used       free     shared    buffers     cached
Mem:       1027848     921204     106644          0     212716     408036
-/+ buffers/cache:     300452     727396
Swap:      2097148          0    2097148
```

【例10.46】 以MB为单位查看系统的物理内存和交换分区的总量。

```
[root@PC-LINUX ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          1003        899        104          0        207        398
-/+ buffers/cache:        293        710
Swap:         2047          0       2047
```

【例10.47】 显示系统的物理内存和交换分区容量的总和。

```
[root@PC-LINUX ~]# free -t
             total       used       free     shared    buffers     cached
Mem:       1027848     921460     106388          0     212940     408036
-/+ buffers/cache:     300484     727364
Swap:      2097148          0    2097148
Total:     3124996     921460    2203536
```

10.7.2 查看磁盘空间占用情况

使用df命令可以显示磁盘的相关信息。检查文件系统的磁盘空间占用情况，利用该命令获取硬盘占用了多少空间、目前还剩下多少空间等相关信息。

命令语法：

df [选项][--block-size=<区块大小>][-t <文件系统类型>][-x <文件系统类型>][文件或设备]

【例10.48】 显示系统的磁盘空间使用量。

```
[root@PC-LINUX ~]# df
文件系统          1K-块    已用    可用 已用% 挂载点
rootfs        15674684 9193680 5694572   62% /
devtmpfs        504956       0  504956    0% /dev
tmpfs           513924     228  513696    1% /dev/shm
tmpfs           513924    1328  512596    1% /run
/dev/sda2      15674684 9193680 5694572   62% /
tmpfs           513924       0  513924    0% /sys/fs/cgroup
tmpfs           513924       0  513924    0% /media
/dev/sda1       100604   43739   51745   46% /boot
// /dev/shm代表系统的虚拟内存文件系统。
```

【例10.49】 以MB和GB为单位显示系统的磁盘空间使用量。

```
[root@PC-LINUX ~]# df -h
文件系统        容量  已用  可用 已用% 挂载点
rootfs          15G  8.8G  5.5G   62% /
devtmpfs       494M     0  494M    0% /dev
tmpfs          502M  228K  502M    1% /dev/shm
tmpfs          502M  1.3M  501M    1% /run
/dev/sda2       15G  8.8G  5.5G   62% /
tmpfs          502M     0  502M    0% /sys/fs/cgroup
tmpfs          502M     0  502M    0% /media
/dev/sda1       99M   43M   51M   46% /boot
```

【例10.50】 显示ext4文件系统类型磁盘空间使用量。

```
[root@PC-LINUX ~]# df -t ext4
文件系统         1K-块    已用    可用 已用% 挂载点
/dev/sda2     15674684 9193680 5694572   62% /
/dev/sda1       100604   43739   51745   46% /boot
```

【例10.51】在显示磁盘空间使用量时也显示文件系统类型。

```
[root@PC-LINUX ~]# df -T
文件系统      类型       1K-块    已用    可用 已用% 挂载点
rootfs        rootfs   15674684 9193680 5694572   62% /
devtmpfs      devtmpfs   504956       0  504956    0% /dev
tmpfs         tmpfs      513924     228  513696    1% /dev/shm
tmpfs         tmpfs      513924    1328  512596    1% /run
/dev/sda2     ext4     15674684 9193680 5694572   62% /
tmpfs         tmpfs      513924       0  513924    0% /sys/fs/cgroup
tmpfs         tmpfs      513924       0  513924    0% /media
/dev/sda1     ext4       100604   43739   51745   46% /boot
```

【例10.52】 以MB和GB为单位显示磁盘空间使用量时也显示文件系统类型。

```
[root@PC-LINUX ~]# df -hT
文件系统       类型      容量  已用  可用 已用% 挂载点
rootfs         rootfs     15G  8.8G  5.5G   62% /
devtmpfs       devtmpfs  494M     0  494M    0% /dev
tmpfs          tmpfs     502M  228K  502M    1% /dev/shm
tmpfs          tmpfs     502M  1.3M  501M    1% /run
/dev/sda2      ext4       15G  8.8G  5.5G   62% /
tmpfs          tmpfs     502M     0  502M    0% /sys/fs/cgroup
tmpfs          tmpfs     502M     0  502M    0% /media
/dev/sda1      ext4       99M   43M   51M   46% /boot
```

【例10.53】 显示磁盘空间i节点使用情况。

```
[root@PC-LINUX ~]# df -i
文件系统        Inode 已用(I) 可用(I) 已用(I)% 挂载点
rootfs         983040  329578  653462      34% /
devtmpfs       126239     392  125847       1% /dev
tmpfs          128481       6  128475       1% /dev/shm
tmpfs          128481     462  128019       1% /run
/dev/sda2      983040  329578  653462      34% /
tmpfs          128481      12  128469       1% /sys/fs/cgroup
tmpfs          128481       1  128480       1% /media
/dev/sda1       25688     339   25349       2% /boot
```

10.7.3 显示目录或文件的大小

使用du命令可以显示目录或文件的大小。
命令语法：
du［选项］［文件名或目录名］

【例10.54】分屏显示每个目录或文件的大小。

```
[root@PC-LINUX ~]# du |more
8       ./.mission-control/accounts
12      ./.mission-control
8       ./.gconf/apps/nm-applet
8       ./.gconf/apps/evolution/calendar
12      ./.gconf/apps/evolution
24      ./.gconf/apps
28      ./.gconf
712     ./.cache/ibus/bus
716     ./.cache/ibus
4       ./.cache/libsocialweb/cache
8       ./.cache/libsocialweb
12      ./.cache/zeitgeist
……
```

【例10.55】 显示文件/etc/inittab的大小。

```
[root@PC-LINUX ~]# du /etc/inittab
4       /etc/inittab
```

【例10.56】 显示/root目录所占用的数据块总数。

```
[root@PC-LINUX ~]# du –s /root
6748    .
```

【例10.57】 以MB为单位显示/root目录所占用的数据块总数。

```
[root@PC-LINUX ~]# du –sh /root
6.6M
```

小　结

目前在众多的Linux系统上多采用RPM软件包，对于终端用户来说，RPM简化了系统安装、卸装、更新和升级的过程，只需要使用简短的命令就可完成。RPM维护一个已安装软件包和它们的文件的数据库，因此，可以在系统上使用查询和校验软件包功能。RPM主要有5种基本操作模式：安装、卸装、刷新、升级、查询。

小　结

Linux系统下最常用的打包程序是tar，使用tar程序打出来的包称为tar包，tar包文件的名称通常都是以.tar结尾的。生成tar包后，就可以用其他的程序来进行压缩了。利用tar，用户可以为某一特定文件创建备份，也可以在备份中改变文件，或者向备份中加入新的文件。

小　结

Linux系统用分时管理方法使所有的任务共同分享系统资源。Linux系统有好几种进程，每一种都有其各自的特性。要查看Linux系统中的进程信息可以使用ps和top这两个命令。要关闭某个应用程序可以通过结束其进程的方式实现，如果进程一时无法结束，可以将其强制结束。启动一个进程有两种主要途径：手工启动和调度启动，后者是事先进行设置，根据用户要求自行启动。

小　结

如果要在固定的时间上触发某个作业，就需要创建任务计划，按时执行该作业，在Linux系统中常用cron和at实现该功能。使用cron实现任务自动化可以通过修改/etc/crontab文件以及使用crontab命令实现，其结果是一样的。cron命令被用来调度重复的任务，而at命令被用来在指定时间内调度一次性的任务。一旦一个作业被提交，at命令将会保留所有当前的环境变量，包括路径，该作业的所有输出都将以电子邮件的形式发送给用户。

小　结

Linux系统的启动其实是一个复杂的过程，分别要经过BIOS自检、启动GRUB2、加载内核、执行systemd进程以及执行/bin/login程序等步骤。systemd是Linux系统执行的第一个进程，进程ID为1，是系统所有进程的起点，主要用来执行一些开机初始化脚本和监视进程。Linux系统在完成核内引导以后就开始运行systemd程序。

小　结

使用引导加载程序可以引导操作系统的启动，一般情况下引导加载程序都安装在MBR（主引导扇区）中。在Linux系统中使用的引导装载程序主要是GRUB和LILO，但是目前主流的Linux发行版本多数采用GRUB。

小　结

GRUB 2的配置是通过位于/boot/grub2/grub.conf的一个配置文件来完成的。

小　结

使用free命令显示系统的物理内存和交换分区的总量，以及已使用的、空闲的、共享的、在内核缓冲内的和被缓存的内存数量。

使用df命令可以显示磁盘的相关信息，检查文件系统的磁盘空间占用情况，利用该命令获取硬盘占用了多少空间、目前还剩下多少空间等相关信息。

使用du命令可以显示目录或文件的大小。