# Unix/Linux操作系统笔记

# 1. 第一章

开源软件：所有人都可以获得软件和源码

自由软件基金会提出的自由软件运动：

- 出于任何目的自由地运行程序
- 自由地学习和修改源码
- 自由地重新发布程序
- 自由地创建衍生的程序

## 1.1 概述

### 1.1.1 Linux的诞生

GNU是“GNU's Not UNIX”的递归缩写。

GNU是自由软件基金会发起的一项计划：

- 也是一个组织
- 也是一种操作系统
- 也是一类规范

GNU计划下的任何软件，不只提供软件使用权，也提供软件源代码；当使用者对于GNU计划下的软件进行修改时，仍必须维持GNU的精神，即修改后的软件也应该无条件地奉献。

Linux是操作系统，严格意义上是GNU/Linux操作系统。1991年10月5日，**Linus Torvalds**正式对外宣布Linux内核系统诞生(Free minix-like kernel sources for 386-AT)。

### 1.1.2 Linux 内核与版本

1994年3月14日，发布了1.0版本的内核。采用GPL（通用公共许可）协议开放源代码。

Linux内核版本采用双轨制：稳定版和开发版。

Linux内核的命名格式：num1.num2.num3或者num1.num2.num3-num4

num1是主版本号，num2是次版本号，num3是修订版本号，num4是补丁号。如果次版本号是偶数，那么该内核就是稳定版的；若是奇数，则是开发版的。

### 1.1.3 Linux的发行版本

一些组织或厂商将Linux系统的内核与系统应用程序、说明文档包装起来，并提供若干系统安装界面和系统配置、设定、与管理工具，就构成了一种发行版本，实际上就是Linux核心加上一系列的系统程序组成的一个大软件包。

主要的发行版有：Red Hat、CentOS、Slackware、Debian、Ubuntu、archlinux、Deepin、Red Falg/红旗等等。

### 1.1.4 Linux的特点

1. 多用户
2. 多任务
3. 移植性
4. 开放性
5. 稳定性
6. 安全性
7. 设备独立性

8. 强大的网络功能

## 1.2 Linux安装

此节网络上各类发行版教程丰富，不再赘述。

# 2. 系统的启动与关闭

## 2.1 概述

内核的分类：

- 宏内核：独立的大进程，所有子系统都打包在一个文件中，每个函数都可以访问其他部分。（UNIX/Liunx）
- 微内核：各个独立进程

操作系统的运行空间：用户空间与内核空间，系统在运行过程中不断在内核态与用户态之间切换。

## 2.2 系统的引导配置

Linux多用GRUB来引导，相关资料网络上较为丰富，不再赘述。

## 2.3 系统的运行级

- 运行级0：关闭计算机
- 运行级1：单用户模式（系统维修模式）
- 运行级2：多用户模式不带网络文件系统NFS（不常见）
- 运行级3：**带有网络文件系统的多用户模式**（最常用）
- 运行级4：系统保留
- 运行级5：**图形方式**
- 运行级6：重新启动

### 2.3.1 运行级的切换

```
# 切换为多用户模式
systemctl isolate multi-user.target
systemctl isolate runlevel3.target
init 3

# 切换为图形模式
systemctl isolate graphical.target
init 5

# 设置默认模式为
systemctl set-default multi-user.targrt

# 查看默认模式
sysytemctl get-default
```

Fence 1

### 2.3.2 查看当前运行级

`runlevel`命令

### 2.3.3 系统运行级的服务

通过命令`chkconfig -list`命令查看服务在各运行级下是否启动。

常见服务有`keytable`（键盘服务）、`atd`(定时器)等。

## 2.4 系统的启动和引导过程

### 2.4.1 系统引导步骤

1. BIOS自检
2. MBR中的GRUB启动
3. Linux 操作系统内核运行（加载内核）
4. 启动系统第一个进程`systemd`
5. 用户登录

MBR:Master Boot Recode 主引导记录。MBR格式的磁盘分区表只支持最大4个分区，可以创建3个主分区和1个扩展分区；可以在扩展分区中创建更多的逻辑分区。

### 2.4.2 加载内核

GRUB是一个引导程序装载器，会完成用户信息的显示、操作系统的选择、命令行参数的传递，然后加载对应的操作系统的内核映像文件，如vmlinuz，最后将控制权交给已经被载入到内存的操作系统映像文件。

### 2.4.3 内核运行

Linux的内核被加载到内存后，先进行自解压到/boot目录，检测电脑各设备并加载驱动，然后启动`systemd`进程。

### 2.4.4 系统的关闭

1. `shutdown`命令

```
# 参数
-t sec 指定秒数后关机
-h 系统关闭后停机
-r 重启

# 实例
shutdown -h now # 立即关机
shutdown -h +5 # 5min后关机
```

Fence 2

2. `poweroff`命令
3. `halt`命令

```
halt -p # 关闭后断电
```

Fence 3

4. `init`命令

```
init 0 # 关机
init 6 # 重启
```

Fence 4

5. reboot命令（只重新启动，不关机）

# 3. 用户登录与账号管理

## 3.1 用户登录

### 3.1.1 终端登录

Linux系统提供了6个虚拟控制台终端，按`Ctrl+Alt+F1~F6`切换。

### 3.1.2 远程登录

- telnet命令：`telnet host`
- ssh命令：`ssh user@hostname`

## 3.2 管理用户账号

### 3.2.1 添加用户

#### 3.2.1.1 命令添加

使用`useradd [-d home_dir] [-s shell] username`命令添加用户。

```
useradd zxj # 添加一个用户名为zxj的用户
```

Fence 5

| 命令参数 | 选项说明 |
| --- | --- |
| -c 备注 | 给用户添加备注，保存在passwd文件的备注栏中 |
| -g 用户群组 | 指定所属群组，不写则为用户名同名组 |
| -G 用户群组 | 指定附加群组 |
| -p 口令 | 指定密码 |
| -s shell | 指定该用户默认Shell程序，如`/bin/bash` |
| -u 用户id | 指定用户id,1-999保留，自动取递增值,最大60000 |

使用`passwd [username]`命令设置密码，用户名省略则为当前用户设置密码。

| 命令参数 | 选项说明 |
| --- | --- |
| -l | 锁住账号，只有超级用户可以使用 |
| -u | 解锁账号，只有超级用户可以使用 |
| -S | 列出信息，只有超级用户可以使用 |

#### 3.2.1.2 修改文件添加

用`useradd`命令添加用户实际上就是向用户信息文件（只存信息不存密码）/etc/passwd文件、影子文件（存密码）/etc/shadow文件、组信息文件/etc/group文件中写入信息。

passwd文件格式：

| 用户名 | 密码 | 用户ID | 组ID | 用户全称 | 家目录 | shell |
|---|---|---|---|---|---|---|
| Login name | passwd | user ID | group ID | user full name | home directory | login shell |

以`:`分隔，每行一个用户

user ID为0的是root用户，1-999是系统用户，其他为普通用户。

shadow文件格式：

| 用户名 | 密码 | 创建时间 | 修改密码时间间隔 | 密码过期宽限天数 | 失效时间 | 保留 |
|---|---|---|---|---|---|---|
| | 密码加密存储，`*`表示无密码 | 时间戳表示 | 0 | 99999（不失效） | | |

## 3.3 删除用户

使用`userdel [-r] username`命令删除用户，`-r`选项表示递归删除该用户家目录下所有文件。

## 3.4 修改用户属性

使用`usermod`命令，选项与`useradd`和`passwd`命令一致。

## 3.5 管理群组账号

### 3.5.1 添加用户群组

#### 3.5.1.1 终端添加

使用`groupadd groupname`命令添加用户群组。

group文件格式：

| 用户组名 | 组密码 | 组ID | 用户列表 |
|---|---|---|---|
| group name | passwd | group ID | user list |

## 3.6 查看登录用户

1. 使用`who`命令
2. 使用`w`命令，不仅显示有那些用户登录到该系统，还显示用户正在执行的程序、现在的系统时间、系统已启动多久、目前共有多少用户、过去一段时间内的 平均负载程度。
3. 使用`finger`命令，查找并显示用户信息。

## 3.7 改变用户身份

### 3.7.1 `su`命令

`su`命令只会临时切换到超级用户，使用exit退出

```
# 选项
-c command # 执行一条命令后返回
-l # 同时改变工作目录及环境变量
```

Fence 6

### 3.7.2 `sudo`命令

命令前加`sudo`则以超级用户身份执行该条命令。

可以使用`sudo`命令的用户配置写在/etc/sudoers中，可以使用vim直接修改或者使用`visudo`修改。

# 4. 文件系统管理

## 4.1 文件与文件系统

### 4.1.1 文件类型

- 普通文件（f）
- 目录文件（d）
- 链接文件（l）
    - 硬链接`ln`命令创建，直接指向文件存储区块
    - 软连接`ln -s`命令创建，指向文件节点号，节点号再指向存储区块，类似于windows快捷方式
- 设备文件（c）（/dev目录下）

### 4.1.2 文件权限

文件与用户的关系：

- 文件所有者（owner）
- 同组用户（group）
- 其他人（other）

文件的权限：

- 可读（r）
- 可写（w）
- 可执行（x）

使用文件类型+所有者权限+同组权限+其他人权限表示一个文件的权限控制。

并且可将权限按照与用户的关系三位一组用二进制表示。常见权限644、755、777等。

```
❯ ls
a.txt  b.txt  c.sh
❯ ll
总计 0
-rw-r--r-- 1 ldsr ldsr 0 12月27日 10:58 a.txt # 644
-rwxr-xr-x 1 ldsr ldsr 0 12月27日 10:59 b.txt # 755
-rwxrwxrwx 1 ldsr ldsr 0 12月27日 11:00 c.sh  # 777
```

Fence 7

### 4.1.3 特殊文件权限

在文件用户的关系最前面加一组特殊权限控制，分别为：

- SUID 用户ID：开启后任意用户可访问文件所有权限（开启二进制表示：4755）
- SGID 组ID：只对目录有效，开启后目录内的所有操作都被设为目录的所属组操作（开启二进制表示：2755）
- Sticky 粘附：只对目录有效，开启后即便用户有权限只可操作拥有者为自己的文件（开启二进制表示：1755）

### 4.1.4 文件系统

文件系统是操作系统中实现对文件的组织、管理和存取的一组系统程序和数据结果，或者说它是管理软件资源的软件，对于用户来说它提供了一中便捷地村组信息的方法。

**虚拟文件系统**：位于层次结构的最上层，是用户与逻辑文件系统交互的接口。

**逻辑文件系统**：

- ext系列：ext2、ext3、ext4等
- xfs
- brtfs

## 4.2 Linux目录

```
❯ cd /
❯ ll
总计 17G
lrwxrwxrwx   1 root root    7 11月21日 16:56 bin -> usr/bin
drwxr-xr-x   1 root root  188 12月23日 19:15 boot
drwxr-xr-x  21 root root 4.5K 12月26日 22:04 dev
drwxr-xr-x   1 root root 4.0K 12月27日 10:34 etc
drwxr-xr-x   1 root root    8 11月 1日 10:27 home
lrwxrwxrwx   1 root root    7 11月21日 16:56 lib -> usr/lib
lrwxrwxrwx   1 root root    7 11月21日 16:56 lib64 -> usr/lib
drwxr-xr-x   1 root root    6 12月15日 17:31 mnt
drwxr-xr-x   1 root root  174 12月15日 17:51 opt
dr-xr-xr-x 433 root root    0 12月27日 06:04 proc
drwxr-x---   1 root root  274 12月15日 17:44 root
drwxr-xr-x  26 root root  680 12月27日 10:01 run
lrwxrwxrwx   1 root root    7 11月21日 16:56 sbin -> usr/bin
drwxr-xr-x   1 root root   14 10月 9日 21:41 srv
dr-xr-xr-x  13 root root    0 12月27日 06:04 sys
drwxrwxrwt  20 root root  600 12月27日 11:16 tmp
drwxr-xr-x   1 root root   86 12月27日 10:34 usr
drwxr-xr-x   1 root root  126 12月26日 22:04 var
-rw-------   1 root root  16G 10月 9日 22:10 swapfile
```

Fence 8

### 4.2.1 基本目录

1. /bin目录：大多是二进制可执行程序，如cp、date、ls、rm、mv各种命令等
2. /sbin目录：与/bin目录类似，存放二进制可执行程序、shell脚本、符号连接文件等
3. /lib目录：动态连接库程序、shell脚本程序和一些符号连接文件。
4. /boot目录：系统引导、启动使用的一些文件和目录，如grub、vmlinuz内核压缩文件和内核映像文件
5. /root目录：超级用户目录
6. /mnt目录：临时挂载设备使用的目录
7. /tmp目录：临时文件目录
8. /opt 目录：一些软件会安装到这里
9. /etc目录：保存着关系系统运行方式的重要配置文件
10. /dev目录：系统中所有的块设备和字符设备的文件

### 4.2.2 特殊目录

1. /home目录：存放普通用户家目录
2. /usr目录：许多应用程序默认安装在该目录或该目录的一些子目录下
3. /var目录：存放系统运行中随时要改变的数据，如系统日志文件、系统临时文件等
4. /porc目录：保存的是系统的动态信息，包括cpu、内存、硬盘分区表、中断、系统各种外设等计算机硬件信息，以及内存资源和外设的使用情况，系统中所有当前进程的工作情况等。

## 4.3 文件系统维护

### 4.3.1 创建文件系统

使用`fdisk`命令创建磁盘分区，`fdisk`采用一种问答式界面。

```
# 命令格式
fdisk device

# 示例
fdisk /dev/sda1 # sata硬盘

fdisk /dev/nvme0n1p1  # nvme硬盘
```

Fence 9

| 命令 | 功能 |
| --- | --- |
| p | 显示当前磁盘分区表 |
| n | 添加一个新分区 |
| w | 保存更改并退出 |
| q | 不保存退出 |

使用`mkfs`建立文件系统`mkfs [-t type] filesys`

```
mkfs.ext4
mkfs.btrfs

# 示例
mkfs.btrfs /dev/sda1 # 给sda1格式化为btrfs文件系统
```

Fence 10

### 4.3.2 安装与卸载文件系统

安装实际上是把文件系统挂载到Linux目录的节点上，即某个路径。

使用`mount`命令挂载设备，使用`umount`命令卸载设备。

```
mount [-t type] device dir
umount dir | device
# 示例
mount -t ext4 /dev/sda1 /mnt
umount /mnt
umount /dev/sda1
```

Fence 11

# 5. 磁盘文件与目录管理

## 5.1 磁盘管理

`df`命令（disk free,磁盘剩余空间）查看计算机系统上每个文件系统的磁盘空间使用情况。

```
# 选项
-h 以容易理解的单位显示

# 示例
› df -h
文件系统         大小  已用  可用 已用% 挂载点
dev              7.7G     0  7.7G    0% /dev
run              7.7G  2.0M  7.7G    1% /run
efivarfs         268K  161K  103K   62% /sys/firmware/efi/efivars
/dev/nvme0n1p5   116G  107G  6.0G   95% /
tmpfs            7.7G   66M  7.7G    1% /dev/shm
tmpfs            7.7G   16M  7.7G    1% /tmp
tmpfs            1.0M     0  1.0M    0% /run/credentials/systemd-
journald.service
/dev/nvme0n1p5   116G  107G  6.0G   95% /home
/dev/nvme0n1p1   256M   48M  209M   19% /boot/efi
tmpfs            1.0M     0  1.0M    0% /run/credentials/getty@tty3.service
tmpfs            1.6G  140K  1.6G    1% /run/user/1000
/dev/nvme0n1p4   176G  129G   48G   74% /run/media/ldsr/Data
```

Fence 12

`du`命令(disk usage，磁盘使用量)，查看计算机系统上每个目录或文件战役磁盘空间的情况。

```
# 选项
-h 以容易理解的单位显示大小
du [option] [file]
```

Fence 13

## 5.2 文件与目录管理

### 5.2.1 目录切换

`cd [dir]`

父目录用`..`表示，当前目录用`.`表示。

dir选项可以是相对路径也可以是绝对路径。

### 5.2.2 显示当前目录

`pwd`命令显示当前所在目录的绝对路径。

### 5.2.3 建立目录或文件

`mkdir [option] directory`命令在指定路径下建立目录。默认权限755。

`touch`命令可以创建文件，包括文件名和文件后缀，默认权限644。

```
# 选项
-p # 如果要建立的上级目录不存在，则一并建立
-m # 创建时设置权限

# 示例
mkdir test
mkdir -p ~/father/son
touch a.txt
```

Fence 14

### 5.2.4 复制文件或目录

`cp`命令将指定文件或者目录复制到指定目录下。

```
cp [option] source dest
# 选项
-R或-r # 递归处理，复制子目录下的所有文件和目录
-S # 修改源后缀
cp -S  bak  sudo.txt  sd.txt
```

Fence 15

### 5.2.5 移动文件或目录

`mv`命令将文件或目录移动到指定文件。也可用于文件或目录更名。

```
mv [option] source dest
# 选项
-R或-r # 递归处理，移动子目录下所有文件和目录
-S # 修改源后缀
mv -b -S  bak  sudo.txt  sd.txt
```

Fence 16

### 5.2.6 删除文件或目录

`rm`命令可以删除文件或目录，要求用户对要删掉的文件有写权限，对目录有写和执行权限。

```
rm [option] file
# 选项
-r # 递归处理
-f # 强制删除
-d # 删除目录

# 示例
rm a.txt
rm -rf test/
```

Fence 17

### 5.2.7 显示目录内容

`ls`列出路径下的文件和目录。

| 命令选项 | 选项说明 |
| --- | --- |
| -l | 显示详细信息 |
| -a | 显示所有文件 |

```
ls [option] [file]

# 示例
❯ ls
a.txt  b.txt  c.sh
❯ ls -a
.  ..  a.txt  b.txt  c.sh
❯ ls -l
总计 0
-rw-r--r-- 1 ldsr ldsr 0 12月27日 10:58 a.txt
-rwxr-xr-x 1 ldsr ldsr 0 12月27日 10:59 b.txt
-rwxrwxrwx 1 ldsr ldsr 0 12月27日 11:00 c.sh
```

Fence 18

## 5.3 查找文件

### 5.3.1 `find`命令

find命令会在执行是扫描指定的目录读取文件。

| 选项命令 | 命令说明 |
| --- | --- |
| -amin n | n分钟前访问过的文件或目录 |
| -atime n | n天前访问过的文件或目录 |
| -cmin n | n分钟前修改过的文件或目录 |
| -ctime n | n天前修改过的文件或目录 |
| -mmin n | 指定时间被修改过的文件或目录，单位为min |
| -mtime n | 指定时间被修改过的文件或目录，单位为min |
| -name filename | 匹配文件名 |
| -size [bckw] | 指定大小的文件;b表示块，单位512B、c表示以B为单位、k表示以KB为单位、w表示以两位字节为单位 |
| !expr或-not expr | 非运算 |
| expr1 -a expr2或<br>expr1 -and expr2 | 与运算 |
| expr1 -o expr2 或<br>expr1 -or expr2 | 或运算 |
| -exec command | 找到文件之后之后执行命令，命令格式为`-exec command {} \;` |

```
find [path] [-name filename]
find [path] [expression] [exec]

# 示例
find / -size +204800 # 在根目录下寻找大于100MB的文件
find /usr -name *.c -print # 把找到的文件输出到标准输出
find /tmp -ctime +3 -user joe -ok rm {} \ # 提示删除存在时间超过3天以上的joe的临时文件
find /data -type f -perm 644 -name "*.sh" -exec chmod 755 {} \; # 查找/data下的权限为644，后缀为sh的普通文件，增加执行权限
```

Fence 19

### 5.3.2 `locate`命令

`locate`命令会定期构建文件数据库，从数据库中搜索索引，所以速度比`find`更快。

`localte [option] expr`

### 5.3.3 其他查找命令

`which`：搜索命令路径及别名

`whereis`：搜索命令路径及帮助信息

```
❯ which ls
ls: aliased to ls --group-directories-first --color=auto
❯ which python
/usr/bin/python
❯ whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
❯ whereis python
python: /usr/bin/python /usr/share/man/man1/python.1.gz
```

Fence 20

## 5.4 修改文件权限

使用`chmod [-R]`命令可以设置文件或目录的权限。`-R`参数表示递归处理子目录及文件。

权限表示方式通常为数字表示或者ugo表示。

数字表示法使用四组三位二进制表示，在转成十进制便于记忆。第一组表示SUID、SGID、Sticky，第二组表示owner（user）的读、写、执行，第三组表示group的读、写、执行，第四组表示other的读、写、执行。

ugo表示法使用u、g、o缩写表示owner（user）、group、other，也可使用a指代三个全部；使用+或-赋予或者收回权限，也可以使用=指定权限；使用r、w、x表示读、写、执行权限。

文件夹的执行权限表示进入该目录的权限。

```
❯ chmod 0000 test.txt
'test.txt' 的模式已由 1777 (rwxrwxrwt) 更改为 0000 (---------)
❯ chmod 0755 test.txt
'test.txt' 的模式已由 0000 (---------) 更改为 0755 (rwxr-xr-x)
❯ chmod g+w test.txt
'test.txt' 的模式已由 0755 (rwxr-xr-x) 更改为 0775 (rwxrwxr-x)
❯ chmod a-r test.txt
'test.txt' 的模式已由 0775 (rwxrwxr-x) 更改为 0331 (-wx-wx--x)

❯ chmod 777 test.txt
'test.txt' 的模式已保留为 0777 (rwxrwxrwx)
❯ chmod 4777 test.txt
'test.txt' 的模式已由 0777 (rwxrwxrwx) 更改为 4777 (rwsrwxrwx)
❯ chmod 2777 test.txt
'test.txt' 的模式已由 4777 (rwsrwxrwx) 更改为 2777 (rwxrwsrwx)
❯ chmod 1777 test.txt
'test.txt' 的模式已由 2777 (rwxrwsrwx) 更改为 1777 (rwxrwxrwt)

❯ chmod u=rwx,g=r-x,o=r-x test.txt
'test.txt' 的模式已由 0331 (-wx-wx--x) 更改为 0744 (rwxr--r--)
```

Fence 21

除了更改文件权限外，还有更改文件所有者`chown`、更改文件所属组`chgrp`等命令。

## 5.5 查看文件

### 5.5.1 `cat`命令

`cat [option] [file]`

`cat -n`：显示行行号

`tca`：按行逆序显示内容

`head -n num file`：显示文件前num行

`tail -n num file`：显示文件最后num行

### 5.5.2 `more`命令

`more`命令的作用是分页显示文件内容，使用`-num`指定每页显示多少行。

```
❯ more -20 steam.desktop
[Desktop Entry]
Name=Steam (Runtime)
Comment=Application for managing and playing games on Steam
Comment[pt_BR]=Aplicativo para jogar e gerenciar jogos no Steam
Comment[bg]=Приложение за ръководене и пускане на игри в Steam
Comment[cs]=Aplikace pro spravování a hraní her ve službě Steam
Comment[da]=Applikation til at håndtere og spille spil på Steam
Comment[nl]=Applicatie voor het beheer en het spelen van games op Steam
Comment[fi]=Steamin pelien hallintaan ja pelaamiseen tarkoitettu sovellus
Comment[fr]=Application de gestion et d'utilisation des jeux sur Steam
Comment[de]=Anwendung zum Verwalten und Spielen von Spielen auf Steam
Comment[el]=Εφαρμογή διαχείρισης παιχνιδιών στο Steam
Comment[hu]=Alkalmazás a Steames játékok futtatásához és kezeléséhez
Comment[it]=Applicazione per la gestione e l'esecuzione di giochi su Steam
Comment[ja]=Steam 上でゲームを管理&プレイするためのアプリケーション
Comment[ko]=Steam에 있는 게임을 관리하고 플레이할 수 있는 응용 프로그램
Comment[no]=Program for å administrere og spille spill på Steam
Comment[pt_PT]=Aplicação para organizar e executar jogos no Steam
Comment[pl]=Aplikacja do zarządzania i uruchamiania gier na platformie
Steam
Comment[ro]=Aplicație pentru administrarea și jucatul jocurilor pe Steam
--更多--(18%)
```

Fence 22

## 5.6 文件压缩与解压缩、打包与解包

### 5.6.1 压缩

`gzip`与`gunzip`

之将文件体积缩，生成一个同名的.gz后缀的文件。

```
gzip /home/stu/*.txt # 压缩所有txt文件
gunzip -r /home/stu #解压目录下所有压缩文件，并删除压缩包
```

Fence 23

### 5.6.2 打包

`tar [option] [file]`

将多个文件打包成一个文件，后缀为.tar

| 功能选项 | 选项说明 |
| --- | --- |
| -c或--create | 建立新的备份文件 |
| -x或--extract或--get | 从备份中新建文件 |
| -f或--file | 指定备份文件 |
| -v或--verbose | 显示命令执行的详细过程 |
| -z或--gzip或--ungzip | 通过gzip命令处理备份文件 |

```
tar -czvf file.tar.gz file1 file2 file3
tar -zxvf file.tar.gz
```

Fence 24

### 5.6.3 `grep`命令

`grep PATTERN [FILE...]`

`grep`命令在文件中查找所需的信息（用字符串表示），一旦查找成功，grep命令将给出信息所在行的全部内容。

### 5.6.4 `uniq`命令

`uniq`命令用于去除重复行。

# 6. Linux的包管理

## 6.1 包管理概述

RPM原来是Red Hat Linux发行版专门用来管理Linux各项软件包的程序，是个开放的软件包管理器。

## 6.2 包管理特色

1. 易用性
2. 面向软件包
3. 包的升级性
4. 探测包的依赖性
5. 强大的查询能力
6. 软件包校验
7. 支持多重结构
8. 保持软件包原始特征

## 6.3 命令行下的包管理

`rpm [OPTION...] packge`

| 功能选项 | 选项说明 |
| --- | --- |
| -i或--install | 安装 |
| -h | 安装时输出“#”显示进度 |
| -v | 输出指令执行的详细信息 |
| -q或--query | 查询 |
| -l或--list package_name | 列出指定软件包中所有的文件 |
| -U或--upgrade | 升级 |
| -e或--erase | 删除 |

# 7. 进程管理

## 7.1 进程概述

### 7.1.1 进程的概念

进程是程序在某个数据集合上的运行活动，是系统进行资源分配和调度的一个独立单位。简单的说，进程是程序的一次执行。

### 7.1.2 进程与程序的区别

进程是由可执行成、该程序所需的相关数据集合和进程控制块（PCB）组成的。在PCB中包含进程表示（PID）、处理器状态信息、进程控制信息、操作系统通过进程控制块对进程加以管理。

进程与程序的主要区别是：

1. 进程是程序处理数据的过程、而程序只是彝族指令的有序集合。
2. 进程具有动态性、并发性、独立性和异步性，而程序只是静态的代码，不久有这些特性。
3. 程序与进程并非一一对应的，一个进程可能对应一个程序，也可能多个进程对应一个程序。

### 7.1.3 多任务的实现

- 进程调度，多级反馈队列
- 每个进程都有优先级，创建时加入到相应优先级队列
- 实时和系统进程：优先级和先来先服务相结合的调度算法
- 普通的用户进程：优先级与时间片轮转相结合的调度算法

进程与线程的区别：进程是操作系统资源分配的基本单位，而线程是处理器任务调度和执行的基本单位。
一个进程至少有一个线程，线程也被称为轻量级进程。

### 7.1.4 进程的类型

- 交互进程：它是由某种shell程序启动的进程，如执行一个命令。交互进程既可以在前台运行，也可以在后台运行。
- 批处理进程：这种进程和终端没有联系，但它是一系列进程。如执行一个shell脚本程序。
- 监控进程：也称为守护进程，它是在Linux 系统启动时运行的进程，并且运行在后台。

## 7.2 进程的启动

一般运行的命令即位前台启动，将命令置于后台可以在命令后加`&`。可以使用`jobs`命令查看后台进程列表。

在命令执行时，按下`Ctrl+Z`暂停命令，放入后台。`fg command`使得进程在前台恢复执行，`bg command`使进程在后台恢复执行。

```
tar -zcf etc.tar.gz /etc &
❯ more -20 steam.desktop &
[1] 148567
❯ jobs
[1]  + suspended (tty output)  more -20 steam.desktop
❯ fg more
[1]  + 148745 continued  more -20 steam.desktop
[Desktop Entry]
Name=Steam (Runtime)
Comment=Application for managing and playing games on Steam
Comment[pt_BR]=Aplicativo para jogar e gerenciar jogos no Steam
Comment[bg]=Приложение за ръководене и пускане на игри в Steam
Comment[cs]=Aplikace pro spravování a hraní her ve službě Steam
Comment[da]=Applikation til at håndtere og spille spil på Steam
Comment[nl]=Applicatie voor het beheer en het spelen van games op Steam
Comment[fi]=Steamin pelien hallintaan ja pelaamiseen tarkoitettu sovellus
Comment[fr]=Application de gestion et d'utilisation des jeux sur Steam
Comment[de]=Anwendung zum Verwalten und Spielen von Spielen auf Steam
Comment[el]=Εφαρμογή διαχείρισης παιχνιδιών στο Steam
Comment[hu]=Alkalmazás a Steames játékok futtatásához és kezeléséhez
Comment[it]=Applicazione per la gestione e l'esecuzione di giochi su Steam
Comment[ja]=Steam 上でゲームを管理＆プレイするためのアプリケーション
Comment[ko]=Steam에 있는 게임을 관리하고 플레이할 수 있는 응용 프로그램
Comment[no]=Program for å administrere og spille spill på Steam
Comment[pt_PT]=Aplicação para organizar e executar jogos no Steam
Comment[pl]=Aplikacja do zarządzania i uruchamiania gier na platformie
Steam
Comment[ro]=Aplicație pentru administrarea și jucatul jocurilor pe Steam
```

Fence 25

## 7.3 查看进程

`ps`命令可以查看系统内当前在运行的进程。（`aux`显示所有进程详细信息）

`pstree`以父子进程的形式构成树的方式查看进程。(`-p`显示pid)

`top`命令用交互的方式实时查看进程信息。

```
❯ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  22228 11692 ?        Ss   12月26   0:05
/sbin/init
root           2  0.0  0.0      0     0 ?        S    12月26   0:00
[kthreadd]
root           3  0.0  0.0      0     0 ?        S    12月26   0:00
[pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   12月26   0:00
[kworker/R-rcu_gp]
```

```
root           5  0.0  0.0      0     0 ?        I<   12月26   0:00 [kworker/R-sync_wq]
root           6  0.0  0.0      0     0 ?        I<   12月26   0:00 [kworker/R-slub_flushwq]
root           7  0.0  0.0      0     0 ?        I<   12月26   0:00 [kworker/R-netns]
root           9  0.0  0.0      0     0 ?        I<   12月26   0:00 [kworker/0:0H-events_highpri]
root          12  0.0  0.0      0     0 ?        I<   12月26   0:00 [kworker/R-mm_percpu_wq]
root          14  0.0  0.0      0     0 ?        I    12月26   0:00 [rcu_tasks_kthread]
root          15  0.0  0.0      0     0 ?        I    12月26   0:00 [rcu_tasks_rude_kthread]
root          16  0.0  0.0      0     0 ?        I    12月26   0:00 [rcu_tasks_trace_kthread]
...
systemd─┬─NetworkManager───3*[{NetworkManager}]
        ├─accounts-daemon───3*[{accounts-daemon}]
        ├─agetty
        ├─bluetoothd
        ├─clamd───{clamd}
        ├─clash-verge-ser─┬─verge-mihomo───16*[{verge-mihomo}]
        |                 └─12*[{clash-verge-ser}]
        ├─crond
        ├─dbus-broker-lau───dbus-broker
        ├─dirmngr
        ├─freshclam
        ├─kwalletd6───5*[{kwalletd6}]
        ├─polkitd───3*[{polkitd}]
        ├─power-profiles-───3*[{power-profiles-}]
        ├─rtkit-daemon───2*[{rtkit-daemon}]
        ├─s302.run.sh───sudo───steamcommunity_───17*[{steamcommunity_}]
        ├─sddm─┬─sddm-helper───startplasma-way───{startplasma-way}
        |      └─{sddm}
        ├─sshd
...
❯ top
top - 10:41:49 up 1 day, 12:36,  1 user,  load average: 0.36, 0.35, 0.35
任务: 347 总共, 2 运行中, 345 睡眠, 0 不可中断睡眠, 0 已停止, 0 僵尸
%Cpu(s):  0.7 用,  0.6 系,  0.0 ni, 98.3 闲,  0.1 等,  0.2 硬,  0.1 软,  0.0 丢
MiB 内存:  15714.5 总共,   5163.2 空闲,   8446.2 已用,   4923.4 缓冲/缓存
MiB 交换:      0.0 总共,      0.0 空闲,      0.0 已用.   7268.3 可用 内存

    PID 用户       优  NI    虚拟    驻留    共享 S  %CPU  %MEM     时间+ 命令
    932 ldsr       -2   0 3772652 518204 297384 S   3.7   3.2  33:50.31 kwin_wayland
 147994 ldsr       20   0  951412 117952  87128 S   2.7   0.7   0:01.34 alacritty
 133262 ldsr       20   0 5118360 188768 133152 S   1.0   1.2   1:32.32 wpscloudsvr
   1162 ldsr       20   0 8042328 601100 191228 S   0.3   3.7   6:48.56 plasmashell
   1606 root       20   0 1265020  39112  18600 S   0.3   0.2   0:12.68 verge-mihomo
 109427 ldsr       12  -8   33.2g 361884 173584 S   0.3   2.2   7:26.91 msedge
```

```
143758 root      20   0       0      0      0 R   0.3   0.0   0:01.74
kworker/u48:39-btrfs_discard
144484 root       0 -20       0      0      0 I   0.3   0.0   0:02.37
kworker/u49:0-i915_flip
146834 root      20   0       0      0      0 I   0.3   0.0   0:00.70
kworker/u48:5-events_power_efficient
     1 root      20   0   22228  11692   7940 S   0.0   0.1   0:05.53
       systemd
```

Fence 26

## 7.3.1 关闭进程

### 7.3.1.1 `kill`命令

`kill [-s singal] pid`

常用信号量一般有:

| 信号代号 | 信号名称 | 说明 |
| --- | --- | --- |
| 9 | SIGKILL | 立即结束程序的运行，不能被阻塞、处理和忽略一般用于强制终止进程。 |
| 15 | SIGTERM | 进程正常结束信号，kill命令的默认信号。 |

`pkill/killall [option] [-s signal] pname`根据进程名结束进程。

## 7.3.2 定时任务

### 7.3.2.1 `at`命令

`at`是一次性命令，用于在用户指定的时刻执行指定的命令或命令序列。

`at [-ldc] time`

```
# at now + 2 minutes
at> /hello.sh
at> useradd student1
at> echo “11” >> /tmp/test
at> <EOT>   # 在输入命令后 按 ctrl +d结束输入
# at -l
1 2022-10-01 08:00 a root
2 2023-07-08 19:15 a root
3 2023-07-09 19:01 a root
```

Fence 27

### 7.3.2.2 `crontab`命令

`crontab [-elr]`选项分别为编辑、列出、删除定时任务。

crontab文件格式要求:

```
* * * * *  执行的任务
# 示例
*/5 * * * * 命令  */5 * * * * /bin/echo ”11” >> /tmp/test
0 17 * * 1 命令   0 17 * * 1 /sbin/shutdown -r now
0 5 1,15 * * 命令 0 5 1,15 * * /root/sh/autobak.sh
```

Fence 28

| 项目 | 含义 | 取值范围 |
| --- | --- | --- |
| 第一个“*” | 第几分钟 | 0-59分钟 |
| 第二个“*” | 第几小时 | 0-23 |
| 第三个“*” | 第几天 | 1-31 |
| 第四个“*” | 第几月 | 1-12 |
| 第五个“*” | 星期几 | 0-7(0和7都代表星期日) |

| 特殊符号 | 含义 |
| --- | --- |
| * | 代表任何时间 |
| , | 代表不连续的时间 |
| - | 代表连续的时间范围 |
| */n | 代表每隔多久执行一次 |

# 8. shell编程

## 8.1 vim编辑器

vim是交互式文本编辑器，没有菜单，只有命令。

### 8.1.1 启动vim

`vim [option] file`

如果没有文件则新建再编辑。

启动时，vim默认在命令模式，输入`:`进入行编辑模式，可以输入指令，如`:wq`保存并退出等。

在命令模式按下`A、I、O、a、i、o`文本编辑模式，按下`ECS`返回命令模式。

### 8.1.2 基本命令

| 命令 | 说明 |
| --- | --- |
| w | 光标移动到下个单词的首字符 |
| e/E | 光标移动到下个单词的尾字符 |
| b/B | 光标移动到上个单词的首字符 |
| /string | 向后搜索string |
| ?string | 向前搜索string |
| $ | 光标移动到行末 |
| gg | 光标移动到第一行 |
| G | 光标移动到最后一行 |
| h | 向左移动一字符 |
| l | 向右移动一字符 |
| j | 向上移动一字符 |
| k | 向下移动一字符 |
| i | 在当前光标前插入 |
| I | 在当前行首插入 |
| a | 在当前光标后插入 |
| A | 在当前行尾插入 |
| o | 在当前行后添加新行 |
| O | 在当前行前添加新行 |
| x | 删除光标处字符 |
| dd | 删除/剪切当前行 |
| r | 从当前光标处开始替换 |
| u | 撤销上次操作 |
| >> | 整行向右缩进 |
| << | 整行向左缩进 |
| sw num | 设置缩进长度 |
| ri | 设置右对齐 |
| le | 设置左对齐 |
| ce | 设置居中对齐 |
| %s/old/new/g | 替换old为new，/g表示全文查找 |

## 8.2 shell概念

- Shell是命令行解释器，它为用户提供了向Linux内核发送请求的界面系统级程序，控制程序运行。
- Shell还是一个功能相当强大的编程语言，易编写和调试，灵活性较强。
- Shell是解释执行的脚本语言，在Shell中可以直接调用Linux系统命令。

### 8.2.1 分类

Shell的两种主要语法类型：Bourne和C

- Bourne家族主要包括sh、ksh、Bash、psh、zsh
- C家族主要包括：csh、tcsh

Bash与sh兼容，现在使用的Linux 就是使用Bash作为用户的基本Shell

## 8.3 输入输出重定向

| 设备 | 设备文件名 | 文件描述符 | 类型 |
| --- | --- | --- | --- |
| 键盘 | /dev/stdin | 0 | 标准输入 |
| 显示器 | /dev/sdtout | 1 | 标准输出 |
| 显示器 | /dev/sdterr | 2 | 标准错误输出 |

| 符号 | 说明 |
| --- | --- |
| \| | 管道符，使上一个命令的输出作为下一个命令的输入 |
| < | 输入重定向 |
| 命令 > 文件 | 覆盖，命令的正确输出输出至文件 |
| 命令 >> 文件 | 追加，命令的正确输出输出至文件 |
| 错误命令 2 > 文件 | 覆盖，命令的错误输出输出至文件 |
| 错误命令 2>> 文件 | 追加，命令的错误输出输出至文件 |
| 命令 > 文件 2>&1 | 覆盖，把正确输出和错误输出都保存到同一个文件当中。 |
| 命令 &> 文件 | 覆盖，把正确输出和错误输出都保存到同一个文件当中。 |
| 命令 >> 文件 2>&1 | 追加，把正确输出和错误输出都保存到同一个文件当中。 |
| 命令 &>> 文件 | 追加，把正确输出和错误输出都保存到同一个文件当中。 |
| 命令>>文件1 2>>文件2 | 把正确的输出追加到文件1，把错误的输出追加到文件2。 |

```
tar code.tar.gz code/ >> /dev/null 2>&1
```

Fence 29

## 8.4 多命令

| 多命令执行 | 格式 | 作用 |
| --- | --- | --- |
| ; | 命令1;命令2 | 多个命令顺序执行 |
| && | 命令1&&命令2 | 逻辑与 |
| \|\| | 命令1\|\|命令2 | 逻辑或 |
| \| | 命令1\|命令2 | 管道符，前一命令的执行结果传递为后一命令的参数 |

## 8.5 其他特殊符号

| 符号 | 作用 |
| --- | --- |
| '' | 单引号。在单引号中所有的特殊符号都没有特殊含义。 |
| "" | 双引号。在双引号中特殊符号一般没有特殊含义，但是$、` 和 \ 是例外，拥有"调用变量的值"、"引用命令"和"转义符"的特殊含义。 |
| `` | 反引号。 |
| $() | 同反引号，用于引用系统命令。 |
| # | Shell脚本中，行开头#号代表注释。 |
| $ | 用于调用变量的值。 |
| \ | 转义符。 |

```
❯ name=sc
❯ echo '$name'
$name
❯ echo "$name"
sc
❯ echo '$(date)'
$(date)
❯ echo "$(date)"
2024年 12月 28日 星期六 11:37:11 CST
```

Fence 30

## 8.6 接受键盘输入

```
read [选项] [变量名]
-p # 提示信息
-t # 指定等待时间
-n # 接受指定的字符数
```

Fence 31

## 8.7 shell变量

- 变量名称可以由字母、数字和下划线组成，但是不能以数字开头。
- 在Bash中，变量的默认类型都是字符串型，如果要进行数值运算，则必须指定变量类型为数值型。
- 在bash中，变量的使用不需要显式的声明，赋值认为是变量的声明。

变量的引用：`"$变量名"`，或`${变量名}`。

| 位置参数变量 | 作用 |
| --- | --- |
| $n$\|$n$为数字，0代表命令本身，1—9代表第一到第九个参数，十以上的参数需要用大括号包含，如${10}。 | |
| $*$\|这个变量代表命令行中所有的参数，*把所有的参数看成一个整体。 | |
| @\|这个变量也代表命令行中所有的参数，不过@把每个参数区分对待。 | |
| $# | 这个变量代表命令行中所有参数的个数。 |
| $? | 最后一次执行的命令的返回状态。如果这个变量的值为0，证明上一个命令正确执行；如果这个变量的值为非0，则证明上一个命令执行不正确。 |
| $$ | 当前进程的进程号(PID) |
| $! | 后台运行的最后一个进程的进程号（PID) |

## 8.8 shell表达式

### 8.8.1 `expr`命令

在expr命令的表达式中使用了数值运算，此时需要用空格将数字运算符与操作数分隔开。另外，如果表达式中的运算符是<、>、&、*及 | 等特殊符号，需要使用双引号、单引号括起来，或将反斜杠（\）放在这些符号的前面。
expr 3+2：操作数3、2和运算符+之间没有空格，bash不会报错，把3+2作为字符串来处理。
expr 3 + 2：操作数3、2和运算符+之间有空格，bash认为是数字运算，返回5到标准输出设备。
expr 3"*"2:  使用双引号将操作符*括起，此时bash返回乘积6。

```
#!/bin/bash
num1=$1
num2=$2
sum=$(( $num1 + $num2))    # num1 加 num2
echo $sum                           # 打印变量 sum 的值
```

Fence 32

### 8.8.2 条件判断

`test expression`或者`[expression]`

真为0，假为非零

| 运算符 | 说明 |
| --- | --- |
| string1 = string2 | 相等为真，否则为假 |
| string1 != string2 | 不等为真，否则为假 |
| string | 空字符串为真，否则为假 |
| -n string | 长度非零为真 |
| -z string | 长度为0为真 |
| -eq | 表达式相等为真 |
| -ne | 表达式不等为真 |
| -gt | 大于为真 |
| -ge | 大于等于为真 |
| -lt | 小于为真 |
| -le | 小于等于为真 |
| -d fname | 是目录为真 |
| -f fname | 存在且为普通文件则为真 |
| -r fname | 存在且可读为真 |
| -w fname | 存在且可写为真 |
| -x fname | 存在且可执行为真 |
| ！expr | 非 |
| -a | 与 |
| -o | 或 |

## 8.9 shell结构控制

### 8.9.1 if结构

```
#! /bin/bash
if (test expression) then
...
            else
                ...
            fi

            if [expression]; then
...
                        else
```

```
                fi

                if [expression]
                then
 ...
 else
 ...
                    fi

                                    if [expression1]
                then
 ...
 elif [expression2]
 then
 ...
 else
 ...
                    fi
```

Fence 33

### 8.9.2 case结构

```
case string in
    pattern)
        command ...
        ;;
    *)
        command ...
        ;;
esac
```

Fence 34

### 8.9.3 for循环

```
for item in expression; do
    command ...
done

for item in lists
do
    command ...
done
```

Fence 35

### 8.9.4 while/until循环

```
    while [expression]; do
    command ...
done

    while [expression]
    do
    command ...
done
```

Fence 36