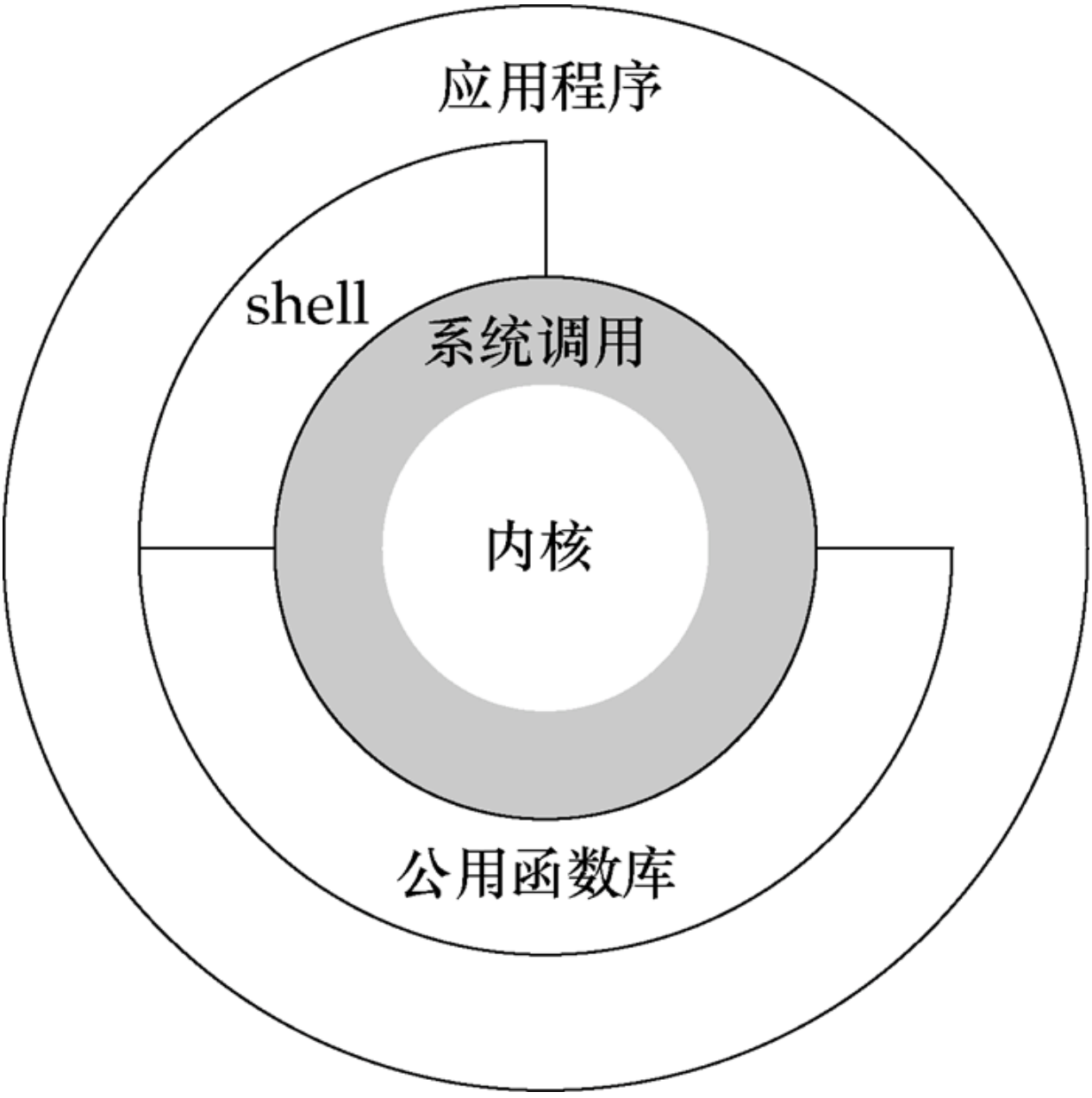
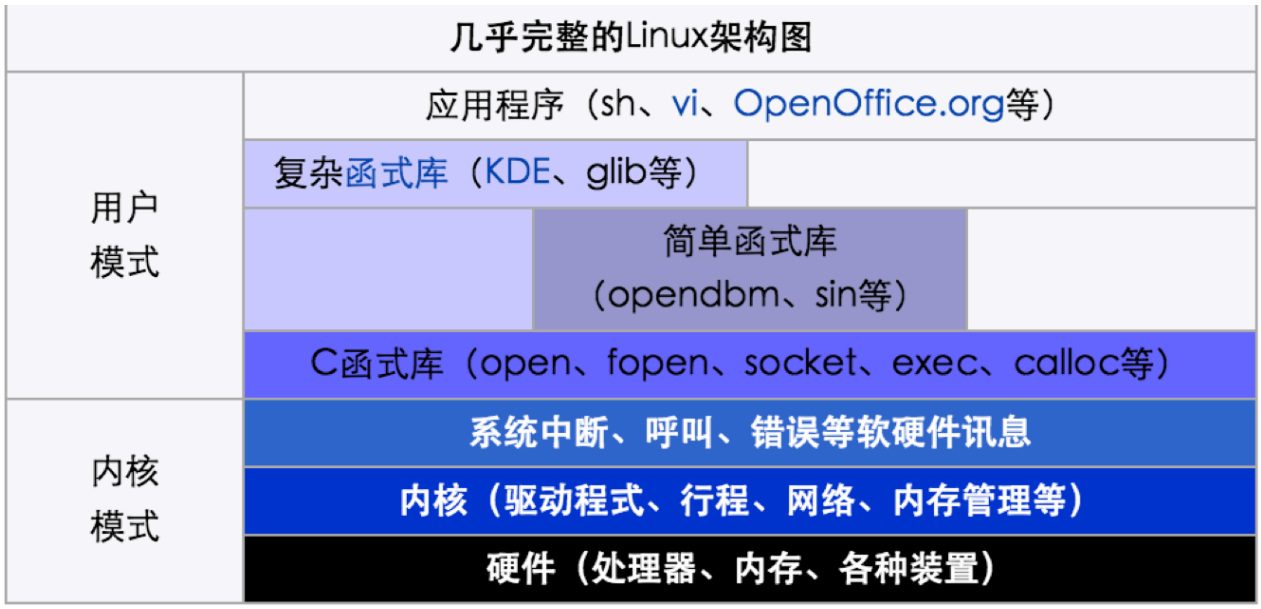
前端 linux 基础大全

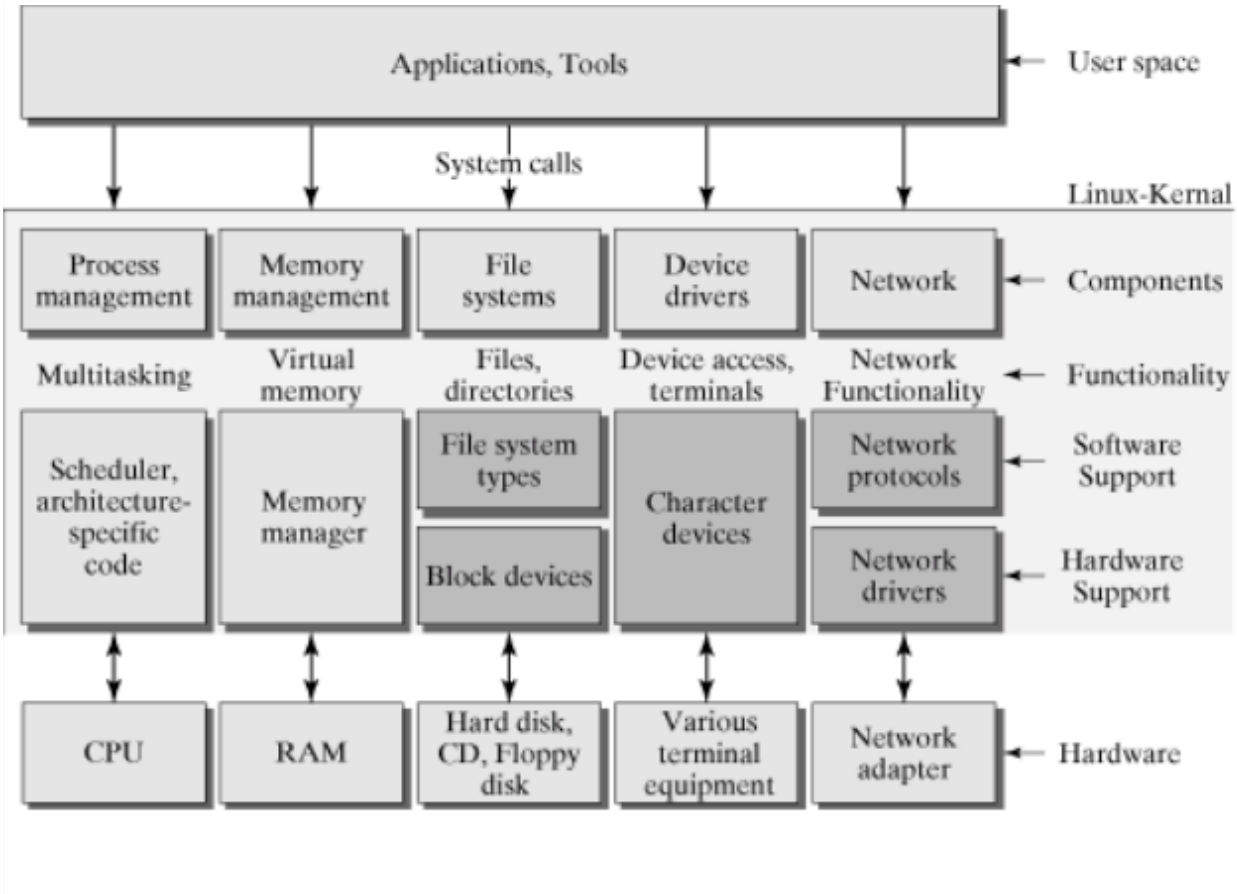
1. Linux 架构体系

1.1 Unix/Linux 的体系架构





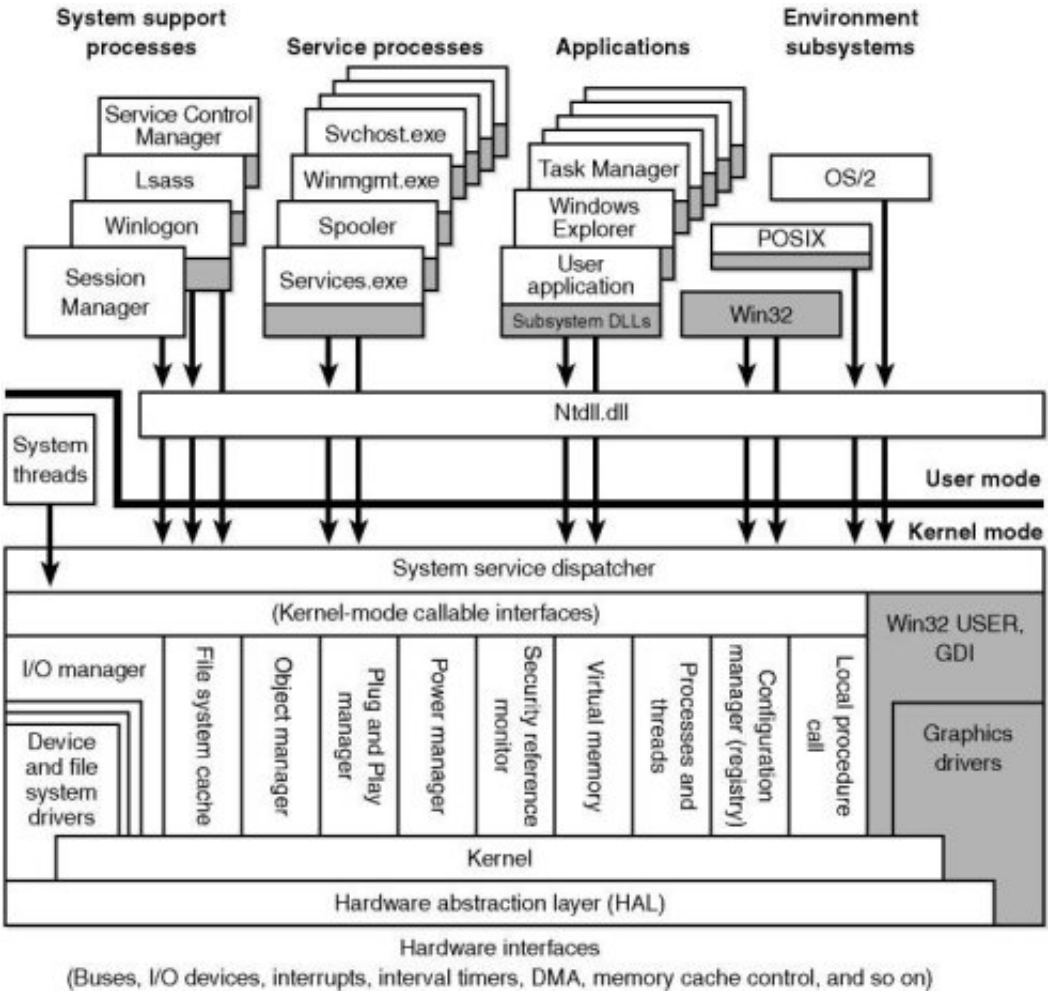
1.2 Linux 系统体系结构



1.3 windows 架构体系

<table>
<tr><td colspan="7">简化版本的Windows NT抽象架构</td></tr>
<tr><td rowspan="3">用户
模式</td><td rowspan="2">OS/2
应用程序</td><td colspan="2">Win32
应用程序</td><td>DOS
程式</td><td>Win16
应用程序</td><td rowspan="2">POSIX
应用程序</td></tr>
<tr><td></td><td>其他DLL
函式库</td><td>DOS系统</td><td>Windows模拟系统</td></tr>
<tr><td>OS/2副系统</td><td colspan="4">Win32副系统</td><td>POSIX.1副系统</td></tr>
<tr><td rowspan="4">内核
模式</td><td colspan="6">系统服务层</td></tr>
<tr><td>输入输出管理
档案系统、网络
系统</td><td colspan="4">物件管理系统 / 安全管理系统 / 行程管理 / 物件间通信管理 / 行程间通信管理 / 虚拟内存管理

微内核</td><td>视窗管理程式</td></tr>
<tr><td>驱动程式</td><td colspan="4">硬件抽象层（HAL）</td><td>图形驱动</td></tr>
<tr><td colspan="6">硬件（处理器、内存、外部装置等）</td></tr>
</table>



2.linux常用命令

2.1 常见目录

/bin	**存放二进制可执行文件(ls,cat,mkdir等)，常用命令一般都在这里。**
/etc	存放系统管理和配置文件
/home	存放所有用户文件的根目录，是用户主目录的基点，比如用户user的主目录就是/home/user，可以用~user表示
/usr	用于存放系统应用程序，比较重要的目录/usr/local 本地系统管理员软件安装目录（安装系统级的应用）。这是最庞大的目录，要用到的应用程序和文件几乎都在这个目录。/usr/x11r6 存放x window的目录/usr/bin 众多的应用程序/usr/sbin 超级用户的一些管理程序/usr/doc linux文档/usr/include linux下开发和编译应用程序所需要的头文件/usr/lib 常用的动态链接库和软件包的配置文件/usr/man 帮助文档/usr/src 源代码，linux内核的源代码就放在/usr/src/linux里/usr/local/bin 本地增加的命令/usr/local/lib 本地增加的库
/opt	额外安装的可选应用程序包所放置的位置。一般情况下，我们可以把tomcat等都安装到这里。
/proc	虚拟文件系统目录，是系统内存的映射。可直接访问这个目录来获取系统信息。
/root	超级用户（系统管理员）的主目录（特权阶级^o^）
/sbin	存放二进制可执行文件，只有root才能访问。这里存放的是系统管理员使用的系统级别的管理命令和程序。如ifconfig等。
/dev	用于存放设备文件。
/mnt	系统管理员安装临时文件系统的安装点，系统提供这个目录是让用户临时挂载其他的文件系统。
/boot	存放用于系统引导时使用的各种文件
/lib	存放跟文件系统中的程序运行所需要的共享库及内核模块。共享库又叫动态链接共享库，作用类似windows里的.dll文件，存放了根文件系统程序运行所需的共享文件。
/tmp	用于存放各种临时文件，是公用的临时文件存储点。
/var	用于存放运行时需要改变数据的文件，也是某些大文件的溢出区，比方说各种服务的日志文件（系统启动日志等。）等。
/lost+found	这个目录平时是空的，系统非正常关机而留下“无家可归”的文件（windows下叫什么.chk）就在这里

2.2 命令基本格式

2.2.1 命令提示符

```
[root@xiaoming ~]#
```

- root 当前登录用户

- localhost 主机名
- ~ 当前工作目录,默认是当前用户的家目录，root就是/root,普通用户是 /home/用户名
- 提示符 超级用户是 #,普通用户是$

2.2.2 命令格式

- 命令 [选项] [参数]
- 当有多个选项时，可以写在一起
- 一般参数有简化和完整写法两种 `-a` 与 `--all` 等效

2.2.3 ls

- 查询目录中的内容
- ls [选项] [文件或者目录]
- 选项
  - -a 显示所有文件，包括隐藏文件
  - -l 显示详细信息
  - -d 查看目录本身的属性而非子文件 ls /etc/
  - -h 人性化的方式显示文件大小
  - -i 显示inode,也就是i节点，每个节点都有ID号
- 默认当前目录下的文件列表

-l

显示详细信息

```
drwxr-xr-x .  1 root  root   800 Sep 16 00:19 logs
```

drwxr-xr-x	.	1	root	root	800	Sep 16 00:19	logs
文件类型和权限	ACL权限	硬链接引用计数	所有者	所属组	文件大小	最后修改时间	文件名

文件类型和权限

```
-rw-r--r--
```

- 文件类型 - 文件、d 目录、l 软链接文件
- u(所有者)、g(所属组)、o(其他人)
- r(read) 读取、w(write) 写入、x(execute) 执行

2.3 文件处理命令

2.3.1 mkdir

- 建立目录 make directory
- mkdir -p [目录名]

    - -p 递归创建

2.3.2 cd

- 切换所在目录 change directory
- cd [目录]
    - ~ 家目录
    - 家目录
    - - 上次目录
    - . 当前目录
    - .. 上级目录
- 相对路径是参照当前所在目录
- 绝对路径是从根目录开始
- 按TAB键可以补全命令和目录

2.3.3 pwd

- 显示当前目录 pwd

2.3.4 rmdir

- 删除目录 remove empty directory
- rmdir [目录名]

2.3.5 rm

- 删除文件或者目录 remove
- rm [文件或者目录]
    - -r 删除目录
    - -f 强制删除
- rm -rf 文件或者目录] 递归强制删除所有目录

2.3.6 cp

- copy 复制命令
- copy [源文件或者目录] [目标文件]
    - -r 复制目录,默认是复制文件
    - -p 连带文件属性复制
    - -d 若源文件是链接文件，则复制连接属性
    - -a 相当于 -rpd

2.3.7 mv

- 移动文件或者改名 move
- mv [源文件或者目录] [目标文件]

2.3.8 ln

- 链接命令,生成链接文件 `link`

2.3.8.1 硬链接特征

- 拥有相同的i节点和存储block块，可以看作是同一个文件
- 可以通过i节点访问
- 不能跨分区
- 不能针对目录使用
- 一般不使用

2.3.8.2 软链接特征

- ln -s [源文件] [目标文件]
    - -s 创建软链接
- 类似Windows快捷方式
- 软链接拥有自己的i节点和Block块，但是数据块中只保存源文件的文件名和i节点号，并没有实际的文件数据
- lrwxrwxrwx l 软链接 软链接的文件权限都是 777
- 修改任意一个文件，另一个都会改变
- 删除源文件，软链接不能使用
- 软链接源文件必须写绝对路径

2.4 文件搜索命令

2.4.1 locate

- 在后台数据库中按文件名搜索，速度比较快
- 数据保存在`/var/lib/mlocate`后台数据库，每天更新一次
- 可以`updatedb`命令立刻更新数据库
- 只能搜索文件名

```
/etc/updatedb.conf
```

建立索引的配置文件

- PRUNE_BIND_MOUNTS = "yes" 全部生效，开启搜索限制
- PRUNEFS 不搜索的文件系统
- PRUNENAMES 忽略的文件类型
- PRUNEPATHS 忽略的路径 /tmp

2.4.2 whereis

- 搜索命令所在路径以及帮助文档所在位置
- whereis 命令名

```
whereis ls
```

- -b 只查找可执行文件
- -m 只查找帮助文件

2.4.3 which

- 可以看到别名 `which ls`
- 能看到的都是外部安装的命令
- 无法查看Shell自带的命令，如 `which cd`

2.4.4 环境变量

```
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

- 定义的是系统搜索命令的路径
- echo $PATH

2.4.5 find

- 文件搜索命令
- find [搜索范围] [搜索条件]

2.4.5.1 按名称搜索

- 避免大范围的搜索，会非常消耗系统资源

```
find / -name aaa.log
```

2.4.5.2 通配符

- find是在系统当中搜索符合条件的文件名，如果需要匹配，使用通配符匹配，通配符是完全匹配
- 通配符
  - `*` 匹配任意内容
  - `?` 匹配任意一个字符
  - `[]` 匹配任意一个中括号内的字符

```
find . -name "ab[cdef]"
```

2.4.5.3 -i

不区分大小写

```
find / -iname A.log
```

2.4.5.4 -user

按所有者进行搜索

```
find /root -user root
find /root -nouser
```

2.4.5.5 按时间搜索

```
find /nginx/access.log -mtime +5
```

参数	含义
atime	文件访问时间
ctime	改变文件属性
mtime	修改文件内容

参数	含义
-5	5天内修改的文件
5	5天前当前修改的文件
+5	5天前修改的文件

2.4.5.6 按大小搜索

- k小写,M大写

```
find . -size 100k
```

参数	含义
-8k	小于8K
8k	等于8K
+8k	大于8K
+8M	小于8M

2.4.5.7 按i节点搜索

```
find . -inum 123456
```

2.4.5.8 综合应用

```
find /tmp -size +10k -a -size -20k
```

- 查找/etc目录下，大于10KB并且小于20KB的文件

- -a and 逻辑与，两个条件都满足
- -o or 逻辑或，两个条件满足一个就可以

```
find /tmp -size +10k -a -size -20k -exec ls -lh {} \;
```

- exec 对上个命令的结果进行操作

2.4.5.9 grep

- 在文件当中匹配符合条件的字符串
- grep "10" access.log
  - `-i` 忽略大小写
  - `-v` 排除指定字符串
- find命令，在系统当中搜索符合条件的文件名，如果需要匹配，使用通配符匹配，通配符是完全匹配
- grep命令 在文件当中搜索符合条件的字符串，如果需要匹配，使用正则表达式进行匹配，正则表达式时包含匹配

2.5 帮助命令

2.5.1 基本用法

- man 命令 获取指定命令的帮助
- `man ls` 查看ls的帮助

```
man -f ls
whatis ls
man 1 ls
man 1p ls
```

2.5.2 关键字搜索

```
- man -k passwd
```

2.5.3 shell 内部帮助

```
whereis
```

找到就是外部，找不到就是内部

```
help cd
```

2.6 压缩与解压缩命令

```
.zip` `.gz` `.bz2` `.tar.gz` `.tar.bz2
```

2.6.1 zip格式

- 压缩文件 zip 压缩文件名 源文件
- 压缩目录 zip -r 压缩文件名 源目录
- 解压 unzip 压缩文件名

```
mkdir book
touch book/1.txt
touch book/2.txt
zip -r book.zip book
unzip book.zip
```

2.6.2 gzip

命令	示例	含义
gzip 源文件	gzip a.txt	压缩为.gz格式的压缩文件，源文件会消失
gzip -c 源文件 > 压缩文件	gzip -c yum.txt > yum.txt.gz	压缩为.gz格式的压缩文件，源文件不会消失
gzip -r 目录	gzip -r xx	压缩目录下的所有子文件，但是不压缩目录
gzip -d 压缩文件名	gzip -d yum.txt.gz	解压缩文件,不保留压缩包
gunzip 压缩文件	gunzip yum.txt.gz	解压缩文件,不保留压缩包

- 压缩是压缩目录下的文件

2.6.3 .bz2格式压缩

命令	示例	含义
bzip2 源文件	bzip2 1.txt	压缩为.bz2格式的文件，不保留源文件
bzip2 -k 源文件	zip2 -k 1.txt	压缩为.bz2格式的文件，保留源文件
bzip2 -d 压缩文件名	bzip2 -d 1.txt.bz2	解压压缩包
bunzip2 压缩文件名	bunzip2 1.txt.bz2	解压压缩包

- bzip2 不能压缩目录

2.6.4 tar

- 打包命令
- tar -cvf 打包文件名 源文件

- -c 打包
- -v 显示过程
- -f 指定打包后的文件名

```
tar -cvf book.tar book
gzip book.tar
bzip2 book.tar
```

- x 解打包

```
tar -xvf book.tar
```

2.6.4 压缩格式

压缩

> tar –cvf jpg.tar *.jpg //将目录里所有jpg文件打包成tar.jpg tar –czf jpg.tar.gz *.jpg //将目录里所有jpg文件打包成jpg.tar后，并且将其用gzip压缩，生成一个gzip压缩过的包，命名为jpg.tar.gz tar –cjf jpg.tar.bz2 *.jpg //将目录里所有jpg文件打包成jpg.tar后，并且将其用bzip2压缩，生成一个bzip2压缩过的包，命名为jpg.tar.bz2 tar –cZf jpg.tar.Z *.jpg //将目录里所有jpg文件打包成jpg.tar后，并且将其用compress压缩，生成一个umcompress压缩过的包，命名为jpg.tar.Z rar a jpg.rar *.jpg //rar格式的压缩，需要先下载rar for linux zip jpg.zip *.jpg //zip格式的压缩，需要先下载zip for linux

解压

> tar –xvf file.tar //解压 tar包 tar -xzvf file.tar.gz //解压tar.gz tar -xjvf file.tar.bz2 //解压 tar.bz2 tar –xZvf file.tar.Z //解压tar.Z unrar e file.rar //解压rar unzip file.zip //解压zip

2.7 关机和重启命令

2.7.1 shutdown

shutdown 关机命令

- -c 取消前一个关机命令
- -h 关机
- -r 重启

```
shutdown -r 06:00
shutdown -c
```

2.7.2 init

关机

```
init 0
```

重启

```
init 6
```

系统的运行级别

- 0 关机
- 1 单用户
- 2 不完全多用户，不包含NFS服务
- 3 完全多用户
- 4 未分配
- 5 图形界面
- 6 重启

2.7.3 logout

退出登录

```
logout
```

2.9 查看登录用户信息

2.9.1 w

查看登录用户信息

- USER 登录的用户名
- TTY 登录的终端 tty1 本地终端 pts/0远程终端
- FROM 登录的IP
- LOGIN 登录时间
- IDLE 用户闲置时间
- JCPU 该终端所有进程占用的时间
- PCPU 当前进程所占用的时间
- WHAT 正在执行的命令

2.9.2 who

查看登录用户信息

- USER 登录的用户名
- TTY 登录的终端 tty1 本地终端 pts/0远程终端
- LOGIN 登录时间（登录的IP）

2.9.3 last

查看当前登录和过去登录的用户信息 默认读取 `/var/log/wtmp` 文件

- 用户名
- 登录终端
- 登录IP
- 登录时间
- 退出时间(在线时间)

2.9.4 lastlog

查看所有用户的最后一次登录时间

- 用户名
- 登录终端
- 登录IP
- 最后一次登录时间

3. shell

- shell是一个命令行解释器，它为用户提供了一个向Linux内核发送请求以便运行程序的界面系统级程序
- 用户可以用Shell来启动、挂起、停止或者编写一些程序
- Shell还是一个功能相当强大的编程语言，易编写，易调试，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。

3.1 查看支持的shell

- /etc/shells

3.2 echo

- 输出命令
- --e 支持反斜线控制的字符转换

控制字符	作用
\a	输出警告音
\b	退格键，也就是向左删除键
\n	换行符
\r	回车键
\t	制表符，也就是Tab键
\v	垂直制表符
\onnn	按照八进制ASCII码表输出字符，其中0为数字零，nnn是三位八进制数
\xhh	按照十六进制ASCII码表输出字符，其中hh是两位十六进制数

3.3 编写执行shell

```
#!/bin/bash
echo -e "\e[1;34m hello world \e[0m"
```

赋予执行权限，直接运行

```
chmod 755 hello.sh
./hello.sh
```

通过Bash调用执行脚本

```
bash hello.sh
```

3.4 别名

- 命令别名 == 小名
- 临时生效
- `alias`
- `alias rm="rm -i"`
- 写入环境变量配置文件 vi ~/.bashrc
- `source ~/.bashrc`
- unalias 别名 删除别名

3.5 命令的生效顺序

- 绝对路径或者相对路径
- 别名
- bash内部命令
- 按照$PATH环境变量定义的目录查找顺序找到的第一个命令

3.6 命令快捷键

命令	含义
ctrl+c	强制终止当前命令
ctrl+l	清屏
ctrl+a	光标移动到命令行首
ctrl+e	光标移动到命令行尾
ctrl+u	从光标所在的位置删除到行首
ctrl+z	把命令放入后台
ctrl+r	在历史命令中搜索

3.7 历史命令

- history [选项] [历史命令保存文件]
- 选项
    - -c 清空历史命令

    - -w 把缓存中的历史命令写入历史命令保存文件 ~/.bash_history
- 默认保存1000条 /etc/profile HISSIZE=10000

3.8 调用

- 使用上下箭头调用以前的历史命令
- 使用 !n 重复执行第n条历史命令
- 使用 !! 重复执行上一条命令
- 使用 !字符 重复执行最后一条以该字符串开头的命令

3.9 管道符号

3.9.1 多命令顺序执行

(1) ; 分号，没有任何逻辑关系的连接符。当多个命令用分号连接时，各命令之间的执行成功与否彼此没有任何影响，都会一条一条执行下去。

(2) || 逻辑或，当用此连接符连接多个命令时，前面的命令执行成功，则后面的命令不会执行。前面的命令执行失败，后面的命令才会执行。

(3) && 逻辑与，当用此连接符连接多个命令时，前面的命令执行成功，才会执行后面的命令，前面的命令执行失败，后面的命令不会执行，与 || 正好相反。

(4) | 管道符，当用此连接符连接多个命令时，前面命令执行的正确输出，会交给后面的命令继续处理。若前面的命令执行失败，则会报错，若后面的命令无法处理前面命令的输出，也会报错。

```
- date;ls;date;ls
- ls && echo yes || echo no
```

3.9.2 管道符号

- 命令1的正确输出会作为命令2的操作对象
- 命令1|命令2

```
ls /etc/ | more
netstat -an | grep ESTABLISHED | wc -l
```

3.9.3 通配符

匹配文件名和目录名 |通配符|作用| |:----|:----| |?|匹配一个任意字符| |*|匹配0个或任意字符，也就是可以匹配任意内容| |[]|匹配中括号中任意一个字符| |[-]|匹配中括号中任意一个字符,-代表范围| |[^]|匹配不是中括号中的一个字符|

3.9.4 其它符号

符号	作用
''	单引号。在单引号中所有的特殊符号，如$和`都没有特殊含义
""	双引号，在双引号里特殊符号都没有特殊含义，但是 $ ` \ 例外，拥有调用变量值，引用命令和转义的含义
``	反引号，扩起来的是系统命令
$()	和反引号一样
#	在shell脚本中，#开头的行代表注释
$	用于调用变量的值
\	转义符号

```
- a=`ls`
- b=$(ls)
```

4. vi编辑器

- VI visual interface
- 可视化接口
- 类似与windows中的记事本
- vim支持多级撤销
- 跨平台
- 语法高亮
- 支持图形界面

4.1 操作模式

- :w 保存
- :q 退出
- :! 强制保存
- :ls 列出所有的文件
- :n 下一个
- :N 上一个
- :15 跳转到指定行
- /xxx 从光标位置开始向后搜索 xxx 字符串
- ?xxx 从光标位置开始向前搜索

5. 用户和用户组

- 使用操作系统的人都是用户
- 用户组是具有相同系统权限的一组用户

5.1 用户组

5.1.1 /etc/group

- /etc/group 存储当前系统中所有用户组信息
- `group:x:123:abc,def`
- 组名称:组密码占位符:组编号:组中用户名列表
- root 组编号为0
- 1-499系统预留的编号 预留给安装的软件和服务的
- 用户手动创建的用户组从500开始
- 组密码占位符都是x

5.1.2 /etc/gshadow

- 存放当前系统中用户组的密码信息
- 和group中的记录一一对应
- `Group: * : :abc`
- 组名称 组密码 组管理者 组中用户名

5.1.3 /etc/passwd

- 存储当前系统中所有用户的信息
- `user:x:123:456:xxxxx:/home/user:/bin/bash`
- 用户名:密码占位符:用户编号: 用户注释信息:用户主目录:shell类型

5.1.4 /etc/shadow

- 存放当前系统中所有用户的密码信息
- `user:xxx:::::::`
- 用户名:密码:

6. 用户操作

添加组

```
groupadd student
```

修改组名称

```
groupmod -n stu student
```

修改组编号

```
groupmod -g 111 stu
```

添加分组并指定编号

```
groupadd -g 222 teacher
```

删除分组

```
groupdel 222
```

添加分组

```
groupadd teacher
```

为用户指定所属组

```
useradd -g teacher zhangsan
```

为用户指定所属组

```
useradd -g teacher lisi
```

为用户指定工作目录

```
useradd -d /home/zhangsan zhangsan
```

指定注释

```
usermod -c iamateacher zhangsan
```

修改用户名

```
usermod -l zhangsan zhangsan2
```

指定文件夹

```
usermod -d /home/zhangsan2 zhangsan2
```

修改用户所属组

```
usermod -g stu zhangsan2
```

删除用户

```
userdel zhangsan2
```

删除所属文件夹

```
userdel -r lisi
```

7. 用户命令

显示登录的用户名

```
whoami
```

显示指定用户信息，包括用户编号，用户名 主要组的编号及名称，附属组列表

```
id zhangsan
```

显示zhangsan用户所在的所有组

```
groups zhangsan
```

显示用户详细资料

```
finger zhangsan
```

8. 附录

8.1 系统启动

8.1.1 BIOS

- 计算机通电后，第一件事就是读取刷入ROM芯片的开机程序，这个程序叫做(Basic Input/Output System)

8.1.2 硬件自检

- BIOS程序首先检查，计算机硬件能否满足运行的基本条件，这叫做"硬件自检"（Power-On Self-Test)
- 如果硬件出现问题，主板会发出不同含义的蜂鸣，启动中止。如果没有问题，屏幕就会显示出CPU、内存、硬盘等信息。

8.1.3 启动顺序

- 硬件自检完成后，BIOS把控制权转交给下一阶段的启动程序。
- 这时，BIOS需要知道，"下一阶段的启动程序"具体存放在哪一个设备
- BIOS需要有一个外部储存设备的排序，排在前面的设备就是优先转交控制权的设备。这种排序叫做"启动顺序"（Boot Sequence）
- BIOS按照"启动顺序"，把控制权转交给排在第一位的储存设备。
- 这时，计算机读取该设备的第一个扇区，也就是读取最前面的512个字节。如果这512个字节的最后两个字节是0x55和0xAA，表明这个设备可以用于启动；如果不是，表明设备不能用于启动，控制权于是被转交给"启动顺序"中的下一个设备。

- 这最前面的512个字节，就叫做"主引导记录"（Master boot record，缩写为MBR）

8.1.4 主引导记录的结构

- "主引导记录"只有512个字节，放不了太多东西。它的主要作用是，告诉计算机到硬盘的哪一个位置去找操作系统。
    - （1） 第1-446字节：是用来记录系统的启动信息的,调用操作系统的机器码
    - （2） 第447-510字节(64个字节)：分区表（Partition table），分区表的作用，是将硬盘分成若干个区
    - （3） 第511-512字节：主引导记录签名（0x55和0xAA）

8.1.5 分区表

- 磁盘分区是使用分区编辑器在磁盘上划分几个逻辑部分
- 磁盘一旦划分成多个分区，不同类的目录与文件可以存储进不同的分区内
- "主引导记录"因此必须知道将控制权转交给哪个区
- 分区表的长度只有64个字节，里面又分成四项，每项16个字节。所以，一个硬盘最多只能分四个一级分区，又叫做"主分区"
    - （1） 第1个字节：如果为0x80，就表示该主分区是激活分区，控制权要转交给这个分区。四个主分区里面只能有一个是激活的。
    - （2） 第2-4个字节：主分区第一个扇区的物理位置（柱面、磁头、扇区号等等）。
    - （3） 第5个字节：主分区类型，比如FAT32、NTFS等。
    - （4） 第6-8个字节：主分区最后一个扇区的物理位置。
    - （5） 第9-12字节：该主分区第一个扇区的逻辑地址。
    - （6） 第13-16字节：主分区的扇区总数。

8.1.6 硬盘启动

- 计算机的控制权就要转交给硬盘的某个分区了
- 四个主分区里面，只有一个是激活的。计算机会读取激活分区的第一个扇区，叫做"卷引导记录"（Volume boot record，缩写为VBR）

8.1.7 操作系统

- 控制权转交给操作系统后，操作系统的内核首先被载入内存。
- 以Linux系统为例，先载入`/boot`目录下面的`kernel`。内核加载成功后，第一个运行的程序是`/sbin/init`。它根据配置文件（Debian系统是/etc/initab）产生init进程。这是Linux启动后的第一个进程，pid进程编号为1，其他进程都是它的后代
- 然后，`init`线程加载系统的各个模块，比如窗口程序和网络程序，直至执行`/bin/login`程序，跳出登录界面，等待用户输入用户名和密码。
