基础篇

环境搭建

镜像安装

- CentOS
- Ubuntu

配置下载源

CentOS

1. **进入/etc/yum.repos.d目录**

```
cd /etc/yum.repos.d/
```

2. **重命名**

```
mv CentOS-Base.repo CentOS-Base.repo.Back
```

3. **下载阿里云的repo文件**

```
wget http://mirrors.aliyun.com/repo/Centos-7.repo
```

4. 修改名称

```
mv Centos-7.repo CentOS-Base.repo
```

5. 执行yum源跟新mingl

```
yum clean all
yum makecache
```

6. 查看yum源

```
yum repolist all
```

Ubuntu

1. **进入/etc/apt/目录**

```
cd /etc/apt/
```

2. 备份

```
cp sources.list sources.list.backup
```

3. 编辑sources.list

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted
universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main
restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main
restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main
restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main
restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted
universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main
restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main
restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main
restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main
restricted universe multiverse
```

4. 更新下载源

```
apt update
```

5. 更新软件

```
# 修复损坏的软件包
apt install -f

# 更新软件
apt upgrade
```

安装Vmware Tools

CentOS

Ubuntu

静态IP设置

CentOS

Ubuntu

修改IP地址后可能遇到的问题

物理机能ping通虚拟机，但是虚拟机ping不通物理机

一般都是因为物理机的防火墙问题,把防火墙关闭就行

虚拟机能Ping通物理机,但是虚拟机Ping不通外网

一般都是因为DNS的设置有问题

虚拟机Ping www.baidu.com 显示域名未知等信息

一般查看GATEWAY和DNS设置是否正确

> - 如果以上全部设置完还是不行,需要关闭(禁用)NetworkManager服务
> - 如果检查发现systemctl status network有问题 需要检查ifcfg-ens33

知识点

入门

Linux背景常识

- 作者：`林纳斯·托瓦兹 (Linus Torvalds)`
- 是什么：是一个基于POSIX和UNIX的多用户、多任务、支持多线程和多CPU的操作系统
- 主要优点
    - 稳定性和可靠性
    - 多任务和多用户
    - 安全性高
    - 成本低
- 常见的发行版
    - RedHat
    - CentOS
    - Ubuntu

Linux内核与Linux发行版的区别

Linux组成

Linux与Windows比较

比较	Linux	Windows

文件

如何理解Linux一切皆文件(Everything is a file)

Linux系统中的一切东西全都可以通过文件的方式进行访问或者管理。反过来说，任何被挂在系统中的东西，即使它们的本质并不是文件，也会被OS以文件的眼光来呈现。

> 1. Linux所有文件的后缀只是方便用户使用,对系统没有任何作用
> 2. 文件=内容+属性。使用C函数接口对文件的操作只是对其内容进行增删改；而只有使用的chmod、chgrep命令才是对文件的属性操作

Linux目录结构

目录	说明
/bin	存放经常使用的命令
/sbin	s表示Super User,存放系统管理员使用的系统管理程序
/home	普通用户的主目录
/root	超级管理员的主目录
/lib	系统开机所需要的最基本的动态连接共享库,类似Windows的DLL文件
/lost+found	一般情况下为空,当系统非法关机后,就会存放一些文件
/etc	所有的系统管理所需要的配置文件和子目录
/usr	用户的应用程序和文件默认存放位置
/boot	Linux启动时的核心文件
/proc	虚拟目录,是系统内存的映射,可以访问这个目录获取系统信息
/srv	存放一些服务启动后需要提取的数据
/sys	
/tmp	存放一些临时文件

目录	说明
/dev	所有硬件以文件形式存储
/mnt	临时挂载别的文件系统
/opt	主机额外安装软件所在目录
/var	存放不断扩充或经常被修改的文件

系统开机流程

`开机 -> BIOS -> /boot -> init进程 -> 运行级别 -> 运行级别对应的服务`

系统运行级别

级别	意义	简化
0	这是系统关闭或关机的级别。在运行级别 0 下，系统会停止所有服务，卸载文件系统，然后关闭计算机。	
1	也称为单用户模式，通常用于维护或修复系统。在运行级别1下，系统启动为单用户模式，只有 root 用户能够登录，网络服务被禁用。	
2	这是多用户模式，但没有 NFS（网络文件系统）支持。在运行级别 2 下，系统会启动多个用户，但不挂载远程文件系统。	
3	这是标准多用户模式，它会启动多个用户并启用所有网络服务。	multi-user.target
4	这通常未使用，您可以根据需要将其配置为自定义模式。	
5	这通常与运行级别 3 相同，但它会使用图形用户界面（GUI）。在运行级别 5 下，通常会启动 X 服务器，以便用户可以使用图形桌面环境。	graphical.target
6	这是系统重新启动的级别。在运行级别 6 下，系统会执行重启操作，类似于运行级别 0。	

进程和服务的区别

计算机中，一个正在执行的程序或命令，被叫做"进程"（process）。启动之后一只存在、常驻内存的进程，一般被称作"服务"（service）。

内置命令与外部命令

一部分基础功能的系统命令是直接内嵌在shell中的，系统加载启动之后会随着shell一起加载，常驻系统内存中。这部分命令被称为"内置（built-in）命令"；相应的其它命令被称为"外部命令"

文件类型

- `-` 表示文件
- `d` 表示目录
- `l` 表示链接文件

rwx对文件和目录的作用

1. 文件
    - `r`:可以读取查看
    - `w`:可以修改，但不能删除文件（删除一个文件的前提是对该文件所在目录有写权限）
    - `x`:可以被系统执行
2. 目录
    - `r`:可以读取，ls查看你目录内容
    - `w`:可以修改，**目录内创建+删除+重命名目录**
    - `x`:可以进入该目录

网络

VMware提供3种网络设置

- 桥接模式

虚拟机直接接连接外部物理网络的模式，主机起到了桥接网络的作用，这种模式下，虚拟机可以直接访问外部网络，并且对外部网络是可见的。

- NAT模式

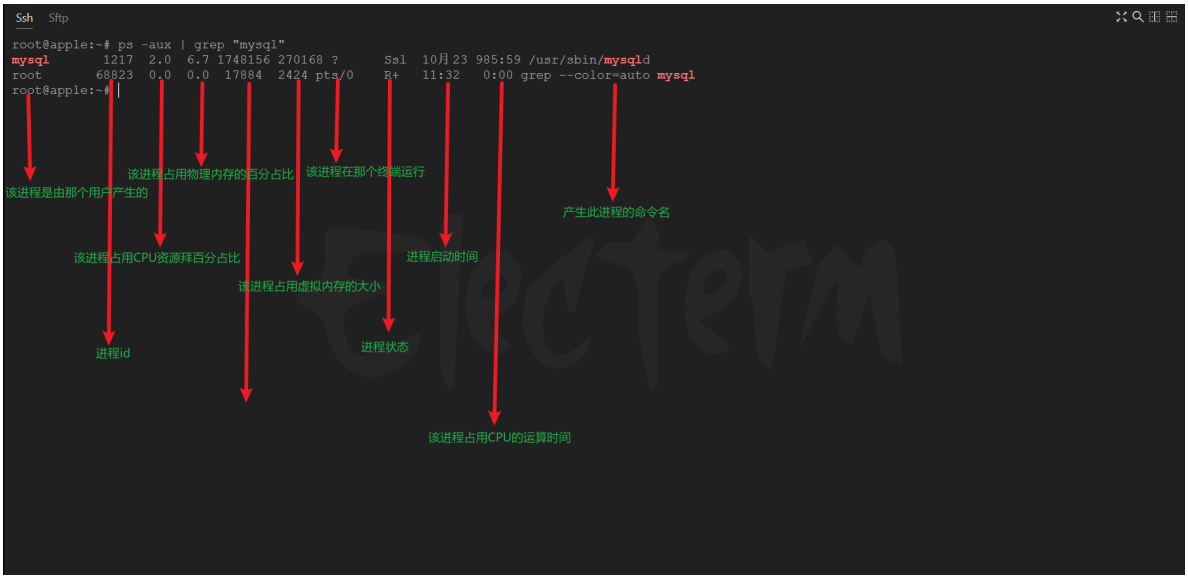
虚拟机和主机构建一个专用网络，并通过虚拟机网络地址转换（NAT）设备对IP进行转换，**虚拟机通过共享主机IP可以访问外部网络，但外部网络无法访问虚拟机**。

- 主机模式

虚拟机只与主机共享一个专用网络，**与外部网络无法通信。**

进程

进程信息



终端类型

- `tty1`:图形化终端
- `pts`:虚拟终端
- `tty2`:本地的字符界面终端

线程状态

- `R(运行状态)`
- `S(睡眠状态)`
- `T(暂停状态)`
- `Z(僵尸状态)`
- `s(包含子进程)`
- `l(多线程)`
- `+(前台显示)`

shell

shell脚本中''与""的区别

`""会解析执行(所见非所得)`,`''原样输出(所见即所得)`

$变量和${变量}的区别

高级篇

Vim使用

一般模式

- 编辑行

功能	语法
复制	yy:复制光标当前行 y数字y:从当前行开始复制几行 y$:复制行首到光标位置 y^:复制光标位置到行尾
粘贴	p
撤销	u
删除	dd:删除当前行 d数字d:从当前行开始删除几行 y$:删除行首到光标位置 y^:删除光标位置到行尾

- 编辑字符

功能	语法
删除单个字符	x:删除当前光标所在字符 X:删除当前光标前一个字符
复制一个单词	yw(**光标需置于单词前**)
删除一个单词	dw(**光标需置于单词前**)

- 移动光标

功能	语法
行首	shift+6(^)
行尾	shift+4($)
页头	1+G

功能	语法
页尾	G
目的行	数字+G(数字+gg)
第一行	gg(H)
屏幕顶行	H
屏幕低行	L

编辑模式

功能	按键
当前光标前	i
光标所在行最前	I
当前光标后	a
光标所在行最后	A
当前光标的上一行	O
当前光标的下一行	o
推出编辑模式	esc

指令模式

- 保存文件

功能	指令
保存	:w
退出	:q
强制执行	:!

- 查找

功能	指令
高亮	:noh(取消高亮显示)
行号	:set nu(设置行号) :set nonu(关闭行号)
/要查找的字符串	n查找下一个,N查找上一个

- 替换

功能	语法
替换当前行第一个匹配的内容	s/old/new
替换当前行所有匹配的内容	s/old/new/g
替换文件每一行匹配的内容	%s/old/new
替换文件所有匹配的内容	%s/old/new/g

软件安装

Linux工具

wget

```
yum -y install wget
```

定时任务

语法

> * * * * * 任务

```
# 编辑
crontab -e

# 查询
crontab -l

# 删除
crontab -r

# 重启定时任务服务
systemctl restart crond
```

时间

序号	含义	范围
第一个*	**一小时中的第几分钟**	0-59
第二个*	**一天中的第几个小时**	0-23

序号	含义	范围
第三个*	一月中的第几天	1-31
第四个*	一年中的第几个月	1-12
第五个*	一周中的星期几	0-7(0和7都代表星期天)

特殊符号	含义
*	代表任何时间。比如第一个"*"就代表一小时中每分钟 都执行一次的意思。
,	代表不连续的时间。比如"0 8,12,16 * * * 命令"，就代表在每天的8点0分，12点0分，16点0分都执行一次命令
-	代表连续的时间范围。比如"0 5 * * 1-6 命令"，代表在周一到周六的凌晨5点0分执行命令
*/n	代表每隔多久执行一次。比如"*/10 * * * * 命令"，代表每隔10分钟就执行一遍命令

设备挂载

正则表达式

特殊字符

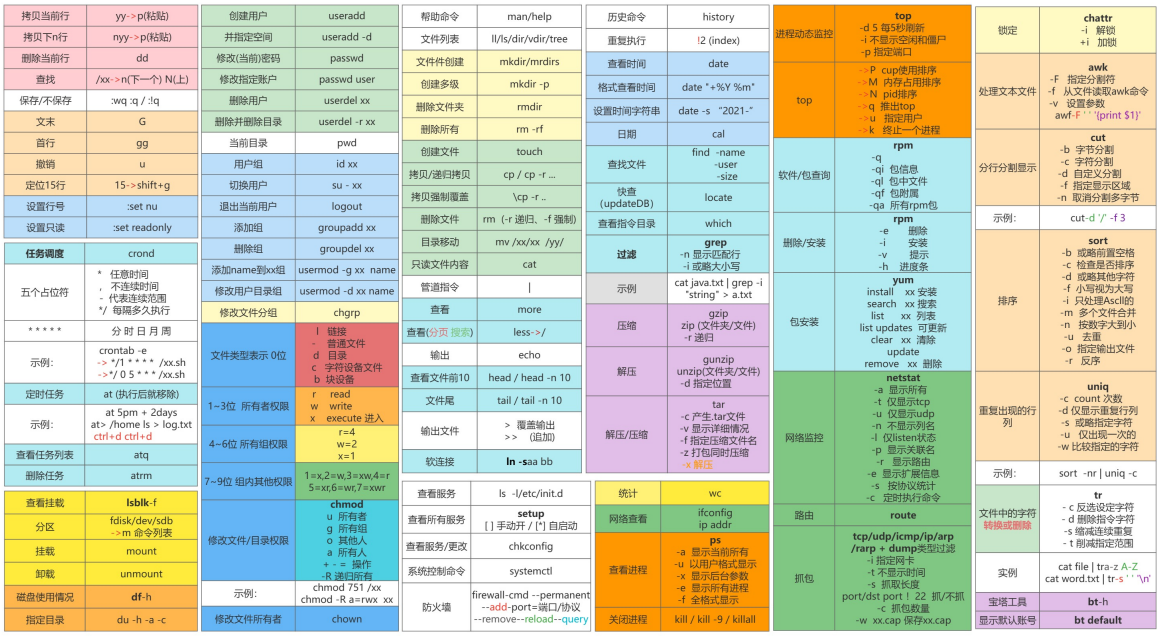
符号	功能	实例
^	匹配开头行	^a(以a开头)
$	匹配结尾行	t$(以t结尾)
.	匹配任意一个字符	r..t(如root)
*	不单独使用，和上一个字符连用，表示匹配上一个字符0次或多	ro*t(如 roooot,rt)

字符区间

符号	功能	实例
[]	配置指定范围的一个字符	[6,8] (匹配6或8),[0-9] (匹配uige0到9的字符),[0-9]*

符号	功能	实例
\	表示转移	'a\$b'(匹配a$b)

命令篇



系统命令

同步

sync

- 将数据由内存同步到硬盘

> Linux 系统中为了提高磁盘的读写效率，对磁盘采取了 "预读迟写"操作方式。当用户保存文件时，Linux 核心并不一定立即将保存数据写入物理磁盘中，而是将数据保存在缓冲区中，等缓冲区满时再写入磁盘，这种方式可以极大的提高磁盘写入数据的效率。但是也带来了安全隐患，如果数据还未写入磁盘时，系统掉电或者其他严重问题出现，则将导致数据丢失。使用sync指令可以立即将缓冲区的数据写入磁盘

关机

halt

- 关闭系统,但不断电

poweroff

- 关机

reboot

- 重启

shutdown

- 关机或重启

```
# 计算机将在 1 分钟后关机，并且会显示在登录用户的当前屏幕中
shutdown -h 1 'This server will shutdown after 1 mins'

# 立马关机（等同于poweroff）
shutdown -h now

# 系统立马重启（等同于reboot）
shutdown -r now
```

用户命令

用户组

groupadd

```
# 添加一个新用户组
groupadd gou

# 查看所有用户组
cat /etc/group
```

groupdel

```
# 删除一个用户组
groupdel gou
```

groupmod

```
# 修改用户组名
groupmod -n xu gou
```

用户

id

```
# 查看用户是否存在
id ming

# 查看有哪些用户
cat /etc/passwd
```

who

```
# 显示自身用户名称
whoami

# 显示登录用户的用户名以及登陆时间
who am i
```

su

```
# 切换用户获取权限,不能获取环境变量
su root

# 切换用户获取权限,获取该用户的环境变量
su - root
```

useradd

```
# 新增用户
useradd ming

# 新增用户到组
useradd -g gou ming
```

passwd

```
# 修改用户密码
passwd ming
```

userdel

```
# 删除用户,保存用户主目录
userdel ming

# 删除用户和主目录
userdel -r ming
```

usermod

```
# 修改用户组
usermod -g root ming
```

权限

```
#  sudo设置普通用户具有root权限
vim /etc/passwd
ming ALL=(ALL) NOPASSWD:ALL
```

chomd

```
# 改变文件权限
# 方式一 chomd {ugoa} {+-=} {rwx} 目录或文件
chmod g+x,a-r apple.txt

# 方式二 chmod {mode-421} 文件或目录
chmod 666 apple.txt

# 修改目录内所有权限
chmod -R 777 fruit/
```

chown

```
# 改变文件所有者
chown tian apple.txt

# 改变文件所有者和所属组
chown -R ming:gou fruit/
```

chgrp

```
# 改变文件所属组
chgrp gou pear.txt
```

文件命令

目录

pwd

- 显示当前工作目录的绝对路径

ls

- 列出目录内容

```
# 列出全部目录,包括隐藏文件
ls -a

# 列出目录详情
ls -l
ll
```

cd

- 切换目录

```
# 切换到指定目录
cd /root/

# 切换到用户家目录
cd
cd ~

# 切换到上次目录
cd -

# 切换到上级目录
cd ../
```

mkdir

- 创建目录

```
# 创建单个目录
mkdir apple

# 创建多级目录
mkdir -p fruit/apple
```

rmdir

- 删除目录

```
# 删除一个空目录
rmdir /apple
```

文件

touch

- 创建空文件

```
# 创建空文件
touch dog.txt
```

cp

- 复制文件或目录

```
# 复制文件
cp apple/hello.txt pear/
cp hello.txt world.txt

# 复制文件夹
cp -r apple/ pear/temp/
```

rm

- 删除文件或目录

```
# 强制删除目录中的所有内容
# -r表示递归删除 -f表示强制执行删除操作 -v表示显示指令的执行过程
rm -rf apple/
```

mv

- 移动文件或目录重命名

```
# 重命名
mv dog.txt cat.txt

# 移动文件
mv apple/hello.txt pear/
```

查看

cat more less

命令	功能
cat	查看较小文件
more	分屏查看文件
less	按需加载显示文件,适合超大文件

- cat

```
# 查看文件内容并显示行号
cat -n dog.txt
```

- more

按键	功能
sapce(空白键)	向下翻一页
enter	向下翻一行
q	退出more显示
ctrl+f	向下滚动一屏

- less

按键	功能
space(空白键)	向下翻一页
pagedown	向下翻一页
pageup	向上翻一页
/ **字符串**	向下搜索 **字符串** :n向下查找,N向上查找
? **字符串**	向上搜索 **字符串** :n向下查找,N向上查找
q	离开less显示

echo

- 输出内容到控制台

```
# 换行符
echo -e aa\\nbb

# 制表符
echo -e aa\\tbb
```

top

- 显示文件前n行内容

```
# 默认查看前10行
top dog.txt

# 查看前20行
top -n 20 dog.txt
```

tail

- 查看文件后n行

```
# 模拟查看文件后10行
tail dog.txt

# 查看文件后20行
tail -n 20 dog.txt

# 实时追踪文档的更新
tail -f dog.txt
```

搜索

find

```
# 在指定目录下根据名称查找文件
find fruit/ --name "*.txt"

# 在指定目录下根据用户查找文件
find fruit/ --user ming

# 在指定目录下根据大小查找文件(+n大于 -n小于 n等于)
find fruit/ --size +200
```

locate

```
# 使用指令前创建locate数据库
updatedb

# 查找文件
# locate apple.txt
```

查找

grep

> 管道符(|):将前一个命令的标准输出传递给后一个命令的标准输入
>
> `grep 选项 查找内容 源文件`

- 选项
  - `-n:显示匹配行及行号`
  -

```
# 管道符(|):将前一个命令的标准输出传递给后一个命令的标准输入
```

awk

- 作用
  一个强大的文本分析工具，把文件逐行的读入，以空格为默认分隔符将每行切片，切开 的部分再进行分析处理
- 语法
  `awk [选项参数] 'pattern1/action1' 'pattern2/action2' filename`
  - 选线参数
    - -F:指定输入文件分隔符
    - -v:赋值一个用户定义变量
  - pattern:表示awk在数据中查找的内容，就是匹配模式
  - action:在找到匹配内容时所执行的一系列命令
- 内置变量
  - FILENAME:文件名
  - NR:已读的记录数
  - NF:浏览记录的域的个数((切割后，列的个数)
- 命令使用
  awk命令详解

cut

- 作用
  cut命令从文件的每一行剪切字节、字符和字段并将这些字节、字符和字段输出
- 参数
  - `-f:列号,提取第几列`
  - `-d:分隔符,按照指定分隔符分割列，默认是制表符“\t"`
  - `-c:按字符进行切割,后加加n表示取第几列`

压缩

gzip

1. 只能将文件压缩成*.gz文件
2. 只能压缩文件,不能压缩目录
3. 不保留原来的文件
4. 同时压缩多个文件会产生多个压缩包

5. 文件后缀为 `.gz`

```
# 压缩文件
gzip apple.txt

# 解压文件
gunzip apple.txt.gz
```

zip

1. 可以压缩文件和目录
2. 保留源文件
3. 文件后缀为 `.zip`

```
# 压缩文件
zip fruit.zip apple.txt pear.txt

# 压缩目录
zip -r /ming

# 解压文件
unzip fruit.zip

# 解压文件到指定目录
unzip fruit.zip -d /ming
```

tar

1. 归档并压缩
2. 文件后缀为 `.tar.gz`

```
# 压缩文件
tar -zcvf fruit.tar.gz apple.txt pear.txt

# 解压文件
tar -zxvf fruit.tar.gz

# 解压文件到指定目录
tar -zxvf fruit.tar.gz -C /ming
```

输出

`> >>`

语法	功能
`>`	覆盖输出(输出重定向)
`>>`	追加输出

`echo`

其他

`ln`

- 创建软连接

```
# 创建软连接
ls -s apple/ apple-link

# 进入软连接对应的目录
cd -P apple-link/
```

> - 查看软连接
>   通过ll就可以查看，列表属性第1位是l，尾部会有位置指向
> - 删除软连接
>   `rm -rf 软链接名`，而不是`rm -rf 软链接名/`,如果使用`rm -rf 软链接名/`删除，会把软链接对应的真实目录下内容删掉

网络命令

查看

`ifconfig`

- 功能:查看网络配置信息

`ping`

- 测试网络连通性

```
ping www.baidu.com
```

hostname

- 修改主机名称`

```
# 查看主机名
hostname

# 修改主机名
vim /etc/hostname
hostnamectl set-hostname 主机名

# 修改hosts映射文件
vim /etc/hosts
```

uname

- 查看内核版本

```
# 详细信息
uname -a
```

netstat

> 参数
>
> - a:显示所有正在监听（listen）和未监听的套接字（socket）
> - n:拒绝显示别名，能显示数字的全部转化成数字
> - l:仅列出在监听的服务状态
> - p:表示显示哪个进程在调用

```
# 查看指定进程的网络信息
netstat -anp | grep sshd

# 查看网络端口的占用情况
netstat -nlp | grep 22
```

磁盘命令

查看

df

```
# 查看磁盘使用清空
df -h
```

lsblk

```
# 查看设备挂在清空
lsblk -f
```

fdisk

```
# 查看系统分区情况
fdisk -l
```

挂载

mount

```
# 挂载设备

# 卸载设备
```

进程命令

查看

ps

> 参数
>
> - a(**列出带有终端的所有用户的进程**)
> - x(**列出当前用户的所有进程，包括没有终端的进程**)
> - u(**面向用户友好的显示风格**)
> - e(**列出所有进程**)
> - u(**列出某个用户关联的所有进程**)
> - f(**显示完整格式的进程列表**)

```
# 查看指定进程
ps -aux | grep "mysql"

# 查看父子进程之间的关系
ps -ef | grep "mysql"
```

runlevel

```
# 查看运行级别
runlevel
systemctl get-default

# 修改当前运行级别  multi-user.target(运行级别3) graphical.target(运行级别5)
# TARGET 取 multi-user 或者 graphical
systemctl set-default TARGET.target
```

pstree

```
# 显示进程id
pstree -p

# 显示进程所属用户
pstree -u
```

top

> 按键
>
> - `P`:以 CPU 使用率排序，默认就是此项
> - `M`:以内存的使用率排序
> - `N`:以 PID 排序
> - `q`:退出 to

```
# 每隔3s跟新系统进程状态
top -d 3

# 不显示任何闲置或僵死进程
top -i

# 根据进程id监控指定进程
top -p 2534
```

设置

systemctl

```
# 查看服务开机启动状态
systemctl list-unit-files

# 关掉服务的自动启动
systemctl disable firewalld
```

```
sudo ufw disable

# 开启服务的自动启动
systemctl enable firewalld
sudo ufw enable

# 查看服务状态
systemctl status firewalld
sudo ufw status

# 临时关闭服务
systemctl stop firewalld
sudo systemctl stop ufw
```

kill

```
# 根据进程id停止进程
kill -9 5425

# 根据进程名称停止进程
killall mysql
```

软件命令

CentOS

rpm

> - 是什么
>   RPM (RedHat Package Manager) , RedHat软件包管理工具, 类似windows里面的setup.exe 是Linux这系列操作系统里面的打包安装工具.
> - 格式
>   - Apache-1.3.23-11.i386.rpm
>   - **apache** 是软件名称
>   - **1.3.23-11** 是软件的版本号, 主版本和此版本
>   - **i386** 是软件所运行的硬件平台, Intel 32位处理器的统称
>   - **rpm** 文件扩展名, 代表RPM包

```
# 查询指定安装包
rpm -qa | grep mysql

# 安装软件包
rpm -ivh mysql
# 安装软件包不检查依赖
rpm -ivh --nodeps mysql

# 卸载软件包
rpm -e mysql
# 卸载软件包时不检查依赖
rpm -e --nodeps mysql
```

yum

> YUM（全称为 Yellow dog Updater, Modified）是一个在 Fedora 和 RedHat 以及 CentOS 中的 Shell 前端软件包管理器。基于 RPM 包管理，能够从指定的服务器自动下载 RPM 包 并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软件包，无须繁琐地一次 次下载、安装，

参数	功能
install	安装rpm软件包
update	更新rpm软件包
check-update	检查是否有可用的更新rpm软件
remove	删除指定的rpm软件
list	显示软件包信息
clean	清理yum过期的缓
deplist	显示yum软件包的所有依赖关

```
# 安装软件
yum -y install firefox
```

其他命令

帮助

man

- 查看命令详情

```
man ls
```

man命令比较适合用来查看shell外部命令的帮助文档，内部命令是直接内嵌在shell中的，系统加载启动 之后会随着shell一起加载，常驻系统内存中；其他的命令成为外部命令，外部命令有些比较大，用的时候才会去加载执行。

help

- 查看shell内置命令信息
- `命令 --help`:查看外部命令的帮助信息

type

- 查看命令类型

clear

- 清屏

reset

- 彻底清屏

时间

date

```
# 显示当前时间
date
# 显示当前年份 月份 那一天
date +%Y
date +%m
date +%d

# 按照指定格式显示时间
date "+%Y-%m-%d %H:%M:%S"

# 显示指定字符串的时间
date -d "2022-10-10 12:12:30"
# 显示前一天
date -d '1 days ago'
```

```
# 显示后一天
date -d '-1 days ago'

# 设置系统时间
date -s "2023-06-19 20:52:10
```

cal

```
# 显示本月日历
cal

# 显示只从年份日历
cal 2023
```

Shell篇

概述

shell是什么

shell是一个命令行解释器,它应用程序或用户命令,然后调用操作系统内核

shell解析器有哪些

- `bash`
- sh(bash的软链接)
- `dash`

shell脚本格式

```
# 指定解析器
#!/bin/bash

# 命令内容
echo "hello world"
```

脚本执行方式

- `bash` 脚本
  - 不需要权限,解释器直接执行
- `./` 脚本
  - `#!/bin/bash` 告诉系统使用什么解释器执行
  - 脚本需要执行权限
- `.或source` 脚本

> 1. 前两种方式都是在当前shell中打开一个子shell来执行脚本内容，当脚本内容结束，则子shell关闭，回到父shell中
> 2. 第三种在脚本路径前加""`.`"或者 `source` 的方式，可以使脚本内容在当前shell里执行，而无需打开子shell,这也是为什么我们每次要修改完/etc/profile 文件以后，需要source一下的原因
> 3. 开子shell与不开子shell的区别就在于 **环境变量的继承关系** ，如在子shell中设置的当前变量，父shell是不可见的

变量

系统变量

- 常见的系统变量
  - **$HOME**
  - **$USER**
  - $PWD
- 查看所有系统变量

```
set
```

- 加载系统变量

```
# CentOS
source /etc/init.d/functions

# Ubuntu
```

自定义变量

定义语法

- 定义变量： **变量名=变量值**
    - =号前后不能有空格
- 撤销变量： unset 变量名
- 声明静态变量(不能unset)： readonly 变量名=变量值

定义规则

- 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 ，**环境变量名建议大写**
- 等号两侧不能有空格
- **在bash中，变量默认类型都是字符串类型，无法直接进行数值运算**
- 变量的值如果有空格，需要使用双引号或单引号括起来

提升全局变量

```
export 变量名

# 将变量user提升全局变量
export user
```

方法变量

$n

> n为数字， **$0代表该脚本名称** ， $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}

$#

> 获取 **所有输入参数个数** ， 常用于循环,判断参数的个数是否正确以及加强脚本的健壮性

$*

> 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， **$@把每个参数区分对待**

$?

> 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（具体是哪个数，由命令自己来决定），则证明上一个命令执行不正确了

条件判断

判断语法

运算符

`$((运算式))` `$[运算式]`

判断逻辑

- `test condition`
- `[ condition ]` （condition前后要有空格）

> 条件非空即为true,[test]返回true,[ ]返回false

常见判断条件

- 数值

符号	作用
`-eq`	等于

符号	作用
`-ne`	不等于
`-lt`	小于
`-le`	小于等于
`-gt`	大于
`-ge`	大于等于

- 字符串

符号	作用
=	等于
!=	不等于

- 文件权限

符号	作用
-r	读权限
-w	写权限
-x	执行权限

- 文件类型

符号	作用
-e	文件或目录存在
-f	文件存在
-d	目录存在

- 逻辑判断

符号	作用
&&	表示前一条命令执行成功时，才执行后一条命令
||	表示上一 条命令执行失败后，才执行下一条命令

```
[ test ] && echo "OK" || echo "NOT OK"
```

```
# 数值
[ 23 -ge 22]

# 文件权限
[ -w test.sh]
```

if

- 单分支

```
if [ ondition ]
then
    程序
fi

if [ ondition ]
then
    程序1
else
    程序2
fi
```

- 多分枝

```
if [ condition ]
then
    程序1
elif [ condition ]
then
    程序2
else
    程序3
fi
```

1. [条件判断式]条件判断的中括号和条件判断式之间必须有空格
2. if后要空格

case

- 语法

```
case $变量名 in
"值1")
    程序1
;;
"值2")
    程序2
;;
*)
    上述判断都不成立,执行此程序
;;
esac
```

1. case行尾必须为单词"in"，每一个模式匹配必须以右括号")"结束
2. 双分号;;"表示命令序列结束，相当于java中的break
3. 最后的""*)"表示默认模式，相当于java中的default

循环控制

for

- 方式一

```
for ((初始值;循环控制条件;变量变化))
do
    程序
done
```

```
# 从1加到100
#!/bin/bash

sum=0
for(i=0;i<=100;i++)
do
    sum=$[$sum+$i]
done

echo $sum
```

- 方式er

```
for 变量 in 值1 值2 值3 ...
do
    程序
done
```

```
# 打印所有输入参数
#!/bin/bash

for i in $@
do
    echo "the value is $i"
done
```

1. $*和$@都表示传递给函数或脚本的所有参数，不被双引号""包含时，都以$1 $2 …$n 的形式输出所有参数
2. 当它们被双引号""包含时，$*会将所有的参数作为一个整体，以""$1 $2 …$n"的形式输出所有参数；$@会将各个参数分开，以"$1" "$2"…"$n"的形式输出所有参数。

while

```
while [ condition ]
do
    程序
done
```

```
# 从1加到100
#!/bin/bash

sum=0
i=1
while [ $i -le 100 ]
do
    sum=$[$sum+$i]
    i=$[$i+i]
done

echo $sum
```

函数控制

- 控制台输入

```
#!/bin/bash

read -t 7 -p "Enter your name in 7 seconds : " name
echo "your name is $name"
```

1. **-p指定读取值时的提示符**
2. **-t指定读取值等待的时间(秒),如果不加-t表示一直等待**

3. 最后变量为指定读取值的名称

系统函数

basename

```
# 获取路径里的文件名称
basename /root/fruit/apple.txt

# 获取路径里的文件名称,不带后缀
basename /root/fruit/apple.txt .txt
```

dirname

```
# 获取文件路径的绝对路径
dirname /root/fruit/apple.txt
```

自定义函数

语法

```
[ function ] functionName[()]
{
    Action;
    [return int;]
}
```

1. 必须在调用函数地方之前，先声明函数，shell脚本是逐行运行。不会像其它语言一样先编译
2. 函数返回值只能通过$?系统变量获得，可以显示加：return返回，如果不加，将以最后一条命令运行结果，作为返回值。return后跟数值 n(0-255)

```
#!/bin/bash
function sum()
{
s=0
s=$[$1+$2]
echo "$s"
}

read -p "Please input the number1: " n1;
read -p "Please input the number2: " n2;
sum $n1 $n2;
```