一　虚拟机的联网方式

1　桥接

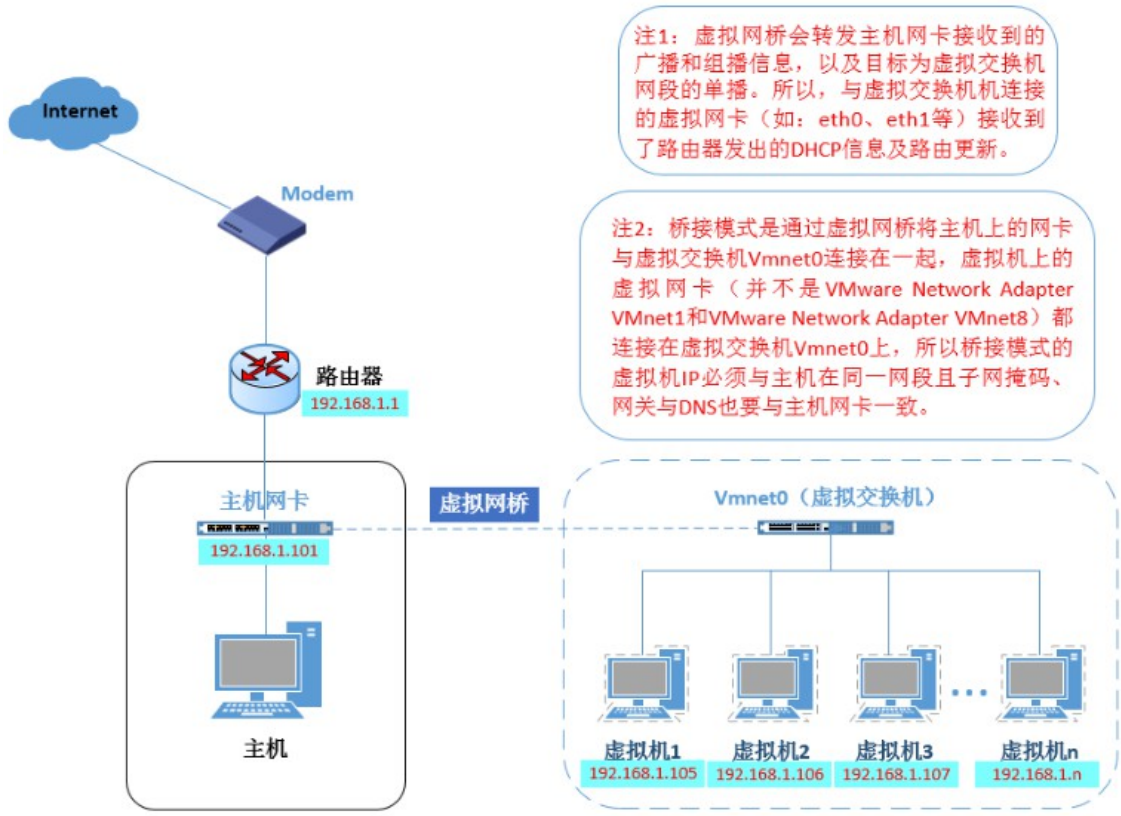


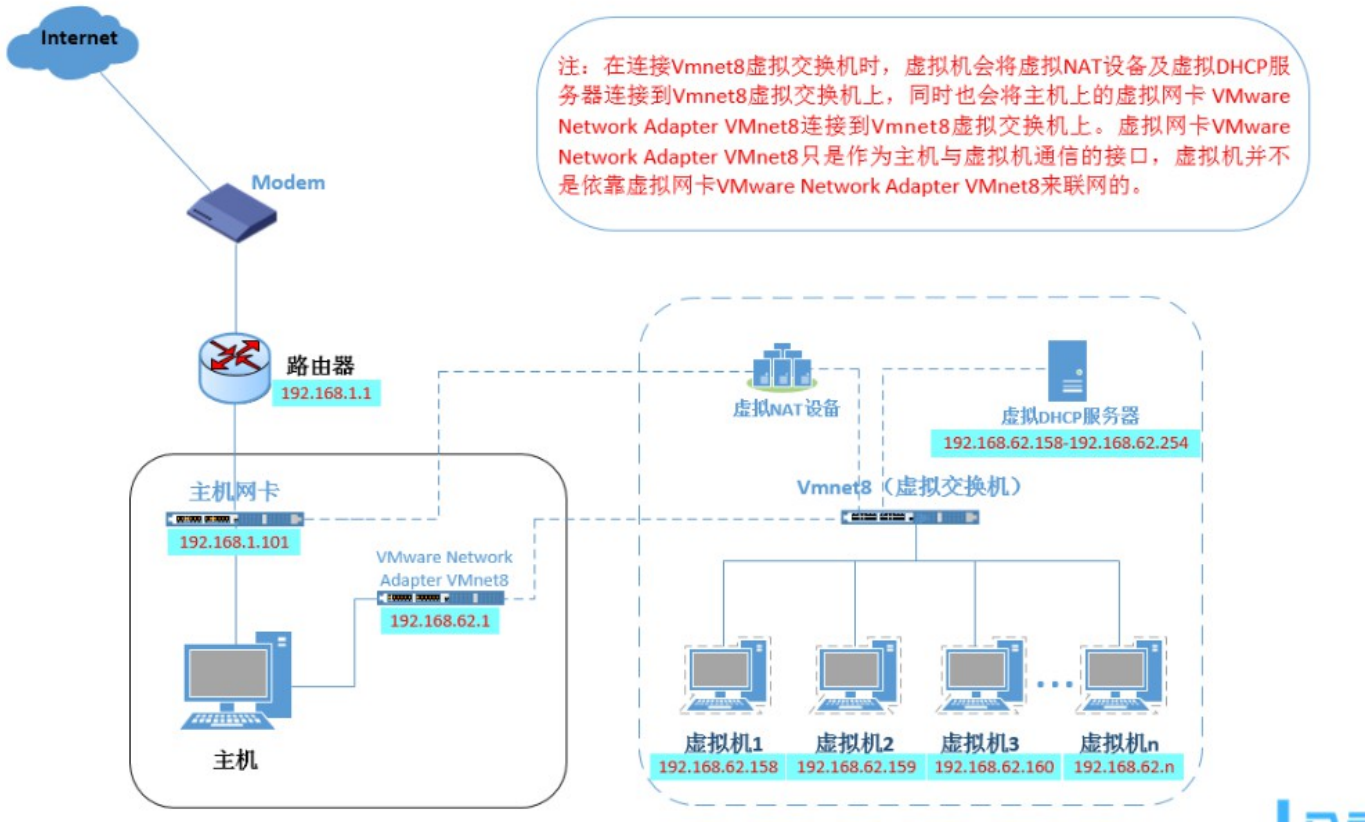
图1 桥接模式

1.A 和主机共用网段

1.B 同一网段可以直接相互通信

1.C 虚拟机的网络依赖于主机的网络环境

2 NAT



2.A 其他物理机想要访问 NAT 模式下的虚拟机时，比较麻烦。

2.B 可以无视物理机(宿主主机)网络环境。即便是物理机没有网络，也不影响本机和虚拟机进行通信，也不影响本机上的其他虚拟机之间互相通信。因为虚拟机真正通信网卡是 VMNet8 提供(网络环境)。

2.C 即使 VMNet8 虚拟网卡被禁用，虚拟机同样能访问外网，VMNet8 虚拟网卡只是用来主机访问虚拟机使用

二 Linux 网络相关命令

1 查看域名 IP(host)

```
[root@SHU1 home]# host www.baidu.com
www.baidu.com is an alias for www.a.shifen.com.
www.a.shifen.com has address 14.215.177.38
www.a.shifen.com has address 14.215.177.39
```

2 远程拷贝文件(scp)

2.A 从本机拷贝数据到远程的服务器上

要求：必须知道对方的账户和密码，且具备相应的权限。

语法：scp [-r] <fie | dir> {UserName}@Host_IP:/[path]

-r	该选项用于传输文件夹的时候使用。

注意， 如果是第一次访问该服务器，那么会询问，是否要继续连接。每次访问都需要输入远程服务器的密码。

拷贝 123 文件到 192.168.237.12 的/root/home 目录，并用 root 账户登录

```
[root@SHU1 ~]# scp 123 root@192.168.237.12:/root/home
123 100%    0     0.0KB/s   00:00
```

2.B 从远程获取

要求：必须知道对方的账户和密码，且具备相应的权限。

语法：scp {UserName}@Host_IP:/[path]/file /[path]

将远程主机上的/home/install.log 文件拷贝到本机的/root 文件夹

```
[root@SHU1 ~]# scp root@192.168.237.12:/home/install.log  /root
```

3 远程登录(ssh)

3.A 语法：

ssh [-p port]<用户名>@<主机 IP>
默认是 22 端口，若不是则需通过-p 指定
回车之后，如果首次访问，会提示是否继续连接。接下来要求输入远程服务器的密码。

3.B 登录

```
[root@SHU1 ~]# ssh root@192.168.237.12
```

3.C 退出

```
[root@SHU2 ~]#exit
```

3.D 通过主机名登录

远程登录的时候可以通过主机名进行登录

3.D.a 配置服务器的主机名

1、永久修改：vim /etc/sysconfig/network -->HOSTNSME=主机名（重启才能生效）
2、临时修改：hostname 主机名

3.D.b 配置/etc/hosts 文件，将服务器 ip 和主机名做一一对应

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
calhost.localdomain localhost6 localhost6.localdomain6
192.168.237.12 SHU2
~
```

4 远程下载(wget)

用于从网络上下载资源，没有指定目录，下载资源默认存储到当前目录。

4.A 格式：

wget [参数] <URL 地址>

4.B 特点

–支持断点下载功能
–同时支持 FTP 和 HTTP 下载方式
–支持代理服务器

4.C 下载单个文件

```
[root@SHU1 ~]# wget http://www.shuxinsheng.com
```

4.D 下载单个文件并以不同名称保存(-O)

```
[root@SHU1 ~]# wget -O NewName.new  http://www.shuxinsheng.com
```

4.E 限速下载(--limit-rate)

```
[root@SHU1 ~]# wget --limit-rate=300k  http://www.shuxinsheng.com
```

4.F 断点续传(-c)

```
[root@SHU1 ~]# wget -c www.shu.com
```

4.G 后台下载(-b)

```
[root@SHU1 ~]# wget -b www.baidu.com
```

4.H 下载多个文件(-i)

```
[root@SHU1 ~]# wget -i urlfile.txt
```

urifile 文件名称仅仅为了见名知意。

urlfile.txt 内容为

www.baidu.com/123.txt

www.baidu.com/456.txt

5 防火墙

它具备一定的防护功能，比如说端口的开放和禁止，也可做数据的转发(类似路由功能)，策略及其他功能。

5.A 临时处理防火墙

如果系统重启，那么防火墙将恢复到之前的状态。

5.A.a 开启

```
[root@SHU1 ~]# service iptables start
```

Or

```
[root@SHU1 ~]#  /etc/init.d/iptables start
```

5.A.b 关闭

```
[root@SHU1 ~]# service iptables stop
```

Or

```
[root@SHU1 ~]# /etc/init.d/iptables stop
```

5.A.c 重启

```
[root@SHU1 ~]# service iptables restart
```

Or

```
[root@SHU1 ~]# /etc/init.d/iptables restart
```

5.A.d 查看

```
[root@SHU1 ~]# service iptables status
```

Or

```
[root@SHU1 ~]# /etc/init.d/iptables status
```

5.B 永久处理防火墙

永久处理防火墙：(需重启系统后才能生效)

5.B.a 开启

```
[root@SHU1 ~]# chkconfig iptables on
```

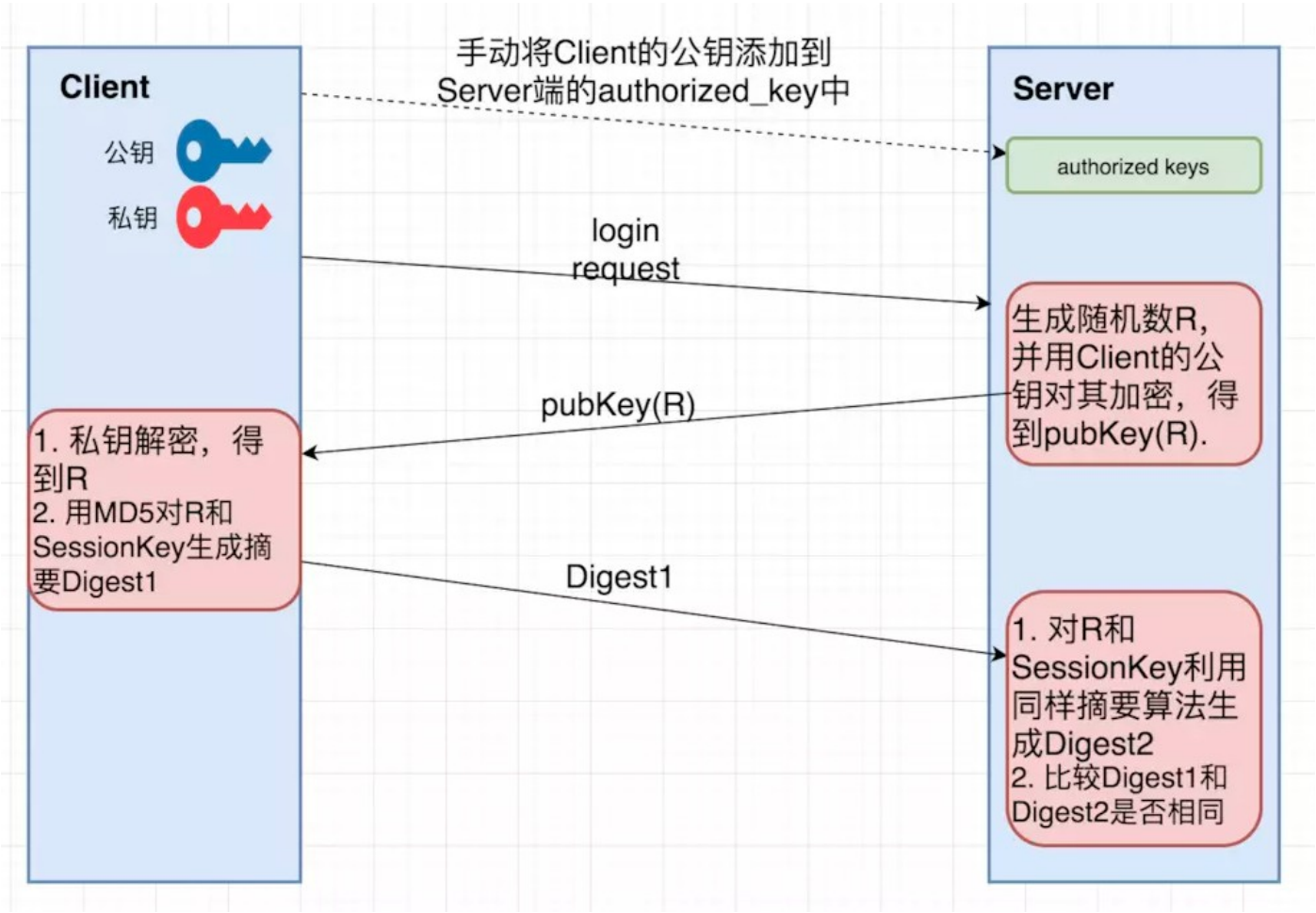
5.B.b 关闭

```
[root@SHU1 ~]# chkconfig iptables off
```

5.B.c 查看状态

```
[root@SHU1 ~]# chkconfig iptables --list
```

三 Linux 免密登录



1 证书使用场景：

1.A 场景一：只是单纯的使用证书来登录服务器。

使用证书的登录方式可以避免密码遗忘、泄漏的问题。

使用证书登录服务器的方式也是服务器加固(服务器安全相关问题)的方式。

服务器可以设置不允许使用密码进行远程登录。只允许证书的方式登录。

证书本身支持加密，就算证书丢失，再不知道证书密码的情况，证书属于无效文件。

1.B 场景二：集群中使用证书进行免密登录。

因为但凡设计到集群的时候，一般都不会是小数目的服务器数量。众多的服务器之间进行互相访问，频繁的输入密码的事情将会成为开发工程师噩梦。

所以，使用证书管理集群的时候，可以免除集群中的服务器互相访问时工程师手工输入密码的问题。

2 方式一(适用大量服务器)

2.A Server① 访问 Server②

2.A.a Server① 生成密钥 会生成公钥和私钥

第一次提示：你的证书文件存放位置

第二次提示：对私钥加密，输入密码。如果不需要输入密码，直接回车。

第三次提示：私钥证书的密码确认操作。

生成的二个文件 id_rsa

```
[root@SHU1 .ssh]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
6b:f8:b8:d2:b4:57:2e:4c:ce:06:69:b9:fd:90:40:0b root@SHU1
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|   E .           |
|    o .          |
|     ooS         |
|     *o.o.       |
|    +.X=o        |
|   . ++O..       |
|    .o+.o.       |
+-----------------+
```

```
[root@SHU1 .ssh]# ll
总用量 8
-rw-------. 1 root root 1675 5 月  15 21:23 id_rsa //私钥
-rw-r--r--. 1 root root  391 5 月  15 21:23 id_rsa.pub//公钥
```

2.A.b Server① 注册到 Server2

```
ssh-copy-id {UserName}@Host_IP
```

```
[root@SHU1 .ssh]# ssh-copy-id root@192.168.237.12
The authenticity of host '192.168.237.12 (192.168.237.12)' can't be established.
RSA key fingerprint is c2:af:8e:d4:15:8e:5d:93:21:83:9e:fe:de:20:89:13.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.237.12' (RSA) to the list of known hosts.
root@192.168.237.12's password://输入密码
```

Server2 会生成一个 authorized_keys，实际就是保存 Server1 注册过来的公钥

```
[root@root2 .ssh]# ll
总用量 4
-rw-------. 1 root root 789 5 月  15 21:29 authorized_keys
```

此时可以通过 Server① 免密登录 Server2, 但 Server2 无法登录 Server①

2.B 使 Server2 能访问 Server1:

在之前的基础上，将私钥拷贝到 Server2，同时在 Server1 上给自己注册一下
此时二者共用一对私钥

2.B.a Server1 拷贝私钥到 Server2

```
[root@SHU1 .ssh]# scp id_rsa root@192.168.237.12 /root/.ssh
```

2.B.b Server1 给自己注册

```
[root@SHU1 .ssh]# ssh-copy-id 127.0.0.1
```

以上的操作都是在一台服务器上运行，所以适用大量服务器

3 方式二(适用少量服务器)

方式一种，Server1 访问 Server2，通过生成公私钥，然后注册到 Server2 即可

另一种方式 Server2 访问 Server1，同样在 Server2 上生成公私钥，注册到 Server1 即可

这种方式需要双方操作，大量服务器增加工作量

4 生成的文件解释

证书文件会存放在当前账户的家目录下的隐藏目录".ssh"目录下，在该目录下会有以下 4 个文件：

id_rsa	私钥	执行证书生成命令才会有
id_rsa.pub	公钥	执行证书生成命令才会有
known_hosts	曾经访问过的服务器信息	每次 ssh、scp、ssh-copy-id 到远程服务器时就会保存记录到此文件中，以后再此访问该服务器时就不会再提示那一句"你确定要继续访问吗 yes/NO？"
authorized_keys	记录来访服务器的公钥文件内容	该文件会记录访问本机的远程服务器的公钥证书文件内容，只有对应的私钥才能进行验

四 Linux 进程

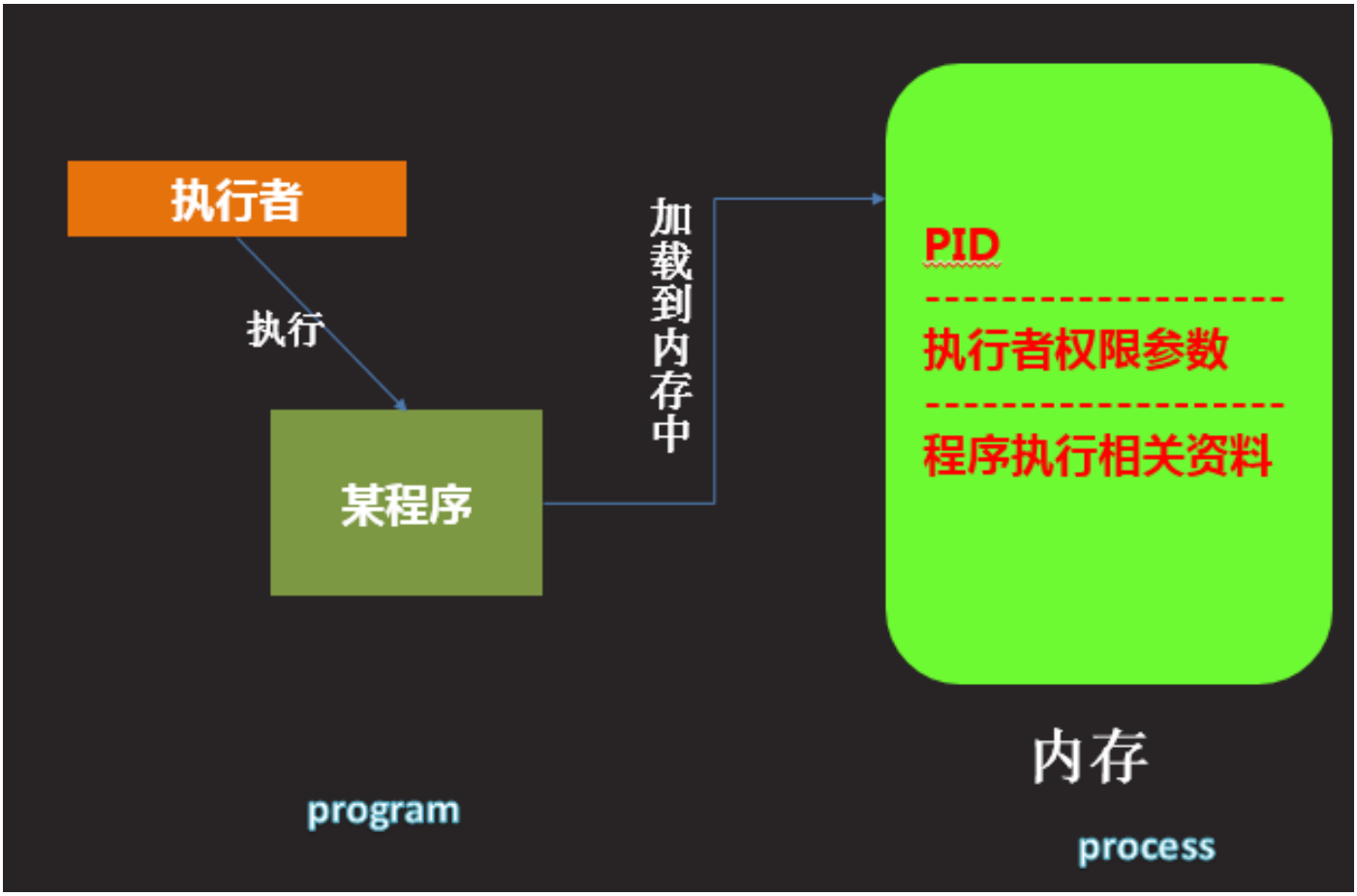
进程通俗来说，运行中的程序，

在 linux 下，知道程序要运行，首先就是将磁盘中相应的可执行文件加载到内存中，那么我们怎么知道他在内存中哪呢？这个时候就需要我们通过一个叫做进程标识符的东西找到它。类似于我们自己的身份证

进程分为临时进程和持久进程（守护进程）

临时进程	执行完命令，自动结束
持久进程	程序运行后，需要手动结束。

程序被加载为进程的示意图：



1、用户执行程序。

2、程序加载到内存中。

2.1、给程序一个临时的 pid

2.2、查看执行的权限，如果用户没有执行权限，那么拒绝操作，如果有，开始加载程序执行的相关资料（内存指针开始扫描响应数据和执行文件）

2.3、确认临时的 PID。

1 进程查看

1.A 静态查询(ps/pstree)

语法: ps [参数]

1.A.a 常用参数 -aux

a	关联的所有 process，通常与 x 一起使用，列出完整信息。
x	后台进程
u	有效使用者的相关联的进程
ajxf	可以让 ps 的结果以树状的格式显示出来。

查询特定进程用

```
ps -aux | grep  sshd
```

1.A.b 查询结果解释

ps 查询结果各项解释：

USER	用户
PID	进程 ID
%CPU	cpu 占用率
%MEM	内存使用率
VSS	虚拟内存使用量（swap 交换分区使用量）
RSS	物理内存使用量
TTY	tty1-tty6 是本机上面的登入者程序。 pts/0 等等的,则表示为由网络连接进主机的程序。 如果显示？则表示与终端机无关。
STAT	进程的状态
START	进程启动的时间
TIME	累计消耗 CPU 的时间
COMMAND	表示哪个命令/程序运行的该进程

状态标识：

R	正在运行，或在队列中的进程
S	处于休眠状态
I	多进程
Z	僵尸进程
T	停止或者被追踪
<	高优先级
N	低优先级
s	包含子进程
+	位于后台的进程组

1.A.c 僵尸进程：

由于该进程已经执行完毕，但是父进程没有终止或其他原因导致该进程并没有真正的结束，所形成的进程称之为僵尸进程。
此进程对服务器的危害在于它会持续的消耗服务器资源，消耗量会越来越大。最终导致其他的进程无资源可用，服务器崩溃。

1.A.d 显示进程树(pstree)

选项：

-A	各程序之间的连接以 ASCII 字符来连接
-U	各程序之间的连接以 UTF-8 的字符来连接

-u	列出每个 process 的所属账号名称
-p	同时列出每个程序的进程的 ID

1.B 动态查询(top)

动态查询系统的进程状态。默认是 3 秒一更新。

1.B.a 参数

-d	跟时间，可以修改 top 默认更新(刷新)的时间
-b :	以批次的方式执行 top ,还有更多的参数可以使用,通常会搭配数据流重导向来将批次的结果输出成为档案;
-n Number:	与 -b 搭配,意义是需要进行几次 top 的输出结果;
-p :	指定某些个 PID 来进行观察监测而已;

1.B.b 每秒刷新一次 top

```
[root@SHU1 .ssh]# top -d 1
```

1.B.c 每 2 秒刷新一次 top，以批次输出 2 次。

```
[root@SHU1 .ssh]# top -d 2 –n 2
```

1.B.d 每秒刷新一次 top，以批次输出 5 次。

```
[root@SHU1 .ssh]# top -d 1 -b –n 5 >>top.log
```

 # >>表示以追加的方式输出，>表示以覆盖的方式输出

1.B.e 交互式按键：(并不常用)

用了 top 命令后，可以按以下键交互

? :	显示在 top 当中可以输入的按键指令
P :	以 CPU 的使用资源排序显示
M :	以 Memory 的使用资源排序显示
N :	以 PID 来排序
T :	由该 Process 使用的 CPU 时间累积 (TIME+) 排序
q :	离开 top 软件的按键

2 进程管理

2.A 单进程管理（kill）

结束某个线程

2.A.a 语法：

kill <信号量> <PID>

2.B 多进程管理(killall)

结束基于某个程序运行的进程(结束进程树？)

2.B.a 语法

killall <信号量> <程序名/命令名>

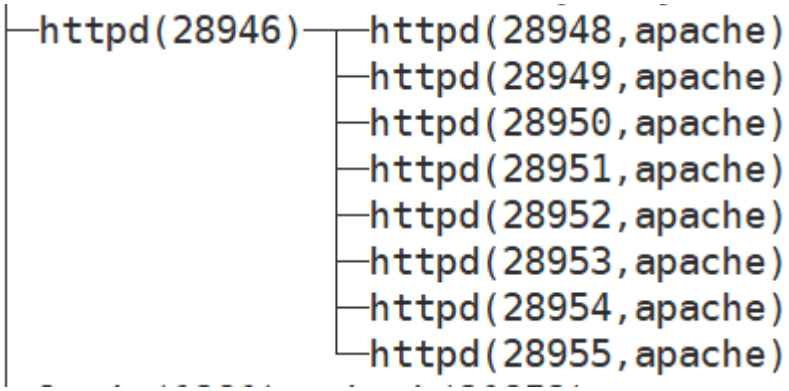
2.C 信号量：

-15:	以正常的程序方式终止一个进程！！！
-9 :	立刻强制终止一个进程！！！ （！！不能强制结束系统级别的进程）
-2 :	代表由键盘输入 [ctrl] + c 同样的动作;
-1 :	对于 sshd 这样的守护进程，重新读取一次参数的配置文件 (类似 reload)，如果进程为非守护进程，默认为终止进程;！！

2.D 案例

2.D.a 结束所有 httpd 的进程（如果没有可以先安装一下 yum install -y httpd）

```
# bash
    killall -9 httpd
```



2.D.b 结束所有 java 的进程

```
# bash
    killall -9 java
```

五 Linux 系统资源监控

1 (free)内存监控

1.A 选项：

-b	bytes
-k	kb
-m	mb
-g	gb
-t	统计总量

1.B 清理缓冲区

```
[root@SHU1 .ssh]# echo 3 > /proc/sys/vm/drop_caches
```

2 (uname)查阅系统与核心相关信息

2.A 选项：

-a	所有系统相关的信息,包括以下的数据都会被列出来;

-s	系统内核名称
-r	内核版本
-m	本系统的硬件名称,例如 i686 或 x86_64 等;
-p	CPU 的类型,与 -m 类似,是显示的是 CPU 的类型;
-i	硬件的平台(ix86);

3 (uptime)观察系统启动时间与工作负载

```
[root@SHU1 tomcat6]# uptime
 19:47:11 up  5:26,  1 user,  load average: 0.00, 0.00, 0.00
[root@SHU1 tomcat6]#
```

19:47:11	系统当前的时间
up 5:26	系统运行时间
1 user	当前有两个用户登录
load average: 0.00, 0.00, 0.00	系统过去的 1,5,15 分钟的平均负载

4 (netstat)网络监控

4.A 选项

-a	将目前系统上所有的已经连接、监听、Socket 数据都列出来
-t	列出 tcp 网络包的信息
-u	列出 udp 网络包的信息
-n	以端口(port number)方式来显示（不以程序的服务名称）
-l	列出目前正在监听(listen)的服务;
-p	列出该网络服务的进程 id（PID）、程序名

4.B 案例

列出当前系统中正在监听的 TCP 服务。

```
# bash
netstat -lt
```

列出当前系统中正在监听的 TCP 服务，并且显示进程 ID。

```
# bash
netstat -ltp
```

列出当前系统中正在监听的 TCP 服务，并且显示进程 ID、端口号。

bash

netstat -lntp

列出当前系统中已连接的 TCP 服务，并显示进程 ID、端口号。

bash

netstat -tnp

监听 udp 一般监听不出来

4.C 结果解释

```
[root@SHU1 tomcat6]# netstat -tnp
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 192.168.237.11:22       192.168.237.1:53793     ESTABLISHED 2074/sshd
```

Proto	协议名
Recv-Q	接收消息缓冲区
Send-Q	发送消息缓冲区
Local Address	本地地址和端口号
Foreign Address	远程地址和端口号
State	状态。连接、监听
PID/Program name	进程 ID 和程序名

5 (vmstat)侦测系统资源变化

统计目前主机 CPU 状态,每秒一次,共计四次

[root@SHU1 ~]# vmstat 1 4

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 662856  54876 151320    0    0     2     1   10   10  0  0 100  0  0
 0  0      0 662816  54876 151320    0    0     0     0   17   10  0  0 100  0  0
 0  0      0 662816  54876 151320    0    0     0     0   20   12  0  0 100  0  0
 0  0      0 662816  54876 151320    0    0     0     0   13   10  0  0 100  0  0
```

5.A 参数解释

- procs（进程字段 ）

r :	等待运行的进程数量; cup 处理不过来
b:	不可被唤醒的进程数量

这两个项目越多,代表系统越忙碌 (因为系统太忙,所以很多进程就无法被执行或一直在等待而无法被唤醒)

- memory (内存字段)

swpd:	虚拟内存被使用的容量;
free:	未被使用的内存容量;
buff/cache:	用于缓冲的内存;

- swap（交换分区字段） （重点记忆下 si 和 so）

si:	每秒从交换分区写到内存的数据量大小，由磁盘->内存;
so:	每秒写入交换分区的内存数据量大小，由内存->磁盘。

如果 si/so 的数值太大,表示内存内的数据常常得在磁盘与主存储器之间传来传去,系统效能会很差

- io(磁盘读写字段)

bi:	从块设备读入数据的总量（读磁盘）（每秒 kb）;
bo:	从块设备写入数据的总量（写磁盘）（每秒 kb） 。

如果这部份的值越高,代表系统的 I/O 非常忙碌

- system（系统字段 ）

in:	每秒被中断的进程次数; 发生在 cup 争抢的过程中
cs:	每秒钟进行的事件切换次数。发生在 cup 争抢的过程中

这两个数值越大,代表系统与接口设备的通信非常频繁

- CPU (cpu 字段)

us:	(user)非内核态的（用户进程） CPU 使用情况;
sy:	(system）内核态所使用（系统进程）的 CPU 情况;
id:	(idle ）闲置的 CPU 情况;
wa:	(wait)等待 I/O 所耗费的 CPU;
st:	被虚拟机(virtual machine)所盗用的 CPU(2.6.11 以后才支持)

六 任务管理

1 分类

前台任务：可以控制与执行命令的 bash 环境称为前台。
后台任务：在操作系统中自行运行,你无法使用[ctrl]+c 终止称为后台。

2 管理

2.A 前台任务切换到后台并暂停

Ctrl + z 就可以将前台的任务放置后台

2.B 如何运行任务时，使其在后台运行：

在运行命令之前加上"&"
例如：

cp file1 file2 &

不是所有的任务都能够在后台运行的，比如需要与用户进行交互的程序或命令就不允许在后台运行， 比如 vi 文本编辑器

2.C 查看后台任务(jobs)

-r	仅查看后台运行的任务
-s	仅查看后台暂停的任务
-l	查看后台的任务，并显示其 PID

2.D 如何将后台任务调至前台：

fg [jobnumber]

fg 命令+jobnumber 来把后台任务调至前台。(无论在后台是暂停还是运行)

fg 命令不加 jobnumber 也是可以调后台的任务，但是默认就会调取后台带有+号的那个任务。最后放置后台的任务就会带有+号。

+	表示最近一次放置后台的任务

2.D.a 案例：

```
[root@bogon ~]# jobs
[1]   Stopped                 vi 1.txt
[2]-  Stopped                 vi 2.txt
[3]+  Stopped                 vi 3.txt
```

调取 2 号任务：

bash

fg 2

2.E 如何将后台任务修改为运行状态：

bg [joinnumber]

bg 命令 + jobnumber 可以将后台任务的暂停状态修改为运行状态。(交互式的应用无法修改为运行状态)

bg 命令不加 jobnumber 也是可以调后台的任务，但是默认就会调取后台带有+号的那个任务。（最后放置后台的任务就会带有+号。）

案例:

```
[root@bogon ~]# jobs
[1]   Stopped                 vi 1.txt
[2]-  Stopped                 vi 2.txt
[3]+  Stopped                 vi 3.txt
```

修改2号任务的后台状态:

```
# bash
    bg 2
```

失败案例！！

```
[root@bogon ~]# bg 2
[2]- vi 2.txt &
[root@bogon ~]#

[2]+  Stopped                 vi 2.txt
[root@bogon ~]#
```

```
[root@bogon ~]# jobs
[1]   Stopped                 vi 1.txt
[2]-  Stopped                 vi 2.txt
[3]   Stopped                 vi 3.txt
[4]+  Stopped                 cp -i -r /home/cdrom/ /root/home/
```

修改4号任务在后台的工作状态。

```
# bash
    bg
```

```
[root@bogon ~]# bg
[4]+ cp -i -r /home/cdrom/ /root/home/ &
[root@bogon ~]# jobs
[1]-  Stopped                 vi 1.txt
[2]+  Stopped                 vi 2.txt
[3]   Stopped                 vi 3.txt
[4]   Running                 cp -i -r /home/cdrom/ /root/home/ &
[root@bogon ~]#
```

2.F 终止 job:

jobs -l 查询出 ID，之后通过 kill -9 PID 结束