一、远程管理Linux服务器

1、windows下远程连接工具

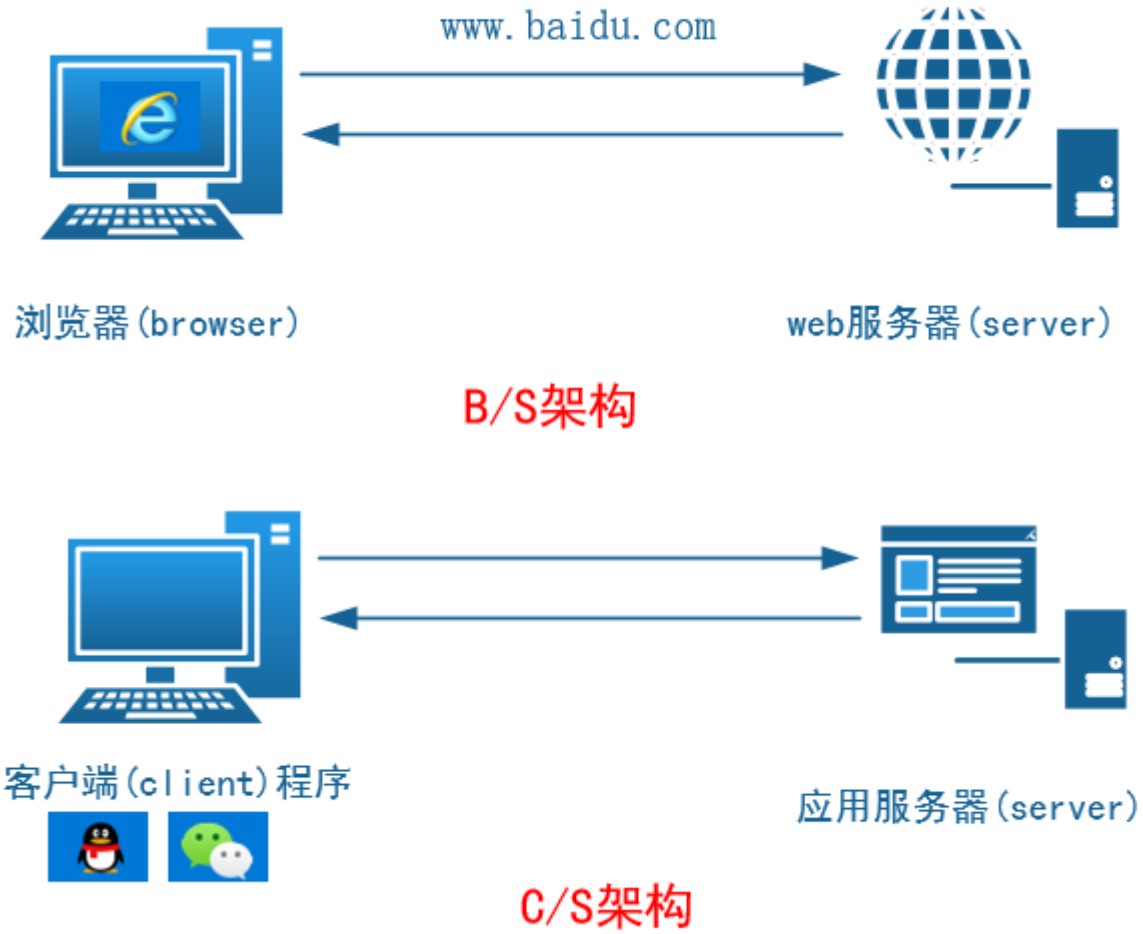

2、windows下文件传输工具

二、远程连接管理服务SSH

1、了解服务端和客户端

服务端：提供服务

客户端：享受服务



思考:

我们通过网络是如何找到我们想要访问的服务的?

IP(提供服务的服务器)+Port(找到相应的服务)

2、了解端口号的设定

说明:端口号只有整数，范围是从0 到65535

- 1～255：**一般是知名端口号**，如:ftp 21号、web 80、ssh 22、telnet 23号等
- 256～1023：通常都是由Unix系统占用来提供特定的服务
- 1024~5000：客户端的临时端口，随机产生
- 大于5000：为互联网上的其他服务预留

3、了解ssh服务的作用

用于Linux下远程连接管理服务器的安全协议。

功能：远程管理Linux服务器

- SSH服务默认端口22
- SSH服务是C/S架构

- 服务器端（开启ssh服务）：安装软件（openssh-server）
- 客户端（专门的客户端工具）：
  - windows：SecureCRT、MobaXterm
  - linux：openssh-clients ssh/scp

查看默认是否开机自启动：

```
[root@rhel8 ~]# systemctl list-unit-files |grep sshd.service
anaconda-sshd.service                         static
sshd.service                                  enabled
```

4、SSH服务的重启/停止

```
[root@rhel8 ~]# systemctl status sshd.service
[root@rhel8 ~]# systemctl stop sshd.service
[root@rhel8 ~]# systemctl start sshd.service
[root@rhel8 ~]# systemctl restart sshd.service
[root@rhel8 ~]# systemctl disable sshd.service
[root@rhel8 ~]# systemctl enable sshd.service
```

5、修改ssh服务的默认端口

㈠ 查看ssh服务端口

```
netstat命令：用于查看网络连接状态
-n:不显示名称
-l:查看监听状态
-t:TCP协议
-p:查看程序名字

[root@rhel8 ~]# netstat -nltp|grep :22
[root@rhel8 ~]# lsof -i :22
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    1017 root    5u  IPv4  32009      0t0  TCP *:ssh (LISTEN)
sshd    1017 root    8u  IPv6  32011      0t0  TCP *:ssh (LISTEN)
```

㈡ 修改ssh服务默认端口

① 修改配置文件

```
[root@rhel8 ~]# vim /etc/ssh/sshd_config
#在文件中增加以下行即可
Port 10022
```

② 重启服务

```
[root@rhel8 ~]# systemctl restart sshd.service
```

③ 客户端测试验证

```
```

㈢ 搭建服务总结

1. 关闭防火墙和selinux（实验）
2. 配置软件仓库
3. 软件三步曲
    - 安装相应软件（程序）
    - 确认软件是否成功安装
    - 查看软件的文件列表（配置文件、程序本身、man文档手册等）
4. 了解配置文件（man 5 xxx.conf）
5. 根据需求通过修改配置文件完成服务的搭建
6. 启动服务，开机自启动
7. 测试验证

6、基于SSH服务的命令

㈠ Linux下客户端工具ssh

```
Client：安装客户端软件，openssh-clients
功能1：客户端远程连接登录Linux服务器（ssh服务）端
ssh [选项] 用户名@远程IP
-l:指定访问用户
-p:指定端口号

ssh -p 10022 -l user01 192.168.159.100
ssh -p 10022 user01@192.168.159.100

注意：用户名一定是远程服务器端的用户，而不是客户端！

功能2：客户端远程访问Linux服务器执行相应的命令（未登录）
ssh [选项] 远程IP 执行相应的命令

ssh -p10022 user01@192.168.159.100 hostname
```

㈡ Linux下远程拷贝命令scp

```
用法1：将本地文件远程拷贝到远端（push）
scp [选项] 本地文件   远程服务器IP:/路径


用法2：将远程服务器上的文件拷贝到本地（pull）
scp [选项]  远程服务器IP:/文件  本地路径

注意：本地存放文件路径是否对当前拷贝文件的用户可以写
```

三、进程的检测与控制

1、认识管道

㈠ 什么是管道

管道，指在类UNIX系统中，进程之间通讯的一种方式或机制。

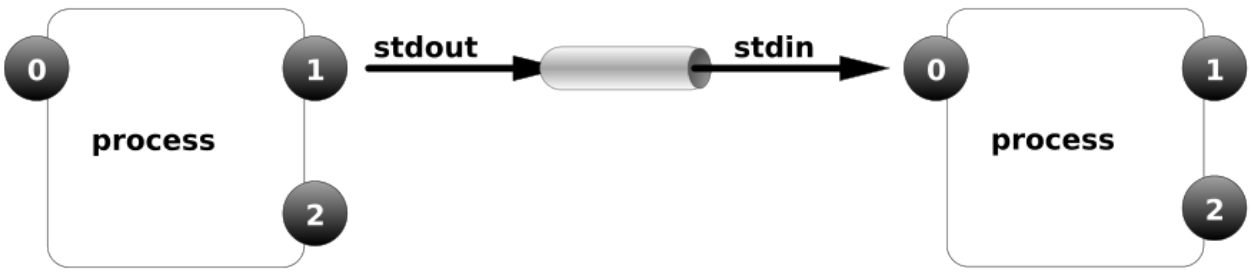
管道，也指一种特殊的文件，叫管道文件。

㈡ 管道的分类

① 匿名管道

匿名管道，顾名思义，就是没有名字的管道，常常用于父子关系的进程之间通讯一种方式。

匿名管道，在bash中，用符号"|"来表示。在同一个终端通讯。




```
标准输出：1        正确结果
标准错误：2        错误结果
标准输入：0        键盘输入

[root@localhost ~]# rpm -aq|grep vsftpd
vsftpd-2.2.2-24.el6.x86_64

[root@localhost ~]# cat /etc/passwd|less
```

匿名管道作用：将上一个命令所执行的结果（标准输出）作为下一条命令的标准输入

② 命名管道

- 命名管道，顾名思义，就是有名字的管道，既可以用于任何进程之间通讯。
- 命名管道，可以使用mkfifo命令创建。

```
[root@localhost ~]# mkfifo /tmp/p_file                    //创建一个命名管道文件
[root@localhost ~]# file /tmp/p_file                      //判断该文件的类型
/tmp/p_file: fifo (named pipe)
[root@localhost ~]# ll /tmp/p_file
prw-rw-r--. 1 root root 0 Mar 10 15:28 /tmp/p_file

[root@localhost ~]# tty
/dev/pts/1
[root@localhost ~]# echo "hello world" > /tmp/p_file

[root@localhost ~]# tty
/dev/pts/2
[root@localhost ~]# cat /tmp/p_file
hello world
[root@localhost ~]#
```

㈢ 引申xargs命令

场景：找出某些文件将其删除或者找出某个进程将它结束，如何实现?

```
需求：
在/tmp/dir1目录里有一个目录aaa和5个文件file1~file5，现需要删除该目录下的file1~file5

以下命令不能删除：
find /tmp/dir1 -name file* |rm -rf
但是，管道|后面加上xargs就可以删除：
find /tmp/dir1 -name file* |xargs rm -rf

以下命令可以成功删除：
find /tmp/dir1 -type f -exec rm -f {} \;
find /tmp/dir1 -type f -delete
```

xargs: 将上一条命令所执行的结果作为下一条命令的参数

```
命令 [可选项] 参数
[root@localhost ~]# ls -l /root

命令：整条shell命令的主体
选项：会影响或微调命令的行为
参数：命令作用的对象


举例说明：
cat -n  filename
命令  选项   参数

[root@localhost ~]# echo --help|cat
--help
[root@localhost ~]# echo --help|xargs cat

cat --help

[root@localhost tmp]# pwd
/tmp
[root@localhost tmp]# cat 1.sh
/root
[root@localhost tmp]# cat 1.sh|ls
1.sh
[root@localhost tmp]# cat 1.sh|xargs ls
aaa              Desktop    Downloads    install.log.syslog  Pictures  Templates
anaconda-ks.cfg  Documents  install.log  Music               Public    Videos
```

xargs其他选项(了解)：

```
常见选项
-n：指定单行显示的参数个数
-d：定义分割符，默认是以空格和换行符
[root@localhost ~]# cat 1.txt
a b c d
10.1.1.254
A       B       C
```

```
[root@localhost ~]# cat 1.txt |xargs -n 3
a b c
d 10.1.1.254 A
B C
[root@localhost ~]# cat 1.txt |xargs -n 4
a b c d
10.1.1.254 A B C
[root@localhost ~]# cat 1.txt |xargs -d'\t' -n 3
a b c d
10.1.1.254
A B C

[root@localhost ~]# cat 1.txt |xargs -d'.' -n 3
a b c d
10 1 1
254
A       B       C
```

2、进程概述

㈠ 什么是进程

进程，由程序产生，是正在运行的程序，或者说是已启动的可执行程序的运行实例。

进程，具有自己的生命周期和各种不同的状态。

引申：什么是线程?

线程，也被称作轻量级进程，线程是进程的执行单元，一个进程可以有多个线程。线程不拥有资源，它与父进程的其它线程共享该进程所拥有的资源。线程的执行是抢占式的。

张三（砌墙的）：a b c

李四（贴瓷砖）

王五（装水电）

㈡ 进程有什么特点

- 独立性

  进程是系统中独立存在的实体，它可以拥有自己的独立资源，每一个进程都有自己的私有地址空间；

  在没有经过进程本身允许的情况下，一个用户进程不可以直接访问其他进程的地址空间。

- 动态性

  进程与程序的区别在于，**程序**只是一个静态的指令集合，而进程是一个正在系统中活动的指令集合；

  进程具有自己的生命周期和各种不同的状态。

- 并发性

  多个进程可以在单个处理器上并发执行，多个进程之间不会互相影响。

程序和进程有什么区别?

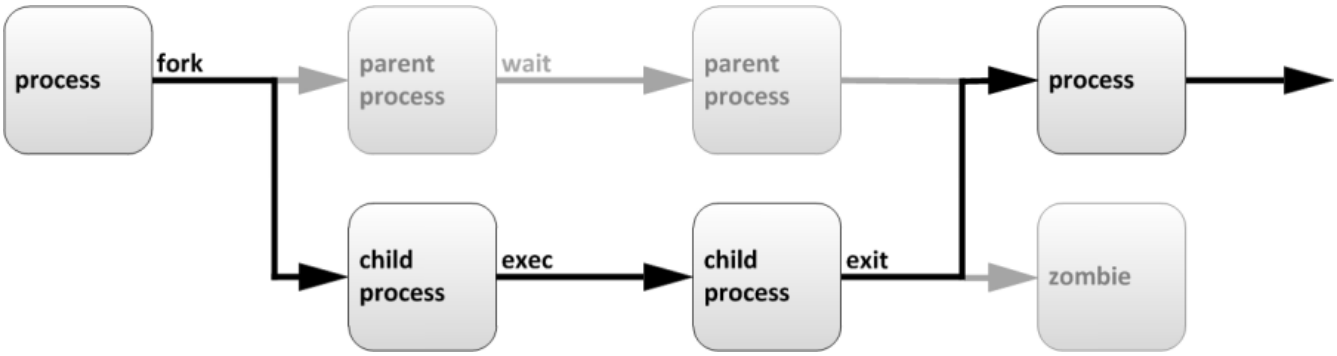
- 程序。二进制的文件，静态的。如:/usr/sbin/vsftpd，/usr/sbin/sshd

- 进程。程序的运行过程，动态的，有生命周期及运行状态的。

㈢ 进程生命周期

父进程,复制自己的地址空间（fork）创建一个新的（子）进程结构。每个新进程分配一个唯一的进程 ID （PID），满足跟踪安全性之需。PID 和 父进程 ID (PPID）是子进程环境的元素，任何进程都可以创建子进程，所有进程都是第一个系统进程的后代：Centos5/6: init RedHat7/8: systemd

子进程,继承父进程的安全性身份、过去和当前的文件描述符、端口和资源特权、环境变量，以及程序代码。随后，子进程exec 自己的程序代码。通常，父进程在子进程运行期间处于睡眠（sleeping）状态。当子进程完成时发出（exit）信号请求，在退出时， 子进程会关闭或丢弃了其资源环境，剩余的部分称之为僵停（僵尸Zombie）。父进程在子进程退出时收到信号而被唤醒，清理剩余的结构，然后继续执行其自己的程序代码。



3、进程信息查看

㈠ 静态查看ps命令

```
File  Edit  View  Search  Terminal  Help
[root@localhost Desktop]# ps -ef|head
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 14:46 ?        00:00:02 /sbin/init
root         2     0  0 14:46 ?        00:00:00 [kthreadd]
root         3     2  0 14:46 ?        00:00:00 [migration/0]
root         4     2  0 14:46 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 14:46 ?        00:00:00 [stopper/0]
root         6     2  0 14:46 ?        00:00:00 [watchdog/0]
root         7     2  0 14:46 ?        00:00:16 [events/0]
root         8     2  0 14:46 ?        00:00:00 [events/0]
root         9     2  0 14:46 ?        00:00:00 [events_long/0]
[root@localhost Desktop]# ps -eF|head
UID         PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
root          1     0  0  4838  1560   0 14:46 ?        00:00:02 /sbin/init
root          2     0  0     0     0   0 14:46 ?        00:00:00 [kthreadd]
root          3     2  0     0     0   0 14:46 ?        00:00:00 [migration/0]
root          4     2  0     0     0   0 14:46 ?        00:00:00 [ksoftirqd/0]
root          5     2  0     0     0   0 14:46 ?        00:00:00 [stopper/0]
root          6     2  0     0     0   0 14:46 ?        00:00:00 [watchdog/0]
root          7     2  0     0     0   0 14:46 ?        00:00:16 [events/0]
root          8     2  0     0     0   0 14:46 ?        00:00:00 [events/0]
root          9     2  0     0     0   0 14:46 ?        00:00:00 [events_long/0]
[root@localhost Desktop]#
```

```
File  Edit  View  Search  Terminal  Help
[root@localhost Desktop]# ps aux|head
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.1  19352  1560 ?        Ss   14:46   0:02 /sbin/init
root          2  0.0  0.0      0     0 ?        S    14:46   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        S    14:46   0:00 [migration/0]
root          4  0.0  0.0      0     0 ?        S    14:46   0:00 [ksoftirqd/0]
root          5  0.0  0.0      0     0 ?        S    14:46   0:00 [stopper/0]
root          6  0.0  0.0      0     0 ?        S    14:46   0:00 [watchdog/0]
root          7  0.2  0.0      0     0 ?        S    14:46   0:16 [events/0]
root          8  0.0  0.0      0     0 ?        S    14:46   0:00 [events/0]
root          9  0.0  0.0      0     0 ?        S    14:46   0:00 [events_long/0]
[root@localhost Desktop]# ps auxf|head
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          2  0.0  0.0      0     0 ?        S    14:46   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        S    14:46   0:00  \_ [migration/0]
root          4  0.0  0.0      0     0 ?        S    14:46   0:00  \_ [ksoftirqd/0]
root          5  0.0  0.0      0     0 ?        S    14:46   0:00  \_ [stopper/0]
root          6  0.0  0.0      0     0 ?        S    14:46   0:00  \_ [watchdog/0]
root          7  0.2  0.0      0     0 ?        S    14:46   0:16  \_ [events/0]
root          8  0.0  0.0      0     0 ?        S    14:46   0:00  \_ [events/0]
root          9  0.0  0.0      0     0 ?        S    14:46   0:00  \_ [events_long/0]
root         10  0.0  0.0      0     0 ?        S    14:46   0:00  \_ [events_power_ef]
[root@localhost Desktop]#
```

父子关系

- 常见组合

```
ps -ef
ps -eF
ps -ely
ps aux
ps auxf
```

a	显示当前终端下的所有进程，包括其他用户的进程
u	显示进程拥有者、状态、资源占用等的详细信息（注意有“-”和无“-”的区别）
x	显示没有控制终端的进程。通常与a这个参数一起使用，可列出较完整信息
o	自定义打印内容
-e	显示所有进程。
-f	完整输出显示进程之间的父子关系
-l	较长、较详细的将该进程的信息列出

- 进程信息解释说明

```
[root@rhel6 ~]# ps aux|head
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  19356  1432 ?        Ss   19:41   0:03 /sbin/init

[root@rhel8 ~]# ps aux|head
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.5 254968  9660 ?        Ss   08:39   0:10 /usr/lib/systemd/systemd -
-switched-root --system --deserialize 18

USER:           运行进程的用户
PID:            进程ID
%CPU:           CPU占用率
%MEM:           内存占用率
VSZ:            占用虚拟内存
```

```
RSS:            占用实际内存,驻留内存
TTY:            进程运行的终端
STAT:           进程状态,man ps获取帮助(/STATE)
      R         运行
      S         可中断睡眠 Sleep
      D 不可中断睡眠
      T         停止的进程
      Z         僵尸进程

     Ss         s进程的领导者，父进程
     S<                 <优先级较高的进程
     SN         N优先级较低的进程
     R+         +表示是前台的进程组
     Sl         以线程的方式运行

START           进程的启动时间
TIME            进程占用CPU的总时间
COMMAND 进程文件，进程名

其他命令查看进行信息
pidof           查看指定进程的PID
pstree  查看进程树
```

㈡ 动态查看top命令

第一部分：统计信息

```
[root@localhost ~]# top
top - 10:55:44 up  3:19,  2 users,  load average: 1.03, 0.50, 2.10
Tasks: 112 total,   1 running, 111 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:   1004412k total,   331496k used,   672916k free,    58364k buffers
Swap:  2031608k total,        0k used,  2031608k free,   107896k cached
```

关注load average:系统1分钟、5分钟、15分钟内的平均负载，判断一个系统负载是否偏高需要计算单核CPU的平均负载，如下图，一般以1以内为比较合适的值。偏高说明有比较多的进程在等待使用CPU资源。

```
# From /proc/cpuinfo, system has four logical CPUs, so divide by 4:
#                             load average: 2.92, 4.48, 5.20
#          divide by number of logical CPUs:    4     4     4
#                                             ----  ----  ----
#                      per-CPU load average: 0.73  1.12  1.30
#
# This system's load average appears to be decreasing.
# With a load average of 2.92 on four CPUs, all CPUs were in use ~73% of the time.
# During the last 5 minutes, the system was overloaded by ~12%.
# During the last 15 minutes, the system was overloaded by ~30%.
```

计算方法：

平均负载 / 逻辑cpu数量

```
物理CPU(N路)：主板上CPU插槽的个数
CPU核数：一块CPU上面能处理数据的芯片组的数量
逻辑CPU：一般情况，一颗cpu可以有多核，加上intel的超线程技术(HT)，可以在逻辑上再分一倍数量的cpu core出来；
逻辑CPU数量=物理cpu数量 x cpu核数。如果支持HT,还要更多。

查看物理CPU的个数
# cat /proc/cpuinfo |grep "physical id"|sort |uniq|wc -l
查看逻辑CPU的个数
# cat /proc/cpuinfo |grep "processor"|wc -l
查看CPU是几核
# cat /proc/cpuinfo |grep "cores"|uniq
```

第三行：当前的CPU运行情况

```
us              用户进程占用CPU的比率
sy              内核、内核进程占用CPU的比率；
ni              如果一些用户进程修改过优先级，这里显示这些进程占用CPU时间的比率；
id              CPU空闲比率，如果系统缓慢而这个值很高，说明系统慢的原因不是CPU负载高；
wa              CPU等待执行I/O操作的时间比率，该指标可以用来排查磁盘I/O的问题，通常结合wa和id判断
hi              CPU处理硬件中断所占时间的比率；
si              CPU处理软件中断所占时间的比率；
st              其他任务所占CPU时间的比率；

说明：
1. 用户进程占比高，wa低，说明系统缓慢的原因在于进程占用大量CPU，通常还会伴有教低的id，说明CPU空闲时间很少；
2. wa低，id高，可以排除CPU资源瓶颈的可能。
3. wa高，说明I/O占用了大量的CPU时间，需要检查交换空间的使用；如果内存充足，但wa很高，说明需要检查哪个进程占用了大量的I/O资源。
```

第二部分：进程信息

```
top - 17:40:03 up  2:53,  4 users,  load average: 0.00, 0.00, 0.00
Tasks: 163 total,   1 running, 162 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.0%us,  2.3%sy,  0.0%ni, 96.0%id,  0.3%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:   1004112k total,   914024k used,    90088k free,    96544k buffers
Swap:  2031612k total,        8k used,  2031604k free,   401456k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
16335 root      20   0 15036 1272  944 R  1.0  0.1   0:00.08 top
13408 root      20   0  173m 7868 4468 S  0.3  0.8   0:33.50 vmtoolsd
13586 root      20   0  197m 4932 4048 S  0.3  0.5   0:11.18 ManagementAgent
14993 root      20   0  291m  17m  14m S  0.3  1.8   0:36.42 vmtoolsd
    1 root      20   0 19352 1560 1236 S  0.0  0.2   0:02.72 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.48 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.23 watchdog/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:21.44 events/0
```

- **top命令常用按键命令**
  - 在top的执行过程中，还可以使用以下的按键命令：

```
h|?      帮助
```

```
M       按内存的使用排序
P       按CPU使用排序
T       按该进程使用的CPU时间累积排序
k       给某个PID一个信号（signal），默认值是信号15
1       显示所有CPU的负载
s       改变两次刷新之间的时间。默认是5秒
q       退出程序

N       以PID的大小排序
R       对排序进行反转
f       自定义显示字段
r       重新安排一个进程的优先级别
```

- **top命令常用的选项**

```
-d              后面可以接秒数，指定每两次屏幕信息刷新之间的时间间隔；
-p              指定某个进程来进行监控；
-u              指定进程的拥有者
-b -n   以批处理方式执行top命令。通常使用数据流重定向，将处理结果输出为文件；

[root@MissHou ~]# top
[root@MissHou ~]# top -d 1
[root@MissHou ~]# top -d 1 -p 10126                                     查看指定进程的动态信息
[root@MissHou ~]# top -d 1 -u apache                                    查看指定用户的进程
[root@MissHou ~]# top -d 1 -b -n 2 > top.txt        将2次top信息写入到文件
```

4、进程控制

㈠ 进程的优先级控制

① 调整正在运行进程的优先级(renice)

1）使用top按"r"来调整

```
改变NICE--->PR
优先级的范围：
-20--19  数字越低，优先级越高，系统会按照更多的cpu时间给该进程
```

2）命令行使用renice调整

```
sleep命令没有实际意义，延迟(睡觉)5000秒
[root@localhost ~]# sleep 5000 &
[1] 2544

sleep程序已经运行，通过renice命令调整优先级
[root@localhost ~]# renice -20 2544
2544: old priority 0, new priority -20
```

② 程序运行时指定优先级(nice)

```
启动进程时，通常会继承父进程的 nice级别，默认为0。
# nice -n -5 sleep 6000 &
# ps axo command,pid,nice |grep sleep
```

㈡ 进程的运行状态控制

① 如何控制进程的状态?

用户通过给进程**发送信号**来控制进程的状态。

② 常见的信号有哪些?

信号编号	信号名	解释说明
1	SIGHUP	默认终止控制终端进程(可用来重新加载配置文件，平滑重启)
2	SIGINT	键盘中断(ctrl+c)
3	SIGQUIT	键盘退出(ctrl+\)，一般指程序异常产生core文件
9	SIGKILL	强制终止
15	SIGTERM	正常结束，默认信号
18	SIGCONT	继续
19	SIGSTOP	停止
20	SIGTSTP	暂停(ctrl+z)

③ 如何给进程发送信号?

```
kill    [信号]   进程PID
killall
pkill

给进程号为15621的进程发送默认信号(-15可以省略)
kill -15 15621
给stu1用户的所有进程发送9号信号（结束stu1的所有进程），根据用户结束进程
pkill -9 -u stu1
给进程名为vsftpd的进程发送9号信号(根据进程名来结束进程)
pkill -9 vsftpd
killall -15 vsftpd
```

㈢ 进程其他控制命令

```
# command &                放到后台运行
# jobs           查看当前终端后台的进程
# fg             把后台进程放到前台来运行
```

```
# bg              把后台暂停的进程放到后台运行

# fg %1 将作业1调回到前台
# bg %2           把后台编号为2的进程恢复运行状态

# kill -20 %3            给job编号为3的进程发送信号
# firefox www.baidu.com &        打开浏览器放到后台运行

kill 信号         进程编号 或者 job编号
pkill   信号      进程名字
killall 信号   进程名字

fg  %2  把后台job编号为2的进程放到前台来运行
bg  %3  把后台job编号为3的进程放到后台继续运行
```

总结

1. 如何查看进程相关信息

   1）静态查看 ps

   ```
   ps
   ps -ef|grep httpd
   ps -eF
   ps -ely

   ps aux
   ps auxf|grep httpd
   ps axo user,pid,nice
   ```

   2）动态查看 top

   ```
   # top
   P
   M
   T
   r
   k
   ```

2. 如何控制进程的状态或优先级

   1）设置进程优先级（-20-19）

   - renice调整正在运行进程优先级
   - nice指定程序运行时的优先级

   2）发送信号控制进程的状态

   - 常见信号（kill -l） 9 15 18 19 20 1
   - 使用哪些命令发送信号（kill/pkill/killall）
   - 进程前台和后台运行（jobs/fg/bg）

四、时间同步服务NTP

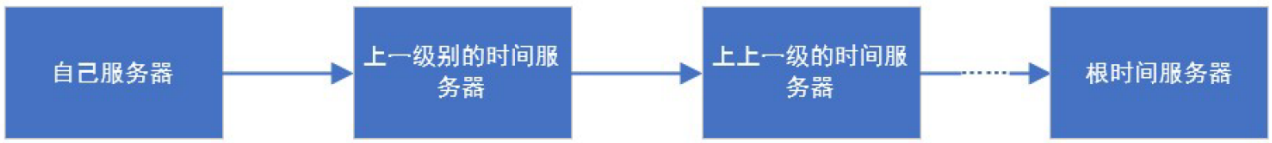
1、了解时间同步的重要性

由于IT系统中，准确的计时非常重要，有很多种原因需要准确计时，如

- 在网络传输中，数据包和日志需要准确的时间戳
- 各种应用程序中，如订单信息，交易信息等都需要准确的时间戳

在Linux系统中， Network Time Protocol (NTP)，NTP协议由在用户空间中运行的守护程序实现，即ntp和chrony。

2、了解时间同步服务器



查看可以提供时间同步的服务器：

http://www.ntp.org.cn/pool.php

3、如何同步自己系统时间

- 修改配置文件指定ntp服务器同步

```
# vim /etc/chrony.conf
pool cn.ntp.org.cn iburst
或
server 203.107.6.88 iburst

重启服务
```

- 临时指定ntp服务器同步

```
# chronyd -q "server cn.ntp.org.cn iburst"
```

4、搭建时间同步服务

㈠ 了解RHEL7和RHEL8区别

- 在RHEL7中，用户可以在**ntp**和**chrony**之间进行选择，以确保准确的计时。
- 在RHEL8中，不再支持**ntp**,使用**chronyd**守护进程来实现NTP,默认已启用,从chrony软件包获得。
- chrony和ntp之间的差异
  - ntpd服务器在控制客户端访问方式上默认响应来自任何地址的请求，相对不安全

    - chronyd默认不允许访问，要想提供时间同步服务端需要在allow中指定
    - `ntpd`和`chronyd`在系统时钟校正方面的默认行为也不同。
    - 客户端同步方式有差异
- 还有很多不同可以参考官方网站：https://chrony.tuxfamily.org/comparison.html

总结：

chrony在各种条件下表现良好，包括间歇性网络连接，高度拥挤的网络，温度变化（普通计算机时钟对温度敏感），以及不能连续运行或在虚拟机上运行的系统。

㈡ RHEL8配置时间同步服务

1. 关闭防火墙和selinux（实验）
2. 配置软件仓库
3. 软件三部曲
    - 安装软件
    - 确认软件是否成功安装
    - 查看软件的文件列表（配置文件、程序本身、man手册）
4. 了解配置文件（man 5 xxx.conf）
5. 根据需求通过修改配置文件来完成服务搭建
6. 启动服务，开机自启动
7. 测试验证

- 时间同步服务器端

```
# vim /etc/chrony.conf
#自己本身向谁来同步时间
pool cn.ntp.org.cn iburst
#允许哪些客户端来同步
allow 192.168.159.0/24


重启服务
```

- 客户端

```
[root@client ~]# chronyc sources
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? 192.168.159.100               3   6    37     5  +10766h[+10766h] +/- 3872us

[root@client ~]# chronyc makestep
200 OK
```

五、计划任务服务crond

1、了解计划任务的作用

作用：解放我们的双手，解放我们的时间

- 计划任务，让系统在将来的指定时间点执行某些任务（程序）
- 计划任务，可以周期性执行也可以仅仅执行一次
- Linux系统中的计划任务at和crond服务是操作系统内置的2个服务，默认情况是安装好的。

2、编写简单的周期性计划任务

㈠ 了解系统计划任务相关文件

```
/etc/cron.d/
/etc/cron.d/0hourly       系统每小时第一分钟需要执行的任务
/etc/cron.deny        用户拒绝列表（在该文件中的用户不能使用cron服务）
/etc/crontab          该文件的作用相当于/etc/cron.d/下面的某一个文件，可以定义系统计划任务
/etc/cron.monthly/    存放系统每个月需要执行的脚本
/etc/cron.weekly/     存放系统每周需要执行的脚本
/etc/cron.daily/      存放系统每天需要执行的脚本
/etc/cron.hourly/     存放系统每小时需要执行的脚本
/var/spool/cron       这个目录用来存放各个用户自己设定的定时任务，普通用户没有权限直接访问
```

㈡ 计划任务的周期编写

```
[root@rhel8 ~]# cat /etc/crontab
SHELL=/bin/bash
默认的shell，告诉系统使用哪个shell
PATH=/sbin:/bin:/usr/sbin:/usr/bin
定义命令的路径
MAILTO=root
结果以邮件的形式发送给root（不管是正确还是错误结果），如果MAILTO=""代表不会发邮件给任何人。
# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed

前5个字段分别表示：
    分钟：0-59
    小时：0-23
    日期：1-31
    月份：1-12
    星期：0-7（0表示周日或者7表示周日）

5 * * * *
```

```
05 * * * *
20 08 * * *
00 06 * 12 *

03 01 * * 7

还可以用一些特殊符号:
,         表示分割，间隔
-         表示一个段
/n        表示每个n的单位执行一次

*/5 * * * *

03 03,06 1-15 * *

礼拜1和礼拜3每隔2个小时去执行一个任务
00 */2 * * 1,3

每个月的1号-15号，和25号的早上8:00到晚上6:00每隔2小时执行一个任务
00 08-18/2 1-15,25 * *


以下4行是rhel5里的配置；rhel6看不到，但是也生效
01 * * * *      root run-parts /etc/cron.hourly/
02 04 * * *     root run-parts /etc/cron.daily/
22 04 * * 7     root run-parts /etc/cron.weekly/
42 04 1 * *     root run-parts /etc/cron.monthly/


注意:
run-parts:
crond用这个工具来执行某个目录下所有的可执行脚本，定时任务中的每小时/每天/每周/每月任务就是通过这个工具来触发的.
```

㈢ 创建、查看、删除计划任务

1. 修改配置文件 /etc/crontab 管理员root----》指定执行任务用户
2. 让用户自己使用命令创建 crontab

① 用户编辑自己的定时任务

```
crontab -e              编辑当前用户自己的定时任务（使用环境变量EDITOR指定的默认编辑器）
crontab -l              列出当前用户自己所有的定时任务
crontab -r              删除当前用户自己所有的定时任务
```

② 管理员编写其他用户的定时任务

```
crontab -e -u redhat            编辑指定用户的定时任务（使用环境变量EDITOR指定的默认编辑器）
crontab -l -u redhat            列出指定用户所有的定时任务
crontab -r -u redhat            删除指定用户所有的定时任务
```

六、课后实战案例

每天备份etc目录，要求如下：

1. 每天4:00备份/etc目录到/data/bak目录里
2. 将备份命令写在脚本中，如/scripts/back.sh，加执行权限
3. 每天备份的文件名包含当天的日期，如2019-08-15_etc.tar.gz
4. 计划任务执行时，屏幕不产生任何输出
5. 只保留最近5天的备份

七、今日目标打卡

- [x] 了解管道符号"|"的作用
- [x] 了解程序和进程之间的关系
- [x] 了解进程的特点
- [x] 能够使用ps和top命令查看进程信息
- [x] 能够使用kill/pkill等命令给进程发送信号
- [x] 能够设置进程的优先级
- [x] 能够使用客户端工具远程连接Linux服务器
- [x] 能够编写计划任务