# Linux

Base

uname –a

locate cmd

whereis

## 6.1 Base

### 系统配置

 版本：cat /proc/version

uname

-a print all info.

–-help

### 系统目录

/bin 二进制可执行命令。一般用户可用。ls,cp

/sbin 系统管理命令。shutdown,reboot。

/etc 系统管理和配置文件

|--/etc/rc.d 启动的配置文件和脚本

|--/etc/init.d

/home 用户主目录的基点，比如我的主目录就是/home/nadong，可用~表示

/usr 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。其中包含：

|--/usr/local 本地安装的程序，配置和其他。系统升级无需重新安装。

|--/usr/local/bin 本地增加的命令

|--/usr/local/lib 本地增加的库 比如java

|--/usr/local/etc配置文件

|--/usr/local/share其它的资源文件放在。

|--/usr/bin 预装，随系统升级改变。 和/usr/local/bin优先级看path配置。

|--/usr/sbin 非必须的系统管理命令。

|--/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/x11r6 存放x window的目录

/root 系统管理员的主目录（特权阶级）

/lib 标准程序设计库，又叫动态链接共享库，作用类似windows里的.dll文件

/dev 设备特殊文件

/tmp 公用的临时文件存储点

/var 某些大文件的溢出区，比方说各种服务的日志文件

/mnt 系统提供这个目录是让用户临时挂载其他的文件系统。

/proc 虚拟的目录，是系统内存的映射。可直接访问这个目录来获取系统信息。

/lost+found 平时空的，系统非正常关机而留下“无家可归”的文件。

Warning

usr 指 Unix System Resource，而不是User

path配置，前面覆盖后面的。

### 软件安装

#### yum

yum list 所有已安装和可安裝的软件清单

yum search java 查找软件包

yum list installed

yum install xxx

设置代理

/etc/yum.conf

proxy=http://cn-proxy.jp.oracle.com:80/

#### make

configure

生成Makefile:GUN规范

./configure –prefix=/usr 将该软件安装在 /usr 下面，执行文件就会安装在 /usr/bin 资源文件就会安装在 /usr/share

make 编译

默认安装

使用make编译安装软件时，若./configure不指定prefix，则：

可执行文件默认放在/usr/local/bin（/usr/local/libexec中也可能有一些），

库文件默认放在/usr/local/lib，

配置文件默认放在/usr/local/etc，

其它的资源文件放在/usr/local/share。

卸载

1.在原来的make目录下用一次make uninstall（前提是make文件指定过uninstall）

2.去上述目录里面把相关的文件一个个手工删掉。

-

配置文件

通常情况下，用户软件的配置文件放在自已的 home 目录下，比如 /home/riku/ 下，一般以 . 作为目录开始字符，比如 .gnome 。

而系统软件的配置文件一般放在 /etc/ 目录下。

### ssh远程登录

[Reference](https://blog.csdn.net/u011054333/article/details/52443061)

-v debug

-p port

ssh -v -p 8080 nadong@slc11fsp.us.oracle.com

TODO1:如何防止中间人攻击？

### 服务

服务：开机不用登录就可以执行的命令

#### Systemctl

sytemd进程的对应管理命令，CentOS7用systemctl替代原来的service,并兼容service（systemctl start xxx == service xxx start,调用/etc/init.d/下脚本）。

开机启动

1.在路径/usr/lib/systemd/system添加服务文件updateDNS.service

2.开机启动systemctl enable updateDNS.service

3.修改后重启 systemctl daemon-reload

**原理**

systemctl enable/ disable updateDNS

原理：systemctl enable updateDNS.service时，updateDNS.service的链接文件/etc/systemd/system/ multi-user.target.wants/路径下。

**启动模式**

修改文件xx.service

[Install]

WantedBy=服务组 Target

multi-user.target 这个组开机都启动。缺省模式。

graphical.target：图形用户状态。

服务配置

路径1：/usr/lib/systemd/system 开机不用登录

路径2：/usr/lib/systemd/user/

路径3：/etc/systemd/system

路径下有四种类型文件

xxx.mount 挂载点。

xxx.service 定义服务

xxx.target 基础组件，供.service调用

xxx.wants 服务组

xx.service

三部分： Unit、Service、Install

[Unit]

Description=update dns server at home machine

[Service]

ExecStart=/usr/local/myservice/updateDNS/start.sh

[Install]

运行模式。

#### Service

执行/etc/init.d/目录下脚本

增加服务方法

1.脚本，普通shell，但要实现service的start等， 参考/etc/init.d/updateDNS

2.updateDNS脚本放/etc/init.d/目录

开机启动

chkconfig [--level 3] httpd on/off

或者chkconfig --add/del updateDNS。(原理：执行后/etc/rc2.d,/etc/rc3.d等添加K10/S10链接)

chkconfig --list

service updateDNS start 相对于 updateDNS start，不同在于脚本在init.d目录。

很多服务以d结尾，d代表 deamon 守护进程

init.d

/etc/init.d 是 /etc/rc.d/init.d 的软链接(soft link)。放init.d保证不同linux版本兼容性。

chkconfig

配置某服务开关机自动启动。默认启动2345四个级别。

执行级别

等级0表示：表示关机

等级1表示：单用户模式

等级2表示：无网络连接的多用户命令行模式

等级3表示：有网络连接的多用户命令行模式

等级4表示：不可用

等级5表示：带图形界面的多用户模式

等级6表示：重新启动

修改 chkconfig --level 35 updateDNS on

## 6.2 系统资源分析

### 3.1.1 常用CMD

#### top

–p pid 监控某个进程

-H show thread, top –Hp pid

4 users：当前登录user数

Tasks:进程数。-

Cpus(s)

us/sy — 用户/内核 占用CPU的百分比。

ni — 改变过优先级的进程占用CPU的百分比

id — 空闲CPU百分比

wa — IO等待占用CPU的百分比

hi — 硬中断（Hardware IRQ）占用CPU的百分比

si — 软中断（Software Interrupts）占用CPU的百分比

Title

PR 优先级

VIRT 进程使用的虚拟内存总量，单位kb。VIRT=SWAP+RES

RES 进程使用的、未被换出的物理内存大小，单位kb。RES=CODE+DATA

SHR 共享内存大小，单位kb

%CPU  CPU时间占用百分比

%MEM  进程使用的物理内存百分比

TIME+  进程使用的CPU时间总计，单位1/100秒  
CMD 进程名称（命令名/命令行）

**快捷键**

P:CPU排序。默认按cpu占用量排序。

M:内存排序

H: show thread

1:show cups 列出单个cpu的负荷情况

#### ps

查看线程。Process Status

a 显示现行终端机下的所有程序，包括其他用户的程序

f 用ASCII字符显示树状结构，表达程序间的相互关系

e 列出程序时，显示程序所使用的环境变量

-A 显示所有进程

-u 以用户为主的格式来显示程序状况

-x 显示所有程序，不以终端机来区分

常用

ef



USER 用户名

UID User ID

PID Process ID（进程），当然jps命令也可以查到进行ID

PPID parent pid

TIME 进程使用的总cpu时间

COMMAND 正在执行的命令行命令

aux



SID 会话ID（Session id）

%CPU **进程的cpu占用率，单个CPU，双核的最大值为200%**

%MEM 进程的内存占用率

VSZ 进程所使用的虚存的大小（Virtual Size）

RSS 进程使用的驻留集大小或者是实际内存的大小，Kbytes字节。

TTY 与进程关联的终端（tty）

STAT 进程的状态：进程状态使用字符表示的（STAT的状态码）

START 进程启动时间和日期

#### netstat

系统所有的网络套接字连接情况，包括 tcp, udp 以及 unix 套接字。

Active Internet connections，称为有源TCP连接

Active UNIX domain sockets，称为有源Unix域套接口(和网络套接字一样，但是只能用于本机通信，性能可以提高一倍)

Param

-a (all)显示所有选项，默认不显示LISTEN相关

-t (tcp)仅显示tcp相关选项

-n 拒绝显示别名(域名)，能显示数字的全部转化成数字(ip)。

-u (udp)仅显示udp相关选项  
-p 显示建立相关链接的程序名

-l 仅列出有在 Listen (监听) 的服務状态

-x**只列出所有监听 UNIX 端口**

提示：LISTEN和LISTENING的状态只有用-a或者-l才能看到

-p 显示建立相关链接的程序名  
-r 显示路由信息，路由表  
-e 显示扩展信息，例如uid等  
-s 按各个协议进行统计  
-c 每隔一个固定时间，执行该netstat命令

**常用**

1)netstat -at|grep slc01boa

2)统计80端口连接数

netstat -nat|grep -i "8080"|wc –l

3)Linux下查看tomcat连接数

netstat -na | grep ESTABLISHED | grep 8080 | wc -l

1. 统计httpd协议连接数  
   ps -ef|grep httpd|wc –l
2. 统计已连接上的，状态为“established

netstat -na|grep ESTABLISHED|wc -l

3)找出程序运行的端口

netstat -ap | grep java

**4)**[根据端口查找进程](http://www.cnblogs.com/paul8339/p/6638370.html)

netstat -apn | grep 8989

netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'

windows

netstat -ano -p tcp -n 1|find "8080"

-n 1间隔1s自动刷新



PID是进程

统计

netstat -an|find "ESTABLISHED" /c

title

local address本机的地址

foreign address 外部电脑电脑有联系的ip地址

#### free

[Reference](http://www.cnblogs.com/coldplayerest/archive/2010/02/20/1669949.html)



-/+ buffers/cache: 应用程序认为系统被用掉多少内存

设置缓存大小：

#### df

Disk File

-i 查看inodes

df –hl

df -h /home/nadong

#### du

Disk usage of each File

du –sh

分区

## 3.2 文件管理

### 文件属性

文件颜色

绿色文件---------- 可执行文件，可执行的程序

红色文件-----------压缩文件或者包文件

蓝色文件----------目录

白色文件----------一般性文件，如文本文件，配置文件，[源码](https://www.2cto.com/ym/)文件等

浅蓝色文件----------链接文件，主要是使用ln命令建立的文件



链接文件

分配不同权限

缩写路径

软链接 VS 硬链接

软链接和Windows的快捷方式类似

文件权限

Chmod

chmod u+w test

chmod ugo+r test;

### 文件搜索

which       查看可执行文件的位置   
whereis    查看文件的位置   
locate       配合数据库查看文件位置   
find          查询文件名称

find

find [dir] -name filename

1.在当前及子文件夹查找

find –name common.jar

2.在etc及子文件夹查找

find /etc -name httpd.conf

3.模糊查找

find –name ‘httpd\*'

Grep

-n ：输出行号

-r ：在某一目录下递归[所有子目录]查找某一字串:

grep -r ‘str\*' . (注意此处有点)

grep -r a /scratch

Whereis

whereis useradd

### 文件查看及编辑

more

Enter:一行

Space:翻页

less

less 的用法比起 more 更加的有弹性。在 more 的时候，我们并没有办法向前面翻， 只能往后面看

Space:翻页

G:倒序查看

vim

**全部删除：**按esc后，然后dG

lsof

## 3.3 权限

[Reference](https://www.cnblogs.com/cisum/p/8005641.html)

权限管理实际上都是对有关的系统文件进行修改，与用户和用户组相关的信息都存放在一些系统文件中，这些文件包括/etc/passwd, /etc/shadow, /etc/group等。

### 3.3.1 Summary

添加组 groupadd moon

添加用户 adduser moon1 -g moon

修改用户 usermod moon2 -g home

查看组成员

cat /etc/passwd|grep home

删除组成员

vi /etc/group

### 3.3.2 group

每个用户都属于某个用户组，用户组在创建用户时同时创建。一个组中可以有多个用户，一个用户也可以属于不同的组。

#### view

cat /etc/group

groupName:口令:GID:UserList

moon:x:1001:

#### groupadd

groupadd group1

**groups** 查看当前登录用户的组内成员

**groupdel**: groupdel groupName先删除group的用户，才能删除群组

**groupmod:** groupmod -n newName groupName (-n改组名 –g 改gid)

### 3.3.3 User

#### view

cat /etc/passwd

用户名:口令:用户标识号:组标识号:注释性描述:主目录:Shell

moon:x:1001:1001::/home/moon:/bin/bash

x： 口令。

主目录：登录后所处的目录。大多数系统，用户的主目录都在同一个特定的目录下（/home）

#### adduser

-d 指定用户主目录；-g update用户组；-G 附加组(不删除原来组)

-s Shell文件 指定用户的登录Shell。

-c comment

userdel： userdel -r moon （-r把用户的主目录一起删除）

usermod

实例

useradd -d /home/moon moon –m （-m选项，可以创建主目录）

useradd -s /sbin/nologin ipServer

adduser moon1 –g moon

#### nologin

使用nologin账号来运行程序

su -s /bin/bash -c "ls" nginx

方法2：sudo -u nginx command

#### 伪用户

psuedo users.用户在/etc/passwd文件中也占有一条记录，但是不能登录，因为它们的登录Shell为空。它们的存在主要是方便系统管理，满足相应的系统进程对文件属主的要求。

bin 拥有可执行的用户命令文件

sys 拥有系统文件

adm 拥有帐户文件

uucp UUCP使用

lp lp或lpd子系统使用

nobody NFS使用

### 3.3.4 passwd

-l禁用账号。

-u 口令解锁。

-d 使账号无口令。

-f 强迫下次登录时修改口令。

超级用户修改/su不需要输入用户密码



普通用户直接输passwd

2./etc/shadow

moon:$6$007qnwIe$kS:17795:0:99999:7:::

用户名:口令:last\_update:最小时间间隔:最大时间间隔:警告时间:不活动时间:失效时间:标志

passwd文件所有用户可读，如果密码放passwd文件，即便加密也很容易破解。真正的加密后的用户口令字存放到shadow文件中，只有超级用户可读。

其他

**chown**将指定文件的拥有者改为指定的用户或组

chown -R mysql: mysql ./

su:Su switch use /sudo输入当前用户的密码。

sudo1.需要root赋予sudo权限

## 5.5 安全

添加端口

iptables -I INPUT -p tcp --dport 7001 -j ACCEPT

iptables VS Firewalld

从Cent7以后，iptables服务的启动脚本已被忽略。firewalld自身并不具备防火墙的功能，而是和iptables一样需要通过内核的netfilter来实现。

firewalld使用XML进行配置。除非非常特殊的配置，而应该使用firewalld-cmd。

配置文件：

/usr/lib/firewalld    #保存默认配置，避免修改他们；

/etc/firewalld    #保存系统配置文件，这些文佳你将覆盖默认配置

## 3.4 Shell

解释器，看成一个线程

登录shell：是需要用户名、密码登录后才能进入的shell

非登录shell： 不需要输入用户名和密码即可打开的Shell，例如：直接命令“bash”就是打开一个新的非登录shell，在Gnome或KDE中打开一个“终端”（terminal）窗口程序也是一个非登录shell。

/bin/nologin l，

交互式：在终端上执行，shell等待你的输入

非交互式：以shell script方式执行。

交互Shell:

Shell是用户与Linux系统之间的接口。常用的有

Bourne Shell（/usr/bin/sh或/bin/sh）

Bourne Again Shell（/bin/bash）

C Shell（/usr/bin/csh）

K Shell（/usr/bin/ksh）

Shell for Root（/sbin/sh）

系统管理员可以根据系统情况和用户习惯为用户指定某个Shell。

用户的登录Shell也可以指定为某个特定的程序（此程序不是一个命令解释器）。

利用这一特点，我们可以限制用户只能运行指定的应用程序，在该应用程序运行结束后，用户就自动退出了系统。

登陆shell：户登陆时，输入用户名和密码后启动的shell；

### Shell Script

扩展名为 sh（sh代表shell）

1. #!/bin/bash
2. echo "Hello World !"

#! 是一个约定的标记，告诉系统这个脚本需要什么解释器来执行，即使用哪一种 Shell。

#### 运行

1./test.sh 脚本加#!/bin/bash

2.sh test.sh

#### 变量

$# 是传给脚本的参数个数

$0 是脚本本身的名字

$1 是传递给该shell脚本的第一个参数

$2 是传递给该shell脚本的第二个参数

$@ 是传给脚本的所有参数的列表

$\* 是以一个单字符串显示所有向脚本传递的参数，与位置变量不同，参数可超过9个

$$ 是脚本运行的当前进程ID号

$? 是显示最后命令的退出状态，0 = success, 其他 = error

${var}等价于$var（简写）

if/else

[Reference](http://www.cnblogs.com/kangyoung/p/3556173.html)

#### 比较

if ([ condition ] || [condition])

then

cmd

else

cmd

fi

if method()

method return 0=true ,1=false

#### 文件表达式

-e filename 如果 filename存在，则为真  
-d filename 如果 filename为目录，则为真   
-f filename 如果 filename为常规文件，则为真  
-L filename 如果 filename为符号链接，则为真  
-r filename 如果 filename可读，则为真   
-w filename 如果 filename可写，则为真   
-x filename 如果 filename可执行，则为真  
-s filename 如果文件长度不为0，则为真  
-h filename 如果文件是软链接，则为真

整数变量

eq:=

ne:!=

gt:>

ge:>=

lt: <

le <=

函数

LaunchJDeveloper(){

DisplayBanner

}

#### expr

手工命令行计数器



模式匹配