**Netstat -ntlp**

**nohup ./startup.sh &**

**gcc**

1. gcc工作流程

预处理--E

宏替换

头文件展开

注释去掉

xxx.c -> xxx.i

编译--S

xxx.i -> xxx.s

汇编文件

汇编--C

xxx.s -> xxx.o

二进制文件

链接

xxx.o -> xxx(可执行)

2.gcc常用参数

-v/--version 查看gcc的版本

-I 编译时指定头文件的路径

-C 将汇编文件生成二进制文件

-o 指定生成的文件的名字

-g gdb调试时需要加的参数，加这个参数编译成的文件会比不加的大一些

-D 在编译的时候指定一个宏

使用场景：在测试程序的时候

-Wall 添加警告信息

-On 优化代码，n是优化代码：1，2，3

# 库

## 库是什么

1. 二进制文件
2. 将源代码-》二进制格式的源代码
3. 加密

## 库制作出来之后，如何给用户使用

1. 头文件(不加密)
2. 制作出来的库(加密的)

# Linux常用命令：

## Vim

## 一般模式下

### ctrl+f ：向下移动一页

### 一般模式下 ctrl+b ：向上移动一页

### 0 ：移动到该行的最前面

### $ ：移动到该行的最后面

### G ：移动到最后一行

### gg ：移动到第一行

### n+enter ：向下移动n行

### /word ：向下寻找word

### ?word ：向上寻找word

### :n1,n2s/word1/word2/g 在n1到n2行之间把word1替换成word2

### :1.$s/word1/word2/g 在该行把word1替换成word2

### :1,$s/word1/word2/gc 第一行到最后一行把word1替换成word2

### x ：向后删除一个字符

### X ：向前删除一个字符

### dd ：删除光标所在行

### ndd ：删除光标所在的向下n行

### yy ：复制光标所在行

### nyy ：复制光标所在的向下n行

### p ：在光标下一行粘贴

### P：在光标后面粘贴

### u：类似于撤销

### Ctrl+r ：取消撤销

## 编辑模式下

### i：从光标处插入

### a：从光标后插入

### o：从下一列插入

### r：取代光标所在的字符

## 命令模式下：

### ：w

### ：w！

### ：q

### ：q！

### ：wq

### ZZ

区块操作

1. v：字符选择
2. V：列选择
3. Ctrl+v ：区块选择 长方形
4. y：复制
5. d：删除
6. p：粘贴

多文件编辑

Vim -o filename1 filename2 打开上下分屏

Vim -O filename1 filename2 打开左右分屏

:sp filename

1. ：n 编辑下一个文件
2. ：N编辑上一个文件
3. ：files 列出当前vim开启的所有文件

分区窗口

指令模式下输入 ：sp filename 打开上下两部分

Vsp 分成左右两部分 ctrl+w+w切换

1. ctrl+w+上箭头 切换到上一个窗口
2. Ctrl+w+下箭头 切换到下一个窗口
3. ：q all 退出所有
4. ：w all 保存所有
5. ：wq all 保存所有退出

指令操作

1. ctrl+u 从光标处向前删除指令
2. Ctrl+k 从光标处向后删除指令
3. Ctrl+a 让光标移动到最前面
4. Ctrl+e 让光标移动到最后面

变量

1. 打印变量 echo variable
2. 设置变量 variable=value
3. 取消变量 unset varibale
4. 显示shell中的环境变量 env
5. 观察所有的变量(包括环境变量和自定义变量) set
6. 自定义变量转成环境变量(子程序可以使用) export variable
7. read +pt 变量名 ：将输入的值赋值给变量名 p(后面跟提示) t(后面跟输入时间限制)
8. declare -aixr 变量名=xxx 宣告变量的类型 a(数组) i(整数) x(变成环境变量) r(只读 不 能unset和修改)
9. ulimit 限制用户资源使用
10. type -a 查询指令是不是内建指令 以及是哪一个文件下的指令

系统配置

1.Bash读取配置文件的顺序

/etc/profile --> ~/.bash\_profile或者profile --> ~/.bashrc

在profile里面重新设置PATH 在bashrc里面设置别名等

2.source(.) 配置文件名 ：不用重启即可加载

3.stty -a 查看终端机设置

3.ctrl+S 暂停屏幕的输出

4.ctrl+Q 恢复屏幕的输出

数据流重导向

输出：

1. > ：写入正确的信息
2. >> ：累加正确的信息
3. 2> ：写入错误的信息
4. 2>> ：写入错误的信息
5. >filename 2>&1 无论正确或者错误信息都输入到filename中

输入：

1. <
2. << ：后面跟字符串 表示结束的输入字符

多指令执行

1. 指令1 && 指令2 ：指令1执行成功才会执行指令2
2. 指令1 || 指令2 ：指令1执行失败才执行指令2
3. 三元运算符 command1 && command2 || command3 (每条指令执行成功后返回0，否则非 0)

管道(pipe)”|”

1. 管线命令仅会处理正确的输出信息
2. | 后面的命令称为管道命令(截取命令、排序命令)
3. | 后面不加ls cp mv 可加less more head tail
4. cut ：截取一行中的东西 以行为单位
5. cut -d ‘分隔字符’ -f fields(,fields) ：根据分割字符分割成几段 取出第fields段
6. cut -c x-y ：出去x到y之间的数据
7. grep ：取出符合条件的一行 -i(忽略大小写) -n(输出行号) -v(反向选择：选择不符合的)

Grep -inv string filename

1. sort ：排序(默认使用第一个字符排序)
2. uniq ：去除重复 -i(忽略大小写) -c(进行计数)
3. wc ：统计 -l(显示行数) -w(多少字) -m(多少字符)
4. tee(双向重导向) ：将数据同时显示到屏幕和文件中 -a（累加的方式）
5. tr ：删除或替换
6. tr -d string ：删除信息中string字符
7. tr -s ：取代掉重复的字符
8. split -bl file PREFIX ：将file劈成几个小文件 文件名根据PREFIX生成 -b(根据大小劈开 后跟文件大小) -l(根据行数劈开 后跟行数)
9. xargs -0epn command：产生某个指令的参数

如：find /xxx -perm /7000 | xargs ls -l 可以让非管线命令连接在|后面 非常好用

1. sed -n ‘动作’：对输出到屏幕的东西进行处理(crud)-----可重点掌握

-n ：只显示处理过的行数

动作说明 ：[n1[,n2]]function

Function 说明 ：a(增加) c(取代) d(删除) i(插入) p(打印) s(取代)

如：

nl /etc/passwd | sed ‘2,5d’ 将/etc/passwd的内容列出并打印行号，同时 将2~5行删除

nl /etc/passwd | sed ‘2d’ 删除第二行

Nl /etc/passwd | sed ‘3,$d’ 删除3到最后一行

Nl /etc/passwd | sed ‘2a drink tea’ 在第二行后加drink tea

Nl /etc/passwd | sed ‘2,5c hehehe’ 将2-5行的内容取代为hehehe

Nl /etc/passwd | sed -n ‘2,5p’ 打印2-5行的内容

Sed ‘s/要被取代的字符串/新的字符串/g’ 与vim类似

去除空白行与行首为#的行：

Grep -v ‘^$’ filename | grep -v ‘^#’ 或者 egrep -v ‘^$|^#’ filename（|此处类似于or）

格式化打印

1. printf ‘打印格式’ 实际内容
2. awk ‘条件类型1{动作1} 条件类型2{动作2}’ filename：属于管道命令 可以放 在|之后，相比于sed主要处理行 而awk主要处理行之内的数据

条件类型主要用来筛选行

动作则是对筛选出来的行进行处理

以行为处理单位 以字段为最小的处理单位

如：last -n 5 | awk ‘{print $1 “\t” $2}’ $1、$2代表了一行之内的字段 $0代表 了一整行

NF：每一行拥有的字段总数

NR：目前awk所处理的是第几行数据

FS：目前的分割字符 默认是空格符

如：last -n 5 | awk ‘{print $1 “\t lines:” NR “\t columns:” NF}’

文本比对工具

1. diff [-bBi] from-file to-file
   1. From-file 作为原始比对文件
   2. To-file 作为目的比对文件 即以该文件为基准
   3. From-file to-file 可以用-代替 -代表标准输入 即管道之前的东西
   4. b : 忽略一行中 仅有多个空白的差异
   5. B：忽略空白行的差异
   6. I：忽略大小写的差异
2. cmp 用法与上同 只不过cmp用于比对二进制文件

## Shell script

Read -p “xx” 变量名：等待用户输入并把值赋给变量 xx是提示

Echo -e “xxxx” ：e(反斜杠逃脱解释 即正常的反斜杠)

执行脚本的两种方式

1. 直接执行 ./xxx.sh sh xxx.sh xxx.sh
2. Source(.) xxx.sh 这种执行方式类似于加载配置文件 在脚本内定义的变量会在主 bash中生效 而直接执行的不行 必须要export

test -xxx filename：测试(检测)文件的相关属性

-e 该档名(目录或者文件)是否存在

-f 该档名是否存在且为文件

-d 该档名是否存在且为目录

-r 该档名是否存在且可读

-w 该档名是否存在且可写

-x 。。。。。。。。。。。。。。可执行

Test file1 -nt file2 ：file1是否比file2新

-ot 旧

Test n1 -eq n2 ：n1与n2是否相等

-ne 不想等

-gt 大于

-lt 小于

Test -z string :判断是否为空字符串

-n 不为空串

Test str1 == str2 相等否

!= 不等

[] 判断符号 括号内两边必须加空格 用在if判断中

在括号内外 可以用-a(and) -o(or) 连接两个判断

参数传递 ：如./cataline.sh run 这个run就是参数

$0 指令

$1 第一个参数

$2 第二个参数 以此类推

$# 参数个数

Shift n 参数向左偏移几个

条件判断

1.

If [ 条件判断式 ]；then

xxxx

fi

2.

If [ 条件判断是 ]；then

xxxx

else

xxxxx

fi

3.

If [ 条件判断是 ];then

Xxx

Elif [ 条件判断是 ]; then

Xxx

Else

Xxxx

Fi

4.

Case $变量名称 in

“第一个变量内容”）

Xxxx

;;

“第二个变量内容”）

Xxxx

;;

\*)

Xxxxx

;;

Esac

Function

function fname(){

xxxxxx

}

在脚本中直接使用函数名即可执行函数 参数跟在后面 在函数内部依旧使用$0(函数名称) $1 $2 来去参数的值

注意：函数必须放在最前面

循环

1.

while [ 条件判断式 ]

do

Xxx

done

2.

until [ 条件判断式 ]

do

Xxxx

done

3.

for var in con1 con2 con3 ...

do

Xxxx

done

第一次循环时：$var的内容为con1

第二次循环时：$var的内容为con2

4.

For((初始值；限制值；执行步阶))

Do

Xxx

Done

Shell script Debug

sh [-nvx] xxx.sh

-n ：不执行脚本 仅查询语法

-v ：执行前 先输出脚本的内容

-x ：将使用到的script内容显示到屏幕上 (查看每一步做了什么)

## Linux账号管理与权限设定

1. id 账户名 ：查看账户对应的uid gid

系统的用户都在/etc/passwd文件里面

用户uid对应的密码在/etc/shadow里面

1. groups :查看当前用户所在群组(第一个为有效群组，即新创建的文件的所属组)
2. Newgrp 组名 ：变更有效群组 (这个命令会进入一个子bash 需要exit回到原bash 环境)
3. useradd 账号 ：新增用户

-u 后面接uid

-g 后面接初始群组

-r 建立一个系统账号(uid在1000以内)

系统账号不会主动建立家目录

1. Useradd -D：查看新建账户的默认值(/etc/default/useradd里面)
2. Userdel 账号：删除账号(如果初始群组没有其他用户的话 一同删除初始群组) -r(连 同家目录一并删除)
3. Passwd 账号：根据提示修改密码
4. Passwd ：修改当前账号的密码(还有许多参数 --help查看)
5. chage [-ldeImMW] 账户名：修改密码相关参数
6. Usermod [-cdegGlsuLU] username ：修改账户名的相关数据 而不用去改配置文件
7. Id 账户名(可不加，查询当前)：查询账户的uid等信息
8. Finger ：列出/etc/passwd里面关于当前用户的信息
9. Chsh [-ls]：改变shell -l(列出目前系统上面可用的shell 即/etc/shells里面的内容)

-s 设定修改自己的shell

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*群组\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. groupadd [-g gid] [-r] 组名：新建组 -r(建立系统群组)
2. Groupmod [-g gid] [-n groupname] 群组名：将群组名修改为groupname
3. Groupdel 群组名：删除群组（但不能删除有成员用户的初始群组）
4. gpasswd ：群组管理员

ACL：细化权限控制 可针对某一用户、某一文件或者目录来设定权限 更加强大

Setfacl ：设定某个目录/文件的ACL规范

用法：setfacl [参数] u:账户:权限 文件名称--》设定该账户对该文件所拥有的权限

用法：setfacl [参数] g:群组列表:权限 文件名--》设定该群组对于该文件的权限

Setfacl d:[u|g]:[user|group]:权限 设置该文件(目录)下新创建的文件的权限

Setfacl -b 文件名：取消该文件的ACL权限

getfacl ：取得某个文件/目录的acl设定项目

用法：getfacl 文件名

身份切换

1. su [-lm] [-c 指令] [用户名]：l后加用户名 表示切换身份 m表示使用当前环境 的设定 而不是用新用户的配置 c表示仅用新身份执行一次后面的指令
2. Su -：切换为root

Su - -c 指令 :以root身份执行一次指令`

1. Sudo [-b] [-u 新使用者账号]：b(将后续的指令放到背景让系统自动运行 而不 语目前的shell产生影响) u(后面接欲切换的用户 若没有 则为root)

只有在/etc/sudoers中设定的账户才能使用sudo命令 sudo命令是通过输入自 己的密码 而不是root的密码来执行的

1. 系统账号的shell使用的是/sbin/nologin 系统账号不能登陆取得bash或者其他 shell 但是他们可以使用系统的其他资源

PAM ：身份认证API(密码验证等)

查询使用者：w、who、last、lastlog(查询每个账号的最近登入时间)

与使用者对话(需要同时在线) write、mesg、wall

Write 使用者账户 [用户所在终端接口(who可以看到)]

mesg ：查看当前用户是否接受信息

Wall “xxxxx” ：发送给所有的人

使用邮箱(不需要同时在线)

寄出邮件

mail -s “邮件标题” username@localhost

Mail -s “邮件标题” username@localhost <filename

收信

Mail 在&提示符后面输入？号根据提示查看

Quota(限额)

限制每个人的网页空间容量、邮件空间容量、最大的可用网络硬盘空间

、某一群组所能使用的最大磁盘空间、某一用户最大磁盘配额、某一目录最大磁盘 空间等。通过限制inode和block来实现

例行工作日程

at :只执行一次 需要atd服务支持

Crontab:循环执行，需要crond服务支持

Systemctl restart atd ：重启atd服务

Systemctl enable atd ：开机即启动这个服务

Systemctl status atd：查询atd目前的状态

查询工作例程：atq

取消工作历程：atrm 工作号

查询工作历程：crontab -l

移除工作历程：crontab -r (移除所有)

编辑工作历程：crontab -e

进程

ps -l ：列出进程清单(当前bash下的进程)

Ps -aux ：列出所有进程(更加详细)

指令后面跟&：将该指令放到背景里面去执行 为了让执行过程中的输出不显示到 前景(会影响我们前景的工作) 可用数据重导向 在指令后面加

>xxx.txt 2>&1 再加&

所有在bash环境又指令中产生的进程都是bash的子进程 每一个bash只能控制 自己的子进程 不能跨bash去控制别人的子进程 就算root也不行。

Ctrl+Z：将目前的工作丢到背景中暂停 需要时在放到前景中继续执行（fg指令）

Jobs [-lrs] ：观察目前的背景工作状态 -l(列出job number和指令串和pid)

-r(仅列出正在背景run的工作) -s（仅列出在背景当中暂停的工作）

显示中的+号表示最近被放到背景的进程 而—号则代表倒数第二个 被放到背景的进程 其他的就没有这些符号了

fg %jobnumber：从背景拿到前景来工作 %号可用可无 或者fg +(-)

bg %jobnumber：让在背景的工作状态变为运行中 %可用可无

Kill -signal %jobnumber(或者pid) ：对进程进行操作

上面三个指令中%号的说明：加%号代表工作 不加代表进程(1号进程代表系统 进程 如果不加%号杀死的话 则系统奔溃)

Kill -l：列出能被使用的signal

-1 ：重新读取一次参数的配置文件 相当于reload

-2 ：类似于ctrl+c

-3 ：强制删除一个工作

-15 ：正常的进程方式结束一个进程(默认的)

脱机问题：这里的放到背景中去执行的 指的是放到bash的背景而不是系统背景

因此，当终端断开时，这些在背景执行的工作会终端 可以使用at或者

Nohup这样的指令加上& 则是放到系统背景里面去 输出的内容不是到终 端而是到指定文件

进程相关命令：

Ps -l ：查询当前bash的进程 注意查看每一列代表什么(重点)

Ps aux：查询系统的所有进程

Ps -lA ：查看系统所有的进程 与上同 同ps -e

Ps axjf ：查看进程树

进程信息后面跟<defunct> 代表僵尸进程

top [-d 数字]：动态查看进程信息 根据数字刷新

Top [-bnp]：-b批次的方式执行top 通常搭配数据流重导向将批次的结果输出为文 件 -n 与-b搭配 意思是需要进行几次top的输出结果

-p 指定某些个pid来进行观察监测(常用)

Top执行过程中可以使用的按键：

？：显示在top当中可以输入的按键指令

P：以cpu的使用率排序

M：以内存的使用率排序

N：以pid来排序

T：由该进程使用cpu的时间积累来排序

k：给予某个pid一个讯号

r：给某个pid重新制定一个nice值(用于提高优先级)

q：离开

Echo $$：获取当前bash的pid

pstree ：查看进程树 用于查找进程之间的相关性

-A：各进程树之间以ASCII字符连接

-U：各进程数之间以UTF-8来连接

-p：同时列出pid

-u：同时列出每个进程的所属账户名称

Kill -signal PID：给进程传达讯号

Killall -signal 指令名称：对由该指令创建的进程发讯号(如杀死等)

Priority：影响进程的优先级，动态生成

Nice：影响进程的优先级，可手动指定

设置进程的优先级：

nice [-n 数字] command：下达指令时就指定nice值

Renice [number] pid ：重新调整nice值

观察系统资源的命令：

free [-bkmg] ：显示内存的使用情况(硬盘) 参数bkmg代表单位

uname [-asrmpi]：观察系统与核心相关信息

-a:所有系统相关信息，包括底下的数据都会被列出来

-s:系统核心名称

-r:核心的版本

-m：硬件名称

-p：cpu的型号

-i：硬件的平台

Uptime:观察系统启动时间与工作负载

Netstat [-atunlp]:追踪网络或插槽文件

-a：将目前系统上所有的联机、监听、Socket数据都列出来

-t：列出tcp网络封包的数据

-u：列出udp网络封包的数据

-l：列出目前正在网络监听的服务

-p：列出该网络服务的进程的pid

-n：不以进程的服务名称，以端口号显示

Dmesg：分析核心产生的讯息

vmstat：侦测系统资源变化

在内存里面的进程等相关数据都被写入到/proc目录下（\*）

fuser [-umv] [-k [i] [-signal]] file/dir：找出正在使用该文件(或目录)的进程

-u：列出进程的pid和进程的拥有者

-v：列出相关指令

-k：给该进程发讯号

-i：配合-k使用 在删除之前询问

-signal：默认-9(杀死)

losf [-aUu] [+d]：列出被进程开启的文件档名

Pidof [-sx] program\_name1 program\_name2：根据进程名找pid

-s：仅列出一个pid 而不是列出所有的pid

## SELinux（安全强化）

针对进程读取文件的权限进行设定，在这之前，只要改进程的拥有着是root，那么该进程就可读写所有文件，这是非常危险的，现在对进程进行设定，只为他开放某些文件

可作为专题来学习

## 系统服务

提供服务的那一只进程称为daemon

服务启动的分类(老的ini方式使用 CentOS不在使用 而是使用systemd)：

独立启动的模式：独立启动。常驻内存，提供服务，反应快

总管程序：提供socket和port的管理，有用户请求时，总管程序回去唤醒相关的 程序。

Systemd管理的服务有很多分类，如service、socket(进程之间的数据交换)、target(一堆服务的集合)、mount(文件挂载相关的服务)、path(侦探文件和目录)、timer(循环执行的服务)等，看扩展名即可区分

服务管理指令：systemctl [command] [unit]

command主要有

Start：立即启动后面接的服务单元

Stop：立即关闭服务

Restart：重启服务

Reload：重载配置文件 让设定生效

Enable：设置为开机启动

Disable：开机不会启动

Status：查看状态

Is-active：目前有没有在运行中

Is-enable：有没有设置为开机启动

服务状态：

Running：正在运行

Exited：仅执行一次就正常结束的服务，目前没有在进行

Waiting：正在执行中，但等待处理事情

Inactive：没有执行的意思

Dead：没有执行

预设状态

Enable：开机自启动

Disable：开机不启动

Static：不可自己启动（disable），不过可能会被其他的enable唤醒

Mask：无论如何都不会被启动，因为已经被强制注销 可以通过systemctl unmask 改回来

注销服务 systemctl mask(unmask 还原回来) 服务名

观察系统上所有的服务

Systemctl [command] [--type=TYPE] [--all]

Command:

List-units：依据unit列出目前有启动的unit(默认)。若加上--all才会列出没有启 动的

List-unit-files：依据/usr/lib/systemd/system/内的文件 ，将所有文件列表说 明

TYPE:

主要有service、socket、target(看操作环境时可以用)等

观察当前的操作环境是有那个服务提供的：

Systemctl [command] [unit.target]

Command:

Get-default：取得目前的target

Set-default：设定后面接的target成为默认的操作模式

Isolate：切换到后面接的模式

分析各服务之间的依赖性

Systemctl list-dependencies [unit] [--reverse]

--reverse：反响追踪谁使用这个unit的意思

查看服务对应的端口：

/etc/services文件

关闭网络服务：systemctl stop 服务名称

Systemctl配置文件

/usr/lib/systemd/system/xxx：官方的配置文件 建议不要修改

/etc/systemd/system/xxx：我们自己可以定于一些服务在这里

怎样写一个服务呢？

1. 首先血一个shell脚本，以sh结尾的脚本
2. 写一个启动脚本设定，以service结尾
3. 启动服务：systemctl start xxx.service

疑问：直接执行脚本不行么？？？

提供定时执行的服务：timer.target

## 登陆档（系统、服务等日志）

都放在/var/log/目录下，主要有：

1. /var/log/boot.log

开机信息日志，包括硬件检测、核心等

1. /var/log/cron

例行工作排程 即定时器执行情况

1. /var/log/dmesg

开机启动核心侦探产生的信息

1. /var/log/lastlog

系统所有账号最近一次登陆的相关信息

1. /var/log/maillog

邮件相关日志

1. /var/log/messages

系统所产生的错误（重要的）信息 如果系统发生莫名其妙的错误 一定要看

1. /var/log/secure

只要牵涉到需要输入账号密码的软件 在登入是都会被记录

1. /var/log/wtmp，/var/log/faillog

记录正确登入系统这得账户信息

1. /var/log/httpd/xxx

记录不同服务所产生的日志

CentOS提供rsyslog.service服务来管理日志（写入文件）提供systemd-journald.service 记录日志在内存中(因为rsyslog.service要在服务启动后才能处理日志)，提供logrotate 服务来对日志进行清理

与讯息相关的配置文件在/etc/rsyslog.conf里面 设置讯息等级 以及将信息写入到什么地方

登陆档的安全性设置：

设置登陆档部可以被删除，防止黑客入侵时删除数据：chattr +a xxx：只能增加数据而不能被删除，root除外，此外，使用vim等编辑日志文件后，该日志文件就不能继续添加日志了，必须重启rsyslogd.service服务

登陆档服务器：

将主机A产生的日志传到另外一台主机B(登陆档服务器)上，方便查看。

做法：在主机B上修改rsyslog.conf（启动514端口）

在主机A上修改rsyslog.conf（指定讯息传到B的ip地址）

Logrotate：对日志进行清理，可在配置文件/etc/logrotate.conf中修改清理的参数

Journalctl：查看在rsyslog.service启动之前的日志 ，这些日志是保存在内存中的

Logwatch(需要安装)：分析日志工具

Mount：测试挂载

开机加载顺序：

BIOS-->MBR-->boot Loader-->Kernel+initramfs-->systemd

## 系统基本设定

网络设定：

1. 设定ip网络参数
   1. Nmcli connection show [网卡代号]：查看所有网卡或者查看指定网卡信息
   2. Nmcli connection modify 网卡代号 参数名 参数值 参数名 参数值.....

不知道参数名和参数值 可直接查看：Nmcli connection show网卡代号

* 1. 第二步只是修改了配置 还要启动才能生效：nmcli connection up 网卡代 号

1. 修改主机名
   1. Hostnamectl [set-hostname 主机名]
   2. Hostnamectl：显示当前主机信息
2. 日期与时间设定
3. Timedatectl：查看本机时间与日期

list-timezones：列出系统上所有支持的时区名称

Set-timezone：设定时区位置

Set-time：设定时间

Set-ntp：设定网络矫时系统

1. 时间的调整
2. Timedatectl set-time “yyyy-mm-dd HH：MM”
3. 手动网络矫正时间：ntpdate tock.stdtime.gov.tw

4. 语系设定

Localectl locale：查看语系

## 服务器硬件参数的收集 用于升级或者加内存时

Dmidecode -t type

Type:

1. 详细的系统数据

4、Cpu的详细信息

9、系统插槽的相关信息

17、内存插槽信息

## 软件安装

1、File + 文件 ：查看文件的类型

2、Make指令时为了简化编译过程，会去找makefile文本文件，里面有关于如何编译的详细信息，而makefile文本文件则是由configure(侦探环境，由软件开发商提供)程序产生

3、makefile的语法规则

目标(target) : 目标文件1 目标文件2

<tab> gcc -o 欲建立的执行文件 目标文件1 目标文件2

一个makefile可有多个目标 在命令行使用make target进行不同的动作，如清楚、 编译等

1. 源码安装步骤：
   * 1. 取得原始码：将tarball文件在/usr/local/src目录下解压缩
     2. 取得步骤流程：进入目录下查阅install与readme等相关文件内容
     3. 依赖软件安装：根据install/readme的内容查看并安装一些依赖的软件
     4. 建立makefile：以自动侦测程序侦测作业环境，并建立makefile文件
     5. 编译：以makefile这个程序并使用该目录下的makefile作为他的参数配置文件，来进行编译动作
     6. 安装：以make这个程序，并以makefile这个参数配置文件，依赖install这个目标（target）的指定来安装到正确的路径

注：成功安装软件的目录下一般会有etc(配置文件)、bin(执行档)、lib(函数库)、 man(帮助文档)这几个目录，如/usr/local/apache/etc、usr/local/bin等，而如果 默认安装(即不指定目录) 则相关文件会被放到 /usr/xx、/usr/lib、/usr/bin下 面 这样不便于升级和清除 建议一个软件放到/usr/local的一个目录下 安装 好以后还需要配置一下路径就ok了 不过现在一般不用源码安装 太麻烦！！！

1. 利用patch更新原始码

函数库：

静态函数库：xxx.a

动态函数库：xxx.so

## 软件安装

CentOS软件管理机制：RPM 升级机制：YUM

Ubuntu软件管理机制：dpkg 升级机制：apt

RPM：将编译好的数据、依赖软件等打包。在安装的时候将相关信息放到数据库中