- 指令、选项或参数之间不论空几个格，shell 都视为一个空格。
- 指令太长时，可以使用“\”（反斜杠）符号使指令连续到下一行。
- Linux 环境下，字母**区分大小写**。
- shell 就是用户接口，Linux 下默认的用户接口就是 bash shell。
- 如果想让当前程序**终止**，可以键入 **ctrl+c**，这就是中断当前程序的按键。（q：有很多程序在运行时，如果想跳出来，按下 q 即可！）
- 如：drwxr--r-- 3 root root 4096 Jun 25 08:35 .ssh

  非 root 这个账号的其它使用者均不可进入 .ssh 这个目录，为什么呢？因为 x 与目录的关系相当的重要，如果您在该目录底下不能执行任何指令的话，那么自然也就无法进入了。（如果无法执行指令，则无法进入目录）
- 在 Windows 下一个文件是否具有执行能力是由后缀名来判断的，例如：.exe, .bat, .com 等等，但**在 Linux 下，文件是否能执行，则是由是否具有 x 这个属性来决定的**，跟后缀名没有绝对的关系。
- 如果文件名之前多一个“.”，则表明这个文件为“**隐藏文档**”。ls -al 可以看到这种文件的文件名及其相关属性。
- **Linux 的正统文件格式是 ext2**。常用的带有日志管理功能的文件格式有 **ext3**、**reiserfs**。
- Linux 下有关目录下存放的内容：

  /boot ： 存放 **Linux 核心与启动相关文件**的地方；

  /dev ： 存放与**设备**有关的文件；

  /etc ： 系统在**启动过程中要读取的文件**均在该目录下；

  /etc/rc.d/init.d ： 所有**服务**默认的**启动脚本**文件都放在这里；

  /bin，/sbin，/usr/bin，/usr/sbin ： 系统默认的**执行文件**的放置目录。其中/bin，/usr/bin 是**系统用户**使用的目录，而/sbin，/usr/sbin 则是**系统管理员**使用的目录；

  /var/log ： 存放所有**服务**的**登录文件**或**错误信息文件**。
- **绝对路径**：路径的写法一定是**从根目录“/”写起**，例如：/usr/share/doc 目录。

  相对路径：路径的写法不是由“/”写起，例如从/usr/share/doc 转到/usr/share/man 下时，可以写成 cd ../man。

  （Windows 下直接输盘符如：E: 即可进入 E 盘，查看磁盘下的目录用 dir）
- **.** 代表当前层目录

  .. 代表上层目录

  ~ 代表自己的根目录

  ~user 代表到 user 这个人的根目录

- 当我们执行一个指令时，系统会依照环境变量 PATH 的设定到 PATH 定义的每个路径下搜寻文件，先搜寻到的指令文件先被执行。
- 当需要备份很大的文件但这个文件的更新率很低时，可以使用指令“cp –u 源文件 目标文件”进行备份。因为只有当源文件被改变后，才会进行复制操作。否则不做任何动作。（将源文件复制成目标文件）
- 若用户建立的是“文件”，则默认没有可执行（x）项，即只有 rw 这两项，最大为 666（rw- rw- rw-）；若用户建立的是“目录”，则由于 x 与是否可以进入此目录有关，因此默认为所有权限均开放，即 777（drwxrwxrwx）。
- umask 用于指定当前用户在建立文件或目录时的属性默认值。其指定的是“该默认值需要取消的权限”。（如：先用 umask 设定权限，再创建目录时权限就会被应用到创建的目录上。）

- **SUID**
  当一个设置了 SUID 位的可执行文件被执行时，该文件将以所有者的身份运行，也就是说无论谁来执行这个文件,他都有文件所有者的特权。如果所有者是 root 的话，那么执行人就有超级用户的特权了。
- **SGID**
  当一个设置了 SGID 位的可执行文件运行时，该文件将具有所属组的特权，任意存取整个组所能使用的系统资源。
  若一个目录设置了 SGID,则所有被复制到这个目录下的文件，其所属的组都会被重设为和这个目录一样，除非在复制文件时加上-p （preserve，保留文件属性）的参数，才能保留原来所属的群组设置。
- **sticky-bit**
  对一个文件设置了 sticky-bit 之后，尽管其他用户有写权限，也由属主执行删除、移动等操作。
  对一个目录设置了 sticky-bit 之后,存放在该目录的文件仅准许其属主执行删除、移动等操作。
- 命令摘录：
  # exit：退出系统，相当于退出当前登录用户
  # Ctrl+Alt+[F1]~[F6]：文字界面 tty1~tty6 终端
  # Ctrl+Alt+[F7]：图形界面
  # startx：启动图形界面
  # bc：计算器
  # man+指令名称：请求系统给出指令的操作说明
  # shutdown –h now：立刻关机，now 相当于时间为 0
  # chgrp 群组名称 文件或目录（**改变文件或目录的群组**，要求改变的目标群组名称必须在/etc/group 中存在）
  # chown [-R] **账号名称**[:群组名称] **文件或目录**（**改变文件拥有者**，要求拥有者名称在/etc/passwd 文件中存在，加参数-R 可以直接将目录下的所有子目录或文件同时更改文件拥有者）
  # cp 源文件 目标文件（复制文件）
  # chmod [-R] rwx 文件或目录（更改文件权限，rwx 为三组属性每组内 rwx 相加结果，如 770 代表 rwxrwx---）
  # pwd（显示当前所在目录，**p**rint **w**orking **d**irectory）
  # mkdir [目录名称]（创建目录）
  # rmdir [目录名称]（删除目录）
  # rm [-fir] [文件名]（删除文件或目录，参数说明：-i 为提供用户确认，此为默认值；-r 为循环删除，直到没有东西为止；-f 为 force，强制删除。如 rm –rf test 为连续删除该目录下的所有文件与目录）
  # mv [-u] [源文件] [目标文件]（可用来移动文件或更改文件名；参数说明：-u 为 update 简写，当源文件比目标文件还新时才会动作！可用来测试新旧文件，看是否需要搬移）
  # cat 由第一行开始显示文件内容，读取文本文件
  # tac 从最后一行开始显示，可以看出 tac 是 cat 的倒写
  # nl 显示时同时输出行号
  # more 一页一页地显示文件内容
  # less 与 more 类似，但是比 more 更好的是，可以往前翻页

head 只看头几行
tail 只看末尾几行
od 以二进制的方式读取文件内容
touch 命令参数可更改文档或目录的日期时间，包括存取时间和更改时间。
chattr 设定文件隐藏属性
lsattr 显示文件隐藏属性
which 查看可执行文件的位置（通过 PATH 环境变量到该路径内可执行文件）
whereis 查看文件的位置
locate 配合数据库查看文件位置
find 实际搜寻硬盘查询文件名称
grep ^[w-z] /etc/* (在/etc 里，只要句首是 w-z 的就将它列出来)

- 硬链接和符号链接（软链接）

  Block 是记录文件内容数据的区域，它是磁盘可以记录的最小单位，由数个扇区组成；**inode** 则是记录该文件的属性及其放置在哪个 Block 之内的信息。每个文件都会占用一个 inode。

  当系统要读取某个文件时，它会先读 inode table，然后根据 inode 的信息到数据区域将数据取出备用。

  硬链接：在原有的 inode 引用上加一；限制：（1）不能跨文件系统，因为不同的文件系统有不同的 inode table；（2）**不能链接目录**。

  符号链接：建立了一个 inode，用来指向源文件的 inode，类似快捷方式。当源文件被删除，符号链接的文件就打不开了。

  区别：（1）硬链接的源文件和链接文件共用一个 inode，而软链接源文件和链接文件拥有不同的 inode；（2）在文件属性上，软链接明确写出了是链接文件，而硬链接没有写出；（3）文件大小不同，硬链接文件显示的大小与源文件相同，软链接显示的大小与源文件不同（较小）。

  命令： ln [-s] [源文件] [目标文件]

  参数说明：-s :提供符号链接，如果直接使用 ln 而不加任何参数，就属于硬链接。

- vi 是 Unix/Linux 默认的字处理软件，分为 3 种模式：

  一般模式：以 vi 处理文件时，已进入该文件就是一般模式，可处理删除、复制、粘贴等动作，无法编辑；

  编辑模式：在一般模式下按下 i,I,o,O,a,A,r,R 等字母之后才会进入编辑模式，按下 Esc 键回到一般模式；

  命令行模式：在一般模式中，输入“:”、“/”就可以将光标移动到最末一行。可查找数据、读取、存盘、大量字符替换、退出 vi、显示行号等。

- 常用 vi 指令

  一般模式

Ctrl + f	屏幕向前翻动一页
Ctrl + b	屏幕向后翻动一页
0	数字 0 表示移动到这一行的第一个字符处
$	移动到这一行的最后一个字符处
G	移动到这个文件的最后一行
n<Enter>	光标向下移动 n 行
/word	在光标之后**查找**一个名为 word 的字符串
:n1,n2s/word1/word2/g	在第 n1 与 n2 行之间**查找** word1 这个字符串，并将该字符串**替**

	换为 word2
:1,$s/word1/word2/g	从第一行到最后一行**查找** word1 这个字符串，并将该字符串**替换**为 word2
:1,$s/word1/word2/gc	从第一行到最后一行查找 word1 这个字符串，并将该字符串替换为 word2，且在替换前显示提示符让用户确认（confirm），之后弹出 replace with word2 (y/n/a/q/l/^E/^Y)? The "y" and "n" are self-explanatory, but what about the rest? To tell Vim to go ahead and replace all instances of the matched string, answer with a. If you realize that you don't really want to make the changes, you can tell Vim to quit the operation using q. To tell Vim to make the current change and then stop, use l, for last. ^E and ^Y allow you to scroll the text using Ctrl-e and Ctrl-y.
x, X	x 为向后**删除**一个字符，X 为向前删除一个字符
dd	**删除**光标所在的那一行（实际并未删除，相当于**剪切**，按 p 会粘贴刚才删除的内容）
ndd	删除光标所在行的向下 n 行
yy	**复制**光标所在行
nyy	复制光标所在行的向下 n 行
p,P	p 为复制的数据**粘贴**在光标下一行，P 则为粘贴在光标上一行
J	将光标所在行与下一行的数据结合成一行
u	**恢复前一个动作**

编辑模式

i, I	插入：在当前光标所在处插入输入的文字，已存在的字符会向后退
a, A	添加：由当前光标所在处的下一个字符开始输入，已存在的字符会向后退
o, O	插入新的一行：从光标所在处的下一行行首开时输入字符
r, R	替换：r 会替换光标所指的那一个字符，R 会一直替换光标所指的文字，直到按下 Esc 为止
Esc	退出编辑模式，回到一般模式

命令行模式

:w	将编辑的数据写入硬盘文件中
:q	退出 vi；若曾修改过文件，又不想保存，使用:q! 为强制退出不保存文件
:wq	保存后退出，若为：wq!，则为强制保存后退出

- **BASH Shell** = Bourne（人名） Again Shell（Linux 使用的 shell，也是 GNU 操作系统中标准的 shell）主要**优点**如下：

（1）**命令编辑能力**：能记忆使用过的命令，但若黑客入侵，只需翻查执行过的指令（如 MySQL 的密码），就可能破解 Linux 主机；

（2）**补全功能**：指令补全和文件名称补全；

（3）**命令别名（alias）设定功能**：输入 alias 可查看当前的命令别名，可用形如“alias lm='ls -al'”设定别名；（注：lm、=、'ls -al'间不能有空格）；

（4）**作业控制、前景背景控制**

（5）**Shell scripts 的强大功能**：将频繁输入的连续指令写成一个文件，也可借由 shell 提供的环境变量及相关指令编写一个小型的程序语言。

- root 的根目录在/root 下，一般用户的根目录则在/etc/passwd 文件中设定。
- 变量就是以一组文字或符号取代一些设定或一串数据。
- 显示变量要用到 echo 指令，Linux 系统预设变量名称前会加一个$符号，如以 echo $PATH 可显示 PATH 变量的具体内容。
- Linux 默认情况下，使用大写字母设定的变量一般都是系统的预设变量。使用指令 env 可以查看当前系统中的主要环境变量。set 指令可以将当前系统中所有的变量数据都读出来。登入 Linux 后会取得一个 PID，而该次的设定将只对这个 PID 及其子程序有关。另外，这次登入所做的变量设定，如果没有影响配置文件，那么此次设定的变量在下次登入时将被取消（因为程序 PID 不见了）。所以，如果想每次登入时自动设定好变量，必须将设定写入登入时加载的配置文件。
- 设定变量时需要注意一下规则：
  （1）等号两边不能直接接空格符；
  （2）若该变量为扩增变量内容时，则需以双引号及$变量名称（如“$PATH”:/home）继续累加内容；
  （3）若该变量需要在其他子程序执行，则以 export 使变量可以动作，如 export PATH。
- 在设定变量时，单引号与双引号有什么不同？
  答：最大不同在于双引号仍然可以保留变量的内容，但单引号内仅能是一般字符，即用单引号括起来后，变量失去了原有的意义，只是普通的字符组合而已。
- 在一串指令中，在 ` 之内的指令将被首先执行，而其执行结果将作为外部的输入信息。
- 如果要执行上一个指令，除了使用上下键外，还可以直接以!!来表示；要执行第 n 条指令，可以使用!n 表示。
- 如果需要将当前的配置文件内容读入一次，需要重新注销再登录；若想不注销而直接读入变量配置文件，使用“source 变量配置文件”即可。
- 用“echo $?”输出的结果代表前一个执行的指令内容有没有错误，如果有错误就返回 1，没有错误就返回 0。
- 要执行上一层目录中的命令，可以输入“../command”，其中的 command 指的是存在的**可执行文件**。
- 在执行文件时，系统默认是不主动搜寻当前目录下的执行文件，而是按照 PATH 的设定搜寻。要执行当前目录下的执行文件，使用“./command”即可。
- 命令重定向就是将目前所得数据转到其他地方。
  # ls –al > list.txt 注：将显示结果输出到 list.txt 文件中，若该文件已存在则予以**取代**
  # ls –al >> list.txt 注：将显示结果**累加**到 list.txt 文件中，该文件为累加的，**旧数据保留**！
  # ls –al 1> list.txt 2> list.err 注：将显示数据正确输出到 list.txt，错误的数据输出到 list.err
  # ls –al 1> list.txt 2> &1 注：将显示数据不论正确或错误均输出到 list.txt 中，注意，**错误与正确信息输出到同一个文件中，则必须这样写**，不能写成其他格式！
  # ls –al 1> list.txt 2> /dev/null 注：将显示的数据，正确的输出到 list.txt，错误的数据予以丢弃！
- 什么时候需要使用命令输出重定向？
  答：（1）当屏幕输出的信息很重要，而且需要将它保存时；
  （2）背景执行中的程序，不希望它干扰屏幕正常的输出结果时；
  （3）一些系统的例行性命令（如写在/etc/crontab[crontab 命令用于设置周期性被执行的指令]中的文件）的执行结果，希望它可以保存下来时；
  （4）一些执行命令，已知道可能的错误信息，想将其丢弃时；
  （5）错误信息与正确信息需要分别输出时。

- 如果数据必需经过几道手续之后才能得到想要的格式，需要使用 pipe 命令，其使用“|”界定符号，另外，pipe 命令仅能处理经由前一个指令传来的正确信息，也就是标准输出信息，对于标准错误信息并没有直接处理能力。
- 环境变量文件的加载顺序是什么？
  答：先由/etc/passwd 取得 bash，再到/etc/profile 读取主要的环境变量，同时将/etc/inputrc 及/etc/profile.d 内容读入。之后，再到个人的根目录读取~/.bash_profile 及~/.bashrc 等文件。
- 连续输入命令时，“; && ||”有何不同？
  答：分号可以让两个 command 连续运作，不考虑 command1 的输出状态；&&则前一个指令必需没有错误信息，亦即返回值需为 0，则 command2 才会被执行；||则与&&相反。
- 常见文件压缩名后缀：
  *.Z compress 程序压缩的文件；
  *.bz2 bzip2 程序压缩的文件；
  *.gz gzip 程序压缩的文件；
  *.tar tar 程序打包的数据，没有压缩过；
  *.tar.gz tar 程序打包的文件，并且经过 gzip 的压缩。
  bzip2, gzip 与 compress 在没有加入特殊参数时，原先的文件会被取代掉，但是使用 tar 则原来的与后来的文件都会存在。
- 脚本的后缀名最好为.sh；但并非加上.sh 就是可执行文件，还需要查看其属性中是否有 x 属性。
- 调试脚本的命令：# sh [-nvx] scripts
  -n：不执行脚本，查询脚本内的语法，若有错误则列出
  -v：在执行脚本之前，先将脚本的内容显示在屏幕上
  -x：将用到的脚本内容显示在屏幕上，与-v 稍微不同
- shell 脚本代码示例

```
# !/bin/bash
# This script is written to show the basic grammar that shell uses.
# Date:2016/01/09
# Author:lyn
declare -i year=2016
echo "enter the month:(int)"

read month
if [ "$month" = "1" ] && [ "$year" != "" ]; then
      echo "It's January,$year"
elif [ "$month" = "1" ]; then
      echo "It's February"
else
      echo "Lucky to know you"
fi

case $1 in
   bird)
```

```
                echo "bird is flying"
                ;;
        fish)
                echo "fish is swimming"
                ;;
        *)
                echo "you are thinking"
esac

account=$(cut -d ":" -f1 /etc/passwd|sort)
echo "The following are accounts existing in your linux server:"
for i in $account
do
     echo $i
done
```

保存文件为 lyn.sh，并在 Linux 执行输出结果（部分）如下：

```
[root@lyn doleria]# lyn.sh bird
enter the month:(int)
1
It's January,2016
bird is flying
The following are accounts existing in your linux server:
abrt
adm
apache
avahi-autoipd
bin
daemon
dbus
doleria
```