1. How to check linux is 32-bit or 64-bit
   1. cat /proc/version
   2. uname -a
   3. uname -m
   4. arch
   5. file /sbin/init
2. RISC (reduced instruction set computer)-majority
   1. Oracle: SPARC CPU
   2. IBM: PowerPC
   3. ARM: ARM CPU
3. CISC (complex instruction set computer)
   1. AMD: x86
   2. Intel: x86
   3. VIA: x86
4. Units of computer memory measurement
   1. Bit
   2. Byte
   3. Kilo Byte
   4. Mega Byte
   5. Giga Byte
   6. Terra Byte
   7. Peta Byte
   8. Exa Byte
   9. Zetta Byte
   10. Yotta Byte
   11. Bronto Byte
   12. Geop Byte
5. 购买硬盘的容量是十进制，等格式化完后是二进制形式
   1. 硬盘的最小物理单位是512bytes
   2. 最小的组成单位是扇区（sector）
   3. 通常硬盘的计算采用“多少个sector”,所以才采用十进制。
6. 计算机性能
   1. 运行速度
      1. MIPS: Million Instructions Per Second
      2. MHZ: Mega Hertz
      3. GHZ: Giga Hertz
      4. Clock rate:
      5. 将在单位时间（如1秒）内所产生的脉冲个数称为频率。
      6. 频率是描述周期性循环信号（包括脉冲信号）在单位时间内所出现的脉冲数量多少的计量名称
      7. 频率的标准计量单位是Hz（赫）
      8. CPU的主频，即CPU内核工作的时钟频率（CPU Clock Speed）
      9. CPU的频率就是CPU每秒钟可以进行的工作次数。
      10. CPU频率越高表示这颗CPU单位时间内可以做更多的事情
      11. 不同的CPU之间不能单纯的以频率来判断运算性能
      12. 所谓的外频指的是CPU与外部元件进行数据传输时的速度
      13. 倍频则是CPU内部用来加速工作性能的一个倍数
      14. 外频与倍频两者相乘才是CPU的频率速度
      15. 所谓的超频指的是: 将CPU的倍频或者是外频通过主板的设置功能更改成较高频率的一种方式
      16. 在目前32位系统分页大小为4K, 寻址位宽为32位, 用到了32位中的高20位来标记物理页，剩余的12位用于添加一些标记信息等。这样，就可以支持到最大(2^20)\*4096=4GB内存
      17. Intel的turbo技术: CPU会根据当前的任务量自动调整CPU主频，从而重任务时发挥最大的性能，轻任务时发挥最大节能优势
   2. 内存的容量: 内存越大，系统越强，能处理的数据就越庞大
   3. **内存除了频率/带宽与型号需要考虑之外,内存的容量也是很重要**的喔! 因为所有的数据都得 要载入内存当中才能够被CPU判读,如果内存容量不够大的话将会导致某些大容量数据无法 被完整的载入, 此时已存在内存当中但暂时没有被使用到的数据必须要先被释放,使得可用 内存容量大于该数据,那份新数据才能够被载入呢。
   4. 外存的容量 : 硬盘
   5. 字长： 64-bit （since 2009）
7. 超线程 (Hyper-Threading, HT) ，可以有两个程序“同时竞 CPU 的运算单元”,而非通过操作系统的多任务切换! 这一过程就会让 CPU 好像“同时有两个核心”的模样! case by case
8. 个人电脑的内存主要元件为动态随机存取内存 (Dynamic Random Access Memory, **DRAM**), 随机存取内存只有在通电时才能记录与使 用,断电后数据就消失了。因此我们也称这种RAM为挥发性内存 。
9. **DDR**是所谓的双倍数据传送 速度(Double Data Rate), 他可以在一次工作周期中进行两次数据的传送,感觉上就好像是CPU的倍频。
10. **DDR SDRAM**又依据技术的发展,有DDR, DDR2, DDR3, DDR4等等,其中,DDR2 的频率 倍数则是 4 倍而DDR3 则是 8 倍。
11. DDR3 标准电压为 1.5V,但 DDR3L 则仅须 1.35V 喔! 通常可以用在耗电量需求更低的笔记本中! 但并非所有的系统都同步支持! 这就得要看主板的支持规格。
12. 除了内存条之外, 事实上整部个人电脑当中还有许许多多的内存存在喔!最为我们所知的就是CPU内的第二层高速缓存内存。 我们现在知道CPU的数据都是由内存提供,但CPU到内存 之间还是得要通过内存控制器啊! 如果某些很常用的程序或数据可以放置到CPU内部的话, 那么CPU数据的读取就不需要跑到内存重新读取了。
13. 因为第二层高速缓存**(L2 cache**)整合到CPU内部,因此这个L2内存的速度必须要CPU频率 相同。 使用DRAM是无法达到这个频率速度的,此时就需要静态随机存取内存(Static Random Access Memory, **SRAM**)的帮忙了.
14. CMOS主要的 功能为记录主板上面的重要参数, 包括系统时间、CPU电压与频率、各项设备的I/O位址与 IRQ等,由于这些数据的记录要花费电力,因此主板上面才有电池.
15. BIOS为写入到主板上某 一块 flash 或 EEPROM 的程序,他可以在开机的时候执行,以载入CMOS当中的参数, 并尝 试调用储存设备中的开机程序,进一步进入操作系统当中。BIOS程序也可以修改CMOS中的 数据, 每种主板调用BIOS设置程序的按键都不同,一般台式机常见的是使用[del]按键进入 BIOS设置画面
16. 固件(firmware)很多也是使用ROM来进行软件的写入的。 固件像软件一样也是 一个被电脑所执行的程序,然而他是对于硬件内部而言更加重要的部分，很多的硬件上头都会有ROM来写入固件这个软件。现在的 BIOS 通常是写入类似闪存 (flash) 或 EEPROM中
17. 显卡又称为**VGA**(Video Graphics Array),他对于图形影像的显示扮演相当关键的角色。 一 般对于图形影像的显示重点在于分辨率与色彩深度,因为每个图像显示的颜色会占用掉内 存, 因此显卡上面会有一个内存的容量,这个显存容量将会影响到你的屏幕分辨率与色彩深度。显卡主要也是通过CPU的控制芯片来与CPU、内存等沟通。
18. PCIe 3.0 的 16x,因此最大带宽就可以到达接近 32GBytes/s的传输量
19. 假设你的桌面使用1024x768分辨率,且使用全彩(每个像素占用3Bytes的容量),请问你的显卡至少需要多少内存才能使用这样的彩度?答:因为1024x768分辨率中会有786432个像素, 每个像素占用3Bytes,所以总共需要2.25MBytes以上才行。
20. 显卡是通过什么格式与电脑屏幕 (或电视) 连接的呢?目前主要的连接接口有：

D-Sub (**VGA**端子):为较早之前的连接接口,主要为 15 针的连接,为模拟讯号的传 输,当初设计是针对传统图像管屏幕而来

**DVI**: 三种类型包括DVI-A、DVI-D和DVI-I的接口形式。DVI-D只有数字接口，DVI-I有数字和模拟接口，目前应用主要以DVI-D为主，同时DVI-D和DVI-I又有单通道（Single Link）和双通道（Dual Link）之分，我们平时见到的都是单通道版的，双通道版的成本很高

**HDMI**: 高清输入，最高数据传输速度为5Gbps。 相对于 D-sub 与 DVI 仅能传送影像数据,HDMI 可以同时传送影像与声音,因此 被广泛的使用于电视屏幕中!电脑屏幕目前也经常都有支持 HDMI 格式

Display port:与 HDMI 相似的,可以同时传输声音与影像. 目前也是电视机屏幕的主流连接接口喔!

1. 硬盘依据台式机与笔记本电脑而有分为3.5英寸及2.5英寸的大小
2. 磁盘的最小物理储存单位,称之为扇区 (**sector**),那同一个同心圆的扇区 组合成的圆就是所谓的磁道(**track**)。 由于磁盘里面可能会有多个盘片,因此在所有盘片上 面的同一个磁道可以组合成所谓的柱面 (**cylinder**) 。通常数据的读写会由外圈开始往内写的喔!这是默认值啊!
3. 传统磁盘接口包括有 SATA, SAS, IDE 与 SCSI 等等。 若考虑外接式磁盘,那就还 包括了 USB, eSATA 等等接口。
   1. SATA接口：SATA 传输接口传输时,通过的数据演算法的关系,当传输 10 位编码时,仅有 8 位为数 据,其余 2 位为检验之用
   2. SAS接口:
   3. USB接口: USB 3.0 虽然速 度很快,但如果你去市面上面买 USB 的传统磁盘或闪存盘, 其实他的读写速度还是差不多在 100MBytes/s 而已.
4. 文字编码系统
   1. 常用的英文编码表为ASCII系统,这个编码系统中, 每个符号(英文、数字或符号等)都会占 用1Bytes的记录, 因此总共会有28=256种变化
   2. big5码的中文字编码对于某些数据库系统来说是很有问题的,某些字码例如“许、盖、功”等 字, 由于这几个字的内部编码会被误判为单/双引号,在写入还不成问题,在读出数据的对照 表时, 常常就会变成乱码
   3. Unicode编码系统, 我们常常称 呼的UTF8或万国码的编码就是这个咚咚
5. OS Kernel,整部系统只有Kernel的话,我们就只能看着已经准备好运行(Ready)的电脑系统, 但无法操作他! 好像有点望梅止渴的那种感觉啦!这个时候就需要软件的帮忙了
   1. System Call, 既然我的硬件都是由核心管理,那么如果我想要开发软件的话,自然就得要去参考这个核心 的相关功能! 操作系统通常会提供一整组的开发接口给工程师来开发软件.为了保护核心,并且让程序设计师比较容易开发软件,因此**操作系统除了核心程序之外,通 常还会提供一整组开发接口, 那就是系统调用层**
   2. 电脑系统主要由硬件构成,然后核心程序主要在管理硬件,提供合理的电脑系统资源分配 (包括CPU资源、内存使用资源等等), 因此只要硬件不同(如x86架构与RISC架构的 CPU),核心就得要进行修改才行。 而由于核心只会进行电脑系统的资源分配,所以在上头 还需要有应用程序的提供,使用者才能够操作系统的
   3. 核心功能, 既然核心主要是在负责整个电脑系统相关的资源分配与管理,那我们知道其实整部电脑系统 最重要的就是CPU与内存
      1. 系统调用接口(System call interface) , 这是为了方便程序开发者可以轻易 的通过与核心的沟通,将硬件的资源进一步的利用, 于是需要有这个简易的接口来方便 程序开发者
      2. 程序管理(Process control) , 多任务环境 , 核心这个时候必须要能够控制这些工作,让CPU的资 源作有效的分配才行
      3. 内存管理(Memory management) , 控制整个系统的内存管理,这个内存控制是非常重 要的,因为系统所有的程序码与数据都必须要先存放在内存当中。 通常核心会提供虚拟 内存的功能,当内存不足时可以提供内存交换(swap)的功能
      4. 文件系统管理(Filesystem management) 文件系统的管理 ,例如数据的输入输出 (I/O)等等的工作啦!还有不同文件格式的支持啦等等
      5. 设备的驱动(Device drivers) 就如同上面提到的,硬件的管理是核心的主要工作之一, 当然 ,设备的驱动程序就是核心需要做的事情啦! 好在目前都有所谓的“可载入模 块”功能,可以将驱动程序编辑成模块,就不需要重新的编译核心
   4. 操作系统与驱动程序, 驱动程序可以说是操作系统里面相当重要的一环了!不过,硬件可是持续在进步当 中的! 包括主板、显卡、硬盘等等。那么比较晚推出的较新的硬件, 为了克服这个问题,操作系统 通常会提供一个开发接口给硬件开发商, 让他们可以根据这个接口设计可以驱动他们硬件 的“驱动程序”,如此一来,只要使用者安装驱动程序后, 自然就可以在他们的操作系统上面 驱动这块硬件。
6. 操作系统
   1. 操作系统必须要能够驱动硬件,如此应用程序才能够使用该硬件功能
   2. 一般来说,操作系统会提供开发接口,让开发商制作他们的驱动程序
   3. 要使用新硬件功能,必须要安装厂商提供的驱动程序才行
   4. 驱动程序是由厂商提供的,与操作系统开发者无关
7. 电脑的五大单元, 输入单元、输出单元、控制单元、算数逻辑单元、存储单元五大 部分。其中CPU占有控制、算术逻辑单元,存储单元又包含内存与辅助内存
8. CPU的频率, 外频指的是CPU与外部元件进行数据传输时的速度,倍频则是 CPU内部用来加速工作性能的一个倍数, 两者相乘才是CPU的频率速度
9. CPU每次能够处理的数据量称为字组大小(word size),字组大小依据CPU的设计而有 32位与64位。 我们现在所称的电脑是32或64位主要是依据这个 CPU解析的字组大小而 来的
10. 个人电脑的内存主要元件为动态随机存取内存(Dynamic Random Access Memory, DRAM), 至于CPU内部的第二层高速缓存则使用静态随机存取内存(Static Random Access Memory, SRAM)
11. 目前主流的外接卡接口大多为 PCIe 接口,且最新为 PCIe 3.0,单信道速度高达 1GBytes/s
12. 磁盘连接到主板的接口大多为 SATA 或 SAS,目前台式机主流为 SATA 3.0,理论极速可 达 600MBytes/s
13. 常见的文字编码为 ASCII,繁体中文编码主要有 Big5 及 UTF8 两种,目前主流为 UTF8
14. 虽然Linux仅是其核心与核心提供的工具,不过由于核心、核心工具与这些软件开发者提供的 软件的整合, 使得Linux成为一个更完整的、功能强大的操作系统
15. Linux的核心版本编号 :
    1. 3.10.0-123.el7.x86\_64
    2. 主版本.次版本.释出版本-修改版本
16. 所谓的“虚拟化”指的是:在一部实体主机上面仿真出多个逻辑上完全独立的硬件,这个 假的虚拟出来的硬件主机,可以用来安装一部逻辑上完全独立的操作系统! 因此,通过虚拟 化技术,你可以将一部实体主机安装多个同时运行的操作系统 (非多重开机),以达到将硬 件资源完整利用的效果
17. 端点设备, 那就是近年来很热门又流行的树莓派 (Raspberry Pi) 与 香蕉派 (Banana Pi)
18. Tarball/RPM/DPKG/YUM/APT等软件管理员的安装方式
19. 在Windows里面,程序有问题时,如果可能的话先将所有其它程序保存并结束,然后尝 试按救命三键 (Ctrl+Alt+Delete),将有问题的程序(不要选错了程序哦)“结束工作”, 看看能不能恢复系统。不要动不动就直接关机或reset
20. Linux对于计算机各元件/设备的分辨, 与大家惯用的 Windows系统完全不一样!因为,各个元件或设备在Linux下面都是“一个文件!”
21. RAM 内存是越大越好!事实上在Linux服务器中,内存的重要性比CPU还要高的多,建议至少也要512MB， 以上的内存容量较佳。老实说,目前 DDR3 的硬件环境中, 新 购系统动不动就是 4~16GB 的内存,真的是很够用了
22. 磁盘阵列(RAID)是利用硬件技术将数个硬盘整合成为一个大硬盘的方法,操作系 统只会看到最后被整合起来的大硬盘。 由于磁盘阵列是由多个硬盘组成,所以可以达成 速度性能、备份等任务
23. Network Interface Card 网卡是服务器上面最重要的元件之一了!目前的主板大多拥有内 置10/100/1000Mbps的超高速以太网卡。如果你的服务器是网络 I/O 行 为非常频繁的网站, 好一点的Intel/boradcom等公司的网卡应该是比较适合
24. 桌上型(Desktop)Linux系统/含X Window:
    1. 用途:Linux的练习机或办公室(Office)工作机。(一般我们会用到的环境)
    2. CPU:最好等级高一点,例如 Intel I5, I7 以上等级。
    3. RAM:一定要大于1GB比较好!否则容易有图形接口停顿的现象
    4. 网卡:普通的以太网卡就好了
    5. 显卡:使用256MB以上内存的显卡!(入门级的都这个容量以上了)
    6. 硬盘:越大越好,最好有60GB
25. 中型以上Linux服务器
    1. 用途:中小型企业/学校单位的FTP/mail/WWW等网络服务主机
    2. CPU:最好等级高一点,例如 I5, I7 以上的多核心系统
    3. RAM:最好能够大于1GB以上,大于4GB更好
    4. 网卡:知名的broadcom或Intel等厂牌,比较稳定性能较佳
    5. 显卡:如果有使用到图形功能,则一张64MB内存的显卡是需要的
    6. 硬盘:越大越好,如果可能的话,使用磁盘阵列,或者网络硬盘等等的系统架构, 能够具有更稳定安全的传输环境,更佳
    7. 建议企业用计算机不要自行组装,可购买商用服务器较佳, 因为商用服务器已经通 过制造商的散热、稳定性等测试,对于企业来说,会是一个比较好的选择
26. 各硬件设备在Linux中的文件名
    1. SATA接口的硬盘的文件名称即为/dev/sd[a-d] ,其中, 括号内的字母为a-d当中的任意一个, 亦即有/dev/sda, /dev/sdb, /dev/sdc, 及 /dev/sdd这四个文件的意思
    2. 打印机 是/dev/lp0
    3. 软盘 是/dev/fd0
    4. SCSI/SATA/USB硬 盘机 /dev/sd[a-p]
    5. USB闪存盘 /dev/sd[a-p] (与SATA相同)
    6. VirtI/O界面 /dev/vd[a-p] (用于虚拟机内)
    7. 软盘机 /dev/fd[0-7]
    8. 打印机 /dev/lp[0-2] (25针打印机) /dev/usb/lp[0-15] (USB 接口)
    9. 鼠标 /dev/input/mouse[0-15] (通用) /dev/psaux (PS/2界面) /dev/mouse (当前鼠标)
    10. CDROM/DVDROM /dev/scd[0-1] (通用) /dev/sr[0-1] (通用,CentOS 较常见) /dev/cdrom (当前 CDROM)
    11. 磁带机 /dev/ht0 (IDE 界面) /dev/st0 (SATA/SCSI界面) /dev/tape (当前磁带)
    12. IDE硬盘机 /dev/hd[a-d] (旧式系统才有)
27. 虚拟化的软件
    1. 系统是 windows 系列 的话,鸟哥个人推荐你使用 virtualbox 这个软件
    2. 如果你原本就用 Linux 系统,例如 Fedora/Ubuntu 等系列的话,那么建议你使用原本系统内就有的虚拟机管理员来处理即可
    3. 目前 Linux 系统大多使用 KVM 这个虚拟化软件就是了
28. 第一个扇区 512Bytes 会有这两个数据
    1. 主要开机记录区(Master Boot Record, MBR):可以安装开机管理程序的地方,有446 Bytes
    2. 分区表(partition table):记录整颗硬盘分区的状态,有64 Bytes
    3. 硬盘的容量＝主分区的容量＋扩展分区的容量
    4. 扩展分区的容量＝各个逻辑分区的容量之和
    5. 在MBR分区表中最多4个主分区或者3个主分区+1个扩展分区，也就是说扩展分区只能有一个，然后可以再细分为多个逻辑分区。
29. MBR 主要分区、延伸分区与逻辑分区的特性
    1. 主要分区与延伸分区最多可以有四笔(硬盘的限制)
    2. 延伸分区最多只能有一个(操作系统的限制)
    3. 逻辑分区是由延伸分区持续切割出来的分区
30. MBR 分区表除了上述的主分区、延伸分区、逻辑分区需要注意之外,由于每组分区表仅有 16Bytes 而已,因此可纪录的信息真的是相当有限的! 所以,在过去 MBR 分区表的限制中 经常可以发现如下的问题
    1. 操作系统无法抓取到 2.2T 以上的磁盘容量
    2. MBR 仅有一个区块,若被破坏后,经常无法或很难救援
    3. MBR 内的存放开机管理程序的区块仅 446Bytes,无法容纳较多的程序码
31. GUID partition table, GPT 磁盘分区表, GPT 将磁盘所有区块以此 LBA(默认为 512Bytes 喔!) 来规划, 而第一个 LBA 称为 LBA0 (从 0 开始编号)
    1. GPT 使用了 34 个 LBA 区块来纪录分区 信息!同时与过去 MBR 仅有一的区块,被干掉就死光光的情况不同, GPT 除了前面 34 个 LBA 之外,整个磁盘的最后 33 个 LBA 也拿来作为另一个备份
    2. fdisk 这个老牌的软件并不认识 GPT
    3. **类似 gdisk 或者是 parted 指令才 认识 GPT**
    4. grub2 以后才会认识 GPT
32. BIOS 搭配 MBR/GPT 的开机流程
    1. BIOS:开机主动执行的固件,会认识第一个可开机的设备
    2. MBR:第一个可开机设备的第一个扇区内的主要开机记录区块,内含开机管理程序
    3. 开机管理程序(bootloader):一支可读取核心文件来执行的软件
    4. 核心文件:开始操作系统的功能...
33. boot loader的主要任务有下面这些项目
    1. 提供菜单:使用者可以选择不同的开机项目,这也是多重开机的重要功能
    2. 载入核心文件:直接指向可开机的程序区段来开始操作系统
    3. 转交其他loader:将开机管理功能转交给其他loader负责
34. 假设MBR内安装的是可同时认识Windows/Linux操作系统的开机管理程序, 那么整个流 程
    1. 每个分区都拥有自己的开机扇区(boot sector)
    2. 图中的系统盘为第一及第二分区
    3. 实际可开机的核心文件是放置到各分区内的!
    4. loader只会认识自己的系统盘内的可开机核心文件,以及其他loader而已
    5. loader可直接指向或者是间接将管理权转交给另一个管理程序
35. 如果要安装多重开机, 最好先安装Windows再安装 Linux ,为什么 ?
    1. Linux在安装的时候,你可以选择将开机管理程序安装在MBR或各别分区的开机扇区, 而 且Linux的loader可以手动设置菜单(就是上图的M1, M2...),所以你可以在Linux的boot loader里面加入Windows开机的选项
    2. Windows在安装的时候,他的安装程序会主动的覆盖掉MBR以及自己所在分区的开机扇 区,你没有选择的机会, 而且他没有让我们自己选择菜单的功能
    3. 如果先安装Linux再安装Windows的话,那MBR的开机管理程序就只会有Windows的项 目,而不会有Linux的项目 (因为原本在MBR内的Linux的开机管理程序就会被覆盖掉)。 那 需要重新安装Linux一次吗?当然不需要,你只要用尽各种方法来处理MBR的内容即可
36. UEFI BIOS 搭配 GPT 开机的流程
    1. GPT 可以提供到 64bit 的寻址 , 但是 BIOS 其实不懂 GPT 还得要通过 GPT 提供相容模式才能够读写这个磁盘设备 ~而且 BIOS 仅为 16 位的程序 因此就有了 UEFI (Unified Extensible Firmware Interface) 这个统一可延伸固件界 面的产生
    2. 一般来说,使用 UEFI 接口的主机,在开机的速度上要比 BIOS 来的 快上许多
37. 目录树结构 (directory tree) 。 所谓的目录树架构(directory tree)就是以根目录为主,然 后向下呈现分支状的目录结构的一种文件架构
38. 文件系统与目录树的关系(挂载): 所谓的“挂载”就是利用一个目录当成进入点,将磁盘分区的数据放置在该目录下; 也就是 说,进入该目录就可以读取该分区的意思。这个动作我们称为“挂载”,那个进入点的目录我们 称为“挂载点”。 由于整个Linux系统最重要的是根目录,因此根目录一定需要挂载到某个分区 的。 至于其他的目录则可依使用者自己的需求来给 挂载到不同的分区
    1. Linux是将光驱的数据放置到/media/cdrom里头去的
    2. Windows: 桌面\我的计算机\E:\我的文件
39. distributions安装时,挂载点与磁盘分区的规划
    1. 强烈建议使用“自订安装, Custom ”这个安装模式
    2. 在某些Linux distribution中,会将这个模式写的很厉害,叫做是“Expert, 专家模式”
    3. A:初次接触Linux:只要分区“ / ”及“swap”即可:
    4. B:建议分区的方法:预留一个备用的剩余磁盘容量!
    5. 对于首次接触Linux的朋友们,鸟哥通常不建议使用各个distribution所提供默认的Server 安装方式, 选择Server的时候,请“确定”你的硬盘数据是不再需要!因为Linux会自 动的把你的硬盘里面旧有的数据全部杀掉!
40. 如果是将 Linux定位在服务器上面的话,那么Red Hat Enterprise Linux及SuSE Enterprise Linux应该是 很不错的选择
41. 你所下载的文件扩展名是.iso,这就是所谓的image文件(镜像文件)
    1. 这种image文件 是由光盘直接烧录成文件的, 文件非常的大,建议你不要使用浏览器(IE/Firefox..)来下 载,可以使用FTP用户端程序来下载,例如Filezilla
    2. 这种镜像文件可不能以数据格式烧录成为光盘/DVD的!你必须要使用烧录程序的功 能, 将他<u>以“镜像文件格式”烧录成为光盘或DVD才行</u>!切记不要使用烧录数据文件格 式来烧录喔
42. NAT(达成IP分享器的功能)
    1. 在这种环境中,由于Linux作为一个内/外分离的实体,因此网络流量会比较大一点。 此时 Linux主机的网卡就需要比较好些的配备。其他的CPU、RAM、硬盘等等的影响就小很多
    2. Linux NAT还可以额外的加装很多分析软件, 可以用来分析用户端的连线,或者是用来控制带宽与流量,达到更公平的带宽使用
43. SAMBA(加入Windows网络上的芳邻)
44. Mail(邮件服务器) 。在mail server上面,重要的也 是硬盘容量与网卡速度,在此情境中,也可以将/var目录独立出来,并加大容量
45. Web(WWW服务器) CentOS使用的是Apache这套软件来达成WWW网站的功能,在WWW服务器上面,如果你还 有提供数据库系统的话, 那么CPU的等级就不能太低,而最重要的则是RAM了!要增加 WWW服务器的性能,通常提升RAM是一个不错的考虑
46. DHCP(提供用户端自动取得IP的功能)
47. FTP: FTP传输再怎么地下化也是很容易被捉到的
48. 我的Linux主机因为跳电的关系,造成不正常的关机,结果导 致无法开机,这该如何是好?, 幸运一点的可以使用fsck来解决硬盘的问题,麻烦一点 的可能还需要重新安装Linux呢
49. 在默认的CentOS环境中,下面的 目录是比较符合容量大且(或)读写频繁的目录
    1. /boot
    2. /
    3. /home
    4. /var
    5. Swap
50. 提供Linux的PC丛集(Cluster)计算机群
    1. 主机硬件配备
       1. 利用两部多核系统处理器 (一部 20核 40绪,一部 12核 24绪),搭配 10G 网卡组 合而成
       2. 使用内置的显卡
       3. 运算用主机仅一颗磁盘,储存用主机提供 8 颗 2TB 磁盘组成的磁盘阵列
       4. 一部 128GB 内存,一部 96GB 内存
    2. 硬盘分区:
       1. 运算主机方面,整颗磁盘仅分 /boot, / 及 swap 而已
       2. 储存主机方面,磁盘阵列分成两颗磁盘,一颗 100G 给系统用,一颗 12T 给数据 用。系统磁盘用的分区为 /boot, /, /home, /tmp, /var 等分区, 数据磁盘全部容量规划 在同一个分区而已。
       3. 安装最新的 CentOS 7.x 版
51. 开机的流程由:BIOS-->MBR-->-->boot loader-->核心文件
52. boot loader的功能主要有:提供菜单、载入核心、转交控制权给其他loader
53. boot loader可以安装的地点有两个,分别是 MBR 与 boot sector
54. 因为 DVD 实在太慢了,所以,比较聪明的朋友或许会将下载的镜像文件通过类 似 dd 或者是其他烧录软件, 直接烧录到 U盘上面,然后在 BIOS 里面调整成为可携式设备优 先开机的模式,这样就可以使用速度较快的 USB 开机来安装 Linux 了
    1. # 假设你的 USB 设备为 /dev/sdc ,而 ISO 文件名为 centos7.iso 的话:
    2. [root@study ~]# dd if=centos7.iso of=/dev/sdc
55. 如果想要强制使用 GPT 分区表的话
    1. 使用方向键,将图3.2.8的光标移动到“InstallCentOS7”的项目中
    2. 按下键盘的[Tab]按钮,让光标跑到画面最下方等待输入额外的核心参数
    3. 在出现的画面中,输入如下画面的数据(注意,各个项目要有空格,最后一个是光标本   
       身而非底线) 其实重点就是输入“ **inst.gpt** ”这个关键字
56. ext2/ext3/ext4:Linux早期适用的文件系统类型。由于ext3/ext4文件系统多了日志的记 录, 对于系统的复原比较快速。不过由于磁盘容量越来越大,ext 家族似乎有点挡不住了 ~所以除非你有特殊的设置需求,否则近来比较少使用 ext4 项目了!
57. swap:就是磁盘仿真成为内存,由于swap并不会使用到目录树的挂载,所以用swap就 不需要指定挂载点喔
58. BIOS Boot:就是 GPT 分区表可能会使用到的项目,若你使用 MBR 分区,那就不需要 这个项目了
59. xfs:这个是目前 CentOS 默认的文件系统,最早是由大型服务器所开发出来的! 他对于 大容量的磁盘管理非常好,而且格式化的时候速度相当快,很适合当今动不动就是好几 个 TB 的磁盘的环境喔!因此我们主要用这玩意儿
60. vfat:同时被Linux与Windows所支持的文件系统类型。如果你的主机硬盘内同时存在 Windows与Linux操作系统,为了数据的交换, 确实可以创建一个vfat的文件系统喔
61. 现在的内存都够大了,swap 虽然最好还是保持存 在比较好,不过也不需要太大啦!大约 1~2GB 就好了
62. 点选“系统”下的“KDUMP”项目,这个项目主要在处理,当 Linux 系 统因为核心问题导致的死机事件时, 会将该死机事件的内存内数据储存出来的一项特色!不 过,这个特色似乎比较偏向核心开发者在除错之用~如果你有需要的话,也可以启动它
63. How to check hostname in Linux: hostname
64. For MAC: choose Apple menu > System Preferences, then click Sharing. Computer name
65. CentOS7
    1. 包括 root 的密码等等,通通都会被纪 录到 /root/anaconda-ks.cfg 这个文件内
66. 内存压力测试:memtest86, “烧机”的任务 当你组装了一部新的个人计算 机,想要测试这部主机是否稳定时, 就在这部主机上面运行一些比较耗系统资源的程序,让 系统在高负载的情况下去运行一阵子(可能是一天), 去测试稳定性的一种情况,就称为“烧 机”
67. 重新开机,然后依序选 择“Troubleshooting”、 “Run a memory test”的项目
68. 安装笔记本电脑或其他类PC计算机的参数
    1. 只要在安装的时候,告诉安装程序的linux核心不要载入一些特殊功能即可
    2. 最 常使用的方法就是,在使用DVD开机时,选择“”然后按下 [tab] 按键后,加入下面这些选项
    3. **nofb apm=off acpi=off pci=noacpi**
    4. apm(Advanced Power Management)是早期的电源管理模块
    5. acpi(Advanced Configuration and Power Interface)则是近期的电源管理模块
    6. nofb则是取消显卡上面的缓冲内存侦测。因为笔记本电脑的显卡常常是整合型的, Linux安装 程序本身可能就不是很能够侦测到该显卡模块。此时加入nofb将可能使得你的安装过程顺利 一些
69. 安装过程进入分区后,请以“自订的分区模式”来处理自己规划的分区方式
70. 在安装的过程中,可以创建逻辑卷轴管理员 (LVM)
71. 一般要求swap应该要是1.5~2倍的实体内存量,但即使没有swap依旧能够安装与运行 Linux操作系统
72. CentOS 7默认使用 xfs 作为文件系统
73. 没有连上Internet时,可尝试关闭防火墙,但SELinux最好选择“强制”状态
74. 在 Linux 下面,隐藏文件并不是什么特殊的权限, 单纯是因为文件名命名的处 理方式来搞定的
75. 通常我们也称文字模式为终端机接口, terminal 或 console喔!Linux默认的情 况下会提供六个Terminal来让使用者登陆, 切换的方式为使用:[Ctrl] + [Alt] + [F1]~[F6]的 组合按钮
76. 下达指令
    1. command [-options] parameter1 parameter2 ...
    2. 指令 选项 参数(1) 参数(2)
    3. 不论空几格shell都视为一格。所以空格是很重要的特殊字符!
    4. 按下[Enter]按键后,该指令就立即执行。[Enter]按键代表着一行指令的开始启动
    5. 指令太长的时候,可以使用反斜线(\)来跳脱[Enter]符号,使指令连续到下一行。注 意!反斜线后就立刻接特殊字符,才能跳脱
    6. **ls -al ~** is different with **ls -al**
    7. 显示日历的指令: **cal [month] [year**]
    8. 显示目前支持的语系： **Locale**
    9. 简单好用的计算机: bc
    10. 指令求助说明 --help or man \*\*
77. [Tab]按键
    1. [Tab] 接在一串指令的第一个字的后面,则为“命令补全”
    2. [Tab] 接在一串指令的第二个字以后时,则为“文件补齐”!
    3. 若安装 bash-completion 软件,则在某些指令后面使用 [tab] 按键时,可以进行“选项/参 数的补齐功能!
78. [Ctrl]-d 按键 ,键盘输入结 束(End Of File, EOF 或 End Of Input) 另外,他也可以用来取代exit的输入
79. [shift]+{[PageUP]|[Page Down]}按键,可以使用 [Shift]+[Page Up] 来往前翻页,也能够使用 [Shift]+[Page Down] 来往后翻页
80. 在查询数据的后面的数字是有意义的:
    1. **1: 使用者在shell环境中可以操作的指令或可可执行文件**
    2. 2: 系统核心可调用的函数与工具等
    3. 3: 一些常用的函数(function)与函数库(library),大部分为C的函数库(libc)
    4. 4: 设备文件的说明,通常在/dev下的文件
    5. **5: 配置文件或者是某些文件的格式**
    6. 6: 游戏(games)
    7. 7: 惯例与协定等,例如Linux文件系统、网络协定、ASCII code等等的说明
    8. **8: 系统管理员可用的管理指令**
    9. 9: 跟kernel有关的文件
81. 搜寻特定指令/文件的man page说明文档
    1. man -f man
    2. man -k man
    3. 两个特殊指令要能使用,必须要有创建 whatis 数据库才行 : mandb
    4. whatis [指令或者是数据] &lt;==相当于 man -f [指令或者是数据]
    5. apropos [指令或者是数据] &lt;==相当于 man -k [指令或者是数据]
    6. info的说明文档将内容分成多个node,并且每个node都有定位与链接 : **info info**
    7. man, info, /usr/share/doc/
    8. 在终端机模式中,如果你知道某个指令,但却忘记了相关选项与参数,请先善用 --help 的功能来查询相关信息
    9. 当有任何你不知道的指令或文件格式这种玩意儿,但是你想要了解他,请赶快使用man 或者是info来查询!
    10. 而如果你想要架设一些其他的服务,或想要利用一整组软件来达成某项功能时,请赶快 到/usr/share/doc 下面查一查有没有该服务的说明文档喔!
    11. 另外,再次的强调,因为Linux毕竟是外国人发明的,所以中文文件确实是比较少的! 但 是不要害怕,拿本英文字典在身边吧!随时查阅!不要害怕英文喔!
    12. 指数符号(^)代 表的是键盘的[Ctrl]按键
82. 要关机时需要注意下面几 件事
    1. 如果要看目前有谁在线上,可以下达“who”
    2. 如果要看 网络的连线状态,可以下达 “ netstat -a ”
    3. 要看背景执行的程序可以执行“ ps -aux ”这个指令
    4. 将数据同步写入硬盘中的指令: sync
    5. 惯用的关机指令: shutdown
    6. 重新开机,关机: reboot, halt, poweroff
    7. 虽然目前的 shutdown/reboot/halt 等等指令均已经在关机前进行了 sync 这个工具的调用,不 过,多做几次总是比较放心点
    8. 实际使用管理工具 systemctl 关机
       1. init 这个指令可以切换不同 的执行等级~ 执行等级共有 0~6 七个,其中 0 就是关机、6 就是重新开机等等
       2. 面谈到的 halt, poweroff, reboot, shutdown 等等,其实都是调用这个 systemctl 指令
       3. systemctl reboot # 系统重新开机
       4. systemctl poweroff # 系统关机
    9. sync; sync; sync; reboot
    10. Send a warning message: sudo shutdown -k now 'I will shutdown after 10 mins'
83. info page可将一份说明文档拆成多个节点(node)显示,并具有类似超链接的功能,增 加易读性
84. 系统需正确的关机比较不容易损坏,可使用shutdown, poweroff等指令关机
85. 发生乱码可能是因为语系的问题所致。 可以利用 export LANG=en\_US.utf8 或者是 export LC\_ALL=en\_US.utf8 等设置来修订这个问题
86. Linux一般将文件可存取的身份分为三个类 别,分别是 owner/group/others,且三种身份各有 read/write/execute 等权限
87. Linux 使用者身份与群组记录的文件
    1. 默认的情况下,所有的系统上的帐号与一般身份使用者,还有那个 root的相关信息, 都是记录在/etc/passwd这个文件内
    2. 个人的密码则是记录 在/etc/shadow这个文件下
    3. Linux所有的群组名称都纪录在/etc/group内
88. 文件属性
    1. 第一栏代表这个文件的类型与权限(permission)
       1. **第一个字符代表这个文件是“目录、文件或链接文件等等**”
          1. 当为[ d ]则是目录
          2. 当为[ - ]则是文件
             1. 纯文本文件(ASCII)
             2. 二进制档(binary)
             3. 数据格式文件(data)
          3. 若是[ l ]则表示为链接文件(link file);
          4. 若是[ b ]则表示为设备文件里面的可供储存的周边设备(可随机存取设备);
             1. 区块(block)设备文件
             2. 字符(character)设备文件
          5. 若是[ c ]则表示为设备文件里面的序列埠设备,例如键盘、鼠标(一次性读取设 备)
          6. 若是[ s ]则表示数据接口文件(sockets) 最常在/run或/tmp这些个目录中看到这种文件类型
          7. 若是[ p ]则表示FIFO也是一种特殊的文件类型,他主要的目的在解决多个 程序同时存取一个文件所造成的错误问题
       2. 接下来的字符中,以三个为一组,且均为“rwx” 的三个参数的组合
          1. 第一组为“文件拥有者可具备的权限”
          2. 第二组为“加入此群组之帐号的权限”
          3. 第三组为“非本人且没有加入本群组之其他帐号的权限”
          4. [ r ]代表可读 (read)、[ w ]代表可写(write)、[ x ]代表可执行(execute)
          5. ls -l --full-time
          6. 至于other的权限中[r--]虽然有r ,但是由于没有x的权限,因此others的使用者,并不能进 入此目录
    2. 第二栏表示有多少文件名链接到此节点(i-node)
    3. 第三栏表示这个文件(或目录)的“拥有者帐号”
    4. 第四栏表示这个文件的所属群组
    5. 第五栏为这个文件的容量大小,默认单位为Bytes
    6. 第六栏为这个文件的创建日期或者是最近的修改日期
    7. 第七栏为这个文件的文件名, 如果文件名之前多一个“ . ”,则代表这个文件为“隐藏 文件”
89. 如何改变文件属性与权限
    1. chgrp :改变文件所属群组
    2. chown :改变文件拥有者
    3. chmod :改变文件的权限, SUID, SGID, SBIT等等的特性
    4. 改变所属群组, chgrp
    5. 符号类型改变文件权限
       1. (1)user (2)group (3)others三种身份啦!那么我们就可以借由u, g, o来代表三种 身份的权限
       2. chmod a+x filename
       3. r (read):可读取此一文件的实际内容,如读取文本文件的文字内容等;
       4. w (write):可以编辑、新增或者是修改该文件的内容(**但不含删除该文件**);
       5. x (eXecute):该文件具有可以被系统执行的权限
90. 权限对目录的重要性
    1. r (read contents in directory) 表示你 可以查询该目录下的文件名数据
    2. w (modify contents of directory
       1. 创建新的文件与目录
       2. 删除已经存在的文件与目录(不论该文件的权限为何!)
       3. 将已存在的文件或目录进行更名
       4. 搬移该目录内的文件、目录位置。 总之,目录的w权限就与该目录下面的文件名异 动有关就对了啦
    3. x (access directory) 目录的x代表的是使用者能否进入该目录成为工作目录的用途
    4. 创建空文件可用**touch**来 处理
    5. 通常要开放的目录, 至少会具备 rx 这两个权限
91. Linux文件扩展名
    1. \*.sh : 脚本或批处理文件 (scripts),因为批处理文件为使用shell写成的,所以扩展名 就编成 .sh
    2. Z, .tar, .tar.gz, .zip, \*.tgz: 经过打包的压缩文件。这是因为压缩软件分别为 gunzip, tar 等等的,由于不同的压缩软件,而取其相关的扩展名
    3. .html, .php:网页相关文件,分别代表 HTML 语法与 PHP 语法的网页文件
       1. .html 的 文件可使用网页浏览器来直接打开
       2. .php 的文件, 则可以通过 client 端的浏览器来 server 端浏览,以得到运算后的网页结果
92. 从网络上传送到你的 Linux系统中,文件的属性与权限确实是会被改变的
93. Filesystem Hierarchy Standard (FHS)标准
    1. / (root, 根目录):与开机系统有关
    2. /usr (unix software resource):与软件安装/执行有关
    3. /var (variable):与系统运行过程有关
94. FHS定义出根目录(/)下面应该要有下面这些次目录的存在才好
    1. /bin 系统有很多放置可执行文件的目录,但/bin比较特殊。因为/bin放置的是在 单人维护模式下还能够被操作的指令。 在/bin下面的指令可以被root与一般 帐号所使用,主要有:cat, chmod, chown, date, mv, mkdir, cp, bash等等常 用的指令。
    2. /boot 这个目录主要在放置开机会使用到的文件,包括Linux核心文件以及开机菜 单与开机所需配置文件等等。 Linux kernel常用的文件名为:vmlinuz,如果 使用的是grub2这个开机管理程序, 则还会存在/boot/grub2/这个目录喔!
    3. /dev 在Linux系统上,任何设备与周边设备都是以文件的型态存在于这个目录当 中的。 你只要通过存取这个目录下面的某个文件,就等于存取某个设备 ~ 比要重要的文件有/dev/null, /dev/zero, /dev/tty, /dev/loop, /dev/sd等等
    4. /etc 系统主要的配置文件几乎都放置在这个目录内,例如人员的帐号密码档、 各种服务的启始档等等。一般来说,这个目录下的各文件属性是可以让一般 使用者查阅的, 但是只有root有权力修改。FHS建议不要放置可可执行文件 (binary)在这个目录中喔。比较重要的文件有: /etc/modprobe.d/, /etc/passwd, /etc/fstab, /etc/issue 等等。另外 FHS 还规范几个重要的目录 最好要存在 /etc/ 目录下喔:/etc/opt(必要):这个目录在放置第三方协力 软件 /opt 的相关配置文件 /etc/X11/(建议):与 X Window 有关的各种配 置文件都在这里,尤其是 xorg.conf 这个 X Server 的配置文件。 /etc/sgml/ (建议):与 SGML 格式有关的各项配置文件 /etc/xml/(建议):与 XML 格式有关的各项配置文件
    5. /lib 系统的函数库非常的多,而/lib放置的则是在开机时会用到的函数库, 以及 在/bin或/sbin下面的指令会调用的函数库而已。 什么是函数库呢?你可以将 他想成是“外挂”,某些指令必须要有这些“外挂”才能够顺利完成程序的执行 之意。 另外 FSH 还要求下面的目录必须要存在:/lib/modules/:这个目录 主要放置可抽换式的核心相关模块(驱动程序)
    6. /media media是“媒体”的英文,顾名思义,这个/media下面放置的就是可移除的设 备啦! 包括软盘、光盘、DVD等等设备都暂时挂载于此。常见的文件名 有:/media/floppy, /media/cdrom等等
    7. /mnt 如果你想要暂时挂载某些额外的设备,一般建议你可以放置到这个目录中。 在古早时候,这个目录的用途与/media相同啦!只是有了/media之后,这个 目录就用来暂时挂载用了
    8. /opt 这个是给第三方协力软件放置的目录。什么是第三方协力软件啊? 举例来 说,KDE这个桌面管理系统是一个独立的计划,不过他可以安装到Linux系 统中,因此KDE的软件就建议放置到此目录下了。 另外,如果你想要自行 安装额外的软件(非原本的distribution提供的),那么也能够将你的软件安 装到这里来。 不过,以前的Linux系统中,我们还是习惯放置在/usr/local目 录下呢
    9. /run 早期的 FHS 规定系统开机后所产生的各项信息应该要放置到 /var/run 目录 下,新版的 FHS 则规范到 /run 下面。 由于 /run 可以使用内存来仿真,因 此性能上会好很多
    10. /sbin Linux有非常多指令是用来设置系统环境的,这些指令只有root才能够利用 来“设置”系统,其他使用者最多只能用来“查询”而已。 放在/sbin下面的为开 机过程中所需要的,里面包括了开机、修复、还原系统所需要的指令。 至于某些服务器软件程序,一般则放置到/usr/sbin/当中。至于本机自行安装的 软件所产生的系统可执行文件(system binary), 则放置到/usr/local/sbin/ 当中了。常见的指令包括:fdisk, fsck, ifconfig, mkfs等等
    11. /srv srv可以视为“service”的缩写,是一些网络服务启动之后,这些服务所需要 取用的数据目录。 常见的服务例如WWW, FTP等等。举例来说,WWW服 务器需要的网页数据就可以放置在/srv/www/里面。 不过,系统的服务数据 如果尚未要提供给网际网络任何人浏览的话,默认还是建议放置到 /var/lib 下面即可
    12. /tmp 这是让一般使用者或者是正在执行的程序暂时放置文件的地方。 这个目录 是任何人都能够存取的,所以你需要定期的清理一下。当然,重要数据不可 放置在此目录啊! 因为FHS甚至建议在开机时,应该要将/tmp下的数据都 删除
    13. /usr 是Unix Software Resource的缩写, 也就 是“Unix操作系统软件资源”所放置的目录,而不是使用者的数据啦
    14. /var /var目录主要针对常态性变动的文件,包括高速缓存(cache)、登录文件 (log file)以及某些软件运行所产生的文件, 包括程序文件(lock file, run file),或者例如 MySQL数据库的文件等等
    15. /home 这是系统默认的使用者主文件夹(home directory)。在你新增一个一般使 用者帐号时, 默认的使用者主文件夹都会规范到这里来。比较重要的是, 主文件夹有两种代号喔:~:代表目前这个使用者的主文件夹 ~dmtsai :则 代表 dmtsai 的主文件夹
    16. /lib<qual> 用来存放与 /lib 不同的格式的二进制函数库,例如支持 64 位的 /lib64 函数 库等
    17. /root 系统管理员(root)的主文件夹。之所以放在这里,是因为如果进入单人维 护模式而仅挂载根目录时, 该目录就能够拥有root的主文件夹,所以我们会 希望root的主文件夹与根目录放置在同一个分区中
    18. /lost+found 这个目录是使用标准的ext2/ext3/ext4文件系统格式才会产生的一个目录, 目的在于当文件系统发生错误时, 将一些遗失的片段放置到这个目录下。 不过如果使用的是 xfs 文件系统的话,就不会存在这个目录了
    19. /proc 这个目录本身是一个“虚拟文件系统(virtual filesystem)”喔!他放置的数 据都是在内存当中, 例如系统核心、行程信息(process)、周边设备的 状态及网络状态等等。因为这个目录下的数据都是在内存当中, 所以本身 不占任何硬盘空间啊!比较重要的文件例如:/proc/cpuinfo, /proc/dma, /proc/interrupts, /proc/ioports, /proc/net/\* 等等
    20. /sys 这个目录其实跟/proc非常类似,也是一个虚拟的文件系统,主要也是记录 核心与系统硬件信息较相关的信息。 包括目前已载入的核心模块与核心侦 测到的硬件设备信息等等。这个目录同样不占硬盘容量喔
    21. /usr/bin/ 所有一般用户能够使用的指令都放在这里!目前新的 CentOS 7 已经将 全部的使用者指令放置于此,而使用链接文件的方式将 /bin 链接至此! 也就是说, /usr/bin 与 /bin 是一模一样了!另外,FHS 要求在此目录 下不应该有子目录!
    22. /usr/lib/ 基本上,与 /lib 功能相同,所以 /lib 就是链接到此目录中的!
    23. /usr/local/ 系统管理员在本机自行安装自己下载的软件(非distribution默认提供 者),建议安装到此目录, 这样会比较便于管理。举例来说,你的 distribution提供的软件较旧,你想安装较新的软件但又不想移除旧版, 此时你可以将新版软件安装于/usr/local/目录下,可与原先的旧版软件 有分别啦! 你可以自行到/usr/local去看看,该目录下也是具有bin, etc, include, lib...的次目录喔
    24. /usr/sbin/ 非系统正常运行所需要的系统指令。最常见的就是某些网络服务器软件 的服务指令(daemon) !不过基本功能与 /sbin 也差不多, 因此目 前 /sbin 就是链接到此目录中的
    25. /usr/share/ 主要放置只读架构的数据文件,当然也包括共享文件。在这个目录下放 置的数据几乎是不分硬件架构均可读取的数据, 因为几乎都是文字文 件嘛!在此目录下常见的还有这些次目录:/usr/share/man:线上说明 文档 /usr/share/doc:软件杂项的文件说明 /usr/share/zoneinfo:与时 区有关的时区文件
    26. /usr/games/ 与游戏比较相关的数据放置处
    27. /usr/include/ c/c++等程序语言的文件开始(header)与包含档(include)放置处, 当我们以tarball方式 (\*.tar.gz 的方式安装软件)安装某些数据时,会 使用到里头的许多包含档喔!
    28. /usr/libexec/ 某些不被一般使用者惯用的可执行文件或脚本(script)等等,都会放 置在此目录中。例如大部分的 X 窗口下面的操作指令, 很多都是放在 此目录下的
    29. /usr/lib<qual>/ 与 /lib<qual>/功能相同,因此目前 /lib<qual> 就是链接到此目录中
    30. /usr/src/ 一般源代码建议放置到这里,src有source的意思。至于核心源代码则 建议放置到/usr/src/linux/目录下
    31. /var/cache/ 应用程序本身运行过程中会产生的一些暂存盘;
    32. /var/lib/ 程序本身执行的过程中,需要使用到的数据文件放置的目录。在此目录下 各自的软件应该要有各自的目录。 举例来说,MySQL的数据库放置 到/var/lib/mysql/而rpm的数据库则放到/var/lib/rpm去!
    33. /var/lock/ 某些设备或者是文件资源一次只能被一个应用程序所使用,如果同时有两 个程序使用该设备时, 就可能产生一些错误的状况,因此就得要将该设备 上锁(lock),以确保该设备只会给单一软件所使用。 举例来说,烧录机 正在烧录一块光盘,你想一下,会不会有两个人同时在使用一个烧录机烧 片? 如果两个人同时烧录,那片子写入的是谁的数据?所以当第一个人在 烧录时该烧录机就会被上锁, 第二个人就得要该设备被解除锁定(就是前 一个人用完了)才能够继续使用 。目前此目录也已经挪到 /run/lock 中
    34. /var/log/ 重要到不行!这是登录文件放置的目录!里面比较重要的文件 如/var/log/messages, /var/log/wtmp(记录登陆者的信息)等
    35. /var/mail/ 放置个人电子邮件信箱的目录,不过这个目录也被放置到/var/spool/mail/ 目录中! 通常这两个目录是互为链接文件啦!
    36. /var/run/ 某些程序或者是服务启动后,会将他们的PID放置在这个目录下喔!至于 PID的意义我们会在后续章节提到的。 与 /run 相同,这个目录链接到 /run 去了!
    37. /var/spool/ 这个目录通常放置一些伫列数据,所谓的“伫列”就是排队等待其他程序使 用的数据啦! 这些数据被使用后通常都会被删除。举例来说,系统收到新 信会放置到/var/spool/mail/中, 但使用者收下该信件后该封信原则上就会 被删除。信件如果暂时寄不出去会被放到/var/spool/mqueue/中, 等到被 送出后就被删除。如果是工作调度数据(crontab),就会被放置 到/var/spool/cron/目录中!
95. 通过uname 检查linux核心与操作系统的位数
    1. uname -r # 查看核心版本
    2. uname -m # 查看操作系统的位版本
    3. 假设你的 CentOS 7 确实有网络可以使用的情况下 （使用root的身份）
    4. lsb\_release -a
96. 若需要root的权限时,可以使用 su - 这个指令来切换身份。处理完毕则使用 exit 离开 su 的指令环
97. 更改文件的群组支持可用chgrp,修改文件的拥有者可用chown,修改文件的权限可用 chmod
98. chmod修改权限的方法有两种,分别是符号法与数字法,数字法中r,w,x分数为4,2,1
    1. r:可读取此一文件的实际内容,如读取文本文件的文字内容等
    2. w:可以编辑、新增或者是修改该文件的内容(但不含删除该文件)
    3. x:该文件具有可以被系统执行的权限
99. 要开放目录给任何人浏览时,应该至少也要给 r及x的权限,但w权限不可随便给 。能否读取到某个文件内容,跟该文件所在的目录权限也有关系 (目录至少需要有 x 的权 限)。对目录来说,权限的性能为
    1. r (read contents in directory)
    2. w (modify contents of directory)
    3. x (access directory)
100. FHS订定出来的四种目录特色为:shareable, unshareable, static, variable等四类; FHS所定义的三层主目录为:/, /var, /usr三层而已
101. 当一个一般文件权限为 -rwxrwxrwx 则表示这个文件的意义为?任何人皆可读取、修改或 编辑、可以执行,**但不一定能删除**
102. 下面的目录与主要放置什么数据
     1. /etc/:几乎系统的所有设置文件均在此,尤其 passwd,shadow
     2. /boot:开机配置文件,也是默认摆放核心 vmlinuz 的地方
     3. /usr/bin, /bin:一般可执行文件摆放的地方
     4. /usr/sbin, /sbin:系统管理员常用指令集
     5. /dev:摆放所有系统设备文件的目录
     6. /var/log:摆放系统登录文件的地方

/run:CentOS 7 以后才有,将经常变动的项目(每次开机都不同,如程序的PID)

移动到内存暂存,所以 /run 并不占实际磁盘容量

1. 目录的相关操作
   1. . 代表此层目录
   2. 代表上一层目录
   3. 代表“目前使用者身份”所在的主文件夹
   4. ~account 代表 account 这个使用者的主文件夹(account是个帐号名称)
   5. cd是Change Directory的缩写,这是用来变换工作目录的指令
   6. 如果 仅输入 cd 时,代表的就是“ cd ~ ”的意思喔~ 亦即是会回到自己的主文件夹
   7. pwd是Print Working Directory的缩写
   8. pwd -P 可以让我们取得正确的目录名称,而不是以链接文件的路 径来显示的
   9. mkdir (make directory) , 我们可以利用 -m 来强制给 一个新的目录相关的权限
   10. mkdir -p 依 序的创建起目录!并且, 如果该目录本来就已经存在时,系统也不会显示错误讯息喔 .不过鸟哥不建议常用-p这个选项,因为担心如果你打错字,那么目录名称就会 变的乱七八糟的
   11. 如果要将所有目录 下的东西都杀掉呢?! 这个时候就必须使用“ rm -r test ” !不过,还是使用 rmdir 比较不危险!你也可以尝试以 -p 的选项加入,来删除上层的目录喔
   12. PATH(一定是大写)这个变量的内容是由一堆目录所组成的,每个目录中间用冒号(:)来隔 开, 每个目录是有“顺序”之分的
   13. **cp -u 这个 -u 的特性,是在目标文件与来源文件有差异时,才会复制的**。所以,比较常被用于“备份”的工作当中
2. 关于可执行文件路径的变量: $PATH
   1. 不同身份使用者默认的PATH不同,默认能够随意执行的指令也不同(如root与 dmtsai)
   2. PATH是可以修改的
   3. 使用绝对路径或相对路径直接指定某个指令的文件名来执行,会比搜寻PATH来的正确
   4. 指令应该要放置到正确的目录下,执行才会比较方便
   5. 本目录(.)最好不要放到PATH当中
   6. 链接文件(link file) ; 实体链接(hard link) ; 符号链接(symbolic link)
3. cp 有种种的文件属性与权限的特性
   1. 是否需要完整的保留来源文件的信息?
   2. 来源文件是否为链接文件 (symbolic link file)
   3. 来源文件是否为特殊的文件,例如 FIFO, socket 等
   4. 来源文件是否为目录
4. 文件内容查阅
   1. cat 由第一行开始显示文件内容
   2. tac 从最后一行开始显示,可以看出 tac 是 cat 的倒着写!
   3. nl 显示的时候,顺道输出行号!
   4. more 一页一页的显示文件内容
   5. less 与 more 类似,但是比 more 更好的是,他可以往前翻页!
   6. head 只看头几行
   7. tail 只看尾巴几行
   8. od 以二进制的方式读取文件内容!
5. 直接检视文件内容
   1. cat 是 Concatenate (连续) 的简写
   2. tac (反向列示)
   3. nl (添加行号打印)
   4. more (一页一页翻动)
      1. 空白键 (space):代表向下翻一页;
      2. Enter :代表向下翻“一行”;
      3. /字串 :代表在这个显示的内容当中,向下搜寻“字串”这个关键字;
      4. :f :立刻显示出文件名以及目前显示的行数;
      5. q :代表立刻离开 more ,不再显示该文件内容。
      6. b 或 [ctrl]-b :代表往回翻页,不过这动作只对文件有用,对管线无用。
   5. less (一页一页翻动)
      1. 空白键 :向下翻动一页;
      2. [pagedown]:向下翻动一页;
      3. [pageup] :向上翻动一页;
      4. /字串 :向下搜寻“字串”的功能;
      5. ?字串 :向上搜寻“字串”的功能;
      6. n :重复前一个搜寻 (与 / 或 ? 有关!)
      7. N :反向的重复前一个搜寻 (与 / 或 ? 有关!)
      8. g :前进到这个数据的第一行去;
      9. G :前进到这个数据的最后一行去 (注意大小写);
      10. q :离开 less 这个程序;
   6. head (取出前面几行)
   7. tail (取出后面几行)
   8. 两个指令中间有个管线 (|) 的符号存在,这个管线的意思是:“前面的指令所输出的讯 息,请通过管线交由后续的指令继续使用”的意思
6. 非纯文本文件: od 。由 于可执行文件通常是 binary file ,使用上头提到的指令来读取他的内容时, 确实会产生类似 乱码的数据啊!那怎么办?没关系,我们可以利用 od 这个指令来读取喔
7. 修改文件时间或创建新文件: touch
   1. modification time (mtime): 当该文件的“内容数据”变更时,就会更新这个时间!内 容数据指的是文件的内容,而不是文件的属性或权限喔
   2. status time (ctime): 当该文件的“状态 (status)”改变时,就会更新这个时间,举 例来说,像是权限与属性被更改了,都会更新这个时间啊
   3. access time (atime): 当“该文件的内容被取用”时,就会更新这个读取时间 (access)。举例来说,我们使用 cat 去读取 /etc/man\_db.conf , 就会更新该文件的 atime 了
8. touch 这个指令最常被使用的情况是
   1. 创建一个空的文件;
   2. 将某个文件日期修订为目前 (mtime 与 atime)
9. 文件默认权限:umask, umask 就是指定 “目前使用者在创建文件或目录时 候的权限默认值"
   1. 一种可以直接输入 umask ,就可以看到数字体态的权限设置分数
   2. 一种 则是加入 -S (Symbolic) 这个选项,就会以符号类型的方式来显示出权限了
   3. 若使用者创建为“文件”则默认“没有可执行( x )权限”,亦即只有 rw 这两个项目,也就 是最大为 666 分,默认权限如下: -rw-rw-rw-
   4. 若使用者创建为“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,亦即为 777 分,默认权限如下: drwxrwxrwx
   5. 创建文件时:(-rw-rw-rw-) - (-----w--w-) ==> -rw-r--r--
   6. 创建目录时:(drwxrwxrwx) - (d----w--w-) ==> drwxr-xr-x
   7. 这个文件的权限 为:“---------- 1 root root”,意思是这个文件仅有root可读且仅有root可以强制写入而已
10. 文件隐藏属性
    1. chattr (设置文件隐藏属性) , 由于这些属性是隐藏的性质,所以需 要以 lsattr 才能看到该属性. 其中,个人认为最重要的当属 +i 与 +a 这个属性了。+i 可以让 一个文件无法被更动,对于需要强烈的系统安全的人来说, 真是相当的重要的
    2. lsattr (显示文件隐藏属性) , 使用 chattr 设置后,可以利用 lsattr 来查阅隐藏的属性。不过, 这两个指令在使用上必须要 特别小心,否则会造成很大的困扰
11. 文件特殊权限: SUID, SGID, SBIT
    1. Set UID 当 s 这个标志出现在文件拥有者的 x 权限上时 此时就被称为 Set UID,简称为 SUID 的特殊权限
       1. SUID 权限仅对二进制程序(binary program)有效;
       2. 执行者对于该程序需要具有 x 的可执行权限;
       3. 本权限仅在执行该程序的过程中有效 (run-time);
       4. 执行者将具有该程序拥有者 (owner) 的权限。
    2. Set GID 当 s 标志在文件拥有者的 x 项目为 SUID,那 s 在群组的 x 时则称为 Set GID, SGID
       1. 与 SUID 不同的是,SGID 可以针对文件或目录来设置!如果是对文件来说, SGID 有如下的 功能:
       2. SGID 对二进制程序有用;
       3. 程序执行者对于该程序来说,需具备 x 的权限;
       4. 执行者在执行的过程中将会获得该程序群组的支持!
       5. 当一个目录设置了 SGID 的权限后,他将具有如下的功能:
       6. 使用者若对于此目录具有 r 与 x 的权限时,该使用者能够进入此目录;
       7. 使用者在此目录下的有效群组(effective group)将会变成该目录的群组;
       8. 用途:若使用者在此目录下具有 w 的权限(可以新建文件),则使用者所创建的新文 件,该新文件的群组与此目录的群组相同。
    3. Sticky Bit , SBIT 对于目录的作用 是:
       1. 当使用者对于此目录具有 w, x 权限,亦即具有写入的权限时;
       2. 当使用者在该目录下创建文件或目录时,仅有自己与 root 才有权力删除该文件
    4. SUID/SGID/SBIT 权限设置
       1. 4 为 SUID
       2. 2 为 SGID
       3. 1 为 SBIT
       4. 大写的 S, T 代表的就 是“空的”啦
       5. 你也可以通过符号法来处理喔!其中 SUID 为 u+s ,而 SGID 为 g+s , SBIT 则是 o+t
12. 观察文件类型:file 可以利用 file 这个指令 来检阅是属于 ASCII 或者是 data 文件,或者是 binary , 且其中有没有使用到动态函数库 (share library) 等等的信息
    1. file filename
13. 指令与文件的搜寻
    1. which (寻找“可执行文件”) 且 which 后面接的是“完整文件名”喔!若加上 -a 选项,则可以 列出所有的可以找到的同名可执行文件,而非仅显示第一个而已
    2. whereis 或者是 locate 两者就相当的快速, 并且没有实 际的搜寻硬盘内的文件系统状态,比较省时间
    3. whereis (由一些特定的目录中寻找文件文件名) 只找系统中某些特定目录下 面的文件而已 whereis 主要是针对 /bin /sbin 下面的可执行文件, 以及 /usr/share/man 下面的 man page 文 件,跟几个比较特定的目录来处理而已
    4. locate / updatedb 则是利用数据库来搜寻文件名 这个 locate 的使用更简单,直接在后面输入“文件的部分名称”后,就能够得到结果
       1. updatedb:根据 /etc/updatedb.conf 的设置去搜寻系统硬盘内的文件名,并更新 /var/lib/mlocate 内的数据库文件
       2. locate:依据 /var/lib/mlocate 内的数据库记载,找出使用者输入的关键字文件名。
       3. find
       4. 一个文件记录的时间参数有三种,分别是 **access time(atime), status time (ctime), modification time(mtime)**,ls 默认显示的是 mtime。
       5. find atime, ctime 与 mtime 的意义
          1. find / -name passwd 找出文件名为 passwd 这个文件
          2. find / -name "\*passwd\*" 找出文件名包含了 passwd 这个关键字的文件
          3. find /run -type s 找出 /run 目录下,文件类型为 Socket 的文件名有哪些?
          4. # find / -perm /7000 搜寻文件当中含有 SGID 或 SUID 或 SBIT 的属性
          5. find / -size +1M 找出系统中,大于 1MB 的文件
          6. {} 代表的是“由 find 找到的内容”,如上图所示,find 的结果会被放置到 {} 位置中;
          7. -exec 一直到 \; 是关键字,代表 find 额外动作的开始 (-exec) 到结束 (\;) ,在这中 间的就是 find 指令内的额外动作。 在本例中就是“ ls -l {} ” !
          8. 因为“ ; ”在 bash 环境下是有特殊意义的,因此利用反斜线来跳脱。
14. 让使用者能进入某目录成为“可工作目录”的基本权限为何
    1. 可使用的指令:例如 cd 等变换工作目录的指令;
    2. 目录所需权限:使用者对这个目录至少需要具有 x 的权限
    3. 额外需求:如果使用者想要在这个目录内利用 ls 查阅文件名,则使用者对此目录还需要 r 的权限。
15. 使用者在某个目录内读取一个文件的基本权限为何?
    1. 可使用的指令:例如本章谈到的 cat, more, less等等
    2. 目录所需权限:使用者对这个目录至少需要具有 x 权限;
    3. 文件所需权限:使用者对文件至少需要具有 r 的权限才行!
16. 让使用者可以修改一个文件的基本权限为何
    1. 可使用的指令:例如 nano 或未来要介绍的 vi 编辑器等;
    2. 目录所需权限:使用者在该文件所在的目录至少要有 x 权限;
    3. 文件所需权限:使用者对该文件至少要有 r, w 权限
17. 让一个使用者可以创建一个文件的基本权限为何?
    1. 目录所需权限:使用者在该目录要具有 w,x 的权限,重点在 w 啦!
18. 让使用者进入某目录并执行该目录下的某个指令之基本权限为何?
    1. 目录所需权限:使用者在该目录至少要有 x 的权限;
    2. 文件所需权限:使用者在该文件至少需要有 x 的权限
19. 搜寻指令的完整文件名可用 which 或 type ,这两个指令都是通过 PATH 变量来搜寻文件 名;
20. 搜寻文件的完整文件名可以使用 whereis 找特定目录或 locate 到数据库去搜寻,而不实 际搜寻文件系统;
21. 利用 find 可以加入许多选项来直接查询文件系统,以获得自己想要知道的文件名。
22. 找出 /etc 下面,文件大小介于 50K 到 60K 之间的文件,并且将权限完整的列出 (ls - l):
    1. find /etc -size +50k -a -size -60k -exec ls -l {} \; 注意到 -a ,那个 -a 是 and 的意 思,为符合两者才算成功
23. 找出 /etc 下面,文件大小大于 50K 且文件所属人不是 root 的文件名,且将权限完整的列 出 (ls -l)
    1. find /etc -size +50k -a ! -user root -exec ls -ld {} \; find /etc -size +50k -a ! - user root -type f -exec ls -l {}
24. 找出 /etc 下面,容量大于 1500K 以及容量等于 0 的文件
    1. find /etc -size +1500k -o -size 0 相对于 -a ,那个 -o 就是或 (or) 的意思
25. 数据储存与读取的重点在于盘片,而盘片上的物理组成则为
    1. 扇区(Sector)为最小的物理储存单位,且依据磁盘设计的不同,目前主要有 512Bytes 与 4K 两种格式;
    2. 将扇区组成一个圆,那就是柱面(Cylinder)
    3. 早期的分区主要以柱面为最小分区单位,现在的分区通常使用扇区为最小分区单位(每 个扇区都有其号码喔,就好像座位一样)
    4. 磁盘分区表主要有两种格式,一种是限制较多的 MBR 分区表,一种是较新且限制较少的 GPT 分区表
    5. MBR 分区表中,第一个扇区最重要,里面有:(1)主要开机区(Master boot record, MBR)及分区表(partition table), 其中 MBR 占有 446 Bytes,而 partition table 则占 有 64 Bytes
    6. GPT 分区表除了分区数量扩充较多之外,支持的磁盘容量也可以超过 2TB
26. 实体磁盘及虚拟磁盘
    1. /dev/sd[a-p][1-128]:为实体磁盘的磁盘文件名
    2. /dev/vd[a-d][1-128]:为虚拟磁盘的磁盘文件名
27. 文件系统通常会将这两部份的数据分别存放在不同 的区块,权限与属性放置到 inode 中,至于实际数据则放置到 data block 区块中。 另外,还 有一个超级区块 (superblock) 会记录整个文件系统的整体信息,包括 inode 与 block 的总 量、使用量、剩余量等. 每个 inode 与 block 都有编号,至于这三个数据的意义可以简略说明如下:
    1. superblock:记录此 filesystem 的整体信息,包括inode/block的总量、使用量、剩余量, 以及文件系统的格式与相关信息等;
    2. inode:记录文件的属性,一个文件占用一个inode,同时记录此文件的数据所在的 block 号码;
    3. block:实际记录文件的内容,若文件太大时,会占用多个 block 。
    4. 索引式文件系统(indexed allocation)
    5. U盘使用的文件系统一般 为 FAT 格式
    6. Ext2 文件系统在格式化的时候基本上是区分为多个区块群组 (block group) 的,每个区块群组都有独立的 inode/block/superblock 系统
28. 至于每一个区块群组(block group)的六个主要内容说明如后
    1. data block (数据区块) data block 是用来放置文件内容数据地方,在 Ext2 文件系统中所支持的 block 大小有 1K, 2K 及 4K 三种而已
       1. 原则上,block 的大小与数量在格式化完就不能够再改变了(除非重新格式化);
       2. 每个 block 内最多只能够放置一个文件的数据;
       3. 承上,如果文件大于 block 的大小,则一个文件会占用多个 block 数量;
       4. 承上,若文件小于 block ,则该 block 的剩余容量就不能够再被使用了(磁盘空间会浪 费)
    2. inode table (inode 表格)
       1. 该文件的存取模式(read/write/excute);
       2. 该文件的拥有者与群组(owner/group);
       3. 该文件的容量;
       4. 该文件创建或状态改变的时间(ctime);
       5. 最近一次的读取时间(atime);
       6. 最近修改的时间(mtime);
       7. 定义文件特性的旗标(flag),如 SetUID...;
       8. 该文件真正内容的指向 (pointer);
       9. 每个 inode 大小均固定为 128 Bytes (新的 ext4 与 xfs 可设置到 256 Bytes);
       10. 每个文件都仅会占用一个 inode 而已;
       11. 承上,因此文件系统能够创建的文件数量与 inode 的数量有关;
       12. 系统读取文件时需要先找到 inode,并分析 inode 所记录的权限与使用者是否符合,若符 合才能够开始实际读取 block 的内容。
    3. Superblock (超级区块) Superblock 是记录整个 filesystem 相关信息的地方, 没有 Superblock ,就没有这个 filesystem 了。他记录的信息主要有:
       1. block 与 inode 的总量;
       2. 未使用与已使用的 inode / block 数量;
       3. block 与 inode 的大小 (block 为 1, 2, 4K,inode 为 128Bytes 或 256Bytes);
       4. filesystem 的挂载时间、最近一次写入数据的时间、最近一次检验磁盘 (fsck) 的时间 等文件系统的相关信息;
       5. 一个 valid bit 数值,若此文件系统已被挂载,则 valid bit 为 0 ,若未被挂载,则 valid bit 为1。
    4. Filesystem Description (文件系统描述说明)
       1. 这个区段可以描述每个 block group 的开始与结束的 block 号码,以及说明每个区段 (superblock, bitmap, inodemap, data block) 分别介于哪一个 block 号码之间
    5. block bitmap (区块对照表)
       1. 记录的是使用与未使用的 block 号码
    6. inode bitmap (inode 对照表)
       1. 记录使用与未使用的 inode 号码
    7. dumpe2fs: 查询 Ext 家族 superblock 信息的指令 可以使用 **dumpe2fs** 这个指令来查询的
29. 目录 当我们在 Linux 下的文件系统创建一个目录时,文件系统会分配一个 inode 与至少一块 block 给该目录
30. 当我们在 Linux 下的 ext2 创建一个一般文件时, ext2 会分配一个 inode 与相对于该文件大小 的 block 数量给该文件
31. 当我们要读取某个文件时,就务必会经过目录的 inode 与 block ,然后才能够找到那个待读取 文件的 inode 号码, 最终才会读到正确的文件的 block 内的数据
32. 日志式文件系统 (Journaling filesystem)
    1. 预备:当系统要写入一个文件时,会先在日志记录区块中纪录某个文件准备要写入的信 息;
    2. 实际写入:开始写入文件的权限与数据;开始更新metadata的数据;
    3. 结束:完成数据与metadata的更新后,在日志记录区块当中完成该文件的纪录。
33. Linux 系统上面文件系统与内存有非 常大的关系
    1. 系统会将常用的文件数据放置到内存的缓冲区,以加速文件系统的读/写;
    2. 承上,因此 Linux 的实体内存最后都会被用光!这是正常的情况!可加速系统性能;
    3. 你可以手动使用 sync 来强迫内存中设置为 Dirty 的文件回写到磁盘中;
    4. 若正常关机时,关机指令会主动调用 sync 来将内存的数据回写入磁盘内;
    5. 但若不正常关机(如跳电、死机或其他不明原因),由于数据尚未回写到磁盘内, 因此 重新开机后可能会花很多时间在进行磁盘检验,甚至可能导致文件系统的损毁(非磁盘 损毁)
34. 每个 filesystem 都有独立的 inode / block / superblock 等信息,这个文件系统要能够链接到目 录树才能被我们使用. 重点是:挂载点一定是目录,该目录为进入该文件系统的入口。
35. 常见的支持文件系统有
    1. 传统文件系统:ext2 / minix / MS-DOS / FAT (用 vfat 模块) / iso9660 (光盘)等等;
    2. 日志式文件系统: ext3 /ext4 / ReiserFS / Windows' NTFS / IBM's JFS / SGI's XFS / ZFS
    3. 网络文件系统: NFS / SMBFS
36. 想要知道你的 Linux 支持的文件系统有哪些,可以察看下面这个目录:
    1. ls -l /lib/modules/$(uname -r)/kernel/fs
37. 系统目前已载入到内存中支持的文件系统则有:
    1. cat /proc/filesystems
38. 整个 Linux 的系统都是通过一个名为 Virtual Filesystem Switch 的核 心功能去读取 filesystem 的
39. EXT 家族当前较伤脑筋的地方:支持度最广,但格式化超慢!
40. XFS 文件系统在数据的分佈上,主要规划为三个部份,一个数据区 (data section)、一个文 件系统活动登录区 (log section)以及一个实时运行区 (realtime section) , XFS 文件系统的配置
    1. 数据区 (data section)
    2. 文件系统活动登录区 (log section)
    3. 实时运行区 (realtime section)
41. 找出系统 /boot 这个挂载点下面的文件系统的 superblock 纪录
    1. df -T /boot
42. 磁盘与目录的容量
    1. df:列出文件系统的整体磁盘使用量;
    2. du:评估文件系统的磁盘使用量(常用在推估目录所占容量)
    3. 将系统内的所有特殊文件格式及名称都列出来 **df -aT**
    4. 将目前各个 partition 当中可用的 inode 数量列出 **df -ih**
    5. 至于那个 /dev/shm/ 目录,其实是利用内存虚拟出来的磁盘空间,通常是总实体内存的一半
    6. 列出目前目录下的所有文件大小 **du -a**
    7. 检查根目录下面每个目录所占用的容量 **du -sm /\***
43. 实体链接与符号链接: ln
    1. Hard Link (实体链接, 硬式链接或实际链接) hard link 只是在某个目录下新增一笔文件名链接到某 inode 号码的关连记录而已. hard link 只是在某个目录下的 block 多写入一个关连 数据而已,既不会增加 inode 也不会耗用 block 数量 **ln** 
       1. 不能跨 Filesystem;
       2. 不能 link 目录。
    2. Symbolic Link (符号链接,亦即是捷径) Symbolic link 就是在创建一个 独立的文件,而这个文件会让数据的读取指向他 link 的那个文件的文件名 **ln -s**
44. 观察磁盘分区状态
    1. lsblk 列出系统上的所有磁盘列表 lsblk 可以看成“ list block device ”的缩写
    2. blkid 列出设备的 UUID 等参数 UUID 是全域单一识别码 (universally unique identifier)
    3. parted 列出磁盘的分区表类型与分区信息
45. MBR 分区表请使用 fdisk 分区, GPT 分区 表请使用 gdisk 分区
    1. partprobe 更新 Linux 核心的分区表信息
    2. 用 gdisk 删除一个分区
    3. 用 gdisk 新增分区
46. 磁盘格式化(创建文件系统)
    1. XFS 文件系统 mkfs.xfs
    2. XFS 文件系统 for RAID 性能优化
    3. 磁盘阵列 (RAID) 就是通过将文件先细分为数个小型的分区区块 (stripe) 之 后,然后将众多的 stripes 分别放到磁盘阵列里面的所有磁盘, 所以一个文件是被同时写入到 多个磁盘去,当然性能会好一些
    4. EXT4 文件系统 mkfs.ext4
    5. 其他文件系统 mkfs
47. 文件系统检验
    1. xfs\_repair 处理 XFS 文件系统
    2. fsck.ext4 处理 EXT4 文件系统
48. 文件系统挂载与卸载 使用 **mount** 这个指令
    1. 单一文件系统不应该被重复挂载在不同的挂载点(目录)中;
    2. 单一目录不应该重复挂载多个文件系统;
    3. 要作为挂载点的目录,理论上应该都是空目录才是。
    4. /etc/filesystems:系统指定的测试挂载文件系统类型的优先顺序
    5. /proc/filesystems:Linux系统已经载入的文件系统类型
    6. /lib/modules/$(uname -r)/kernel/fs/
    7. 挂载 xfs/ext4/vfat 等文件系统
    8. 挂载 CD 或 DVD 光盘
    9. 挂载 vfat 中文U盘 (USB磁盘)
    10. 重新挂载根目录与挂载不特定目录
    11. umount (将设备文件卸载) 卸载之后,可以使用 df 或 mount 看看是否还存 在目录树中
49. 磁盘/文件系统参数修订
    1. mknod
    2. xfs\_admin 修改 XFS 文件系统的 UUID 与 Label name
    3. tune2fs 修改 ext4 的 label name 与 UUID
50. 常见的磁盘文件名 /dev/sda 与 /dev/loop0 设备代码
    1. /dev/sda 8 0-15
    2. /dev/sdb 8 16-31
    3. /dev/loop0 7 0
    4. /dev/loop1 7 1
51. 开机挂载 /etc/fstab
    1. 根目录 / 是必须挂载的,而且一定要先于其它 mount point 被挂载进来。
    2. 其它 mount point 必须为已创建的目录,可任意指定,但一定要遵守必须的系统目录架构 原则 (FHS)
    3. 所有 mount point 在同一时间之内,只能挂载一次。
    4. 所有 partition 在同一时间之内,只能挂载一次。
    5. 如若进行卸载,您必须先将工作目录移到 mount point(及其子目录) 之外。
52. cat /etc/fstab : [设备/UUID等] [挂载点] [文件系统] [文件系统参数] [dump] [fsck]
    1. 第一栏:磁盘设备文件名/UUID/LABEL name:
       1. 文件系统或磁盘的设备文件名,如 /dev/vda2 等
       2. 文件系统的 UUID 名称,如 UUID=xxx
       3. 文件系统的 LABEL 名称,例如 LABEL=xxx
    2. 第二栏:挂载点 (mount point)
    3. 第三栏:磁盘分区的文件系统, 包括 xfs, ext4, vfat, reiserfs, nfs 等等
    4. 第四栏:文件系统参数:
    5. 第五栏:能否被 dump 备份指令作用
    6. 第六栏:是否以 fsck 检验扇区
53. /etc/fstab 是开机时的配置文件,不过,实际 filesystem 的挂载是记录到 /etc/mtab 与 /proc/mounts 这两个文件当中的
54. 特殊设备 loop 挂载 (镜像文件不烧录就挂载使用)
    1. 挂载光盘/DVD镜像文件 **mount -o loop**
    2. 创建大文件以制作 loop 设备文件
55. 安装时一定 需要的两个 partition ,一个是根目录,另外一个就是 swap(内存交换空间) . 由于 swap 是用磁盘来暂时放置内存中的信息,所以用到 swap 时, 你的主机磁盘灯就会开始闪个不停啊
    1. 设置一个 swap partition
    2. 创建一个虚拟内存的文件
56. 使用实体分区创建swap
    1. 分区:先使用gdisk在你的磁盘中分区出一个分区给系统作为swap。由于Linux的 gdisk 默认会将分区的 ID 设置为 Linux 的文件系统,所以你可能还得要设置一下 system ID 就是了。
    2. 格式化:利用创建swap格式的“mkswap设备文件名”就能够格式化该分区成为swap格 式
    3. 使用:最后将该swap设备启动,方法为:“swapon设备文件名”。
       1. 列出目前使用的 swap 设备有哪些的意思! swapon -s
    4. 观察:最终通过free与swapon-s这个指令来观察一下内存的用量吧!
57. 使用文件创建swap
    1. 使用 dd 这个指令来新增一个 128MB 的文件在 /tmp 下面
    2. 使用 mkswap 将 /tmp/swap 这个文件格式化为 swap 的文件格式
    3. 使用 swapon 来将 /tmp/swap 启动
    4. 使用 swapoff 关掉 swap file,并设置自动启用
58. 每个文件所使用掉 block 的容量 **ll -sh**
59. 利用 GNU 的 parted 进行分区行为 parted 可以直接在一行命令行就完成分区,是一个非常好用的指令 **parted [设备] [指令 [参数]]**可以使用“ man parted ”,或者是“ parted /dev/vda help mkpart ”去查询更详细的数据
    1. Number:这个就是分区的号码啦!举例来说,1号代表的是/dev/vda1的意思
    2. Start:分区的起始位置在这颗磁盘的多少MB处?有趣吧!他以容量作为单位喔
    3. End:此分区的结束位置在这颗磁盘的多少MB处
    4. Size:由上述两者的分析,得到这个分区有多少容量
    5. Filesystem:分析可能的文件系统类型为何的意思
    6. Name:就如同gdisk的SystemID之意
60. 一个可以被挂载的数据通常称为“文件系统, filesystem”而不是分区 (partition) 喔
61. 基本上 Linux 的传统文件系统为 Ext2 ,该文件系统内的信息主要有
    1. superblock:记录此 filesystem 的整体信息,包括inode/block的总量、使用量、剩余 量, 以及文件系统的格式与相关信息等
    2. inode:记录文件的属性,一个文件占用一个inode,同时记录此文件的数据所在的 block 号码
    3. block:实际记录文件的内容,若文件太大时,会占用多个 block
62. Ext2 文件系统的数据存取为索引式文件系统(indexed allocation)
63. Ext2文件系统主要有:boot sector, superblock, inode bitmap, block bitmap, inode table, data block 等六大部分
64. data block 是用来放置文件内容数据地方,在 Ext2 文件系统中所支持的 block 大小有 1K, 2K 及 4K 三种而已
65. inode 记录文件的属性/权限等数据,其他重要项目为: 每个 inode 大小均为固定,有 128/256Bytes 两种基本容量。每个文件都仅会占用一个 inode 而已; 因此文件系统能够 创建的文件数量与 inode 的数量有关
66. 文件的 block 在记录文件的实际数据,目录的 block 则在记录该目录下面文件名与其 inode 号码的对照表
67. 日志式文件系统 (journal) 会多出一块记录区,随时记载文件系统的主要活动,可加快 系统复原时间
68. Linux 文件系统为增加性能,会让内存作为大量的磁盘高速缓存
69. 磁盘的使用必需要经过:分区、格式化与挂载,分别惯用的指令为:gdisk, mkfs, mount 三个指令
70. 分区时,应使用 parted 检查分区表格式,再判断使用 fdisk/gdisk 来分区,或直接使用 parted 分区
71. 开机自动挂载可参考/etc/fstab之设置,设置完毕务必使用 mount -a 测试语法正确否
72. Linux上常见的压缩指令就是 gzip, bzip2 以及最新的 xz. 常见的压缩文件扩展名
    1. \*.Z compress 程序压缩的文件;
    2. \*.zip zip 程序压缩的文件;
    3. \*.gz gzip 程序压缩的文件;
    4. \*.bz2 bzip2 程序压缩的文件;
    5. xz 程序压缩的文件;
    6. .tar tar 程序打包的数据,并没有压缩过;
    7. \*.tar.gz tar 程序打包的文件,其中并且经过 gzip 的压缩
    8. \*.tar.bz2 tar 程序打包的文件,其中并且经过 bzip2 的压缩
    9. \*.tar.xz tar 程序打包的文件,其中并且经过 xz 的压缩
73. gzip, zcat/zmore/zless/zgrep
    1. 当使用 gzip 进行压缩时,默认的状态下原本的文件会被压缩成为 .gz 的文件名,原始文件就不再存在了
    2. **zcat 文件名.gz**
    3. gzip 压缩 **gzip -v services**
    4. 文件解压缩 **gzip -d services.gz**
    5. 创建的 services.gz 中,找出 http 这个关键字在哪几行? **zgrep -n 'http' services.gz**
74. bzip2, bzcat/bzmore/bzless/bzgrep
    1. bzip2 真是很不错用的东西~这玩意的压缩比竟然比 gzip 还要好
    2. bzip2 [-cdkzv#] 文件名
    3. bzcat 文件名.bz2
    4. bzip2 压缩 **bzip2 -v services**
    5. 文件内容读出来 **bzcat services.bz2**
    6. 文件解压缩 **bzip2 -d services.bz2**
75. xz, xzcat/xzmore/xzless/xzgrep
    1. xz [-dtlkc#] 文件名
    2. xcat 文件名.xz
    3. xz 来压缩 **xz -v services**
    4. 读出这个压缩文件的内容 **xz -l services.xz**
    5. 文件内容读出来 **xzcat services.xz**
    6. 解压缩  **xz -d services.xz**
    7. 保留原文件的文件名,并且创建压缩文件 **xz -k services**
76. 打包指令: tar
    1. 压 缩:tar -j<u>c</u>v -f filename.tar.bz2 要被压缩的文件或目录名称
    2. 查 询:tar -j<u>t</u>v -f filename.tar.bz2
    3. 解压缩:tar -j<u>x</u>v -f filename.tar.bz2 -C 欲解压缩的目录
    4. 使用 tar 加入 -z, -j 或 -J 的参数备份 /etc/ 目录
       1. time tar -jpcv -f /root/etc.tar.bz2 /etc
       2. time tar -Jpcv -f /root/etc.tar.xz /etc
    5. 查阅 tar 文件的数据内容 (可察看文件名),与备份文件名有否根目录的意义
       1. tar -jtv -f /root/etc.tar.bz2
    6. 将备份的数据解压缩,并考虑特定目录的解压缩动作 (-C 选项的应用)
       1. 直接进行解打包 **tar -jxv -f /root/etc.tar.bz2**
       2. 指定欲解 开的目录  **tar -jxv -f /root/etc.tar.bz2 -C /tmp**
    7. 仅解开单一文件的方法
       1. 假设解开 shadow 文件 
          1. **tar -jtv -f /root/etc.tar.bz2 &#124; grep 'shadow'**
          2. **tar -jxv -f 打包档.tar.bz2 待解开文件名**
          3. **tar -jxv -f /root/etc.tar.bz2 etc/shadow**
       2. 打包某目录,但不含该目录下的某些文件之作法
          1. **tar -jcv -f /root/system.tar.bz2 --exclude=/root/etc\* \**

**&gt; --exclude=/root/system.tar.bz2 /etc /root**

1. 仅备份比某个时刻还要新的文件
   1. find /etc -newer /etc/passwd
   2. 使用 tar 来进行打包吧!日期为上面看到的 2015/06/17

tar -jcv -f /root/etc.newer.then.passwd.tar.bz2 \

&gt; --newer-mtime="2015/06/17" /etc/\*

1. 显示出文件即可

tar -jtv -f /root/etc.newer.then.passwd.tar.bz2 &#124; grep -v '/$'

1. 基本名称: tarfile, tarball ?
   1. 如果仅是打 包而已,就是“ tar -cv -f file.tar ”而已,这个文件我们称呼为 tarfile
   2. 如果还有进行压缩的支 持,例如“ tar -jcv -f file.tar.bz2 ”时,我们就称呼为 tarball (tar 球?)
2. 特殊应用:利用管线命令与数据流
   1. 将 /etc 整个目录一边打包一边在 /tmp 解开  **tar -cvf - /etc &#124; tar -xvf -**
3. 解压缩后的 SELinux 课题
   1. 通过各种可行的救援方式登陆系统,然后修改 /etc/selinux/config 文件,将 SELinux 改成 permissive 模式,重新开机后系统就正常了;
   2. 在第一次复原系统后,不要立即重新开机!先使用 restorecon -Rv /etc 自动修复一下 SELinux 的类型即可。
   3. 通过各种可行的方式登陆系统,创建 /.autorelabel 文件,重新开机后系统会自动修复 SELinux 的类型,并且又会再次重新开机,之后就正常了
4. XFS 文件系统备份 xfsdump
   1. xfsdump 不支持没有挂载的文件系统备份!所以只能备份已挂载的!
   2. xfsdump 必须使用 root 的权限才能操作 (涉及文件系统的关系)
   3. xfsdump 只能备份 XFS 文件系统啊!
   4. xfsdump 备份下来的数据 (文件或储存媒体) 只能让 xfsrestore 解析
   5. xfsdump 是通过文件系统的 UUID 来分辨各个备份文件的,因此不能备份两个具有相同 UUID 的文件系统喔!
   6. xfsdump [-L S\_label] [-M M\_label] [-l #] [-f 备份文件] 待备份数据
   7. 用 xfsdump 备份完整的文件系统
      1. 先确定 /boot 是独立的文件系统喔! df -h /boot
      2. 将完整备份的文件名记录成为 /srv/boot.dump xfsdump -l 0 -L boot\_all -M boot\_all -f /srv/boot.dump /boot
   8. 用 xfsdump 进行累积备份 (Incremental backups)
      1. 看一下有没有任何文件系统被 xfsdump 过的数据? **xfsdump -I**
      2. 创建一个大约 10 MB 的文件在 /boot 内 **dd if=/dev/zero of=/boot/testing.img bs=1M count=10**
      3. 开始创建差异备份文件,此时我们使用 level 1 xfsdump -l 1 **-L boot\_2 -M boot\_2 -f /srv/boot.dump1 /boot**
      4. 最后再看一下是否有记录 level 1 备份的时间点呢?  **xfsdump -I**
5. XFS 文件系统还原 xfsrestore
   1. 用 xfsrestore 观察 xfsdump 后的备份数据内容
   2. 简单复原 level 0 的文件系统
      1. 直接将数据给它覆盖回去 **xfsrestore -f /srv/boot.dump -L boot\_all /boot**
      2. 将备份数据在 /tmp/boot 下面解开! **xfsrestore -f /srv/boot.dump -L boot\_all /tmp/boot du -sm /boot /tmp/boot**
      3. 检查两者一致否： **diff -r /boot /tmp/boot**
   3. 仅还原部分文件的 xfsrestore 互动模式 **xfsrestore -f /srv/boot.dump -i /tmp/boot3**
6. 光盘写入工具
   1. 先将所需要备份的数据创建成为一个镜像文件(iso),利用 mkisofs 指令来处理
      1. mkisofs [-o 镜像文件] [-Jrv] [-V vol] [-m file] 待备份文件... &gt; -graft-point isodir=systemdir …
   2. 将该镜像文件烧录至光盘或 DVD 当中,利用 cdrecord 指令来处理
      1. mkisofs -r -v -o /tmp/system.img /root /home /etc
      2. mkisofs -r -V 'linux\_file' -o /tmp/system.img &gt; -m /root/etc -graft-point /root=/root /home=/home /etc=/etc
      3. 如果你是有自己要制作的数 据内容,其实最简单的方法,就是将所有的数据预先处理到某一个目录中, 再烧录该目录即 可
   3. 制作/修改可开机光盘图像挡
      1. 先观察一下这片光盘里面有啥东西?是否是我们需要的光盘系统!
         1. isoinfo -d -i /home/CentOS-7-x86\_64-Minimal-1503-01.iso
      2. 开始挂载这片光盘到 /mnt ,并且将所有数据完整复制到 /srv/newcd 目录去喔
         1. mount /home/CentOS-7-x86\_64-Minimal-1503-01.iso /mnt
         2. mkdir /srv/newcd
         3. rsync -a /mnt/ /srv/newcd
      3. 假设已经处理完毕你在 /srv/newcd 里面所要进行的各项修改行为,准备创建 ISO 档!
         1. cd /srv/newcd
         2. Mkisofs -o /custom.iso -b isolinux/isolinux.bin -c isolinux/boot.cat &gt; -no-emul-boot -V 'CentOS 7 x86\_64' -boot-load-size 4 -boot-info-table -R -J -v -T .
7. cdrecord:光盘烧录工具 wodim 这个文字界面指令来进行烧录的行为
   1. 侦测你的烧录机所在位置
      1. ll /dev/sr0
      2. wodim --devices dev=/dev/sr0
      3. wodim --devices dev=/dev/sr0
   2. 进行 CD/DVD 的烧录动作
      1. 先抹除光盘的原始内容:(非可重复读写则可略过此步骤) wodim -v dev=/dev/sr0 blank=fast
      2. 开始烧录 wodim -v dev=/dev/sr0 speed=4 -dummy -eject /tmp/system.img
      3. 烧录完毕后,测试挂载一下,检验内容: mount /dev/sr0/mnt umount /mnt
8. dd 指令
   1. 将 /etc/passwd 备份到 /tmp/passwd.back 当中 **dd if=/etc/passwd of=/tmp/passwd.back**
   2. 将刚刚烧录的光驱的内容,再次的备份下来成为图像挡 **dd if=/dev/sr0 of=/tmp/system.iso**
   3. 假设你的 USB 是 /dev/sda 好了,请将刚刚范例二的 image 烧录到 USB 磁盘中

**lsblk /dev/sda dd if=/tmp/system.iso of=/dev/sda mount /dev/sda /mnt**

1. 将你的 /boot 整个文件系统通过 dd 备份下来  **dd if=/dev/vda2 of=/tmp/vda2.img**
2. 未来你想要创建两颗一模一样的磁盘时, 只要 下达类似: dd if=/dev/sda of=/dev/sdb ,就能够让两颗磁盘一模一样,甚至 /dev/sdb 不需要 分区与格式化, 因为该指令可以将 /dev/sda 内的所有数据,包括 MBR 与 partition table 也复 制到 /dev/sdb 说
3. cpio 备份
   1. 找出 /boot 下面的所有文件,然后将他备份到 /tmp/boot.cpio 去:

find boot &#124; cpio -ocvB &gt; /tmp/boot.cpio

1. 将刚刚的文件给他在 /root/ 目录下解开 Cpio -idvc &lt; /tmp/boot.cpio
2. 事实上 cpio 可以将系统的数据完整的备份到磁带机上头去喔!如果你有磁带机的话
   1. 备份:find / | cpio -ocvB > /dev/st0
   2. 还原:cpio -idvc < /dev/st0
3. 其实系统里面已经含有一个使用 cpio 创建的文件喔!那就是 /boot/initramfs-xxx 这个文件 啦
   1. 我们先来看看该文件是属于什么文件格式,然后再加以处理: file /boot/initramfs-3.10.0-229.el7.x86\_64.img
   2. cd /tmp/initramfs
   3. cpio -idvc &lt; /boot/initramfs-3.10.0-229.el7.x86\_64.img
4. 压 缩:tar -Jcv -f filename.tar.xz 要被压缩的文件或目录名称
5. 查 询:tar -Jtv -f filename.tar.xz
6. 解压缩:tar -Jxv -f filename.tar.xz -C 欲解压缩的目录
7. xfsdump 指令可备份文件系统或单一目录 xfsdump 的备份若针对文件系统时,可进行 0-9 的 level 差异备份!其中 level 0 为完整 备份
8. xfsrestore 指令可还原被 xfsdump 创建的备份文件
9. 要创建光盘烧录数据时,可通过 mkisofs 指令来创建
10. 可通过 wodim 来写入 CD 或 DVD 烧录机
11. dd 可备份完整的 partition 或 disk ,因为 dd 可读取磁盘的 sector 表面数据
12. cpio 为相当优秀的备份指令,不过必须要搭配类似 find 指令来读入欲备份的文件名数 据,方可进行备份动作
13. 文书编辑器有哪些 emacs, pico, nano, joe, 与 vim
14. vi 的使用
    1. 一般指令模式 (command mode)
    2. 编辑模式 (insert mode) 等到你按下“i, I, o, O, a, A, r, R”等任何一个字母之后才会进入编辑模式 , 回到一般指令模式时, 则必须要按下“Esc”这个按键即可退出编辑模 式
    3. 命令行命令模式 (command-line mode) 输入“ : / ? ”三个中的任何一个按钮,就可以将光标移动到最下面那一列
    4. 可以使用“强制写入”的方式吗?可以!使 用“ **:wq!** ” 多加一个惊叹号即可
15. vi 一般指令模式可用的按钮说明,光标移动、复制贴上、搜寻取代等
    1. h 或 向左方向键(←)
    2. j 或 向下方向键(↓)
    3. k 或 向上方向键(↑)
    4. l 或 向右方向键(→)
    5. 向下移动 30 列, 可以使用 "**30j**" 或 "**30↓**" 的组合按键, 亦即加上想要进行的次数(数字)后, 按下动作即可
    6. **[Ctrl] + [f] 屏幕“向下”移动一页,相当于 [Page Down]按 键 (常用)**
    7. **[Ctrl] + [b] 屏幕“向上”移动一页,相当于 [Page Up] 按键 (常用**
    8. [Ctrl] + [d] 屏幕“向下”移动半页
    9. [Ctrl] + [u] 屏幕“向上”移动半页
    10. + 光标移动到非空白字符的下一列
    11. - 光标移动到非空白字符的上一列
    12. n<space> 那个 n 表示“数字”, 例如 20<space> 则光标会向后面移动 20 个字符距离
    13. **0 或功能键[Home] 这是数字“ 0 ”:移动到这一列的最前面字符处 (常用)**
    14. **$ 或功能键[End] 移动到这一列的最后面字符处(常用)**
    15. H 光标移动到这个屏幕的最上方那一列的第一个字符
    16. M 光标移动到这个屏幕的中央那一列的第一个字符
    17. L 光标移动到这个屏幕的最下方那一列的第一个字符
    18. **G 移动到这个文件的最后一列(常用)**
    19. nG n 为数字。移动到这个文件的第 n 列 例如 20G 则会移动到这个文件的第 20 列(可配合 **:set nu**)
    20. **gg 移动到这个文件的第一列,相当于 1G 啊! (常用)**
    21. **n<Enter> n 为数字。光标向下移动 n 列(常用)**
    22. **/word 向光标之下寻找一个名称为 word 的字串 (常用)**
    23. ?word 向光标之上寻找一个字串名称为 word 的字 串。
    24. n 这个 n 是英文按键。代表“<u>重复前一个搜 寻的动作</u>”。举例来说, 如果刚刚我们执 行 /vbird 去向下搜寻 vbird 这个字串,则按下 n 后,会向下继续搜寻下一个名称为 vbird 的 字串。如果是执行 ?vbird 的话,那么按下 n 则会向上继续搜寻名称为 vbird 的字串
    25. N 这个 N 是英文按键。与 n 刚好相反,为“反 向”进行前一个搜寻动作。 例如 /vbird 后,按 下 N 则表示“向上”搜寻 vbird
    26. **:n1,n2s/word1/word2/g n1 与 n2 为数字。在第 n1 与 n2 列之间寻找 word1 这个字串,并将该字串取代为 word2 !举例来说,在 100 到 200 列之间搜寻 vbird 并取代为 VBIRD 则:“:100,200s/vbird/VBIRD/g”。(常用)**
    27. **:1,$s/word1/word2/g 从第一列到最后一列寻找 word1 字串,并将 该字串取代为 word2 !(常用)**
    28. **:1,$s/word1/word2/gc 从第一列到最后一列寻找 word1 字串,并将 该字串取代为 word2 !且在取代前显示提示 字符给使用者确认 (confirm) 是否需要取 代!(常用)**
    29. **x, X 在一列字当中,x 为向后删除一个字符 (相 当于 [del] 按键), X 为向前删除一个字符 (相当于 [backspace] 亦即是倒退键) (常 用)**
    30. nx n 为数字,连续向后删除 n 个字符。举例来 说,我要连续删除 10 个字符, “10x”。
    31. **dd 删除光标所在的那一整列(常用)**
    32. **ndd n 为数字。删除光标所在的向下 n 列,例如 20dd 则是删除 20 列 (常用)**
    33. d1G 删除光标所在到第一列的所有数据
    34. dG 删除光标所在到最后一列的所有数据
    35. d$ 删除光标所在处,到该列的最后一个字符
    36. d0 那个是数字的 0 ,删除光标所在处,到该列 的最前面一个字符
    37. **yy 复制光标所在的那一列(常用)**
    38. **nyy n 为数字。复制光标所在的向下 n 列,例如 20yy 则是复制 20 列(常用)**
    39. y1G 复制光标所在列到第一列的所有数据
    40. yG 复制光标所在列到最后一列的所有数据
    41. y0 复制光标所在的那个字符到该列行首的所有数据
    42. y$ 复制光标所在的那个字符到该列行尾的所有数据
    43. **p, P p 为将已复制的数据在光标下一列贴上,P 则 为贴在光标上一列! 举例来说,我目前光标 在第 20 列,且已经复制了 10 列数据。则按 下 p 后, 那 10 列数据会贴在原本的 20 列之 后,亦即由 21 列开始贴。但如果是按下 P 呢? 那么原本的第 20 列会被推到变成 30 列。 (常用)**
    44. J 将光标所在列与下一列的数据结合成同一列
    45. c 重复删除多个数据,例如向下删除 10 列,[ 10cj ]
    46. **u 复原前一个动作。(常用)**
    47. **[Ctrl]+r 重做上一个动作。(常用)**
    48. **. 不要怀疑!这就是小数点!意思是重复前一 个动作的意思。 如果你想要重复删除、重复 贴上等等动作,按下小数点“.”就好了! (常 用)**
16. vi 一般指令模式切换到编辑模式的可用的按钮说明
    1. **i, I 进入插入模式(Insert mode):i 为“从目前光标所在处插入”, I 为“在目前所在列的第一个非空白 字符处开始插入”。 (常用)**
    2. **a, A 进入插入模式(Insert mode):a 为“从目前光标所在的下一个字符 处开始插入”, A 为“从光标所在列 的最后一个字符处开始插入”。 (常用**
    3. **o, O 进入插入模式(Insert mode): 这是英文字母 o 的大小写。o 为“在目前光标所在的下一列处插 入新的一列”; O 为在目前光标所 在处的上一列插入新的一列!(常 用)**
    4. **r, R 进入取代模式(Replace mode):r 只会取代光标所在的 那一个字符一次;R会一直取代光 标所在的文字,直到按下 ESC 为 止;(常用)**
    5. **[Esc] 退出编辑模式,回到一般指令模式中(常用)**
17. vi一般指令模式切换到命令行界面的可用按钮说明
    1. **:w 将编辑的数据写入硬盘文件中(常用)**
    2. **:q 离开 vi (常用)**
    3. **:wq 储存后离开,若为 :wq! 则为强制储存后离开 (常用)**
    4. ZZ 这是大写的 Z 喔!若文件没有更动,则不储存离开,若 文件已经被更动过,则储存后离开
    5. :w [filename] 将编辑的数据储存成另一个文件(类似另存新文件)
    6. :r [filename] 在编辑的数据中,读入另一个文件的数据。亦即将 “filename” 这个文件内容加到光标所在列后面
    7. :n1,n2 w [filename] 将 n1 到 n2 的内容储存成 filename 这个文件。
    8. :! command 暂时离开 vi 到命令行界面下执行 command 的显示结 果!例如 “:! ls /home”即可在 vi 当中察看 /home 下面以 ls 输出的文件信息!
    9. :set nu 显示行号,设置之后,会在每一列的字首显示该列的行号
    10. :set nonu 与 set nu 相反,为取消行号!
18. vim 区块选择(Visual Block)
    1. v 字符选择,会将光标经过的地方反白选择!
    2. V 列选择,会将光标经过的列反白选择!
    3. [Ctrl]+v 区块选择,可以用长方形的方式选择数据
    4. y 将反白的地方复制起来
    5. d 将反白的地方删除掉
    6. p 将刚刚复制的区块,在光标所在处贴上!
19. vim 多文件编辑
    1. :n 编辑下一个文件
    2. :N 编辑上一个文件
    3. :files 列出目前这个 vim 的打开的所有文件
20. vim 多窗口功能
    1. :sp [filename] 打开一个新窗口,如果有加 filename, 表示在新窗口打开一个新文件,否 则表示两个窗口为同一个文件内容(同步显示)。
    2. [ctrl]+w+ j [ctrl]+w+↓ 按键的按法是:先按下 [ctrl] 不放, 再按下 w 后放开所有的按键,然后再按 下 j (或向下方向键),则光标可移动到下方的窗口
    3. [ctrl]+w+ k [ctrl]+w+↑ 同上,不过光标移动到上面的窗口。
    4. [ctrl]+w+ q 其实就是 :q 结束离开啦! 举例来说,如果我想要结束下方的窗口,那么利 用 [ctrl]+w+↓ 移动到下方窗口后,按下 :q 即可离开, 也可以按下 [ctrl]+w+q 啊
21. vim 的挑字补全功能
    1. [ctrl]+x -> [ctrl]+n 通过目前正在编辑的这个“文件的内容文字”作为关键字, 以补齐
    2. [ctrl]+x -> [ctrl]+f 以当前目录内的“文件名”作为关键字, 以补齐
    3. [ctrl]+x -> [ctrl]+o 以扩展名作为语法补充,以 vim 内置的关键字, 以补齐
22. vim 环境设置与记录: ~/.vimrc, ~/.viminfo
    1. vim 会主动的将你曾经做过的行为登录下来,好让你下次可以轻松的作业 啊! 那个记录动作的文件就是: **~/.viminfo**
    2. 通过配置文件来直接规定我们习惯的 vim 操作 环境呢, 整体 vim 的设置值一般是放置在 **/etc/vimrc** 这个文件 ,不过,不建议你修改他! 你 可以修改 ~/.vimrc 这个文件 (默认不存在,请你自行手动创建!),将你所希望的设置值写入
23. 其他 vim 使用注意事项
    1. DOS 与 Linux 的断行字符
       1. dos2unix [-kn] file [newfile]
       2. unix2dos [-kn] file [newfile]
    2. 语系编码转换
       1. iconv --list
       2. iconv -f 原本编码 -t 新编码 filename [-o newfile]
24. Vim 指令
    1. 数字是有意义的,用来说明重复进行几次动作的意思,如 5yy 为复制 5 列之意
    2. 光标的移动中,大写的 G 经常使用,尤其是 1G, G 移动到文章的头/尾功能
    3. vi 的取代功能也很棒! :n1,n2s/old/new/g 要特别注意学习起来
    4. 小数点“ . ”为重复进行前一次动作,也是经常使用的按键功能
    5. 进入编辑模式几乎只要记住: i, o, R 三个按钮即可!尤其是新增一列的 o 与取代的 R
    6. vim 会主动的创建 swap 暂存盘,所以不要随意断线
    7. 如果在文章内有对齐的区块,可以使用 [ctrl]-v 进行复制/贴上/删除的行为
    8. 使用 :sp 功能可以分区窗口
    9. 在 vi 下面作了很多的编辑动作之后,却想还原成原来的文件内容,应该怎么进行?直接 按下 :e! 即可恢复成文件的原始状态
25. Bash shell 的功能
    1. 命令编修能力 (history)
       1. ~/.bash\_history 记录的是前一次登陆以前所执行过的指令, 而至于这一次登陆所执行的指令 都被暂存在内存中,当你成功的登出系统后,该指令记忆才会记录到 .bash\_history 当中
    2. 命令与文件补全功能: ([tab] 按键的好处)
       1. [Tab] 接在一串指令的第一个字的后面,则为命令补全
       2. [Tab] 接在一串指令的第二个字以后时,则为“文件补齐”
       3. 若安装 bash-completion 软件,则在某些指令后面使用 [tab] 按键时,可以进行“选项/参 数的补齐”功能
    3. 命令别名设置功能: (alias)
       1. alias lm='ls -al'
    4. 工作控制、前景背景控制: (job control, foreground, background)
    5. 程序化脚本: (shell scripts)
    6. 万用字符: (Wildcard)
    7. 查询指令是否为 Bash shell 的内置命令: **type**
26. Bash shell指令的下达与快速编辑按钮
    1. **[ctrl]+u/[ctrl]+k 分别是从光标处向前删除指令串 ([ctrl]+u) 及向后删除指令串 ([ctrl]+k)**
    2. **[ctrl]+a/[ctrl]+e 分别是让光标移动到整个指令串的最前面 ([ctrl]+a) 或最后面 ([ctrl]+e)**
    3. 为了区别与自订变量的不同,环境变量通常以大写字符来表示呢
27. Bash shell变量的设置规则
    1. 变量与变量内容以一个等号“=”来链接
    2. 等号两边不能直接接空白字符,如下所示为错误: “myname = VBird”
    3. 变量名称只能是英文字母与数字,但是开头字符不能是数字,如下为错误: “2myname=VBird”
    4. 变量内容若有空白字符可使用双引号“"”或单引号“'”将变量内容结合起来,但
       1. 双引号内的特殊字符如 $ 等,可以保有原本的特性,如下所示: “var="lang is $LANG"”则“echo $var”可得“lang is zh\_TW.UTF-8”
       2. 单引号内的特殊字符则仅为一般字符 (纯文本),如下所示: “var='lang is $LANG'”则“echo $var”可得“lang is $LANG”
    5. 可用跳脱字符“ \ ”将特殊符号(如 [Enter], $, \, 空白字符, '等)变成一般字符,如: “myname=VBird\ Tsai”
    6. 在一串指令的执行中,还需要借由其他额外的指令所提供的信息时,可以使用反单引 号“ 指令 ”或 “$(指令)”。特别注意,那个 ` 是键盘上方的数字键 1 左边那个按键,而不 是单引号! 例如想要取得核心版本的设置: “version=$(uname -r)”再“echo $version”可得“3.10.0-229.el7.x86\_64”
    7. 若该变量为扩增变量内容时,则可用 "$变量名称" 或 ${变量} 累加内容,如下所示: “PATH="$PATH":/home/bin”或“PATH=${PATH}:/home/bin”
    8. 若该变量需要在其他子程序执行,则需要以 export 来使变量变成环境变量: “export PATH”
    9. 取消变量的方法为使用 unset :“unset 变量名称”例如取消 myname 的设置: “unset myname”
    10. locate 指令可以列出所有的相关文件文件名
28. 如何进入到您目前核心的模块目录
    1. cd /lib/modules/`uname -r`/kernel
    2. cd /lib/modules/$(uname -r)/kernel # 以此例较佳!
29. 环境变量的功能
    1. 用 env 观察环境变量与常见环境变量说明
    2. 关于本 shell 的 PID echo $$
    3. ?:(关于上个执行指令的回传值) echo $?
    4. export: 自订变量转成环境变量
    5. 子程序仅会继承父程序的环境变量, 子 程序不会继承父程序的自订变量 export 指令 如你想要让该变量 内容继续的在子程序中使用  **export 变量名称**
    6. 影响显示结果的语系变量 (locale)
30. 变量的有效范围
    1. “全域变量, global variable”与“区域变量, local variable” 环境变量=全域变量 自订变量=区域变量
    2. 当启动一个 shell,操作系统会分配一记忆区块给 shell 使用,此内存内之变量可让子程 序取用
    3. 若在父程序利用 export 功能,可以让自订变量的内容写到上述的记忆区块当中(环境变 量);
    4. 当载入另一个 shell 时 (亦即启动子程序,而离开原本的父程序了),子 shell 可以将父 shell 的环境变量所在的记忆区块导入自己的环境变量区块当中。
31. 变量键盘读取、阵列与宣告: read, array, declare
    1. 要读取来自键盘输入的变量,就是用 read 这个指令 **read [-pt] variable**
    2. declare 或 typeset 是一样的功能,就是在“宣告变量的类型”
       1. 如果使用 declare 后面并没有接 任何参数,那么 bash 就会主动的将所有的变量名称与内容通通叫出来  **declare [-aixr] variable**
       2. 变量类型默认为“字串”,所以若不指定变量类型,则 1+2 为一个“字串”而不是“计算式”。 所以上述第一个执行的结果才会出现那个情况的
       3. bash 环境中的数值运算,默认最多仅能到达整数形态,所以 1/3 结果是 0
       4. sum=100+300+50 echo ${sum} 100+300+50
       5. Declare sum=100+300+50 echo ${sum} 450
    3. 阵列 (array) 变量类型 建议直接以 ${阵列} 的方式来读取
32. 与文件系统及程序的限制关系: ulimit
    1. ulimit [-SHacdfltu] [配额]
    2. 列出你目前身份(假设为一般帐号)的所有限制数据数值 ulimit -a
    3. 限制使用者仅能创建 10MBytes 以下的容量的文件 ulimit -f 10240
33. 变量内容的删除、取代与替换
    1. 变量内容的删除与取代
       1. 在 PATH 这个变量的内容中,每个目录都是以冒号“:”隔开的, 所以 要从头删除掉目录就是介于斜线 (/) 到冒号 (:) 之间的数据!但是 PATH 中不止一个冒号 (:) 啊! 所以 # 与 ## 就分别代表:

符合取代文字的“最短的”那一个; 符合取代文字的“最长的”那一个

1. 想要“从后面向前删除变量内容”呢? 这个 时候就得使用百分比 (%) 符号了
2. ${变量#关键字} ${变量##关键字} 若变量内容从头开始的数据符合“关键字”,则将符合的最短数据 删除 若变量内容从头开始的数据符合“关键字”,则将符合的最 长数据删除
3. ${变量%关键字} ${变量%%关键字} 若变量内容从尾向前的数据符合“关键字”,则将符合的最短数据 删除 若变量内容从尾向前的数据符合“关键字”,则将符合的最 长数据删除
4. ${变量/旧字串/新字串} ${变量//旧字串/新字串} 若变量内容符合“旧字串”则“第一个旧字串会被新字串取代” 若变 量内容符合“旧字串”则“全部的旧字串会被新字串取代”
5. 命令别名设置: alias, unalias
   1. alias 的定义规则与变量定义规则几乎相同 你只要在 alias 后面加上你的 {“别名”='指令 选项...' }, 以后你只要输入 lm 就相当于输入了 ls -al|more 这一串指令
   2. 如果要取消命令别名的话,那么就使用 unalias unalias lm
   3. 命令别名与变量有什么不同呢?命令别名是“新创一个新的指令, 你可以直接下达该指 令”的,至于变量则需要使用类似“ echo ”指令才能够调用出变量的内容! 这两者当然不一 样
6. 历史命令:history
   1. !number 执行第几笔指令的意思;
   2. Ommand command :由最近的指令向前搜寻“指令串开头为 command”的那个指令,并执行;
   3. !! !! : 就是执行上一 个指令(相当于按↑按键后,按 Enter)
7. 路径与指令搜寻顺序
   1. 以相对/绝对路径执行指令,例如“/bin/ls”或“./ls”;
   2. 由alias找到该指令来执行;
   3. 由bash内置的(builtin)指令来执行;
   4. 通过$PATH这个变量的顺序搜寻到的第一个指令来执行。
   5. 如果想要了解指令搜寻的顺序,其实通过 type -a ls 也可以查询的到
8. bash 的进站与欢迎讯息: /etc/issue, /etc/motd
   1. cat /etc/issue
   2. vim /etc/motd 那么当你的使用者(包括所有的一般帐号与 root)登陆主机后,就会显示这样的讯息出来
9. bash 的环境配置文件: 命令别名啦、自订的变量啦,在你 登出 bash 后就会失效,所以你想要保留你的设置, 就得要将这些设置写入配置文件才行.
   1. /etc/profile (login shell 才会读) /etc/profile 可不止会做这些事而已,他还会去调用外部的设置数据
   2. ~/.bash\_profile (login shell 才会读)
   3. source :读入环境配置文件的指令
   4. 利用 source 或小数点 (.) 都可以将配置文件的内容读进来目前的 shell 环境中
   5. ~/.bashrc (non-login shell 会读)
   6. 如果你想要将命令提示字符捉回来,那么可以复制 **/etc/skel/.bashrc** 到你的主文件夹,再修订一下你所想要的内容, 并使用 source 去调用 ~/.bashrc ,那你的命令提示字符就会回来啦
   7. /etc/man\_db.conf
   8. ~/.bash\_history
   9. ~/.bash\_logout
10. 终端机的环境设置: stty, set
    1. stty (setting tty 终端机的意思) 呢! stty 也可以帮助设置终端机的输入按键代表意义 **stty [-a]**
    2. set 还可以帮我们设 置整个指令输出/输入的环境。 例如记录历史命令、显示错误内容等等

**set [-uvCHhmBx]**

1. 万用字符与特殊符号
   1. \* 代表“ 0 个到无穷多个”任意字符
   2. ? 代表“一定有一个”任意字符
   3. [] 同样代表“一定有一个在括号内”的字符(非任意字符)。例如 [abcd] 代表“一定有一 个字符, 可能是 a, b, c, d 这四个任何一个”
   4. [- ] 若有减号在中括号内时,代表“在编码顺序内的所有字符”。例如 [0-9] 代表 0 到 9 之 间的所有数字,因为数字的语系编码是连续的!
   5. [^ ] 若中括号内的第一个字符为指数符号 (^) ,那表示“反向选择”,例如 [^abc] 代表 一定有一个字符,只要是非 a, b, c 的其他字符就接受的意思。
2. 数据流重导向
   1. standard output 指令执行所回传的正确的讯息
   2. standard error output 指令 执行失败后,所回传的错误讯息
   3. 标准输入 (stdin):代码为0,使用<或<<;
   4. 标准输出 (stdout):代码为1,使用>或>>;
   5. 标准错误输出(stderr):代码为2,使用2>或2>>;
   6. 1> :以覆盖的方法将“正确的数据”输出到指定的文件或设备上;
   7. 1>>:以累加的方法将“正确的数据”输出到指定的文件或设备上;
   8. 2> :以覆盖的方法将“错误的数据”输出到指定的文件或设备上;
   9. 2>>:以累加的方法将“错误的数据”输出到指定的文件或设备上;
   10. /dev/null 垃圾桶黑洞设备与特殊写法
   11. standard input : < 与 << 将原本需要由键盘输入的数据,改由文件内容来取代
3. 命令执行的判断依据: ; , &&, ||
   1. cmd ; cmd (不考虑指令相关性的连续指令下达) sync; sync; shutdown -h now
   2. $? (指令回传值) 与 && 或 ||
      1. cmd1 && cmd2 若 cmd1 执行完毕且正确执行($?=0),则开始执行 cmd2。 2. 若 cmd1 执行完毕且为错误 ($?≠0),则 cmd2 不执行
      2. cmd1 || cmd2 若 cmd1 执行完毕且正确执行($?=0),则 cmd2 不执行。 2. 若 cmd1 执 行完毕且为错误 ($?≠0),则开始执行 cmd2
   3. 以 ls 测试 /tmp/vbirding 是否存在,若存在则显示 "exist" ,若不存在,则显示 "not exist"!

**ls /tmp/vbirding && echo "exist" || echo "not exist"**

1. 管线命令 (pipe)
   1. 管线命令仅会处理 standard output,对于 standard error output 会 以忽略
   2. 管线命令必须要能够接受来自前一个指令的数据成为 standard input 继续处理才行
   3. cut -d'分隔字符' -f fields &lt;==用于有特定分隔字符
   4. cut -c 字符区间 &lt;==用于排列整齐的讯息
   5. grep [-acinv] [--color=auto] '搜寻字串' filename
   6. sort [-fbMnrtuk] [file or stdin] eg: sort -k1,1d -k2,2n -T ./tmp $1 > $1.reorder
   7. uniq [-ic]
   8. wc [-lwm]
   9. 双向重导向: tee tee [-a] file tee 可以让 standard output 转存一份到文件内并将同样的数据继续送到屏幕去处理
2. 字符转换命令: tr, col, join, paste, expand
   1. Tr tr 可以用来删除一段讯息当中的文字,或者是进行文字讯息的替换 **tr [-ds] SET1 …**
      1. **tr "[:lower:]" "[:upper:]" < file1** Translate the contents of file1 to upper-case.
      2. **tr -cs "[:alpha:]" "\n" < file1** Create a list of the words in file1, one per line, where a word is taken to be a maximal string of letters
   2. col 可以用来简单的处理将 [tab] 按键取代成为空 白键 **col [-xb]**
   3. join 在使用 join 之前,你所需要处理的文件应该要事先经过排序 (sort) 处理  **join [-ti12] file1 file2**
   4. paste 就 直接“将两行贴在一起,且中间以 [tab] 键隔开” **paste [-d] file1 file2**
   5. Expand 将 [tab] 按键转成空白键  **expand [-t] file** 并不是一个 ^I 就会换成 8 个 空白
   6. split 将一个大文件,依据文件大小或行数来分区,分区成为小文件 **split [-bl] file PREFIX**
3. 参数代换: xargs: x 是加减乘除的乘号,args 则是 arguments (参数) , 就是在产生某个指令的参数
   1. xargs [-0epn] command
   2. 会使用 xargs 的原因是, 很多指令其 实并不支持管线命令,因此我们可以通过 xargs 来提供该指令引用 standard input 之用
4. 关于减号 - 的用途
   1. 在管线命令当中,常常会使用到前一个指令的 stdout 作为这次的 stdin , 某些指令需要用到文件名称 (例如 tar) 来进行处理时,该 stdin 与 stdout 可以利用减号 "-" 来替代
   2. tar -cvf - /home |tar -xvf - -C /tmp/homeback
5. type 可以用来找到执行指令为何种类型,亦可用于与 which 相同的功能
6. 使用 env 与 export 可观察环境变量,其中 export 可以将自订变量转成环境变量
7. bash 的配置文件主要分为 login shell 与 non-login shell。login shell 主要读取 /etc/profile 与 ~/.bash\_profile, non-login shell 则仅读取 ~/.bashrc
8. 在使用 vim 时,若不小心按了 [crtl]+s 则画面会被冻结。你可以使用 [ctrl]+q 来解除冻结
9. Linux 默 认的 shell 是 bash ,亦即是 /bin/bash
10. 想要删除光标所在处到最前 面的指令串内容 按下 [crtl]+u 组合键即可
11. bash 环境配置文件主要分为哪两种类型的读取?分别读取哪些重要文件?(1)login shell:主要读取 /etc/profile 及 ~/.bash\_profile (2)non-logni shell:主要读取 ~/.bashrc 而已
12. 如何将 last 的结果中,独立出帐号,并且印出曾经登陆过的帐号? **last | cut -d ' ' -f1 | sort | uniq**
13. 以正则表达式的方式处理字串,就得要使用支持正则表达式的工具程序才行, 这类的工具 程序很多,例如 vi, sed, awk 等等。 正则表达式就是处理字串的方法,他是以行为单位来进行字串的处理行为, 正则 表达式通过一些特殊符号的辅助,可以让使用者轻易的达到“搜寻/删除/取代”某特定字串的处 理程序 。
14. 万用字符 (wildcard) 代表的是 bash 操作接口的一个功能，但正则表达式则 是一种字串处理的表示方式
15. grep 的一些进阶选项 **grep [-A] [-B] [--color=auto] '搜寻字串' filename**
    1. dmesg | grep 'vfs'
    2. 搜寻特定字串 **grep -n 'the' regular\_express.txt**
    3. 当该行没有 'the' 这个字串时才显示在屏幕上 **grep -vn 'the' regular\_express.txt**
    4. 搜寻不论大小写的 the 这个字串 **grep -in 'the' regular\_express.txt**
    5. 利用中括号 [] 来搜寻集合字符 搜寻 test 或 taste 这两个单字

**grep -n 't[ae]st' regular\_express.txt**

1. 如果想要搜寻到有 oo 的字符 **grep -n 'oo' regular\_express.txt**
2. 如果我不想要 oo 前面有 g 的话 ,可以利用在集合字符的反向选择 来达成

**grep -n '[^g]oo' regular\_express.txt**

1. 假设我 oo 前面不想要有小写字符,所以,我可以这样写 abcd....zoo ,由于小写字符的 ASCII 上编码的顺序是连续的, 可以将之简化为 **grep -n '[^a-z]oo' regular\_express.txt**

**grep -n '[^[:lower:]]oo' regular\_express.txt [:lower:] 代表的就是 a-z 的意思**

1. 当我们在一组集合字符中,如果该字符组是连续的,例如大写英文/小写英文/数字 等等, 就可以使用[a-z],[A-Z],[0-9]等方式来书写,那么如果我们的要求字串是数字与英文, 就将他全部写在一起,变成:[a-zA-Z0-9]. 我们要取得有数字的那一行: **grep -n '[0-9]' regular\_express.txt**

**grep -n '[[:digit:]]' regular\_express.txt**

1. 行首与行尾字符 ^ $ 如果我想要让 the 只在行首列出 要使用定位字符了

**grep -n '^the' regular\_express.txt**

1. 如果我想要开头是 小写字符的那一行就列出 **grep -n '^[a-z]' regular\_express.txt**

**grep -n '^[[:lower:]]' regular\_express.txt**

1. 如果我不想要开头是英文字母 **grep -n '^[^a-zA-Z]'** regular**\_express.txt**

**grep -n '^[^[:alpha:]]' regular\_express.txt**

1. **那个 ^ 符号,在字符集合符号(括号[])之内与之外是不同的! 在 [] 内代表“反向选择”,在 [] 之外则代表定位在行首的意义**
2. 我想要找出 来,行尾结束为小数点 (.) 的那一行 **grep -n '\.$' regular\_express.txt**
3. 因为小数点具有其他意义(下面会介绍),所以必须要使用跳脱字符(\)来加 以解除其特殊意义
4. 如果我想要找出来,哪一行 是“空白行” 因为只有行首跟行尾 (^$),所以,这样就可以找出空白行啦

**grep -n '^$' regular\_express.txt**

1. 省略掉 空白行与开头为 # 的注解行 **grep -n '^$' /etc/rsyslog.conf | grep -v '^#'**
2. 任意一个字符 . 与重复字符 \* . (小数点):代表“一定有一个任意字符”的意思; \* (星星号):代表“重复前一个字符, 0 到无穷多次”的意思,为组合形态.
3. 假设我需要找出 g??d 的字串,亦即共有四个字符, 起头是 g 而结束是 d

**grep -n 'g..d' regular\_express.txt**

1. 如果我想要列出有 oo, ooo, oooo 等等的数据, 也就是说,至少要有两个 (含) o 以上

**grep -n 'ooo\*' regular\_express.txt**

1. 如果我想要字串开头与结尾都是 g,但是两个 g 之间仅能存在至少一个 o ,亦即是 gog, goog, gooog.... 等等 **grep -n 'goo\*g' regular\_express.txt**
2. 如果我想要找出 g 开头与 g 结尾的字串,当中的字符可有可无

**grep -n 'g.\*g' regular\_express.txt**  “.\* 就代表零个或多个任 意字符”的意思

1. 如果我想 要找出“任意数字”的行列 **grep -n '[0-9][0-9]\*' regular\_express.txt**

**grep -n '[0-9]' regular\_express.txt**

1. 限定连续 RE 字符范围 {}
2. 我想要找出两个到五个 o 的连续字 串 **grep -n 'go\{2,5\}g' regular\_express.txt**
3. 如果我想要的是 2 个 o 以上的 goooo....g 呢 **grep -n 'go\{2,\}g' regular\_express.txt**

**grep -n 'gooo\*g' regular\_express.txt**

1. 在正则表达式中,我们要找到 含有以 a 为开头的文件 **ls | grep -n '^a.\*'**
2. 以 ls -l 配合 grep 找出 /etc/ 下面文件类型为链接文件属性的文件名答:由于 ls -l 列出 链接文件时标头会是“ lrwxrwxrwx ”,因此使用如下的指令即可找出结果  **ls -l /etc | grep '^l'**
3. 基础正则表达式字符汇整 (characters)
   1. ^word 意义:待搜寻的字串(word)在行首!范例:搜寻行首为 # 开始的那一行,并 列出行号 > grep -n '^#' regular\_express.txt
   2. word$ 意义:待搜寻的字串(word)在行尾!范例:将行尾为 ! 的那一行打印出来, 并列出行号 > grep -n '!$' regular\_express.txt
   3. . 意义:代表“一定有一个任意字符”的字符!范例:搜寻的字串可以是 (eve) (eae) (eee) (e e), 但不能仅有 (ee) !亦即 e 与 e 中间“一定”仅有 一个字符,而空白字符也是字符! > grep -n 'e.e' regular\_express.txt
   4. \ 意义:跳脱字符,将特殊符号的特殊意义去除!范例:搜寻含有单引号 ' 的那一 行! > grep -n \' regular\_express.txt
   5. \* 意义:重复零个到无穷多个的前一个 RE 字符 范例:找出含有 (es) (ess) (esss) 等等的字串,注意,因为 可以是 0 个,所以 es 也是符合带搜寻字 串。另外,因为 为重复“前一个 RE 字符”的符号, 因此,在 之前必须要紧接着 一个 RE 字符喔!例如任意字符则为 “.” ! > grep -n 'ess\*' regular\_express.txt
   6. [list] 意义:字符集合的 RE 字符,里面列出想要撷取的字符!范例:搜寻含有 (gl) 或 (gd) 的那一行,需要特别留意的是,在 [] 当中“谨代表一个待搜寻 的字符”, 例如“ a[afl]y ”代表搜寻的字串可以是 aay, afy, aly 即 [afl] 代表 a 或 f 或 l 的意思! > grep -n 'g[ld]' regular\_express.txt
   7. [n1- n2] 意义:字符集合的 RE 字符,里面列出想要撷取的字符范围!范例:搜寻含有 任意数字的那一行!需特别留意,在字符集合 [] 中的减号 - 是有特殊意义的, 他代表两个字符之间的所有连续字符!但这个连续与否与 ASCII 编码有关,因 此,你的编码需要设置正确(在 bash 当中,需要确定 LANG 与 LANGUAGE 的变量是否正确!) 例如所有大写字符则为 [A-Z] > grep -n '[A-Z]' regular\_express.txt
   8. [^list] 意义:字符集合的 RE 字符,里面列出不要的字串或范围!范例:搜寻的字串 可以是 (oog) (ood) 但不能是 (oot) ,那个 ^ 在 [] 内时,代表的意义 是“反向选择”的意思。 例如,我不要大写字符,则为 [^A-Z]。但是,需要特别 注意的是,如果以 grep -n [^A-Z] regular\_express.txt 来搜寻,却发现该文件内 的所有行都被列出,为什么?因为这个 [^A-Z] 是“非大写字符”的意思, 因为每 一行均有非大写字符,例如第一行的 "Open Source" 就有 p,e,n,o.... 等等的小 写字 > grep -n 'oo[^t]' regular\_express.txt
   9. {n,m}

意义:连续 n 到 m 个的“前一个 RE 字符” 意义:若为 {n} 则是连续 n 个的前一

个 RE 字符, 意义:若是 {n,} 则是连续 n 个以上的前一个 RE 字符! 范例:在

g 与 g 之间有 2 个到 3 个的 o 存在的字串,亦即 (goog)(gooog)

> grep -n 'go\{2,3\}g' regular\_express.txt

1. sed 可以将数据进行取代、删除、新增、撷取特定行等等的功能
   1. sed [-nefr] [动作]
   2. 以行为单位的新增/删除功能, 将 /etc/passwd 的内容列出并且打印行号,同时,请将第 2~5 行删除 **nl sed.txt | sed '2,5d'**
   3. 至于若是要删除第 3 到最后一行,则是 **nl /etc/passwd | sed '3,$d'** “ $ ”代表最后一行
   4. 在第二行后(亦即是加在第三行)加上“drink tea?”字样 **nl sed.txt | sed '2a drink tea'**
   5. 如果是要在第二行前加上“drink tea?”字样 **nl sed.txt | sed '2i drink tea'**
   6. 在第二行后面加入两行字,例如“Drink tea or .....”与“drink beer?”

**nl sed.txt | sed '2a Drink tea or ......\  
> drink beer ?'**

我们可以新增不只一行喔!可以新增好几行”但是每一行之间都必须要以反 斜线“ \ ”来进行新行的增加

1. 我想将第2-5行的内容取代成为“No 2-5 number”呢 **nl sed.txt | sed '2,5c No 2-5 number'**
2. 仅列出 /etc/passwd 文件内的第 5-7 行 **nl sed.txt | sed -n '5,7p'** 这个 -n 代表的是“安静模式”
3. 部分数据的搜寻并取代的功能, 除了整行的处理模式之外, sed 还可以用行为单位进行部分数据的搜寻并取代的功能 sed 's/要被取代的字串/新的字串/g' , 上表中特殊字体的部分为关键字,请记下来!至于三个斜线分成两栏就是新旧字串的替换
4. **/sbin/ifconfig eth0 | grep 'inet '**

inet addr:10.57.2.15  Bcast:10.57.255.255  Mask:255.255.0.0

1. **/sbin/ifconfig eth0 | grep 'inet ' | sed 's/^.\*inet addr://g'** 10.57.2.15  Bcast:10.57.255.255  Mask:255.255.0.0
2. **/sbin/ifconfig eth0 | grep 'inet ' | sed 's/^.\*inet addr://g' | sed 's/ \*Bcast.\*$//g'**

10.57.2.15

1. 假设我只要 MAN 存在的那几行数据, 但是含有 # 在内的注解我不想要,而且空白行我也不要

**cat /etc/man.config | grep 'MAN'| sed 's/#.\*$//g' | sed '/^$/d'**

1. 直接修改文件内容(危险动作) 利用 sed 将 regular\_express.txt 内每一行结尾若为 . 则换成 !

**sed -i 's/\.$/\!/g' regular\_express.txt**

1. 利用 sed 直接在 regular\_express.txt 最后一行加入“# This is a test”

**sed -i '$a # This is a test' regular\_express.txt**

由于 $ 代表的是最后一行,而 a 的动作是新增,因此该文件最后新增

1. 延伸正则表达式
   1. 我们要去除空白行与行首为 # 的 行列

**grep -v '^$' regular\_express.txt | grep -v '^#' or egrep -v '^$|^#' regular\_express.txt**

1. + 意义:重复“一个或一个以上”的前一个 RE 字符 范例:搜寻 (god) (good) (goood)... 等等的字串。 那个 o+ 代表“一个以上的 o ”所以,下面的执行成果 会将第 1, 9, 13 行列出来。

**egrep -n 'go+d' regular\_express.txt**

1. ? 意义:“零个或一个”的前一个 RE 字符 范例:搜寻 (gd) (god) 这两个字串。 那个 o? 代表“空的或 1 个 o ”所以,上面的执行成果会将第 13, 14 行列出来。 有 没有发现到,这两个案例( 'go+d' 与 'go?d' )的结果集合与 'go\*d' 相同? 想想 看,这是为什么喔!  **egrep -n 'go?d' regular\_express.txt**
2. | 意义:用或( or )的方式找出数个字串 范例:搜寻 gd 或 good 这两个字串,注意,是“或”! 所以,第 1,9,14 这三行都可以被打印出来喔!那如果还想要找出dog 呢?

**egrep -n 'gd|good' regular\_express.txt  egrep -n 'gd|dog' regular\_express.txt**

1. () 意义:找出“群组”字串 范例:搜寻 (glad) 或 (good) 这两个字串,因为 g 与 d 是重复的,所以, 我就可以将 la 与 oo 列于 ( ) 当中,并以 | 来分隔开来, 就可以啦!

**egrep -n 'g(la|oo)d' regular\_express.txt**

1. () + 意义:多个重复群组的判别 范例:将“AxyzxyzxyzxyzC”用 echo 叫出,然后再使 用如下的方法搜寻一下!

**echo 'AxyzxyzxyzxyzC' | egrep 'A(xyz)+C'**

1. 如果你想要查出来文件中含有 ! 与 > 的字行 **grep -n '[!>]' regular\_express.txt**
2. 反向选择这样对否? '[!a-z]'? 呵呵!是错 的呦~要 'a-z 才是对的
3. 格式化打印: printf printf '打印格式' 实际内容
   1. 将刚刚上头数据的文件 (printf.txt) 内容仅列出姓名与成绩:(用 [tab] 分隔)

**printf '%s\t %s\t %s\t %s\t %s\t \n' $(cat printf.txt)**

1. 将上述数据关于第二行以后,分别以字串、整数、小数点来显示

**printf '%10s %5i %5i %5i %8.2f \n' $(cat printf.txt | grep -v Name)**

1. 字符宽度: 12345678 %8.2f意义:00000.00 全部的宽度仅有 8 个字符,整数部分占有 5 个字符,小数点本身 (.) 占一位, 小数点下的位数则有两位
2. 列出 16 进位数值 45 代表的字符为何  **printf '\x45\n'**
3. awk:好用的数据处理工具 awk 是“以行为一次处理的单位”, 而“以字段为最小的处理单 位”
   1. awk '条件类型1{动作1} 条件类型2{动作2} ...' filename
   2. 我们用 last 可以将登陆者的数据取出来, 仅取出前五行 **last -n 5**
   3. 若我想要取出帐号与登陆者的 IP ,且帐号与 IP 之间以 [tab] 隔开 **last -n 5 | awk '{print $1 "\t" $3}'**
   4. awk 的内置变 量 : NF 每一行 ($0) 拥有的字段总数
   5. awk 的内置变 量 : NR 目前 awk 所处理的是“第几行”数据
   6. awk 的内置变 量 : FS 目前的分隔字符,默认是空白键
   7. 列出每一行的帐号(就是 $1);
   8. 列出目前处理的行数(就是 awk 内的 NR 变量)
   9. 并且说明,该行有多少字段(就是 awk 内的 NF 变量)
   10. last -n 5| awk '{print $1 "\t lines: " NR "\t columns: " NF}'
   11. 那假设我要查阅,第三栏小于 10 以下的数据,并且仅列出帐号与第三栏:

cat /etc/passwd | awk '{FS=":"} $3 < 30 {print $1 "\t " $3}'

1. cat /etc/passwd | awk 'BEGIN {FS=":"} $3 < 10 {print $1 "\t " $3}'
2. 如何帮我计算每个人的总额呢?而且我还想要格式化输出. 第一行只是说明, 所以第一行不要进行加总 (NR==1 时处理), 第二行以后就会有加总的情况出现 (NR>=2 以后处理) :

cat pay.txt | awk '**NR==1{**printf "%10s %10s %10s %10s %10s\n",$1,$2,$3,$4,"Total" } NR>=2{total = $2 + $3 + $4; printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total}'

1. awk 的指令间隔:所有 awk 的动作,亦即在 {} 内的动作,如果有需要多个指令辅助时, 可利用分号“;”间隔, 或者直接以 [Enter] 按键来隔开每个指令,例如上面的范例中,鸟哥 共按了三次 [enter]
2. 逻辑运算当中,如果是“等于”的情况,则务必使用两个等号“==”
3. 如果是直接给 一个值,例如变量设置时,就直接使用 = 而已
4. 格式化输出时,在 printf 的格式设置当中,务必加上 \n ,才能进行分行
5. 与 bash shell 的变量不同,在 awk 当中,变量可以直接使用,不需加上 $ 符号
6. awk 的动作内 {} 也是支持 if (条件) 的喔

cat pay.txt | **awk '{if(NR==1)** printf "%10s %10s %10s %10s %10s\n",$1,$2,$3,$4,"Total"}

NR>=2{total = $2 + $3 + $4

printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total}'

1. 文件比对工具
   1. diff
      1. 我们要将 /etc/passwd 处理成为一个新的版本,处理方 式为: 将第四行删除,第六行则取代成为“no six line”,新的文件放置到 /tmp/vitest 里面.

cat /etc/passwd | sed -e '4d' -ed '6c no six line' > /tmp/vitest/passwd

1. **diff [-bBi] from-file to-file**
2. diff passwd.old passwd.new
3. 4d3 <==這裡是說，左邊檔案(/etc/passwd)第四行被刪除 (d)
4. 6c5 <==這裡是說，左邊檔案的第六行被取代成右邊檔案(/tmp/test/passwd)的第五行

1. cmp 主要也是在比对两个文件,他 主要利用“字节”单位去比对.
   1. cmp [-l] file1 file2
   2. 用 cmp 比较一下 passwd.old 及 passwd.new cmp passwd.old passwd.new
2. patch 如果要“升级”呢?就是“将旧的文件升级成为新的文件”时 先比较先旧版本的差异,并将差异档制作成为补丁文件,再由补丁 文件更新旧文件 一般来说,使用 diff 制作出来的比较文件通常使用扩展名为 .patch
3. 那 么如何将旧的文件更新成为新的内容呢 ? 就是将 passwd.old 改成与 passwd.new 相同
   1. mount /dev/sr0 /mnt
   2. rpm -ivh /mnt/Packages/patch-2.\*
   3. cd /tmp ; diff -Naur old/ new/ > test.patch
   4. patch -pN < patch\_file
   5. 将刚刚制作出来的 patch file 用来更新旧版数据 patch -p1 < /tmp/test.patch
   6. 恢复旧文件的内容 patch -R -p0 < passwd.patch
   7. 为什么这里会使用 -p0 呢?因为我们在比对新旧版的数据时是在同一个目录下, 因此不需要 减去目录啦
4. 文件打印准备: pr pr /etc/man\_db.conf
5. grep 与 egrep 在正则表达式里面是很常见的两支程序,其中, egrep 支持更严谨的正则 表达式的语法
6. 基础正则表达式的特殊字符有: \*, ., [], [-], , ^, $ 等!
7. 文件的比对中,可利用 diff 及 cmp 进行比对,其中 diff 主要用在纯文本方面的新旧版本 比对
8. patch 指令可以将旧版数据更新到新版 (主要亦由 diff 创建 patch 的补丁来源文件)
9. grep在单引号内的星号是正则表达式的字符,但我们要找的是星 号,因此需要加上跳脱字符 (\)
10. 找出在 /etc 下面含有星号 (\*) 的文件与内容 grep '\\*' /etc/\*
11. 如果想要连同完整的 /etc 次目录数据下面含有星号 (\*) 的文件与内容 grep '\\*' $(find /etc -type f)
12. 如果要找的是全系统 (/) 目录数据下面含有星号 (\*) 的文件与内容 find / -type f | xargs -n 10 grep '\\*'
    1. 先用 find 去找出文件；
    2. 用 xargs 将这些文件每次丢 10 个给 grep 来作为参数处理；
    3. grep 实际开始搜寻文件内容。
13. 如果要找的是全系统 (/) 目录数据下面含有星号 (\*) 的, 只是想要知道档名 find / -type f | xargs -n 10 grep -l '\\*'
14. 在 /etc 下面,只要含有 XYZ 三个字符的任何一个字符的那一行就列出来 grep [XYZ] /etc/\*
15. 将 /etc/kdump.conf 内容取出后,(1)去除开头为 # 的行 (2)去除空白行 (3)取出 开头为英文字母的那几行 (4)最终统计总行数该如何进行 grep -v '^#' /etc/kdump.conf | grep -v '^$' | grep '^[[:alpha:]]' | wc -l
16. 如何执行这个文件 shell.sh
    1. 直接指令下达: shell.sh 文件必须要具备可读与可执行 (rx) 的权限,然后
    2. 以 bash 程序来执行:通过“ bash shell.sh ”或“ sh shell.sh ”来执行
    3. 变量“PATH”功能:将 shell.sh 放在 PATH 指定的目录内,例如: ~/bin/
    4. 建议务必要将一些重要的环境变量设置好,鸟哥个人认为, PATH 与 LANG (如果有使用到输出相关的信息时) 是当中最重要的
    5. 一个指令的执行成功与 否,可以使用 $? 这个变量来观察
    6. 利用直接执行的方式来执行 script sh showname.sh
    7. 利用 source 来执行脚本:在父程序中执行 source showname.sh
17. 随日期变化:利用 date 进行文件的创建
    1. date --date='2 days ago' +%Y%m%d 20170212
18. test 指令的测试功能
    1. 关于某个文件名的“文件类 型”判断,如 test -e filename 表 示存在否
    2. -e 该“文件名”是否存在?(常用)
    3. -f 该“文件名”是否存在且为文件(file)?(常用)
    4. -d 该“文件名”是否存在且为目录(directory)?(常 用)
19. 利用判断符号 [ ]
    1. 在中括号 [] 内的每个元件都需要有空白键来分隔;
    2. 在中括号内的变量,最好都以双引号括号起来;
    3. 在中括号内的常数,最好都以单或双引号括号起来。
20. Shell script 的默认变量($0, $1...)
    1. /path/to/scriptname opt1 opt2 opt3 opt4
    2. $0 $1 $2 $3 $4
    3. $# :代表后接的参数“个数”,以上表为例这里显示为“ 4 ”;
    4. $@ :代表“ "$1" "$2" "$3" "$4" ”之意,每个变量是独立的(用双引号括起来);
    5. $\* :代表“ "$1<u>c</u>$2<u>c</u>$3<u>c</u>$4" ”,其中 <u>c</u> 为分隔字符,默 认为空白键, 所以本例中代表“ "$1 $2 $3 $4" ”之意。
21. Shell script 的shift：造成参数变数号码偏移，shift 会移动变量,而且 shift 后面可以接数字,代表拿掉最前面的几个参数的意思
    1. echo "Total parameter number is ==> $#"
    2. echo "Your whole parameter is ==> '$@'"
    3. shift # 进行第一次‘一个变数的 shift ’
    4. echo "Total parameter number is ==> $#"
    5. echo "Your whole parameter is ==> '$@'"
    6. shift 3 # 进行第二次‘三个变数的 shift ’
    7. echo "Total parameter number is ==> $#"
    8. echo "Your whole parameter is ==> '$@'"
22. Shell script 的条件判断式
    1. **单层、简单条件判断式**

if [ 条件判断式 ]; then

当条件判断式成立时，可以进行的指令工作内容；

fi <==将 if 反过来写，就成为 fi 啦！结束 if 之意

1. **多重、复杂条件判断式**

**# 一个条件判断，分成功进行与失败进行 (else)**

if [ 条件判断式 ]; then

当条件判断式成立时，可以进行的指令工作内容；

else

当条件判断式不成立时，可以进行的指令工作内容；

fi

**# 多个条件判断 (if ... elif ... elif ... else) 分多种不同情况执行**

if [ 条件判断式一 ]; then

当条件判断式一成立时，可以进行的指令工作内容；

elif [ 条件判断式二 ]; then

当条件判断式二成立时，可以进行的指令工作内容；

else

当条件判断式一与二均不成立时，可以进行的指令工作内容；

fi

1. netstat -tuln
2. Shell script 利用 case ..... esac 判断

**case $變數名稱 in** <==關鍵字為 case ，還有變數前有錢字號

**"第一個變數內容")** <==每個變數內容建議用雙引號括起來，關鍵字則為小括號 )

**程式段**

**;;** <==每個類別結尾使用兩個連續的分號來處理！

**"第二個變數內容")**

**程式段**

**;;**

**\*)** <==最後一個變數內容都會用 \* 來代表所有其他值

**不包含第一個變數內容與第二個變數內容的其他程式執行段**

**exit 1**

**;;**

**esac** <==最終的 case 結尾！『反過來寫』思考一下！

1. 一般来说,使用“ case $变量 in ”这个语法中,当中的那个“ $变量 ”大致有两种取得的方式:
2. 直接下达式:例如上面提到的,利用“ script.sh variable ” 的方式来直接给 $1 这个变量 的内容,这也是在 /etc/init.d 目录下大多数程序的设计方式。
3. 互动式:通过 read 这个指令来让使用者输入变量的内容。
4. /etc/init.d/netconsole restart
5. Shell script 的利用 function 功能

**function fname() {**

**程式段**

**}**

因 为 shell script 的执行方式是由上而下,由左而右, 因此在 shell script 当中的 function 的设置一定要在程序的最前面, 这样才能够在执行时被找到可用的程序段喔

1. Shell script 的loop
   1. **while do done (不定迴圈)**

while [ condition ] <==中括號內的狀態就是判斷式

do <==do 是迴圈的開始！

程式段落

done <==done 是迴圈的結束

1. **until do done (不定迴圈)**

until [ condition ]

do

程式段落

done

1. **for...do...done (固定迴圈)**

for var in con1 con2 con3 ...

do

程式段

done

1. **for...do...done 的數值處理**

for (( 初始值; 限制值; 執行步階 ))

do

程式段

done

1. 11
2. 11

1. How to check linux is 32-bit or 64-bit
   1. cat /proc/version
   2. uname -a
   3. uname -m
   4. arch
   5. file /sbin/init
2. RISC (reduced instruction set computer)-majority
   1. Oracle: SPARC CPU
   2. IBM: PowerPC
   3. ARM: ARM CPU
3. CISC (complex instruction set computer)
   1. AMD: x86
   2. Intel: x86
   3. VIA: x86
4. Units of computer memory measurement
   1. Bit
   2. Byte
   3. Kilo Byte
   4. Mega Byte
   5. Giga Byte
   6. Terra Byte
   7. Peta Byte
   8. Exa Byte
   9. Zetta Byte
   10. Yotta Byte
   11. Bronto Byte
   12. Geop Byte
5. 购买硬盘的容量是十进制，等格式化完后是二进制形式
   1. 硬盘的最小物理单位是512bytes
   2. 最小的组成单位是扇区（sector）
   3. 通常硬盘的计算采用“多少个sector”,所以才采用十进制。
6. 计算机性能
   1. 运行速度
      1. MIPS: Million Instructions Per Second
      2. MHZ: Mega Hertz
      3. GHZ: Giga Hertz
      4. Clock rate:
      5. 将在单位时间（如1秒）内所产生的脉冲个数称为频率。
      6. 频率是描述周期性循环信号（包括脉冲信号）在单位时间内所出现的脉冲数量多少的计量名称
      7. 频率的标准计量单位是Hz（赫）
      8. CPU的主频，即CPU内核工作的时钟频率（CPU Clock Speed）
      9. CPU的频率就是CPU每秒钟可以进行的工作次数。
      10. CPU频率越高表示这颗CPU单位时间内可以做更多的事情
      11. 不同的CPU之间不能单纯的以频率来判断运算性能
      12. 所谓的外频指的是CPU与外部元件进行数据传输时的速度
      13. 倍频则是CPU内部用来加速工作性能的一个倍数
      14. 外频与倍频两者相乘才是CPU的频率速度
      15. 所谓的超频指的是: 将CPU的倍频或者是外频通过主板的设置功能更改成较高频率的一种方式
      16. 在目前32位系统分页大小为4K, 寻址位宽为32位, 用到了32位中的高20位来标记物理页，剩余的12位用于添加一些标记信息等。这样，就可以支持到最大(2^20)\*4096=4GB内存
      17. Intel的turbo技术: CPU会根据当前的任务量自动调整CPU主频，从而重任务时发挥最大的性能，轻任务时发挥最大节能优势
   2. 内存的容量: 内存越大，系统越强，能处理的数据就越庞大
   3. **内存除了频率/带宽与型号需要考虑之外,内存的容量也是很重要**的喔! 因为所有的数据都得 要载入内存当中才能够被CPU判读,如果内存容量不够大的话将会导致某些大容量数据无法 被完整的载入, 此时已存在内存当中但暂时没有被使用到的数据必须要先被释放,使得可用 内存容量大于该数据,那份新数据才能够被载入呢。
   4. 外存的容量 : 硬盘
   5. 字长： 64-bit （since 2009）
7. 超线程 (Hyper-Threading, HT) ，可以有两个程序“同时竞 CPU 的运算单元”,而非通过操作系统的多任务切换! 这一过程就会让 CPU 好像“同时有两个核心”的模样! case by case
8. 个人电脑的内存主要元件为动态随机存取内存 (Dynamic Random Access Memory, **DRAM**), 随机存取内存只有在通电时才能记录与使 用,断电后数据就消失了。因此我们也称这种RAM为挥发性内存 。
9. **DDR**是所谓的双倍数据传送 速度(Double Data Rate), 他可以在一次工作周期中进行两次数据的传送,感觉上就好像是CPU的倍频。
10. **DDR SDRAM**又依据技术的发展,有DDR, DDR2, DDR3, DDR4等等,其中,DDR2 的频率 倍数则是 4 倍而DDR3 则是 8 倍。
11. DDR3 标准电压为 1.5V,但 DDR3L 则仅须 1.35V 喔! 通常可以用在耗电量需求更低的笔记本中! 但并非所有的系统都同步支持! 这就得要看主板的支持规格。
12. 除了内存条之外, 事实上整部个人电脑当中还有许许多多的内存存在喔!最为我们所知的就是CPU内的第二层高速缓存内存。 我们现在知道CPU的数据都是由内存提供,但CPU到内存 之间还是得要通过内存控制器啊! 如果某些很常用的程序或数据可以放置到CPU内部的话, 那么CPU数据的读取就不需要跑到内存重新读取了。
13. 因为第二层高速缓存**(L2 cache**)整合到CPU内部,因此这个L2内存的速度必须要CPU频率 相同。 使用DRAM是无法达到这个频率速度的,此时就需要静态随机存取内存(Static Random Access Memory, **SRAM**)的帮忙了.
14. CMOS主要的 功能为记录主板上面的重要参数, 包括系统时间、CPU电压与频率、各项设备的I/O位址与 IRQ等,由于这些数据的记录要花费电力,因此主板上面才有电池.
15. BIOS为写入到主板上某 一块 flash 或 EEPROM 的程序,他可以在开机的时候执行,以载入CMOS当中的参数, 并尝 试调用储存设备中的开机程序,进一步进入操作系统当中。BIOS程序也可以修改CMOS中的 数据, 每种主板调用BIOS设置程序的按键都不同,一般台式机常见的是使用[del]按键进入 BIOS设置画面
16. 固件(firmware)很多也是使用ROM来进行软件的写入的。 固件像软件一样也是 一个被电脑所执行的程序,然而他是对于硬件内部而言更加重要的部分，很多的硬件上头都会有ROM来写入固件这个软件。现在的 BIOS 通常是写入类似闪存 (flash) 或 EEPROM中
17. 显卡又称为**VGA**(Video Graphics Array),他对于图形影像的显示扮演相当关键的角色。 一 般对于图形影像的显示重点在于分辨率与色彩深度,因为每个图像显示的颜色会占用掉内 存, 因此显卡上面会有一个内存的容量,这个显存容量将会影响到你的屏幕分辨率与色彩深度。显卡主要也是通过CPU的控制芯片来与CPU、内存等沟通。
18. PCIe 3.0 的 16x,因此最大带宽就可以到达接近 32GBytes/s的传输量
19. 假设你的桌面使用1024x768分辨率,且使用全彩(每个像素占用3Bytes的容量),请问你的显卡至少需要多少内存才能使用这样的彩度?答:因为1024x768分辨率中会有786432个像素, 每个像素占用3Bytes,所以总共需要2.25MBytes以上才行。
20. 显卡是通过什么格式与电脑屏幕 (或电视) 连接的呢?目前主要的连接接口有：

D-Sub (**VGA**端子):为较早之前的连接接口,主要为 15 针的连接,为模拟讯号的传 输,当初设计是针对传统图像管屏幕而来

**DVI**: 三种类型包括DVI-A、DVI-D和DVI-I的接口形式。DVI-D只有数字接口，DVI-I有数字和模拟接口，目前应用主要以DVI-D为主，同时DVI-D和DVI-I又有单通道（Single Link）和双通道（Dual Link）之分，我们平时见到的都是单通道版的，双通道版的成本很高

**HDMI**: 高清输入，最高数据传输速度为5Gbps。 相对于 D-sub 与 DVI 仅能传送影像数据,HDMI 可以同时传送影像与声音,因此 被广泛的使用于电视屏幕中!电脑屏幕目前也经常都有支持 HDMI 格式

Display port:与 HDMI 相似的,可以同时传输声音与影像. 目前也是电视机屏幕的主流连接接口喔!

1. 硬盘依据台式机与笔记本电脑而有分为3.5英寸及2.5英寸的大小
2. 磁盘的最小物理储存单位,称之为扇区 (**sector**),那同一个同心圆的扇区 组合成的圆就是所谓的磁道(**track**)。 由于磁盘里面可能会有多个盘片,因此在所有盘片上 面的同一个磁道可以组合成所谓的柱面 (**cylinder**) 。通常数据的读写会由外圈开始往内写的喔!这是默认值啊!
3. 传统磁盘接口包括有 SATA, SAS, IDE 与 SCSI 等等。 若考虑外接式磁盘,那就还 包括了 USB, eSATA 等等接口。
   1. SATA接口：SATA 传输接口传输时,通过的数据演算法的关系,当传输 10 位编码时,仅有 8 位为数 据,其余 2 位为检验之用
   2. SAS接口:
   3. USB接口: USB 3.0 虽然速 度很快,但如果你去市面上面买 USB 的传统磁盘或闪存盘, 其实他的读写速度还是差不多在 100MBytes/s 而已.
4. 文字编码系统
   1. 常用的英文编码表为ASCII系统,这个编码系统中, 每个符号(英文、数字或符号等)都会占 用1Bytes的记录, 因此总共会有28=256种变化
   2. big5码的中文字编码对于某些数据库系统来说是很有问题的,某些字码例如“许、盖、功”等 字, 由于这几个字的内部编码会被误判为单/双引号,在写入还不成问题,在读出数据的对照 表时, 常常就会变成乱码
   3. Unicode编码系统, 我们常常称 呼的UTF8或万国码的编码就是这个咚咚
5. OS Kernel,整部系统只有Kernel的话,我们就只能看着已经准备好运行(Ready)的电脑系统, 但无法操作他! 好像有点望梅止渴的那种感觉啦!这个时候就需要软件的帮忙了
   1. System Call, 既然我的硬件都是由核心管理,那么如果我想要开发软件的话,自然就得要去参考这个核心 的相关功能! 操作系统通常会提供一整组的开发接口给工程师来开发软件.为了保护核心,并且让程序设计师比较容易开发软件,因此**操作系统除了核心程序之外,通 常还会提供一整组开发接口, 那就是系统调用层**
   2. 电脑系统主要由硬件构成,然后核心程序主要在管理硬件,提供合理的电脑系统资源分配 (包括CPU资源、内存使用资源等等), 因此只要硬件不同(如x86架构与RISC架构的 CPU),核心就得要进行修改才行。 而由于核心只会进行电脑系统的资源分配,所以在上头 还需要有应用程序的提供,使用者才能够操作系统的
   3. 核心功能, 既然核心主要是在负责整个电脑系统相关的资源分配与管理,那我们知道其实整部电脑系统 最重要的就是CPU与内存
      1. 系统调用接口(System call interface) , 这是为了方便程序开发者可以轻易 的通过与核心的沟通,将硬件的资源进一步的利用, 于是需要有这个简易的接口来方便 程序开发者
      2. 程序管理(Process control) , 多任务环境 , 核心这个时候必须要能够控制这些工作,让CPU的资 源作有效的分配才行
      3. 内存管理(Memory management) , 控制整个系统的内存管理,这个内存控制是非常重 要的,因为系统所有的程序码与数据都必须要先存放在内存当中。 通常核心会提供虚拟 内存的功能,当内存不足时可以提供内存交换(swap)的功能
      4. 文件系统管理(Filesystem management) 文件系统的管理 ,例如数据的输入输出 (I/O)等等的工作啦!还有不同文件格式的支持啦等等
      5. 设备的驱动(Device drivers) 就如同上面提到的,硬件的管理是核心的主要工作之一, 当然 ,设备的驱动程序就是核心需要做的事情啦! 好在目前都有所谓的“可载入模 块”功能,可以将驱动程序编辑成模块,就不需要重新的编译核心
   4. 操作系统与驱动程序, 驱动程序可以说是操作系统里面相当重要的一环了!不过,硬件可是持续在进步当 中的! 包括主板、显卡、硬盘等等。那么比较晚推出的较新的硬件, 为了克服这个问题,操作系统 通常会提供一个开发接口给硬件开发商, 让他们可以根据这个接口设计可以驱动他们硬件 的“驱动程序”,如此一来,只要使用者安装驱动程序后, 自然就可以在他们的操作系统上面 驱动这块硬件。
6. 操作系统
   1. 操作系统必须要能够驱动硬件,如此应用程序才能够使用该硬件功能
   2. 一般来说,操作系统会提供开发接口,让开发商制作他们的驱动程序
   3. 要使用新硬件功能,必须要安装厂商提供的驱动程序才行
   4. 驱动程序是由厂商提供的,与操作系统开发者无关
7. 电脑的五大单元, 输入单元、输出单元、控制单元、算数逻辑单元、存储单元五大 部分。其中CPU占有控制、算术逻辑单元,存储单元又包含内存与辅助内存
8. CPU的频率, 外频指的是CPU与外部元件进行数据传输时的速度,倍频则是 CPU内部用来加速工作性能的一个倍数, 两者相乘才是CPU的频率速度
9. CPU每次能够处理的数据量称为字组大小(word size),字组大小依据CPU的设计而有 32位与64位。 我们现在所称的电脑是32或64位主要是依据这个 CPU解析的字组大小而 来的
10. 个人电脑的内存主要元件为动态随机存取内存(Dynamic Random Access Memory, DRAM), 至于CPU内部的第二层高速缓存则使用静态随机存取内存(Static Random Access Memory, SRAM)
11. 目前主流的外接卡接口大多为 PCIe 接口,且最新为 PCIe 3.0,单信道速度高达 1GBytes/s
12. 磁盘连接到主板的接口大多为 SATA 或 SAS,目前台式机主流为 SATA 3.0,理论极速可 达 600MBytes/s
13. 常见的文字编码为 ASCII,繁体中文编码主要有 Big5 及 UTF8 两种,目前主流为 UTF8
14. 虽然Linux仅是其核心与核心提供的工具,不过由于核心、核心工具与这些软件开发者提供的 软件的整合, 使得Linux成为一个更完整的、功能强大的操作系统
15. Linux的核心版本编号 :
    1. 3.10.0-123.el7.x86\_64
    2. 主版本.次版本.释出版本-修改版本
16. 所谓的“虚拟化”指的是:在一部实体主机上面仿真出多个逻辑上完全独立的硬件,这个 假的虚拟出来的硬件主机,可以用来安装一部逻辑上完全独立的操作系统! 因此,通过虚拟 化技术,你可以将一部实体主机安装多个同时运行的操作系统 (非多重开机),以达到将硬 件资源完整利用的效果
17. 端点设备, 那就是近年来很热门又流行的树莓派 (Raspberry Pi) 与 香蕉派 (Banana Pi)
18. Tarball/RPM/DPKG/YUM/APT等软件管理员的安装方式
19. 在Windows里面,程序有问题时,如果可能的话先将所有其它程序保存并结束,然后尝 试按救命三键 (Ctrl+Alt+Delete),将有问题的程序(不要选错了程序哦)“结束工作”, 看看能不能恢复系统。不要动不动就直接关机或reset
20. Linux对于计算机各元件/设备的分辨, 与大家惯用的 Windows系统完全不一样!因为,各个元件或设备在Linux下面都是“一个文件!”
21. RAM 内存是越大越好!事实上在Linux服务器中,内存的重要性比CPU还要高的多,建议至少也要512MB， 以上的内存容量较佳。老实说,目前 DDR3 的硬件环境中, 新 购系统动不动就是 4~16GB 的内存,真的是很够用了
22. 磁盘阵列(RAID)是利用硬件技术将数个硬盘整合成为一个大硬盘的方法,操作系 统只会看到最后被整合起来的大硬盘。 由于磁盘阵列是由多个硬盘组成,所以可以达成 速度性能、备份等任务
23. Network Interface Card 网卡是服务器上面最重要的元件之一了!目前的主板大多拥有内 置10/100/1000Mbps的超高速以太网卡。如果你的服务器是网络 I/O 行 为非常频繁的网站, 好一点的Intel/boradcom等公司的网卡应该是比较适合
24. 桌上型(Desktop)Linux系统/含X Window:
    1. 用途:Linux的练习机或办公室(Office)工作机。(一般我们会用到的环境)
    2. CPU:最好等级高一点,例如 Intel I5, I7 以上等级。
    3. RAM:一定要大于1GB比较好!否则容易有图形接口停顿的现象
    4. 网卡:普通的以太网卡就好了
    5. 显卡:使用256MB以上内存的显卡!(入门级的都这个容量以上了)
    6. 硬盘:越大越好,最好有60GB
25. 中型以上Linux服务器
    1. 用途:中小型企业/学校单位的FTP/mail/WWW等网络服务主机
    2. CPU:最好等级高一点,例如 I5, I7 以上的多核心系统
    3. RAM:最好能够大于1GB以上,大于4GB更好
    4. 网卡:知名的broadcom或Intel等厂牌,比较稳定性能较佳
    5. 显卡:如果有使用到图形功能,则一张64MB内存的显卡是需要的
    6. 硬盘:越大越好,如果可能的话,使用磁盘阵列,或者网络硬盘等等的系统架构, 能够具有更稳定安全的传输环境,更佳
    7. 建议企业用计算机不要自行组装,可购买商用服务器较佳, 因为商用服务器已经通 过制造商的散热、稳定性等测试,对于企业来说,会是一个比较好的选择
26. 各硬件设备在Linux中的文件名
    1. SATA接口的硬盘的文件名称即为/dev/sd[a-d] ,其中, 括号内的字母为a-d当中的任意一个, 亦即有/dev/sda, /dev/sdb, /dev/sdc, 及 /dev/sdd这四个文件的意思
    2. 打印机 是/dev/lp0
    3. 软盘 是/dev/fd0
    4. SCSI/SATA/USB硬 盘机 /dev/sd[a-p]
    5. USB闪存盘 /dev/sd[a-p] (与SATA相同)
    6. VirtI/O界面 /dev/vd[a-p] (用于虚拟机内)
    7. 软盘机 /dev/fd[0-7]
    8. 打印机 /dev/lp[0-2] (25针打印机) /dev/usb/lp[0-15] (USB 接口)
    9. 鼠标 /dev/input/mouse[0-15] (通用) /dev/psaux (PS/2界面) /dev/mouse (当前鼠标)
    10. CDROM/DVDROM /dev/scd[0-1] (通用) /dev/sr[0-1] (通用,CentOS 较常见) /dev/cdrom (当前 CDROM)
    11. 磁带机 /dev/ht0 (IDE 界面) /dev/st0 (SATA/SCSI界面) /dev/tape (当前磁带)
    12. IDE硬盘机 /dev/hd[a-d] (旧式系统才有)
27. 虚拟化的软件
    1. 系统是 windows 系列 的话,鸟哥个人推荐你使用 virtualbox 这个软件
    2. 如果你原本就用 Linux 系统,例如 Fedora/Ubuntu 等系列的话,那么建议你使用原本系统内就有的虚拟机管理员来处理即可
    3. 目前 Linux 系统大多使用 KVM 这个虚拟化软件就是了
28. 第一个扇区 512Bytes 会有这两个数据
    1. 主要开机记录区(Master Boot Record, MBR):可以安装开机管理程序的地方,有446 Bytes
    2. 分区表(partition table):记录整颗硬盘分区的状态,有64 Bytes
    3. 硬盘的容量＝主分区的容量＋扩展分区的容量
    4. 扩展分区的容量＝各个逻辑分区的容量之和
    5. 在MBR分区表中最多4个主分区或者3个主分区+1个扩展分区，也就是说扩展分区只能有一个，然后可以再细分为多个逻辑分区。
29. MBR 主要分区、延伸分区与逻辑分区的特性
    1. 主要分区与延伸分区最多可以有四笔(硬盘的限制)
    2. 延伸分区最多只能有一个(操作系统的限制)
    3. 逻辑分区是由延伸分区持续切割出来的分区
30. MBR 分区表除了上述的主分区、延伸分区、逻辑分区需要注意之外,由于每组分区表仅有 16Bytes 而已,因此可纪录的信息真的是相当有限的! 所以,在过去 MBR 分区表的限制中 经常可以发现如下的问题
    1. 操作系统无法抓取到 2.2T 以上的磁盘容量
    2. MBR 仅有一个区块,若被破坏后,经常无法或很难救援
    3. MBR 内的存放开机管理程序的区块仅 446Bytes,无法容纳较多的程序码
31. GUID partition table, GPT 磁盘分区表, GPT 将磁盘所有区块以此 LBA(默认为 512Bytes 喔!) 来规划, 而第一个 LBA 称为 LBA0 (从 0 开始编号)
    1. GPT 使用了 34 个 LBA 区块来纪录分区 信息!同时与过去 MBR 仅有一的区块,被干掉就死光光的情况不同, GPT 除了前面 34 个 LBA 之外,整个磁盘的最后 33 个 LBA 也拿来作为另一个备份
    2. fdisk 这个老牌的软件并不认识 GPT
    3. **类似 gdisk 或者是 parted 指令才 认识 GPT**
    4. grub2 以后才会认识 GPT
32. BIOS 搭配 MBR/GPT 的开机流程
    1. BIOS:开机主动执行的固件,会认识第一个可开机的设备
    2. MBR:第一个可开机设备的第一个扇区内的主要开机记录区块,内含开机管理程序
    3. 开机管理程序(bootloader):一支可读取核心文件来执行的软件
    4. 核心文件:开始操作系统的功能...
33. boot loader的主要任务有下面这些项目
    1. 提供菜单:使用者可以选择不同的开机项目,这也是多重开机的重要功能
    2. 载入核心文件:直接指向可开机的程序区段来开始操作系统
    3. 转交其他loader:将开机管理功能转交给其他loader负责
34. 假设MBR内安装的是可同时认识Windows/Linux操作系统的开机管理程序, 那么整个流 程
    1. 每个分区都拥有自己的开机扇区(boot sector)
    2. 图中的系统盘为第一及第二分区
    3. 实际可开机的核心文件是放置到各分区内的!
    4. loader只会认识自己的系统盘内的可开机核心文件,以及其他loader而已
    5. loader可直接指向或者是间接将管理权转交给另一个管理程序
35. 如果要安装多重开机, 最好先安装Windows再安装 Linux ,为什么 ?
    1. Linux在安装的时候,你可以选择将开机管理程序安装在MBR或各别分区的开机扇区, 而 且Linux的loader可以手动设置菜单(就是上图的M1, M2...),所以你可以在Linux的boot loader里面加入Windows开机的选项
    2. Windows在安装的时候,他的安装程序会主动的覆盖掉MBR以及自己所在分区的开机扇 区,你没有选择的机会, 而且他没有让我们自己选择菜单的功能
    3. 如果先安装Linux再安装Windows的话,那MBR的开机管理程序就只会有Windows的项 目,而不会有Linux的项目 (因为原本在MBR内的Linux的开机管理程序就会被覆盖掉)。 那 需要重新安装Linux一次吗?当然不需要,你只要用尽各种方法来处理MBR的内容即可
36. UEFI BIOS 搭配 GPT 开机的流程
    1. GPT 可以提供到 64bit 的寻址 , 但是 BIOS 其实不懂 GPT 还得要通过 GPT 提供相容模式才能够读写这个磁盘设备 ~而且 BIOS 仅为 16 位的程序 因此就有了 UEFI (Unified Extensible Firmware Interface) 这个统一可延伸固件界 面的产生
    2. 一般来说,使用 UEFI 接口的主机,在开机的速度上要比 BIOS 来的 快上许多
37. 目录树结构 (directory tree) 。 所谓的目录树架构(directory tree)就是以根目录为主,然 后向下呈现分支状的目录结构的一种文件架构
38. 文件系统与目录树的关系(挂载): 所谓的“挂载”就是利用一个目录当成进入点,将磁盘分区的数据放置在该目录下; 也就是 说,进入该目录就可以读取该分区的意思。这个动作我们称为“挂载”,那个进入点的目录我们 称为“挂载点”。 由于整个Linux系统最重要的是根目录,因此根目录一定需要挂载到某个分区 的。 至于其他的目录则可依使用者自己的需求来给 挂载到不同的分区
    1. Linux是将光驱的数据放置到/media/cdrom里头去的
    2. Windows: 桌面\我的计算机\E:\我的文件
39. distributions安装时,挂载点与磁盘分区的规划
    1. 强烈建议使用“自订安装, Custom ”这个安装模式
    2. 在某些Linux distribution中,会将这个模式写的很厉害,叫做是“Expert, 专家模式”
    3. A:初次接触Linux:只要分区“ / ”及“swap”即可:
    4. B:建议分区的方法:预留一个备用的剩余磁盘容量!
    5. 对于首次接触Linux的朋友们,鸟哥通常不建议使用各个distribution所提供默认的Server 安装方式, 选择Server的时候,请“确定”你的硬盘数据是不再需要!因为Linux会自 动的把你的硬盘里面旧有的数据全部杀掉!
40. 如果是将 Linux定位在服务器上面的话,那么Red Hat Enterprise Linux及SuSE Enterprise Linux应该是 很不错的选择
41. 你所下载的文件扩展名是.iso,这就是所谓的image文件(镜像文件)
    1. 这种image文件 是由光盘直接烧录成文件的, 文件非常的大,建议你不要使用浏览器(IE/Firefox..)来下 载,可以使用FTP用户端程序来下载,例如Filezilla
    2. 这种镜像文件可不能以数据格式烧录成为光盘/DVD的!你必须要使用烧录程序的功 能, 将他<u>以“镜像文件格式”烧录成为光盘或DVD才行</u>!切记不要使用烧录数据文件格 式来烧录喔
42. NAT(达成IP分享器的功能)
    1. 在这种环境中,由于Linux作为一个内/外分离的实体,因此网络流量会比较大一点。 此时 Linux主机的网卡就需要比较好些的配备。其他的CPU、RAM、硬盘等等的影响就小很多
    2. Linux NAT还可以额外的加装很多分析软件, 可以用来分析用户端的连线,或者是用来控制带宽与流量,达到更公平的带宽使用
43. SAMBA(加入Windows网络上的芳邻)
44. Mail(邮件服务器) 。在mail server上面,重要的也 是硬盘容量与网卡速度,在此情境中,也可以将/var目录独立出来,并加大容量
45. Web(WWW服务器) CentOS使用的是Apache这套软件来达成WWW网站的功能,在WWW服务器上面,如果你还 有提供数据库系统的话, 那么CPU的等级就不能太低,而最重要的则是RAM了!要增加 WWW服务器的性能,通常提升RAM是一个不错的考虑
46. DHCP(提供用户端自动取得IP的功能)
47. FTP: FTP传输再怎么地下化也是很容易被捉到的
48. 我的Linux主机因为跳电的关系,造成不正常的关机,结果导 致无法开机,这该如何是好?, 幸运一点的可以使用fsck来解决硬盘的问题,麻烦一点 的可能还需要重新安装Linux呢
49. 在默认的CentOS环境中,下面的 目录是比较符合容量大且(或)读写频繁的目录
    1. /boot
    2. /
    3. /home
    4. /var
    5. Swap
50. 提供Linux的PC丛集(Cluster)计算机群
    1. 主机硬件配备
       1. 利用两部多核系统处理器 (一部 20核 40绪,一部 12核 24绪),搭配 10G 网卡组 合而成
       2. 使用内置的显卡
       3. 运算用主机仅一颗磁盘,储存用主机提供 8 颗 2TB 磁盘组成的磁盘阵列
       4. 一部 128GB 内存,一部 96GB 内存
    2. 硬盘分区:
       1. 运算主机方面,整颗磁盘仅分 /boot, / 及 swap 而已
       2. 储存主机方面,磁盘阵列分成两颗磁盘,一颗 100G 给系统用,一颗 12T 给数据 用。系统磁盘用的分区为 /boot, /, /home, /tmp, /var 等分区, 数据磁盘全部容量规划 在同一个分区而已。
       3. 安装最新的 CentOS 7.x 版
51. 开机的流程由:BIOS-->MBR-->-->boot loader-->核心文件
52. boot loader的功能主要有:提供菜单、载入核心、转交控制权给其他loader
53. boot loader可以安装的地点有两个,分别是 MBR 与 boot sector
54. 因为 DVD 实在太慢了,所以,比较聪明的朋友或许会将下载的镜像文件通过类 似 dd 或者是其他烧录软件, 直接烧录到 U盘上面,然后在 BIOS 里面调整成为可携式设备优 先开机的模式,这样就可以使用速度较快的 USB 开机来安装 Linux 了
    1. # 假设你的 USB 设备为 /dev/sdc ,而 ISO 文件名为 centos7.iso 的话:
    2. [root@study ~]# dd if=centos7.iso of=/dev/sdc
55. 如果想要强制使用 GPT 分区表的话
    1. 使用方向键,将图3.2.8的光标移动到“InstallCentOS7”的项目中
    2. 按下键盘的[Tab]按钮,让光标跑到画面最下方等待输入额外的核心参数
    3. 在出现的画面中,输入如下画面的数据(注意,各个项目要有空格,最后一个是光标本   
       身而非底线) 其实重点就是输入“ **inst.gpt** ”这个关键字
56. ext2/ext3/ext4:Linux早期适用的文件系统类型。由于ext3/ext4文件系统多了日志的记 录, 对于系统的复原比较快速。不过由于磁盘容量越来越大,ext 家族似乎有点挡不住了 ~所以除非你有特殊的设置需求,否则近来比较少使用 ext4 项目了!
57. swap:就是磁盘仿真成为内存,由于swap并不会使用到目录树的挂载,所以用swap就 不需要指定挂载点喔
58. BIOS Boot:就是 GPT 分区表可能会使用到的项目,若你使用 MBR 分区,那就不需要 这个项目了
59. xfs:这个是目前 CentOS 默认的文件系统,最早是由大型服务器所开发出来的! 他对于 大容量的磁盘管理非常好,而且格式化的时候速度相当快,很适合当今动不动就是好几 个 TB 的磁盘的环境喔!因此我们主要用这玩意儿
60. vfat:同时被Linux与Windows所支持的文件系统类型。如果你的主机硬盘内同时存在 Windows与Linux操作系统,为了数据的交换, 确实可以创建一个vfat的文件系统喔
61. 现在的内存都够大了,swap 虽然最好还是保持存 在比较好,不过也不需要太大啦!大约 1~2GB 就好了
62. 点选“系统”下的“KDUMP”项目,这个项目主要在处理,当 Linux 系 统因为核心问题导致的死机事件时, 会将该死机事件的内存内数据储存出来的一项特色!不 过,这个特色似乎比较偏向核心开发者在除错之用~如果你有需要的话,也可以启动它
63. How to check hostname in Linux: hostname
64. For MAC: choose Apple menu > System Preferences, then click Sharing. Computer name
65. CentOS7
    1. 包括 root 的密码等等,通通都会被纪 录到 /root/anaconda-ks.cfg 这个文件内
66. 内存压力测试:memtest86, “烧机”的任务 当你组装了一部新的个人计算 机,想要测试这部主机是否稳定时, 就在这部主机上面运行一些比较耗系统资源的程序,让 系统在高负载的情况下去运行一阵子(可能是一天), 去测试稳定性的一种情况,就称为“烧 机”
67. 重新开机,然后依序选 择“Troubleshooting”、 “Run a memory test”的项目
68. 安装笔记本电脑或其他类PC计算机的参数
    1. 只要在安装的时候,告诉安装程序的linux核心不要载入一些特殊功能即可
    2. 最 常使用的方法就是,在使用DVD开机时,选择“”然后按下 [tab] 按键后,加入下面这些选项
    3. **nofb apm=off acpi=off pci=noacpi**
    4. apm(Advanced Power Management)是早期的电源管理模块
    5. acpi(Advanced Configuration and Power Interface)则是近期的电源管理模块
    6. nofb则是取消显卡上面的缓冲内存侦测。因为笔记本电脑的显卡常常是整合型的, Linux安装 程序本身可能就不是很能够侦测到该显卡模块。此时加入nofb将可能使得你的安装过程顺利 一些
69. 安装过程进入分区后,请以“自订的分区模式”来处理自己规划的分区方式
70. 在安装的过程中,可以创建逻辑卷轴管理员 (LVM)
71. 一般要求swap应该要是1.5~2倍的实体内存量,但即使没有swap依旧能够安装与运行 Linux操作系统
72. CentOS 7默认使用 xfs 作为文件系统
73. 没有连上Internet时,可尝试关闭防火墙,但SELinux最好选择“强制”状态
74. 在 Linux 下面,隐藏文件并不是什么特殊的权限, 单纯是因为文件名命名的处 理方式来搞定的
75. 通常我们也称文字模式为终端机接口, terminal 或 console喔!Linux默认的情 况下会提供六个Terminal来让使用者登陆, 切换的方式为使用:[Ctrl] + [Alt] + [F1]~[F6]的 组合按钮
76. 下达指令
    1. command [-options] parameter1 parameter2 ...
    2. 指令 选项 参数(1) 参数(2)
    3. 不论空几格shell都视为一格。所以空格是很重要的特殊字符!
    4. 按下[Enter]按键后,该指令就立即执行。[Enter]按键代表着一行指令的开始启动
    5. 指令太长的时候,可以使用反斜线(\)来跳脱[Enter]符号,使指令连续到下一行。注 意!反斜线后就立刻接特殊字符,才能跳脱
    6. **ls -al ~** is different with **ls -al**
    7. 显示日历的指令: **cal [month] [year**]
    8. 显示目前支持的语系： **Locale**
    9. 简单好用的计算机: bc
    10. 指令求助说明 --help or man \*\*
77. [Tab]按键
    1. [Tab] 接在一串指令的第一个字的后面,则为“命令补全”
    2. [Tab] 接在一串指令的第二个字以后时,则为“文件补齐”!
    3. 若安装 bash-completion 软件,则在某些指令后面使用 [tab] 按键时,可以进行“选项/参 数的补齐功能!
78. [Ctrl]-d 按键 ,键盘输入结 束(End Of File, EOF 或 End Of Input) 另外,他也可以用来取代exit的输入
79. [shift]+{[PageUP]|[Page Down]}按键,可以使用 [Shift]+[Page Up] 来往前翻页,也能够使用 [Shift]+[Page Down] 来往后翻页
80. 在查询数据的后面的数字是有意义的:
    1. **1: 使用者在shell环境中可以操作的指令或可可执行文件**
    2. 2: 系统核心可调用的函数与工具等
    3. 3: 一些常用的函数(function)与函数库(library),大部分为C的函数库(libc)
    4. 4: 设备文件的说明,通常在/dev下的文件
    5. **5: 配置文件或者是某些文件的格式**
    6. 6: 游戏(games)
    7. 7: 惯例与协定等,例如Linux文件系统、网络协定、ASCII code等等的说明
    8. **8: 系统管理员可用的管理指令**
    9. 9: 跟kernel有关的文件
81. 搜寻特定指令/文件的man page说明文档
    1. man -f man
    2. man -k man
    3. 两个特殊指令要能使用,必须要有创建 whatis 数据库才行 : mandb
    4. whatis [指令或者是数据] &lt;==相当于 man -f [指令或者是数据]
    5. apropos [指令或者是数据] &lt;==相当于 man -k [指令或者是数据]
    6. info的说明文档将内容分成多个node,并且每个node都有定位与链接 : **info info**
    7. man, info, /usr/share/doc/
    8. 在终端机模式中,如果你知道某个指令,但却忘记了相关选项与参数,请先善用 --help 的功能来查询相关信息
    9. 当有任何你不知道的指令或文件格式这种玩意儿,但是你想要了解他,请赶快使用man 或者是info来查询!
    10. 而如果你想要架设一些其他的服务,或想要利用一整组软件来达成某项功能时,请赶快 到/usr/share/doc 下面查一查有没有该服务的说明文档喔!
    11. 另外,再次的强调,因为Linux毕竟是外国人发明的,所以中文文件确实是比较少的! 但 是不要害怕,拿本英文字典在身边吧!随时查阅!不要害怕英文喔!
    12. 指数符号(^)代 表的是键盘的[Ctrl]按键
82. 要关机时需要注意下面几 件事
    1. 如果要看目前有谁在线上,可以下达“who”
    2. 如果要看 网络的连线状态,可以下达 “ netstat -a ”
    3. 要看背景执行的程序可以执行“ ps -aux ”这个指令
    4. 将数据同步写入硬盘中的指令: sync
    5. 惯用的关机指令: shutdown
    6. 重新开机,关机: reboot, halt, poweroff
    7. 虽然目前的 shutdown/reboot/halt 等等指令均已经在关机前进行了 sync 这个工具的调用,不 过,多做几次总是比较放心点
    8. 实际使用管理工具 systemctl 关机
       1. init 这个指令可以切换不同 的执行等级~ 执行等级共有 0~6 七个,其中 0 就是关机、6 就是重新开机等等
       2. 面谈到的 halt, poweroff, reboot, shutdown 等等,其实都是调用这个 systemctl 指令
       3. systemctl reboot # 系统重新开机
       4. systemctl poweroff # 系统关机
    9. sync; sync; sync; reboot
    10. Send a warning message: sudo shutdown -k now 'I will shutdown after 10 mins'
83. info page可将一份说明文档拆成多个节点(node)显示,并具有类似超链接的功能,增 加易读性
84. 系统需正确的关机比较不容易损坏,可使用shutdown, poweroff等指令关机
85. 发生乱码可能是因为语系的问题所致。 可以利用 export LANG=en\_US.utf8 或者是 export LC\_ALL=en\_US.utf8 等设置来修订这个问题
86. Linux一般将文件可存取的身份分为三个类 别,分别是 owner/group/others,且三种身份各有 read/write/execute 等权限
87. Linux 使用者身份与群组记录的文件
    1. 默认的情况下,所有的系统上的帐号与一般身份使用者,还有那个 root的相关信息, 都是记录在/etc/passwd这个文件内
    2. 个人的密码则是记录 在/etc/shadow这个文件下
    3. Linux所有的群组名称都纪录在/etc/group内
88. 文件属性
    1. 第一栏代表这个文件的类型与权限(permission)
       1. **第一个字符代表这个文件是“目录、文件或链接文件等等**”
          1. 当为[ d ]则是目录
          2. 当为[ - ]则是文件
             1. 纯文本文件(ASCII)
             2. 二进制档(binary)
             3. 数据格式文件(data)
          3. 若是[ l ]则表示为链接文件(link file);
          4. 若是[ b ]则表示为设备文件里面的可供储存的周边设备(可随机存取设备);
             1. 区块(block)设备文件
             2. 字符(character)设备文件
          5. 若是[ c ]则表示为设备文件里面的序列埠设备,例如键盘、鼠标(一次性读取设 备)
          6. 若是[ s ]则表示数据接口文件(sockets) 最常在/run或/tmp这些个目录中看到这种文件类型
          7. 若是[ p ]则表示FIFO也是一种特殊的文件类型,他主要的目的在解决多个 程序同时存取一个文件所造成的错误问题
       2. 接下来的字符中,以三个为一组,且均为“rwx” 的三个参数的组合
          1. 第一组为“文件拥有者可具备的权限”
          2. 第二组为“加入此群组之帐号的权限”
          3. 第三组为“非本人且没有加入本群组之其他帐号的权限”
          4. [ r ]代表可读 (read)、[ w ]代表可写(write)、[ x ]代表可执行(execute)
          5. ls -l --full-time
          6. 至于other的权限中[r--]虽然有r ,但是由于没有x的权限,因此others的使用者,并不能进 入此目录
    2. 第二栏表示有多少文件名链接到此节点(i-node)
    3. 第三栏表示这个文件(或目录)的“拥有者帐号”
    4. 第四栏表示这个文件的所属群组
    5. 第五栏为这个文件的容量大小,默认单位为Bytes
    6. 第六栏为这个文件的创建日期或者是最近的修改日期
    7. 第七栏为这个文件的文件名, 如果文件名之前多一个“ . ”,则代表这个文件为“隐藏 文件”
89. 如何改变文件属性与权限
    1. chgrp :改变文件所属群组
    2. chown :改变文件拥有者
    3. chmod :改变文件的权限, SUID, SGID, SBIT等等的特性
    4. 改变所属群组, chgrp
    5. 符号类型改变文件权限
       1. (1)user (2)group (3)others三种身份啦!那么我们就可以借由u, g, o来代表三种 身份的权限
       2. chmod a+x filename
       3. r (read):可读取此一文件的实际内容,如读取文本文件的文字内容等;
       4. w (write):可以编辑、新增或者是修改该文件的内容(**但不含删除该文件**);
       5. x (eXecute):该文件具有可以被系统执行的权限
90. 权限对目录的重要性
    1. r (read contents in directory) 表示你 可以查询该目录下的文件名数据
    2. w (modify contents of directory
       1. 创建新的文件与目录
       2. 删除已经存在的文件与目录(不论该文件的权限为何!)
       3. 将已存在的文件或目录进行更名
       4. 搬移该目录内的文件、目录位置。 总之,目录的w权限就与该目录下面的文件名异 动有关就对了啦
    3. x (access directory) 目录的x代表的是使用者能否进入该目录成为工作目录的用途
    4. 创建空文件可用**touch**来 处理
    5. 通常要开放的目录, 至少会具备 rx 这两个权限
91. Linux文件扩展名
    1. \*.sh : 脚本或批处理文件 (scripts),因为批处理文件为使用shell写成的,所以扩展名 就编成 .sh
    2. Z, .tar, .tar.gz, .zip, \*.tgz: 经过打包的压缩文件。这是因为压缩软件分别为 gunzip, tar 等等的,由于不同的压缩软件,而取其相关的扩展名
    3. .html, .php:网页相关文件,分别代表 HTML 语法与 PHP 语法的网页文件
       1. .html 的 文件可使用网页浏览器来直接打开
       2. .php 的文件, 则可以通过 client 端的浏览器来 server 端浏览,以得到运算后的网页结果
92. 从网络上传送到你的 Linux系统中,文件的属性与权限确实是会被改变的
93. Filesystem Hierarchy Standard (FHS)标准
    1. / (root, 根目录):与开机系统有关
    2. /usr (unix software resource):与软件安装/执行有关
    3. /var (variable):与系统运行过程有关
94. FHS定义出根目录(/)下面应该要有下面这些次目录的存在才好
    1. /bin 系统有很多放置可执行文件的目录,但/bin比较特殊。因为/bin放置的是在 单人维护模式下还能够被操作的指令。 在/bin下面的指令可以被root与一般 帐号所使用,主要有:cat, chmod, chown, date, mv, mkdir, cp, bash等等常 用的指令。
    2. /boot 这个目录主要在放置开机会使用到的文件,包括Linux核心文件以及开机菜 单与开机所需配置文件等等。 Linux kernel常用的文件名为:vmlinuz,如果 使用的是grub2这个开机管理程序, 则还会存在/boot/grub2/这个目录喔!
    3. /dev 在Linux系统上,任何设备与周边设备都是以文件的型态存在于这个目录当 中的。 你只要通过存取这个目录下面的某个文件,就等于存取某个设备 ~ 比要重要的文件有/dev/null, /dev/zero, /dev/tty, /dev/loop, /dev/sd等等
    4. /etc 系统主要的配置文件几乎都放置在这个目录内,例如人员的帐号密码档、 各种服务的启始档等等。一般来说,这个目录下的各文件属性是可以让一般 使用者查阅的, 但是只有root有权力修改。FHS建议不要放置可可执行文件 (binary)在这个目录中喔。比较重要的文件有: /etc/modprobe.d/, /etc/passwd, /etc/fstab, /etc/issue 等等。另外 FHS 还规范几个重要的目录 最好要存在 /etc/ 目录下喔:/etc/opt(必要):这个目录在放置第三方协力 软件 /opt 的相关配置文件 /etc/X11/(建议):与 X Window 有关的各种配 置文件都在这里,尤其是 xorg.conf 这个 X Server 的配置文件。 /etc/sgml/ (建议):与 SGML 格式有关的各项配置文件 /etc/xml/(建议):与 XML 格式有关的各项配置文件
    5. /lib 系统的函数库非常的多,而/lib放置的则是在开机时会用到的函数库, 以及 在/bin或/sbin下面的指令会调用的函数库而已。 什么是函数库呢?你可以将 他想成是“外挂”,某些指令必须要有这些“外挂”才能够顺利完成程序的执行 之意。 另外 FSH 还要求下面的目录必须要存在:/lib/modules/:这个目录 主要放置可抽换式的核心相关模块(驱动程序)
    6. /media media是“媒体”的英文,顾名思义,这个/media下面放置的就是可移除的设 备啦! 包括软盘、光盘、DVD等等设备都暂时挂载于此。常见的文件名 有:/media/floppy, /media/cdrom等等
    7. /mnt 如果你想要暂时挂载某些额外的设备,一般建议你可以放置到这个目录中。 在古早时候,这个目录的用途与/media相同啦!只是有了/media之后,这个 目录就用来暂时挂载用了
    8. /opt 这个是给第三方协力软件放置的目录。什么是第三方协力软件啊? 举例来 说,KDE这个桌面管理系统是一个独立的计划,不过他可以安装到Linux系 统中,因此KDE的软件就建议放置到此目录下了。 另外,如果你想要自行 安装额外的软件(非原本的distribution提供的),那么也能够将你的软件安 装到这里来。 不过,以前的Linux系统中,我们还是习惯放置在/usr/local目 录下呢
    9. /run 早期的 FHS 规定系统开机后所产生的各项信息应该要放置到 /var/run 目录 下,新版的 FHS 则规范到 /run 下面。 由于 /run 可以使用内存来仿真,因 此性能上会好很多
    10. /sbin Linux有非常多指令是用来设置系统环境的,这些指令只有root才能够利用 来“设置”系统,其他使用者最多只能用来“查询”而已。 放在/sbin下面的为开 机过程中所需要的,里面包括了开机、修复、还原系统所需要的指令。 至于某些服务器软件程序,一般则放置到/usr/sbin/当中。至于本机自行安装的 软件所产生的系统可执行文件(system binary), 则放置到/usr/local/sbin/ 当中了。常见的指令包括:fdisk, fsck, ifconfig, mkfs等等
    11. /srv srv可以视为“service”的缩写,是一些网络服务启动之后,这些服务所需要 取用的数据目录。 常见的服务例如WWW, FTP等等。举例来说,WWW服 务器需要的网页数据就可以放置在/srv/www/里面。 不过,系统的服务数据 如果尚未要提供给网际网络任何人浏览的话,默认还是建议放置到 /var/lib 下面即可
    12. /tmp 这是让一般使用者或者是正在执行的程序暂时放置文件的地方。 这个目录 是任何人都能够存取的,所以你需要定期的清理一下。当然,重要数据不可 放置在此目录啊! 因为FHS甚至建议在开机时,应该要将/tmp下的数据都 删除
    13. /usr 是Unix Software Resource的缩写, 也就 是“Unix操作系统软件资源”所放置的目录,而不是使用者的数据啦
    14. /var /var目录主要针对常态性变动的文件,包括高速缓存(cache)、登录文件 (log file)以及某些软件运行所产生的文件, 包括程序文件(lock file, run file),或者例如 MySQL数据库的文件等等
    15. /home 这是系统默认的使用者主文件夹(home directory)。在你新增一个一般使 用者帐号时, 默认的使用者主文件夹都会规范到这里来。比较重要的是, 主文件夹有两种代号喔:~:代表目前这个使用者的主文件夹 ~dmtsai :则 代表 dmtsai 的主文件夹
    16. /lib<qual> 用来存放与 /lib 不同的格式的二进制函数库,例如支持 64 位的 /lib64 函数 库等
    17. /root 系统管理员(root)的主文件夹。之所以放在这里,是因为如果进入单人维 护模式而仅挂载根目录时, 该目录就能够拥有root的主文件夹,所以我们会 希望root的主文件夹与根目录放置在同一个分区中
    18. /lost+found 这个目录是使用标准的ext2/ext3/ext4文件系统格式才会产生的一个目录, 目的在于当文件系统发生错误时, 将一些遗失的片段放置到这个目录下。 不过如果使用的是 xfs 文件系统的话,就不会存在这个目录了
    19. /proc 这个目录本身是一个“虚拟文件系统(virtual filesystem)”喔!他放置的数 据都是在内存当中, 例如系统核心、行程信息(process)、周边设备的 状态及网络状态等等。因为这个目录下的数据都是在内存当中, 所以本身 不占任何硬盘空间啊!比较重要的文件例如:/proc/cpuinfo, /proc/dma, /proc/interrupts, /proc/ioports, /proc/net/\* 等等
    20. /sys 这个目录其实跟/proc非常类似,也是一个虚拟的文件系统,主要也是记录 核心与系统硬件信息较相关的信息。 包括目前已载入的核心模块与核心侦 测到的硬件设备信息等等。这个目录同样不占硬盘容量喔
    21. /usr/bin/ 所有一般用户能够使用的指令都放在这里!目前新的 CentOS 7 已经将 全部的使用者指令放置于此,而使用链接文件的方式将 /bin 链接至此! 也就是说, /usr/bin 与 /bin 是一模一样了!另外,FHS 要求在此目录 下不应该有子目录!
    22. /usr/lib/ 基本上,与 /lib 功能相同,所以 /lib 就是链接到此目录中的!
    23. /usr/local/ 系统管理员在本机自行安装自己下载的软件(非distribution默认提供 者),建议安装到此目录, 这样会比较便于管理。举例来说,你的 distribution提供的软件较旧,你想安装较新的软件但又不想移除旧版, 此时你可以将新版软件安装于/usr/local/目录下,可与原先的旧版软件 有分别啦! 你可以自行到/usr/local去看看,该目录下也是具有bin, etc, include, lib...的次目录喔
    24. /usr/sbin/ 非系统正常运行所需要的系统指令。最常见的就是某些网络服务器软件 的服务指令(daemon) !不过基本功能与 /sbin 也差不多, 因此目 前 /sbin 就是链接到此目录中的
    25. /usr/share/ 主要放置只读架构的数据文件,当然也包括共享文件。在这个目录下放 置的数据几乎是不分硬件架构均可读取的数据, 因为几乎都是文字文 件嘛!在此目录下常见的还有这些次目录:/usr/share/man:线上说明 文档 /usr/share/doc:软件杂项的文件说明 /usr/share/zoneinfo:与时 区有关的时区文件
    26. /usr/games/ 与游戏比较相关的数据放置处
    27. /usr/include/ c/c++等程序语言的文件开始(header)与包含档(include)放置处, 当我们以tarball方式 (\*.tar.gz 的方式安装软件)安装某些数据时,会 使用到里头的许多包含档喔!
    28. /usr/libexec/ 某些不被一般使用者惯用的可执行文件或脚本(script)等等,都会放 置在此目录中。例如大部分的 X 窗口下面的操作指令, 很多都是放在 此目录下的
    29. /usr/lib<qual>/ 与 /lib<qual>/功能相同,因此目前 /lib<qual> 就是链接到此目录中
    30. /usr/src/ 一般源代码建议放置到这里,src有source的意思。至于核心源代码则 建议放置到/usr/src/linux/目录下
    31. /var/cache/ 应用程序本身运行过程中会产生的一些暂存盘;
    32. /var/lib/ 程序本身执行的过程中,需要使用到的数据文件放置的目录。在此目录下 各自的软件应该要有各自的目录。 举例来说,MySQL的数据库放置 到/var/lib/mysql/而rpm的数据库则放到/var/lib/rpm去!
    33. /var/lock/ 某些设备或者是文件资源一次只能被一个应用程序所使用,如果同时有两 个程序使用该设备时, 就可能产生一些错误的状况,因此就得要将该设备 上锁(lock),以确保该设备只会给单一软件所使用。 举例来说,烧录机 正在烧录一块光盘,你想一下,会不会有两个人同时在使用一个烧录机烧 片? 如果两个人同时烧录,那片子写入的是谁的数据?所以当第一个人在 烧录时该烧录机就会被上锁, 第二个人就得要该设备被解除锁定(就是前 一个人用完了)才能够继续使用 。目前此目录也已经挪到 /run/lock 中
    34. /var/log/ 重要到不行!这是登录文件放置的目录!里面比较重要的文件 如/var/log/messages, /var/log/wtmp(记录登陆者的信息)等
    35. /var/mail/ 放置个人电子邮件信箱的目录,不过这个目录也被放置到/var/spool/mail/ 目录中! 通常这两个目录是互为链接文件啦!
    36. /var/run/ 某些程序或者是服务启动后,会将他们的PID放置在这个目录下喔!至于 PID的意义我们会在后续章节提到的。 与 /run 相同,这个目录链接到 /run 去了!
    37. /var/spool/ 这个目录通常放置一些伫列数据,所谓的“伫列”就是排队等待其他程序使 用的数据啦! 这些数据被使用后通常都会被删除。举例来说,系统收到新 信会放置到/var/spool/mail/中, 但使用者收下该信件后该封信原则上就会 被删除。信件如果暂时寄不出去会被放到/var/spool/mqueue/中, 等到被 送出后就被删除。如果是工作调度数据(crontab),就会被放置 到/var/spool/cron/目录中!
95. 通过uname 检查linux核心与操作系统的位数
    1. uname -r # 查看核心版本
    2. uname -m # 查看操作系统的位版本
    3. 假设你的 CentOS 7 确实有网络可以使用的情况下 （使用root的身份）
    4. lsb\_release -a
96. 若需要root的权限时,可以使用 su - 这个指令来切换身份。处理完毕则使用 exit 离开 su 的指令环
97. 更改文件的群组支持可用chgrp,修改文件的拥有者可用chown,修改文件的权限可用 chmod
98. chmod修改权限的方法有两种,分别是符号法与数字法,数字法中r,w,x分数为4,2,1
    1. r:可读取此一文件的实际内容,如读取文本文件的文字内容等
    2. w:可以编辑、新增或者是修改该文件的内容(但不含删除该文件)
    3. x:该文件具有可以被系统执行的权限
99. 要开放目录给任何人浏览时,应该至少也要给 r及x的权限,但w权限不可随便给 。能否读取到某个文件内容,跟该文件所在的目录权限也有关系 (目录至少需要有 x 的权 限)。对目录来说,权限的性能为
    1. r (read contents in directory)
    2. w (modify contents of directory)
    3. x (access directory)
100. FHS订定出来的四种目录特色为:shareable, unshareable, static, variable等四类; FHS所定义的三层主目录为:/, /var, /usr三层而已
101. 当一个一般文件权限为 -rwxrwxrwx 则表示这个文件的意义为?任何人皆可读取、修改或 编辑、可以执行,**但不一定能删除**
102. 下面的目录与主要放置什么数据
     1. /etc/:几乎系统的所有设置文件均在此,尤其 passwd,shadow
     2. /boot:开机配置文件,也是默认摆放核心 vmlinuz 的地方
     3. /usr/bin, /bin:一般可执行文件摆放的地方
     4. /usr/sbin, /sbin:系统管理员常用指令集
     5. /dev:摆放所有系统设备文件的目录
     6. /var/log:摆放系统登录文件的地方

/run:CentOS 7 以后才有,将经常变动的项目(每次开机都不同,如程序的PID)

移动到内存暂存,所以 /run 并不占实际磁盘容量

1. 目录的相关操作
   1. . 代表此层目录
   2. 代表上一层目录
   3. 代表“目前使用者身份”所在的主文件夹
   4. ~account 代表 account 这个使用者的主文件夹(account是个帐号名称)
   5. cd是Change Directory的缩写,这是用来变换工作目录的指令
   6. 如果 仅输入 cd 时,代表的就是“ cd ~ ”的意思喔~ 亦即是会回到自己的主文件夹
   7. pwd是Print Working Directory的缩写
   8. pwd -P 可以让我们取得正确的目录名称,而不是以链接文件的路 径来显示的
   9. mkdir (make directory) , 我们可以利用 -m 来强制给 一个新的目录相关的权限
   10. mkdir -p 依 序的创建起目录!并且, 如果该目录本来就已经存在时,系统也不会显示错误讯息喔 .不过鸟哥不建议常用-p这个选项,因为担心如果你打错字,那么目录名称就会 变的乱七八糟的
   11. 如果要将所有目录 下的东西都杀掉呢?! 这个时候就必须使用“ rm -r test ” !不过,还是使用 rmdir 比较不危险!你也可以尝试以 -p 的选项加入,来删除上层的目录喔
   12. PATH(一定是大写)这个变量的内容是由一堆目录所组成的,每个目录中间用冒号(:)来隔 开, 每个目录是有“顺序”之分的
   13. **cp -u 这个 -u 的特性,是在目标文件与来源文件有差异时,才会复制的**。所以,比较常被用于“备份”的工作当中
2. 关于可执行文件路径的变量: $PATH
   1. 不同身份使用者默认的PATH不同,默认能够随意执行的指令也不同(如root与 dmtsai)
   2. PATH是可以修改的
   3. 使用绝对路径或相对路径直接指定某个指令的文件名来执行,会比搜寻PATH来的正确
   4. 指令应该要放置到正确的目录下,执行才会比较方便
   5. 本目录(.)最好不要放到PATH当中
   6. 链接文件(link file) ; 实体链接(hard link) ; 符号链接(symbolic link)
3. cp 有种种的文件属性与权限的特性
   1. 是否需要完整的保留来源文件的信息?
   2. 来源文件是否为链接文件 (symbolic link file)
   3. 来源文件是否为特殊的文件,例如 FIFO, socket 等
   4. 来源文件是否为目录
4. 文件内容查阅
   1. cat 由第一行开始显示文件内容
   2. tac 从最后一行开始显示,可以看出 tac 是 cat 的倒着写!
   3. nl 显示的时候,顺道输出行号!
   4. more 一页一页的显示文件内容
   5. less 与 more 类似,但是比 more 更好的是,他可以往前翻页!
   6. head 只看头几行
   7. tail 只看尾巴几行
   8. od 以二进制的方式读取文件内容!
5. 直接检视文件内容
   1. cat 是 Concatenate (连续) 的简写
   2. tac (反向列示)
   3. nl (添加行号打印)
   4. more (一页一页翻动)
      1. 空白键 (space):代表向下翻一页;
      2. Enter :代表向下翻“一行”;
      3. /字串 :代表在这个显示的内容当中,向下搜寻“字串”这个关键字;
      4. :f :立刻显示出文件名以及目前显示的行数;
      5. q :代表立刻离开 more ,不再显示该文件内容。
      6. b 或 [ctrl]-b :代表往回翻页,不过这动作只对文件有用,对管线无用。
   5. less (一页一页翻动)
      1. 空白键 :向下翻动一页;
      2. [pagedown]:向下翻动一页;
      3. [pageup] :向上翻动一页;
      4. /字串 :向下搜寻“字串”的功能;
      5. ?字串 :向上搜寻“字串”的功能;
      6. n :重复前一个搜寻 (与 / 或 ? 有关!)
      7. N :反向的重复前一个搜寻 (与 / 或 ? 有关!)
      8. g :前进到这个数据的第一行去;
      9. G :前进到这个数据的最后一行去 (注意大小写);
      10. q :离开 less 这个程序;
   6. head (取出前面几行)
   7. tail (取出后面几行)
   8. 两个指令中间有个管线 (|) 的符号存在,这个管线的意思是:“前面的指令所输出的讯 息,请通过管线交由后续的指令继续使用”的意思
6. 非纯文本文件: od 。由 于可执行文件通常是 binary file ,使用上头提到的指令来读取他的内容时, 确实会产生类似 乱码的数据啊!那怎么办?没关系,我们可以利用 od 这个指令来读取喔
7. 修改文件时间或创建新文件: touch
   1. modification time (mtime): 当该文件的“内容数据”变更时,就会更新这个时间!内 容数据指的是文件的内容,而不是文件的属性或权限喔
   2. status time (ctime): 当该文件的“状态 (status)”改变时,就会更新这个时间,举 例来说,像是权限与属性被更改了,都会更新这个时间啊
   3. access time (atime): 当“该文件的内容被取用”时,就会更新这个读取时间 (access)。举例来说,我们使用 cat 去读取 /etc/man\_db.conf , 就会更新该文件的 atime 了
8. touch 这个指令最常被使用的情况是
   1. 创建一个空的文件;
   2. 将某个文件日期修订为目前 (mtime 与 atime)
9. 文件默认权限:umask, umask 就是指定 “目前使用者在创建文件或目录时 候的权限默认值"
   1. 一种可以直接输入 umask ,就可以看到数字体态的权限设置分数
   2. 一种 则是加入 -S (Symbolic) 这个选项,就会以符号类型的方式来显示出权限了
   3. 若使用者创建为“文件”则默认“没有可执行( x )权限”,亦即只有 rw 这两个项目,也就 是最大为 666 分,默认权限如下: -rw-rw-rw-
   4. 若使用者创建为“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,亦即为 777 分,默认权限如下: drwxrwxrwx
   5. 创建文件时:(-rw-rw-rw-) - (-----w--w-) ==> -rw-r--r--
   6. 创建目录时:(drwxrwxrwx) - (d----w--w-) ==> drwxr-xr-x
   7. 这个文件的权限 为:“---------- 1 root root”,意思是这个文件仅有root可读且仅有root可以强制写入而已
10. 文件隐藏属性
    1. chattr (设置文件隐藏属性) , 由于这些属性是隐藏的性质,所以需 要以 lsattr 才能看到该属性. 其中,个人认为最重要的当属 +i 与 +a 这个属性了。+i 可以让 一个文件无法被更动,对于需要强烈的系统安全的人来说, 真是相当的重要的
    2. lsattr (显示文件隐藏属性) , 使用 chattr 设置后,可以利用 lsattr 来查阅隐藏的属性。不过, 这两个指令在使用上必须要 特别小心,否则会造成很大的困扰
11. 文件特殊权限: SUID, SGID, SBIT
    1. Set UID 当 s 这个标志出现在文件拥有者的 x 权限上时 此时就被称为 Set UID,简称为 SUID 的特殊权限
       1. SUID 权限仅对二进制程序(binary program)有效;
       2. 执行者对于该程序需要具有 x 的可执行权限;
       3. 本权限仅在执行该程序的过程中有效 (run-time);
       4. 执行者将具有该程序拥有者 (owner) 的权限。
    2. Set GID 当 s 标志在文件拥有者的 x 项目为 SUID,那 s 在群组的 x 时则称为 Set GID, SGID
       1. 与 SUID 不同的是,SGID 可以针对文件或目录来设置!如果是对文件来说, SGID 有如下的 功能:
       2. SGID 对二进制程序有用;
       3. 程序执行者对于该程序来说,需具备 x 的权限;
       4. 执行者在执行的过程中将会获得该程序群组的支持!
       5. 当一个目录设置了 SGID 的权限后,他将具有如下的功能:
       6. 使用者若对于此目录具有 r 与 x 的权限时,该使用者能够进入此目录;
       7. 使用者在此目录下的有效群组(effective group)将会变成该目录的群组;
       8. 用途:若使用者在此目录下具有 w 的权限(可以新建文件),则使用者所创建的新文 件,该新文件的群组与此目录的群组相同。
    3. Sticky Bit , SBIT 对于目录的作用 是:
       1. 当使用者对于此目录具有 w, x 权限,亦即具有写入的权限时;
       2. 当使用者在该目录下创建文件或目录时,仅有自己与 root 才有权力删除该文件
    4. SUID/SGID/SBIT 权限设置
       1. 4 为 SUID
       2. 2 为 SGID
       3. 1 为 SBIT
       4. 大写的 S, T 代表的就 是“空的”啦
       5. 你也可以通过符号法来处理喔!其中 SUID 为 u+s ,而 SGID 为 g+s , SBIT 则是 o+t
12. 观察文件类型:file 可以利用 file 这个指令 来检阅是属于 ASCII 或者是 data 文件,或者是 binary , 且其中有没有使用到动态函数库 (share library) 等等的信息
    1. file filename
13. 指令与文件的搜寻
    1. which (寻找“可执行文件”) 且 which 后面接的是“完整文件名”喔!若加上 -a 选项,则可以 列出所有的可以找到的同名可执行文件,而非仅显示第一个而已
    2. whereis 或者是 locate 两者就相当的快速, 并且没有实 际的搜寻硬盘内的文件系统状态,比较省时间
    3. whereis (由一些特定的目录中寻找文件文件名) 只找系统中某些特定目录下 面的文件而已 whereis 主要是针对 /bin /sbin 下面的可执行文件, 以及 /usr/share/man 下面的 man page 文 件,跟几个比较特定的目录来处理而已
    4. locate / updatedb 则是利用数据库来搜寻文件名 这个 locate 的使用更简单,直接在后面输入“文件的部分名称”后,就能够得到结果
       1. updatedb:根据 /etc/updatedb.conf 的设置去搜寻系统硬盘内的文件名,并更新 /var/lib/mlocate 内的数据库文件
       2. locate:依据 /var/lib/mlocate 内的数据库记载,找出使用者输入的关键字文件名。
       3. find
       4. 一个文件记录的时间参数有三种,分别是 **access time(atime), status time (ctime), modification time(mtime)**,ls 默认显示的是 mtime。
       5. find atime, ctime 与 mtime 的意义
          1. find / -name passwd 找出文件名为 passwd 这个文件
          2. find / -name "\*passwd\*" 找出文件名包含了 passwd 这个关键字的文件
          3. find /run -type s 找出 /run 目录下,文件类型为 Socket 的文件名有哪些?
          4. # find / -perm /7000 搜寻文件当中含有 SGID 或 SUID 或 SBIT 的属性
          5. find / -size +1M 找出系统中,大于 1MB 的文件
          6. {} 代表的是“由 find 找到的内容”,如上图所示,find 的结果会被放置到 {} 位置中;
          7. -exec 一直到 \; 是关键字,代表 find 额外动作的开始 (-exec) 到结束 (\;) ,在这中 间的就是 find 指令内的额外动作。 在本例中就是“ ls -l {} ” !
          8. 因为“ ; ”在 bash 环境下是有特殊意义的,因此利用反斜线来跳脱。
14. 让使用者能进入某目录成为“可工作目录”的基本权限为何
    1. 可使用的指令:例如 cd 等变换工作目录的指令;
    2. 目录所需权限:使用者对这个目录至少需要具有 x 的权限
    3. 额外需求:如果使用者想要在这个目录内利用 ls 查阅文件名,则使用者对此目录还需要 r 的权限。
15. 使用者在某个目录内读取一个文件的基本权限为何?
    1. 可使用的指令:例如本章谈到的 cat, more, less等等
    2. 目录所需权限:使用者对这个目录至少需要具有 x 权限;
    3. 文件所需权限:使用者对文件至少需要具有 r 的权限才行!
16. 让使用者可以修改一个文件的基本权限为何
    1. 可使用的指令:例如 nano 或未来要介绍的 vi 编辑器等;
    2. 目录所需权限:使用者在该文件所在的目录至少要有 x 权限;
    3. 文件所需权限:使用者对该文件至少要有 r, w 权限
17. 让一个使用者可以创建一个文件的基本权限为何?
    1. 目录所需权限:使用者在该目录要具有 w,x 的权限,重点在 w 啦!
18. 让使用者进入某目录并执行该目录下的某个指令之基本权限为何?
    1. 目录所需权限:使用者在该目录至少要有 x 的权限;
    2. 文件所需权限:使用者在该文件至少需要有 x 的权限
19. 搜寻指令的完整文件名可用 which 或 type ,这两个指令都是通过 PATH 变量来搜寻文件 名;
20. 搜寻文件的完整文件名可以使用 whereis 找特定目录或 locate 到数据库去搜寻,而不实 际搜寻文件系统;
21. 利用 find 可以加入许多选项来直接查询文件系统,以获得自己想要知道的文件名。
22. 找出 /etc 下面,文件大小介于 50K 到 60K 之间的文件,并且将权限完整的列出 (ls - l):
    1. find /etc -size +50k -a -size -60k -exec ls -l {} \; 注意到 -a ,那个 -a 是 and 的意 思,为符合两者才算成功
23. 找出 /etc 下面,文件大小大于 50K 且文件所属人不是 root 的文件名,且将权限完整的列 出 (ls -l)
    1. find /etc -size +50k -a ! -user root -exec ls -ld {} \; find /etc -size +50k -a ! - user root -type f -exec ls -l {}
24. 找出 /etc 下面,容量大于 1500K 以及容量等于 0 的文件
    1. find /etc -size +1500k -o -size 0 相对于 -a ,那个 -o 就是或 (or) 的意思
25. 数据储存与读取的重点在于盘片,而盘片上的物理组成则为
    1. 扇区(Sector)为最小的物理储存单位,且依据磁盘设计的不同,目前主要有 512Bytes 与 4K 两种格式;
    2. 将扇区组成一个圆,那就是柱面(Cylinder)
    3. 早期的分区主要以柱面为最小分区单位,现在的分区通常使用扇区为最小分区单位(每 个扇区都有其号码喔,就好像座位一样)
    4. 磁盘分区表主要有两种格式,一种是限制较多的 MBR 分区表,一种是较新且限制较少的 GPT 分区表
    5. MBR 分区表中,第一个扇区最重要,里面有:(1)主要开机区(Master boot record, MBR)及分区表(partition table), 其中 MBR 占有 446 Bytes,而 partition table 则占 有 64 Bytes
    6. GPT 分区表除了分区数量扩充较多之外,支持的磁盘容量也可以超过 2TB
26. 实体磁盘及虚拟磁盘
    1. /dev/sd[a-p][1-128]:为实体磁盘的磁盘文件名
    2. /dev/vd[a-d][1-128]:为虚拟磁盘的磁盘文件名
27. 文件系统通常会将这两部份的数据分别存放在不同 的区块,权限与属性放置到 inode 中,至于实际数据则放置到 data block 区块中。 另外,还 有一个超级区块 (superblock) 会记录整个文件系统的整体信息,包括 inode 与 block 的总 量、使用量、剩余量等. 每个 inode 与 block 都有编号,至于这三个数据的意义可以简略说明如下:
    1. superblock:记录此 filesystem 的整体信息,包括inode/block的总量、使用量、剩余量, 以及文件系统的格式与相关信息等;
    2. inode:记录文件的属性,一个文件占用一个inode,同时记录此文件的数据所在的 block 号码;
    3. block:实际记录文件的内容,若文件太大时,会占用多个 block 。
    4. 索引式文件系统(indexed allocation)
    5. U盘使用的文件系统一般 为 FAT 格式
    6. Ext2 文件系统在格式化的时候基本上是区分为多个区块群组 (block group) 的,每个区块群组都有独立的 inode/block/superblock 系统
28. 至于每一个区块群组(block group)的六个主要内容说明如后
    1. data block (数据区块) data block 是用来放置文件内容数据地方,在 Ext2 文件系统中所支持的 block 大小有 1K, 2K 及 4K 三种而已
       1. 原则上,block 的大小与数量在格式化完就不能够再改变了(除非重新格式化);
       2. 每个 block 内最多只能够放置一个文件的数据;
       3. 承上,如果文件大于 block 的大小,则一个文件会占用多个 block 数量;
       4. 承上,若文件小于 block ,则该 block 的剩余容量就不能够再被使用了(磁盘空间会浪 费)
    2. inode table (inode 表格)
       1. 该文件的存取模式(read/write/excute);
       2. 该文件的拥有者与群组(owner/group);
       3. 该文件的容量;
       4. 该文件创建或状态改变的时间(ctime);
       5. 最近一次的读取时间(atime);
       6. 最近修改的时间(mtime);
       7. 定义文件特性的旗标(flag),如 SetUID...;
       8. 该文件真正内容的指向 (pointer);
       9. 每个 inode 大小均固定为 128 Bytes (新的 ext4 与 xfs 可设置到 256 Bytes);
       10. 每个文件都仅会占用一个 inode 而已;
       11. 承上,因此文件系统能够创建的文件数量与 inode 的数量有关;
       12. 系统读取文件时需要先找到 inode,并分析 inode 所记录的权限与使用者是否符合,若符 合才能够开始实际读取 block 的内容。
    3. Superblock (超级区块) Superblock 是记录整个 filesystem 相关信息的地方, 没有 Superblock ,就没有这个 filesystem 了。他记录的信息主要有:
       1. block 与 inode 的总量;
       2. 未使用与已使用的 inode / block 数量;
       3. block 与 inode 的大小 (block 为 1, 2, 4K,inode 为 128Bytes 或 256Bytes);
       4. filesystem 的挂载时间、最近一次写入数据的时间、最近一次检验磁盘 (fsck) 的时间 等文件系统的相关信息;
       5. 一个 valid bit 数值,若此文件系统已被挂载,则 valid bit 为 0 ,若未被挂载,则 valid bit 为1。
    4. Filesystem Description (文件系统描述说明)
       1. 这个区段可以描述每个 block group 的开始与结束的 block 号码,以及说明每个区段 (superblock, bitmap, inodemap, data block) 分别介于哪一个 block 号码之间
    5. block bitmap (区块对照表)
       1. 记录的是使用与未使用的 block 号码
    6. inode bitmap (inode 对照表)
       1. 记录使用与未使用的 inode 号码
    7. dumpe2fs: 查询 Ext 家族 superblock 信息的指令 可以使用 **dumpe2fs** 这个指令来查询的
29. 目录 当我们在 Linux 下的文件系统创建一个目录时,文件系统会分配一个 inode 与至少一块 block 给该目录
30. 当我们在 Linux 下的 ext2 创建一个一般文件时, ext2 会分配一个 inode 与相对于该文件大小 的 block 数量给该文件
31. 当我们要读取某个文件时,就务必会经过目录的 inode 与 block ,然后才能够找到那个待读取 文件的 inode 号码, 最终才会读到正确的文件的 block 内的数据
32. 日志式文件系统 (Journaling filesystem)
    1. 预备:当系统要写入一个文件时,会先在日志记录区块中纪录某个文件准备要写入的信 息;
    2. 实际写入:开始写入文件的权限与数据;开始更新metadata的数据;
    3. 结束:完成数据与metadata的更新后,在日志记录区块当中完成该文件的纪录。
33. Linux 系统上面文件系统与内存有非 常大的关系
    1. 系统会将常用的文件数据放置到内存的缓冲区,以加速文件系统的读/写;
    2. 承上,因此 Linux 的实体内存最后都会被用光!这是正常的情况!可加速系统性能;
    3. 你可以手动使用 sync 来强迫内存中设置为 Dirty 的文件回写到磁盘中;
    4. 若正常关机时,关机指令会主动调用 sync 来将内存的数据回写入磁盘内;
    5. 但若不正常关机(如跳电、死机或其他不明原因),由于数据尚未回写到磁盘内, 因此 重新开机后可能会花很多时间在进行磁盘检验,甚至可能导致文件系统的损毁(非磁盘 损毁)
34. 每个 filesystem 都有独立的 inode / block / superblock 等信息,这个文件系统要能够链接到目 录树才能被我们使用. 重点是:挂载点一定是目录,该目录为进入该文件系统的入口。
35. 常见的支持文件系统有
    1. 传统文件系统:ext2 / minix / MS-DOS / FAT (用 vfat 模块) / iso9660 (光盘)等等;
    2. 日志式文件系统: ext3 /ext4 / ReiserFS / Windows' NTFS / IBM's JFS / SGI's XFS / ZFS
    3. 网络文件系统: NFS / SMBFS
36. 想要知道你的 Linux 支持的文件系统有哪些,可以察看下面这个目录:
    1. ls -l /lib/modules/$(uname -r)/kernel/fs
37. 系统目前已载入到内存中支持的文件系统则有:
    1. cat /proc/filesystems
38. 整个 Linux 的系统都是通过一个名为 Virtual Filesystem Switch 的核 心功能去读取 filesystem 的
39. EXT 家族当前较伤脑筋的地方:支持度最广,但格式化超慢!
40. XFS 文件系统在数据的分佈上,主要规划为三个部份,一个数据区 (data section)、一个文 件系统活动登录区 (log section)以及一个实时运行区 (realtime section) , XFS 文件系统的配置
    1. 数据区 (data section)
    2. 文件系统活动登录区 (log section)
    3. 实时运行区 (realtime section)
41. 找出系统 /boot 这个挂载点下面的文件系统的 superblock 纪录
    1. df -T /boot
42. 磁盘与目录的容量
    1. df:列出文件系统的整体磁盘使用量;
    2. du:评估文件系统的磁盘使用量(常用在推估目录所占容量)
    3. 将系统内的所有特殊文件格式及名称都列出来 **df -aT**
    4. 将目前各个 partition 当中可用的 inode 数量列出 **df -ih**
    5. 至于那个 /dev/shm/ 目录,其实是利用内存虚拟出来的磁盘空间,通常是总实体内存的一半
    6. 列出目前目录下的所有文件大小 **du -a**
    7. 检查根目录下面每个目录所占用的容量 **du -sm /\***
43. 实体链接与符号链接: ln
    1. Hard Link (实体链接, 硬式链接或实际链接) hard link 只是在某个目录下新增一笔文件名链接到某 inode 号码的关连记录而已. hard link 只是在某个目录下的 block 多写入一个关连 数据而已,既不会增加 inode 也不会耗用 block 数量 **ln** 
       1. 不能跨 Filesystem;
       2. 不能 link 目录。
    2. Symbolic Link (符号链接,亦即是捷径) Symbolic link 就是在创建一个 独立的文件,而这个文件会让数据的读取指向他 link 的那个文件的文件名 **ln -s**
44. 观察磁盘分区状态
    1. lsblk 列出系统上的所有磁盘列表 lsblk 可以看成“ list block device ”的缩写
    2. blkid 列出设备的 UUID 等参数 UUID 是全域单一识别码 (universally unique identifier)
    3. parted 列出磁盘的分区表类型与分区信息
45. MBR 分区表请使用 fdisk 分区, GPT 分区 表请使用 gdisk 分区
    1. partprobe 更新 Linux 核心的分区表信息
    2. 用 gdisk 删除一个分区
    3. 用 gdisk 新增分区
46. 磁盘格式化(创建文件系统)
    1. XFS 文件系统 mkfs.xfs
    2. XFS 文件系统 for RAID 性能优化
    3. 磁盘阵列 (RAID) 就是通过将文件先细分为数个小型的分区区块 (stripe) 之 后,然后将众多的 stripes 分别放到磁盘阵列里面的所有磁盘, 所以一个文件是被同时写入到 多个磁盘去,当然性能会好一些
    4. EXT4 文件系统 mkfs.ext4
    5. 其他文件系统 mkfs
47. 文件系统检验
    1. xfs\_repair 处理 XFS 文件系统
    2. fsck.ext4 处理 EXT4 文件系统
48. 文件系统挂载与卸载 使用 **mount** 这个指令
    1. 单一文件系统不应该被重复挂载在不同的挂载点(目录)中;
    2. 单一目录不应该重复挂载多个文件系统;
    3. 要作为挂载点的目录,理论上应该都是空目录才是。
    4. /etc/filesystems:系统指定的测试挂载文件系统类型的优先顺序
    5. /proc/filesystems:Linux系统已经载入的文件系统类型
    6. /lib/modules/$(uname -r)/kernel/fs/
    7. 挂载 xfs/ext4/vfat 等文件系统
    8. 挂载 CD 或 DVD 光盘
    9. 挂载 vfat 中文U盘 (USB磁盘)
    10. 重新挂载根目录与挂载不特定目录
    11. umount (将设备文件卸载) 卸载之后,可以使用 df 或 mount 看看是否还存 在目录树中
49. 磁盘/文件系统参数修订
    1. mknod
    2. xfs\_admin 修改 XFS 文件系统的 UUID 与 Label name
    3. tune2fs 修改 ext4 的 label name 与 UUID
50. 常见的磁盘文件名 /dev/sda 与 /dev/loop0 设备代码
    1. /dev/sda 8 0-15
    2. /dev/sdb 8 16-31
    3. /dev/loop0 7 0
    4. /dev/loop1 7 1
51. 开机挂载 /etc/fstab
    1. 根目录 / 是必须挂载的,而且一定要先于其它 mount point 被挂载进来。
    2. 其它 mount point 必须为已创建的目录,可任意指定,但一定要遵守必须的系统目录架构 原则 (FHS)
    3. 所有 mount point 在同一时间之内,只能挂载一次。
    4. 所有 partition 在同一时间之内,只能挂载一次。
    5. 如若进行卸载,您必须先将工作目录移到 mount point(及其子目录) 之外。
52. cat /etc/fstab : [设备/UUID等] [挂载点] [文件系统] [文件系统参数] [dump] [fsck]
    1. 第一栏:磁盘设备文件名/UUID/LABEL name:
       1. 文件系统或磁盘的设备文件名,如 /dev/vda2 等
       2. 文件系统的 UUID 名称,如 UUID=xxx
       3. 文件系统的 LABEL 名称,例如 LABEL=xxx
    2. 第二栏:挂载点 (mount point)
    3. 第三栏:磁盘分区的文件系统, 包括 xfs, ext4, vfat, reiserfs, nfs 等等
    4. 第四栏:文件系统参数:
    5. 第五栏:能否被 dump 备份指令作用
    6. 第六栏:是否以 fsck 检验扇区
53. /etc/fstab 是开机时的配置文件,不过,实际 filesystem 的挂载是记录到 /etc/mtab 与 /proc/mounts 这两个文件当中的
54. 特殊设备 loop 挂载 (镜像文件不烧录就挂载使用)
    1. 挂载光盘/DVD镜像文件 **mount -o loop**
    2. 创建大文件以制作 loop 设备文件
55. 安装时一定 需要的两个 partition ,一个是根目录,另外一个就是 swap(内存交换空间) . 由于 swap 是用磁盘来暂时放置内存中的信息,所以用到 swap 时, 你的主机磁盘灯就会开始闪个不停啊
    1. 设置一个 swap partition
    2. 创建一个虚拟内存的文件
56. 使用实体分区创建swap
    1. 分区:先使用gdisk在你的磁盘中分区出一个分区给系统作为swap。由于Linux的 gdisk 默认会将分区的 ID 设置为 Linux 的文件系统,所以你可能还得要设置一下 system ID 就是了。
    2. 格式化:利用创建swap格式的“mkswap设备文件名”就能够格式化该分区成为swap格 式
    3. 使用:最后将该swap设备启动,方法为:“swapon设备文件名”。
       1. 列出目前使用的 swap 设备有哪些的意思! swapon -s
    4. 观察:最终通过free与swapon-s这个指令来观察一下内存的用量吧!
57. 使用文件创建swap
    1. 使用 dd 这个指令来新增一个 128MB 的文件在 /tmp 下面
    2. 使用 mkswap 将 /tmp/swap 这个文件格式化为 swap 的文件格式
    3. 使用 swapon 来将 /tmp/swap 启动
    4. 使用 swapoff 关掉 swap file,并设置自动启用
58. 每个文件所使用掉 block 的容量 **ll -sh**
59. 利用 GNU 的 parted 进行分区行为 parted 可以直接在一行命令行就完成分区,是一个非常好用的指令 **parted [设备] [指令 [参数]]**可以使用“ man parted ”,或者是“ parted /dev/vda help mkpart ”去查询更详细的数据
    1. Number:这个就是分区的号码啦!举例来说,1号代表的是/dev/vda1的意思
    2. Start:分区的起始位置在这颗磁盘的多少MB处?有趣吧!他以容量作为单位喔
    3. End:此分区的结束位置在这颗磁盘的多少MB处
    4. Size:由上述两者的分析,得到这个分区有多少容量
    5. Filesystem:分析可能的文件系统类型为何的意思
    6. Name:就如同gdisk的SystemID之意
60. 一个可以被挂载的数据通常称为“文件系统, filesystem”而不是分区 (partition) 喔
61. 基本上 Linux 的传统文件系统为 Ext2 ,该文件系统内的信息主要有
    1. superblock:记录此 filesystem 的整体信息,包括inode/block的总量、使用量、剩余 量, 以及文件系统的格式与相关信息等
    2. inode:记录文件的属性,一个文件占用一个inode,同时记录此文件的数据所在的 block 号码
    3. block:实际记录文件的内容,若文件太大时,会占用多个 block
62. Ext2 文件系统的数据存取为索引式文件系统(indexed allocation)
63. Ext2文件系统主要有:boot sector, superblock, inode bitmap, block bitmap, inode table, data block 等六大部分
64. data block 是用来放置文件内容数据地方,在 Ext2 文件系统中所支持的 block 大小有 1K, 2K 及 4K 三种而已
65. inode 记录文件的属性/权限等数据,其他重要项目为: 每个 inode 大小均为固定,有 128/256Bytes 两种基本容量。每个文件都仅会占用一个 inode 而已; 因此文件系统能够 创建的文件数量与 inode 的数量有关
66. 文件的 block 在记录文件的实际数据,目录的 block 则在记录该目录下面文件名与其 inode 号码的对照表
67. 日志式文件系统 (journal) 会多出一块记录区,随时记载文件系统的主要活动,可加快 系统复原时间
68. Linux 文件系统为增加性能,会让内存作为大量的磁盘高速缓存
69. 磁盘的使用必需要经过:分区、格式化与挂载,分别惯用的指令为:gdisk, mkfs, mount 三个指令
70. 分区时,应使用 parted 检查分区表格式,再判断使用 fdisk/gdisk 来分区,或直接使用 parted 分区
71. 开机自动挂载可参考/etc/fstab之设置,设置完毕务必使用 mount -a 测试语法正确否
72. Linux上常见的压缩指令就是 gzip, bzip2 以及最新的 xz. 常见的压缩文件扩展名
    1. \*.Z compress 程序压缩的文件;
    2. \*.zip zip 程序压缩的文件;
    3. \*.gz gzip 程序压缩的文件;
    4. \*.bz2 bzip2 程序压缩的文件;
    5. xz 程序压缩的文件;
    6. .tar tar 程序打包的数据,并没有压缩过;
    7. \*.tar.gz tar 程序打包的文件,其中并且经过 gzip 的压缩
    8. \*.tar.bz2 tar 程序打包的文件,其中并且经过 bzip2 的压缩
    9. \*.tar.xz tar 程序打包的文件,其中并且经过 xz 的压缩
73. gzip, zcat/zmore/zless/zgrep
    1. 当使用 gzip 进行压缩时,默认的状态下原本的文件会被压缩成为 .gz 的文件名,原始文件就不再存在了
    2. **zcat 文件名.gz**
    3. gzip 压缩 **gzip -v services**
    4. 文件解压缩 **gzip -d services.gz**
    5. 创建的 services.gz 中,找出 http 这个关键字在哪几行? **zgrep -n 'http' services.gz**
74. bzip2, bzcat/bzmore/bzless/bzgrep
    1. bzip2 真是很不错用的东西~这玩意的压缩比竟然比 gzip 还要好
    2. bzip2 [-cdkzv#] 文件名
    3. bzcat 文件名.bz2
    4. bzip2 压缩 **bzip2 -v services**
    5. 文件内容读出来 **bzcat services.bz2**
    6. 文件解压缩 **bzip2 -d services.bz2**
75. xz, xzcat/xzmore/xzless/xzgrep
    1. xz [-dtlkc#] 文件名
    2. xcat 文件名.xz
    3. xz 来压缩 **xz -v services**
    4. 读出这个压缩文件的内容 **xz -l services.xz**
    5. 文件内容读出来 **xzcat services.xz**
    6. 解压缩  **xz -d services.xz**
    7. 保留原文件的文件名,并且创建压缩文件 **xz -k services**
76. 打包指令: tar
    1. 压 缩:tar -j<u>c</u>v -f filename.tar.bz2 要被压缩的文件或目录名称
    2. 查 询:tar -j<u>t</u>v -f filename.tar.bz2
    3. 解压缩:tar -j<u>x</u>v -f filename.tar.bz2 -C 欲解压缩的目录
    4. 使用 tar 加入 -z, -j 或 -J 的参数备份 /etc/ 目录
       1. time tar -jpcv -f /root/etc.tar.bz2 /etc
       2. time tar -Jpcv -f /root/etc.tar.xz /etc
    5. 查阅 tar 文件的数据内容 (可察看文件名),与备份文件名有否根目录的意义
       1. tar -jtv -f /root/etc.tar.bz2
    6. 将备份的数据解压缩,并考虑特定目录的解压缩动作 (-C 选项的应用)
       1. 直接进行解打包 **tar -jxv -f /root/etc.tar.bz2**
       2. 指定欲解 开的目录  **tar -jxv -f /root/etc.tar.bz2 -C /tmp**
    7. 仅解开单一文件的方法
       1. 假设解开 shadow 文件 
          1. **tar -jtv -f /root/etc.tar.bz2 &#124; grep 'shadow'**
          2. **tar -jxv -f 打包档.tar.bz2 待解开文件名**
          3. **tar -jxv -f /root/etc.tar.bz2 etc/shadow**
       2. 打包某目录,但不含该目录下的某些文件之作法
          1. **tar -jcv -f /root/system.tar.bz2 --exclude=/root/etc\* \**

**&gt; --exclude=/root/system.tar.bz2 /etc /root**

1. 仅备份比某个时刻还要新的文件
   1. find /etc -newer /etc/passwd
   2. 使用 tar 来进行打包吧!日期为上面看到的 2015/06/17

tar -jcv -f /root/etc.newer.then.passwd.tar.bz2 \

&gt; --newer-mtime="2015/06/17" /etc/\*

1. 显示出文件即可

tar -jtv -f /root/etc.newer.then.passwd.tar.bz2 &#124; grep -v '/$'

1. 基本名称: tarfile, tarball ?
   1. 如果仅是打 包而已,就是“ tar -cv -f file.tar ”而已,这个文件我们称呼为 tarfile
   2. 如果还有进行压缩的支 持,例如“ tar -jcv -f file.tar.bz2 ”时,我们就称呼为 tarball (tar 球?)
2. 特殊应用:利用管线命令与数据流
   1. 将 /etc 整个目录一边打包一边在 /tmp 解开  **tar -cvf - /etc &#124; tar -xvf -**
3. 解压缩后的 SELinux 课题
   1. 通过各种可行的救援方式登陆系统,然后修改 /etc/selinux/config 文件,将 SELinux 改成 permissive 模式,重新开机后系统就正常了;
   2. 在第一次复原系统后,不要立即重新开机!先使用 restorecon -Rv /etc 自动修复一下 SELinux 的类型即可。
   3. 通过各种可行的方式登陆系统,创建 /.autorelabel 文件,重新开机后系统会自动修复 SELinux 的类型,并且又会再次重新开机,之后就正常了
4. XFS 文件系统备份 xfsdump
   1. xfsdump 不支持没有挂载的文件系统备份!所以只能备份已挂载的!
   2. xfsdump 必须使用 root 的权限才能操作 (涉及文件系统的关系)
   3. xfsdump 只能备份 XFS 文件系统啊!
   4. xfsdump 备份下来的数据 (文件或储存媒体) 只能让 xfsrestore 解析
   5. xfsdump 是通过文件系统的 UUID 来分辨各个备份文件的,因此不能备份两个具有相同 UUID 的文件系统喔!
   6. xfsdump [-L S\_label] [-M M\_label] [-l #] [-f 备份文件] 待备份数据
   7. 用 xfsdump 备份完整的文件系统
      1. 先确定 /boot 是独立的文件系统喔! df -h /boot
      2. 将完整备份的文件名记录成为 /srv/boot.dump xfsdump -l 0 -L boot\_all -M boot\_all -f /srv/boot.dump /boot
   8. 用 xfsdump 进行累积备份 (Incremental backups)
      1. 看一下有没有任何文件系统被 xfsdump 过的数据? **xfsdump -I**
      2. 创建一个大约 10 MB 的文件在 /boot 内 **dd if=/dev/zero of=/boot/testing.img bs=1M count=10**
      3. 开始创建差异备份文件,此时我们使用 level 1 xfsdump -l 1 **-L boot\_2 -M boot\_2 -f /srv/boot.dump1 /boot**
      4. 最后再看一下是否有记录 level 1 备份的时间点呢?  **xfsdump -I**
5. XFS 文件系统还原 xfsrestore
   1. 用 xfsrestore 观察 xfsdump 后的备份数据内容
   2. 简单复原 level 0 的文件系统
      1. 直接将数据给它覆盖回去 **xfsrestore -f /srv/boot.dump -L boot\_all /boot**
      2. 将备份数据在 /tmp/boot 下面解开! **xfsrestore -f /srv/boot.dump -L boot\_all /tmp/boot du -sm /boot /tmp/boot**
      3. 检查两者一致否： **diff -r /boot /tmp/boot**
   3. 仅还原部分文件的 xfsrestore 互动模式 **xfsrestore -f /srv/boot.dump -i /tmp/boot3**
6. 光盘写入工具
   1. 先将所需要备份的数据创建成为一个镜像文件(iso),利用 mkisofs 指令来处理
      1. mkisofs [-o 镜像文件] [-Jrv] [-V vol] [-m file] 待备份文件... &gt; -graft-point isodir=systemdir …
   2. 将该镜像文件烧录至光盘或 DVD 当中,利用 cdrecord 指令来处理
      1. mkisofs -r -v -o /tmp/system.img /root /home /etc
      2. mkisofs -r -V 'linux\_file' -o /tmp/system.img &gt; -m /root/etc -graft-point /root=/root /home=/home /etc=/etc
      3. 如果你是有自己要制作的数 据内容,其实最简单的方法,就是将所有的数据预先处理到某一个目录中, 再烧录该目录即 可
   3. 制作/修改可开机光盘图像挡
      1. 先观察一下这片光盘里面有啥东西?是否是我们需要的光盘系统!
         1. isoinfo -d -i /home/CentOS-7-x86\_64-Minimal-1503-01.iso
      2. 开始挂载这片光盘到 /mnt ,并且将所有数据完整复制到 /srv/newcd 目录去喔
         1. mount /home/CentOS-7-x86\_64-Minimal-1503-01.iso /mnt
         2. mkdir /srv/newcd
         3. rsync -a /mnt/ /srv/newcd
      3. 假设已经处理完毕你在 /srv/newcd 里面所要进行的各项修改行为,准备创建 ISO 档!
         1. cd /srv/newcd
         2. Mkisofs -o /custom.iso -b isolinux/isolinux.bin -c isolinux/boot.cat &gt; -no-emul-boot -V 'CentOS 7 x86\_64' -boot-load-size 4 -boot-info-table -R -J -v -T .
7. cdrecord:光盘烧录工具 wodim 这个文字界面指令来进行烧录的行为
   1. 侦测你的烧录机所在位置
      1. ll /dev/sr0
      2. wodim --devices dev=/dev/sr0
      3. wodim --devices dev=/dev/sr0
   2. 进行 CD/DVD 的烧录动作
      1. 先抹除光盘的原始内容:(非可重复读写则可略过此步骤) wodim -v dev=/dev/sr0 blank=fast
      2. 开始烧录 wodim -v dev=/dev/sr0 speed=4 -dummy -eject /tmp/system.img
      3. 烧录完毕后,测试挂载一下,检验内容: mount /dev/sr0/mnt umount /mnt
8. dd 指令
   1. 将 /etc/passwd 备份到 /tmp/passwd.back 当中 **dd if=/etc/passwd of=/tmp/passwd.back**
   2. 将刚刚烧录的光驱的内容,再次的备份下来成为图像挡 **dd if=/dev/sr0 of=/tmp/system.iso**
   3. 假设你的 USB 是 /dev/sda 好了,请将刚刚范例二的 image 烧录到 USB 磁盘中

**lsblk /dev/sda dd if=/tmp/system.iso of=/dev/sda mount /dev/sda /mnt**

1. 将你的 /boot 整个文件系统通过 dd 备份下来  **dd if=/dev/vda2 of=/tmp/vda2.img**
2. 未来你想要创建两颗一模一样的磁盘时, 只要 下达类似: dd if=/dev/sda of=/dev/sdb ,就能够让两颗磁盘一模一样,甚至 /dev/sdb 不需要 分区与格式化, 因为该指令可以将 /dev/sda 内的所有数据,包括 MBR 与 partition table 也复 制到 /dev/sdb 说
3. cpio 备份
   1. 找出 /boot 下面的所有文件,然后将他备份到 /tmp/boot.cpio 去:

find boot &#124; cpio -ocvB &gt; /tmp/boot.cpio

1. 将刚刚的文件给他在 /root/ 目录下解开 Cpio -idvc &lt; /tmp/boot.cpio
2. 事实上 cpio 可以将系统的数据完整的备份到磁带机上头去喔!如果你有磁带机的话
   1. 备份:find / | cpio -ocvB > /dev/st0
   2. 还原:cpio -idvc < /dev/st0
3. 其实系统里面已经含有一个使用 cpio 创建的文件喔!那就是 /boot/initramfs-xxx 这个文件 啦
   1. 我们先来看看该文件是属于什么文件格式,然后再加以处理: file /boot/initramfs-3.10.0-229.el7.x86\_64.img
   2. cd /tmp/initramfs
   3. cpio -idvc &lt; /boot/initramfs-3.10.0-229.el7.x86\_64.img
4. 压 缩:tar -Jcv -f filename.tar.xz 要被压缩的文件或目录名称
5. 查 询:tar -Jtv -f filename.tar.xz
6. 解压缩:tar -Jxv -f filename.tar.xz -C 欲解压缩的目录
7. xfsdump 指令可备份文件系统或单一目录 xfsdump 的备份若针对文件系统时,可进行 0-9 的 level 差异备份!其中 level 0 为完整 备份
8. xfsrestore 指令可还原被 xfsdump 创建的备份文件
9. 要创建光盘烧录数据时,可通过 mkisofs 指令来创建
10. 可通过 wodim 来写入 CD 或 DVD 烧录机
11. dd 可备份完整的 partition 或 disk ,因为 dd 可读取磁盘的 sector 表面数据
12. cpio 为相当优秀的备份指令,不过必须要搭配类似 find 指令来读入欲备份的文件名数 据,方可进行备份动作
13. 文书编辑器有哪些 emacs, pico, nano, joe, 与 vim
14. vi 的使用
    1. 一般指令模式 (command mode)
    2. 编辑模式 (insert mode) 等到你按下“i, I, o, O, a, A, r, R”等任何一个字母之后才会进入编辑模式 , 回到一般指令模式时, 则必须要按下“Esc”这个按键即可退出编辑模 式
    3. 命令行命令模式 (command-line mode) 输入“ : / ? ”三个中的任何一个按钮,就可以将光标移动到最下面那一列
    4. 可以使用“强制写入”的方式吗?可以!使 用“ **:wq!** ” 多加一个惊叹号即可
15. vi 一般指令模式可用的按钮说明,光标移动、复制贴上、搜寻取代等
    1. h 或 向左方向键(←)
    2. j 或 向下方向键(↓)
    3. k 或 向上方向键(↑)
    4. l 或 向右方向键(→)
    5. 向下移动 30 列, 可以使用 "**30j**" 或 "**30↓**" 的组合按键, 亦即加上想要进行的次数(数字)后, 按下动作即可
    6. **[Ctrl] + [f] 屏幕“向下”移动一页,相当于 [Page Down]按 键 (常用)**
    7. **[Ctrl] + [b] 屏幕“向上”移动一页,相当于 [Page Up] 按键 (常用**
    8. [Ctrl] + [d] 屏幕“向下”移动半页
    9. [Ctrl] + [u] 屏幕“向上”移动半页
    10. + 光标移动到非空白字符的下一列
    11. - 光标移动到非空白字符的上一列
    12. n<space> 那个 n 表示“数字”, 例如 20<space> 则光标会向后面移动 20 个字符距离
    13. **0 或功能键[Home] 这是数字“ 0 ”:移动到这一列的最前面字符处 (常用)**
    14. **$ 或功能键[End] 移动到这一列的最后面字符处(常用)**
    15. H 光标移动到这个屏幕的最上方那一列的第一个字符
    16. M 光标移动到这个屏幕的中央那一列的第一个字符
    17. L 光标移动到这个屏幕的最下方那一列的第一个字符
    18. **G 移动到这个文件的最后一列(常用)**
    19. nG n 为数字。移动到这个文件的第 n 列 例如 20G 则会移动到这个文件的第 20 列(可配合 **:set nu**)
    20. **gg 移动到这个文件的第一列,相当于 1G 啊! (常用)**
    21. **n<Enter> n 为数字。光标向下移动 n 列(常用)**
    22. **/word 向光标之下寻找一个名称为 word 的字串 (常用)**
    23. ?word 向光标之上寻找一个字串名称为 word 的字 串。
    24. n 这个 n 是英文按键。代表“<u>重复前一个搜 寻的动作</u>”。举例来说, 如果刚刚我们执 行 /vbird 去向下搜寻 vbird 这个字串,则按下 n 后,会向下继续搜寻下一个名称为 vbird 的 字串。如果是执行 ?vbird 的话,那么按下 n 则会向上继续搜寻名称为 vbird 的字串
    25. N 这个 N 是英文按键。与 n 刚好相反,为“反 向”进行前一个搜寻动作。 例如 /vbird 后,按 下 N 则表示“向上”搜寻 vbird
    26. **:n1,n2s/word1/word2/g n1 与 n2 为数字。在第 n1 与 n2 列之间寻找 word1 这个字串,并将该字串取代为 word2 !举例来说,在 100 到 200 列之间搜寻 vbird 并取代为 VBIRD 则:“:100,200s/vbird/VBIRD/g”。(常用)**
    27. **:1,$s/word1/word2/g 从第一列到最后一列寻找 word1 字串,并将 该字串取代为 word2 !(常用)**
    28. **:1,$s/word1/word2/gc 从第一列到最后一列寻找 word1 字串,并将 该字串取代为 word2 !且在取代前显示提示 字符给使用者确认 (confirm) 是否需要取 代!(常用)**
    29. **x, X 在一列字当中,x 为向后删除一个字符 (相 当于 [del] 按键), X 为向前删除一个字符 (相当于 [backspace] 亦即是倒退键) (常 用)**
    30. nx n 为数字,连续向后删除 n 个字符。举例来 说,我要连续删除 10 个字符, “10x”。
    31. **dd 删除光标所在的那一整列(常用)**
    32. **ndd n 为数字。删除光标所在的向下 n 列,例如 20dd 则是删除 20 列 (常用)**
    33. d1G 删除光标所在到第一列的所有数据
    34. dG 删除光标所在到最后一列的所有数据
    35. d$ 删除光标所在处,到该列的最后一个字符
    36. d0 那个是数字的 0 ,删除光标所在处,到该列 的最前面一个字符
    37. **yy 复制光标所在的那一列(常用)**
    38. **nyy n 为数字。复制光标所在的向下 n 列,例如 20yy 则是复制 20 列(常用)**
    39. y1G 复制光标所在列到第一列的所有数据
    40. yG 复制光标所在列到最后一列的所有数据
    41. y0 复制光标所在的那个字符到该列行首的所有数据
    42. y$ 复制光标所在的那个字符到该列行尾的所有数据
    43. **p, P p 为将已复制的数据在光标下一列贴上,P 则 为贴在光标上一列! 举例来说,我目前光标 在第 20 列,且已经复制了 10 列数据。则按 下 p 后, 那 10 列数据会贴在原本的 20 列之 后,亦即由 21 列开始贴。但如果是按下 P 呢? 那么原本的第 20 列会被推到变成 30 列。 (常用)**
    44. J 将光标所在列与下一列的数据结合成同一列
    45. c 重复删除多个数据,例如向下删除 10 列,[ 10cj ]
    46. **u 复原前一个动作。(常用)**
    47. **[Ctrl]+r 重做上一个动作。(常用)**
    48. **. 不要怀疑!这就是小数点!意思是重复前一 个动作的意思。 如果你想要重复删除、重复 贴上等等动作,按下小数点“.”就好了! (常 用)**
16. vi 一般指令模式切换到编辑模式的可用的按钮说明
    1. **i, I 进入插入模式(Insert mode):i 为“从目前光标所在处插入”, I 为“在目前所在列的第一个非空白 字符处开始插入”。 (常用)**
    2. **a, A 进入插入模式(Insert mode):a 为“从目前光标所在的下一个字符 处开始插入”, A 为“从光标所在列 的最后一个字符处开始插入”。 (常用**
    3. **o, O 进入插入模式(Insert mode): 这是英文字母 o 的大小写。o 为“在目前光标所在的下一列处插 入新的一列”; O 为在目前光标所 在处的上一列插入新的一列!(常 用)**
    4. **r, R 进入取代模式(Replace mode):r 只会取代光标所在的 那一个字符一次;R会一直取代光 标所在的文字,直到按下 ESC 为 止;(常用)**
    5. **[Esc] 退出编辑模式,回到一般指令模式中(常用)**
17. vi一般指令模式切换到命令行界面的可用按钮说明
    1. **:w 将编辑的数据写入硬盘文件中(常用)**
    2. **:q 离开 vi (常用)**
    3. **:wq 储存后离开,若为 :wq! 则为强制储存后离开 (常用)**
    4. ZZ 这是大写的 Z 喔!若文件没有更动,则不储存离开,若 文件已经被更动过,则储存后离开
    5. :w [filename] 将编辑的数据储存成另一个文件(类似另存新文件)
    6. :r [filename] 在编辑的数据中,读入另一个文件的数据。亦即将 “filename” 这个文件内容加到光标所在列后面
    7. :n1,n2 w [filename] 将 n1 到 n2 的内容储存成 filename 这个文件。
    8. :! command 暂时离开 vi 到命令行界面下执行 command 的显示结 果!例如 “:! ls /home”即可在 vi 当中察看 /home 下面以 ls 输出的文件信息!
    9. :set nu 显示行号,设置之后,会在每一列的字首显示该列的行号
    10. :set nonu 与 set nu 相反,为取消行号!
18. vim 区块选择(Visual Block)
    1. v 字符选择,会将光标经过的地方反白选择!
    2. V 列选择,会将光标经过的列反白选择!
    3. [Ctrl]+v 区块选择,可以用长方形的方式选择数据
    4. y 将反白的地方复制起来
    5. d 将反白的地方删除掉
    6. p 将刚刚复制的区块,在光标所在处贴上!
19. vim 多文件编辑
    1. :n 编辑下一个文件
    2. :N 编辑上一个文件
    3. :files 列出目前这个 vim 的打开的所有文件
20. vim 多窗口功能
    1. :sp [filename] 打开一个新窗口,如果有加 filename, 表示在新窗口打开一个新文件,否 则表示两个窗口为同一个文件内容(同步显示)。
    2. [ctrl]+w+ j [ctrl]+w+↓ 按键的按法是:先按下 [ctrl] 不放, 再按下 w 后放开所有的按键,然后再按 下 j (或向下方向键),则光标可移动到下方的窗口
    3. [ctrl]+w+ k [ctrl]+w+↑ 同上,不过光标移动到上面的窗口。
    4. [ctrl]+w+ q 其实就是 :q 结束离开啦! 举例来说,如果我想要结束下方的窗口,那么利 用 [ctrl]+w+↓ 移动到下方窗口后,按下 :q 即可离开, 也可以按下 [ctrl]+w+q 啊
21. vim 的挑字补全功能
    1. [ctrl]+x -> [ctrl]+n 通过目前正在编辑的这个“文件的内容文字”作为关键字, 以补齐
    2. [ctrl]+x -> [ctrl]+f 以当前目录内的“文件名”作为关键字, 以补齐
    3. [ctrl]+x -> [ctrl]+o 以扩展名作为语法补充,以 vim 内置的关键字, 以补齐
22. vim 环境设置与记录: ~/.vimrc, ~/.viminfo
    1. vim 会主动的将你曾经做过的行为登录下来,好让你下次可以轻松的作业 啊! 那个记录动作的文件就是: **~/.viminfo**
    2. 通过配置文件来直接规定我们习惯的 vim 操作 环境呢, 整体 vim 的设置值一般是放置在 **/etc/vimrc** 这个文件 ,不过,不建议你修改他! 你 可以修改 ~/.vimrc 这个文件 (默认不存在,请你自行手动创建!),将你所希望的设置值写入
23. 其他 vim 使用注意事项
    1. DOS 与 Linux 的断行字符
       1. dos2unix [-kn] file [newfile]
       2. unix2dos [-kn] file [newfile]
    2. 语系编码转换
       1. iconv --list
       2. iconv -f 原本编码 -t 新编码 filename [-o newfile]
24. Vim 指令
    1. 数字是有意义的,用来说明重复进行几次动作的意思,如 5yy 为复制 5 列之意
    2. 光标的移动中,大写的 G 经常使用,尤其是 1G, G 移动到文章的头/尾功能
    3. vi 的取代功能也很棒! :n1,n2s/old/new/g 要特别注意学习起来
    4. 小数点“ . ”为重复进行前一次动作,也是经常使用的按键功能
    5. 进入编辑模式几乎只要记住: i, o, R 三个按钮即可!尤其是新增一列的 o 与取代的 R
    6. vim 会主动的创建 swap 暂存盘,所以不要随意断线
    7. 如果在文章内有对齐的区块,可以使用 [ctrl]-v 进行复制/贴上/删除的行为
    8. 使用 :sp 功能可以分区窗口
    9. 在 vi 下面作了很多的编辑动作之后,却想还原成原来的文件内容,应该怎么进行?直接 按下 :e! 即可恢复成文件的原始状态
25. Bash shell 的功能
    1. 命令编修能力 (history)
       1. ~/.bash\_history 记录的是前一次登陆以前所执行过的指令, 而至于这一次登陆所执行的指令 都被暂存在内存中,当你成功的登出系统后,该指令记忆才会记录到 .bash\_history 当中
    2. 命令与文件补全功能: ([tab] 按键的好处)
       1. [Tab] 接在一串指令的第一个字的后面,则为命令补全
       2. [Tab] 接在一串指令的第二个字以后时,则为“文件补齐”
       3. 若安装 bash-completion 软件,则在某些指令后面使用 [tab] 按键时,可以进行“选项/参 数的补齐”功能
    3. 命令别名设置功能: (alias)
       1. alias lm='ls -al'
    4. 工作控制、前景背景控制: (job control, foreground, background)
    5. 程序化脚本: (shell scripts)
    6. 万用字符: (Wildcard)
    7. 查询指令是否为 Bash shell 的内置命令: **type**
26. Bash shell指令的下达与快速编辑按钮
    1. **[ctrl]+u/[ctrl]+k 分别是从光标处向前删除指令串 ([ctrl]+u) 及向后删除指令串 ([ctrl]+k)**
    2. **[ctrl]+a/[ctrl]+e 分别是让光标移动到整个指令串的最前面 ([ctrl]+a) 或最后面 ([ctrl]+e)**
    3. 为了区别与自订变量的不同,环境变量通常以大写字符来表示呢
27. Bash shell变量的设置规则
    1. 变量与变量内容以一个等号“=”来链接
    2. 等号两边不能直接接空白字符,如下所示为错误: “myname = VBird”
    3. 变量名称只能是英文字母与数字,但是开头字符不能是数字,如下为错误: “2myname=VBird”
    4. 变量内容若有空白字符可使用双引号“"”或单引号“'”将变量内容结合起来,但
       1. 双引号内的特殊字符如 $ 等,可以保有原本的特性,如下所示: “var="lang is $LANG"”则“echo $var”可得“lang is zh\_TW.UTF-8”
       2. 单引号内的特殊字符则仅为一般字符 (纯文本),如下所示: “var='lang is $LANG'”则“echo $var”可得“lang is $LANG”
    5. 可用跳脱字符“ \ ”将特殊符号(如 [Enter], $, \, 空白字符, '等)变成一般字符,如: “myname=VBird\ Tsai”
    6. 在一串指令的执行中,还需要借由其他额外的指令所提供的信息时,可以使用反单引 号“ 指令 ”或 “$(指令)”。特别注意,那个 ` 是键盘上方的数字键 1 左边那个按键,而不 是单引号! 例如想要取得核心版本的设置: “version=$(uname -r)”再“echo $version”可得“3.10.0-229.el7.x86\_64”
    7. 若该变量为扩增变量内容时,则可用 "$变量名称" 或 ${变量} 累加内容,如下所示: “PATH="$PATH":/home/bin”或“PATH=${PATH}:/home/bin”
    8. 若该变量需要在其他子程序执行,则需要以 export 来使变量变成环境变量: “export PATH”
    9. 取消变量的方法为使用 unset :“unset 变量名称”例如取消 myname 的设置: “unset myname”
    10. locate 指令可以列出所有的相关文件文件名
28. 如何进入到您目前核心的模块目录
    1. cd /lib/modules/`uname -r`/kernel
    2. cd /lib/modules/$(uname -r)/kernel # 以此例较佳!
29. 环境变量的功能
    1. 用 env 观察环境变量与常见环境变量说明
    2. 关于本 shell 的 PID echo $$
    3. ?:(关于上个执行指令的回传值) echo $?
    4. export: 自订变量转成环境变量
    5. 子程序仅会继承父程序的环境变量, 子 程序不会继承父程序的自订变量 export 指令 如你想要让该变量 内容继续的在子程序中使用  **export 变量名称**
    6. 影响显示结果的语系变量 (locale)
30. 变量的有效范围
    1. “全域变量, global variable”与“区域变量, local variable” 环境变量=全域变量 自订变量=区域变量
    2. 当启动一个 shell,操作系统会分配一记忆区块给 shell 使用,此内存内之变量可让子程 序取用
    3. 若在父程序利用 export 功能,可以让自订变量的内容写到上述的记忆区块当中(环境变 量);
    4. 当载入另一个 shell 时 (亦即启动子程序,而离开原本的父程序了),子 shell 可以将父 shell 的环境变量所在的记忆区块导入自己的环境变量区块当中。
31. 变量键盘读取、阵列与宣告: read, array, declare
    1. 要读取来自键盘输入的变量,就是用 read 这个指令 **read [-pt] variable**
    2. declare 或 typeset 是一样的功能,就是在“宣告变量的类型”
       1. 如果使用 declare 后面并没有接 任何参数,那么 bash 就会主动的将所有的变量名称与内容通通叫出来  **declare [-aixr] variable**
       2. 变量类型默认为“字串”,所以若不指定变量类型,则 1+2 为一个“字串”而不是“计算式”。 所以上述第一个执行的结果才会出现那个情况的
       3. bash 环境中的数值运算,默认最多仅能到达整数形态,所以 1/3 结果是 0
       4. sum=100+300+50 echo ${sum} 100+300+50
       5. Declare sum=100+300+50 echo ${sum} 450
    3. 阵列 (array) 变量类型 建议直接以 ${阵列} 的方式来读取
32. 与文件系统及程序的限制关系: ulimit
    1. ulimit [-SHacdfltu] [配额]
    2. 列出你目前身份(假设为一般帐号)的所有限制数据数值 ulimit -a
    3. 限制使用者仅能创建 10MBytes 以下的容量的文件 ulimit -f 10240
33. 变量内容的删除、取代与替换
    1. 变量内容的删除与取代
       1. 在 PATH 这个变量的内容中,每个目录都是以冒号“:”隔开的, 所以 要从头删除掉目录就是介于斜线 (/) 到冒号 (:) 之间的数据!但是 PATH 中不止一个冒号 (:) 啊! 所以 # 与 ## 就分别代表:

符合取代文字的“最短的”那一个; 符合取代文字的“最长的”那一个

1. 想要“从后面向前删除变量内容”呢? 这个 时候就得使用百分比 (%) 符号了
2. ${变量#关键字} ${变量##关键字} 若变量内容从头开始的数据符合“关键字”,则将符合的最短数据 删除 若变量内容从头开始的数据符合“关键字”,则将符合的最 长数据删除
3. ${变量%关键字} ${变量%%关键字} 若变量内容从尾向前的数据符合“关键字”,则将符合的最短数据 删除 若变量内容从尾向前的数据符合“关键字”,则将符合的最 长数据删除
4. ${变量/旧字串/新字串} ${变量//旧字串/新字串} 若变量内容符合“旧字串”则“第一个旧字串会被新字串取代” 若变 量内容符合“旧字串”则“全部的旧字串会被新字串取代”
5. 命令别名设置: alias, unalias
   1. alias 的定义规则与变量定义规则几乎相同 你只要在 alias 后面加上你的 {“别名”='指令 选项...' }, 以后你只要输入 lm 就相当于输入了 ls -al|more 这一串指令
   2. 如果要取消命令别名的话,那么就使用 unalias unalias lm
   3. 命令别名与变量有什么不同呢?命令别名是“新创一个新的指令, 你可以直接下达该指 令”的,至于变量则需要使用类似“ echo ”指令才能够调用出变量的内容! 这两者当然不一 样
6. 历史命令:history
   1. !number 执行第几笔指令的意思;
   2. Ommand command :由最近的指令向前搜寻“指令串开头为 command”的那个指令,并执行;
   3. !! !! : 就是执行上一 个指令(相当于按↑按键后,按 Enter)
7. 路径与指令搜寻顺序
   1. 以相对/绝对路径执行指令,例如“/bin/ls”或“./ls”;
   2. 由alias找到该指令来执行;
   3. 由bash内置的(builtin)指令来执行;
   4. 通过$PATH这个变量的顺序搜寻到的第一个指令来执行。
   5. 如果想要了解指令搜寻的顺序,其实通过 type -a ls 也可以查询的到
8. bash 的进站与欢迎讯息: /etc/issue, /etc/motd
   1. cat /etc/issue
   2. vim /etc/motd 那么当你的使用者(包括所有的一般帐号与 root)登陆主机后,就会显示这样的讯息出来
9. bash 的环境配置文件: 命令别名啦、自订的变量啦,在你 登出 bash 后就会失效,所以你想要保留你的设置, 就得要将这些设置写入配置文件才行.
   1. /etc/profile (login shell 才会读) /etc/profile 可不止会做这些事而已,他还会去调用外部的设置数据
   2. ~/.bash\_profile (login shell 才会读)
   3. source :读入环境配置文件的指令
   4. 利用 source 或小数点 (.) 都可以将配置文件的内容读进来目前的 shell 环境中
   5. ~/.bashrc (non-login shell 会读)
   6. 如果你想要将命令提示字符捉回来,那么可以复制 **/etc/skel/.bashrc** 到你的主文件夹,再修订一下你所想要的内容, 并使用 source 去调用 ~/.bashrc ,那你的命令提示字符就会回来啦
   7. /etc/man\_db.conf
   8. ~/.bash\_history
   9. ~/.bash\_logout
10. 终端机的环境设置: stty, set
    1. stty (setting tty 终端机的意思) 呢! stty 也可以帮助设置终端机的输入按键代表意义 **stty [-a]**
    2. set 还可以帮我们设 置整个指令输出/输入的环境。 例如记录历史命令、显示错误内容等等

**set [-uvCHhmBx]**

1. 万用字符与特殊符号
   1. \* 代表“ 0 个到无穷多个”任意字符
   2. ? 代表“一定有一个”任意字符
   3. [] 同样代表“一定有一个在括号内”的字符(非任意字符)。例如 [abcd] 代表“一定有一 个字符, 可能是 a, b, c, d 这四个任何一个”
   4. [- ] 若有减号在中括号内时,代表“在编码顺序内的所有字符”。例如 [0-9] 代表 0 到 9 之 间的所有数字,因为数字的语系编码是连续的!
   5. [^ ] 若中括号内的第一个字符为指数符号 (^) ,那表示“反向选择”,例如 [^abc] 代表 一定有一个字符,只要是非 a, b, c 的其他字符就接受的意思。
2. 数据流重导向
   1. standard output 指令执行所回传的正确的讯息
   2. standard error output 指令 执行失败后,所回传的错误讯息
   3. 标准输入 (stdin):代码为0,使用<或<<;
   4. 标准输出 (stdout):代码为1,使用>或>>;
   5. 标准错误输出(stderr):代码为2,使用2>或2>>;
   6. 1> :以覆盖的方法将“正确的数据”输出到指定的文件或设备上;
   7. 1>>:以累加的方法将“正确的数据”输出到指定的文件或设备上;
   8. 2> :以覆盖的方法将“错误的数据”输出到指定的文件或设备上;
   9. 2>>:以累加的方法将“错误的数据”输出到指定的文件或设备上;
   10. /dev/null 垃圾桶黑洞设备与特殊写法
   11. standard input : < 与 << 将原本需要由键盘输入的数据,改由文件内容来取代
3. 命令执行的判断依据: ; , &&, ||
   1. cmd ; cmd (不考虑指令相关性的连续指令下达) sync; sync; shutdown -h now
   2. $? (指令回传值) 与 && 或 ||
      1. cmd1 && cmd2 若 cmd1 执行完毕且正确执行($?=0),则开始执行 cmd2。 2. 若 cmd1 执行完毕且为错误 ($?≠0),则 cmd2 不执行
      2. cmd1 || cmd2 若 cmd1 执行完毕且正确执行($?=0),则 cmd2 不执行。 2. 若 cmd1 执 行完毕且为错误 ($?≠0),则开始执行 cmd2
   3. 以 ls 测试 /tmp/vbirding 是否存在,若存在则显示 "exist" ,若不存在,则显示 "not exist"!

**ls /tmp/vbirding && echo "exist" || echo "not exist"**

1. 管线命令 (pipe)
   1. 管线命令仅会处理 standard output,对于 standard error output 会 以忽略
   2. 管线命令必须要能够接受来自前一个指令的数据成为 standard input 继续处理才行
   3. cut -d'分隔字符' -f fields &lt;==用于有特定分隔字符
   4. cut -c 字符区间 &lt;==用于排列整齐的讯息
   5. grep [-acinv] [--color=auto] '搜寻字串' filename
   6. sort [-fbMnrtuk] [file or stdin] eg: sort -k1,1d -k2,2n -T ./tmp $1 > $1.reorder
   7. uniq [-ic]
   8. wc [-lwm]
   9. 双向重导向: tee tee [-a] file tee 可以让 standard output 转存一份到文件内并将同样的数据继续送到屏幕去处理
2. 字符转换命令: tr, col, join, paste, expand
   1. Tr tr 可以用来删除一段讯息当中的文字,或者是进行文字讯息的替换 **tr [-ds] SET1 …**
      1. **tr "[:lower:]" "[:upper:]" < file1** Translate the contents of file1 to upper-case.
      2. **tr -cs "[:alpha:]" "\n" < file1** Create a list of the words in file1, one per line, where a word is taken to be a maximal string of letters
   2. col 可以用来简单的处理将 [tab] 按键取代成为空 白键 **col [-xb]**
   3. join 在使用 join 之前,你所需要处理的文件应该要事先经过排序 (sort) 处理  **join [-ti12] file1 file2**
   4. paste 就 直接“将两行贴在一起,且中间以 [tab] 键隔开” **paste [-d] file1 file2**
   5. Expand 将 [tab] 按键转成空白键  **expand [-t] file** 并不是一个 ^I 就会换成 8 个 空白
   6. split 将一个大文件,依据文件大小或行数来分区,分区成为小文件 **split [-bl] file PREFIX**
3. 参数代换: xargs: x 是加减乘除的乘号,args 则是 arguments (参数) , 就是在产生某个指令的参数
   1. xargs [-0epn] command
   2. 会使用 xargs 的原因是, 很多指令其 实并不支持管线命令,因此我们可以通过 xargs 来提供该指令引用 standard input 之用
4. 关于减号 - 的用途
   1. 在管线命令当中,常常会使用到前一个指令的 stdout 作为这次的 stdin , 某些指令需要用到文件名称 (例如 tar) 来进行处理时,该 stdin 与 stdout 可以利用减号 "-" 来替代
   2. tar -cvf - /home |tar -xvf - -C /tmp/homeback
5. type 可以用来找到执行指令为何种类型,亦可用于与 which 相同的功能
6. 使用 env 与 export 可观察环境变量,其中 export 可以将自订变量转成环境变量
7. bash 的配置文件主要分为 login shell 与 non-login shell。login shell 主要读取 /etc/profile 与 ~/.bash\_profile, non-login shell 则仅读取 ~/.bashrc
8. 在使用 vim 时,若不小心按了 [crtl]+s 则画面会被冻结。你可以使用 [ctrl]+q 来解除冻结
9. Linux 默 认的 shell 是 bash ,亦即是 /bin/bash
10. 想要删除光标所在处到最前 面的指令串内容 按下 [crtl]+u 组合键即可
11. bash 环境配置文件主要分为哪两种类型的读取?分别读取哪些重要文件?(1)login shell:主要读取 /etc/profile 及 ~/.bash\_profile (2)non-logni shell:主要读取 ~/.bashrc 而已
12. 如何将 last 的结果中,独立出帐号,并且印出曾经登陆过的帐号? **last | cut -d ' ' -f1 | sort | uniq**
13. 以正则表达式的方式处理字串,就得要使用支持正则表达式的工具程序才行, 这类的工具 程序很多,例如 vi, sed, awk 等等。 正则表达式就是处理字串的方法,他是以行为单位来进行字串的处理行为, 正则 表达式通过一些特殊符号的辅助,可以让使用者轻易的达到“搜寻/删除/取代”某特定字串的处 理程序 。
14. 万用字符 (wildcard) 代表的是 bash 操作接口的一个功能，但正则表达式则 是一种字串处理的表示方式
15. grep 的一些进阶选项 **grep [-A] [-B] [--color=auto] '搜寻字串' filename**
    1. dmesg | grep 'vfs'
    2. 搜寻特定字串 **grep -n 'the' regular\_express.txt**
    3. 当该行没有 'the' 这个字串时才显示在屏幕上 **grep -vn 'the' regular\_express.txt**
    4. 搜寻不论大小写的 the 这个字串 **grep -in 'the' regular\_express.txt**
    5. 利用中括号 [] 来搜寻集合字符 搜寻 test 或 taste 这两个单字

**grep -n 't[ae]st' regular\_express.txt**

1. 如果想要搜寻到有 oo 的字符 **grep -n 'oo' regular\_express.txt**
2. 如果我不想要 oo 前面有 g 的话 ,可以利用在集合字符的反向选择 来达成

**grep -n '[^g]oo' regular\_express.txt**

1. 假设我 oo 前面不想要有小写字符,所以,我可以这样写 abcd....zoo ,由于小写字符的 ASCII 上编码的顺序是连续的, 可以将之简化为 **grep -n '[^a-z]oo' regular\_express.txt**

**grep -n '[^[:lower:]]oo' regular\_express.txt [:lower:] 代表的就是 a-z 的意思**

1. 当我们在一组集合字符中,如果该字符组是连续的,例如大写英文/小写英文/数字 等等, 就可以使用[a-z],[A-Z],[0-9]等方式来书写,那么如果我们的要求字串是数字与英文, 就将他全部写在一起,变成:[a-zA-Z0-9]. 我们要取得有数字的那一行: **grep -n '[0-9]' regular\_express.txt**

**grep -n '[[:digit:]]' regular\_express.txt**

1. 行首与行尾字符 ^ $ 如果我想要让 the 只在行首列出 要使用定位字符了

**grep -n '^the' regular\_express.txt**

1. 如果我想要开头是 小写字符的那一行就列出 **grep -n '^[a-z]' regular\_express.txt**

**grep -n '^[[:lower:]]' regular\_express.txt**

1. 如果我不想要开头是英文字母 **grep -n '^[^a-zA-Z]'** regular**\_express.txt**

**grep -n '^[^[:alpha:]]' regular\_express.txt**

1. **那个 ^ 符号,在字符集合符号(括号[])之内与之外是不同的! 在 [] 内代表“反向选择”,在 [] 之外则代表定位在行首的意义**
2. 我想要找出 来,行尾结束为小数点 (.) 的那一行 **grep -n '\.$' regular\_express.txt**
3. 因为小数点具有其他意义(下面会介绍),所以必须要使用跳脱字符(\)来加 以解除其特殊意义
4. 如果我想要找出来,哪一行 是“空白行” 因为只有行首跟行尾 (^$),所以,这样就可以找出空白行啦

**grep -n '^$' regular\_express.txt**

1. 省略掉 空白行与开头为 # 的注解行 **grep -n '^$' /etc/rsyslog.conf | grep -v '^#'**
2. 任意一个字符 . 与重复字符 \* . (小数点):代表“一定有一个任意字符”的意思; \* (星星号):代表“重复前一个字符, 0 到无穷多次”的意思,为组合形态.
3. 假设我需要找出 g??d 的字串,亦即共有四个字符, 起头是 g 而结束是 d

**grep -n 'g..d' regular\_express.txt**

1. 如果我想要列出有 oo, ooo, oooo 等等的数据, 也就是说,至少要有两个 (含) o 以上

**grep -n 'ooo\*' regular\_express.txt**

1. 如果我想要字串开头与结尾都是 g,但是两个 g 之间仅能存在至少一个 o ,亦即是 gog, goog, gooog.... 等等 **grep -n 'goo\*g' regular\_express.txt**
2. 如果我想要找出 g 开头与 g 结尾的字串,当中的字符可有可无

**grep -n 'g.\*g' regular\_express.txt**  “.\* 就代表零个或多个任 意字符”的意思

1. 如果我想 要找出“任意数字”的行列 **grep -n '[0-9][0-9]\*' regular\_express.txt**

**grep -n '[0-9]' regular\_express.txt**

1. 限定连续 RE 字符范围 {}
2. 我想要找出两个到五个 o 的连续字 串 **grep -n 'go\{2,5\}g' regular\_express.txt**
3. 如果我想要的是 2 个 o 以上的 goooo....g 呢 **grep -n 'go\{2,\}g' regular\_express.txt**

**grep -n 'gooo\*g' regular\_express.txt**

1. 在正则表达式中,我们要找到 含有以 a 为开头的文件 **ls | grep -n '^a.\*'**
2. 以 ls -l 配合 grep 找出 /etc/ 下面文件类型为链接文件属性的文件名答:由于 ls -l 列出 链接文件时标头会是“ lrwxrwxrwx ”,因此使用如下的指令即可找出结果  **ls -l /etc | grep '^l'**
3. 基础正则表达式字符汇整 (characters)
   1. ^word 意义:待搜寻的字串(word)在行首!范例:搜寻行首为 # 开始的那一行,并 列出行号 > grep -n '^#' regular\_express.txt
   2. word$ 意义:待搜寻的字串(word)在行尾!范例:将行尾为 ! 的那一行打印出来, 并列出行号 > grep -n '!$' regular\_express.txt
   3. . 意义:代表“一定有一个任意字符”的字符!范例:搜寻的字串可以是 (eve) (eae) (eee) (e e), 但不能仅有 (ee) !亦即 e 与 e 中间“一定”仅有 一个字符,而空白字符也是字符! > grep -n 'e.e' regular\_express.txt
   4. \ 意义:跳脱字符,将特殊符号的特殊意义去除!范例:搜寻含有单引号 ' 的那一 行! > grep -n \' regular\_express.txt
   5. \* 意义:重复零个到无穷多个的前一个 RE 字符 范例:找出含有 (es) (ess) (esss) 等等的字串,注意,因为 可以是 0 个,所以 es 也是符合带搜寻字 串。另外,因为 为重复“前一个 RE 字符”的符号, 因此,在 之前必须要紧接着 一个 RE 字符喔!例如任意字符则为 “.” ! > grep -n 'ess\*' regular\_express.txt
   6. [list] 意义:字符集合的 RE 字符,里面列出想要撷取的字符!范例:搜寻含有 (gl) 或 (gd) 的那一行,需要特别留意的是,在 [] 当中“谨代表一个待搜寻 的字符”, 例如“ a[afl]y ”代表搜寻的字串可以是 aay, afy, aly 即 [afl] 代表 a 或 f 或 l 的意思! > grep -n 'g[ld]' regular\_express.txt
   7. [n1- n2] 意义:字符集合的 RE 字符,里面列出想要撷取的字符范围!范例:搜寻含有 任意数字的那一行!需特别留意,在字符集合 [] 中的减号 - 是有特殊意义的, 他代表两个字符之间的所有连续字符!但这个连续与否与 ASCII 编码有关,因 此,你的编码需要设置正确(在 bash 当中,需要确定 LANG 与 LANGUAGE 的变量是否正确!) 例如所有大写字符则为 [A-Z] > grep -n '[A-Z]' regular\_express.txt
   8. [^list] 意义:字符集合的 RE 字符,里面列出不要的字串或范围!范例:搜寻的字串 可以是 (oog) (ood) 但不能是 (oot) ,那个 ^ 在 [] 内时,代表的意义 是“反向选择”的意思。 例如,我不要大写字符,则为 [^A-Z]。但是,需要特别 注意的是,如果以 grep -n [^A-Z] regular\_express.txt 来搜寻,却发现该文件内 的所有行都被列出,为什么?因为这个 [^A-Z] 是“非大写字符”的意思, 因为每 一行均有非大写字符,例如第一行的 "Open Source" 就有 p,e,n,o.... 等等的小 写字 > grep -n 'oo[^t]' regular\_express.txt
   9. {n,m}

意义:连续 n 到 m 个的“前一个 RE 字符” 意义:若为 {n} 则是连续 n 个的前一

个 RE 字符, 意义:若是 {n,} 则是连续 n 个以上的前一个 RE 字符! 范例:在

g 与 g 之间有 2 个到 3 个的 o 存在的字串,亦即 (goog)(gooog)

> grep -n 'go\{2,3\}g' regular\_express.txt

1. sed 可以将数据进行取代、删除、新增、撷取特定行等等的功能
   1. sed [-nefr] [动作]
   2. 以行为单位的新增/删除功能, 将 /etc/passwd 的内容列出并且打印行号,同时,请将第 2~5 行删除 **nl sed.txt | sed '2,5d'**
   3. 至于若是要删除第 3 到最后一行,则是 **nl /etc/passwd | sed '3,$d'** “ $ ”代表最后一行
   4. 在第二行后(亦即是加在第三行)加上“drink tea?”字样 **nl sed.txt | sed '2a drink tea'**
   5. 如果是要在第二行前加上“drink tea?”字样 **nl sed.txt | sed '2i drink tea'**
   6. 在第二行后面加入两行字,例如“Drink tea or .....”与“drink beer?”

**nl sed.txt | sed '2a Drink tea or ......\  
> drink beer ?'**

我们可以新增不只一行喔!可以新增好几行”但是每一行之间都必须要以反 斜线“ \ ”来进行新行的增加

1. 我想将第2-5行的内容取代成为“No 2-5 number”呢 **nl sed.txt | sed '2,5c No 2-5 number'**
2. 仅列出 /etc/passwd 文件内的第 5-7 行 **nl sed.txt | sed -n '5,7p'** 这个 -n 代表的是“安静模式”
3. 部分数据的搜寻并取代的功能, 除了整行的处理模式之外, sed 还可以用行为单位进行部分数据的搜寻并取代的功能 sed 's/要被取代的字串/新的字串/g' , 上表中特殊字体的部分为关键字,请记下来!至于三个斜线分成两栏就是新旧字串的替换
4. **/sbin/ifconfig eth0 | grep 'inet '**

inet addr:10.57.2.15  Bcast:10.57.255.255  Mask:255.255.0.0

1. **/sbin/ifconfig eth0 | grep 'inet ' | sed 's/^.\*inet addr://g'** 10.57.2.15  Bcast:10.57.255.255  Mask:255.255.0.0
2. **/sbin/ifconfig eth0 | grep 'inet ' | sed 's/^.\*inet addr://g' | sed 's/ \*Bcast.\*$//g'**

10.57.2.15

1. 假设我只要 MAN 存在的那几行数据, 但是含有 # 在内的注解我不想要,而且空白行我也不要

**cat /etc/man.config | grep 'MAN'| sed 's/#.\*$//g' | sed '/^$/d'**

1. 直接修改文件内容(危险动作) 利用 sed 将 regular\_express.txt 内每一行结尾若为 . 则换成 !

**sed -i 's/\.$/\!/g' regular\_express.txt**

1. 利用 sed 直接在 regular\_express.txt 最后一行加入“# This is a test”

**sed -i '$a # This is a test' regular\_express.txt**

由于 $ 代表的是最后一行,而 a 的动作是新增,因此该文件最后新增

1. 延伸正则表达式
   1. 我们要去除空白行与行首为 # 的 行列

**grep -v '^$' regular\_express.txt | grep -v '^#' or egrep -v '^$|^#' regular\_express.txt**

1. + 意义:重复“一个或一个以上”的前一个 RE 字符 范例:搜寻 (god) (good) (goood)... 等等的字串。 那个 o+ 代表“一个以上的 o ”所以,下面的执行成果 会将第 1, 9, 13 行列出来。

**egrep -n 'go+d' regular\_express.txt**

1. ? 意义:“零个或一个”的前一个 RE 字符 范例:搜寻 (gd) (god) 这两个字串。 那个 o? 代表“空的或 1 个 o ”所以,上面的执行成果会将第 13, 14 行列出来。 有 没有发现到,这两个案例( 'go+d' 与 'go?d' )的结果集合与 'go\*d' 相同? 想想 看,这是为什么喔!  **egrep -n 'go?d' regular\_express.txt**
2. | 意义:用或( or )的方式找出数个字串 范例:搜寻 gd 或 good 这两个字串,注意,是“或”! 所以,第 1,9,14 这三行都可以被打印出来喔!那如果还想要找出dog 呢?

**egrep -n 'gd|good' regular\_express.txt  egrep -n 'gd|dog' regular\_express.txt**

1. () 意义:找出“群组”字串 范例:搜寻 (glad) 或 (good) 这两个字串,因为 g 与 d 是重复的,所以, 我就可以将 la 与 oo 列于 ( ) 当中,并以 | 来分隔开来, 就可以啦!

**egrep -n 'g(la|oo)d' regular\_express.txt**

1. () + 意义:多个重复群组的判别 范例:将“AxyzxyzxyzxyzC”用 echo 叫出,然后再使 用如下的方法搜寻一下!

**echo 'AxyzxyzxyzxyzC' | egrep 'A(xyz)+C'**

1. 如果你想要查出来文件中含有 ! 与 > 的字行 **grep -n '[!>]' regular\_express.txt**
2. 反向选择这样对否? '[!a-z]'? 呵呵!是错 的呦~要 'a-z 才是对的
3. 格式化打印: printf printf '打印格式' 实际内容
   1. 将刚刚上头数据的文件 (printf.txt) 内容仅列出姓名与成绩:(用 [tab] 分隔)

**printf '%s\t %s\t %s\t %s\t %s\t \n' $(cat printf.txt)**

1. 将上述数据关于第二行以后,分别以字串、整数、小数点来显示

**printf '%10s %5i %5i %5i %8.2f \n' $(cat printf.txt | grep -v Name)**

1. 字符宽度: 12345678 %8.2f意义:00000.00 全部的宽度仅有 8 个字符,整数部分占有 5 个字符,小数点本身 (.) 占一位, 小数点下的位数则有两位
2. 列出 16 进位数值 45 代表的字符为何  **printf '\x45\n'**
3. awk:好用的数据处理工具 awk 是“以行为一次处理的单位”, 而“以字段为最小的处理单 位”
   1. awk '条件类型1{动作1} 条件类型2{动作2} ...' filename
   2. 我们用 last 可以将登陆者的数据取出来, 仅取出前五行 **last -n 5**
   3. 若我想要取出帐号与登陆者的 IP ,且帐号与 IP 之间以 [tab] 隔开 **last -n 5 | awk '{print $1 "\t" $3}'**
   4. awk 的内置变 量 : NF 每一行 ($0) 拥有的字段总数
   5. awk 的内置变 量 : NR 目前 awk 所处理的是“第几行”数据
   6. awk 的内置变 量 : FS 目前的分隔字符,默认是空白键
   7. 列出每一行的帐号(就是 $1);
   8. 列出目前处理的行数(就是 awk 内的 NR 变量)
   9. 并且说明,该行有多少字段(就是 awk 内的 NF 变量)
   10. last -n 5| awk '{print $1 "\t lines: " NR "\t columns: " NF}'
   11. 那假设我要查阅,第三栏小于 10 以下的数据,并且仅列出帐号与第三栏:

cat /etc/passwd | awk '{FS=":"} $3 < 30 {print $1 "\t " $3}'

1. cat /etc/passwd | awk 'BEGIN {FS=":"} $3 < 10 {print $1 "\t " $3}'
2. 如何帮我计算每个人的总额呢?而且我还想要格式化输出. 第一行只是说明, 所以第一行不要进行加总 (NR==1 时处理), 第二行以后就会有加总的情况出现 (NR>=2 以后处理) :

cat pay.txt | awk '**NR==1{**printf "%10s %10s %10s %10s %10s\n",$1,$2,$3,$4,"Total" } NR>=2{total = $2 + $3 + $4; printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total}'

1. awk 的指令间隔:所有 awk 的动作,亦即在 {} 内的动作,如果有需要多个指令辅助时, 可利用分号“;”间隔, 或者直接以 [Enter] 按键来隔开每个指令,例如上面的范例中,鸟哥 共按了三次 [enter]
2. 逻辑运算当中,如果是“等于”的情况,则务必使用两个等号“==”
3. 如果是直接给 一个值,例如变量设置时,就直接使用 = 而已
4. 格式化输出时,在 printf 的格式设置当中,务必加上 \n ,才能进行分行
5. 与 bash shell 的变量不同,在 awk 当中,变量可以直接使用,不需加上 $ 符号
6. awk 的动作内 {} 也是支持 if (条件) 的喔

cat pay.txt | **awk '{if(NR==1)** printf "%10s %10s %10s %10s %10s\n",$1,$2,$3,$4,"Total"}

NR>=2{total = $2 + $3 + $4

printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total}'

1. 文件比对工具
   1. diff
      1. 我们要将 /etc/passwd 处理成为一个新的版本,处理方 式为: 将第四行删除,第六行则取代成为“no six line”,新的文件放置到 /tmp/vitest 里面.

cat /etc/passwd | sed -e '4d' -ed '6c no six line' > /tmp/vitest/passwd

1. **diff [-bBi] from-file to-file**
2. diff passwd.old passwd.new
3. 4d3 <==這裡是說，左邊檔案(/etc/passwd)第四行被刪除 (d)
4. 6c5 <==這裡是說，左邊檔案的第六行被取代成右邊檔案(/tmp/test/passwd)的第五行

1. cmp 主要也是在比对两个文件,他 主要利用“字节”单位去比对.
   1. cmp [-l] file1 file2
   2. 用 cmp 比较一下 passwd.old 及 passwd.new cmp passwd.old passwd.new
2. patch 如果要“升级”呢?就是“将旧的文件升级成为新的文件”时 先比较先旧版本的差异,并将差异档制作成为补丁文件,再由补丁 文件更新旧文件 一般来说,使用 diff 制作出来的比较文件通常使用扩展名为 .patch
3. 那 么如何将旧的文件更新成为新的内容呢 ? 就是将 passwd.old 改成与 passwd.new 相同
   1. mount /dev/sr0 /mnt
   2. rpm -ivh /mnt/Packages/patch-2.\*
   3. cd /tmp ; diff -Naur old/ new/ > test.patch
   4. patch -pN < patch\_file
   5. 将刚刚制作出来的 patch file 用来更新旧版数据 patch -p1 < /tmp/test.patch
   6. 恢复旧文件的内容 patch -R -p0 < passwd.patch
   7. 为什么这里会使用 -p0 呢?因为我们在比对新旧版的数据时是在同一个目录下, 因此不需要 减去目录啦
4. 文件打印准备: pr pr /etc/man\_db.conf
5. grep 与 egrep 在正则表达式里面是很常见的两支程序,其中, egrep 支持更严谨的正则 表达式的语法
6. 基础正则表达式的特殊字符有: \*, ., [], [-], , ^, $ 等!
7. 文件的比对中,可利用 diff 及 cmp 进行比对,其中 diff 主要用在纯文本方面的新旧版本 比对
8. patch 指令可以将旧版数据更新到新版 (主要亦由 diff 创建 patch 的补丁来源文件)
9. grep在单引号内的星号是正则表达式的字符,但我们要找的是星 号,因此需要加上跳脱字符 (\)
10. 找出在 /etc 下面含有星号 (\*) 的文件与内容 grep '\\*' /etc/\*
11. 如果想要连同完整的 /etc 次目录数据下面含有星号 (\*) 的文件与内容 grep '\\*' $(find /etc -type f)
12. 如果要找的是全系统 (/) 目录数据下面含有星号 (\*) 的文件与内容 find / -type f | xargs -n 10 grep '\\*'
    1. 先用 find 去找出文件；
    2. 用 xargs 将这些文件每次丢 10 个给 grep 来作为参数处理；
    3. grep 实际开始搜寻文件内容。
13. 如果要找的是全系统 (/) 目录数据下面含有星号 (\*) 的, 只是想要知道档名 find / -type f | xargs -n 10 grep -l '\\*'
14. 在 /etc 下面,只要含有 XYZ 三个字符的任何一个字符的那一行就列出来 grep [XYZ] /etc/\*
15. 将 /etc/kdump.conf 内容取出后,(1)去除开头为 # 的行 (2)去除空白行 (3)取出 开头为英文字母的那几行 (4)最终统计总行数该如何进行 grep -v '^#' /etc/kdump.conf | grep -v '^$' | grep '^[[:alpha:]]' | wc -l
16. 如何执行这个文件 shell.sh
    1. 直接指令下达: shell.sh 文件必须要具备可读与可执行 (rx) 的权限,然后
    2. 以 bash 程序来执行:通过“ bash shell.sh ”或“ sh shell.sh ”来执行
    3. 变量“PATH”功能:将 shell.sh 放在 PATH 指定的目录内,例如: ~/bin/
    4. 建议务必要将一些重要的环境变量设置好,鸟哥个人认为, PATH 与 LANG (如果有使用到输出相关的信息时) 是当中最重要的
    5. 一个指令的执行成功与 否,可以使用 $? 这个变量来观察
    6. 利用直接执行的方式来执行 script sh showname.sh
    7. 利用 source 来执行脚本:在父程序中执行 source showname.sh
17. 随日期变化:利用 date 进行文件的创建
    1. date --date='2 days ago' +%Y%m%d 20170212
18. test 指令的测试功能
    1. 关于某个文件名的“文件类 型”判断,如 test -e filename 表 示存在否
    2. -e 该“文件名”是否存在?(常用)
    3. -f 该“文件名”是否存在且为文件(file)?(常用)
    4. -d 该“文件名”是否存在且为目录(directory)?(常 用)
19. 利用判断符号 [ ]
    1. 在中括号 [] 内的每个元件都需要有空白键来分隔;
    2. 在中括号内的变量,最好都以双引号括号起来;
    3. 在中括号内的常数,最好都以单或双引号括号起来。
20. Shell script 的默认变量($0, $1...)
    1. /path/to/scriptname opt1 opt2 opt3 opt4
    2. $0 $1 $2 $3 $4
    3. $# :代表后接的参数“个数”,以上表为例这里显示为“ 4 ”;
    4. $@ :代表“ "$1" "$2" "$3" "$4" ”之意,每个变量是独立的(用双引号括起来);
    5. $\* :代表“ "$1<u>c</u>$2<u>c</u>$3<u>c</u>$4" ”,其中 <u>c</u> 为分隔字符,默 认为空白键, 所以本例中代表“ "$1 $2 $3 $4" ”之意。
21. Shell script 的shift：造成参数变数号码偏移，shift 会移动变量,而且 shift 后面可以接数字,代表拿掉最前面的几个参数的意思
    1. echo "Total parameter number is ==> $#"
    2. echo "Your whole parameter is ==> '$@'"
    3. shift # 进行第一次‘一个变数的 shift ’
    4. echo "Total parameter number is ==> $#"
    5. echo "Your whole parameter is ==> '$@'"
    6. shift 3 # 进行第二次‘三个变数的 shift ’
    7. echo "Total parameter number is ==> $#"
    8. echo "Your whole parameter is ==> '$@'"
22. Shell script 的条件判断式
    1. **单层、简单条件判断式**

if [ 条件判断式 ]; then

当条件判断式成立时，可以进行的指令工作内容；

fi <==将 if 反过来写，就成为 fi 啦！结束 if 之意

1. **多重、复杂条件判断式**

**# 一个条件判断，分成功进行与失败进行 (else)**

if [ 条件判断式 ]; then

当条件判断式成立时，可以进行的指令工作内容；

else

当条件判断式不成立时，可以进行的指令工作内容；

fi

**# 多个条件判断 (if ... elif ... elif ... else) 分多种不同情况执行**

if [ 条件判断式一 ]; then

当条件判断式一成立时，可以进行的指令工作内容；

elif [ 条件判断式二 ]; then

当条件判断式二成立时，可以进行的指令工作内容；

else

当条件判断式一与二均不成立时，可以进行的指令工作内容；

fi

1. netstat -tuln
2. Shell script 利用 case ..... esac 判断

**case $變數名稱 in** <==關鍵字為 case ，還有變數前有錢字號

**"第一個變數內容")** <==每個變數內容建議用雙引號括起來，關鍵字則為小括號 )

**程式段**

**;;** <==每個類別結尾使用兩個連續的分號來處理！

**"第二個變數內容")**

**程式段**

**;;**

**\*)** <==最後一個變數內容都會用 \* 來代表所有其他值

**不包含第一個變數內容與第二個變數內容的其他程式執行段**

**exit 1**

**;;**

**esac** <==最終的 case 結尾！『反過來寫』思考一下！

1. 一般来说,使用“ case $变量 in ”这个语法中,当中的那个“ $变量 ”大致有两种取得的方式:
2. 直接下达式:例如上面提到的,利用“ script.sh variable ” 的方式来直接给 $1 这个变量 的内容,这也是在 /etc/init.d 目录下大多数程序的设计方式。
3. 互动式:通过 read 这个指令来让使用者输入变量的内容。
4. /etc/init.d/netconsole restart
5. Shell script 的利用 function 功能

**function fname() {**

**程式段**

**}**

因 为 shell script 的执行方式是由上而下,由左而右, 因此在 shell script 当中的 function 的设置一定要在程序的最前面, 这样才能够在执行时被找到可用的程序段喔

1. Shell script 的loop
   1. **while do done (不定迴圈)**

while [ condition ] <==中括號內的狀態就是判斷式

do <==do 是迴圈的開始！

程式段落

done <==done 是迴圈的結束

1. **until do done (不定迴圈)**

until [ condition ]

do

程式段落

done

1. **for...do...done (固定迴圈)**

for var in con1 con2 con3 ...

do

程式段

done

1. **for...do...done 的數值處理**

for (( 初始值; 限制值; 執行步階 ))

do

程式段

done

1. 11