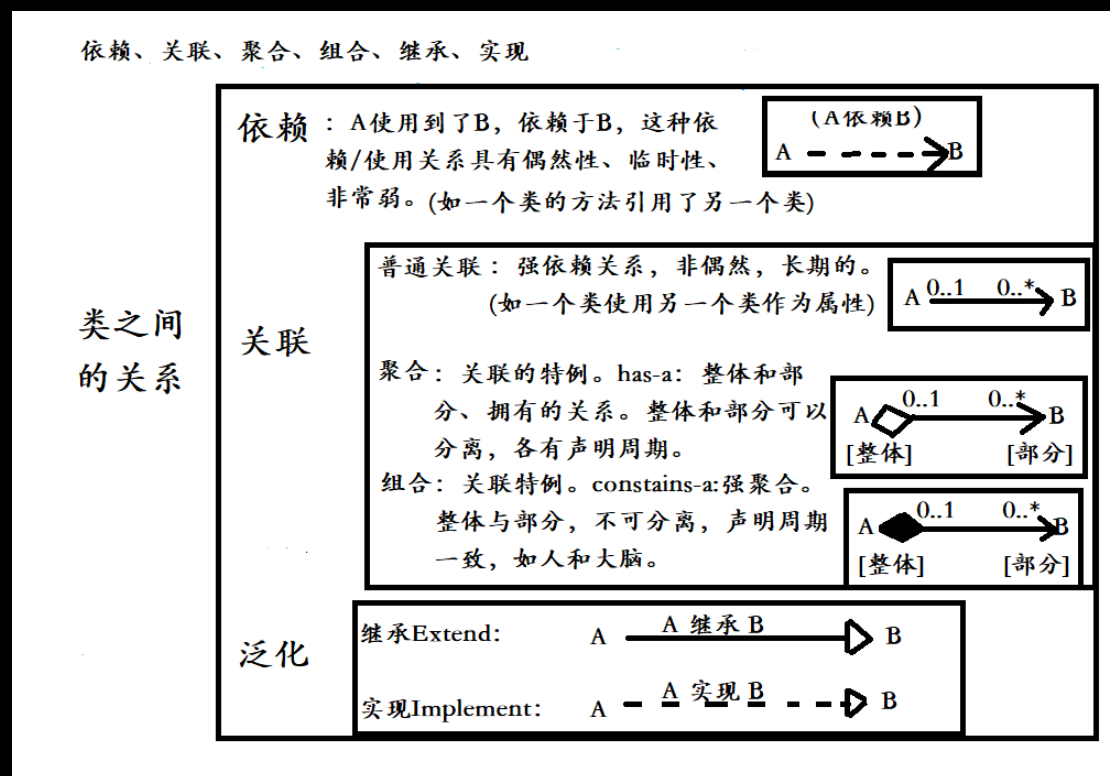


# 计算机知识积累



Chrome:

F12, Ctrl+Shift+P

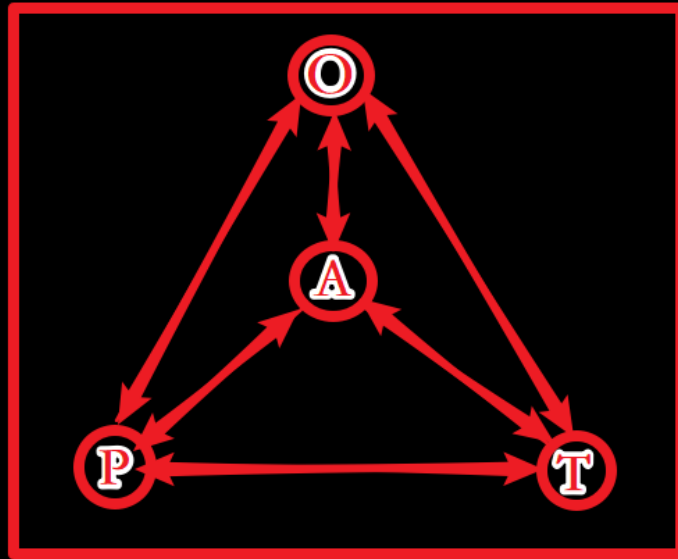
工具集合

## 软件复杂度

张贺:

所有的软件系统都几个有方面:

1. Business: 目标
2. Organization
3. Process
4. Technology
5. Architecture



复杂度之间相互转换：

当复杂度不能在简化时，为了优化系统性能，可以将复杂度转移到其他部分。

如微服务，将复杂度外化，其他三个方面需要消化这些增加的复杂度

## 回车和换行

历史：

计算机出现之前，有一个打字机较电传打字机，每秒可以打印 10 个字符，但是打完一行到下一行时会耽误 0.2 秒，那么这时就会丢失两个字符，于是发明了两个字符：

1. 回车：carriage return (**CR**)，把打印机打印头定位到左边界
2. 换行：Line feed (**LF**)，打印头向下移动一行

但是，在哪个存储十分贵的年代，很多人认为两个字符太浪费资源了，于是出现分歧：


1. Unix 系统：行尾是 “\n”。
2. Windows 系统：行尾是 “\n\r”
3. MacOS 系统：每行结尾是 “\r”

一个直接的后果是：Unix/Mac 系统的文件在 win 里打开的话所有文字变成一行，而 win 里面的文件在 Unix/Mac 里面打开的话行尾可能多一个 ^M 符号。

软回车和硬回车

1. 硬回车：Enter，回车并换行
2. 软回车：shift+Enter 换行不换段

如：

This is paragraph.  软回车，箭头向下，这两行是同一段  
 this is in the same paragraph. 硬回车，箭头下偏左  
 But this is not in the same paragraph. 这行和上一行是不同段

## Java 的几个 ‘J’

JDK: Java Development Kit Java 开发工具

for: 开发者

JRE: Java Runtime Environment Java 运行环境

for: 用户

JVM: Java Virtual Machine java 虚拟机

for :用户

JDK 包含 JRE, JRE 包含 JVM

对于一个程序，一般用户只有安装了 JRE 才能使用，这是 Java 的巨大缺点，不像 c/c++那样生成 exe 在所有电脑中都可以运行。

Java 长于 web (Html CSS js 负责页面的设计, Java 负责网络, 架构)、Android 开发; 不长于桌面应用, GUI 设计 (awt/Swing 不长于图形)

桌面应用是 C、C++, c# 的长处

## ArrayList LinkedList

1. ArrayList 是实现了基于动态数组的数据结构, LinkedList 基于链表的数据结构。

2. 对于随机访问 get 和 set, ArrayList 觉得优于 LinkedList, 因为 LinkedList 要移动指针。

3. 对于新增和删除操作 add 和 remove, LinedList 比较占优势, 因为 ArrayList 要移动数据 in a word:

ArrayList 和 LinkedList 在性能上各有优缺点, 都有各自所适用的地方, 总的说来可以描述如下:

1. 对 `ArrayList` 和 `LinkedList` 而言，在列表末尾增加一个元素所花的开销都是固定的。对 `ArrayList` 而言，主要是在内部数组中增加一项，指向所添加的元素，偶尔可能会导致对数组重新进行分配；而对 `LinkedList` 而言，这个开销是统一的，分配一个内部 `Entry` 对象。
  2. 在 `ArrayList` 的中间插入或删除一个元素意味着这个列表中剩余的元素都会被移动；而在 `LinkedList` 的中间插入或删除一个元素的开销是固定的。
  3. `LinkedList` 不支持高效的随机元素访问。
  4. `ArrayList` 的空间浪费主要体现在在 `list` 列表的结尾预留一定的容量空间，而 `LinkedList` 的空间花费则体现在它的每一个元素都需要消耗相当的空间
- 可以这样说：当操作是在一系列数据的后面添加数据而不是在前面或中间，并且需要随机地访问其中的元素时，使用 `ArrayList` 会提供比较好的性能；当你的操作是在一系列数据的前面或中间添加或删除数据，并且按照顺序访问其中的元素时，就应该使用 `LinkedList` 了。

## 防御式编程 VS 进攻式编程

### 1. 防御式编程：

业务的流转应该在参数的正确性下进行。简言之，就是先进行安全性或者存在性检查，然后再执行业务流程。

在没有真正的操作数据之前，要保证所有的操作或者参数都是正确的，要满足所有的约束——只做正确的事

### 2. 进攻式编程：

一般情况下，调用环境已经把要传入的程序参数进行验证以保证他们大部分是正确的，大部分约束是满足的，再采用防御式编程浪费资源，这是就该采用进攻式编程：先不管参数是否完全满足，约束是否全满足，先执行操作，出现错误时再进行处理。

## FORTRAN

Fortran 源自于“公式翻译”（英语：Formula Translation）的缩写，是一种编程语言。它是世界上最早出现的计算机高级程序设计语言，广泛应用于科学和工程计算领域。FORTRAN 语言以其特有的功能在数值、科学和工程计算领域发挥着重要作用。

## URL 转义

URL 中的字符只能是 ASCII 码，非 ASCII 字符需要进行编码/转义。

一个非 `ascii` 被替换为：`%XX` （两个 16 进制数），它对应于 ISO-8859-1 字符集里的编码值

Edwin Xu

中国”：“%D6%D0%B9%FA”（一个汉字 2 字符）

常见：

空格	-	%20
"	-	%22
#	-	%23
%	-	%25
&	-	%26
(	-	%28
)	-	%29
+	-	%2B
,	-	%2C
/	-	%2F
:	-	%3A
;	-	%3B
<	-	%3C
=	-	%3D
>	-	%3E
?	-	%3F
@	-	%40
\	-	%5C
	-	%7C

## Java 编译还是解释

一、你可以说它是编译型的。因为所有的 Java 代码都是要编译的，.java 不经过编译就什么用都没有。

二、你可以说它是解释型的。因为 java 代码编译后不能直接运行，它是解释运行在 JVM 上的，所以它是解释运行的，那也就算是解释的了。

三、但是，现在的 JVM 为了效率，都有一些 JIT 优化。它又会把.class 的二进制代码编译为本地的代码直接运行，所以，又是编译的。

## 环回

环回：一个回环，绕一圈，哪里出哪里入，没离开设备

### 环回接口

在网络设备（一般是路由器）上是一种特殊的接口，它不是物理接口，而是一种看不见摸不着的逻辑接口（也称虚拟接口），但是对于网络设备来说却是至关重要的。

在网络设备上可以通过配置命令来创建一个或多个环回接口，并且可以和配置物理接口一样，配置环回接口的 IP 地址和掩码，环回接口的掩码一般为全 1，即 255.255.255.255。环回接口有一个特性，除非设备瘫痪，否则其状态一直是 up。这个特性对于路由协议来说非常重要。环回接口是使用广泛的一种逻辑接口。在一个网络中，不同设备的环回接口地址以及同一设备上的不同环回接口地址应该统一规划，避免重复。

### 环回地址

windows 系统自带的虚拟网卡，这块网卡与任何外部网络不能通讯，只限于本机通讯。

**127.0.0.1**，通常被称为本地回环地址(Loop back address)，不属于任何一个有类别地址类。

它代表设备的本地虚拟接口，所以默认被看作是永远不会宕掉的接口

## 手机内存

为什么国内手机一般是 8G，而国外很多手机都是 4G？

一般来说，手机软件即使在关闭状态下也能接受消息，比如微信关闭，但是有人发消息给你，你仍然是可以收到的，这是什么情况？

先看国内的情况：

国内的手机软件在关闭时仍可以收到消息，说明这个软件没有完全关闭。是的，需要消息推送的软件虽然关闭了，但是后台还运行着一个进程，专门用于信息推送的接收。

那么一个软件的后台进程可能只需要很小内存，但是手机系统几十上百应用累加起来会占用很大内存，这就是为什么国内手机内存 4G 太小，现在已经不匹配了。

国外：

国外很多手机厂商不是这样的。软件关了就是完全关了，手机是通过实时推送服务系统来推送消息的，意思是消息推送是系统服务实现的，而不是软件本省。

实现方式是一个服务器，和手机系统相连，软件有消息先发给服务器在转给手机。

比如，手机微信关闭状态下。国内安卓手机微信，别人给你发消息，先到微信服务器，服务器

在转给你手机后台运行的微信进程。而国内 Apple 手机，别人消息发给微信服务器，微信服务器在发给 Apple 服务器，Apple 服务器在发给手机推送服务系统。

于是国外有自己的推送服务集成服务的手机都不需要大的内存，因为没必要，用不了。

以此观之，国内的手机生厂商还是乱套的。

## X86 vs x64

X86 是 32 位，最大识别内存 3.75G

X86 架构 (The X86 architecture) 是微处理器执行的计算机语言指令集，指一个 intel 通用计算机系列的标准编号缩写，也标识一套通用的计算机指令集合。

1978 年 6 月 8 日，Intel 发布了新款 16 位微处理器“8086”，也同时开创了一个新时代：x86 架构诞生了。Intel 8086 处理器 x86 指的是特定微处理器执行的一些计算机语言指令集，定义了芯片的基本使用规则，一如今天的 x64、IA64

## OS 内存分配和调用

很多书表明，现代操作系统不需要 delete 来释放内存。

现代 OS 中，new 操作最后被编译器链接到 malloc 上，malloc 是一个系统调用，因此内存的分配最终是由 OS 而不是编译器完成。

一般 OS 有一块很大的自由内存区——堆，使用 new 时就是在堆上分配。使用 new 产生一个系统调用，OS 先检查这块堆，找到一块大小适合且被标记为‘未分配’的内存，并返回它的指针，然后在标记为‘已分配’，调用 delete 时再将其标记为‘可分配’，在 free 或 delete 时不需要指明长度，只需要首地址，因为在分配时 OS 已经记下了长度。

现代 OS，在分配内存时，还会记录内存被分配给哪个进程了，于是在进程退出时就可以强制回收分配给它的所有空间。于是似乎不用 delete 也可以回收内存，但是是只有在进程挂掉才回收。

OS 的强制回收和垃圾回收机制不同，在 java 的回收机制中，先不断分配内存，能分配就分配，如果不能分配了，就查找哪些不能被程序使用的内存并回收它。OS 不具备这样的能力，OS 不知道哪些内存已经不被程序使用了，它认为只要没有 delete 就还在使用。如在 java 和 C 中，java 在运行过程中边分配边回收，而 c 只分配不回收，当没有内存可用的时候，只有杀死进程一次全部回收。

但是在有的 OS 上，进程结束并不能全部回收。因为有的数据是全局数据，这个内存块是多个进程共享，这时对于每一块内存的使用，有一个引用计数值，new 一次就+1，delete 就-1，只有计数=0 才回收，如果忘了 delete 就会导致这块内存一直被保留。

综上，内存分配和回收是 OS 的责任，而编译器只是调用了相关 calls，new 的内存有可能一直保留，所以一定要 delete。

## URL 中的+

HTML 有一些非标准的做法——**将+当做空格处理**

在 URL 中输入加法，会被转义为空格

要想不被转化，一种方法是**使用%2B 替换+**

注意：不同浏览器的解析不一样，百度浏览器就会解析为空

## 随机存储器的[随机]

所谓「随机访问」，指的是当存储器中的讯息被读取或写入时，所需要的时间与这段信息所在的位置无关。相对的，存取顺序访问 (Sequential Access) 存储设备中的信息时，其所需要的时间与位置就会有关系（如磁带）。

即随机存储器可以直接访问任意位置，每一个位置的访问花的时间一样。而其他存储器则不能直接访问，比如串行，每个地方都可以读取，但是需要排队，每个地方存取的时间不同。

## 原语

Primitive(or: atomic action), 原语是由**若干条指令组成**的用于完成一定功能的一段程序，

具有**不可分割性**，即原语是连续的，**执行时不被中断**。

## 原子性

程序的原子性指：整个程序中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。

原子性在一个操作是**不可中断**的，要么全部执行成功要么全部执行失败，有着“同生共死”的感觉。及时在多个线程一起执行的时候，一个操作一旦开始，就不会被其他线程所干扰。

Edwin Xu



如果要保证原子性，必须符合以下两条规则：

- 1、运算结果并不依赖于变量的当前值，或者能够确保只有一个线程修改变量的值。
- 2、变量不需要与其他的状态变量共同参与不变约束。

```
>>> 111111111*111111111
12345678987654321
```

## 命令行设置环境变量

- Set：查看所有
- Set A：查看 A
- Set A = yourpath：添加(覆盖)环境变量
- Set A = %PATH%;yourpath；追加环境变量

如配置 MongoDB 环境变量：

```
Set path = %PATH%;D:\MongoDB\bin
```

(注意：中间的分号不能省略)

## 计算机发展 4 阶段

- 第一代计算机 1946 1957 **电子管** 运算速度较低，耗电量大存储容量小。
- 第二代计算机 1958 1964 **晶体管** 体积小，耗电量较少，运算速度高，价格下降。
- 第三代计算机 1965 1971 **中小规模集成电路** 体积功能进一步减少，可靠性及速度进一步提高。
- 第四代计算机 **1972 年至今** **大规模及超大规模集成电路** 性能到规模提高，价格大幅度降低，广泛应用于社会生活的各个领域，走进办公室和家庭

## 鲁棒

是 Robust 的音译，也就是健壮和强壮的意思。它是在异常和危险情况下系统生存的关键。比如说，计算机软件在输入错误、磁盘故障、网络过载或有意攻击情况下，能否不死机、不崩溃，就是该软件的鲁棒性。所谓“鲁棒性”，是指控制系统在一定（结构，大小）的参数扰动下，维持其它某些性能的特性。根据对性能的不同定义，可分为稳定鲁棒性和性能鲁棒性。以闭环系统的鲁棒性作为目标设计得到的固定控制器称为鲁棒控制器。

# BSS

## Block Storage Space

表示未被初始化的数据空间

### 1. 注销

关闭当前用户，释放当前用户使用的系统资源。不影响其它用户正常使用。

### 2. 休眠

当前内存运行状态和数据保存在硬盘中，cpu、硬盘、内存 不供电，这种模式完全不耗电。下次可以恢复到休眠前的状态，从硬盘恢复速度会慢一些。

### 3. 睡眠

内存运行状态和数据保存在硬盘中，给内存供电，  
如果没有断电，下次直接读取内存状态，快速恢复计算机状态。  
如果断电，下次从硬盘恢复数据，恢复速度会慢一点。

# 精神模型

## Wi-Fi

Wi-Fi 这个术语是指无线保真 (Wireless Fidelity)，Wi-Fi 联盟本身也经常的新闻稿和文件中使用”无线保真”这个词，Wi-Fi 还是出现在 ITAA 的一个论文中。然而，根据菲尔贝朗格的语句，Wi-Fi 术语应该是没有任何意义的。IEEE 802.11 第一个版本发表于 1997 年，其中定义了介质访问接入控制层和物理层。物理层定义了工作在 2.4GHz 的 ISM 频段上的两种无线调频方式和一种红外传输的方式，总数据传输速率设计为 2Mbit/s。两个设备之间的通信可以自由直接 (ad hoc) 的方式进行，也可以在基站 (Base Station, BS) 或者访问点 (Access Point, AP) 的协调下进行。

1999 年加上了两个补充版本：802.11a 定义了一个在 5GHz ISM 频段上的数据传输速率可达

54Mbit/s 的物理层，802.11b 定义了一个在 2.4GHz 的 ISM 频段上但数据传输速率高达 11Mbit/s 的物理层。

2.4GHz 的 ISM 频段为世界上绝大多数国家通用，因此 802.11b 得到了最为广泛的应用。苹果公司把自己开发的 802.11 标准起名叫 AirPort。1999 年工业界成立了 Wi-Fi 联盟，致力解决符合 802.11 标准的产品的生产和设备兼容性问题。Wi-Fi 为制定 802.11 无线网路的组织，并非代表无线网路。

## DoS 攻击、DDoS 攻击和 DRDoS 攻击

| =

```

..if (ptcb->OSTCBStat == OS_STAT_RDY) {
    ..OSRdyGrp |= bity;
    ..OSRdyTbl[y] |= bitx;
..}
..return (prio);

```

差点没看出来，| 是按位或

A | = B 就是 A = A | B

## DSP

**项目文件名后缀为 dsp** (保存项目设置)，它维护应用程序中所有的源代码文件，以及

Visual C++ 如何编译、连接应用程序，以便创建可执行程序。Visual C++6 的集成开发环境中，通过 "File" 菜单的 "New" 命令创建一个新的项目。创建一个项目的同时，也创建了一个项目工作区，**项目工作区文件的后缀名为 dsw** (保存项目工作区的设置)。一个应用程序可以有一个项目及若干个子

Edwin Xu

项目，但只有一个活动的项目。

## 软盘

软盘 (**Floppy Disk**) 是个人计算机 (PC) 中最早使用的可移介质。软盘的读写是通过软盘驱动器完成的。软盘驱动器设计能接收可移动式软盘，目前常用的就是容量为 **1.44MB 的 3.5 英寸软盘**。

软盘存取速度慢，容量也小，但可装可卸、携带方便。它作为一种可移储存硬件，适用于一些**需要被物理移动的小文件**。

- 尺寸容量

有八寸、五又四分之一寸、三寸半之分。

当中又分为硬磁区 Hard-sectored 及软磁区 Soft-Sectored。

磁盘片的容量有 5.25” 的 1.2MB，3.5” 的 1.44MB。以 3.5” 的磁盘片为例，其容量的计算如下：

$$80 \text{ (磁道)} \times 18 \text{ (扇区)} \times 512 \text{ bytes (扇区的大小)} \times 2 \text{ (双面)} = 1440 \times 1024 \text{ bytes} = 1440 \text{ KB} = 1.44\text{MB}$$

- 软盘片的存储格式：指盘片的每面划分为多少个同心圆式的磁道，以及每个磁道划分成多少个存储信息的扇区。
- 软盘驱动器曾经是电脑一个不可缺少的部件，在必要的时候，它可以为我们 启动计算机，还能用它来传递和备份一些比较小的文件。
- 软盘都是 3.5 英寸的，通常简称 3 寸。3 寸软盘都有一个塑料外壳，比较硬，它的作用是保护里边的盘片。盘片上涂有一层磁性材料(如氧化铁)，它是记录数据的介质。在外壳和盘片之间有一层保护层，防止外壳对盘片的磨损。

- 软盘在使用之前必须要先格式化

## 第一个浏览器

蒂姆·伯纳斯-李在 1990 年发明了第一个网页浏览器 WorldWideWeb，此浏览器后改名为 Nexus。

## 五大浏览器

- Chrome
- Opera
- FireFox
- Safari
- IE

## DLL

DLL(Dynamic Link Library)文件为动态链接库文件, 又称"应用程序拓展", 是软件文件类型。在 Windows 中, 许多应用程序并不是一个完整的可执行文件, 它们被分割成一些相对独立的动态链接库, 即 DLL 文件, 放置于系统中。当我们执行某一个程序时, 相应的 DLL 文件就会被调用。一个应用程序可使用多个 DLL 文件, 一个 DLL 文件也可能被不同的应用程序使用, 这样的 DLL 文件被称为共享 DLL 文件。

## GPU

图形处理器 (英语: Graphics Processing Unit, 缩写: GPU), 又称显示核心、视觉处理器、显示芯片, 是一种专门在个人电脑、工作站、游戏机和一些移动设备 (如平板电脑、智能手机等) 上做图像和图形相关运算工作的微处理器。

GPU 使显卡减少了对 CPU 的依赖, 并进行部分原本 CPU 的工作, 尤其是在 3D 图形处理时 GPU 所采用的核心技术有硬件 T&L (几何转换和光照处理)、立方环境材质贴图和顶点混合、纹理压缩和凹凸映射贴图、双重纹理四像素 256 位渲染引擎等, 而硬件 T&L 技术可以说是 GPU 的标志。GPU 的生产商主要有 NVIDIA 和 ATI。

## Img 格式

img 格式是一种文件压缩格式 (archive format), 主要是为了创建软盘的镜像文件

**(disk image)**，它可以用来压缩整个软盘（通常指软软盘，Floppy Disk 或 Diskette）或整片光盘的内容，使用".IMG"这个扩展名的文件就是利用这种文件格式来创建的。.IMG 这个文件格式可视为.ISO 格式的一种超集合。

img 格式是图像文件的一种格式，它具有很高的压缩效率，IMG 格式支持任意大小的图像。

图像的数据是以类似二维数组格式存放的。在其第一行的头两个位置存放的是图像的宽度，其后面的两位是存放着图像的高度，接着的一个位置里存放着图像的灰度级，而其剩下的所有位置存放的都是图像的灰度。

img 格式属于镜像的一种，可以通过制作数据光盘或者使用虚拟光驱(如 WinMount)安装 IMG 数据文件。由于.ISO 只能压缩使用 ISO9660 和 UDF 这两种文件系统的存储媒介，意即.ISO 只能拿来压缩 CD 或 DVD，因此才发展出了.IMG，它是以.ISO 格式为基础另外新增可压缩使用其它文件系统的存储媒介的能力，.IMG 可向后兼容于.ISO，如果是拿来压缩 CD 或 DVD，则使用.IMG 和.ISO 这两种格式所压缩出来的内容是一样的。img 格式的打开方式可以是光盘刻录，也可以用软件解压。IMG 可以做为以下用途：数字存储、传输、以及整片软盘内容的复制，可挂载到虚拟软盘上。

## WinImage

首先对软件进行压缩处理，然后将软件 COPY 到软盘上去。当然，他们也能分拆那些实在在一张盘上容纳不下的软件，将它分开 COPY 到几张软盘上，最后再将它组合起来。

## WSL

### Windows Subsystem for Linux

是一个在 Windows 10 上能够运行原生 Linux 二进制可执行文件（ELF 格式）的兼容层。它是由微软与 Canonical 公司合作开发，其目标是使纯正的 Ubuntu 14.04 "Trusty Tahr" 映像能下载和解压到用户的本地计算机，并且映像内的工具和实用工具能在此子系统上原生运行。

WSL 提供了一个微软开发的 Linux 兼容内核接口（不包含 Linux 代码），来自 Ubuntu 的用户模式二进制文件在其上运行。

该子系统不能运行所有 Linux 软件，例如那些图形用户界面，以及那些需要未实现的 Linux 内核服务的软件。不过，这可以用在外部 X 服务器上运行的图形 X Window 系统缓解。

## BNF form

**巴科斯范式** 以美国人巴科斯(Backus)和丹麦人诺尔(Naur)的名字命名的一种形式化的语法表示方法，用来描述语法的一种形式体系，是一种典型的元语言。又称巴科斯-诺尔形式(Backus-Naur  
Edwin Xu

form)。它不仅能严格地表示语法规则，而且所描述的语法是与上下文无关的。它具有语法简单，表示明确，便于语法分析和编译的特点。BNF 表示语法规则的方式为：

非终结符用尖括号括起。每条规则的左部是一个非终结符，右部是由非终结符和终结符组成的一个符号串，中间一般以“::=” 分开。具有相同左部的规则可以共用一个左部，各右部之间以直竖“|” 隔开。

## .msi 和 .exe 文件的区别

如果我们的操作系统(安装环境)没有安装某些程序，则 .MSI 有可能不能运行，这时就要用 Setup.exe 来进行安装了。Setup.exe 可以利用 Setup.ini 来先安装运行 .MSI 需要的软件，建造一个较全的安装环境，最后再调用 .MSI 程序。

.exe 文件进行安装的时会检测安装软件需要的环境和一些必要的组件， 适不适合当前软件安装，如果缺少一些例如 .netframework 一类的组件，就会先进行下载然后再进行安装

.msi 文件不检测当前系统环境是否符合就直接进行安装， 如果环境不符合运行到一半可能会停止安装,并报错或提示,其实是 Windows Installer 在执行 MSI 包定义的各项操作。因此我们需要安装 Windows Installer 的正确版本才能运行 setup.msi

## 磁盘映像

磁盘像 (Disk Image)，是指将有某种储存装置（例如 CD）的完整内容及结构保存为一个电脑档案，所以通常这些档案都会是很大。一般只有一个档案，里面包含了许多的档案备份。镜像的制作方法除了工具把实体磁盘的内容保存起来之外，也有专门的工具可不从实体磁盘制作出镜像（例如不需读取实体 CD 即可制作 CD 镜像）。

最常用到的磁盘镜像是光盘镜像，是指从 CD 或 DVD 制作的镜像。简单地说，光盘镜像就是 CD 或 DVD 的拷贝，所有的资料都存在一个档案中，借此保存 CD 或 DVD 的结构及完整性。光盘镜像通常是以 ISO 9660 格式存储的，其扩展名为 .iso。

随身碟或软碟的镜像档为 IMG 格式；而诺顿魅影系统 (Ghost) 则可以为硬盘产生 GHO 格式（版本 9.0 以后为 V2I 格式）的镜像；Mac OS X 常用的磁盘映像文件为 DMG 格式，常常用于打包软件用于网络分发，或备份磁盘等。

Wiki:

磁盘映像是计算机领域中的一个计算机文件，其包含一个磁盘卷或数据存储设备的内容和结构，包括但不限于硬盘、软盘、磁带、光盘、USB 闪存盘等。磁盘映像通常是按照原介质的扇区级复制，从而完全复制存储设备文件系统的结构和内容。根据磁盘映像的格式不同，一个映像可能表现为一个或多个计算机文件。

磁盘映像的文件格式可能是开放标准，例如用于**光盘映像的 ISO** 映像格式；但也可能是特定应用程序的专有标准。

因为**磁盘映像包含整个磁盘的内容**，所以它们通常体积庞大。部分磁盘映像工具可以识别和忽略源介质中未使用的空间，或者压缩映像内容以减少存储所需空间。

## RAR

RAR 是一种专利文件格式，用于数据压缩与归档打包，开发者为尤金·罗谢尔（俄语：Евгений Лазаревич Рощал，拉丁转写：Yevgeny Lazarevich Roshal），RAR 的全名是“Roshal ARchive”，即“罗谢尔的归档”之意。首个公开版本 RAR 1.3 发布于 1993 年。

## .COM

COM 是 Component Object Model（组件对象模型）的缩写。

COM 是微软公司为了计算机工业的软件生产更加符合人类的行为方式开发的一种新的软件开发技术。在 COM 构架下，人们可以开发出各种各样的功能专一的组件，然后将它们按照需要组合起来，构成复杂的应用系统。

组件实际上是一些小的二进制可执行程序，它们可以给应用程序，操作系统以及其他组件提供服务。开发自定义的 COM 组件就如同开发动态的，面向对象的 API。多个 COM 对象可以连接起来形成应用程序或组件系统。并且组件可以在运行时刻，在不被重新链接或编译应用程序的情况下被卸下或替换掉。

### com 和 exe 文件的区别

- 以 COM 为扩展名的文件的特点如下：

1. 程序只能设置一个段，且不建立堆栈段；
2. 程序的长度必须少于 64K 字节；
3. 程序必须预留 100H 空间，开始处是一条可执行指令；
4. 程序被装入的起始标号必须由 END 语句说明开始地址；
5. 程序中的子程序必须具有进程属性(NEAR)；
6. 如果 COM 文件是由几个不同的目标模块链接生成的，要求所有目标模块具有同一代码段名和类别名(CLASS)，且赋予公共属性(PUBLIC)，而主模块应具有 100H 的入口指针并优先连接。

- EXE 文件的结构特点如下：

1. 程序允许建立若干不同名的代码段、数据段、堆栈段或附加段。



2. 程序的长度仅受当前内存可用空间的限制。
3. 程序的入口随应用而定，只需起始标号与 END 语句说明的起始地址一致。
4. 程序中的各个子程序的属性随段内或段间调用而定为 NEAR 或 FAR。
5. 连接生成 EXE 文件的各个不同的目标模块内的代码段，数据段或附加段可取同名或独立命名。但要求只有主模块的 END 语句指出程序入口的起始标号，并至少有一个具有 STACK 属性的堆栈段

## 终端、Shell、tty 和控制台 (console)

- Console: 在早期的电脑上，往往具有带有大量开关和指示灯的面板，可以对电脑进行一些底层的操作，这个面板就叫做 Console。其概念来自于管风琴的控制台。一台电脑通常只能有一个 Console，很多时候是电脑主机的一部分，和 CPU 共享一个机柜。
- Terminal: 一台大型主机往往需要支持许多用户同时使用，每个用户所使用操作的设备，就叫做 Terminal——终端，终端使用通信电缆与电脑主机连接，甚至可以通过电信网络（电话、电报线路等等）连接另一个城市的电脑。
- TTY: 电传打字机 Teletypewriter 的缩写，在上图中的那种带显示屏的视频终端出现之前，TTY 是最流行的终端设备。
- Shell 不是硬件，而是软件，是操作系统的操作界面，Windows 3.x 可以看做是 DOS 的 Shell，<http://command.com> 也是 DOS 的 shell。

terminal (终端) ——指电线的末端，shell——指乌龟的壳，而 tty——是一个奇怪的缩写。然后是 console——一种机柜。嗯，这只是词源上的意思。在 UNIX 的术语中，最简单的回答是：终端 (terminal) =tty= 文本的输入输出环境控制台 (console) =物理终端 shell=命令行解释器

The **shell** is the program which actually processes commands and returns output. Most shells also manage foreground and background processes, command history and command line editing. These features (and many more) are standard in `bash`, the most common shell in modern linux systems.

A **terminal** refers to a wrapper program which runs a shell. Decades ago, this was a physical device consisting of little more than a monitor and keyboard. As unix/linux systems added better multiprocessing and windowing systems, this terminal concept was abstracted into software. Now you have programs such as **Gnome Terminal** which launches a window in a Gnome windowing environment which will run a *shell* into which you can enter commands.

The **console** is a special sort of *terminal*. Historically, the console was a single keyboard and monitor plugged into a dedicated serial console port on a computer used for direct communication at a low level with the operating system. Modern linux systems provide *virtual consoles*. These are accessed through key combinations (e.g. `Alt + F1` or `Ctrl + Alt + F1`; the **function key** numbers different consoles) which are handled at low levels of the linux operating system -- this means that there is no special service which needs to be installed and configured to run. Interacting with the console is also done using a *shell* program.

## .ini

.ini 文件是 Initialization File 的缩写，即初始化文件 [1]，是 windows 的系统配置文件所采用的存储格式，统管 windows 的各项配置，一般用户就用 windows 提供的各项图形化管理界面就可实现相同的配置了。但在某些情况，还是要直接编辑.ini 才方便，一般只有很熟悉 windows 才能去直接编辑。开始时用于 WIN3X 下面，WIN95 用注册表代替，以及后面的内容表示一个节，相当于注册表中的键。

除了 windows2003 很多其他操作系统下面的应用软件也有.ini 文件，用来配置应用软件以实现不同用户的要求。一般不用直接编辑这些.ini 文件，应用程序的图形界面即可操作以实现相同的功能。它可以用来存放软件信息,注册表信息等。

## Smalltalk 语言

Smalltalk，被公认为历史上第二个面向对象的程序设计语言，和第一个真正的集成开发环境 (IDE)。Smalltalk 由艾伦·凯，Dan Ingalls，Ted Kaehler，Adele Goldberg 等于 70 年代初在 Xerox PARC 开发。

Smalltalk 对其它众多的程序设计语言的产生起到了极大的推动作用，主要有：C++，C#，Objective-C，Actor，Java 和 Ruby 等。90 年代的许多软件开发思想得利于 Smalltalk，例如设计模式、敏捷编程和代码重构等。

## Is-a、has-a、Like-a 区别

- Is-a:  
是 a: A Is B: A 是 B (继承关系，继承)。
- has-a:  
有 a: A has B: A 有 B (从属关系，聚合)。
- like-a:  
像 a: A like B: A 像 B (组合关系，接口)。
- is-like-a
- use-a

如果继承过程中，仅仅是覆盖了父类中的方法，则为 is-a 关系。

如果有新增的方法，则为 is-like-a 关系。

## 灰度测试

灰度测试，就是在某项产品或应用正式发布前，选择特定人群试用，逐步扩大其试用者数量，以便及时发现和纠正其中的问题

灰度发布，又名金丝雀发布，或者灰度测试

是指在黑与白之间能够平滑过渡的一种发布方式。在其上可以进行 A/B testing，即让一部分用户继续用产品特性 A，一部分用户开始用产品特性 B，如果用户对 B 没有什么反对意见，那么逐步扩大范围，把所有用户都迁移到 B 上面来。

灰度发布是对某一产品的发布逐步扩大使用群体范围，也叫灰度放量。灰度发布可以保证整体系统的稳定，在初始灰度的时候就可以发现、调整问题，以保证其影响度。

灰度期：灰度发布开始到结束期间的这一段时间，称为灰度期。

## 管态和目态

1. 管态又叫特权态，系统态或核心态。CPU 在管态下可以执行指令系统的全集。通常，**操作系统在管态下运行**。
2. 目态又叫常态或用户态。机器处于目态时，程序只能执行非特权指令。不能直接使用系统资源，也不能改变 CPU 的工作状态，并且只能访问这个用户程序自己的存储空间。

## 按字节寻址

一般计算机是按字节寻址的

即对于地址线，产生的地址中，每一个对应于一个字节的内存空间

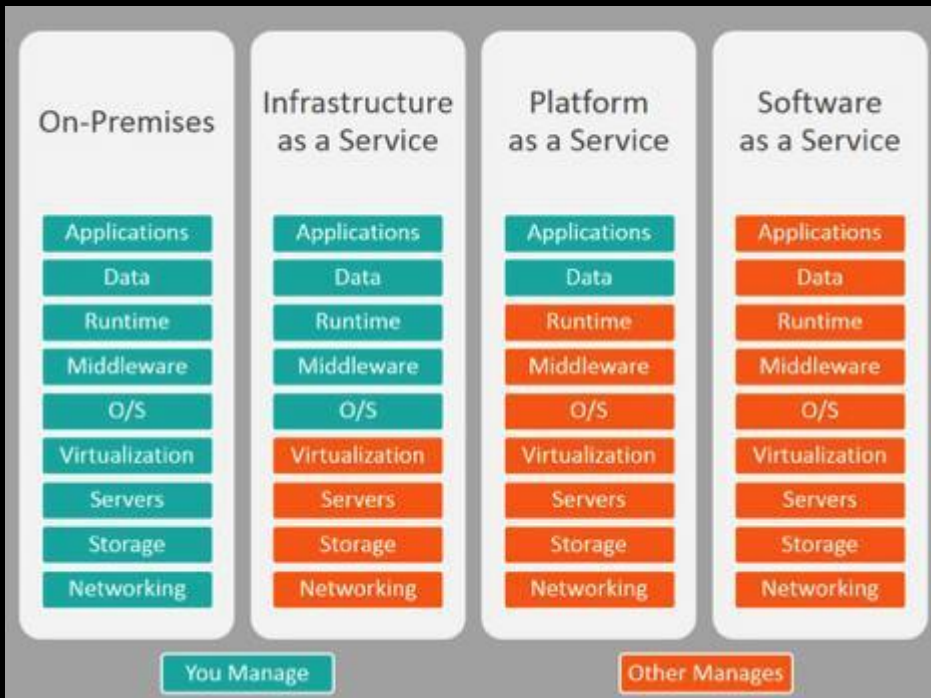
Eg: 12bit 的地址线，那么产生  $2^{12}$  字节

## IaaS vs PaaS vs SaaS

基础设施即服务(IaaS)

平台即服务(PaaS)

软件即服务(SaaS)。



## r、r+、w、w+、a、a+

**r** 打开只读文件，该文件必须存在。

**r+** 打开可读写的文件，该文件必须存在。

**w** 打开只写文件，若文件存在则文件长度清为 0，即该文件内容会消失。若文件不存在则建立该文件。

**w+** 打开可读写文件，若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。

**a** 以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。

**a+** 以附加方式打开可读写的文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾后，即文件原先的内容会被保留。

**wb** 在 windows 下，以二进制进行存储，\r\n 才是换行

**w** 是以文本方式进行存储\n 是换行

**rb** 取出来的也是\r\n

**r** 取出来的是\n

## 测试

### 1. α测试

就是把用户请到公司内部进行测试使用。

$\alpha$ 测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试；

目的：是评价软件产品的 **FLURPS(即功能、局域化、可使用性、可靠性、性能和支持)**。

注意！ $\alpha$ 测试不能由程序员或测试员完成。

## 2. $\beta$ 测试

**用户在不同场所进行测试。**

$\beta$ 测试是一种验收测试。 $\beta$ 测试由软件的终用户们在一个或多个场所进行。

## 3. 区别

1. 它们都是验收测试！
2.  $\alpha$ 测试是指把用户请到开发方的场所来测试
3.  $\beta$ 测试是指在一个或多个用户的场所进行的测试。
4.  $\alpha$ 测试的环境是受开发方控制的,用户的数量相对比较少,时间比较集中。
5.  $\beta$ 测试的环境是不受开发方控制的,用户数量相对比较多,时间不集中。
6.  $\alpha$ 测试先于 $\beta$ 测试执行。通用的软件产品需要较大规模的 $\beta$ 测试,测试周期比较长

注：验收测试包括三类：

1. 正式验收测试
2. 非正式验收测试 -  $\alpha$ 测试（内测） -  $\beta$ 测试（公测）

**封测：**封闭测试。其版本实为未成熟的，有很多的 BUG。就是禁止用户注册，只提供了一些账号分给玩家试玩,如果发现 BUG 了就一定要告诉官方网站,官方才能进行补丁。【给少部分玩家玩,返回 BUG】

**内测：**内部测试。经历了封测后，游戏进一步完善。【给大部分玩家玩，账号难注册，返回 BUG】

**公测：**公开测试。其实就是向广大玩家完全公开，注册的账号数量没有限制，到了公测阶段一般来讲初期是免费的，之后随着玩家数量的多少，游戏运营商会在一时间之后对游戏开始收费。【给全部玩家免费玩，到一定时间会收费】

# 甘特图

**甘特图 (Gantt chart)** 又称为横道图、条状图(Bar chart)。其通过条状图来显示项目，进度，和其他时间相关的系统进展的内在关系随着时间进展的情况。以提出者亨利·劳伦斯·甘特 (Henry Laurence Gantt) 先生的名字命名。

亨利·劳伦斯·甘特是泰勒创立和推广科学管理制度的亲密的合作者，也是科学管理运动的先驱者之一。甘特非常重视工业中人的因素，因此他也是人际关系理论的先驱者之一。其对科学管理理论的重要贡献：

- 1、提出了任务和奖金制度。
- 2、强调对工人进行教育的重要性，重视人的因素在科学管理中的作用。——其在科学管理运动先驱中最早注意到人的因素、“工业的习惯”。
- 3、制定了甘特图——生产计划进度图（是当时管理思想的一次革命）。

他在 20 世纪早期引用了这种工作和方法。在图上，项目的每一步在被执行的时间段中用线条标出。完成以后，甘特图能以时间顺序显示所要进行的活动，以及那些可以在同时进行的活动。

个人甘特图和时间表是两种不同的任务表达方式，个人甘特图使用户可以直观地知道有哪些任务在什么时间段要做，而时间表则提供更精确的时间段数据。此外，用户还可以在时间表中直接更新任务进程。

甘特图以图示通过活动列表和时间刻度表示出特定项目的顺序与持续时间。一条线条图，横轴表示时间，纵轴表示项目，线条表示期间计划和实际完成情况。直观表明计划何时进行，进展与要求的对比。便于管理者弄清项目的剩余任务，评估工作进度。

甘特图是以作业排序为目的，将活动与时间联系起来的 earliest 尝试的工具之一，帮助企业描述工作中心、超时工作等资源的使用。

甘特图包含以下三个含义：

- 1、以图形或表格的形式显示活动；
- 2、通用的显示进度的方法；
- 3、构造时含日历天和持续时间，不将周末节假算在进度内。

简单、醒目、便于编制，在管理中广泛应用。

甘特图按内容不同，分为计划图表、负荷图表、机器闲置图表、人员闲置图表和进度表五种形式。

甘特图的特点是突出了生产管理中最重要因素——时间，它的作用表现在三个方面：

- 1、计划产量与计划时间的对应关系。
- 2、每日的实际产量与预定计划产量的对比关系。
- 3、一定时间内实际累计产量与同时期计划累计产量的对比关系



## Coordinated Universal Time

协调世界时，又称世界统一时间、世界标准时间、国际协调时间。由于英文（CUT）和法文（TUC）的缩写不同，作为妥协，简称UTC。

协调世界时是以原子时秒长为基础，在时刻上尽量接近于世界时的一种时间计量系统。

国际原子时的准确度为每日数纳秒，而世界时的准确度为每日数毫秒。许多应用部门要求时间系统接近世界时UT，对于这种情况，一种称为协调世界时的折中时标于1972年面世。为确保协调世界时与世界时相差不会超过0.9秒，在有需要的情况下会在协调世界时内加上正或负闰秒。因此协调世界时与国际原子时之间会出现若干整数秒的差别，两者之差逐年积累，便采用跳秒（闰秒）的方法使协调时与世界时的时刻相接近，其差不超过1s。它既保持时间尺度的均匀性，又能近似地反映地球自转的变化。 [1] 按国际无线电咨询委员会（CCIR）通过的关于UTC的修正案，从1972年1月1日起UTC与UT1（在UT中加入极移改正得到）之间的差值最大可以达到±0.9s。位于巴黎的国际地球自转事务中央局负责决定何时加入闰秒。一般会在每年的6月30日、12月31日的最后一秒进行调整。

和格林威治时差不多，是在其基础在做调整得到的。

## 域名映射

域名与ip之间映射

Windows 系统下配置

在目录C:\Windows\System32\drivers\etc\hosts 添加映射

```
192.168.20.15 baidu.com
```

## 全角和半角

全角和半角输入法的区别

全角和半角的区别主要在于除汉字以外的其它字符，比如标点符号、英文字母、阿拉伯数字等，全角字符和半角字符所占用的位置的大小不同。

在计算机屏幕上，一个汉字要占两个英文字符的位置，人们把一个英文字符所占的位置称为“半角”，相对地把一个汉字所占的位置称为“全角”。

标点符号、英文字母、阿拉伯数字等这些字符不同于汉字，在半角状态它们被作为英文字符处理，而

