Computing B written exam notes

计算机的信息表示

计算机的发展历史

历史人物

阿兰·图灵 A.M Turing　冯·诺依曼 John Von Neumann　史提芬·库克 Stephen A.Cook

> 计算概论，许卓群等，第2版，清华大学出版社，142页。
>
> 142
>
> **6.1.3　计算机科学理论的发展里程碑**
>
> (1) 图灵(Alan M. Turing)提出了图灵机，分析了图灵机所能解决的计算问题类。并且证明，对于图灵机而言，存在着不可判定的问题类。他证明：对于“停机问题”和“一阶谓词的判定问题”就是不存在通用算法求解的两类问题。
>
> (2) 冯·诺伊曼(J. Von Neumann)和他的同事们，给出了现代电子数字计算机的设计蓝图，明确提出了现代内储程序控制的设计方案。对指令、指令周期、指令系统都给出了明确的方案，从而奠定了数字计算机体系结构的基础。
>
> (3) 库克(Stephen A. Cook)研究了计算复杂性。这一概念的重要性逐渐被人们所认识。虽然对于多项式复杂的计算问题，计算机能够有效地解决它们，求出精确答案。但是从理论上说，多项式复杂的计算问题仅仅是全部可以计算问题中的一小部分，比这类问题更复杂难解的问题非常多。理论上已经证明，存在着很多具体的计算问题类，它们在日常计算中也会遇到，虽然也能编制出求解这些问题的程序，但是一旦在计算机中具体计算它们，当问题的规模增大时，计算机就显出无能为力了。很多难解问题类的大型问题，连最快的计算机用几百年也不能计算出结果。对这些难解问题类，目前的计算机表现出能力局限，无法求出问题的精确答案，而只能求助于求近似解。

图灵机

图灵机由两部分构成，如图1-13所示。

- 一条**存储带**（tape）：双向无限延长，上有一个个方格（field），每个方格可以包含一个有限字母的字符。在一个真正的机器中，磁带必须足够大，以包含算法的所有数据。
- 一个**控制器**：包含一个可以双向移动的**读写头**（head），可以在所处方格中读写一个字符；图灵机每时每刻都处于某种**状态**（current state），是有限数量的状态中的一种；可以接受设定好的**图灵程序**（program），该程序是一个转换列表，它决定了一个给定的 State 和 head 下字符的新状态，一个必须写入 head 下方格的字符和 head 的运动方向，即左、右或静止不动。

> 但是 2024 fall 笔试题目说是三个部分？需要灵活处理一下。

现代计算机

根据冯·诺伊曼结构，计算机由运算器，控制器，存储器，输入设备和输出设备五个部分相互连接组成。

> 5个部件的作用：
>
> 1. 运算器（Arithmetic and Logic Unit, ALU）
>   - 负责执行计算和逻辑运算，比如加法、减法、逻辑与、或等操作。
> 2. 控制器（Control Unit, CU）
>   - 负责解释和执行指令，协调计算机其他部件的工作。它从内存中读取指令并将其转换为具体操作。
> 3. 存储器（Memory）

> - 用于存储程序、数据以及中间结果，通常分为主存储器（如RAM）和辅助存储器（如硬盘）。
> 4. 输入设备（Input Devices）
>   - 用于向计算机输入数据和指令，例如键盘、鼠标、扫描仪等。
> 5. 输出设备（Output Devices）
>   - 用于将计算结果和信息输出给用户，例如显示器、打印机、扬声器等。
>
> 这五大部件通过**总线（Bus）**相互连接，构成一个完整的计算机系统。冯·诺伊曼结构的核心思想是存储程序，即将指令和数据以相同的形式存储在内存中，并依次执行。

代际变化

电子管计算机→**晶体管**计算机→**集成电路**计算机→**大规模集成电路**计算机→**第五代**计算机

摩尔定律

- 每 18 个月，集成电路芯片上集成的晶体管数将翻一番
- 每 18 个月，集成电路芯片的速度将提高一倍
- 每 18 个月，集成电路芯片的价格将降低一半

二进制运算

数值运算与逻辑运算

- 与、或、非跟常识相同
- 异或：两个值相同则异或结果为1，反之则为0

归纳如下表：

逻辑运算

类型：非（NOT)、与（AND）或（OR）和异或（XOR）。

非（NOT）

x	NOT x
0	1
1	0

与（AND）

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

或（OR）

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

异或（XOR）

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

> 对于不是0和1的情形：(X)表示整数
> NOT: not 0 ->True ； not X -> False
>
> AND: 0 and X ->0；X and Y->Y
>
> OR: 0 or X ->X; X or Y->X

进制转化

可以从任意底转换到十进制。将数码乘以其在源系统中的位置量并求和便得到在十进制中的数。

能够将十进制数转换到与其等值的任意底。需要两个过程：一是用于整数部分，另一个是用于小数部分。整数部分需要连除，而小数部分需要连乘。

将数字从二进制转换到十六进制很容易，反之亦然。这是因为二进制中的4位恰好是十六进制中的1位。

将数字从二进制转换到八进制很容易，反之亦然。这是因为二进制中的3位恰好是八进制中的1位。

例 将十进制数0.625转换为二进制数。

解 因为0.625没有整数部分，该例子显示小数部分如何计算。这里是以2为底。在左边一角写上这个十进制数。连续乘2，并记录结果的整数和小数部分。小数部分移到右边，整数部分写在每次运算的下面。当小数部分为0，或达到足够的位数时结束。结果是0.625 = $(0.101)_2$。

```
十进制    0.625 → 0.25 → 0.50 → 0.00
                  ↓      ↓      ↓
二进制  .  1      0      1
```

信息编码

数值与文字

ASCII码

```
ord('0') = 48
ord('a') = 97
ord('A') = 65
```

信息存储单位

- **位，比特（bit）**：是最小的数据单位，表示一位二进制数字，可以是0或1。
- **字节 （Byte）**：由8位2进制数位构成，数据存储中最常用的基本单位。一个字节有256个值，可存放一个半角英文字符（ ASCII码）。

  > 关于汉字编码：UTF-8：一般使用3个字符，生僻字会用4个；GBK一般使用2个。

  1 Byte = 8 bit
- **千字节（KB, Kibibyte, KiB）**：是比字节更大的数据单位。
  - 1 KB (Kilobyte) = 10^3 Bytes（十进制）
  - 1 KiB (Kibibyte) = 1024 Bytes（二进制）
- **兆字节（MB, Mebibyte, MiB）**：
  - 1 MB (Megabyte) = 10^6 Bytes = 1,000,000 Bytes（十进制）
  - 1 MiB (Mebibyte) = 1024^2 Bytes = 1,048,576 Bytes（二进制）
- **吉字节（GB, Gibibyte, GiB）**：
  - 1 GB (Gigabyte) = 10^9 Bytes = 1,000,000,000 Bytes（十进制）
  - 1 GiB (Gibibyte) = 1024^3 Bytes = 1,073,741,824 Bytes（二进制）
- **太字节（TB, Tebibyte, TiB）**：
  - 1 TB (Terabyte) = 10^{12} Bytes = 1,000,000,000,000 Bytes（十进制）
  - 1 TiB (Tebibyte) = 1024^4 Bytes = 1,099,511,627,776 Bytes（二进制）
- **拍字节（PB, Pebibyte, PiB）**：
  - 1 PB (Petabyte) = 10^{15} Bytes = 1,000,000,000,000,000 Bytes（十进制）
  - 1 PiB (Pebibyte) = 1024^5 Bytes = 1,125,899,906,842,624 Bytes（二进制）

> $2^{30} \approx 1G = 10^9$
>
> $2^{32} \approx 40$亿

原码、反码与补码

> 注意区分有无符号整数，数据范围不同但大小相同，有符号整数包括符号位。

原码、反码和补码是计算机科学中用来表示整数（包括正数和负数）的三种编码方式，特别是在二进制系统中。它们主要用于简化算术运算，尤其是减法操作。下面我将分别介绍这三种编码，并给出一个例子来说明它们之间的关系。

原码 (Sign-Magnitude Representation)

原码是最简单的表示方法，它直接使用最高位表示符号（0为正，1为负），其余位表示数值的绝对值。例如：

- 8位二进制数 `0000 0101` 表示 +5
- 8位二进制数 `1000 0101` 表示 -5

反码 (One's Complement)

反码用于表示负数时，对原码除了符号位之外的所有位取反（即将0变为1，1变为0）。对于正数，反码与原码相同。继续上面的例子：

- +5 的反码仍然是 `0000 0101`
- -5 的反码则是 `1111 1010`（原码 `1000 0101` 中除符号位外所有位取反）

补码 (Two's Complement)

补码是对**反码**加1得到的结果。补码表示法在计算机中广泛使用，因为它可以方便地进行加法和减法运算，而且只有一个零的表示形式。同样以+5和-5为例：

- +5 的补码还是 `0000 0101`（与原码和反码相同）
- -5 的补码是 `1111 1011`（反码 `1111 1010` 加1）

补码的一个重要特性是它可以简化硬件设计，因为两个数相加无论正负都可以按照同样的规则处理，而不需要额外的电路来区分加法和减法。此外，在补码系统中，-128到+127之间的所有整数都可以用8位二进制数表示，而不会有重复的零表示。

> 对于负数做如下操作：
>
> 该运算分为两步：首先，对于绝对值从右边复制位，直到有1被复制；接着，反转其余的位（不包含1）。

声音

在时间和波形高度两个维度进行离散化采样,每秒钟对音频采用的次数以Hz为单位。

> **5. 有一种影像，人们对它的要求很高：影像的播放速度是每秒32帧图像，每帧图像的分辨率为2048×1536像素，其颜色系统是512色；而声音的采样频率则要达到65536Hz，采用双声道，每声道用4字节（Byte）存储采样值。请问，要保存10分钟这种原始影像，需要多大的存储空间？（5分）**
>
> 每秒钟影像：
>
> - 颜色编码：
> $2^5 * 2^{11} * 3 * 2^9 * 9/8\mathrm{B} = 4 * 3 * 9\ \mathrm{MB} = 108\ \mathrm{MB}$
> - 每秒钟声音：
>
> $65536 * 2 * 4B = 0.5\ \mathrm{MB}$
>
> 10分钟：
>
> $$10 * 60 * (108 + 0.5)\ \mathrm{MB} = 65100\ \mathrm{MB} \approx 63.6\ \mathrm{GB}$$
>
> 列出正确的计算式子，即可给满分，其余视具体情况酌情给分，不超过3分。

图像

- 位图：由像素点构成，放大失真。宽×高的像素数构成（1920*1080）

一个像素需要一个字节（8bit）

- 矢量图：放大不失真，不由像素点构成

> 采用不同分辨率、不同颜色编码的图像，其图像质量差别非常大。对于同样一幅原始图像，分辨率越高，则图像越精细，质量越好，当然所需要的存储空间也是非常大的。一幅分辨率为4096*2048的真彩色(24位，3Bytes)图像，如果不做压缩，其所需的存储空间为？(结果单位采用MB) (2分)
>
> 4096 x 2048 x 24 / (8 x 1024 x 1024) = 24MB

> 假设视频的帧率为30FPS（即每秒包含30帧图像）。对于8K的电影而言，每帧图像包含7680 x 4320个像素，其中每个像素包含RGB三种颜色，每种颜色采用1个字节进行记录。（计算中7680 x 4320可近似为33,000,000，以下计算中1GB=1000MB，其余进制同理。请写出计算过程）(4分，每小题2分)
>
> （1）一块1000GB的移动硬盘能存储多少秒这样的8K视频？（保留整数即可）
>
> > 33,000,000 x 3 x 1 Byte x 30 = 2,970,000,000 字节/秒
> > 1000 x 1000 x 1000 / 2,970,000,000 ≈ 337 秒
>
> （2）在实际应用中，视频在存储与传输中会经过编码压缩。如果经过编码后，上述1000GB的移动硬盘能够存储56小时的8K视频，请问所采用的视频编码方法的压缩率为多少？（压缩率为x:1表示每x个字节的信息被压缩为1个字节，x保留到整十即可）
>
> > **计算压缩后的数据量**：
> >
> > - 压缩后能存储56小时的视频
> > - 56小时 = 56 x 3600 = 201,600 秒
> > - 压缩后的每秒数据量：1,000,000,000,000 / 201,600 ≈ 4,960,000 字节
> >
> > **计算压缩率**：
> >
> > - 原始每秒数据量：2,970,000,000 字节
> > - 压缩后每秒数据量：4,960,000 字节
> > - 压缩率：2,970,000,000 / 4,960,000 ≈ 600

计算机组成与信息表示处理

计算机硬件

CPU ： 运算器+控制器

微型计算机的 CPU 由寄存器，中断处理器，算术逻辑运算器和程序控制器四个部件组成。

> 四个部件的作用：
>
> - **算术逻辑运算器 (ALU)**：主要负责执行算术运算（加法、减法等）和逻辑运算（与、或、非等）。
> - **寄存器**：是一组小容量但非常快速的存储单元，用于临时存放数据、指令或地址，以加速CPU的操作。
> - **中断处理器**：处理紧急应急情况，如鼠标移动。

> - **程序控制器**：通常指程序计数器（PC），也叫做指令指针（IP），它持有下一条要执行指令的地址。程序控制器帮助CPU跟踪和控制指令的执行顺序。

CPU的性能指标

工作主频（时钟频率）、核心数量（核）、字长（位）、运算速度

> 时钟频率是指 CPU 每秒钟能够执行的时钟周期数，通常以 **GHz（吉赫兹）** 为单位。例如，3.2 GHz 的 CPU 每秒可以执行 32 亿个时钟周期。

> 核心数量是指 CPU 中包含的独立处理单元的数量。现代 CPU 通常有多个核心，如双核、四核、六核、八核甚至更多。

> **CPU 字长（Word Size）** 是指 CPU 在单个操作周期中能够处理的数据量，通常以 **位（bit）** 为单位。它决定了 CPU 一次可以处理的最大数据宽度，直接影响到 CPU 的寻址能力、数据处理能力和指令集设计。

> 为了计算CPU的运算速度（以每秒百万条指令，即MIPS为单位），我们可以使用以下公式：
>
> $$\text{MIPS} = \frac{\text{时钟频率}}{\text{每个指令周期的时钟周期数}}$$

主存储器（主存，内存）

主存是存储单元的集合。每一个单元有一个称为地址的标识符。数据被传输到内存或从内存取出是以称为字的二进制位组的方式。内存中唯一可标识的单元总数称为地址空间。有两种内存可用：随机存取存储器(RAM)和只读存储器(ROM)。

> 主存(内存)管理是以8个二进制位(一个Byte)作为一个管理单元的，并且每个单元都编有一个唯一的地址，内存的访问是通过其地址进行的。

> 随机存储器：每个地址的访问时间与地址无关（或访问每个地址的时间相同等意思）。

计算机存储体系(存储器硬件的金字塔结构)

寄存器--高速缓存--主存储器--外部存储设备--辅助存储设备

> 1. **寄存器（Registers）**
>
> - **位置**：位于 CPU 内部
> - 特点：
>   - **最快**的存储器，直接由 CPU 控制。
> - 用于存储当前正在执行的指令和操作数。
> - 容量非常小，通常只有几十个字节。
> - 每个寄存器都有特定的功能，例如累加器、程序计数器、状态寄存器等。
> - 作用：
>   - 存储 CPU 正在处理的数据或指令，减少访问内存的次数。
> - 提高指令执行的速度，因为寄存器的访问时间几乎为零。
>
> 2. **高速缓存（Cache Memory）**
>
> - **位置**：位于 CPU 内部（L1, L2 缓存）或靠近 CPU（L3 缓存）
> - 特点：
>   - **次快**的存储器，比寄存器稍慢，但比主存储器快得多。
> - 分为多个级别（L1, L2, L3），越靠近 CPU 的缓存级别越快，但容量越小。

    - **L1 缓存**：最小且最快，通常为几 KB 到几十 KB。
    - **L2 缓存**：较大，通常为几百 KB 到几 MB。
    - **L3 缓存**：最大，通常为几 MB 到几十 MB，可能被多个核心共享。
    - 使用局部性原理（时间局部性和空间局部性）来预测未来可能需要的数据，并提前将其加载到缓存中。
- 作用：
    - 减少 CPU 访问主存储器的次数，提高数据访问速度。
- 缓存命中时，CPU 可以直接从缓存中读取数据，避免了访问较慢的主存储器。

3. **主存储器（Main Memory / RAM）**

- **位置**：位于 CPU 和外部存储设备之间
- 特点：
    - **易失性存储器**，断电后数据会丢失。
- **程序储存在内存中。**
- 容量较大，通常为几 GB 到几十 GB。
    - 访问速度比缓存慢，但比外部存储设备快得多。
    - 主要分为两种类型：
        - **DRAM（动态随机存取存储器）**：最常见的主存储器类型，依赖电容器存储数据，需要定期刷新。
        - **SRAM（静态随机存取存储器）**：速度更快，但成本较高，通常用于缓存。
- 作用：
- 存储当前正在运行的程序和数据。
- 作为 CPU 和外部存储设备之间的桥梁，提供快速的数据交换。

4. **外部存储设备（Secondary Storage）**

- **位置**：位于计算机外部或内部（如硬盘、固态硬盘、光盘等）
- 特点：
- **非易失性存储器**，断电后数据不会丢失。
- 容量非常大，通常为几百 GB 到几 TB。
    - 访问速度较慢，远低于主存储器和缓存。
    - 常见的外部存储设备包括：
        - **HDD（硬盘驱动器）**：基于磁盘旋转的技术，访问速度较慢，但容量大、成本低。
        - **SSD（固态硬盘）**：基于 NAND 闪存技术，访问速度比 HDD 快得多，但成本较高。
        - **光盘（CD/DVD/Blu-ray）**：用于存储数据或分发软件，读写速度较慢。
        - **USB 闪存驱动器**：便携式存储设备，容量从小到大不等，速度介于 HDD 和 SSD 之间。
- 作用：
- 存储长期保存的数据和程序，如操作系统、应用程序、用户文件等。
- 作为备份存储，防止数据丢失。
    - 用于扩展主存储器的容量，通过虚拟内存技术将部分磁盘空间用作扩展内存。

> 5. **辅助存储设备（Tertiary Storage）**
>
> - **位置**：通常位于外部，如磁带库、云存储等
> - 特点：
> - **非易失性存储器**，断电后数据不会丢失。
> - 容量极大，通常为几十 TB 到 PB 级别。
>   - 访问速度非常慢，主要用于归档和备份。
>   - 常见的辅助存储设备包括：
>   - **磁带库**：用于大规模数据归档，访问速度较慢，但成本低廉。
>   - **云存储**：通过网络访问的远程存储服务，容量几乎无限，但访问速度取决于网络带宽。
> - 作用：
> - 存储不经常访问的历史数据、备份数据或归档数据。
> - 作为灾难恢复的手段，确保数据的安全性和持久性。
>
> 存储层次结构的工作原理
>
> 计算机存储体系的层次结构遵循“局部性原理”，即程序倾向于重复访问相同的数据或相邻的数据。为了充分利用这一特性，存储层次结构中的每一层都试图在速度、容量和成本之间找到最佳平衡：
>
> - **寄存器**：存储最频繁访问的数据，访问速度最快，但容量最小。
> - **高速缓存**：存储最近访问过的数据和指令，利用时间局部性和空间局部性，减少对主存储器的访问。
> - **主存储器**：存储当前正在运行的程序和数据，提供较快的访问速度，但容量有限。
> - **外部存储设备**：存储长期保存的数据和程序，容量大，但访问速度较慢。
> - **辅助存储设备**：用于归档和备份，容量极大，但访问速度非常慢。

计算机系统由硬件与软件构成，软件系统中最重要的软件是**操作系统**， 主要负责资源管理、任务调度等。

> 主板（硬件）是计算机主机箱中的主要部件，计算机的其他硬件设备通过各种接口与主板相连并发挥作用。

输入（输出）子系统

输入/输出(I/O)子系统的设备集合允许计算机与外界交流，存储程序和数据，即使在计算机已关机时也可以。输入/输出设备分成两大类;非存储设备和存储设备。非存储设备允许CPU/内存与外界通信;存储设备可以存储以后被检索的大量信息。存储设备被分成磁的和光的。

有两种方法输入/输出设备的寻址：I/O独立寻址和I/O存储器映射寻址。在I/O独立寻址方法中用来从内存读/写的指令完全不同于用来从输入/输出设备的读1写指令。在I/O存储器映射寻址方法中，CPU把I/O控制器中的每个寄存器看成是内存中的字。

如今，通用计算机使用称为程序的一组指令来处理数据。计算机执行程序，从输入数据创建输出数据。程序和数据都存储在内存中。CPU使用重复的机器周期一条接一条，从头到尾执行程序中的指令。简化的周期由三阶段组成:**取指令、译码和执行。**

有三种使CPU和输入/输出设备同步的方法：程序控制输入/输出、中断控制输入/输出和直接存储器存取(DMA)。

总线：三个子系统之间的关系

计算机中三个子系统的连接起重要的作用，因为在这些子系统间需要进行信息的通信。CPU和内存通常被三个连接连在一起(每个称为总线)：**数据总线、地址总线和控制总线**。（内部总线）输入/输出设备通过输入/输出控制器或接口与总线相连，使用的控制器有多种，如今最常见的有：SCSI、火线和USB。（外部总线）

- 数据总线：用于传递数据
- 地址总线：用于传递主存储器地址
- 控制总线：用于各种控制信息的传递，如读、写等

⋆ 为了保证性能，**数据总线**的宽度应该与 **CPU 字长**一致。

⋆ **CPU地址总线宽度**决定了**主存储器地址空间的大小。**

> 数据总线的宽度通常与CPU的字长（Word Size）一致是为了保证性能和效率。这里有几个关键点：
>
> 1. **CPU 字长**：指的是CPU一次能够处理的数据量大小，它决定了CPU内部寄存器、算术逻辑单元（ALU）等组件的操作数宽度。比如32位CPU的一次操作可以处理32位的数据，而64位CPU则可以处理64位的数据。
> 2. **数据总线宽度**：是指数据总线一次能传输的数据量，它直接影响到CPU与内存或其他外部设备之间数据交换的速度。如果数据总线的宽度与CPU字长相同，那么在进行数据读写时，就不需要分多次传输，从而提高了数据传输的效率。
> 3. **地址总线宽度**：决定了CPU可以直接寻址的地址空间大小。例如，32位地址总线可以访问2^{32}个不同的地址，即4GB的物理地址空间；而64位地址总线理论上可以访问2^{64}个地址，这远远超过了目前大多数系统的实际需求。（一个地址对应一个字节）
> 4. **控制总线**：它不直接与性能或数据量相关，而是用于传递控制信号，如读/写命令、中断信号等，以协调CPU与其他系统组件之间的操作。
>
>   为了最大化性能，理想情况下，数据总线的宽度应该匹配CPU的字长，这样每次内存访问都可以最有效地利用CPU的能力。不过，在实际设计中，也会考虑到成本、功耗等因素来决定总线宽度。有时，为了降低成本或出于其他考虑，可能会选择较窄的数据总线，但这通常会导致性能上的妥协。

操作系统与计算机网络

软件系统中最重要的软件是操作系统，主要负责资源管理、任务调度等。

互联网通信协议

TCP/IP通讯协议栈

1. 物理层通信协议
2. *数据链路协议*
3. 网际互连层（网络层）通讯协议：负责路由选择和数据包的传输。·协议:IP、ICMP、ARP等:
4. 传输层通讯协议：负责端到端的通信和可靠性保障。协议:TCP、UDP等。
5. 应用层：提供用户应用服务，协议:

   HTTP（超文本传输协议，网页浏览）、FTP（远程文件传输）、SMTP/POP3（电子邮件收发）等。

计算机网络协议（互联网通信协议）

- POP3/SMTP 收发邮件
- UDP 无连接的传输层协议，适用于对实时性要求较高的应用
- VoIP 语音，电话通话
- TCP/IP 是互联网的基础协议套件，确保数据能够在全球范围内可靠传输
- HTTP(S)（安全， secure）超文本传输协议

通讯协议栈（Protocol Stack） 是一组分层的通信协议，用于定义和管理数据在网络中的传输过程。每个层次负责特定的功能，确保数据能够从发送方正确、可靠地传输到接收方。最著名的通讯协议栈是 **OSI 模型** 和 **TCP/IP 模型**，它们为网络通信提供了一个标准化的框架。

> 通信协议栈保证了互联网上的电脑间能够正确的交流，它通常由定义在应用层、传输层、网络层、数据链路层以及物理层上的一系列协议构成。

> IPv4的地址空间有限（只有32位），随着互联网设备的快速增长，IPv4地址逐渐耗尽。IPv6 作为代替IPv4的网络协议，由128位二进制数构成，为了便于阅读和书写，IPv6地址通常每 **16位** 写成一组16进制数，中间用冒号（:）分隔。
>
> 域名：方便人类记忆与使用的。
>
> > ## 域名的结构
> >
> > 域名通常由多个部分组成，每个部分之间用点号（`.`）分隔。从右到左，域名的各个部分依次表示不同的层次：
> >
> > - **顶级域名（TLD, Top-Level Domain）**：位于域名的最右边，表示域名的最高级别分类。常见的顶级域名包括：
> >   - **通用顶级域名（gTLD）**：如 `.com`（商业机构）、`.org`（非营利组织）、`.net`（网络服务提供商）等。
> >   - **国家代码顶级域名（ccTLD）**：如 `.cn`（中国）、`.jp`（日本）、`.uk`（英国）等，用于标识特定国家或地区的域名。
> >   - **新通用顶级域名（new gTLD）**：如 `.blog`、`.shop`、`.xyz` 等，近年来新增的顶级域名。
> > - **二级域名（Second-Level Domain）**：位于顶级域名的左边，通常是用户注册的部分。例如，在 `www.example.com` 中，`example` 就是二级域名。
> > - **子域名（Subdomain）**：位于二级域名的左边，用于进一步细分域名空间。例如，在 `www.example.com` 中，`www` 是一个常见的子域名，表示“World Wide Web”。其他子域名可以是 `mail`、`blog`、`api` 等，用于标识不同的服务或功能。
>
> 在访问网页时，DNS(域名服务器)将域名转化成IP地址。
>
> > ## DNS的工作原理
> >
> > 1. **域名解析请求**：
> >   - 当用户在浏览器中输入一个域名（如 `www.example.com`）时，浏览器会向本地的DNS解析器（通常是用户的ISP提供的DNS服务器）发送一个查询请求，要求解析该域名对应的IP地址。
> > 2. **递归查询**：

> - 如果本地DNS解析器没有缓存该域名的IP地址，它会向上级DNS服务器发起递归查询，逐步查找负责该域名的权威DNS服务器。
> - 查询过程可能涉及多个级别的DNS服务器，包括根域名服务器、顶级域名服务器和二级域名服务器。
>
> 3. **权威DNS服务器**：
>
> - 最终，查询会到达负责该域名的权威DNS服务器。权威DNS服务器存储了该域名的准确信息，包括其对应的IP地址。
> - 权威DNS服务器会将域名的IP地址返回给本地DNS解析器。
>
> 4. **缓存与响应**：
>
> - 本地DNS解析器收到IP地址后，会将其缓存一段时间（称为TTL，Time to Live），以便后续相同的查询可以直接使用缓存结果，而不需要再次发起完整的查询过程。
> - 最后，浏览器收到IP地址后，就可以通过该IP地址与目标服务器建立连接并加载网页。

程序访问数据的局部性原理

局部性原理

局部性（Locality）是指程序倾向于重复访问相同的或在逻辑上相关的数据项和指令。局部性可以分为两大类：时间局部性和空间局部性。

1. 时间局部性（Temporal Locality）

- **定义**：如果一个数据项或指令被访问了一次，那么在不久的将来，它很可能再次被访问。
- 原因：
  - **循环**：程序中的循环会多次访问同一组数据或指令。
  - **函数调用**：函数调用会导致返回地址和参数的重复访问。
  - **用户行为**：用户界面操作（如点击按钮、滚动页面）可能会导致相同的数据或代码路径被频繁访问。
- **影响**：时间局部性使得缓存机制非常有效，因为最近访问过的数据可以保留在高速缓存中，减少访问主存的次数。

2. 空间局部性（Spatial Locality）

- **定义**：如果一个数据项或指令被访问了一次，那么其附近的其他数据项或指令也很可能很快被访问。
- 原因：
  - **数组和结构体**：程序通常按顺序访问数组元素或结构体成员。
  - **连续内存分配**：动态分配的内存块通常是连续的，程序会依次访问这些内存区域。
  - **指令流**：程序通常按顺序执行指令，除非遇到分支或跳转。
- **影响**：空间局部性使得预取（Prefetching）技术非常有用，系统可以在访问某个数据项时，提前将附近的数据加载到缓存中，以提高访问速度。

> 如果一个数据项正在被访问，那么在近期它很可能还会被再次访问，这说的是程序执行时，CPU访问数据的（时间局部性（或局部性））。

> 根据（空间）局部性原理，在程序执行时，如果一个信息项正在被访问，那么近期它很可能也会被再次访问，存储在它附近的信息也很可能被访问到。

变量名的合法性

python变量名的合法性，只能由数字、大小写字母、下划线构成，**不能以数字开始。**

计算机程序结构

计算机程序中的3种基本控制结构是：顺序结构、分支结构和循环结构。

网络信息安全

防火墙是一种内外网的隔离技术

> **防火墙** 是一种网络安全系统，用于监控和控制进出网络流量，通常位于内部网络和外部网络（如互联网）之间。它的主要功能是根据预定义的安全规则过滤数据包，阻止未经授权的访问，同时允许合法的通信。防火墙可以是硬件设备、软件应用程序，或者是两者的结合。它确实起到了内外网隔离的作用，防止外部威胁进入内部网络。

启发式杀毒

> **启发式杀毒** 是一种基于行为分析和模式识别的技术，旨在检测已知病毒的变种以及未知病毒。它通过分析文件的行为特征（如文件结构、代码片段、执行行为等）来判断是否存在恶意活动，而不仅仅是依赖于已知病毒的签名库。

加密和解密

> - **对称加密**：在这种加密方式中，加密和解密使用相同的密钥。例如，AES（高级加密标准）就是一种对称加密算法。发送方和接收方必须共享同一个密钥，才能进行加密和解密。
> - **非对称加密**：在这种加密方式中，加密和解密使用不同的密钥。非对称加密使用一对密钥：公钥和私钥。公钥用于加密，私钥用于解密（或反之亦然）。常见的非对称加密算法包括 RSA 和 ECC。非对称加密的优点是无需共享同一密钥，提高了安全性。