**自我介绍**

你好，我叫邹静思，本科就读于华南理工大学计算机全英联合班，曾多次获得校级三等奖学金及“三好学生”称号。在科研方面，我获得过学校的专项资金资助，前往加拿大麦吉尔大学参加了人工智能课程的学习，发表过一篇CV方向的会议论文，也获得过多项数学建模类的奖项。在实习方面，我有过两段测试相关的实习经历，今年在腾讯担任测试开发实习生。在项目方面，我最近的项目是于今年暑假完成的手工内容录入系统的项目。

我的那篇论文是关于face alignment人脸对准的，主要是对经典的人脸对准算法进行了综述和分析。比如说传统的SDM梯度下降算法，CFSS由粗到细的搜索算法，到现在常用的深度学习方法：DCNN三层级联的神经网络，MDM结合了能够推断上下文的RNN和CNN，TCDCN和MTCNN结合了多任务学习，3D。对比分析常见的数据集LFPW 300W数据集。

**项目介绍**

用户在外通常会收到一些纸质信息：比如名片，手写的电话号码等，之后查找起来不太方便。这个项目主要是将这些纸质信息通过OCR文字识别转化为电子数据，导入数据库，便于用户查找使用。我们当时是两人一组，我作为组长，负责整个项目的方案设计。整个项目从功能模块上可划分为两个部分：（1）.OCR文字识别算法的实现（2）用户界面的搭建。在算法实现部分，一开始，我发现市面上大部分的文字识别软件都是通过调用百度智能云提供的API端口实现的，这种方法的优点是准确率高(将近98%)，可识别的信息种类繁多(车牌、手写、贺卡)，缺点也很明显，需要联网才能运行；第二种是文字识别引擎tesseract，py和cpp都有相应的tesseract接口可以调用，但它的缺点是识别的准确率比较低80%左右；综合考虑到客户信息的隐私需要保护以及准去率等问题后，我们决定采用第三种方法：自己搭建深度学习模型，基于TensorFlow搭建神经网络，并进行训练。至于网页部分，我们使用Django应用框架来开发。

**最难的部分**：我们发现现有的文字识别算法基本都是基于ctpn和crnn。然后我们对最经典的那篇paper进行了复现，在github上找了相关代码，并且在pycharm上配置相关环境，比如TensorFlow、pytorch等，修改了一些版本问题，最后调试debug，跑通。

MVC全称是model view controller，它是创建web应用程序的框架模式。

Model是应用程序中处理数据逻辑的部分，通常负责在数据库中存取数据

View是处理数据显示的部分，通常是依据model里的数据创建的

Controller处理用户交互的部分，负责从view读取数据，控制用户输入，并向model发送数据。控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

Django优缺点

完善的文档：经过十多年的发展和完善，Django有广泛的实践案例和完善的在线文档。开发者遇到问题时可以搜索在线文档寻求解决方案。

强大的数据库访问组件：Django的Model层自带数据库ORM组件，使得开发者无须学习SQL语言即可对数据库进行操作。

自带台管理系统admin：只需要通过简单的几行配置和代码就可以实现一个完整的后台数据管理控制平台。

Django的缺点大包大揽: 对于一些轻量级应用不需要的功能模块Django也包括了，不如Flask轻便。

过度封装: 很多类和方法都封装了，直接使用比较简单，但改动起来就比较困难。

这个数据库采用的是SQLite 不是mysql。其中用户表使用的是Django 自带的User model

**前后端如何交互?**

1.使用Django中自带的模板。前端通过form 表单向后端提交数据；后端通过渲染函数传递数据给模板文件并渲染

2. 使用Ajax。前端通过Ajax向后端发送请求提交数据；后端返回响应；前端收到响应后通过js对页面上显示的东西进行修改。

**实习经历**

第一段实习是在大二寒假，在深圳市优学天下公司担任软件测试实习生。他们公司的主打产品是一款叫优学派的学生平板电脑，而我的主要工作是对各个版本的语音助手进行功能测试并向研发人员进行反馈改进。这里的语音助手就类似苹果手机的Siri以及微软小娜，可以语音控制，实现各个应用的调用和日常对话。这段实习经历让我对功能测试、系统测试、探索测试等测试流程以及流水线生产模式有了初步的了解。

功能测试——>系统测试——>探索测试——>发布

功能测试：保证功能是没问题的。针对一台机器进行登录，点播(打赏特效、互动、评论)、消息通知模块的测试。一个需求对应一个模块，需求转变为测试用例，然后进行测试。

系统测试：保证在各个类型的手机上不会出问题。针对各个版本的手机进行测试

探索测试：找出影响用户体验的地方，提出改进，需求单上没有写的。monkey test 一起探索新功能

可能会遇到前后端的问题：UI显示/调用出错：音量调节相反之类的

第二段实习是今年暑假在腾讯PCG部门担任测试开发实习生。总共参与了两个项目：第一个是完成loki平台支持TRPC协议抓包工作，主要是一个抓包工具的迭代开发；第二个是利用自动化测试框架UiAutomator和unittest(测试用例,测试套件)，编写检测X5内核是否发生内存泄漏。

Loki平台是一个流量监测平台,搭建这个平台主要是为了另外一个搜索算法组选择最优的搜索算法的判断依据。他们负责对搜索引擎的搜索算法进行优化，提供若干搜索算法，而一个好的搜索算法应该在较短时间内便能获得较大的流量，因此便需要这样一个Loki平台进行流量的检测。当时的loki平台的流量获取只支持几个部门的http服务，然后我做的就是重写一个支持rpc协议的接口。

Rpc协议：RPC（Remote Procedure Call Protocol）远程过程调用协议，是一个计算机通信协议。 该协议允许客户端的程序通过网络远程调用服务器的子程序。在我看来，http其实也是可以支持服务间的远程调用，缺点是消息封装臃肿。腾讯要转成rpc的原因是：http协议的接口的开发，一份接口文档，严格地标明输入输出是什么？说清楚每一个接口的请求方法，以及请求参数需要注意的事项等。接口可能返回一个JSON字符串或者是XML文档。然后客户端再去处理这个返回的信息。优点（1）RPC效率高：大型企业来说，内部子系统较多、接口非常多的情况下，RPC框架的好处就显示出来了，首先就是长链接，不必每次通信都要像http一样去3次握手什么的，减少了网络开销。（2）在RPC框架下，对调用方来说是无感知、统一化的操作。程序员无需额外地为这个交互作用编程。说得通俗一点就是：像访问本地系统资源一样，去访问远端系统资源。

http和rpc区别：

速度来看，RPC要比http更快，虽然底层都是TCP，但是http协议的信息往往比较臃肿

难度来看，RPC实现较为复杂，http相对比较简单

灵活性来看，http更胜一筹，因为它不关心实现细节，跨平台、跨语言。（只要遵守rest）

因此，两者都有不同的使用场景：

如果对效率要求更高，并且开发过程使用统一的技术栈，那么用RPC还是不错的。

如果需要更加灵活，跨语言、跨平台，显然http更合适

X5内核是QQ浏览器团队自研的一个web引擎，为了解决系统内核的安全、稳定、兼容性等问题，现在已经接入QQ、WX、游戏，用户在使用它浏览网页的时候就有可能发生内存泄漏。开发同学负责打包发布内核，我负责在wx和qq这两个监测场景下编写自动化脚本，它所需要具备的能力有：（1）构造业务的测试用例 （2）解析测试结果文件，并判断是否存在异常（3）获取并上传hprof文件，便于定位分析问题

内存泄漏：一个对象已经不需要再使用本该被回收时，另外一个正在使用的对象持有它的引用从而导致它不能被回收，这导致本该被回收的对象不能被回收而停留在堆内存中（内存无法释放），这就产生了内存泄漏。内存泄漏是造成应用程序OOM的主要原因之一

常见的内存泄漏：（1）单例造成的内存泄漏（2）非静态内部类创建静态实例造成的内存泄漏（3）Handler造成的内存泄漏（4）线程造成的内存泄漏（5）资源未关闭

**专业技能**

**C++**

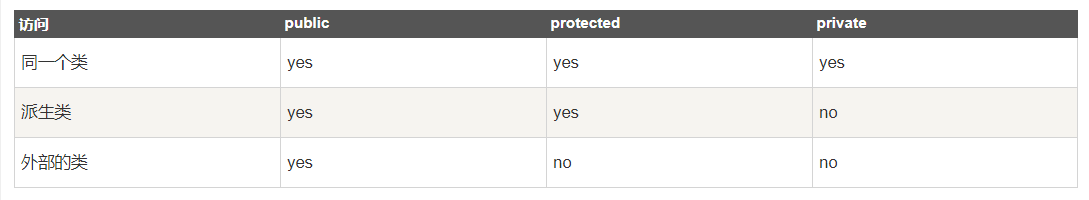
内存管理

1. **五个区：**
   1. 栈：执行函数时，函数内局部变量创建/释放。
   2. 堆：new分配的内存块，如果没有delete，在程序结束后自动回收。
   3. 自由存储区：由malloc等分配的内存块，与堆相似，用free结束
   4. 全局/静态存储区：分配
   5. 常量存储区：存放常量
2. **堆和栈的区别：**
3. 管理方式：栈编译器自动管理；堆程序员控制
4. 空间大小：堆大栈小
5. 能否产生碎片：堆因为频繁的new/delete造成大量碎片；栈先进后出，一一对应没有碎片

封装继承多态

继承的意义：主要实现重用代码，节省开发时间，增加可修改性。继承代表了 **is a** 关系，当创建一个类时，您不需要重新编写新的数据成员和成员函数，只需指定新建的类继承了一个已有的类的成员即可。这个已有的类称为基类，新建的类称为派生类。

派生类可以访问基类中所有的非私有成员。因此基类成员如果不想被派生类的成员函数访问，则应在基类中声明为 private。我们可以根据访问权限总结出不同的访问类型



封装：

**数据封装**是一种把数据和操作数据的函数捆绑在一起的机制，**数据抽象**是一种仅向用户暴露接口而把具体的实现细节隐藏起来的机制。【成员变量都是private，接口函数get和set都是public】

多态：多种形态，类之间是通过继承关联时，就会用到多态。调用成员函数时，会根据调用函数的对象的类型来执行不同的函数。把父类的函数声明成virtual/纯虚函数。每个子类都有一个函数独立实现。这就是**多态**的一般使用方式。有了多态，您可以有多个不同的类，都带有同一个名称但具有不同实现的函数，函数的参数甚至可以是相同的。【Animal类，叫声】

new和malloc总结

1. malloc/free是库函数，需要头文件支持；new/delete是C++关键字，需要编译器支持。new的底层就是malloc ，new先调用malloc申请空间,然后调用构造函数释放内存。
2. malloc申请的内存空间在堆上(用于程序的内存动态分配)，new分配内存的位置是一片自由存储区。
3. malloc在申请空间的时候需要手动计算空间大小 ;new只需要声明空间是什么类型的 不需要指定内存块的大小
4. 返回值类型：malloc返回值是void\* 必要时需要强制转换；new不需要 因为new后面跟的就是空间类型
5. new/delete 允许函数重载；malloc不允许重载
6. new/delete调用构造析构函数 ；malloc/free不调用

**Python**

迭代器、生成器、装饰器

迭代器是一个可以记住遍历的位置的对象;从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退;迭代器有两个基本的方法：iter(): 创建迭代器对象和 next(): 返回迭代器的下一个元素.

yield 的函数被称为生成器（generator）;生成器是一个返回迭代器的函数

用于拓展原来函数功能的一种函数，目的是在不改变原函数名(或类名)的情况下给函数增加新的功能

“可变对象”和“不可变对象”

可变对象包括列表（list）、字典（dict）、集合（set）等；

不可变对象包括数值型：整型（int）浮点型（float）、字符串型（string）和元组（tuple）等。  
不可变对象，该对象所指向的内存中的值不能被改变。当改变某个变量时候，由于其所指的值不能被改变，相当于把原来的值复制一份后再改变，这会开辟一个新的地址，变量再指向这个新的地址。   
可变对象，该对象所指向的内存中的值可以被改变。变量（准确的说是引用）改变后，实际上是其所指的值直接发生改变，并没有发生复制行为，也没有开辟新的出地址，通俗点说就是原地改变。

深浅拷贝

1. 由于 Python 内部引用计数的特性，对于不可变对象，浅拷贝和深拷贝的作用是一致的，就相当于复制了一份副本，原对象内部的不可变对象的改变，不会影响到复制对象
2. 浅拷贝的拷贝。其实是拷贝了原始元素的引用（内存地址），所以当拷贝可变对象时，原对象内可变对象的对应元素的改变，会在复制对象的对应元素上，有所体现
3. 深拷贝在遇到可变对象时，又在内部做了新建了一个副本。所以，不管它内部的元素如何变化，都不会影响到原来副本的可变对象
4. （1）copy浅拷贝，没有拷贝子对象，所以原始数据改变，子对象会改变   
   （2）深拷贝，包含对象里面的自对象的拷贝，所以原始对象的改变不会造成深拷贝里任何子元素的改变

Python的内存管理与垃圾回收机制

Python 内存回收涉及三个概念： 引用计数，标记-清除，分代回收

1. 引用计数：当一个对象的引用计数为0时，回收该对象。编译器必须实时监控程序中对象的状态，一旦发现计数为0，立即释放对象的空间。虽然简单但是编译器的工作量比较大。
2. 标记-清除：在引用计数的基础上，解决循环引用的问题。不改动真实计数，而是生成对象计数的一个副本，对该副本进行操作。副本的作用是寻找root object的集合(该集合中的对象不能被回收)。使用两条链表，一条维护root object，称root链表，一条维护剩下的对象，称unreachable 链表，遍历标记unreachable链表，如果其中有被root链表中对象引用的对象，就把它移到root链表里面。标记后 unreachable 链表中的所有对象都是垃圾对象。
3. 分代回收：0，1，2代，对应3条链表。基本原则是存活越久的对象，不是垃圾的可能性越高，越减少对它的检查。新创建的对象放在0号链表，当某一代对象经历过垃圾回收，依然存活，就被归入下一代对象。可以看出0代扫描最频繁，1代次之，2代最低。

内存池机制：Python的内存池分为大内存和小内存：（256K为界限分大小内存）

第1层和第2层：内存池，有Python的接口函数PyMem\_Malloc实现-----若请求分配的内存在1~256字节之间就使用内存池管理系统进行分配，调用malloc函数分配内存，但是每次只会分配一块大小为256K的大块内存，不会调用free函数释放内存，将该内存块留在内存池中以便下次使用。

第0层：大内存--若请求分配的内存大于256K，malloc函数分配内存，free函数释放内存。

进程和线程

1. 定义：进程就是一个运行的程序，进程是资源分配最小单位：任务管理器，一个个运行的进程；线程是程序执行的最小单位，是CPU调度和分派的基本单位。【python计时器：每隔1分钟要读取一个hprof文件，Timer，一个个任务就是一个个线程】
2. 进程：代码+系统资源：进程；线程占用的资源要⽐进程少很多，基本上不拥有系统资源，只拥有一点在运行上必不可少的资源(程序计数器、寄存器和栈)，但是它可与同属的一个进程的其它的线程共享进程所拥有的全部资源
3. 一个进程必须要包含一个主线程，同时也可以拥有若干线程。
4. 进程有自己独立的地址空间，每启动一个进程，系统都会为其分配地址空间，建立数据表来维护代码段、堆栈段和数据段，线程没有独立的地址空间，它使用相同的地址空间共享数据
5. 进程独立，线程依赖。多进程程序更安全，生命力更强，一个进程死掉不会对另一个进程造成影响（源于有独立的地址空间），多线程程序更不易维护，一个线程死掉，整个进程就死掉了（因为共享地址空间）
6. 进程对资源保护要求高，开销大，效率相对较低，线程资源保护要求不高，但开销小，效率高，可频繁切换

多进程和多线程的区别



并行/并发

并发：任务数多于cpu数，宏观上多个任务一起执行

并发：多核，任务真的一起执行 OpenMI微分法计算pi

多线程multhreading：同一时间，分批量做很多时间，节省计算时间。

建立线程：threading.Thread() start(),在参数target中引用它所需要做的任务的函数名。

GIL:Global Interpreter Lock，全局解释器锁

python里的多线程是单cpu意义上的多线程，不是并行操作，是并发操作，在不停地切换

python实际只能让一个线程在一个时间计算一个东西，在某一线程进行I/O操作的时候，就会给另外一个线程，只节省了读写时间

python多线程对IO密集型任务有用，对计算密集型任务不一定有用

多进程：应对多核处理器，并行操作。Pool池，自动分配，还有返回值

运行时间：Multithread>Normal>multiprecess

总结：多线程有两个好处：CPU并行，IO并行，单核多线程相当于自断一臂。所以，在Python中，可以使用多线程，但不要指望能有效利用多核。Python可以通过多进程实现多核任务。多个Python进程有各自独立的GIL锁，互不影响。

**Linux**

处理目录和文件的常用命令

ls（英文全拼：list files）: 列出目录及文件名

cd（英文全拼：change directory）：切换目录

pwd（英文全拼：print work directory）：显示目前的目录

mkdir（英文全拼：make directory）：创建一个新的目录

rmdir（英文全拼：remove directory）：删除一个空的目录

cp（英文全拼：copy file）: 复制文件或目录

rm（英文全拼：remove）: 移除文件或目录

mv（英文全拼：move file）: 移动文件与目录，或修改文件与目录的名称

cat 由第一行开始显示文件内容

tac 从最后一行开始显示，可以看出 tac 是 cat 的倒着写！

nl 显示的时候，顺道输出行号！

more 一页一页的显示文件内容

less 与 more 类似，但是比 more 更好的是，他可以往前翻页！

head 只看头几行

tail 只看尾巴几行

**数据结构**

数组和链表

相同点：两种数据结构都属于线性表。所谓线性表，就是所有数据都排列在只有一个维度的“线”。只有一个前趋，也只有一个后继。

不同点：

1. 逻辑结构：数组占用的是一块连续的内存区；而链表在内存中，是分散的，又一个个的节点组成(节点包括两个部分：数据部分+指针)，正是由于指针将不连续的他们串起来，才形成了逻辑上的线性表。逻辑结构的不同导致他们在增删改查等操作时的时间复杂度也不一样。
2. 对于访问，数组O(1)：“随机访问”通过数组下标可以直接访问到某一元素；链表O(n)：“顺序访问”
3. 对于增删操作，因为数组在内存中是连续存储的，要想在某个节点之前增加，必须要把此节点往后的元素依次后移，时间复杂度是O(n); 而链表中只需要改变节点中的两个“指针”的指向，时间复杂度是O(1)
4. 从内存角度来说，使用**数组(栈区)**的话，我们需要申请一块连续的内存空间，一开始就必须确定好它的空间的大小。在运行的时候空间的大小是无法随着你的需要进行增加和减少而改变的，当数据量比较大的时候，有可能会出现越界的情况，数据比较小的时候，又有可能会浪费掉内存空间。**链表(堆区)**只需在用的时候申请就可以，根据需要来动态申请或者删除内存空间，对于数据增加和删除以及插入比数组灵活。缺点：数据量比较小的情况下，指针占用的内存比较多，也相当于是一种空间的浪费
5. 数组应用场景：数据比较少；经常做的运算是按序号访问数据元素； 链表应用场景：对线性表的长度或者规模难以估计；频繁做插入删除操作；构建动态性比较强的线性表。

排序算法



希尔排序：希尔排序是按照不同步长对元素进行插入排序，当刚开始元素很无序的时候，步长最大，所以插入排序的元素个数很少，速度很快；当元素基本有序时，步长很小，插入排序对于有序的序列效率很高。

**计算机网络**

网页http请求的整个过程：

1. 打开浏览器，地址栏输入blog.csdn.net。
2. 开始进行域名解析

（1）浏览器自身搜dns缓存 搜blog.csdn.net有没有缓存 看看有没有过期，如果过期就这个结束；

（2）搜索操作系统 自身的dns缓存；

（3）读取本地的host文件；

（4）浏览器发起一个dns的一个系统调用

3. 浏览器获得域名对应的ip地址后 发起http三次握手

4. tcp/ip 链接建立起来后，浏览器就可以向服务器发送http 请求 。

5. 服务器端接受到请求，根据路径参数，经过后端的一些处理之后，把处理后的一个结果数据返回给浏览器，如果是一个完整的网页，就是把完整的html页面代码返回给浏览器。

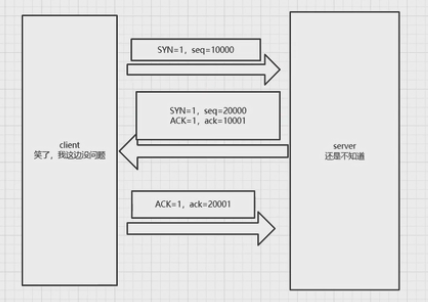
6. 浏览器拿到html页面代码，解析和渲染页面，里面的 js、css图片资源都需要经过上面的步骤。

7. 浏览器拿到资源对页面进行渲染，最终把一个完整的页面呈现给用户。

三次握手

Tcp/ip是传输层面向连接的安全可靠的协议，三次握手的机制是为了保证能建立安全可靠的连接。

TCP建立的连接是安全可靠的连接：客户端知道自己能连上服务端，服务端也能连上上自己；服务端也要知道自己能连上客户端，客户端能连上自己。



Tcp报文中有：32位序号seq(initial sequence number) + 32位确认序号ack(acknowledge number) 还有标记位：ACK

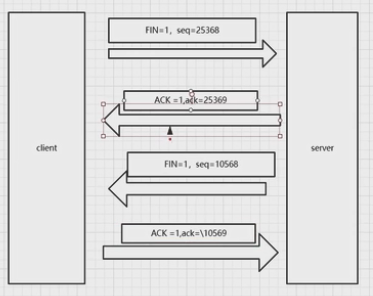
（1）客户端给服务端发起一个建立连接的请求(报文)：标志位SYN=1，同时会随机生成一个32位的initial sequence number初始化序列号X。

（2）服务端在收到了请求之后，就会发出一个确认消息报：ACK=1，同时有一个32位的acknowledge number确认序号X+1。除此以外，它还会发送一个建立连接的请求：SYN=1，seq=Y。它会把上面这些打包发送给客户端。在经过上面两次握手之后，客户端这边其实已经可以满足：自己连上服务端，服务端也能连上自己的条件。现在还不能满足的就是：服务端这边能确定客户端能连上自己，但是不确定自己能否连上服务端。在这种情况下，便产生了第三次握手

（3）客户端在收到服务端的消息之后，立马给服务端一个消息确认包：ACK=1，acknowledge number确认序号Y+1

通过这三次握手，我们就能够保证客户端和服务器双方都知道我既能给对方发消息，也能收到对方的消息，从而建立了安全可靠的传输通道，也就是建立了全双工信道。

四次挥手



1. 客户端请求断开连接：FIN位=1，seq = X
2. 服务端回应应答数据包：ACK=1，ack = X+1
3. 服务端还会发送一个断开连接的数据包：FIN = 1，seq = Y
4. 客户端在收到消息后，会发送一个消息确认包：ACK=1，ack = Y+1

对比三次握手，我们来思考一下第二次和第三次挥手是否能像握手一样，合并成一次？实际情况是不可以的，原因是：tcp连接是安全可靠的，客户端发送FIN之后，已经准备好了断开连接，不会在发送数据给服务端；但是服务端还没有做好准备，可能还有消息需要发送给客户端，因此这个逻辑就需要拆开：所以服务端只能先回复一个ACK：相当于告诉客户端“好的客户端，你的邀请我已经收到了，但是我还没有准备好，你先等我的消息，我这边准备好断开连接之后会通知你的”当服务端也准备好了，才会进行第三次挥手操作。

tcp和udp的区别

3．TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的

UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）

4．每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信

5. TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道

**TCP和UDP的区别【都是传输层】**

TCP：面向连接（如打电话要先拨号建立连接）

1)优点：可靠,无差错、不丢失、不重复，且按序到达。TCP数据传输之前，会有三次握手；而且传输数据时，有滑动窗口。超时重传。拥塞控制等机制；数据传输完后，会断开连接来节约系统资源2)缺点：慢，效率低，占用系统资源高。而且TCP有确认机制、三次握手机制，导致TCP容易被人利用。

UDP：无连接的，即发送数据之前不需要建立连接

1)优点：快，尽最大努力交付，不保证可靠传输。比TCP稍安全，UDP没有TCP的握手、确认、窗口、重传、拥塞控制等机制，UDP是一个无状态的传输协议，所以它在传递数据时非常快。没有TCP的这些机制，UDP较TCP被攻击者利用的漏洞就要少一些。2)缺点：不可靠，不稳定 因为UDP没有TCP那些可靠的机制，在数据传递时，如果网络质量不好，就会很容易丢包。

HTTP 报文结构

HTTP 请求报文由请求行、请求头、空行和请求内容 4 个部分构成。



请求行：由请求方法字段、URL 字段、协议版本字段三部分构成，它们之间由空格隔开。

常用的请求方法有：GET、POST、HEAD、PUT、DELETE、OPTIONS、TRACE、CONNECT。

请求头：请求头由 key/value 对组成，每行为一对，key 和 value之间通过冒号(:)分割。请求头的作用主要用于通知服务端有关于客户端的请求信息

Post和get的区别：

GET和POST是HTTP协议中的两种发送请求时最常用的方法。

1、get请求：可被缓存；参数保留在浏览器历史记录中；可被收藏为书签；有长度限制

2、Post请求：不会被缓存；参数不会保留在浏览器历史记录中；不能被收藏为书签；对数据长度无要求

3、GET把参数包含在URL中，POST通过request body传递参数。

状态码：由 3 位数字组成，第一位标识响应的类型，常用的 5 大类状态码如下：

1xx：表示服务器已接收了客户端的请求，客户端可以继续发送请求

2xx：表示服务器已成功接收到请求并进行处理

3xx：表示服务器要求客户端重定向

4xx：表示客户端的请求有非法内容

5xx：标识服务器未能正常处理客户端的请求而出现意外错误

常见状态码说明：

200 OK： 表示客户端请求成功

400 Bad Request： 表示客户端请求有语法错误，不能被服务器端解析

401 Unauthonzed： 表示请求未经授权，该状态码必须与WWW-Authenticate 报文头一起使用

404 Not Found：请求的资源不存在，例如输入了错误的 url

500 Internal Server Error： 表示服务器发生了不可预期的错误，导致无法完成客户端的请求

503 Service Unavailable：表示服务器当前不能处理客户端的请求，在一段时间后服务器可能恢复正常响应头

重定向：当client向server发送一个请求，要求获取一个资源时，在server接收到这个请求后发现请求的这个资源实际存放在另一个位置，于是server在返回的response中写入那个请求资源的正确的URL，并设置reponse的状态码为301（表示这是一个要求浏览器重定向的response)，当client接受到这个response后就会根据新的URL重新发起请求。

HTTP与HTTPS的区别：

1、https协议需要到ca申请证书，一般免费证书较少，因而需要一定费用。

2、http信息是明文传输，https则是具有安全性的ssl加密传输协议。

3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

4、http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全

**操作系统**

死锁deadlock

定义：一组进程中的每一个进程都在等待只有这组进程中的进程才能触发的事件，僵局

产生死锁的必要条件：

（1）mutable互斥条件：进程对分配给它的资源进行排它性使用，在一段时间内，一个资源只能被一个进程占用

（2）hold and wait请求和保持条件：进程已经只少保持了一个资源，但是又提出了新的资源请求，该资源又被其他进程占用，此进程被阻塞，但是并没有释放自己拥有的资源；

（3）不可抢占条件：分配给进程的资源，除非进程自己释放，否则其他进程不可抢占；

（4）循环等待：发生死锁时，必然存在一个进程-资源的循环链；

只要破坏四个条件中的一个，就能阻止死锁情况的发生；

**测试**

单元测试（unit testing），是指对软件中的最小可测试单元进行检查和验证。(函数/类/代码)

集成测试，在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。一些局部反映不出来的问题，在全局上很可能暴露出来。

黑盒测试：功能测试、数据驱动测试，它将被测软件看作一个打不开的黑盒，主要根据功能需求设计测试用例，进行测试。优点：

① 与软件具体实现无关，如果软件实现发生了变化，测试用例仍可用；

② 设计黑盒测试用例可以和软件实现同时进行，因此可压缩项目总开发时间。

黑盒测试常用方法：等价类划分、边界值分析、因果图、决策表分析

（1）等价类划分：把所有可能的输入数据，即程序输入域划分为若干个互不相交的子集，称为等价类，然后从每个等价类中选取少数具有代表性的数据作为测试用例，进行测试。

（2）边界值分析：在等价类的划分过程中产生了许多等价类边界。边界是最容易出错的地方，所以，从等价类中选取测试数据时应该关注边界值。选取正好等于、刚刚大于或刚刚小于等价类边界的值作为测试数据，而不是选取等价类中的典型值或任意值做为测试数据。

（3）因果图：确定软件规格中的原因和结果。分析规格说明中哪些是原因（即输入条件或输入条件的等价类），哪些是结果（即输出条件），并给每个原因和结果赋予一个标识符。

（4）决策表：是把作为条件的所有输入的各种组合值以及对应输出值都罗列出来而形成的表格。它能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。

“白盒”法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。“白盒”法是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。