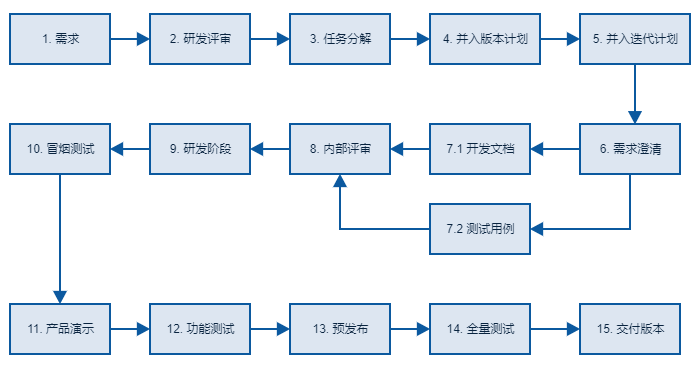
# 【功能方面】

# 1.自我介绍

# 2.让你介绍你做的项目的业务流程



# 3.软件测试的流程

需求调查：全面了解系统概况、应用领域、软件开发周期、软件开发环境、

测试计划

测试用例设计

测试用例评审

搭建测试环境，进行冒烟测试

功能测试阶段：执行用例、发现bug、提交bug、开发人员确认、跟踪bug修复

# 4.你是怎样理解软件测试的

软件质量保证与测试是根据软件开发阶段的规格说明和程序的内部结构而精心设计的一批测试用例（即输入数据和预期的输出结果），并根据这些测试用例去运行程序，以发现错误的过程。它是对应用程序的各个方面进行测试以检查其功能、语言有效性及其外观排布。

# 5.你认为什么是bug，说一下你印象深刻的bug，结合项目说一下你们公司有哪些类型的bug和bug评判严重程度的标准

1. 印象最深刻的bug：

* 平均下浮率错误：在测试某项目平均下浮率时，发现我计算的下浮率和接口返回的下浮率不同，经过确认，我是把每个下浮率都进行四舍五入之后再求平均，而开发没有将每个下浮率都进行四舍五入，直接拿原始（多位数据）数据求平均下浮率。
* 个人中心：修改密码后仍可操作系统。

1. bug评判严重程度的标准：优先级：1最高 5最低 严重程度：1最高 5最低

Severity（严重程度）

1. Blocker 有妨碍的：即系统无法运行、崩溃或严重资源不足、应用模块无法启动或异常退出、无法测试、造成系统不稳定

* 严重花屏
* 内存泄露
* 用户数据丢失或破坏
* 系统崩溃/死机/冻结
* 模块无法启动或异常退出
* 严重的数值计算错误、
* 功能设计与需求严重不符

1. Critical 紧要的：即影响系统功能或操作，主要功能存在严重缺陷，但不会影响到系统稳定性。

* 功能未实现
* 功能错误
* 系统刷新错误
* 数据通讯错误
* 轻微的数值计算错误
* 影响功能及界面的错误字或拼写错误
* 安全性问题

1. Major 严重的：即界面、性能缺陷、兼容性

* 操作界面错误（包括数据窗口内列名定义、含义是否一致）
* 边界条件下错误
* 提示信息错误（包括未给出信息、信息提示错误等）
* 长时间操作无进度提示
* 系统未优化（性能问题）
* 光标跳转设置不好，鼠标（光标）定位错误
* 兼容性问题

1. Minor/Trivial（次要的/不严重的）：即易用性及建议性问题

* 界面格式等不规范
* 辅助说明描述不清楚
* 操作时未给出用户提示
* 可输入区域和只读区域没有明显的区分标志
* 个人不影响产品理解的错别字
* 文字排列不整齐等一些小问题

Priority（优先级）

1. Immediate（立刻）

即“马上解决”，表示问题必须马上解决，否则系统问题根本无法达到预定的要求。

1. Urgent（紧要、优先）

即“积蓄解决”，表示问题的修复很紧要，很急迫，关系到系统的主要功能模块是否正常。

1. Hign（高度重视）

即“高度重视”，表示有时间就要马上解决，否则系统偏离需求较大或预定功能不能正常实现。

1. Normal（正常）

即“正常处理”，进入个人计划解决，表示未不影响需求的实现，但是影响其他使用方面，比如页面调用出错，调用了错误等。

1. Low（稍缓）

即“低优先级”，即问题在系统发布以前必须确认解决或确认可以不予解决

# 6.如何保证软件测试的质量，如何最大限度的保证软件的质量

测试并不能够最大限度的保证软件的质量，软件的高质量是开发和设计出来的，而不是测试出来的，它要通过对软件开发流程的监控，使得软件开发的各个阶段都要按照指定的规程进行，通过对各个阶段产物的评审，QA对流程的监控，对功能及配置的审计来达到开发的最优化，当然测试也是保证软件质量的一个重要方式，是软件质量保证工程的一个重要组成部分。

# 7.如何保证你写的测试用例覆盖所有的场景

1. 测试需求分析需全面

需求分析需两步：

1. 测试需求的来源
2. 显示需求

* 原始需求说明书或需求矩阵列表
* 产品规格书
* 软件需求文档
* 有无继承文档
* 经验库
* 通用的协议规范

1. 隐性需求

用户的主观感受，市场的主流观点，专业人士的评价分析

1. 需求分析、产生测试需求文档

将不同的需求来源划分成一个个需求点，针对每一点进行测试分析：

1. 界定测试范围
2. 利用各种测试设计的方法产生测试点

* 在测试方面，注意：

1）、分析出口入口。从入口分析，将可能出现的环境，条件，操作等内容分类组合，然后根据测试达人的方法进行整合，逐一验证。从出口分析，将可能出现的结果进行统计，根据结果的不同追根溯源，再找到不同的操作以及条件等内容，统计成文档，逐一验证。

2）、多种测试手法的学习和使用。大家可能更多的关心测试方法， 但是具体操作的手法也是需要注意的。毕竟测试方法比容易找到，大家都很熟悉，如果每个人的测试手法总结出来并在自己的测试实施中加以使用，可能会受到意向不到的结果。

* 在测试流程方面，可做如下注意：

1. 初期做好需求分析，将需求主键细化到小功能点，针对每个功能点进行测试设计，对于完成的测试设计文档，经过项目相关人员的检查评审，做成需要的初稿。
2. 在测试过程中，根据需求变更和具体执行过程中遇到的问题完善测试设计文档。
3. 测试执行结束后，对于出现的问题进行总结。其中包括自身本身发现的问题，也可能是客户提出的问题。将总结出来的结果同和到测试设计中汇总，进一步完善测试设计文档

对于一次测试，是不可能覆盖全面的测试的。需要多次去总结积累，才会让测试越来越全面。

* 在测试思维方面，可做如下注意：

1. 测试全面不等于全面测试。不同阶段对于软件测试有不同的要求，比如再前期版本中，对于不重要的画面问题或者是细小的功能问题就不需要关心。但是在验收阶段，这些内容可能更需要注意。
2. 学无止境，只有不断的学习不断的去思考，才能使自己的测试能力更强，测试对象的全面性也更完善。
3. 当测试需求分析完成，并且写成文档后，要进行测试需求评审，保证需求的准确性以及完整性
4. 测试需求完成以后，可以根据测试需求设计测试用例

要保证测试用例能够全面覆盖测试需求，要包含所有的情况。

测试用例设计上分为单功能测试用例和测试场景设计，单功能测试覆盖的需求中的功能点，测试场景覆盖需求中的业务逻辑。

在设计测试用例的时候，可以使用多宗测试用例设计方法。

1. 首先进行等价类划分，包括输入条件和输出条件的等价划分，合理设置有效等价类和无效等价类，这是减少工作量和提高测试效率最有效的方法。
2. 必须使用边界值分析，经验表明，这种方法设计出的用例能发现很多程序错误
3. 可以使用错误推断法追加一些测试用例，这需要依靠智慧和经验
4. 对照程序逻辑检查已设计出的测试用例的逻辑覆盖度，如果没有达到覆盖标准应当再补充足够的测试用例。
5. 如果程序的功能说明中含有输入条件的组合情况，一开始就可选因果图和判定表驱动法。
6. 对于参数配置类的软件，要用正交试验法选择较少的组合方式达到最佳效果。
7. 对于业务清晰的系统，可以利用场景法贯穿整个测试方案过程，在案例中综合使用各种测试方法
8. 当测试用例设计完成后，要组织测试用例的评审，这样可以吸取别人的意见，减少遗漏，补全测试用例。
9. 测试用例编写完成后，就是测试执行

测试用例执行100%覆盖

在测试执行过程中，要继续对测试用例补充完善，确保提高测试覆盖率

1. 在整个测试过程中，需求都是不可能不变的，所以要及时的更新测试需求、测试用例
2. 要将测试需求、测试用例以及发现的bug关联起来，便于管理和跟踪，同时也便于查看覆盖率。

# 8.如果用户在正式环境发现一个bug，你在测试环境没有发现，怎么办

# 9.项目测试到什么程度就可以上线了

测试通过标准：

1. 测试用例执行通过率100%
2. 一、二级bug都已解决完毕，其他未解决bug不能超过5%
3. 回归测试通过，产品验收通过
4. 主要业务能够正常运行，所有输出日志能够正确保存。基本上就可以上线了
5. 日志很重要，建议日志能够追溯出来具体引发异常的原因
6. 其他的非主要业务，有些小bug但是不影响正常使用的，可以在后续版本迭代中修复。
7. 不影响正常使用的是 业务最少能有一种方式流转完成，或者成功执行

# 10、如果有一个bug，怎么判断这个bug是前端代码导致的还是后端代码导致的

首先理解前后端的概念：

* 前端：前端一般指的是我们肉眼可见的界面设计，主要体现在人机交互的视觉效果，往往需要调用后台的一个接口，进行一个HTTP请求，根据后台反馈回来的数据，渲染到前端页面上。
* 后端（后台）：后台无法肉眼可见，但是主要是为了实现某一具体功能，还有关于数据、逻辑、安全性等底层的东西均属于后端。知道前后端概念后，我们不难理解到底自己提交的bug是属于前端的bug还是属于后端的bug，方便bug定位，便于提高工作效率。
* 常用技巧：通常可以用抓包工具Fiddler进行分析，从三个方面进行分析：请求接口，传参，响应内容
* 请求接口的URL是否正确

如果请求的接口URL错误，为前端的bug

* 传参是否正确

HTTP请求中的参数是否正确，如果HTTP请求中的参数不正确，为前端的bug

* 响应内容是否正确

请求接口URL和传参都正确，如果响应内容不正确，则为后端的bug

* 也可以在浏览器控制台输入js代码调试进行分析

如果定位为后端的bug，如何精确定位是哪里出了bug呢？

1. 查看报错日志，通过日志分析问题点
2. 查看数据库确认数据的正确性
3. 查看缓存是否正确

# 11、测试工作中发现一个bug，而开发人员说不是一个bug，你该怎么处理？

* 说法一：（1、需求不确定； 2、这种情况不可能发生）

1、首先明确开发说不是bug的理由。

2、如果是需求变更，那就找产品经理确认是否是需求变更

3、如果开发说测试环境问题，让他说清楚测试环境问题是什么，按照他做的验证一遍，如果确实如他所说，关闭bug，但是不是他说的那样，继续激活bug给开发解决，确保产品质量。

4、如果开发说用户不存在这种场景，但是我们不认可他说的，把这个bug知会到测试经理，让测试经理去判定。

* 说法二：（站在开发的角度换位思考，从开发角度延伸）

1. 测试人员描述不清晰

工作中也有测试人员把某些“Bug操作步骤”描述的只有自己看得懂，开发人员按照步骤复现Bug不知所云，搞错了问题所在。

解决办法（修改bug操作步骤）：清晰描述、无歧义、无冗余步骤，要达到即使给一个不懂的人去重现这个Bug，也能按照你的操作步骤复现。

（重要的是：有图有真相，带有清晰说明的截图比一大推描述来得直观，易懂。注意对问题区域以强调色（如红色）标识，并配以名字说明）

1. 难以复现的bug

不是所有的问题都能用同样的操作步骤来复现的，有的Bug概率出现甚至偶现，或者是只在测试环境里出现。

解决办法：

针对难以复现的Bug，需要保存截图或者记录log保留证据；对于只在测试环境下才会出现的，找研发在测试环境进行确认。这类Bug要慎重对待，规避风险。

1. 有争议的bug

有争议的Bug多发生于建议类型的Bug：与同类软件不符、易用性、美观性等类型的Bug。

解决办法：

这种问题是否要修改需要根据公司的项目类型进行讨论。开Bug评审会，在开发能实现的情况下说出自己的理由，改善产品。

（在时间允许的情况下，在项目测试接近收尾时开一个bug清除会议，对于剩余bug是否修复做明确处理）

1. 功能性bug

与需求不符、与原型设计不符。有时候开发对需求没有深入了解可能会忽略或者搞错个别功能。

解决办法：

拿证据（需求、设计说明书）给他看，这个bug自然合情合理。（最好在提bug时，就把需求、设计截图带上，自然省去了大多争议，除非开发觉得设计有问题，需要重新进行讨论设计的）

# 【计算机网络基础知识】

# 1、http协议的内容是什么?

HTTP简介：

* + HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写，是用于从万维网（WWW: World Wide Web）服务器传输超文本到本地浏览器的传送协议。
  + HTTP是一个基于TCP/IP通信协议来传递数据（HTML文件、图片文件、查询结果等）。
  + HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于1990年提出，经过几年的使用与发展，得到不断地晚完善和扩展。目前在WWW中使用的是HTTP/1.0的第六版，HTTP/1.1的规范化工作正在进行之中，而且HTTP-NG（Next Generation of HTTP）的建议已经提出。
  + HTTP协议工作于客户端-服务器端架构上。浏览器作为HTTP客户端通过URL向服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求，向客户端发送响应消息。

# 2、请求头的内容有哪些

Host：[www.study.com](http://www.study.com) //请求的地址域名和端口，不包括协议

Connection：keep-alive //连接类型，持续连接

**User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36 //浏览器的用户代理信息

Upgrade-Insecure-Requests：1  // http 自动升级到https，防止跨域问题但是域名端口都不同的不会提升

Accept：text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,’\*/\*’;q=0.8 //浏览器支持的请求类型

Accept-Encoding：gzip、deflate、sdch //浏览器能处理的压缩代码

Accept-Language：zh-CN,zh;q=0.8,en;q=0.6 //浏览器当前设置语言

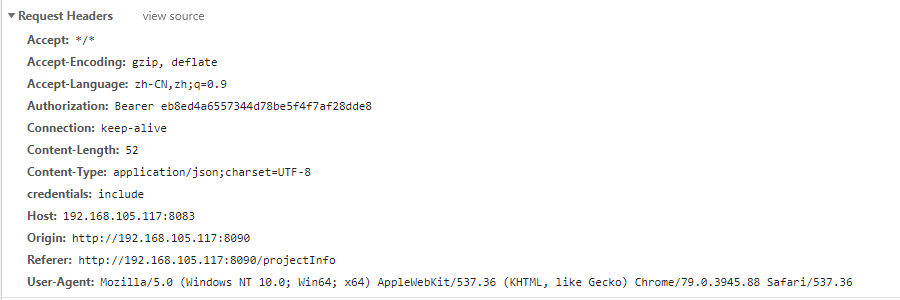
* post新加的请求头

Content-Length：29 //请求参数长度

Cache-Control: max-age=0      //强制要求服务器返回最新的文件内容,也就是不走缓存，返回的200

Origin: http://www.study.com    //请求来源地址，包括协议

referer：<http://www.study.com/day02/01-login.html> //原始的url，不带锚点，比方说在谷歌打开的百度，referer显示的是谷歌的url



# 3、常见的状态码有些

1xx：信息状态码，接收的请求正在处理

2xx：成功状态码，接收的请求成功处理

3xx：重定向状态码，需要进行附加操作以完成请求

4xx：客户端错误状态码，服务器无法处理请求

5xx：服务器端错误状态码，服务器处理请求错误

2xx：成功

200：表示从客户端发来的请求在服务器端被正确处理了。

在响应报文内，随状态码返回的信息会因请求方法的不同而发生改变。

1. get 请求资源是实体会作为响应返回
2. head 请求资源的实体逐日不随报文首部作为响应返回（即在响应中只返回首部，不会返回实体的主体部分）

204：代表服务器接收的请求成功处理，但是不返回实体的主体部分（也不允许）

如果浏览器发送的请求返回204，页面不更新，一般在只需要往服务器端发送信息，而对客户端不需要发送信息内容的情况下使用

206：表示客户端进行了范围请求，而服务器端成功的进行了这部分的GET请求

例如: 响应报文中包含由Content-Range指定范围的实体内容

3xx：重定向

301：永久性重定向，301表示旧地址A的资源已经被永久地移除了（这个资源不可访问了），搜索引擎在抓取新的内容的同时也将旧的地址交换为重定向之后的网址。

302：临时性重定向，302表示旧地址A的资源还在（仍然可以访问），这个重定向只是临时地从旧地址A跳转到地址B，搜索引擎会抓取新的内容而保存旧的网址。

细节：当请求方法是post方法的时候，需要用户确认才可以重定向到新的uri。比如弹框询问用户是否重定向到新的地址。

但是多数浏览器实际实现的时候并没有遵循这一条，所以才在http1.1中引入了303和307两个状态码。302为了兼容老版本也保留了下来。

303：该状态码表示由于请求的资源存在着另一个URI，应使用GET方法定向获取请求的资源，无论请求方法是get或者post，遇到303都直接重定向。

304：该状态码表示客户端发送附带条件的请求时，服务器端允许请求访问资源，但因发送请求未满足条件的情况后，直接返回304Not Modified（服务器端资源未改变，可直接使用客户端未过期的缓存）。

307：当请求方法是post方法的时候，需要用户确认才可以重定向到新的uri。

4xx：客户端错误

400：请求报文中存在语法错误。当错误发生时，需修改请求的内容再次发送请求。

401：未经授权。该状态码表示发送的请求需要有通过HTTP认证（BASIC认证、DIGEST认证）的认证信息。另外若之前已进行过1次请求，则表示用户认证失败。

403：该状态码表明对请求资源的访问被服务器拒绝。（发生403的可能原因，没有IP权限）

404：该状态码表明服务器上无法找到请求的资源。

5xx：服务器端错误

500：该状态码表示服务器端在处理请求时发生了错误。也有可能是web应用存在的bug或某些临时的故障。

503：该状态码表示服务器暂时处于超负荷或正在进行停机维护，现在无法处理请求。

如果事先得知解除以上状况需要的时间，最好写入Retry-After首部字段再返回给客户端。

# 4、http和https的区别有哪些

**HTTPS和HTTP的介绍**

超文本传输协议HTTP协议被用于在Web浏览器和网站服务器之间传递信息，HTTP协议以明文方式发送内容，不提供任何方式的数据加密，如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息，因此，HTTP协议不适合传输一些敏感信息，比如：信用卡号、密码等支付信息。

为了解决HTTP协议的这一缺陷，需要使用另一种协议：安全套接字层超文本传输协议HTTPS，为了数据传输的安全，HTTPS在HTTP的基础上加入了SSL/TLS协议，SSL/TLS依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。

HHTPS协议是由SSL/TLS+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全。

HTTPS协议的主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全；另一种就是确认网站的真实性。

**HTTPS和HTTP的主要区别**

1、申请证书方式不同：

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。

http协议：免费申请。

2、传输信息安全性不同：

http是超文本传输协议，信息是明文传输。如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息。

https则是具有安全性的ssl/tls加密传输协议。为浏览器和服务器之间的通信加密，确保数据传输的安全。

3、端口不同：

http协议：使用的端口是80。

https协议：使用的端口是443。

4、连接方式不同：

http的连接很简单，是无状态的

HTTPS协议是由SSL/TLS+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。

**客户端在使用HTTPS方式与Web服务器通信时的步骤**

1. 客户使用https的URL访问Web服务器，要求与Web服务器建立SSL连接。
2. Web服务器收到客户端请求后，会将网站的证书信息（证书中包含公钥）传送一份给客户端。
3. 客户端的浏览器与Web服务器开始协商SSL/TLS连接的安全等级，也就是信息加密的等级。
4. 客户端的浏览器根据双方同意的安全等级，建立会话密钥，然后利用网站的公钥将会话密钥加密，并传送给网站。
5. Web服务器利用自己的私钥解密出会话密钥。
6. Web服务器利用会话密钥加密与客户端之间的通信。

# 5、tcp和udp的区别

**相同点：**

UDP和TCP协议都是传输层协议。

TCP（Transmission Control Protocol，传输控制协议）提供的是面向连接，可靠的字节流服务。即客户和服务器交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。并且提供超时重发，丢弃重复数据，检验数据，流量控制等功能，保证数据能从一端传到另一端。

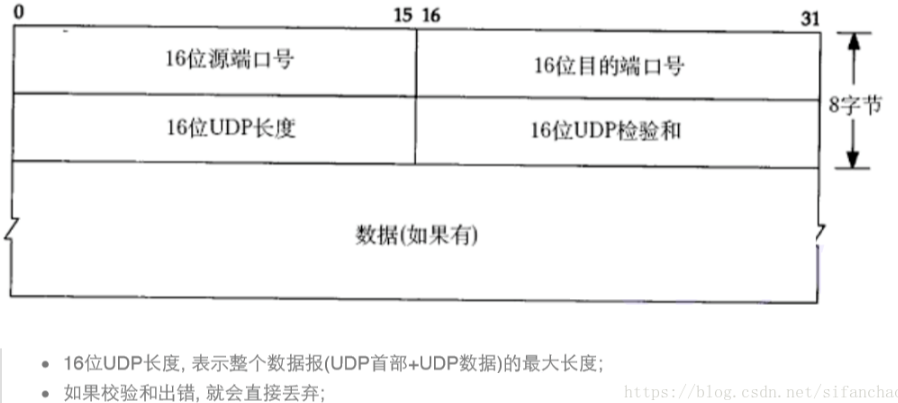
UDP（User Data Protocol，用户数据报协议）是一个简单的面向数据报的传输层协议。它不提供可靠性，只是应用程序传给IP层的数据包发送出去，但是不能保证它们能到达目的地。由于UDP在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，所以传输速度很快。

**不同点：**

1. 报头不同
2. **特点不同**
3. **协议不同**

**UDP：**

1. **报头：**



UDP数据报最大长度64K（包含UDP首部），如果数据长度超过64K就需要在应用层手动分包，UDP无法保证包序，需要在应用层进行编号。

1. **特点：**

* **无连接：知道对端的IP和端口号就直接进行传输, 不需要建立连接。**
* **不可靠：没有确认机制, 没有重传机制; 如果因为网络故障该段无法发到对方, UDP协议层也不会给应用层返回任何错误信息。**
* **面向数据报：不能够灵活的控制读写数据的次数和数量，应用层交给UDP多长的报文, UDP原样发送, 既不会拆分, 也不会合并。**
* **数据收不够灵活，但是能够明确区分两个数据包，避免粘包问题。**

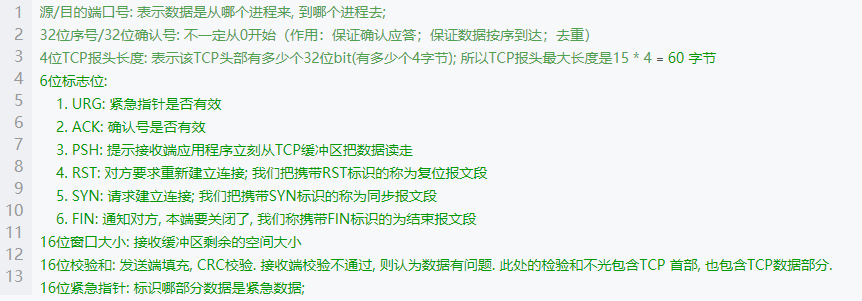
1. **协议**

* NFS: 网络文件系统
* TFTP: 简单文件传输协议
* DHCP: 动态主机配置协议
* BOOTP: 启动协议(用于无盘设备启动)
* DNS: 域名解析协议

**TCP：**

1. **报头：**





1. **特点：**
2. **面向连接（三次握手，四次挥手）**
3. **可靠传输（序列号、确认应答、超时重传、拥塞控制）**

**1、确认应答机制&序列号**

**TCP将每个字节的数据都进行了编号，即为序列号。**

**每一个ACK都带有对应的确认序列号，意思是告诉发送者，我已经收到了哪些数据;；下一次你从哪里开始发。**

**2、超时重传&序列号**

**主机A发送数据给B之后, 可能因为网络拥堵等原因, 数据无法到达主机B; 如果主机A在一个特定时间间隔内没有收到B发来的确认应答, 就会进行重发;**

**主机A未收到B发来的确认应答，也可能是因为ACK丢失了，因此主机B会收到很多重复数据.。那么TCP协议需要能够识别出那些包是重复的包,，并且把重复的丢弃掉.，这时候我们可以利用序列号, 就可以很容易做到去重的效果。**

**3、拥塞控制**

**每次发送数据包的时候, 将拥塞窗口和接收端主机反馈的窗口大小做比较, 取较小的值作为实际发送的窗口。**

**拥塞控制, 归根结底是TCP协议想尽可能快的把数据传输给对方, 但是又要避免给网络造成太大压力的折中方案。**

**提高传输效率：滑动窗口、流量控制、延迟应答、捎带应答。**

**流浪控制：接收端处理数据的速度是有限的. 如果发送端发的太快, 导致接收端的缓冲区被打满, 这个时候如果发送端继续发送, 就会造成丢包, 继而引起丢包重传等等一系列连锁反应。**

**延迟应答：**

如果接收数据的主机立刻返回ACK应答, 这时候返回的窗口可能比较小.  
窗口越大, 网络吞吐量就越大, 传输效率就越高. 我们的目标是在保证网络不拥塞的情况下尽量提高传输效率;

**捎带应答：**

**在延迟应答的基础上, 我们发现, 很多情况下, 客户端服务器在应用层也是 “一发一收” 的.**

**意味着客户端给服务器说了 “How are you”, 服务器也会给客户端回一个 “Fine, thank you”; 那么这个时候ACK就可以搭顺风车, 和服务器回应的 “Fine, thank you” 一起回给客户端。**

1. **面向字节流**

**创建一个TCP的socket, 同时在内核中创建一个发送缓冲区和一个接收缓冲区；**

**另一方面, TCP的一个连接, 既有发送缓冲区, 也有接收缓冲区, 那么对于这一个连接, 既可以读数据, 也可以写数据. 这个概念叫做 全双工 。**

**1、调用write时, 数据会先写入发送缓冲区中;**

**2、如果发送的字节数太长, 会被拆分成多个TCP的数据包发出; 如果发送的字节数太短, 就会先在缓冲区里等待, 等到缓冲区长度差不多了, 或者其他合适的时机发送出去;**

**3、接收数据的时候, 数据也是从网卡驱动程序到达内核的接收缓冲区;**

**4、然后应用程序可以调用read从接收缓冲区拿数据;**

1. **TCP粘包问题**

* **首先要明确, 粘包问题中的 “包” , 是指的应用层的数据包；**
* **在TCP的协议头中, 没有如同UDP一样的 “报文长度” 这样的字段, 但是有一个序号这样的字段；**
* **站在传输层的角度, TCP是一个一个报文过来的，按照序号排好序放在缓冲区中；**
* **站在应用层的角度, 看到的只是一串连续的字节数据. 那么应用程序看到了这么一连串的字节数据, 就不知道从哪个部分开始到哪个部分是一个完整的应用层数据包。**

1. **协议**

**HTTP、HTTPS、SSH、Telnet、FTP、SMTP**

**小结TCP与UDP的区别：**

1. 基本连接和无连接；
2. 对系统资源的要求（TCP较多，UDP较少）
3. UDP程序结构较简单；
4. 流模式和数据报模式；
5. TCP保证数据正确性，UDP可能丢包，TCP保证数据顺序，UDP不保证。

# 6、请描述下tcp的三次握手四次挥手的过程

三次握手与四次挥手：

三次握手通俗版：

* 第一次握手：客户端要和服务端进行通信，首先要告知服务端一声，遂发出一个SYN=1的连接请求信号,”服务端哥哥，我想给你说说话”。
* 第二次握手：当服务端接收到客户端的连接请求，此时要给客户端一个确认信息，”我知道了（ACK）,我这边已经准备好了，你现在能连吗（SYN）”。
* 第三次握手：当客户端收到了服务端的确认连接信息后，要礼貌的告知一下服务端，“好的，咱们开始联通吧（ACK）”。
* 到此整个建立连接的过程已经结束，接下来就是双方你一句我一句甚至同时交流传递信息的过程了。

四次挥手断开连接通俗版：

* 第一次挥手：双方交流的差不多了，此时客户端也已经结尾了，接下来要断开通信连接，所以告诉服务端“我说完了（FIN）”，此时自身形成等待结束连接的状态。
* 第二次挥手：服务端知道客户端已经没话说了，服务端此时还有两句话要给客户端说“我知道你说完了（ACK），我再说两句&\*…%￥”…
* 第三次挥手：此时客户端洗耳恭听继续处于等待结束的状态，服务器端也说完了，自身此时处于等待关闭连接的状态，并对告诉客户端，“我说完了，咱们断了吧（FIN）”。
* 第四次挥手：客户端收知道服务端也说完了，也要告诉服务端一声（ACK），因为连接和断开要双方都按下关闭操作才能断开，客户端同时又为自己定义一个定时器，因为不知道刚才说的这句话能不能准确到达服务端（网络不稳定或者其他因素引起的网络原因）。
* 所以默认时间定为两个通信的最大时间之和，超出这个时间就默认服务器端已经接收到了自己的确认信息，此时客户端就关闭自身连接，服务器端一旦接收到客户端发来的确定通知就立刻关闭服务器端的连接。

到此为止双方整个通信过程就此终结。

这里要声明一下：断开链接不一定就是客户端，谁都可以先发起断开指令，另外客户端和服务端是没有固定标准的，谁先发起请求谁就是客户端。

* 三次握手阐述：

在第一次消息发送中，A随机选取一个序列号作为自己的初始序号发送给B；

第二次消息B使用ack对A的数据包进行确认，因为已经收到了序列号为x的数据包，准备接收序列号为x+1的包，所以ack=x+1，同时B告诉A自己的初始序列号，就是seq=y；

第三条消息A告诉B收到了B的确认消息并准备建立连接，A自己此条消息的序列号是x+1，所以seq=x+1，而ack=y+1是表示A正准备接收B序列号为y+1的数据包。

* 四次挥手阐述：

由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，

收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。

首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，上图描述的即是如此。

（1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态。

（2）第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE\_WAIT状态。

（3）第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST\_ACK状态。

（4）第四次挥手：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

# 7、网关的作用

网关的概念：

网关顾名思义就是连接两个网络的设备，区别于路由器（由于历史的原因，许多有关TCP/IP的文献曾经把网络层使用的路由器Router称为网关，在今天许多局域网采用都是路由来接入网络，因此现在通常指的网关就是路由器的IP），经常在家庭中或者小型企业网络中使用，用于连接局域网和Internet。网关也经常指把一种协议转成另一种协议的设备，比如语音网关。

在传统TCP/IP术语中，网络设备只分成两种，一种为网关（gateway），另一种为主机（host）。网关能在网络间转递数据包，但主机不能 转送数据包。在主机（又称终端系统，end system）中，数据包需经过TCP/IP四层协议处理，但是在网关（又称中介系 统，intermediate system）只需要到达网际层（Internet layer），决定路径之后就可以转送。在当时，网关 （gateway）与路由器（router）还没有区别。

在现代网络术语中，网关（gateway）与路由器（router）的定义不同。网关（gateway）能在不同协议间移动数据，而路由器（router）是在不同网络间移动数据，相当于传统所说的IP网关（IP gateway）。

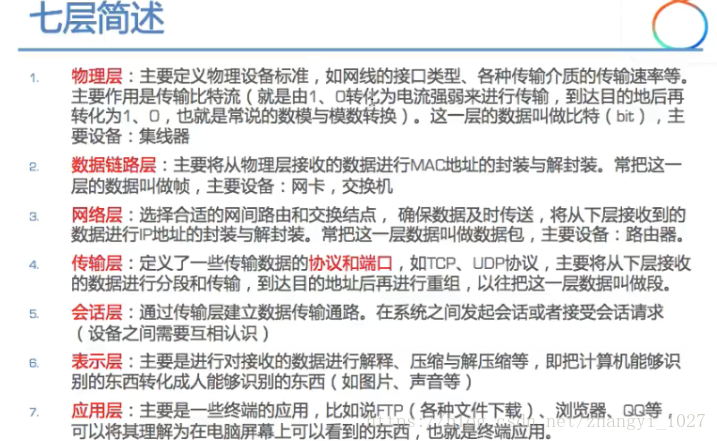
网关是连接两个网络的设备，对于语音网关来说，他可以连接PSTN网络和以太网，这就相当于VOIP，把不同电话中的模拟信号通过网关而转换成数字信号，而且加入协议再去传输。在到了接收端的时候再通过网关还原成模拟的电话信号，最后才能在电话机上听到。

对于以太网中的网关只能转发三层以上数据包，这一点和路由是一样的。而不同的是网关中并没有路由表，他只能按照预先设定的不同网段来进行转发。网关最重要的一点就是端口映射，子网内用户在外网看来只是外网的IP地址对应着不同的端口，这样看来就会保护子网内的用户。

# 8、osi七层模型每一层的作用

OSI（Open System Interconnection）模型，即开放式通信系统互联参考模型，是国际标准化组织提出的一个试图使各种计算机或者通信系统在世界范围内互联为网络的标准框架。整个模型分为七层，物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。

|  |  |  |  |
| --- | --- | --- | --- |
| OSI七层模型 | 功能 | 对应的网络协议 | TCP/IP四层概念模型 |
| 应用层 | 文件传输，文件管理，电子邮件的信息处理—apdu | HTTP、TFTP, FTP, NFS, WAIS、SMTP | 应用层 |
| 表示层 | 确保一个系统的应用层发送的消息可以被另一个系统的应用层读取，编码转换，数据解析，管理数据的解密和加密，最小单位——ppdu | Telnet, Rlogin, SNMP, Gopher |
| 会话层 | 负责在网络中的两节点建立，维持和终止通信，在一层协议中，可以解决节点连接的协调和管理问题。包括通信连接的建立，保持会话过程通信连接的畅通，两节点之间的对话，决定通信是否被终端一斤通信终端是决定从何处重新发送，最小单位——spdu | SMTP, DNS |
| 传输层 | 定义一些传输数据的协议和端口。传输协议同时进行流量控制，或是根据接收方接收数据的快慢程度，规定适当的发送速率，解决传输效率及能力的问题——tpdu | TCP, UDP | 传输层 |
| 网络层 | 控制子网的运行，如逻辑编址，分组传输，路由选择最小单位——分组（包）报文 | IP, ICMP, ARP, RARP, AKP, UUCP | 网络层 |
| 数据链路层 | 主要是对物理层传输的比特流包装，检测保证数据传输的可靠性，将物理层接收的数据进行MAC（媒体访问控制）地址的封装和解封装，也可以简单的理解为物理寻址。交换机就处在这一层，最小的传输单位——帧 | FDDI, Ethernet, Arpanet, PDN, SLIP, PPP，STP。HDLC,SDLC,帧中继 | 数据链路层 |
| 物理层 | 定义物理设备的标准，主要对物理连接方式，电气特性，机械特性等制定统一标准，传输比特流，因此最小的传输单位——位（比特流） | IEEE 802.1A, IEEE 802.2到IEEE 802. |



# 9、session是什么，cookie是什么，有什么使用场景

1、背景介绍

* + - cookie：

在程序中，会话跟踪是很重要的事情。理论上，一个用户的所有请求操作都应该属于同一个会话，而另一个用户的所有请求操作则属于另一个会话，二者不同混淆。

而Web应用程序是使用HTTP协议传输数据的。HTTP协议是无状态的协议，一旦数据交换完毕，客户端与服务器端的链接就会关闭，再次交换数据需要建立新的连接，这就意味着服务器无法从连接上跟踪会话。

怎么办呢？就给客户端们颁发一个通行证吧，每人一个，无论谁访问都必须携带自己通行证。这样服务器就能从通行证上确认客户身份了。这就是Cookie的工作原理。简单例子就是理发店的会员卡，cookie就相当于我们去理发店办的会员卡，凭借此卡，就可以打折并召唤专属Tony老师。

* + - session：

2、知识剖析

cookie的基本属性

session的基本方法

3、常见问题

Cookie和Session的联系和区别？

如果浏览器禁用cookie后怎么办？

4、解决方案

联系：

区别：

解决禁用cookie：

5、拓展思考

session和cookie的应用场景

session的应用场景：

cookie典型应用场景：

6、更多思考

# 10、session和cookie的区别

# 11、dns是什么

# 12、dhcp是什么

# 13、[TCP和UDP的区别及应用场景](https://www.cnblogs.com/liangyc/p/11628208.html)

区别

* 1. 面向连接VS无连接  
     TCP建立一个连接需要3次握手IP数据包，断开连接需要4次握手。另外断开连接时发起方可能进入TIME\_WAIT状态长达数分钟（视系统设置，windows一般为120秒），在此状态下连接（端口）无法被释放。  
     UDP不需要建立连接，可以直接发起。
  2. 可靠VS不可靠  
     TCP利用握手、ACK和重传机制，udp没有。  
     1，校验和（校验数据是否损坏）；  
     2，定时器（分组丢失则重传）；  
     3，序列号（用于检测丢失的分组和重复的分组）；  
     4，确认应答ACK（接收方告知发送方正确接收分组以及期望的下一个分组）；  
     5，否定确认（接收方通知发送方未被正确接收的分组）；  
     6，窗口和流水线（用于增加信道的吞吐量）。（窗口大小：无需等待确认应答而可以继续发送数据的最大值）
  3. 有序性  
     TCP利用seq序列号对包进行排序，udp没有。
  4. 面向字节流vs面向报文
     + 面向报文  
       面向报文的传输方式是应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文。因此，应用程序必须选择合适大小的报文。若报文太长，则IP层需要分片。UDP对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。这也就是说，应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文。（一个upd的最大报文长度2^16-1-20-8,20是ip报文头，8是udp报文头）
     + 面向字节流  
       面向字节流的话，虽然应用程序和TCP的交互是一次一个数据块（大小不等），但TCP把应用程序看成是一连串的无结构的字节流。TCP有一个缓冲，当应用程序传送的数据块太长，TCP就可以把它划分短一些再传送。如果应用程序一次只发送一个字节，TCP也可以等待积累有足够多的字节后再构成报文段发送出去。
  5. tcp有流量控制，udp没有
  6. tcp的头部比20bytes，udp8byres
* TCP应用场景：  
  效率要求相对低，但对准确性要求相对高的场景。因为传输中需要对数据确认、重发、排序等操作，相比之下效率没有UDP高。举几个例子：文件传输（准确高要求高、但是速度可以相对慢）、接受邮件、远程登录。
* UDP应用场景：  
  效率要求相对高，对准确性要求相对低的场景。举几个例子：QQ聊天、在线视频、网络语音电话（即时通讯，速度要求高，但是出现偶尔断续不是太大问题，并且此处完全不可以使用重发机制）、广播通信（广播、多播）。

# 【接口测试】

1.你认为的接口是什么  
2.接口测试工具工作的原理是什么呢？  
3.接口中包含哪些内容  
4.接口用例如何写  
5.你是如何做接口测试的，使用的什么工具？  
6.接口自动化了解过吗  
7.get和post的区别

# 【场景方面】

1.让你写登录，支付，充值，购物车，文件上传，三角形，笔，水杯等的测试点（绝对百分之80的公司如果让分析测试点的话会让写着几个，相信我！！！）

# 【其他测试问题】

1.简述转包过程，你用的抓包工具是什么  
2.如何分析一个bug是前端还是后端bug  
3.url后面跟数据的话如何拼接上去  
4.抓包的原理是什么  
5.你们公司使用的bug管理工具是什么  
6.弱网测试是如何实现的  
7.web测试和app测试的区别是什么  
8.给你一个软件你会如何进行测试，或给你一个页面你会如何进行测试  
9.兼容性测试你是如何做的（从app 和 web两方面回答）  
10.冒泡排序之类的一些代码题  
11. 说一个让你印象深刻的bug

# 【自动化测试】

1.可能会问到selenium  
2.selenium元素定位的方式，显示等待，隐式等待，强制等待的区别等  
3.接口自动化脚本的编写原理和过程  
这块可以不用担心，只要会一些简单的就可以了，因为对于初中级测试来说，自动化是一个加分项，一般如果自动化掌握的很熟练的同学薪资是在12-25k之间的

# 【会做性能测试吗】

这里大家要知道，性能测试是非常非常难的，初中级岗位一般也不会让你去做的  
所以大家就了解一下app和web端的性能指标，就说自己没有做过但是了解过就好了  
app关注：帧率，内存，cpu，耗电量，启动时间，弱网  
web关注：并发，吞吐量，响应时间，资源占用率等  
建议大家学习下app端的adb命令，monkey命令，也有可能问到

# 【hr会问的几个问题】

1.你觉得自己的优缺点是什么（这个是个大坑！不知道多少英雄豪杰技术关过了死在这关了，大家可以百度下，看看别人怎么回答的！！）  
2.你上一家公司的薪资是多少，调了几次薪，多久涨一次薪水  
3.为什么从上一家公司离职呢  
4.离职证明和背调可以吗（这个大家需要的可以联系我，可以告诉大家如何和hr沟通，免费的谢谢，互相尊重）  
5.你还有什么问题要问吗？

你认为接口是什么

给你一个页面，你会如何测试