1. TCP与UDP的区别是什么？他们位于TCP/IP协议的哪一层？各自的场景有哪些？

（1）TCP是面向连接的可靠的基于字节流的传输协议，即需要建立连接后才能发送数据，所以是点对点数据传输，是全双工通信，可能会对数据进行分组，并且有拥塞控制。首部最小20字节。

UDP面向无连接，是不可靠的，即发报文前无需建立连接，想发即发，没有拥塞控制所以发送数据是恒定的。支持一对多多对多的传输方式，首部是8个字节。

（2）它们位于传输层。

（3）TCP：适用于效率要求相对低，需要对可靠性要求高的场景，如文件传输。

UDP：适用于速度要求较高，准确性要求不苛刻的场景，如视频通信，语音等。

1. 请描述TCP协议建立连接与断开连接的交互过程？
2. 建立连接：

第一次握手：客户端向服务器端发送请求，请求建立连接，此时SYN=1,序列号seq=x，客户端进入连接请求发送状态。

第二次握手：服务器端收到客户端发送过来的请求，确认自己可以建立连接，对连接请求进行回复，SYN=1，ACK=1，序列号seq=y；确认号ack=x+1；同时从监听状态变为请求接受状态。

第三次握手：客户端收到服务器端的确认信息可以建立连接，这是发送一个确认报文，其中ACK=1,序列号seq=x+1，ack=y+1。状态变为连接建立，这时可以发送数据给服务器获得服务

服务器在接收到第三次握手发来的确认报文后进入连接建立状态，进行数据传输。

1. 断开连接：

第一次挥手：假设客户机请求完资源了，想要释放连接。首先，客户机的应用进程先向服务器发出连接释放报文段，该报文段中将首部的终止控制位FIN置为１，序号seq=u。此时客户机进入到终止等待1状态，等待服务器的确认。

第二次挥手：服务器收到连接释放报文后发出确认，在发送报文中将首部中的ACK置为1，并且产生序号v，发出确认号为u+1。此时服务器就进入关闭等待状态，客户机进入终止等待2状态。

第三次挥手：服务器发送完数据后，此时可以断开连接，还是有服务器向客户机发送报文，该报文段中将首部的终止控制位FIN和ACK置为1，序号为w，并且发送确认号为u+1。此时服务器就进入了最后确认状态。

客户机收到服务器的连接释放报文后，对此报文进行确认。在该报文段中将ACK置为1，确认号为w+1，产生序号为u+1。

1. 在发送一个https 请求中，获得Connect Timeout 网络错误，是在哪个通讯过程出现了问题？

客户端与服务器建立连接时超时。

1. MySQL中有一个学生成绩表(姓名, 班级, 科目, 分数)， 写一个SQL，查询每个班级的平均分？打印平均分 > 80分的班级？写一个SQL，查询每个班级中每个科目前三名的学生?
2. SELECT 班级 , AVG(分数) as 平均分 From 学生成绩表 Group By 班级;
3. SELECT 班级 , AVG(分数) as 平均分 From 学生成绩表 Group By 班级 HAVING AVG(分数) > 80;
4. SELECT \* FROM ( SELECT 姓名, 班级, 科目, 分数, ROW\_NUMBER() OVER (PARTITION BY 班级, 科目 ORDER BY 分数 DESC) as 排名FROM 学生成绩表) t

WHERE t.排名 <= 3;

1. 给定一个只包含1-100数字的整数数组，数组长度为N(1 < N<2^31)，请打印1-100中缺少的整数数字。(用代码实现)

func MisNum (arr []int,N int) {

flag := make([]bool, 100)

for i := 0; i < 100; i++ {

flag[i] = false

}

// 遍历数组，将数字在布尔数组中标记为true

for i := 0; i <N; i++ {

flag[arr[i]-1] = true

}

// 遍历布尔数组，打印缺失的数字

for i := 0; i < 100; i++ {

if !flag[i] {

fmt.Printf("%d ", i+1)

}

}

}

1. 解释进程和线程的区别，并讨论它们之间的通信机制。

进程是资源分配的基本单位，而线程是执行调度的基本单位。

进程是指正在运行的程序的实例。每个进程都有自己的内存空间、文件句柄、网络连接等资源。进程之间相互独立，可以通过进程间通信来进行数据交换和协作。

线程是进程中的执行单元。一个进程可以包含多个线程，这些线程共享进程的内存空间和资源。线程之间的切换比进程之间的切换更加轻量级，因此多线程的程序可以更高效地利用CPU资源。

线程间通信的机制包括共享内存、消息传递、信号量、互斥锁等。其中，共享内存是最快的通信方式，但需要进行同步，以避免多个线程同时访问同一块共享内存。消息传递是另一种常见的通信方式，它可以通过队列、管道等方式实现。信号量和互斥锁则是用于控制对共享资源的访问，避免多个线程同时访问导致数据不一致问题。

1. 解释栈和队列的概念，并给出它们的应用场景。

（1）栈是一种后进先出的数据结构，最后放入栈中的元素最先被取出。向一个栈中添加元素称为“入栈”，取出元素称为“出栈”。栈的基本操作是压栈、查看栈顶元素和判断栈是否为空。

队列是一种先进先出的数据结构，最先放入队列中的元素最先被取出。向一个队列中添加元素称为“入队”，取出元素称为“出队”。队列还有基本操作是查看队头元素和判断队列是否为空。

（2）应用场景：

栈：程序调用堆栈、表达式求值、撤销操作、递归算法实现、括号匹配问题

队列：任务调度、消息传递、缓存处理、BFS、广度优先搜索等。

1. 解释面向对象编程的概念和特点，以及封装、继承和多态的含义。

对同类对象抽象出其共性，形成类。类中的大多数数据，只能用本类的方法进行处理。类通过一个简单的外部接口与外界发生关系，对象与对象之间通过消息进行通信。

封装指的是将对象的数据和行为封装在一起，并对外部隐藏实现细节。封装有助于提高代码的可维护性、可复用性和安全性。

继承是实现代码复用和扩展的机制。通过继承，子类可以继承父类的属性和方法，并且可以在此基础上添加新的属性和方法。继承有助于减少代码的重复，提高代码的复用性和可维护性。

多态可以实现通用操作，可以让不同类型的对象都可以被当做同一类型的对象来使用，从而提高代码的灵活性和可扩展性。多态有两种实现方式，分别是静态多态（函数重载、运算符重载）和动态多态（虚函数、接口）。

1. 什么是异常/错误处理？处理异常/错误的机制有哪些？

（1）异常/错误处理是指在程序运行过程中遇到意外情况时，采取一定的措施来使程序能够恢复正常运行或者结束运行。通过处理错误可以使程序更加健壮和可靠。

（2）使用if语句结合函数返回值来实现。函数返回一个错误，调用者可以根据返回值中的错误信息来判断函数是否执行成功。通常情况下，如果函数执行成功，返回值中的错误值为nil；如果函数执行失败，则返回一个非nil的错误值。

panic-recover机制，用于处理严重错误。当程序遇到无法恢复的错误时，可以使用panic函数来抛出一个panic异常，然后使用recover函数来捕获这个异常并进行处理。

1. 以下代码运行结果是什么？

package main

import "fmt"

func main() {

fmt.Println("start")

defer fmt.Println(1)

defer fmt.Println(2)

defer fmt.Println(3)

fmt.Println("end")

}

输出的结果是什么？

结果：

start

end

3

2

1