面经

单元测试：

**单元测试：**　　单元测试是对软件基本组成单元（软件设计的最小单位）进行正确性检验的测试工作，如函数、过程(function,procedure)或一个类的方法(method)。

**集成测试：**　　集成测试是在单元测试的基础上，将所有模块按照概要设计要求组装成为子系统或系统，验证组装后功能以及模块间接口是否正确的测试工作。集成测试也叫组装测试、联合测试、子系统测试或部件测试。

**系统测试：**　　系统测试是将经过集成测试的软件，作为计算机系统的一个部分，与系统中其他部分结合起来，在实际运行环境下对计算机系统进行的一系列严格有效地测试，以发现软件潜在的问题，保证系统的正常运行。

**const:**

几乎面C++都会问的问题。

const作用

1、用于定义常量，编译器可以对其进行数据静态类型安全检查

2、修饰函数形式参数，将“值传递”改为“const &传递”可以提高效率

3、修饰函数的返回值，返回值不能被直接修改

4、修饰类的成员函数，任何不会修改数据成员的函数都应该用const来修饰

**1.const声明的变量只能被读**

**2.必须初始化**

**3.常量和指针：**   
常量和指针常量：

int j=0;

const int i=0; //i是常量，i的值不会被修改

const int \*p1=&i; //指针p1所指内容是常量，可以不初始化

int \* const p2=&j //指针p2是常量，所指内容可修改

const int \* const p3i=&i; //指针p3是常量，所指内容也是常量

p1=&j //合法

\*p2=100; //合法

**4.const与define**  
const 常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查。而对后者只进行字符替换，没有类型安全检查，并且在字符替换可能会产生意料不到的错误。

**程序的内存分配**

int a = 0; //全局初始化区

char \*p1;//全局未初始化区

void main(void)

{

  int b; //栈

  char s[] = "abc"; //栈

  char \*p2; //栈

  char \*p3 = "123456"; //123456\0在常量区，p3在栈上

  static int c =0; //全局（静态）初始化区

  p1 = (char \*)malloc(10); //堆

  p2 = (char \*)malloc(20); //堆

}

**栈区（stack）**— 由编译器自动分配释放，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。

（编译器自动分配释放，连续，高地址向低地址，空间限制，分配速度快，函数调用中先入下一条语句位置，从右到左的参数，局部变量）

**堆区（heap）** — 一般由程序员分配释放，若程序员不释放，程序结束时可能由OS回收 。注意它与数据结构中的堆是两回事，分配方式倒是类似于链表。

（不连续空间，速度慢，有碎片，向高地址扩展，比较灵活，大）

**全局区（静态区**）（static）—，全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域， 未初始化的全局变量、未初始化的静态变量在相邻的另一块区域。 - 程序结束后有系统释放

**文字常量区**—常量字符串就是放在这里的。程序结束后由系统释放

**程序代码区**—存放函数体的二进制代码。

**程序段(Text):程序代码在内存中的映射，存放函数体的二进制代码。**

**初始化过的数据(Data):在程序运行初已经对变量进行初始化的数据。**

**未初始化过的数据(BSS):在程序运行初未对变量进行初始化的数据。**

**栈 (Stack):存储局部、临时变量，函数调用时，存储函数的返回指针，用于控制函数的调用和返回。在程序块开始时自动分配内存,结束时自动释放内存，其操作方式类似于数据结构中的栈。**

**堆 (Heap):存储动态内存分配,需要程序员手工分配,手工释放.注意它与数据结构中的堆是两回事，分配方式类似于链表。**

**注：1.Text, BSS, Data段在编译时已经决定了进程将占用多少VM**

**可以通过size，知道这些信息：**

**2. 正常情况下，Linux进程不能对用来存放程序代码的内存区域执行写操作，即程序代码是以只读的方式加载到内存中，但它可以被多个进程安全的共享。**

**static**

**修饰全局变量**

**存储方式：**

全局变量(外部变量)的说明之前再冠以static 就构成了静态的全局变量。

全局变量本身就是静态存储方式， 静态全局变量当然也是静态存储方式。 这两者在存储方式上并无不同。

**作用范围**：

普通的全局变量的作用域是整个源程序， 当一个源程序由多个源文件组成时，普通的（也就是非静态）的全局变量在各个源文件中都是有效的。

而静态全局变量则限制了其作用域， 即只在定义该变量的源文件内有效， 在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用， 因此可以避免在其它源文件中引起错误。

**修饰局部变量**

生存周期 —变量从定义到销毁的时间范围

普通局部变量存储在栈中,数据则随着函数等的作用域结束导致出栈而销毁。

静态局部变量存放在静态区,所以它的生命周期和全局变量相同是从程序开始到程序结束。

**初始化**

静态局部变量只能被初始化一次.

修饰函数

作用范围：

static global函数将其作用域限制在本源文件之内，对其它源文件不可见。而global函数具有全局可见性，对其它源文件则可见。

定义静态函数的好处： 静态函数不能被其它文件所用； 其它文件中可以定义相同名字的函数，不会发生冲突；

**把局部变量改变为静态变量后，是改变了它的存储方式，即改变了它的生存期。**

**把全局变量改变为静态变量后，是改变了它的作用域， 限制了它的使用范围。**

**关键字volatile**

一个定义为 volatile 的变量是说这变量可能会被意想不到地改变， 这样， 编译器就不会去假设这个变量的值了。 精确地说， 就是优化器在用到这个变最时必须每次都小心地重新读取这个变量的值， 而不是使用保存在寄存器里的备份。

**const和define的区别**

define只是用来做文本替换的

在编译的时候，就将define的所有内容进行替换了，它的生命周期止于编译期，它存在于程序的代码段，在实际程序中，它只是一个命令中的参数或者是常数， 并没有实际的存在。

const存在于程序的数据段，并且在堆栈分配了空间，他在程序中可以被调用、传递，const常量有数据类型，而宏常量没有数据类型，编译器可以对const常量进行检查。

**const, volatile区别**

（1）const含义是“请做为常量使用”，而并非“放心吧，那肯定是个常量”，是不可修改的只读变量。

         volatile的含义是“请不要做自以为是的优化，这个值可能变掉的”，而并非“你可以修改这个值”。

（2）const只在编译期有用，在运行期无用

         const在编译期保证在C的“源代码”里面，没有对其修饰的变量进行修改的地方（如有则报错，编译不通过），而运行期该变量的值是否被改变则不受const的限制。

         volatile在编译期和运行期都有用

         在编译期告诉编译器：请不要做自以为是的优化，这个变量的值可能会变掉；

         在运行期：每次用到该变量的值，都直接从内存中取该变量的值。

**extern "C"**

**extern "C"的主要作用就是为了能够正确实现C++代码调用其他C语言代码。加上extern "C"后，会指示编译器这部分代码按C语言（而不是C++）的方式进行编译。由于C++支持函数重载，因此编译器编译函数的过程中会将函数的参数类型也加到编译后的代码中，而不仅仅是函数名；而C语言并不支持函数重载，因此编译C语言代码的函数时不会带上函数的参数类型，一般只包括函数名。**

**strcpy使用可能有什么缺陷（**我说了源字符串中间如果有'\0',得到的目的字符串就是不完整的。面试官问还有什么其他缺陷，没答出来。答案是：strcpy(des,src) 如果des长度小于src会出现数组越界溢出）

**strcpy和memcpy的区别**

1、复制的内容不同。strcpy只能复制字符串， 而memcpy可以复制任意内容， 例如字

符数组、 整型、 结构体、 类等。

2、复制的方法不同。strcpy不需要指定长度， 它是遇到字符串结束符'\0'而结束的。 memcpy则是根据其第三个参数决定复制的长度。

3、用途不同。通常在复制字符串时用strcpy; 而若需要复制其他类型数据，一般用memcpy

**指针和引用的区别是？**

本质（汇编指令）：没有区别的

指针不需要初始化，引用需要初始化

指针有一级二级指针，引用只有左值右值引用

**inline函数**

**new和malloc的区别**

1、new是从自由存储区上为对象动态分配内存，而malloc函数是从堆上动态分配内存

那么自由存储区是否能够是堆（问题等价于new是否能在堆上动态分配内存），这取决于operator new 的实现细节。自由存储区不仅可以是堆，还可以是静态存储区，这都看operator new在哪里为对象分配内存。

2、返回类型安全性

new操作符内存分配成功时，返回的是对象类型的指针，类型严格与对象匹配，无须进行类型转换，故new是符合类型安全性的操作符。而malloc内存分配成功则是返回void \* ，需要通过强制类型转换将void\*指针转换成我们需要的类型。

3、内存失败的返回值

new内存分配失败时，会抛出bac\_alloc异常，它不会返回NULL；malloc分配内存失败时返回NULL。

4、是否需要指定内存大小

使用new操作符申请内存分配时无须指定内存块的大小，编译器会根据类型信息自行计算，而malloc则需要显式地指出所需内存的尺寸。

5、是否调用构造函数/析构函数

new/delete会调用对象的构造函数/析构函数以完成对象的构造/析构。而malloc则不会

6.new与malloc是否可以相互调用

operator new /operator delete的实现可以基于malloc，而malloc的实现不可以去调用new。

**14 空指针和悬垂指针的区别**

答：空指针是指被赋值为NULL的指针；delete指向动态分配对象的指针将会产生悬垂指针。

（1）空指针可以被多次delete，而悬垂指针再次删除时程序会变得非常不稳定。

（2）使用空指针和悬垂指针都是非法的，而且有可能造成程序崩溃，如果指针是空指针，尽管同样是崩溃，但和悬垂指针相比是一种可预料的崩溃。

**15 虚函数如何实现多态**

答：基类的指针指向子类的对象，子类对象重写了基类中的虚函数，那么在用基类指针操作该函数时就会根据其指向对象的类型（基类还是子类）分别调用不同的函数。

**#Linux 线程同步的三种方法**

线程的最大特点是资源的共享性，但资源共享中的同步问题是多线程编程的难点。linux下提供了多种方式来处理线程同步，最常用的是互斥锁、条件变量和信号量。

<https://blog.csdn.net/qq_17308321/article/details/79929623>

**智能指针**

<https://sap-pill.github.io/2020/02/15/%E6%99%BA%E8%83%BD%E6%8C%87%E9%92%88/>

**TCP的服务端和客户端的代码编写流程**

<https://sap-pill.github.io/2020/02/08/TCP%E6%9C%8D%E5%8A%A1%E7%AB%AF%E4%BB%A3%E7%A0%81%E7%BC%96%E5%86%99/>

**UDP的服务端和客户端的代码编写流程**

<https://sap-pill.github.io/2020/02/11/UDP%E6%9C%8D%E5%8A%A1%E5%99%A8%E5%92%8C%E5%AE%A2%E6%88%B7%E7%AB%AF%E7%9A%84%E4%BB%A3%E7%A0%81%E7%BC%96%E5%86%99%E5%8F%8A%E6%B5%8B%E8%AF%95/>

**构造函数和析构函数是否可以被重载**

构造函数可以被重载，因为构造函数可以有多个且可以带参数。

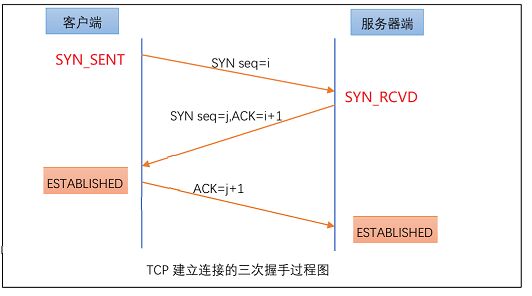
析构函数不可以被重载，因为析构函数只能有一个，且不能带参数。

**三次握手**

1、客户端先发起建立连接的请求，给服务端发一个SYN报文段，并且携带seq编号SYN seq=I;(SYN\_SENT阶段) SYN报文段（同步报文段）

2、服务器收到后，回馈一个SYN+ACK，这个ACK是上次的seq的序号值+1，相当于对第一次SYN收到的确认，SYNj,ACK=i+1（SYN\_RCVD阶段）

3、客户端在再向服务器发送一个ACK，告诉服务端说明自己收到了你发的SYN，然后连接就建立起来了。ACK=j+1;（连接已经建立的阶段）



**四次挥手**

（FIN\_WAIT\_1状态）

1、主动方发送一个FIN seq=i报文段

（CLOSE\_WAIT状态）

2、被动方回馈一个ACK=i+1

（FIN\_WAIT\_2状态）

主动方没有数据可以发了，这个时候断开了主动到被动的写通道，

但是被动方自己的数据没有发完，所以要过段时间。（如果这段时间没有，2-3过程合为一体，那就是三次挥手）

（LAST\_ACK状态）

3、过段时间，被动方发送一个FIN seq=j，意思自己可以断开连接了（LAST\_ACK状态）

（TIME\_WAIT状态->CLOSED）

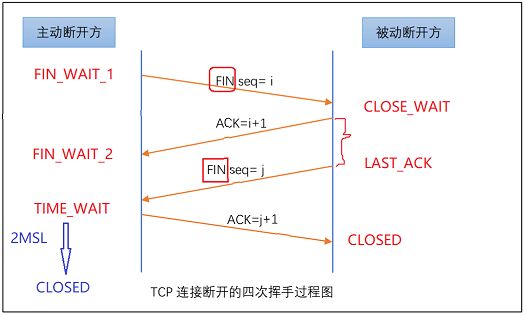
TIME\_WAIT存在的意义：

1、处理迟来的数据报，直接将其丢弃，防止对后续启动进程造成影响

2、可靠地终止TCP连接，防止最后发送的ACK被丢弃

4、主动方收到后，给被动方发送一个ACK=j+1

（CLOSED状态）



**TCP与UDP区别总结：**

1、TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接

2、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保 证可靠交付

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首部开销20字节;UDP的首部开销小，只有8个字节

6、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

**滑动窗口为0时应该怎么处理**

发送窗口探寻

在定时器结束前窗口大小>0，那就继续发送数据

在定时器结束后窗口还为0，一般有三种操作

1、发送一个byte合法数据（滑动窗口内）或者非法数据（滑动窗口外）

2、刷新定时器

3、记录超时次数

然后如果超时次数达到极限，那就**关闭连接**或者**通知应用层出错**

**超时重传**

原理是在发送某一个数据以后就开启一个计时器，在一定时间内如果没有得到发送的数据报的ACK报文，那么就重新发送数据，直到发送成功为止。

　　影响超时重传机制协议效率的一个关键参数是重传超时时间（RTO，Retransmission TimeOut）。RTO的值被设置过大过小都会对协议造成不利影响。

　　（1）RTO设长了，重发就慢，没有效率，性能差。

　　（2）RTO设短了，重发的就快，会增加网络拥塞，导致更多的超时，更多的超时导致更多的重发。

连接往返时间（RTT，Round Trip Time），指发送端从发送TCP包开始到接收它的立即响应所消耗的时间。

**linux命令**

echo 命令用于在终端输出字符串或变量提取后的值，格式为“echo [字符串 | $变量]”。

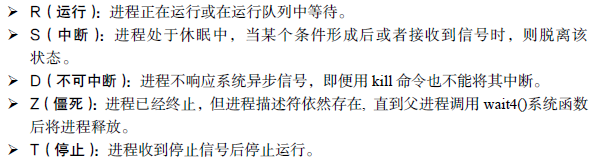
date 命令用于显示及设置系统的时间或日期，格式为“date [选项] [+指定的格式]”。

reboot 命令用于重启系统

poweroff 命令用于关闭系统

wget 命令用于在终端中下载网络文件

ps 命令查看系统中的进程状态



top 动态监视进程活动与系统负载等信息（相当于强化版的windows任务管理器）

pidof 查询某个额指定服务进程的pid值

kill 命令用于终止某个指定PID 的服务进程，

killall 命令用于终止某个指定名称的服务所对应的全部进程

ifconfig 命令用于获取网卡配置与网络状态等信息

uname 命令用于查看系统内核与系统版本等信息

uptime 用于查看系统的负载信息

free 用于显示当前系统中内存的使用量信息

who 用于查看当前登入主机的用户终端信息，格式为“who [参数]”。

last 命令用于查看所有系统的登录记录，格式为“last [参数]”。

history 命令用于显示历史执行过的命令，格式为“history [-c]”。

sosreport 命令用于收集系统配置及架构信息并输出诊断文档，格式为sosreport。

cat 命令用于查看纯文本文件（内容较少的），格式为“cat [选项] [文件]”。

more 命令用于查看纯文本文件（内容较多的），格式为“more [选项]文件”。

head 命令用于查看纯文本文档的前N 行，格式为“head [选项] [文件]”。

tail 命令用于查看纯文本文档的后N 行或持续刷新内容，格式为“tail [选项] [文件]”

tr 命令用于替换文本文件中的字符，格式为“tr [原始字符] [目标字符]”。

wc 命令用于统计指定文本的行数、字数、字节数，格式为“wc [参数] 文本”。

stat 命令用于查看文件的具体存储信息和时间等信息，

cut 命令用于按“列”提取文本字符，格式为“cut [参数] 文本”。

diff 命令用于比较多个文本文件的差异，格式为“diff [参数] 文件”。

touch 命令用于创建空白文件或设置文件的时间，格式为“touch [选项] [文件]”。

mkdir 命令用于创建空白的目录，格式为“mkdir [选项] 目录”。

cp 命令用于复制文件或目录，格式为“cp [选项] 源文件目标文件”。

mv 命令用于剪切文件或将文件重命名，格式为“mv [选项] 源文件 [目标路径|目标文件名]”。

rm 命令用于删除文件或目录，格式为“rm [选项] 文件”。

dd 命令用于按照指定大小和个数的数据块来复制文件或转换文件，格式为“dd [参数]”。

file 命令用于查看文件的类型，格式为“file 文件名”

tar 命令用于对文件进行打包压缩或解压，格式为“tar [选项] [文件]”。

grep 命令用于在文本中执行关键词搜索，并显示匹配的结果，格式为“grep [选项] [文件]”。

lsof -i:端口号 查看某个端口是否被占用