目錄

主要列举一些比较常见的问题，答案仅做为参考。

在使用的过程中，如发现有不合理的地方，请和我(nshuling@163.com)联系。

在 C 语言中，关键字 static 有三个明显的作用：

1. 在函数体，一个被声明为静态的变量在这一函数被调用过程中维 持其值不变。
2. 在模块内（但在函数体外），一个被声明为静态的变量可以被模 块内所用函数访问，但不能被模块外其它函数访问。它是一个本地的 全局变量。
3. 在模块内，一个被声明为静态的函数只可被这一模块内的其它函 数调用。那就是，这个函数被限制在声明它的模块的本地范围内使用。 大多数应试者能正确回答第一部分，一部分能正确回答第二部分，同 是很少的人能懂得第三部分。这是一个应试者的严重的缺点，因为他 显然不懂得本地化数据和代码范围的好处和重要性。

“引用”与指针的区别是：

1）引用必须被初始化，指针不必。

2）引用初始化以后不能被改变，指针可以改变所指的对象。

3）不存在指向空值的引用，但是存在指向空值的指针。 指针通过某个指针变量指向一个

对象后， 对它所指向的变量间接操作。 程序中使用指针，程序的可读性差；而引用本身就是目标

变量的别名， 对引用的操作就是对目标变量的操作。 流操作符<<和>>、赋值操作符=的返回值、

拷贝构造函数的参数、赋值操作符=的参数、其它 情况都推荐使用引用。

防止该头文件被重复引用。

前者是从 Standard Library 的路径寻找和引用 file.h，而后者是从当前工作路径搜寻并引用 file.h。

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。 这两者的区别虽在于非静态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， 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。

从以上分析可以看出， 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。

static 函数与普通函数作用域不同。 仅在本文件。 只在当前源文件中 使用的函数应该说明 为内部函数(static)，内部函数应该在当前源文件 中说明和定义。对于可在当前源文件以外使 用的函数，应该在一个头 文件中说明，要使用这些函数的源文件要包含这个头文件。

1） static 全局变量与普通的全局变量有什么区别： static 全局变量只初使化一次，防止在 其他文件单元中被引用;

2） static 局部变量和普通局部变量有什么区别： static 局部变量只被初始化一次，下一次 依据上一次结果值；

3） static 函数与普通函数有什么区别： static 函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝

Const 作用：定义常量、修饰函数参数、修饰函数返回值三个作 用。被 Const 修饰的东西都受到强制保护，可以预防意外的变动，能 提高程序的健壮性。

1）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。

2） 有些集成化的调试工具可以对 const 常量进行调试，但是 不能对宏常量进行调试

预编译又称为预处理,是做些代码文本的替换工作。处理#开头的 指令,比如拷贝

#include 包含的文件代码， #define 宏定义的替换,条件 编译等， 就是为编译做的预备工作的

阶 段， 主要处理#开始的预编译 指令， 预编译指令指示了在程序正式编译前就由编译器进行的

操 作， 可以放在程序中的任何位置。

c 编译系统在对程序进行通常的编译之前，先进行预处理。 c 提供的 预处理功能主要有

以下三种： 1）宏定义， 2）文件包含 ， 3）条件编译。

1）总是使用不经常改动的大型代码体。

2）程序由多个模块组成，所有模块都使用一组标准的包含文件和相 同的编译选项。在这5

种情况下，可以将所有包含文件预编译为一个预编译头。

```
if ( !a ) or if(a) //BOOL
```

```
if ( a == 0) //int
```

```
const EXPRESSION EXP = 0.000001
```

```
if ( a < EXP && a >-EXP) //float7
```

```
if ( a != NULL) or if(a == NULL) //pointer
```

```
#ifdef cplusplus
    cout<<"c++";
#else
    cout<<"c";
#endif
```

设 2 个栈为 A， B，一开始均为空。

入队：将新元素 push 入栈 A；

出队： (1)判断栈 B 是否为空；

(2)如果不为空，则将栈 A 中所有元素依次 pop 出并 push 到栈 B；

(3)将栈 B 的栈顶 元素 pop 出；

这样实现的队列入队和出队的平摊复杂度都还是 O(1)。

把数据结构和对数据结构进行操作的方法封装形成一个个的对象。

把一些具有共性的对象归类后形成一个集合，也就是所谓的类。

对象都具有的特征是：静态特征和动态特征。静态特征是指能描述对象的一些属性（成员变量），动态特征是指对象表现出来的行为（成员函数）

默认构造函数，拷贝构造函数，析构函数，赋值函数。

1.当类的一个对象去初始化该类的另一个对象时；

2.如果函数的形参是类的对象，调用函数进行形参和实参结合时；

3.如果函数的返回值是类对象，函数调用完成返回时。

构造函数是类的一种特殊成员函数， 一般情况下， 它是专门用来初始化对象成员变 量的。

构造函数的名字必须与类名相同，它不具有任何类型，不返回任何值。

当构造函数涉及到动态存储分配空间时，要自己写拷贝构造函数，并且要深拷贝。

类的成员是常量成员初始化；类的成员是对象成员初始化，而该对象没有无参构造函数。

1. 运算符重载的意义？

为了对用户自定义数据类型的数据的操作与内定义的数据类型的数据的操作形式一致。

2. 不允许重载的运算符有那5个？

1. .*（成员指针访问运算符号）

2.：： 域运算符

3.Sizeof 长度运算符号

4.？：条件运算符号

1. .（成员访问符）

3. 运算符重载的三种方式？

普通函数，友元函数，类成员函数。

4. 流运算符为什么不能通过类的成员函数重载？一般怎么解决？

因为通过类的成员函数重载必须是运算符的第一个是自己， 而对流运算的重载要求第一个参数是流对象。所以一般通过友元来解决。

5. 赋值运算符和拷贝构造函数的区别与联系？

相同点：都是将一个对象 copy 到另一个中去。

不同点：拷贝构造函数涉及到要新建立一个对象。

6.

(1) #define 是 C 语法中定义符号变量的方法， 符号常量只是用来表达一个值， 在编译阶段符号就被值替换了，它没有类型；

(2) Const 是 C++语法中定义常变量的方法，常变量具有变量特性，它具有类型， 内存中存在以它命名的存储单元，可以用 sizeof 测出长度。

（1） 函数体内 static 变量的作用范围为该函数体， 不同于 auto 变量， 该变量的内存只被分配一次，因此其值在下次调用时仍维持上次的值；

（2） 在模块内的 static 全局变量可以被模块内所用函数访问， 但不能被模块外其它 函数访问；

（3） 在模块内的 static 函数只可被这一模块内的其它函数调用， 这个函数的使用范 围被限制在声明它的模块内；

（4）在类中的 static 成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；9

（5） 在类中的 static 成员函数属于整个类所拥有， 这个函数不接收 this 指针， 因而 只能访问类的 static 成员变量。

Extern “C” 是由Ｃ＋＋提供的一个连接交换指定符号，用于告诉Ｃ＋＋这段代码是Ｃ函数。这是因为 C++编译后库中函数名会变得很长，与 C 生成的不一致，造成Ｃ＋＋不能直接调用 C 函数，加上 extren “c” 后， C++就能直接调用 C 函数了。

Extern “C” 主要使用正规 DLL 函数的引用和导出 和 在 C++包含 C 函数或 C 头文件时 使用。使用时在前面加上 extern “c” 关键字即可。可以用一句话概括 extern “C” 这个声明的眞实目的：实现 C++与 C 及其它语言的混合编程。

分配方式有三种，

1、 静态存储区，是在程序编译时就已经分配好的，在整个运行期间都存在，如全 局变量、常量。

2、 栈上分配，函数内的局部变量就是从这分配的，但分配的内存容易有限。

3、 堆上分配，也称动态分配，如我们用 new,malloc 分配内存，用 delete,free 来 释放的内存。

delete 和delete[ ]的区别？

delete 只会调用一次析构函数，而 delete[]会调用每一个成员的析构函数。

Linux中比较常见的问题。

滑动窗口(Sliding Window)是一种流量控制技术。可以改善吞吐量，控制传输。其11征是发送方在接收应答之前传送附加包， 接收方告诉发送方某一时刻能送包的数量(即窗口尺寸)。作用是，如果发送接收端速度不一致时，保证通信速度，不至丢失数据。

NAT映射，功能是把公网的地址转翻译成私有地址。 通过路由器维护的NAT映射表和端口来完成私有IP与公网IP的转换

打洞，路由器的NAT机制决定了内网访问外网容易,而外网访问内网困难,两台私有IP主机之间，借助第三方服务器，完成直接数据传递，双向传递数据，说明打洞成功。

物理层：主要定义物理设备标准

数据链路层：让格式化数据以帧为单位进行传输，错误检查和纠正

网络层：为不同地理位置的网络主机提供连接和路径选择。

传输层：定义了一些传输数据的协议和端口号

会话层：通过端口建立数据传输的通路。

表示层：确保一个系统的应用层所发送的信息可被另一个系统的应用层读取。

应用层：为用户的应用程序提供网络服务。

四层：应用层、传输层、网络层、链路层。

TCP的半关闭是指TCP连接的一端调用shutdown操作，使数据只能往一个方向流动，只有一方发送了FIN，仍然可以正常收（或发）数据。

对于UDP来说，传输数据之前不需要先建立连接，远程主机在接收到UDP报文后，不需要确认。双方通讯的时候，没有应答和重传过程。

优点：开销小，数据传输速度快，实时性强。缺点：传输正确率、传输顺序、流量得不到保证。用于视频会议、电话会议等对实时性要求较高的通信场合。

TCP提供面向连接的服务，传输数据前必须先建立连接，接收方确认，数据传输结束号要释放连接。而在通讯时，是有应答和重传过程的。

优点：、缺点。与UDP相反。用于，对数据正确率要求较严格场合。

TCP---传输控制协议,提供的是面向连接、可靠的字节流服务。当客户和服务器彼此交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。TCP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---用户数据报协议，是一个简单的面向数据报的运输层协议。UDP不提供可靠性，它只是把应用程序传给IP层的数据报发送出去，但是并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，故而传输速度很快

广播，使用UDP协议，借助广播地址，向局域网中所有主机发送数据包。

组播，通过分配组播地址，将终端分组。对各个组播组同一发送数据包。组播，也叫多播。

TCP粘包是指发送方发送的若干包数据到接收方接收时粘成一包，从接收缓冲区看，后一包数据的头紧接着前一包数据的尾。

为了避免粘包现象，可采取以下几种措施。

一是对于发送方引起的粘包现象，用户可通过编程设置来避免，TCP提供了强制数据立即传送的操作指令push，TCP软件收到该操作指令后，就立即将本段数据发送出去，而不必等待发送缓冲区满；

二是对于接收方引起的粘包，则可通过优化程序设计、精简接收进程工作量、提高接收进程优先级等措施，使其及时接收数据，从而尽量避免出现粘包现象；

三是由接收方控制，将一包数据按结构字段，人为控制分多次接收，然后合并，通过这种手段来避免粘包。

（1）select，poll实现需要自己不断轮询所有fd集合，直到设备就绪，期间可能要睡眠和唤醒多次交替。而epoll其实也需要调用epoll_wait不断轮询就绪链表，期间也可能多次睡眠和唤醒交替，但是它是设备就绪时，调用回调函数，把就绪fd放入就绪链表中，并唤醒在epoll_wait中进入睡眠的进程。虽然都要睡眠和交替，但是select和poll在“醒着”的时候要遍历整个fd集合，而epoll在“醒着”的时候只要判断一下就绪链表是否为空就行了，这节省了大量的CPU时间。这就是回调机制带来的性能提升。

（2）select，poll每次调用都要把fd集合从用户态往内核态拷贝一次，并且要把current往设备等待队列中挂一次，而epoll只要一次拷贝，而且把current往等待队列上挂也只挂一次（在epoll_wait的开始，注意这里的等待队列并不是设备等待队列，只是一个epoll内部定义的等待队列）。这也能节省不少的开销。

管道pipe: 编程简单。

fifo：可以用于非血缘关系进程间通信

mmap共享内存：可以用于非血缘关系进程间通信

信号：开销小

套接字：稳定性好。

当进程调用一个阻塞的系统函数时，该进程被置于睡眠（Sleep）状态，这时内核调度其它进程运行，直到该进程等待的事件发生了（比如网络上接收到数据包，或者调用sleep指定的睡眠时间到了）它才有可能继续运行。与睡眠状态相对的是运行（Running）状态。

非阻塞则不会造成进程挂起。通信双方常常处于异步通信状态。

特征：内核使用环形队列机制，借助内核缓冲区实现。其本质是一个伪文件，由两个文件描述符引用，一个表示读端，一个表示写端，数据从管道的写端流入管道，从读端流出。

① 数据自己读不能自己写。

② 数据一旦被读走，便不在管道中存在，不可反复读取。

③ 由于管道采用半双工通信方式。因此，数据只能在一个方向上流动

④ 只能在有公共祖先的进程间使用管道。

优点：简单，相比信号，套接字实现进程间通信，简单很多。

缺点：1. 只能单向通信，双向通信需建立两个管道。

> 2．只能用于父子、兄弟进程\(有共同祖先\)间通信。

初始态、就绪态、运行态、挂起态、终止态。

孤儿进程：父进程先于子进程结束，则子进程成为孤儿进程，子进程的父进程成为init进程，称为init进程领养孤儿进程。

僵尸进程：进程终止，父进程尚未回收，子进程残留资源（PCB）存放于内核中，变成僵尸（Zombie）进程。

守护进程：是Linux中的后台服务进程，通常独立于控制终端并且周期性地执行某种任务或等待处理某些发生的事件。一般采用以d结尾的名字。没有控制终端，不能直接和用户交互。不受用户登录、注销的影响，一直在运行着，

环境变量，是指在操作系统中用来指定操作系统运行环境的一些参数。用于设置、修改、描述当前进程的运行环境信息。

互斥锁也叫互斥量，是Linux系统提供的，用于资源同步的锁机制。取值有2种。加锁和解锁。规定，对共享数据操作前加锁，访问结束后立即解锁。

信号量是进化版的互斥锁。它可看作一个计数器，用于多线程、多进程对共享数据对象的访问。访问共享数据时，先测试该资源的信号量，值为正，则进程(线程)可以使用该资源，并将信号量减1，表示使用了一个资源单位。如信号量的值已经为0，则进程进入休眠态。直至信号量大于0。

线程同步，指一个线程发出某一功能调用时，在没有得到结果之前，该调用不返回。同时其它线程为保证数据一致性，不能调用该功能。线程同步目的：在访问共享资源时，保证据一致性，防止产生与时间有关的错误。

互斥量、读写锁、条件变量、信号量。

举例能自圆其说即可。

静态库：使用静态库制作工具：ar rs libname.a a.o b.o c.o

优点：将函数库中的函数本地化。寻址方便，速度快。

缺点：每个使用静态库的进程都要将库编译到可执行文件中加载到内存。内存消耗严重

动态库：

1 生成“与位置无关”的目标文件gcc -fPIC a.c b.c c.c –c（参数 -fPIC 表示生成与位置无关代码）

2 使用gcc -shared选项：gcc -shared -o libname.so a.o b.o c.o

优点：多进程共享一份库文件，节省内存、易于更新

缺点：相较于静态库而言库函数访问略慢。

进程是最小资源分配单位,

线程是最小执行单位,

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位。线程是进程的一个实体,是CPU调度和分派的基本单位,它是比进程更小的能独立运行的基本单位。线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源。

20170702

v1.0 整理的初版

C 语言

C++

Linux