**C++ 什么情况下必须用初始化列表**

参考答案

成员类型是没有默认构造函数的类。若没有提供显示初始化式，则编译器隐式使用成员类型的默认构造函数，若类没有默认构造函数，则编译器尝试使用默认构造函数将会失败。

const 成员或引用类型的成员。因为 const 对象或引用类型只能初始化，不能对他们赋值。

对于普通数据成员而言，其值的设定可以放在 初始化阶段或者普通计算阶段完成。对于 const类型和&引用类型数据成员，其初始化必须在初始化阶段完成。若通过普通计算阶段来初始化该值，编译器会报错：该变量未初始化。

**面向对象的三大特点**

参考答案

C++为了更好的实现面向对象的编程思想，于是引入了新的数据类型——类。由此衍生出三大特性：（1）封装。（2）继承。（3）多态。

**C++ 多态的种类和表现形式？**

参考答案

静态多态（编译器多态）：重载

动态多态（运行时多态）：多态

**你了解的STL算法和容器？**

参考答案

容器：list、vector、map

**list和vector的应用场景？**

参考答案

数组插入和删除操作的时间复杂度是O(n)。而数组是有序的，可以直接通过下标访问元素，十分高效，访问时间复杂度是O(1)（常数时间复杂度）。

如果某些场景需要频繁插入和删除元素时，这时候不宜选用数组作为数据结构。

频繁访问的场景下，可以使用数组。

反之，频繁插入或删除的场景用链表

**构造函数可以为虚函数吗？父类的析构函数为什么是虚函数？**

参考答案

构构造函数不能为虚函数，当申明一个函数为虚函数时，会创建虚函数表，那么这个函数的调用方式是通过虚函数表来调用。若构造函数为虚函数，说明调用方式是通过虚函数表调用，需要借助虚表指针，但是没构造对象，哪里来的虚表指针？但是没有虚表指针，怎么访问虚函数表从而调用构造函数呢？这就成了一个先有鸡还是先有蛋的问题。

若存在继承关系时，析构函数必须申明为虚函数，这样父类指针指向子类对象，释放基类指针时才会调用子类的析构函数释放资源，否则内存泄漏。

若存在继承关系时，析构函数必须申明为虚函数，这样父类指针指向子类对象，释放基类指针时才会调用子类的析构函数释放资源，否则内存泄漏。

**结构体对齐相关**

参考答案

自身对齐值 数据成员本身的对齐值，如：char类型占1个字节，即自身对齐值为1，short类型占2个字节，自身对齐值为2，int类型占4个字节，自身对齐值为4，double类型占8个字节，自身对齐值为8；

指定对齐值 1、由编译器指定：32位机是个4字节，即指定对齐值为4，64位机是个8字节，即指定对齐值为8； 2、由程序员指定： 使用#pragma pack(n)来指定，n为2的幂次方，如1，2，4，8，16等；

有效对齐值 自身对齐值和指定对齐值中较小的那个 ，如：32位机，char类型，则此成员有效对齐值为1；double类型，则此成员有效对齐值为4；如：程序员指定#pragma pack(2)，char类型，则此成员有效对齐值为1；double类型，则此成员有效对齐值为2；

**如何取消fork的文件描述符的共享？**

参考答案

因为现代fork采用共享文件描述符来实现写时复制，所以就是要进程正常复制。

fork调用了do\_fork这个内核函数，这个函数比较大，主干程序是copy\_process，这里有一系列的copy\_xxx系列产品，这个系列产品会根据传进来的标志位，来决定那些资源子进程需要copy一份，那些不用拷贝了，直接用父进程的就可以了。 我们关注的copy\_mm这个函数，如果用户标志位中的CLONE\_VM置了0，意味着取消共享，需要copy一份

**你了解的锁机制？**

（1）互斥锁：mutex，保证在任何时刻，都只有一个线程访问该资源，当获取锁操作失败时，线程进入阻塞，等待锁释放。

（2）读写锁：rwlock，分为读锁和写锁，处于读操作时，可以运行多个线程同时读。但写时同一时刻只能有一个线程获得写锁。

互斥锁和读写锁的区别：

（a）读写锁区分读锁和写锁，而互斥锁不区分

（b）互斥锁同一时间只允许一个线程访问，无论读写；读写锁同一时间只允许一个线程写，但可以多个线程同时读。

（3）自旋锁：spinlock，在任何时刻只能有一个线程访问资源。但获取锁操作失败时，不会进入睡眠，而是原地自旋，直到锁被释放。这样节省了线程从睡眠到被唤醒的时间消耗，提高效率。

（4）条件锁：就是所谓的条件变量，某一个线程因为某个条件未满足时可以使用条件变量使该程序处于阻塞状态。一旦条件满足了，即可唤醒该线程(常和互斥锁配合使用)

（5）信号量。
**举一个死锁的例子**

参考答案

**死锁:** 是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种**互相等待**的现象。此时称系统处于死锁状态或系统产生了死锁。这些永远在互相等待的进程称为死锁进程。比如两只羊过独木桥。进程比作羊，资源比作桥。若两只羊互不相让，争着过桥，就产生死锁。

多线程与多进程的同步手段

参考答案

线程间的同步方式包括互斥锁、信号量、条件变量、读写锁：

* 互斥锁：采用互斥对象机制，只有拥有互斥对象的线程才可以访问。因为互斥对象只有一个，所以可以保证公共资源不会被多个线程同时访问。
* 信号量：计数器，允许多个线程同时访问同一个资源。
* 条件变量：通过条件变量通知操作的方式来保持多线程同步。
* 读写锁：读写锁与互斥量类似。但互斥量要么是锁住状态，要么就是不加锁状态。读写锁一次只允许一个线程写，但允许一次多个线程读，这样效率就比互斥锁要高。

进程：

* 信号量semaphore：是一个计数器，可以用来控制多个进程对共享资源的访问。信号量用于实现进程间的互斥与同步。P操作(递减操作)可以用于阻塞一个进程，V操作(增加操作)可以用于解除阻塞一个进程。
* 管程：一个进程通过调用管程的一个过程进入管程。在任何时候，只能有一个进程在管程中执行，调用管程的任何其他进程都被阻塞，以等待管程可用。
* 消息队列：消息的链接表，放在内核中。消息队列独立于发送与接收进程，进程终止时，消息队列及其内容并不会被删除；消息队列可以实现消息的随机查询，可以按照消息的类型读取。
* 互斥锁

在linux下你一般怎么做多线程和多进程的开发

参考答案

多进程下如何调试：用set follow-fork-mode child 调试子进程

或者set follow-fork-mode parent 调试父进程

一般怎么调试程序，介绍下常用的gdb命令

参考答案

GDB调试：gdb调试的是可执行文件，在gcc编译时加入 -g ，告诉gcc在编译时加入调试信息，这样gdb才能调试这个被编译的文件 gcc -g tesst.c -o test

GDB命令格式：

quit：退出gdb，结束调试

list：查看程序源代码

list 5，10：显示5到10行的代码

list test.c:5, 10: 显示源文件5到10行的代码，在调试多个文件时使用

list get\_sum: 显示get\_sum函数周围的代码

list test,c get\_sum: 显示源文件get\_sum函数周围的代码，在调试多个文件时使用

reverse-search：字符串用来从当前行向前查找第一个匹配的字符串

run：程序开始执行

help list/all：查看帮助信息

break：设置断点

break 7：在第七行设置断点

break get\_sum：以函数名设置断点

break 行号或者函数名 if 条件：以条件表达式设置断点

watch 条件表达式：条件表达式发生改变时程序就会停下来

next：继续执行下一条语句 ，会把函数当作一条语句执行

step：继续执行下一条语句，会跟踪进入函数，一次一条的执行函数内的代码

条件断点：break if 条件 以条件表达式设置断点

多进程下如何调试：用set follow-fork-mode child 调试子进程

或者set follow-fork-mode parent 调试父进程

其他

1.linux程序编译过程，具体说说

预处理（Preprocessing）
编译（Compilation）
汇编（Assembly）
链接（Linking）
2.linux怎么定位内存泄漏

3.linux检测内存泄漏时，发现是内存太小了，怎么办？

4.linux下gdb的使用

5.快排原理 复杂度 出现最差情况的实际情景(每次选的基准数都很极端)

6.vector插入元素后的详细过程

7.为什么要有迭代器，不是有指针吗？

8.说说STL在多线程编程时的情况，稳定性

9.linux环境下编程

10.项目