问题

什么是内存泄漏？有哪几种情况？如何判断是否存在内存泄漏？

什么是内存泄漏？

内存泄漏（memory leak），是指由于疏忽或错误造成了程序未能释放掉不再使用的内存的情况。内存泄漏并非指内存在物理上的消失，而是应用程序分配某段内存后，由于设计错误，失去了对该段内存的控制，因而造成了内存的浪费或者性能不良的情况。

内存泄漏有哪几种情况？

1.堆内存泄漏 （Heap leak）

堆内存指的是程序运行中根据需要分配通过malloc，realloc，new等从堆中分配的一块内存，在完成相关操作后必须通过调用对应的 free或者delete 删掉。如果程序的设计的错误导致这部分内存没有被释放，那么此后这块内存将不会被使用，就会产生Heap Leak。

2.系统资源泄露（Resource Leak）。

主要是指系统分配给程序的资源没有使用相应的函数释放掉（比如 Bitmap，handle，socket等），导致系统资源的浪费，严重可导致系统效能降低，系统运行不稳定。

3.没有将基类的析构函数定义为虚函数

当基类指针指向派生类对象时，如果基类的析构函数不是virtual，那么子类的析构函数将不会被调用，子类的资源没有正确是释放，因此造成内存泄露。（这一点在之前总结的C++问题 ***16_析构函数*** 中关于**析构函数为什么一般定义成虚函数**中也有说明。）

4.在释放对象数组时没有使用delete[]而是使用了delete

在之前总结的C++问题 ***12_delete和delete[]的区别*** 中有过举例说明，也就是说，当一个数组中的多个元素均为对象时，在使用delete释放该数组时必须加上方括号([])，否则就只会调用一次析构函数释放数组的第一个对象，而剩下的数组元素没有被析构掉，造成了内存泄漏。

5.缺少拷贝构造函数

在C++问题 ***46_深拷贝与浅拷贝的区别*** 中有提到过，如果类中没有手动编写拷贝构造函数，用该类对象进行拷贝赋值时，会使用默认的拷贝构造函数，即浅拷贝，浅拷贝的缺陷之一被赋值对象原本的内存没被释放，因此造成了内存泄漏。

```
//例如,假设有一个String类
String a("hello");
String b("world");
b = a;  //b的指针会指向a所在的内存，但b原本的内存没有被释放，造成了内存泄漏
```

如何判断内存泄漏？

1.**在 Linux 环境下可以使用内存泄漏检查工具 valgrind。**

2.在编写代码时可以**手动添加内存申请和释放的统计功能，根据当前申请和释放的内存是否一致来判断**是否发生内存泄漏。

如何解决内存泄漏的问题？

1.可以按照上述判断内存泄漏的几种方法来防止内存泄漏

2.**使用智能指针**。智能指针可以自行管理指针，因为智能指针就是一个类，在类的作用域结束时会自动调用析构函数来释放内存空间，可以减少内存泄漏的问题（注意！！是减少，而不是完全解决）。

智能指针有内存泄漏的问题吗？

前面说了，智能指针可以减少内存泄漏的问题，但不能完全解决，也就是说**智能指针也是存在内存泄漏的问题的**。那智能指针什么时候会发生内存泄漏呢？

当两个对象同时使用一个`shared_ptr`成员变量指向对方时，会造成循环引用，使引用计数失效，从而导致内存泄漏。

> 补充：如何理解上面这句话？（个人理解，作为参考）
>
> 使用共享指针`shared_ptr`可以使得多个指针同时指向同一个对象，同时对指向该对象的指针进行引用计数，并且该对象会在最后一个引用被销毁时释放，也就是说当引用计数为0时，该对象即相关内存就会被自动释放。因此，问题来了，当两个对象同时使用`shared_ptr`指针指向对方时，彼此会相互引用，引用计数不会为0，因此最后这两个对象的资源不会被释放，造成了内存泄漏。

如何解决上述智能指针的内存泄漏问题？

使用`weak_ptr`弱指针。`shared_ptr`指针指向一个对象时，并不会修改该对象上的引用计数，但可以访问该对象及获取该对象上的引用计数（这也是为什么它叫做弱指针的原因）。因此，可以使用`shared_ptr`指针来避免一些非法访问。

> 关于智能指针的内容，之后再做详细的整理。

参考资料

[牛客网-C++工程师面试宝典]

[C++中内存泄漏的几种情况]

[C++智能指针的内存泄漏和解决方法]