C/C++内存管理

1. C/C++内存管理



①栈又叫堆栈，非静态局部变量/函数参数/返回值等等，栈是向下增长的。

②内存映射段是高效的I/O映射方式，用于装载一个共享的动态内存库。用户可使用系统接口创建共享共

享内存，做进程间通信。（Linux课程如果没学到这块，现在只需要了解一下）

③堆用于程序运行时动态内存分配，堆是可以上增长的。

④数据段--存储全局数据和静态数据。

⑤代码段--可执行的代码/只读常量。

2.C语言中的内存管理

malloc/calloc/realloc

1. C++内存管理

C语言内存管理方式在C++中可以继续使用，但有些地方就无能为力而且使用起来比较麻烦，因此C++又提出了自己的内存管理方式：通过new和delete操作符进行动态内存管理。

* 1. new/delete管理内置类型

int main()

{

int\* k = new int(8);

cout << \*k << endl;

delete k;

return 0;

}

看看

* 1. new/delete管理自定义类型

注意：在申请自定义类型的空间时，new会调用构造函数，delete会调用析构函数，而malloc与free不会。

1. operator new与operator delete函数

new和delete是用户进行动态内存申请和释放的操作符，operator new 和operator delete是系统提供的全局函数，new在底层调用operator new全局函数来申请空间，delete在底层通过operator delete全局函数来释放空间。

5. new和delete的实现原理

5.1 内置类型

如果申请的是内置类型的空间，new和malloc，delete和free基本类似，不同的地方是：new/delete申请和释放的是单个元素的空间，new[]和delete[]申请的是连续空间，而且new在申请空间失败时会抛异常，malloc会返回NULL。

5.2自定义类型

new的原理：

1. 调用operator new函数申请空间

2. 在申请的空间上执行构造函数，完成对象的构造

delete的原理：

1. 在空间上执行析构函数，完成对象中资源的清理工作

2. 调用operator delete函数释放对象的空间

new T[N]的原理：

1. 调用operator new[]函数，在operator new[]中实际调用operator new函数完成N个对象空间的申请

2. 在申请的空间上执行N次构造函数

delete[]的原理：

1. 在释放的对象空间上执行N次析构函数，完成N个对象中资源的清理

2. 调用operator delete[]释放空间，实际在operator delete[]中调用operator delete来释放空间

1. 定位new表达式(placement-new)

5.1定位new表达式是在已分配的原始内存空间中调用构造函数初始化一个对象。

5.2使用格式：

new (place\_address) type或者new (place\_address) type(initializer-list)

place\_address必须是一个指针，initializer-list是类型的初始化列表

5.3 使用场景：

定位new表达式在实际中一般是配合内存池使用。因为内存池分配出的内存没有初始化，所以如果是自定义类型的对象，需要使用new的定义表达式进行显示调构造函数进行初始化。

1. 设计一个类，该类只能在堆上创造对象

方法一：构造函数私有化

1. 将类的构造函数私有，拷贝构造声明成私有。防止别人调用拷贝在栈上生成对象。

2. 提供一个静态的成员函数，在该静态成员函数中完成堆对象的创建

class A

{

public:

static A\* create(int k)

{

return new A(k);

}

static A\* create()

{

return new A;

}

private:

A(){}

A(int k)

: \_k(k)

{}

A(const A& a);

private:

int \_k;

};

方法二：将析构函数私有化

7.3 请设计一个类，该类只能在栈上创建对象

方法一：构造函数私有化

方法二：只能在栈上创建对象，即不能在堆上创建，因此只要将new的功能屏蔽掉即可，即屏蔽掉operator new和定位new表达式，注意：屏蔽了operator new，实际也将定位new屏蔽掉。