Autorelease 详解

相关链接：黑幕背后的Autorelease

前言

MRC 中，[obj autorelease]延迟内存释放。 ARC 中，下文详解。

Autorelease 对象释放时机

- 没有手动加入 Autorelease Pool 的情况：autorelease 对象在当前 runloop 迭代结束时释放的(默认放入最近的一个 pool 中)。

  > 原因：系统在每个 runloop 迭代中都自动加入了 AutoreleasePool（最近的一个） 的 Push 和 Pop 操作。

- 手工干预 Autorelease Pool 的情况：autorelease 对象离开@autoreleasepool{……}时已经释放。

Autorelease 原理

- ARC 中，使用 AutoreleasePool 的方式：

  调用 `@autoreleasepool{……}` ，编译器编译后为：

  ```
  void *context = objc_autoreleasePoolPush();
  // {}中的代码
  objc_autoreleasePoolPop(context);
  ```

- 上述编译后的 Push 和 Pop 方法均是对 `AutoreleasePoolPage` 类的封装。该类是自动释放机制的核心。

  ```
  //AutoreleasePoolPage C++实现
  magic_t const magic;
  id *next;
  pthread_t const thread;

  AutoreleasePollPage *const parent;
  AutoreleasePollPage *child;

  uint32_t const depth;
  uint32_t hiwat;
  ```

- AutoreleasePool:由若干个AutoreleasePoolPage以双向链表的形式组合而成（分别对应结构中的parent指针和child指针)
- AutoreleasePool是按线程一一对应的（结构中的thread指针指向当前线程）
- AutoreleasePoolPage每个对象会开辟4096字节内存（也就是虚拟内存一页的大小），除了上面的实例变量所占空间，剩下的空间全部用来储存autorelease对象的地址.
- 上面的id *next指针作为游标指向栈顶最新add进来的autorelease对象的下一个位置.
- 一个AutoreleasePoolPage的空间被占满时，会新建一个AutoreleasePoolPage对象，连接链表，后来的autorelease对象在新的page加入.

综上：向一个对象发送- autorelease消息，就是将这个对象（地址）加入到当前AutoreleasePoolPage的栈顶next指针指向的位置。

- 释放时刻
  - 每当进行一次objc_autoreleasePoolPush调用时，runtime向当前的AutoreleasePoolPage中add进一个哨兵对象，值为0（也就是个nil）。
  - objc_autoreleasePoolPush的返回值正是这个哨兵对象的地址，被objc_autoreleasePoolPop(哨兵对象)作为入参：
    - 根据传入的哨兵对象地址找到哨兵对象所处的page
    - 在当前page中，将晚于哨兵对象插入的所有autorelease对象都发送一次- release消息，并向回移动next指针到正确位置.
    - 补充2：从最新加入的对象一直向前清理，可以向前跨越若干个page，直到哨兵所在的page.