实现一个简单的动态内存分配器

一、简介

在这个实验中，将用 C 程序编写一个动态内存分配器，即您自己版本的 malloc、free 和 realloc 函数。鼓励有创新性地实现正确、高效和快速的分配器。

二、代码结构

config.h

定义了一些参数常量,例如本次实验模型化的空间大小

实现动态内存分配器：

mm.h , mm.c

需要实现的函数均在 mm.c 中完成

memlib.h , memlib.c

提供一个允许我们在不干涉已存在的系统层 malloc 包的情况下的内存系统模型

性能评估：

mdriver.c

使用 trace 文件(即*.rep)测试我们实现的 mm.c 的正确性，空间利用率和吞吐量。

trace 文件包括了一系列分配、重新分配和释放指示，mdriver.c 根据这些指示调用 mm.c 中的 mm_malloc，mm_realloc 和 mm_free 函数。

三、实验说明

动态内存分配器将由以下四个函数组成，它们在 mm.h 中声明，并在 mm.c 定义实现。

- int mm_init(void);
- void *mm_malloc(size_t size);
- void mm_free(void *ptr);
- void *mm_realloc(void *ptr, size_t size);

提供的 mm.c 文件实现了最简单的 malloc 包。以此为起点，修改这些函数（可能定义其他私有静态函数），使它们完成如下功能：

mm_init：在调用 mm_malloc, mm_realloc 或 mm_free 之前，应用程序（即用于评估实现的跟踪驱动程序）调用 mm_init 来执行任何必要的初始化，例如分配初始堆区域。如果在执行初始化时出现问题，则返回值应为-1，否则为 0。

mm_malloc: 返回一个指针，指向至少 size 字节的已分配块。整个分配的块应位于堆区域内，并且不应与任何其他分配的块重叠。并且我们将把您的实现与标准 C 库（libc）中提供的 malloc 版本进行比较。由于 libc malloc 总是返回与 8 字节对齐的有效负载指针，因此 malloc 实现也应该这样做，并且总是返回与 8 字节对齐的指针。

mm_free：释放 ptr 指向的块。它什么也不返回。只有在前面对 mm_malloc 或 mm_realloc

的调用返回传递指针（ptr）且尚未释放时，此功能才能保证工作。

mm_realloc:返回一个指针，该指针指向具有以下约束的至少大小为 size 字节的已分配区域。
–如果指针为空，则等价于 mm_malloc(size)；
–如果 size 为 0，则等价于 mm_free(ptr)；
–如果指针不为空，它必须是通过先前对 mm_malloc 或 mm_realloc 的调用返回的。对 mm_realloc 的调用将 ptr（旧块）指向的内存块的大小更改为 size 字节，并返回新块的地址。请注意，新块的地址可能与旧块相同，也可能不同，具体取决于：您的实现、旧块中的内部碎片数量以及重新分配请求的大小。
新块的内容与旧块的内容相同，大小为新块和旧块大小的最小值。其他均未初始化。例如，如果旧块是 8 字节，而新块是 12 字节，则新块的前 8 字节与旧块的前 8 字节相同，新块最后 4 个字节未初始化。类似地，如果旧块是 8 字节，而新块是 4 字节，则新块的内容与旧块的前 4 字节相同。
这些功能与 libc 中对应的 malloc、realloc 和 free 功能相匹配。在 shell 中输入 man malloc 以获取完整的文档介绍。

四、性能验证驱动程序 mdriver.c

接受以下命令行参数：
-t<tracedir>：在目录 tracedir 中查找默认跟踪文件，而不是在 config.h 中定义的默认目录。
-f<tracefile>：使用一个特定的 tracefile 进行测试，而不是使用默认的跟踪文件。
-h：打印命令行参数的摘要。
-l：除了编写的 malloc 包外，还运行并测试 libc 的 malloc。
-v: 详细输出。打印每个跟踪文件的性能得分。
-V：更详细的输出。在处理每个跟踪文件时打印其他诊断信息。可在调试期间用于确定哪个跟踪文件导致失败。

五、评分规则

将使用两个性能指标来评估解决方案：
–**空间利用率**：驱动程序使用的内存总量（通过 mm_malloc 或 mm_realloc 分配，但尚未通过 mm free 释放）与分配器使用的堆大小之间的峰值比率。最佳比率等于 1。应该找到好的策略来最小化碎片，以便使此比率尽可能接近最佳值。
–**吞吐量**：平均每秒完成的操作数。
驱动程序通过计算性能索引（performance index）来总结分配器的性能 P，是空间利用率和吞吐量的加权和:

$$P = wU + (1-w)\min\left(1, \frac{T}{T_{libc}}\right)$$

其中 U 是空间利用率，T 是你的分配器的吞吐量，T_{libc}是 libc 的 malloc 包的吞吐量。默认 w=0.6。
鉴于内存和 CPU 都是昂贵的系统资源，采用此公式鼓励平衡优化内存利用率和吞吐量。理想情况下，性能指标将达到 P=w+（1-w）=1 或 100%。由于每个度量对性能索引的贡献分别最多为 w 和 1-w，因此不应为了优化内存利用率或仅优化吞吐量而走极端。要获得好的分数，必须在利用率和吞吐量之间取得平衡。