glibc内存管理实现分析

不管是何种系统，良好的内存管理都是系统稳定运行的基石，在Linux环境下，使用最为广泛的内存管理系统就是glibc的内存管理系统，glibc的内存管理系统也是其他很多更高级语言的内存管理的基石，因此，深入分析这一广泛使用久经考验的工业级内存管理系统对于学习内存管理技术以及对内存管理的理解就有很大的意义。glibc的内存管理实现位于malloc目录中。接下来对其进行深入的分析。

malloc目录下有很多文件，其中很多tst开头的文件，是用于测试的程序，不是我们学习的对象，内存管理系统的实现主要是文件malloc.h和malloc.c。这两个文件实现了glibc的内存管理系统。glibc所用的malloc并不是最快的，也不是最节约内存的，它是在寻找一个平衡点，来达到更好的通用性。显然，在工程领域以及实践中，这个实现确实足够通用，而且久经考验。

该内存管理的实现算法主要有以下特点：

对于大于等于512字节的分配，

对于小于等于64字节的分配请求

对于在(64, 512)的分配请求，

对于非常大的内存分配请求（大于128KB）则直接使用内存映射机制实现，对于Linux而言，是mmap。

该内存分配系统实现了C标准的函数，同时增加了一些自己的扩展：

标准的函数：

malloc(size\_t n);

calloc(size\_t n\_elements, size\_t element\_size);

free(void\* p);

realloc(void\* p, size\_t n);

memalign(size\_t alignment, size\_t n);

valloc(size\_t n);

mallinfo()

mallopt(int parameter\_number, int parameter\_value)

扩展的函数：

independent\_calloc(size\_t n\_elements, size\_t size, void\* chunks[]);

independent\_comalloc(size\_t n\_elements, size\_t sizes[], void\* chunks[]);

pvalloc(size\_t n);

malloc\_trim(size\_t pad);

malloc\_usable\_size(void\* p);

malloc\_stats();