内存管理基础[¶](https://php.net/manual/zh/internals2.memory.management.php" \l "internals2.memory.management)

用 C 语言编程时，开发者要手工地进行内存管理。因为 PHP 经常用作 Web 服务器的模块，内存管理与预防内存泄漏紧密关联。此外要知道 PHP 可能用于线程环境中，这意味着全局变量可能导致竞争状况。有关线程内全局数据处理的信息请参见作为线程隔离设施的 [线程安全的资源管理器](https://php.net/manual/zh/internals2.memory.TSRM.php)。

此外，Zend 引擎要面对一个十分特殊的使用模式：在一段比较短的时间内，许多 zval 结构大小的内存块和其他的小内存块被申请又再被释放。PHP 的内存管理也很重视 [memory\_limit(内存限制)](https://php.net/manual/zh/ini.core.php#ini.memory-limit)。

为了满足以上的需求，Zend 引擎提供为了处理请求相关数据提供了一种特殊的内存管理器。请求相关数据是指只需要服务于单个请求，最迟会在请求结束时释放的数据。扩展开发者主要接触下表中列出的惯例。虽然一些所提供的便捷功能是用宏实现的，但在本文中会象函数一样对待。

| **主要的内存 APIs** | |
| --- | --- |
| **原型** | **说明** |
| void \*emalloc(size\_t size) | 分配 size 字节的内存。 |
| void \*ecalloc(size\_t nmemb, size\_t size) | 给 nmemb 元素分配 size 字节的缓冲区并初始化为零。 |
| void \*erealloc(void \*ptr, size\_t size) | 修改使用 emalloc 分配的缓冲区 ptr 的大小为 size 字节。 |
| void efree(void \*ptr) | 释放 ptr 指向的缓冲区。缓冲区必须是由 emalloc 分配的。 |
| void \*safe\_emalloc(size\_t nmemb, size\_t size, size\_t offset) | 分配缓冲区来存放每块大小为 size 字节的 nmemb 块，并附加 offset 字节。类似于 emalloc(nmemb \* size + offset)，但增加了针对溢出的特殊保护。 |
| char \*estrdup(const char \*s) | 分配一个可存放 NULL 结尾的字符串 s 的缓冲区，并将 s 复制到缓冲区内。 |
| char \*estrndup(const char \*s, unsigned int length) | 类似于 estrdup，但 NULL 结尾的字符串长度是已知的。 |

Note: 和与 C 标准库相似的部分不同，如果分配请求的内存出错，Zend 引擎的内存管理函数不会返回 NULL 值，而会跳出并中止当前请求。

如上所述，防止有内存泄漏并尽可能快地释放所有内存是内存管理的重要组成部分。因为安全原因，在请求结束时， Zend 引擎会释放所有由上面提到的 API 所分配的内存。如果 PHP 使用 --enable-debug 配置选项进行构建，这将产生一个警告。

Example #1 PHP 的泄漏报警

ZEND\_FUNCTION(leak)

{

long leakbytes = 3;

if (zend\_parse\_parameters(ZEND\_NUM\_ARGS() TSRMLS\_CC, "|l", &leakbytes) == FAILURE) {

return;

}

emalloc(leakbytes);

}

以上例程的输出类似于：

[Thu Oct 22 02:14:57 2009] Script: '-'

/home/johannes/src/PHP\_5\_3/Zend/zend\_builtin\_functions.c(1377) : Freeing 0x088888D4 (3 bytes), script=-

=== Total 1 memory leaks detected ===

Note: 当使用 PHP 变量时，需要确认变量的内存要使用 emalloc 来分配，并注意引用计数。相关细节请查看 [变量的使用](https://php.net/manual/zh/internals2.variables.php)。

Note: 内存泄漏检测仅可以发现由 emalloc 分配内存块导致的泄漏。为进行深层分析，建议使用内存检测器，如 valgrind 或 libumem 等。要简化此分析，可在 PHP 启动时通过设置环境变量 USE\_ZEND\_ALLOC=0 来禁用 PHP 的内存管理器。