数据持久性[¶](https://php.net/manual/en/internals2.memory.persistence.php" \l "internals2.memory.persistence)

在这种情况下，数据持久性意味着任何旨在保留当前请求的数据。引擎中的内存管理非常专注于请求绑定分配，但这并不总是实用或合适的。为了满足外部库的需求，有时需要持久化内存，它也可能是有用的Hacking。

持久内存的一种常见用途是启用持久的SQL服务器连接，尽管这种做法令人不悦，但它并不是这个特性最常用的用法。

注意： 以下所有函数都采用附加的持久参数，如果此参数为false，则引擎将使用其常规分配程序（emalloc），并且不应将内存视为持久性。在内存分配为持久性的情况下，调用系统分配器，在大多数情况下，它们仍不能像主内存API那样返回NULL指针。

| **持久性内存API** | |
| --- | --- |
| **原型** | **描述** |
| void \*pemalloc(size\_t size, zend\_bool persistent) | 分配size内存的字节。 |
| void \*pecalloc(size\_t nmemb, size\_t size, zend\_bool persistent) | nmemb为size字节元素 分配一个缓冲区并确保它用零初始化。 |
| void \*perealloc(void \*ptr, size\_t size, zend\_bool persistent) | 调整ptr分配的缓冲区大小， emalloc以保存size内存字节。 |
| void pefree(void \*ptr, zend\_bool persistent) | 释放指向的缓冲区ptr。缓冲区必须由...分配pemalloc。 |
| void \*safe\_pemalloc(size\_t nmemb, size\_t size, size\_t offset, zend\_bool persistent) | 分配一个缓冲区来保存nmemb每个size字节的块 和一个额外的offset字节。这类似于pemalloc(nmemb \* size + offset)增加了对溢出的特殊保护。 |
| char \*pestrdup(const char \*s, zend\_bool persistent) | 分配一个可以保存以NULL结尾的字符串 s并将其复制s到该缓冲区的缓冲区。 |
| char \*pestrndup(const char \*s, unsigned int length, zend\_bool persistent) | 类似于以pestrdupNULL结尾的字符串的长度已知。 |

**警告**

重要的是要记住，分配给持久性的内存不是由引擎优化或跟踪的; 它不受memory\_limit的限制，另外，由Hacker引擎创建的所有变量都不能使用持久性内存。