<http://www.hack6.com/html/bc/C/20141106/52232.html>

在任何一个时间点上，线程是可结合的（joinable）或者是分离的（detached）。一个可结合的线程能够被其他线程收回其资源和杀死。在被其他线程回收之前，它的存储器资源（例如栈）是不释放的。相反，一个分离的线程是不能被其他线程回收或杀死的，它的存储器资源在它终止时由系统自动释放。

    默认情况下，线程被创建成可结合的。为了避免存储器泄漏，每个可结合线程都应该要么被显示地回收，即调用pthread\_join；要么通过调用pthread\_detach函数被分离。

[cpp]

int pthread\_join(pthread\_t tid, void\*\*thread\_return);

                                 若成功则返回0，若出错则为非零。

int pthread\_join(pthread\_t tid, void\*\*thread\_return);

                                 若成功则返回0，若出错则为非零。    线程通过调用pthread\_join函数等待其他线程终止。pthread\_join函数分阻塞，直到线程tid终止，将线程例程返回的(void\*)指针赋值为thread\_return指向的位置，然后回收已终止线程占用的所有存储器资源。[cpp] view plaincopyprint?int pthread\_detach(pthread\_t tid);

                                 若成功则返回0，若出错则为非零。

int pthread\_detach(pthread\_t tid);

                                 若成功则返回0，若出错则为非零。

    pthread\_detach用于分离可结合线程tid。线程能够通过以pthread\_self()为参数的pthread\_detach调用来分离它们自己。

    如果一个可结合线程结束运行但没有被join，则它的状态类似于进程中的Zombie Process，即还有一部分资源没有被回收，所以创建线程者应该调用pthread\_join来等待线程运行结束，并可得到线程的退出代码，回收其资源。

    由于调用pthread\_join后，如果该线程没有运行结束，调用者会被阻塞，在有些情况下我们并不希望如此。例如，在Web服务器中当主线程为每个新来的连接请求创建一个子线程进行处理的时候，主线程并不希望因为调用pthread\_join而阻塞（因为还要继续处理之后到来的连接请求），这时可以在子线程中加入代码

    pthread\_detach(pthread\_self())

或者父线程调用

    pthread\_detach(thread\_id)（非阻塞，可立即返回）

这将该子线程的状态设置为分离的（detached），如此一来，该线程运行结束后会自动释放所有资源。

(责任编辑：编程棋牌开发) 本文章原创来自：http://www.hack6.com QQ：2 （转载请保留网站版权。侵权必究）