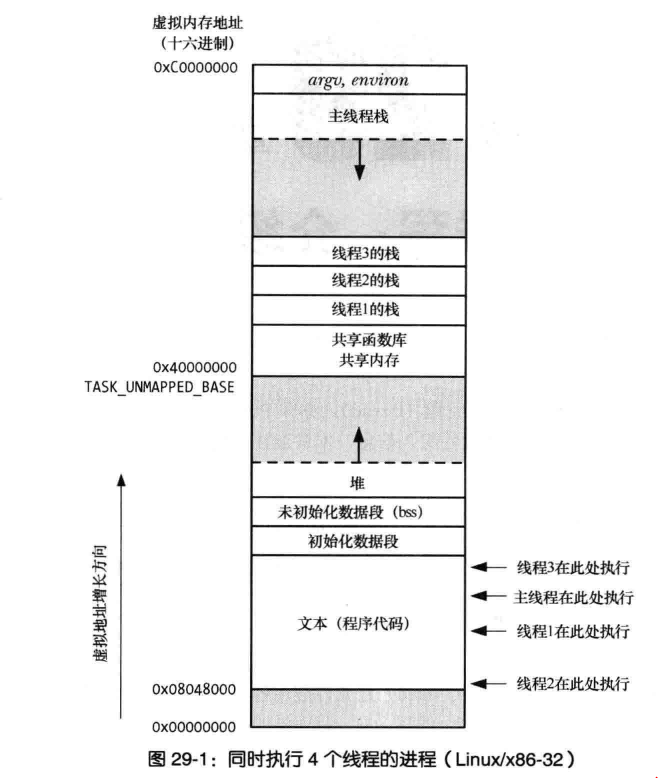
第二十九章

1. 线程概述：

本章及随后几章讨论posix线程。主要描述Pthreads的API所规定的标准行为。当然有不同的实现，如Linux thread和NPTL。

多个线程的虚拟内存映像：

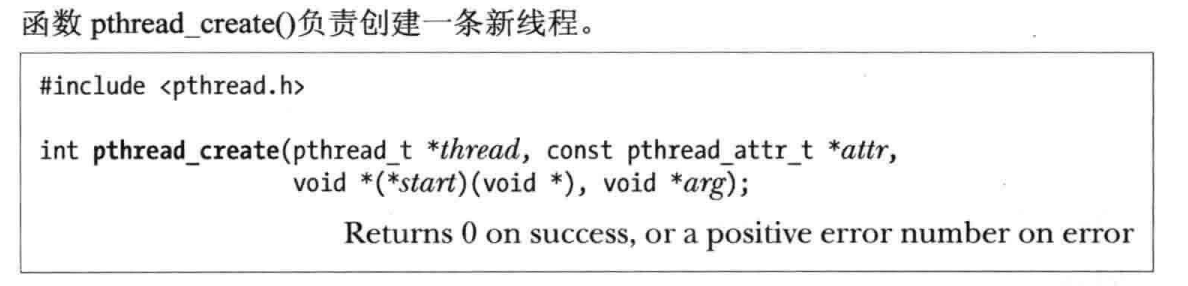


同一程序中的所有线程会独立执行相同的程序，共享同一份内存区域，包括：初始化数据段，未初始化数据段，堆段。

多个进程实现并行任务的缺点：

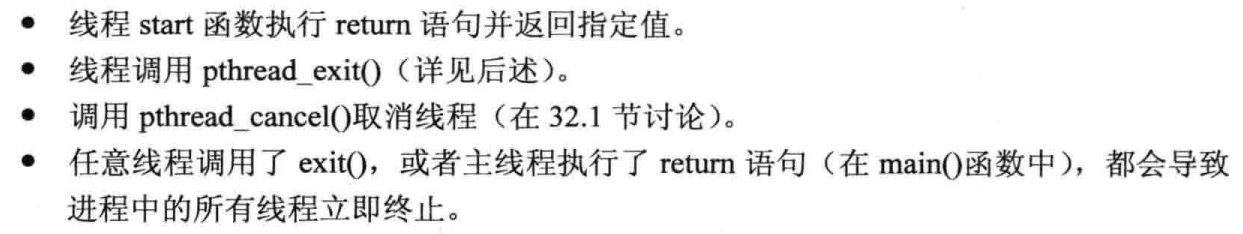
进程间的信息难以共享以及fork所带来的消耗。此外，线程还共享许多除全局外的属性以及各自所特有的属性。具体查看书本。

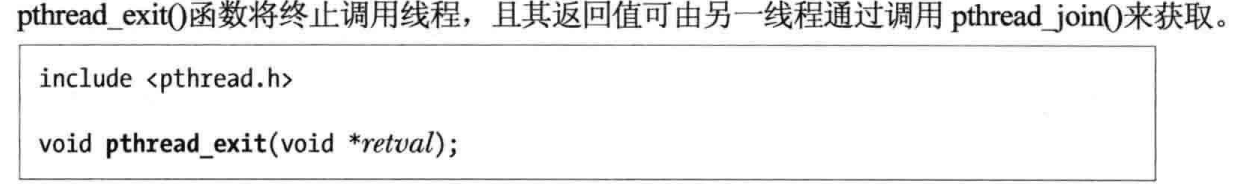
1. Pthreads API的详细背景：
2. 不能对线程数据类型的结构和内容有任何依赖，不能使用比较符比较。
3. 每个线程都有属于自己的errno，实现方式是定义一个宏，这个宏展开是函数，返回一个左值。因为是函数，所以就可以做到属于每个线程。但是每次引用都会有开销。
4. 执行失败返回正值表示失败内容，与errno相同。
5. 编译时要设置编译选项。
6. 创建线程：

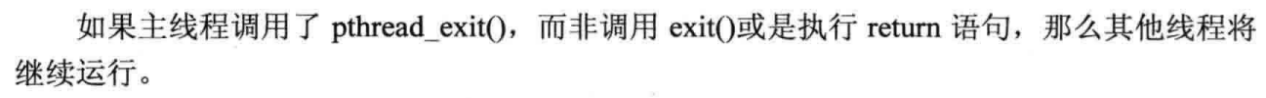


1. 终止线程：

终止线程运行的情况有四种：



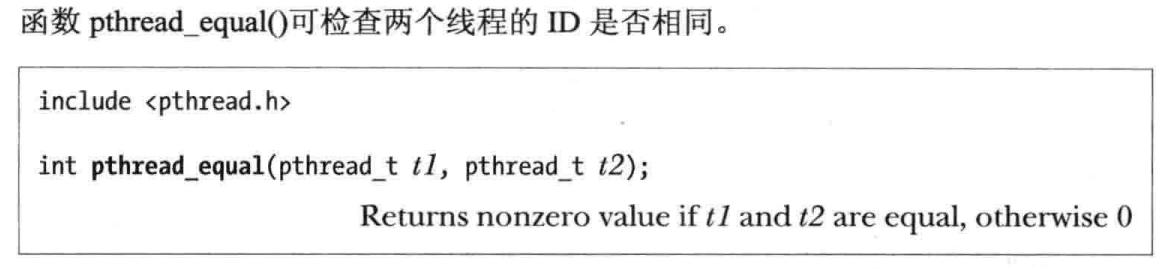


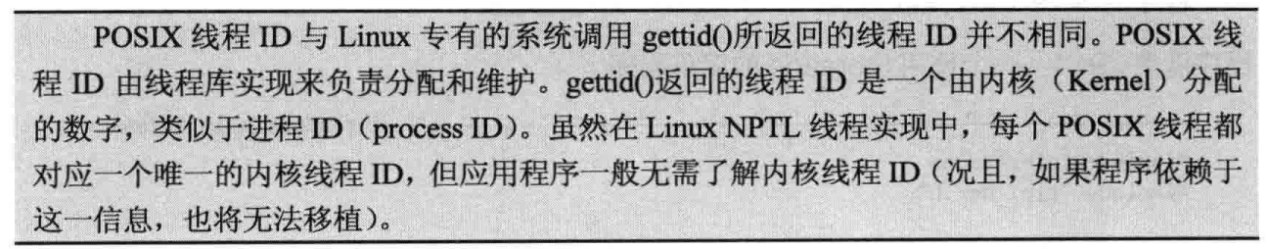


后面了解到，pthread\_exit是调用\_exit，而exit是exit\_group。

1. 线程ID：



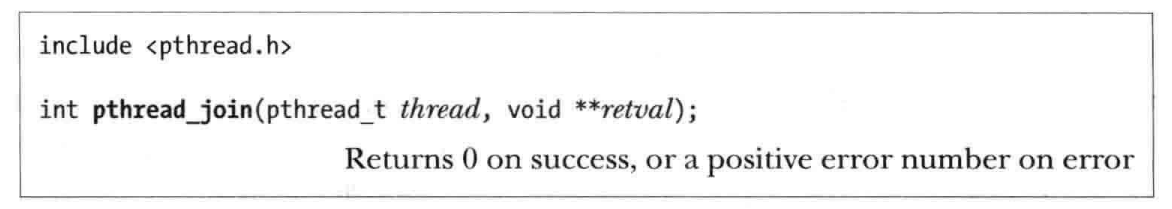


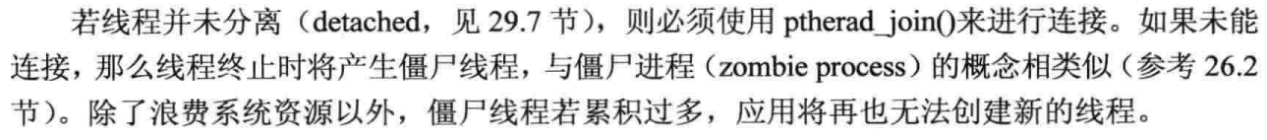


线程ID保证在进程内是唯一的，但不保障在所有进程中是唯一，即使Linux是如此。

不应假设线程ID的类型，它有可能是结构或标量。

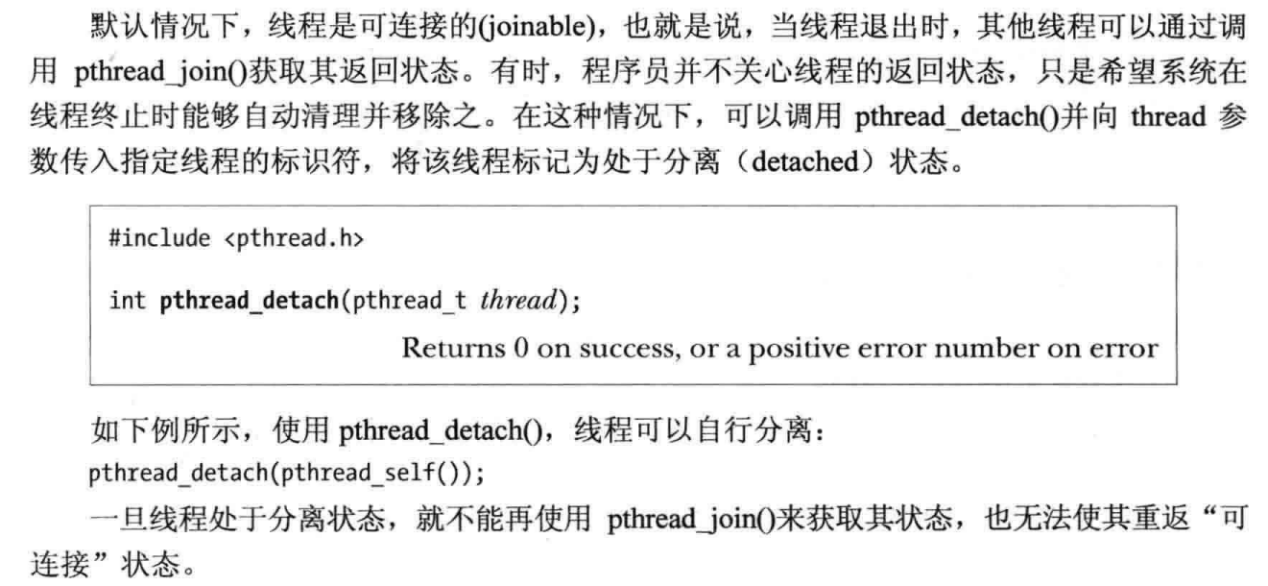
1. 连接已终止的线程：





此函数与waitpid有两个“区别”：没有等级关系和无法等待所有线程和不阻塞。

1. 线程的分离：



1. 线程属性：

创建线程pthread\_create函数的第二个参数可以指定具体线程的具体属性。

1. 进程还是线程：

设计程序的时候需要考虑许多要素。线程的优点：共享数据简单，上下文切换快。缺点是需要考虑线程安全，隔离性低，虚拟地址空间有限。

此外还有信号问题等需要考虑。