在Linux系统中，进程作为系统资源分配的最小单位，而线程作为系统调度的最小单位。也就是说，线程更加接近执行体的概念。但是线程和进程各自有优缺点，进程方便系统资源的管理与维护，但是执行创建进程的系统开销较大，通常为线程创建系统开销的几百倍，而线程则刚好相反，线程的系统开销小，并发性更好。因此线程是一种更加“节俭”，更高效的一种机制。

**线程的创建**

在Linux系统中，线程是通过POSIX提供的线程库创建的，它与进程中的其他线程共享数据段，但线程拥有自己的线程栈以及独立的运行序列。Linux线程的创建实在内核外进行的，有POSIX提供的线程库实现。在进程创建时，内核提供的两个系统调用分别为\_clone()和fork()最终都用不同的参数对应到do\_fork()这个内核API。do\_fork()提供很多参数选项，即CLONE\_VM（共享内存空间）、CLONE\_FS（共享文件系统信息）、CLONE\_FILES（共享文件描述符表）、CLONE\_SIGHAND（共享信号句柄表）和CLONE\_PID（共享进程ID，仅对核内进程，即0号进程有效）等。当执行fork（）时，对应内核调用do\_fork()时不使用上述的任何共享属性，这也导致进程拥有独立的运行环境。相反，在通过pthread\_create()来创建线程时，则通过选项设置所有这些共享属性来调用\_\_clone()，而这些参数又全部传给内核态的do\_fork()，从而导致所创建的“进程”拥有共享的运行环境。因此在Linux系统中，线程通常被称为“轻量级进程”。

**线程的管理**

在Linux内核中，线程是以轻量级进程的形式存在的，拥有独立的进程表项；而所有的线程创建、同步、删除等操作都在核外pthread库中进行。这种模式称为基于核心轻量级进程的"一对一"线程模型，也就是一个线程实体对应一个核心轻量级进程，线程之间的管理在核外函数库中实现。内核为每一个进程构造了一个管理线程，负责处理线程相关的管理工作，这样做的好处就是线程的调度由核心完成了，而其他诸如线程取消、线程间的同步等工作，都是在核外线程库中完成的。