创建线程：

#include <pthread.h>

int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr,

void \*(\*start\_routine) (void \*), void \*arg);

thread ： 线程ID，

attr： 线程的一些属性，这些属性包括很多东西，比如线程的调度策略，线程的优先级，线程栈的大小，…

void \*(\*start\_routine) (void \*arg)：它是线程执行函数的回调。

arg：传递给线程的参数

为什么需要使用多线程：

1. 多个线程，那就是意味着我的应用程序可能会在多个CPU上执行，做到了并发编程实现。
2. 如果有阻塞的情况发送，我们完全可以把这个阻塞放在线程里，让我们程序还可以继续往下执行，

T1：阻塞

T2：执行

main()

{

…阻塞 // 线程挂起来，

}

1. 计算密集型的任务，也可以放在一个线程里去执行。

SCHED\_FIFO, SCHED\_RR：可以设置线程的优先级。

默认是SCHED\_OTHER：此时不能设置线程的优先级

非分离属性：就是子线程和主线程还有关系，这个关系体现在子线程的销毁上，主线程一定要调用pthread\_join去销毁这个线程。

分离：线程创建的时候和主线程没有任何关系，线程退出，线程资源就被销毁。

int pthread\_attr\_setdetachstate(pthread\_attr\_t \*attr, int detachstate);

int pthread\_attr\_getdetachstate(const pthread\_attr\_t \*attr, int \*detachstate);

CPU的亲缘性设置：

int pthread\_setaffinity\_np(pthread\_t thread, size\_t cpusetsize,

const cpu\_set\_t \*cpuset);

int pthread\_getaffinity\_np(pthread\_t thread, size\_t cpusetsize,

cpu\_set\_t \*cpuset);

可以通过获取CPU的数量：sched\_getcpu

栈属性

默认的线程栈的大小只有1M。

int pthread\_attr\_setstack(pthread\_attr\_t \*attr,

void \*stackaddr, size\_t stacksize);

int pthread\_attr\_getstack(const pthread\_attr\_t \*attr,

void \*\*stackaddr, size\_t \*stacksize);