pthread_create 函数创建一个新的线程，并带着一个输入变量 arg，在新线程的上下文中运行线程例程 f。能用 attr 参数来改变新创建线程的默认属性。改变这些属性已超出我们学习的范围，在我们的示例中，总是用一个为 NULL 的 attr 参数来调用 pthread_create 函数。

当 pthread_create 返回时，参数 tid 包含新创建线程的 ID。新线程可以通过调用 pthread_self 函数来获得它自己的线程 ID。

```
#include <pthread.h>

pthread_t pthread_self(void);
                                                    返回调用者的线程 ID。
```

12.3.4　终止线程

一个线程是以下列方式之一来终止的：

- 当顶层的线程例程返回时，线程会隐式地终止。
- 通过调用 pthread_exit 函数，线程会显式地终止。如果主线程调用 pthread_exit，它会等待所有其他对等线程终止，然后再终止主线程和整个进程，返回值为 thread_return。

```
#include <pthread.h>

void pthread_exit(void *thread_return);
                                                    从不返回。
```

- 某个对等线程调用 Linux 的 exit 函数，该函数终止进程以及所有与该进程相关的线程。
- 另一个对等线程通过以当前线程 ID 作为参数调用 pthread_cancel 函数来终止当前线程。

```
#include <pthread.h>

int pthread_cancel(pthread_t tid);
                                          若成功则返回 0，若出错则为非零。
```

12.3.5　回收已终止线程的资源

线程通过调用 pthread_join 函数等待其他线程终止。

```
#include <pthread.h>

int pthread_join(pthread_t tid, void **thread_return);
                                          若成功则返回 0，若出错则为非零。
```

pthread_join 函数会阻塞，直到线程 tid 终止，将线程例程返回的通用(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内存资源。

注意，和 Linux 的 wait 函数不同，pthread_join 函数只能等待一个指定的线程终止。没有办法让 pthread_wait 等待任意一个线程终止。这使得代码更加复杂，因为它迫