年，而且在所有的 Linux 系统上都可用。Pthreads 定义了大约 60 个函数，允许程序创建、杀死和回收线程，与对等线程安全地共享数据，还可以通知对等线程系统状态的变化。

图 12-13 展示了一个简单的 Pthreads 程序。主线程创建一个对等线程，然后等待它的终止。对等线程输出“Hello, world!\n”并且终止。当主线程检测到对等线程终止后，它就通过调用 exit 终止该进程。这是我们看到的第一个线程化的程序，所以让我们仔细地解析它。线程的代码和本地数据被封装在一个线程例程(thread routine)中。正如第二行里的原型所示，每个线程例程都以一个通用指针作为输入，并返回一个通用指针。如果想传递多个参数给线程例程，那么你应该将参数放到一个结构中，并传递一个指向该结构的指针。相似地，如果想要线程例程返回多个参数，你可以返回一个指向一个结构的指针。

code/conc/hello.c

```
1   #include "csapp.h"
2   void *thread(void *vargp);
3
4   int main()
5   {
6       pthread_t tid;
7       Pthread_create(&tid, NULL, thread, NULL);
8       Pthread_join(tid, NULL);
9       exit(0);
10  }
11
12  void *thread(void *vargp) /* Thread routine */
13  {
14      printf("Hello, world!\n");
15      return NULL;
16  }
```

code/conc/hello.c

图 12-13　hello.c：使用 Pthreads 的“Hello, world!”程序

第 4 行标出了主线程代码的开始。主线程声明了一个本地变量 tid，可以用来存放对等线程的 ID(第 6 行)。主线程通过调用 pthread_create 函数创建一个新的对等线程(第 7 行)。当对 pthread_create 的调用返回时，主线程和新创建的对等线程同时运行，并且 tid 包含新线程的 ID。通过在第 8 行调用 pthread_join，主线程等待对等线程终止。最后，主线程调用 exit(第 9 行)，终止当时运行在这个进程中的所有线程(在这个示例中就只有主线程)。

第 12～16 行定义了对等线程的例程。它只打印一个字符串，然后就通过执行第 15 行中的 return 语句来终止对等线程。

12.3.3　创建线程

线程通过调用 pthread_create 函数来创建其他线程。

```
#include <pthread.h>
typedef void *(func)(void *);

int pthread_create(pthread_t *tid, pthread_attr_t *attr,
                   func *f, void *arg);
```

若成功则返回 0，若出错则为非零。