**线程笔记**

目录

[要点： 2](#_Toc486350785)

[相关函数： 3](#_Toc486350786)

[创建线程pthread\_create 3](#_Toc486350787)

[等待线程结束函数pthread\_join 4](#_Toc486350788)

[线程终止函数pthread\_exit 5](#_Toc486350789)

[获取当前线程标识ID pthread\_self 6](#_Toc486350790)

# 要点：

1.多线程编译时要用 “gcc thread.c -lpthread -o thread” 调用静态库编译。

2.线程要在sleep内运行，sleep结束就无法调用线程运行了。(usleep是微秒级的延时)

3.多线程宏观上可以看作是并发的（实际只是分时复用而已），因此不需要考虑sleep切换给其他线程，系统会自动分配时间片。

4.全局变量是共同拥有的。（与进程不同）

5.创建线程传参时最好用结构体

# 相关函数：

## 创建线程pthread\_create

|  |  |
| --- | --- |
| **int pthread\_create((pthread\_t \*thread, pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg)** | |
| thread | 线程标识符； |
| attr | 线程属性设置； |
| start\_routine | 线程函数的起始地址； |
| arg | 传递给start\_routine的参数； |
|  | |
| 头文件 | #include <pthread.h> |
| 返回值 | 成功，返回0；出错，返回-1。 |
| 说明：  创建一个具有指定参数的线程。 | |
| 示例：  pthread\_t thread1；  pthread\_create(&thread1, NULL, (void \*)(&fun), (void \*)arg); | |

## 等待线程结束函数pthread\_join

|  |  |
| --- | --- |
| **int pthread\_join(pthread\_t thread, void \*\*retval)** | |
| thread | 线程标识符； |
| retval | 一个用户定义的指针，获取该结束线程的返回参数 |
|  | |
| 头文件 | #include <pthread.h> |
| 返回值 | 若是成功建立线程返回0,否则返回错误的编号 |
| 说明：  这个函数是一个线程阻塞的函数，调用它的函数将一直等待到被等待的线程结束为止，当函数返回时，被等待线程的资源被收回 | |
| 示例：  pthread\_t thread1；  pthread\_join (&thread1, (void \*) &retval); | |

## 线程终止函数pthread\_exit

|  |  |
| --- | --- |
| **void pthread\_exit(void\* retval)** | |
| retval | 一个用户定义的指针，它可以用来存储被等待线程的返回值 |
|  | |
| 头文件 | #include <pthread.h> |
| 返回值 | None |
| 说明：  终止线程,并传递参数retval的地址。 | |
| 示例：  pthread\_ exit (&retval); | |

## 获取当前线程标识ID pthread\_self

|  |  |
| --- | --- |
| **pthread\_t pthread\_self(void);** | |
| **void** | **None** |
|  | |
| 头文件 | #include <pthread.h> |
| 返回值 | 当前线程的线程ID标识 |
| 说明：  获取当前调用线程的 thread identifier(标识号) | |
| 示例：  pthread\_t thread1；  thread1 = pthread\_self(); | |