# 概述

每个线程也有一个线程号。线程号用 pthread\_t 数据类型来表示，Linux 使用无符号长整数表示。有的系统在实现 pthread\_t 的时候，用一个结构体来表示

# 线程的常用函数

**线程函数的程序在 pthread 库中，故链接时要加上参数 -lpthread**

**获取线程号**

#include <pthread.h>

pthread\_t pthread\_self(void);

功能：

获取线程号

参数：

无

返回值：

调用线程的线程 ID 。

**线程号的比较**

#include <pthread.h>

int pthread\_equal(pthread\_t t1, pthread\_t t2);

功能：

判断线程号 t1 和 t2 是否相等

参数：

**t1，t2**：待判断的线程号。

返回值：

相等：  非 0

不相等：0

**线程的创建**

#include <pthread.h>

int pthread\_create(

    pthread\_t \*thread,

    const pthread\_attr\_t \*attr,

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

    void \*arg );

功能：

创建一个线程。

参数：

**thread**：用于保存返回的线程标识符地址

**attr**：线程属性结构体地址，通常设置为 NULL。

**start\_routine**：线程函数的入口地址。

**arg**：传给线程函数的参数。

返回值：

成功：0

失败：非 0

线程依赖进程存在的，共享进程的资源，如果创建线程的进程结束了，线程也就结束了。

示例：

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

void \*thread\_1(void \*arg)

{

printf("this is my new thread1\n");

    return NULL;

}

void \*thread\_2(void \*arg)

{

    printf("this is my new thread2\n");

    return NULL;

}

int main(int argc, char \*argv[])

{

pthread\_t tid1, tid2;

    //创建两个线程

    pthread\_create(&tid1, NULL, thread\_1, NULL);

pthread\_create(&tid2, NULL, thread\_2, NULL);

    while (1)

    {

        printf("the main thread\n");

        sleep(1);

}

    return 0;

}

**回收线程资源**

#include <pthread.h>

int pthread\_join(pthread\_t thread, void \*\*retval);

功能：

等待线程结束（此函数会阻塞），并回收线程资源

参数：

**thread**：被等待的线程号。  
 **retval**：用来存储线程退出状态的指针的地址。

返回值：

成功：0

失败：非 0

示例代码如下：

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

void \*thead(void \*arg)

{

    static int num = 123; //静态变量

    printf("after 2 seceonds, thread will return\n");

    sleep(2);

    return &num;

}

int main(int argc, char \*argv[])

{

    pthread\_t tid;

    int ret = 0;

    void \*value = NULL;

    // 创建线程

    ret = pthread\_create(&tid, NULL, thead, NULL);

    if(ret != 0){ //创建失败

        perror("pthread\_create");

    }

    // 等待线程号为 tid 的线程，如果此线程结束就回收其资源

    // &value保存线程退出的返回值

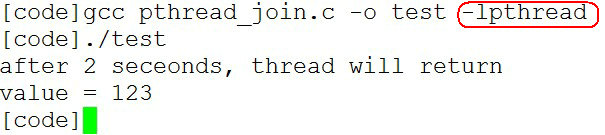
    pthread\_join(tid, &value);

    printf("value = %d\n", \*( (int \*)value ) );

    return 0;

}

运行结果如下：



**线程分离**

#include <pthread.h>

int pthread\_detach(pthread\_t thread);

功能：

使调用线程与当前进程分离，**线程分离的目的**是将线程资源的回收工作交由系统完成

参数：

**thread**：线程号。

返回值：

成功：0

失败：非 0

**线程退出**

#include <pthread.h>

void pthread\_exit(void \*retval);

功能：

退出调用线程。

参数：

**retval**：存储线程退出状态的指针。

返回值：

无

示例代码如下：

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

void \*thread(void \*arg)

{

    static int num = 123; //静态变量

    int i = 0;

    while (1)

    {

        pthread\_exit((void \*)&num);

        // return &num;

}

    return NULL;

}

int main(int argc, char \*argv[])

{

    int ret = 0;

    pthread\_t tid;

void \*value = NULL;

    // 创建线程

    ret = pthread\_create(&tid, NULL, thread, NULL);

    if (ret != 0)

    {

        perror("pthread\_create");

}

    // 等待线程退出

pthread\_join(tid, &value);

printf("value = %d\n", \*(int \*)value);

    return 0;

}