# 线程

#include <pthread.h>  
  
int pthread\_create(pthread\_t \*tid, pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg);

#### 功能

1. 创建一个线程
2. 新线程从start\_routine开始执行
3. 新线程的ID保存在tid指向的位置
4. 成功返回0

#### about start\_routine()

使用 **指针函数（非函数指针）**，用的时候取函数地址。

void \*foo(void \*arg){  
 //do something...  
}  
pthread\_create(&tid, NULL, &foo, NULL);

补充：函数指针是指向函数的指针变量，即本质是一个指针变量。 void (*p)(void* arg); p=foo; /\* 将func函数的首地址赋给指针p \*/

#### about arg

==>main()中：  
int integer = 6;  
int arr[3] = {1,2,3};  
char \*string = "hello world";  
void \*arg = (void \*)&integer;  
void \*arr\_arg = (void \*)arr;  
void \*s\_arg = (void \*)string;  
==>传参：  
arg, arr\_arg, s\_arg  
==>start\_routine()中：  
int a = \*(int \*)arg;  
int \*arr = (int \*)arg;  
char \*s = arg;

## 等待线程

#include <pthread.h>  
  
int pthread\_join(pthread\_t tid, void \*\*result);

1. 等待线程结束
2. result存取返回值（但start\_routine函数还可以是void)
3. 成功返回0

#### 线程返回值

//线程入口函数返回类型为void \*\*类型的结果  
void \*start\_routine(void \*arg)  
{  
 void \*result;  
 ...  
 return result;  
}  
//等待线程函数pthread\_join获取线程的返回结果  
void \*result;  
pthread\_join(tid, &result);

#### 初始化互斥量

#include <pthread.h>  
  
int pthread\_mutex\_init(pthread\_mutex\_t \*mutex, pthread\_mutexattr\_t \*attr);//如果attr等于NULL，则使用缺省的属性进行初始化  
int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex); //销毁互斥量  
int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex);// 加锁  
int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex);// 解锁

使用互斥量 1. 线程在访问global变量前先对互斥量加锁 2. 线程在访问global变量后再对互斥量解锁

## 条件变量

原型

#include <pthread.h>  
  
// 初始化  
int pthread\_cond\_init(pthread\_cond\_t \*cond, pthread\_condattr\_t \*attr);//初始化条件变量,如果attr等于NULL，则使用缺省的属性进行初始化  
int pthread\_cond\_destroy(pthread\_cond\_t \*cond);//释放条件变量

#### 等待

int pthread\_cond\_wait(pthread\_cond\_t \*cond, pthread\_mutex\_t \*mutex);//阻塞当前线程的运行

|  |  |
| --- | --- |
| 参数 | 解释 |
| cond | 当前线程在条件变量上阻塞 |
| mutex | 当前线程阻塞时所在的临界区 |

#### 唤醒线程

int pthread\_cond\_signal(pthread\_cond\_t \*cond);//唤醒阻塞在条件变量上的一个线程  
int pthread\_cond\_broadcast(pthread\_cond\_t \*cond);//唤醒阻塞在条件变量上的所有线程

### 条件变量初始化

#include <pthread.h>  
  
int pthread\_cond\_init(pthread\_cond\_t \*cond, pthread\_condattr\_t \*attr);//初始化条件变量,如果attr等于NULL，则使用缺省的属性进行初始化  
int pthread\_cond\_destroy(pthread\_cond\_t \*cond);//释放条件变量  
  
//等待  
int pthread\_cond\_wait(pthread\_cond\_t \*cond, pthread\_mutex\_t \*mutex);//阻塞当前线程的运行  
/\*  
参数  
> cond 当前线程在条件变量上阻塞  
> mutex 当前线程阻塞时所在的临界区  
\*/  
  
//唤醒线程  
int pthread\_cond\_signal(pthread\_cond\_t \*cond);//唤醒阻塞在条件变量上的一个线程  
int pthread\_cond\_broadcast(pthread\_cond\_t \*cond);//唤醒阻塞在条件变量上的所有线程