### 1 线程的创建

#### 头文件

  #include <pthread.h>

方法

int pthread\_create(pthread\_t \*thread, constpthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg);

#### 参数解释

参数thread为线程标识符

attr为线程属性设置

start\_routine为线程函数起始地址

arg为传递给start\_routine的参数

#### 返回值

创建线程成功时，返回0

创建线程失败，返回错误号

#### 原理解释

pthread\_create是通过系统调用clone来实现的，clone是Linux特有的系统调用，类似进程创建的系统调用fork（）

### 2 获取线程标识符

#### 头文件

 #include <pthread.h>

#### 方法

pthread\_t pthread\_self(void);

#### 返回值

返回线程的标识符

#### 原理解释

pthread\_tpthread\_self（）返回调用的线程的标识符。每个线程都有自己的线程标识符，以便在进程内区分，线程标识符在pthread\_create创建时产生。

### 3线程等待

#### 头文件

 #include <pthread.h>

#### 方法

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

#### 方法参数

Thread表示为被等待的线程标识符

retval为用户定义的指针，存放被等待线程的返回值

#### 返回值

#### 原理解释

pthread\_join（）将调用它的线程阻塞，一直等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。

### 4 线程退出

#### 头文件

 #include <pthread.h>

#### 方法

 void pthread\_exit(void \*retval)------ 终止调用线程

  int pthread\_cancel(pthread\_t thread);------------ 终止由参数thread指定的线程

#### 方法参数

retval为线程的返回值

thread为指定要终止的线程标识符

#### 返回值

#### 原理解释

pthread\_join（）将调用它的线程阻塞，一直等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。

### 5 线程实例

|  |
| --- |
| #include <stdio.h>  #include <pthread.h>  void thread(void)  {  int i;  for (i=0;i<3;i++)  printf("This is a pthread.\n");  }    int main(void)  {  pthread\_t id;  int i,ret;  ret=pthread\_create(&id,NULL,(void \*) thread,NULL);  if (ret!=0)  {  printf ("Create pthread error!\n");  exit (1);  }  for (i=0;i<3;i++)  printf("This is the main process.\n");  pthread\_join(id,NULL);  return(0);  } |