


第7章 线程

——线程基本操作（2）

主　讲：王小银

中国大学MOOC

线程挂起

函数名称	sleep
函数功能	睡眠当前线程直到超时
头文件	#include <unistd.h>
函数原型	unsigned int sleep(unsigned int seconds);
参数	seconds： 要睡眠的秒数。
返回值	0： 成功；， >0: 睡眠时被另一个信号打断（返回剩余的秒数）。

中国大学MOOC

线程挂起

函数名称	usleep
函数功能	睡眠当前线程直到超时
头文件	#include <unistd.h>
函数原型	int usleep(useconds_t usec);
参数	usec：　挂起时间，单位是微秒。
返回值	0：　成功； >0:　睡眠时被另一个信号打断（返回剩余的微秒数）。

中国大学MOOC

线程挂起

函数名称	nanosleep
函数功能	睡眠当前线程直到超时
头文件	#include <time.h>
函数原型	int nanosleep(const struct timespec *req, struct timespec *rem);
参数	req：　　请求挂起的时间； rem:　　被信号打断后的剩余时间。
返回值	0:　　成功； -1:　　出错，并且errno被设置。

中国大学MOOC

示例程序

```
#include<stdio.h>
#include<unistd.h>
#include<time.h>

struct timespec tt={5,100};

int main(void)
{           printf("start!\n");
            sleep(5);
            printf("end!\n");
            printf("start!\n");
            for(int i=0;i<10;i++)
                        usleep(500000);
            printf("end!\n");
            printf("start\n");
                        nanosleep(&tt,NULL);
            printf("end!\n");
            return 0;
}
```

```
struct timespec
{
   time_t  tv_sec;        /* 秒seconds */
   long    tv_nsec;       /* 纳秒nanoseconds */
};
```

线程的分离

线程状态：

- 可结合的（joinable）
- 分离的（detached）

函数名称	pthread_detach
函数功能	使线程处于分离状态
头文件	#include <pthread.h>
函数原型	int pthread_detach(pthread_t threadID);
参数	threaID：　要分离的线程id。
返回值	0：　成功； 错误号：　失败。

中国大学MOOC

线程的一次性初始化

函数名称	pthread_once
函数功能	一次性初始化
头文件	#include <pthread.h>
函数原型	int pthread_once(pthread_once_t *once_control, void (*init_routine)(void));
参数	once_control：　决定init_routine函数是否执行； init_routine：　初始化要执行的函数。
返回值	0：　成功； 错误号：　失败。

如何进行一次性初始化

- 定义一个pthread_once_t变量，使用宏PTHREAD_ONCE_INIT对该变量初始化。

```
pthread_once_t once_control = PTHREAD_ONCE_INIT;
void init_routine()
{
   //初始化互斥量
   //初始化读写锁
   ......
}
```

- 调用pthread_once函数：

```
int pthread_once(pthread_once_t* once_control, void (*init_routine)(void));
```

线程的私有数据

函数名称	pthread_key_create
函数功能	创建一个对同一进程内所有线程都可见的线程私有数据键
头文件	#include <pthread.h>
函数原型	int pthread_key_create(pthread_key_t *key, void (*destructor)(void*));
参数	key：　　线程私有数据键； destructor：　　线程退出时调用的函数。
返回值	0：　　成功； 错误号：　　失败。

线程的私有数据

函数名称	pthread_setspecific
函数功能	为指定键值设置线程私有数据
头文件	#include <pthread.h>
函数原型	int pthread_setspecific(pthread_key_t key, const void *value);
参数	key：　　线程私有数据键； value：　　和key 关联起来的数据的地址。
返回值	0：　　成功； 错误号：　　失败。

线程的私有数据

函数名称	pthread_getspecific
函数功能	从指定键读取线程的私有数据
头文件	#include <pthread.h>
函数原型	void *pthread_getspecific(pthread_key_t key);
参数	key：　　需要获取数据的键。
返回值	0：　　成功； 错误号：　　失败。

中国大学MOOC

线程的私有数据

函数名称	pthread_key_delete
函数功能	删除线程私有数据键
头文件	#include <pthread.h>
函数原型	int pthread_key_delete(pthread_key_t key);
参数	key：　　线程私有数据键。
返回值	0：　　成功； 错误号：　　失败。

中国大学MOOC

谢谢大家！

中国大学MOOC