


第8章 线程间的同步机制
——信号

主　讲：王小银

- 信号是一种IPC通信的形式。
- 信号是一种异步通知进程或同一进程中某个指定线程的方式。
- 线程在信号操作时具有以下特性：
  - 每个线程可以先别的线程发送信号
  - 每个线程都可以设置自己的阻塞集合
  - 每个线程需要设置针对某信号的处理方式
  - 如果别的进程向当前进程发来一个信号，具体由哪个线程去处理，是未知的

向线程发送信号

函数名称	pthread_kill
函数功能	向同一个进程中指定的线程（包括自己）发送信号
头文件	#include <signal.h>
函数原型	int pthread_kill (pthread_t __threadid, int __signo);
参数	__threadid：　　传送信号的目标线程ID； __signo：　　表示要传送给线程的信号，如果为0，则用于检测该线程是否存在，而不发送信号。
返回值	0：　　成功； 非0：　　失败。

调用线程的信号掩码

函数名称	pthread_sigmask
函数功能	更改或检查调用线程的信号掩码
头文件	#include <signal.h>
函数原型	int pthread_sigmask (int __how,const __sigset_t *__restrict __newmask, __sigset_t *__restrict __oldmask);
参数	__how：更改调用线程的信号掩码； __newmask：为NULL时how没有意义，非NULL时通过how指示如何修改线程阻塞信号集； __oldmask：当前线程阻塞集。
返回值	0：成功； 非0：失败。

```
#define   SIG_BLOCK     0
#define   SIG_UNBLOCK   1
#define   SIG_SETMASK   2
```

how的取值有三种：

- SIG_BLOCK
- SIG_UNBLOCK
- SIG_SETMASK

示例程序

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>

void *func1()
{
   sleep(1);
   printf("thread 1 （id： %u） quit.\n",pthread_self());
   pthread_exit((void *)0);
}

void *func2()
{
   sleep(5);
   printf("thread 2 （id： %u） quit.\n",pthread_self());
   pthread_exit((void *)0);
}
```

```
void test_pthread(pthread_t tid)
{   int pthread_kill_err;
     pthread_kill_err = pthread_kill(tid,0);
     if(pthread_kill_err == ESRCH)
        printf("ID为%u的线程不存在或者已经退出。\n",tid);
    else if(pthread_kill_err == EINVAL)
        printf("发送信号非法。\n");
     else       printf("ID为%u的线程目前仍然存活。\n",tid);
}
int main(void)
{  int ret;
   pthread_t tid1,tid2;
   pthread_create(&tid1,NULL,func1,NULL);
   pthread_create(&tid2,NULL,func2,NULL);
    sleep(3);
    test_pthread(tid1);
   test_pthread(tid2);
   return 0;
}
```

示例程序

```
 void sigusr_handle (int arg)
{      printf("thread(id=%u) catch signal %d\n", pthread_self(), arg);
}

void * th1_handle(void *arg)
{   int i;

    sigset_t set;
    sigfillset(&set);
    sigdelset(&set, SIGUSR2);
    pthread_sigmask(SIG_SETMASK, &set, 0);

    signal(SIGUSR1, sigusr_handle);

    for (i=0; i<5; i++) {
        printf("this is in thread 1(id=%u)\n", pthread_self());
        pause();
    }
}
```

```
void * th2_handle(void *arg)
{
    int i;
    signal(SIGUSR2, sigusr_handle);
    for (i=0; i<5; i++) {
        printf("this is in thread 2(id=%u)\n", pthread_self());
        pause();
    }
}
```

示例程序

```
int main(void)
{  pthread_t mythread1, mythread2;
   int ret;
   ret = pthread_create(&mythread1, NULL, th1_handle, NULL);
   if (ret != 0)       printf("create thread failed!\n");
   ret = pthread_create(&mythread2, NULL, th2_handle, NULL);
   if (ret != 0)
      printf("create thread failed!\n");
   sleep(1);
   ret = pthread_kill(mythread1, SIGUSR1);
   if (ret != 0) {
      printf("pthread_kill SIGUSR1 failed!\n");
      exit(1);
   }
   ret = pthread_kill(mythread1, SIGUSR2);
   if (ret != 0) {
      printf("pthread_kill SIGUSR2 failed!\n");
      exit(1);
   }
```

```
   ret = pthread_kill(mythread2, SIGUSR1);
   if (ret != 0) {
      printf("pthread_kill SIGUSR1 failed!\n");
      exit(1);
   }
    sleep(1);
    ret = pthread_kill(mythread2, SIGUSR2);
   if (ret != 0) {
      printf("pthread_kill SIGUSR2 failed!\n");
      exit(1);
   }
    ret = pthread_kill(mythread1, SIGKILL);
   if (ret != 0) {
      printf("pthread_kill SIGKILL failed!\n");
      exit(1);
   }
   pthread_join(mythread1, NULL);
   pthread_join(mythread2, NULL);
   return 0;
}
```

谢谢大家！