多线程编程初步

线程是什么

- 线程是在一个程序内部可以被操作系统调度并发运行的任务
- 线程与进程的相同点
  - 可以被操作系统调度并发运行
- 线程相对于进程而言的优点
  - 在一个程序中需要并发处理多个任务时（例如需要并发监控 2 个不同的 fifo ），以多线程的方式编程更符合人的思维
  - 多线程编程使得多个任务间的通信更加直观和方便（例如可以使用共享的全局变量）
  - 多线程的切换开销更小

线程的创建与交替运行演示

- 创建线程
  - int pthread_create(pthread_t *thread, pthread_attr_t *attr, void (*func)(void *), void *arg)
  - pthread_create(&thrd1, NULL, (void *)task1, (void *)&g1);
  - pthread_t thrd1; // 被创建线程的标识
  - void task1(int *counter); // 被创建线程的程序代码
  - int g1 = 0; //task1 的参数
- 线程自行结束
  - return from thread routine
  - call pthread_exit(void *retval) // retval 是终止线程的返回值
- 等待指定线程结束
  - int pthread_join(pthread_t th, void **thread_return);
  - pthread_join(thrd1, NULL);
  - th 是要等待结束的线程的标识
  - 指针 thread_return 指向的位置存放的是终止线程的返回值

杀死线程

- 杀死线程
  - pthread_cancel(thrd1);
  - thrd1 是要被杀死的线程的标识
- 设置线程属性以拒绝被杀死
  - pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, NULL);
  - pthread_setcancelstate(PTHREAD_CANCEL_ENABLE, NULL);（默认属性）
- 设置线程属性以决定被杀死的时机
  - pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
  - pthread_setcanceltype(PTHREAD_CANCEL_DEFDERRED, NULL);（默认属性）
  - pthread_testcancel(void)

多线程的同步

- 多线程的同步采用加锁的机制，类似于多进程间的信号量机制
- 在主线程中初始化锁
  - pthread_mutex_t mutex;
  - pthread_mutex_init(&mutex, NULL);
- 在编译时初始化锁
  - pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
- 在访问共享对象前进行加锁操作
  - pthread_mutex_lock(&mutex)
- 在访问共享对象后进行解锁操作
  - pthread_mutex_unlock(&mutex)

加锁方式的变体

- pthread_mutex_trylock(&mutex) 与 pthread_mutex_lock(&mutex) 的不同点
  - 如果加锁时，锁未被锁定，这二者行为一样
  - 如果加锁时，锁已被锁定，这后者被阻塞到锁被重新打开；而前者则会立即返回，但返回值为 EBUSY