**线程的创建和终止**

1. 线程的创建：

int pthread\_create(pthread\_t \*thead\_id,

const pthread\_attr\_t \*attr,

void \*(\*pfun)(void \*),

void \*arg)

参数thread\_id：指向新线程ID指针

参数attr：指向线程属性结构体的指针，一般情况下可以传入NULL指针。

参数pfun：线程的执行函数。

参数arg：传给线程执行函数的参数。

1. 线程终止：
2. int pthread\_exit(void \*rval\_ptr) //此函数在线程内部调用，终止线程（调用这个函数前应该先释放动态分配的内存以避免内存泄露，以及释放互斥量和信号量等以避免死锁）。参数reval\_ptr会返回给pthread\_join()函数，所以reval\_ptr应该指向动态分配的内存或者全局变量，pthread\_join()函数读取reval\_ptr的内容后，应该手动释放reval\_ptr指向的内存（如果reval\_ptr指向的是动态分配的内存）。
3. int pthread\_join(pthread\_t thread, void \*\*rval\_ptr) //其它线程阻塞等待调用pthread\_exit()函数的线程的退出。
4. 某线程取消其它线程
5. int pthread\_cancel(pthread\_t tid)
6. void pthread\_cleanup\_push(void (\*rtn)(void \*), void \*arg) //写在线程里，作为线程函数的第一条语句。当然，可以多次push多个清除函数。
7. void pthread\_cleanup\_pop(int execut) //写在线程里，作为线程函数的最后一条语句。

**线程的同步**

1. 互斥量

int pthread\_mutex\_init(pthread\_mutex\_t \*mutex)

int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex)

int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex) //如果上锁失败则阻塞。

int pthread\_mutex\_trylock(pthread\_mutex\_t \*mutex) //如果上锁失败则不会阻塞而是返回错误码EBUSY。

int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex)

1. 读写锁

int pthread\_rwlock\_init(pthread\_rwlock\_t \*rwlock, const pthread\_rwlockattr\_t \*attr)

int pthread\_rwlock\_destroy(pthread\_rwlock\_t \*rwlock)

int pthread\_rwlock\_rdlock(pthread\_rwlock\_t \*rwlock)

int pthread\_rwlock\_wrlock(pthread\_rwlock\_t \*rwlock)

int pthread\_rwlock\_unlock(pthread\_rwlock\_t \*rwlock)

int pthread\_rwlock\_tryrdlock(pthread\_rwlock\_t \*rwlock)

int pthread\_rwlock\_trywrlock(pthread\_rwlock\_t \*rwlock)

读写锁一次只能有一个线程写，而可以有多个线程读。

1. 条件变量

int pthread\_conf\_init(pthread\_cond\_t \*cond, pthread\_condattr\_t \*attr)

int pthread\_conf\_destroy(pthread\_cond\_t \*cond)

int pthread\_cond\_wait(pthread\_cond\_t \*cond, pthread\_mutex\_t \*mutex)

int pthread\_cond\_timewait(pthread\_cond\_t \*cond,

pthread\_mutex\_t \*mutex,

const struct timespec \*timeout)

int pthread\_cond\_signal(pthread\_cond\_t \*cond)

int pthread\_cond\_broadcast(pthread\_cond\_t \*cond)

条件变量的作用是为了实现休眠和唤醒的功能，它总是和互斥量一起使用《unix环境高级编程》311页。pthread\_cond\_wait()主要做2件事：释放互斥量然后休眠；被唤醒然后获取互斥量。pthread\_cond\_signal()用来唤醒一个等待条件的线程，pthread\_cond\_broadcast()用来唤醒所有等待条件的线程，让这些线程竞争互斥量。

1. 信号量

信号量可以在进程间同步，也可以在线程中进行同步。