条件变量

应用场景：生产者消费者问题，是线程同步的一种手段。

必要性：为了实现等待某个资源，让线程休眠。提高运行效率

int pthread\_cond\_wait(pthread\_cond\_t \*restrict cond,

pthread\_mutex\_t \*restrict mutex);

int pthread\_cond\_timedwait(pthread\_cond\_t \*restrict cond,

pthread\_mutex\_t \*restrict mutex,

const struct timespec \*restrict abstime);

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

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

使用步骤：

初始化：

静态初始化

pthread\_cond\_t  cond = PTHREAD\_COND\_INITIALIZER;   //初始化条件变量

pthread\_mutex\_t  mutex = PTHREAD\_MUTEX\_INITIALIZER; //初始化互斥量

或使用动态初始化

pthread\_cond\_init(&cond);

生产资源线程：

pthread\_mutex\_lock(&mutex);

开始产生资源

pthread\_cond\_sigal(&cond); //通知一个消费线程

或者

pthread\_cond\_broadcast(&cond); //广播通知多个消费线程

pthread\_mutex\_unlock(&mutex);

消费者线程：

pthread\_mutex\_lock(&mutex);

while （如果没有资源）{ //防止惊群效应

pthread\_cond\_wait(&cond, &mutex);

}

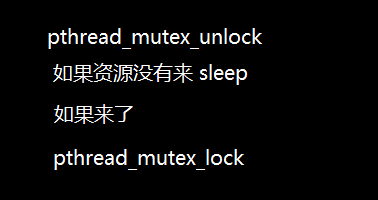
有资源了，消费资源

pthread\_mutex\_unlock(&mutex);

注意：

1 pthread\_cond\_wait(&cond, &mutex)，在没有资源等待是是先unlock 休眠，等资源到了，再lock

所以pthread\_cond\_wait he pthread\_mutex\_lock 必须配对使用。



2 如果pthread\_cond\_signal或者pthread\_cond\_broadcast 早于 pthread\_cond\_wait ，则有可能会丢失信号。

3 pthead\_cond\_broadcast 信号会被多个线程收到，这叫线程的惊群效应。所以需要加上判断条件while循环。