1 信号量

信号量可以有n把锁。

依赖的头文件

#include <semaphore.h>

函数声明

sem\_t 表示信号量

int sem\_init(sem\_t \*sem, int pshared, unsigned int value);

|  |  |
| --- | --- |
| 名称： | sem\_init |
| 功能： | initialize an unnamed semaphore,初始化信号量sem\_t，初始化的时候可以指定信号量的初始值，以及是否可以在多进程间共享。 |
| 头文件： | #include <semaphore.h> |
| 函数原形： | int sem\_init(sem\_t \*sem, int pshared, unsigned int value); |
| 参数： |  |
| 返回值： | sem\_init() returns 0 on success; on error, -1 is returned, and errno is set to indicate the error. |

int sem\_wait(sem\_t \*sem);

int sem\_trywait(sem\_t \*sem);

|  |  |
| --- | --- |
| 名称： | sem\_wait  sem\_trywait |
| 功能： | lock a semaphore 一直阻塞等待直到信号量 > 0. |
| 头文件： | #include <semaphore.h> |
| 函数原形： | int sem\_wait(sem\_t \*sem);  int sem\_trywait(sem\_t \*sem); |
| 参数： |  |
| 返回值： | All of these functions return 0 on success; on error, the value of the semaphore is left unchanged, -1 is returned, and errno is set to indicate the error. |

int sem\_timedwait(sem\_t \*sem, const struct timespec \*abs\_timeout);

|  |  |
| --- | --- |
| 名称： | sem\_timedwait |
| 功能： | lock a semaphore，阻塞等待若干时间直到信号量 > 0 |
| 头文件： | #include <semaphore.h> |
| 函数原形： | int sem\_timedwait(sem\_t \*sem, const struct timespec \*abs\_timeout); |
| 参数： |  |
| 返回值： | All of these functions return 0 on success; on error, the value of the semaphore is left unchanged, -1 is returned, and errno is set to indicate the error. |

int sem\_post(sem\_t \*sem);

|  |  |
| --- | --- |
| 名称： | sem\_post |
| 功能： | unlock a semaphore，使信号量加1。 |
| 头文件： | #include <semaphore.h> |
| 函数原形： | int sem\_post(sem\_t \*sem); |
| 参数： |  |
| 返回值： | sem\_post() returns 0 on success; on error, the value of the semaphore is left unchanged, -1 is returned, and errno is set to indicate the error. |

int sem\_destroy(sem\_t \*sem);

|  |  |
| --- | --- |
| 名称： | sem\_destroy |
| 功能： | destroy an unnamed semaphore 释放信号量。和sem\_init对应 |
| 头文件： | #include <semaphore.h> |
| 函数原形： | int sem\_destroy(sem\_t \*sem); |
| 参数： |  |
| 返回值： | sem\_destroy() return 0 on success;on error,-1 is returned,an errno is set to indicate the error. |

案例说明：

|  |
| --- |
| #include <stdlib.h>  #include <pthread.h>  #include <stdio.h>  #include <unistd.h>  #include <semaphore.h>  #define NUM 5  int queue[NUM];  sem\_t blank\_number,product\_number;  void \*producer(void \*arg)  {  int p = 0;  while(1) {  //blank\_num = 5 生产者最多生产5个  //一直阻塞等待信号量大于0  sem\_wait(&blank\_number);  queue[p] = rand() % 1000 + 1;  printf("Produce %d\n",queue[p]);  //product\_number = 0 ->1  //使信号量加1  sem\_post(&product\_number);  p = (p + 1) % NUM;  sleep(rand() % 5);  }  }  void \*consumer(void \*arg) {  int c = 0;  while (1) {  //等待信号量大于0  sem\_wait(&product\_number);  printf("Consume %d\n",queue[c]);  queue[c] = 0;  //使信号量加1  sem\_post(&blank\_number);  c = (c + 1) % NUM;  sleep(rand() % 5);  }  }  int main(int argc ,char \*argv[])  {  pthread\_t pid,cid;  //将blank\_num信号量初始化的值为5  sem\_init(&blank\_number,0,NUM);  //将product\_number信号量初始化的值变为0  sem\_init(&product\_number,0,0);  pthread\_create(&pid,NULL,producer,NULL);  pthread\_create(&cid,NULL,consumer,NULL);  pthread\_join(pid,NULL);  pthread\_join(cid,NULL);  sem\_destroy(&blank\_number);  sem\_destroy(&product\_number);  return 0;  } |

运行结果：

