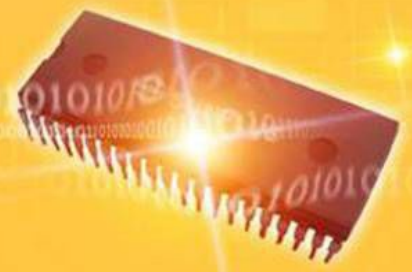
嵌入式系统工程师



多任务互斥和同步

大纲

- 互斥和同步概述
- 互斥锁
- 信号量

大纲

- 互斥和同步概述
- 互斥锁
- 信号量

多任务同步与互斥

在多任务操作系统中，同时运行的多个任务可能

- 都需要访问/使用同一种资源
- 多个任务之间有依赖关系，某个任务的运行依赖于另一个任务

同步和互斥就是用于解决这两个问题的。

多任务同步与互斥

- 互斥:

  一个公共资源同一时刻只能被一个进程或线程使用，多个进程或线程不能同时使用公共资源。POSIX标准中进程和线程同步和互斥的方法,主要有信号量和互斥锁两种方式。

- 同步:

  两个或两个以上的进程或线程在运行过程中协同步调，按预定的先后次序运行。

大纲

- 互斥和同步概述
- 互斥锁
- 信号量

线程同步与互斥

- 互斥锁(mutex)

  mutex是一种简单的加锁的方法来控制对共享资源的访问，mutex只有两种状态,即上锁(lock)和解锁(unlock)。

  - 在访问该资源前，首先应申请mutex，如果mutex处于unlock状态，则会申请到mutex并立即lock；如果mutex处于lock状态，则默认阻塞申请者。
  - unlock操作应该由lock者进行。

线程同步与互斥

- mutex用pthread_mutex_t数据类型表示，在使用互斥锁前,必须先对它进行初始化。
- 静态分配的互斥锁：

```
pthread_mutex_t mutex =
PTHREAD_MUTEX_INITIALIZER;
```

- 动态分配互斥锁:

```
pthread_mutex_t mutex;
pthread_mutex_init(&mutex, NULL);
```

在所有使用过此互斥锁的线程都不再需要使用时候，应调用pthread_mutex_destroy销毁互斥锁。

线程同步与互斥

- #include <pthread.h>

```
int pthread_mutex_init(
        pthread_mutex_t *mutex,
        const pthread_mutexattr_t *attr);
```

功能：

初始化一个互斥锁。

参数：

mutex：互斥锁地址。

attr：互斥锁的属性，NULL为默认的属性。

返回值：

成功返回0，失败返回非0。

线程同步与互斥

- #include <pthread.h>
  int pthread_mutex_lock(pthread_mutex_t *mutex);
  功能：
  对互斥锁上锁，若已经上锁，则调用者一直阻塞到互斥锁解锁。
  参数：
  mutex：互斥锁地址。
  返回值：
  成功返回0，失败返回非0。

线程同步与互斥

- #include <pthread.h>
  int pthread_mutex_trylock(
  pthread_mutex_t *mutex);

  功能：

  对互斥锁上锁，若已经上锁，则上锁失败，函数立即返回。

  参数：

  mutex：互斥锁地址。

  返回值：

  成功返回0，失败返回非0。

线程同步与互斥

- #include <pthread.h>

  int pthread_mutex_unlock(
  pthread_mutex_t * mutex);

  功能：

  对指定的互斥锁解锁。

  参数：

  mutex：互斥锁地址。

  返回值：

  成功返回0，失败返回非0。

线程同步与互斥

- #include <pthread.h>
  int pthread_mutex_destroy(
  pthread_mutex_t *mutex);

  功能：

  销毁指定的一个互斥锁。

  参数：

  mutex：互斥锁地址。

  返回值：

  成功返回0，失败返回非0。

  例：01_pthread_mutex.c

大纲

- 互斥和同步概述
- 互斥锁
- 信号量

线程同步与互斥

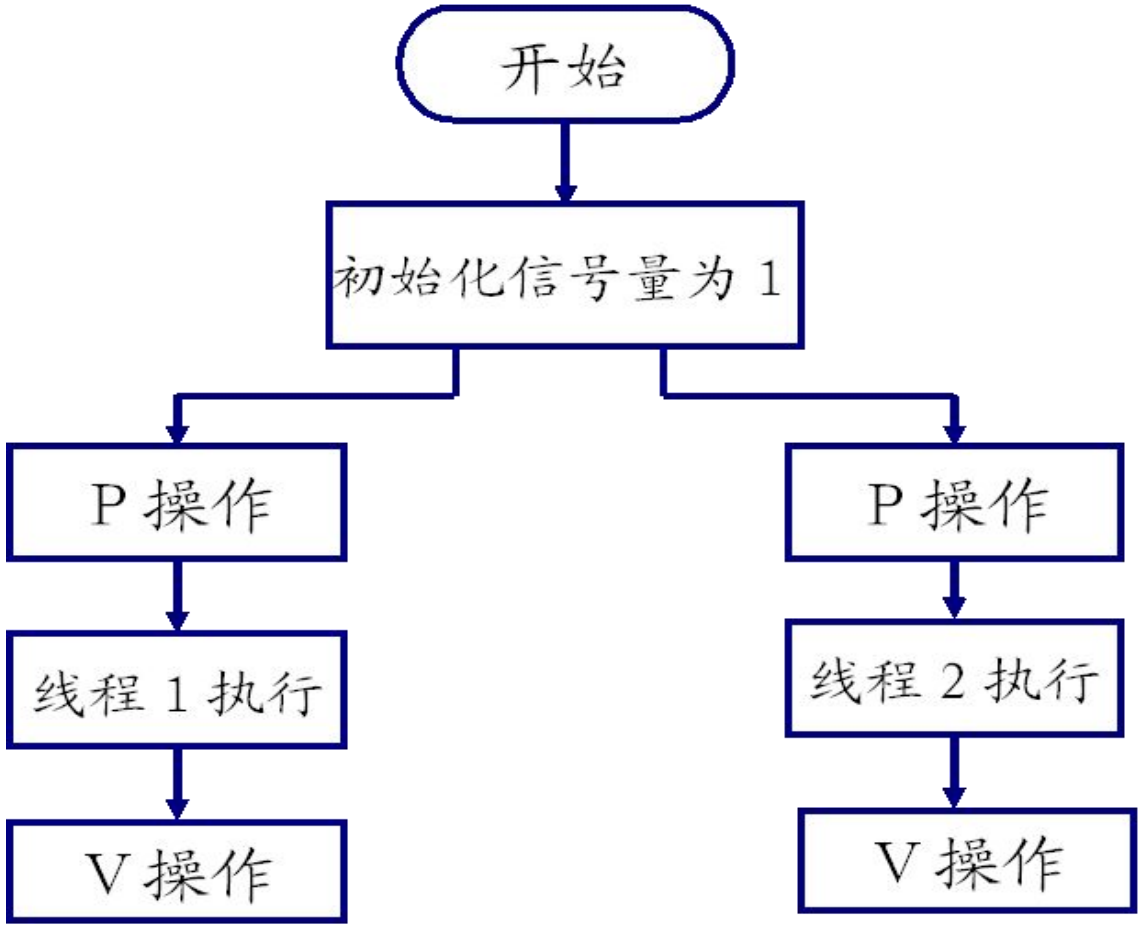
- 信号量广泛用于进程或线程间的同步和互斥，信号量本质上是一个非负的整数计数器，它被用来控制对公共资源的访问。
- 编程时可根据操作信号量值的结果判断是否对公共资源具有访问的权限，当信号量值大于 0 时，则可以访问，否则将阻塞。

线程同步与互斥

- ＰＶ原语是对信号量的操作，一次Ｐ操作使信号量sem减１，一次Ｖ操作使信号量sem加１。
- 信号量主要用于进程或线程间的同步和互斥这两种典型情况。
  - 若用于互斥，几个进程（或线程）往往只设置一个信号量。
  - 若用于同步操作，往往会设置多个信号量，并且安排不同的初始值，来实现它们之间的执行顺序。

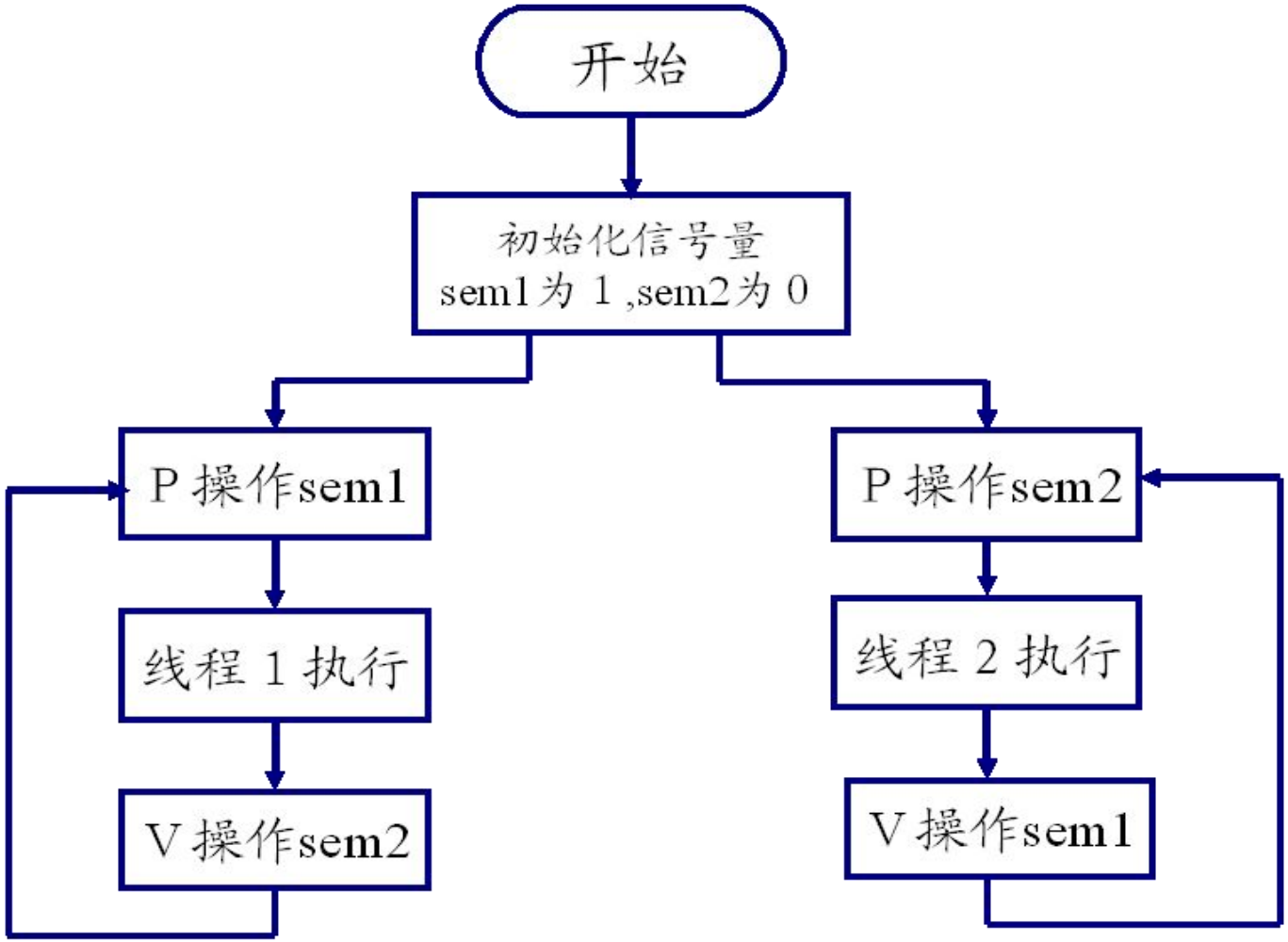
线程同步与互斥

- 信号量用于互斥



线程同步与互斥

- 信号量用于同步



线程同步与互斥

- #include <semaphore.h>

```
int sem_init(sem_t *sem, int pshared,
                                unsigned int value);
```

功能：

创建一个信号量并初始化它的值。

参数：

- sem：信号量的地址。
- pshared：等于0，信号量在线程间共享；不等于0，信号量在进程间共享。
- value：信号量的初始值。

返回值：

成功返回0，失败返回-1。

线程同步与互斥

- #include <semaphore.h>

  int sem_wait(sem_t *sem);

  功能：

  将信号量的值减1，若信号量的值小于0，此函数会引起调用者阻塞。

  参数：

  sem：信号量地址。

  返回值：

  成功返回0，失败返回-1。

线程同步与互斥

- #include <semaphore.h>

  int sem_trywait(sem_t *sem);

  功能：

  将信号量的值减1，若信号量的值小于0，则对信号量的操作失败，函数立即返回。

  参数：

  sem：信号量地址。

  返回值：

  成功返回0，失败返回-1。

线程同步与互斥

- #include <semaphore.h>

  int sem_post(sem_t *sem);

  功能：

  将信号量的值加1并发出信号唤醒等待线程。

  参数：

  sem：信号量地址。

  返回值：

  成功返回0，失败返回-1。

线程同步与互斥

- #include <semaphore.h>

  int sem_getvalue(sem_t *sem, int *sval);

  功能：

  获取sem标识的信号量的值，保存在sval中。

  参数：

  - sem：信号量地址。
  - sval：保存信号量值的地址。

  返回值：

  成功返回0，失败返回-1。

线程同步与互斥

- #include <semaphore.h>
  int sem_destroy(sem_t *sem);
  功能：
  删除sem标识的信号量。
  参数：
  sem：信号量地址。
  返回值：
  成功返回0，失败返回-1。

例：02_semphore_1.c
02_semphore_2.c

进程同步与互斥

- 有名信号量

  其实POSIX的信号量有两种：

  1、无名信号量

  2、有名信号量

  前面我们介绍的就是无名信号量，无名信号量一般用于线程间同步或互斥。

  而有名信号量一般用于进程间同步或互斥。

进程同步与互斥

- #include <fcntl.h>
- #include <sys/stat.h>
- #include <semaphore.h>

当信号量存在时使用：

```
sem_t *sem_open(const char *name, int oflag);
```

当信号量不存在时使用：

```
sem_t *sem_open(const char *name, int oflag,
                mode_t mode, unsigned int value);
```

进程同步与互斥

功能：

创建一个信号量。

参数：

- name：信号量文件名。
- flags：sem_open函数的行为标志。
- mode：文件权限(可读、可写、可执行)的设置。
- value ：信号量初始值。

返回值：

成功返回信号量的地址，失败返回SEM_FAILED。

进程同步与互斥

- #include <semaphore.h>

  int sem_close(sem_t *sem);

  功能：

  关闭有名信号量。

  参数：

  sem：指向信号量的指针。

  返回值：

  成功返回0，失败返回-1。

进程同步与互斥

- #include <semaphore.h>

  int sem_unlink(const char *name);

  功能：

  删除信号量的文件。

  参数：

  name：信号量文件名。

  返回值：

  成功返回0，失败返回-1。

线程同步与互斥

- 练习

  生产者消费者：

  - 有一个仓库，生产者负责生产产品，并放入仓库，消费者会从仓库中拿走产品(消费)。
  - 要求：
    - 仓库中每次只能入一人(生产者或消费者)。
    - 仓库中可存放产品的数量最多10个，当仓库放满时，生产者不能再放入产品。
    - 当仓库空时，消费者不能从中取出产品。
    - 生产、消费速度不同。

凌阳教育官方微信：Sunplusedu

Tel: 400-705-9680 , BBS: www.51develop.net , QQ群：241275518