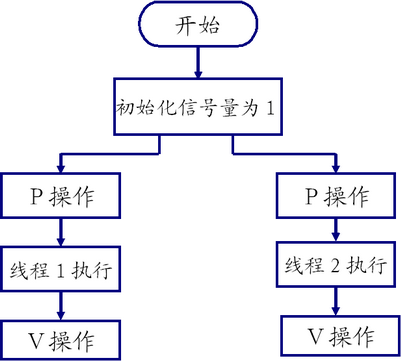
# 信号量概述

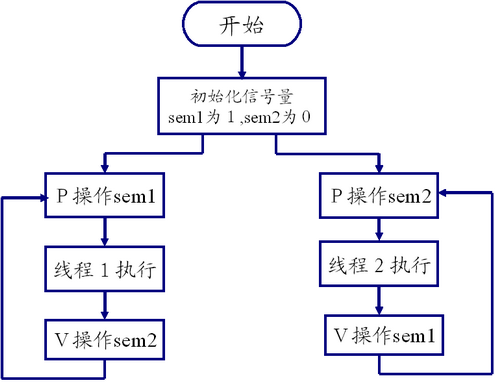
当信号量值大于 0 时，则可以访问资源，否则将阻塞。[PV 原语](http://baike.baidu.com/link?url=g6c4p4ydcoeFO9WC0HdMExCJf9E9NUpm720SQvDZMWrDxYg7BYDyYM3ls-LosEhLskYK2qVnGJcZqvIQBFk8d_)是对信号量的操作，一次 P 操作使信号量减１，一次 V 操作使信号量加１。

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。

信号量用于互斥：



信号量用于同步：



在 POSIX 标准中，信号量分两种

无名信号量：一般用于线程间同步或互斥

有名信号量：一般用于进程间同步或互斥

# 无名信号量

以下函数所需头文件：

#include <semaphore.h>

**注意**：编译信号量操作函数时，需要加上参数**-lpthread**。

信号量数据类型为：**sem\_t**。

**初始化信号量**

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（**常用**）；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：0

失败：-1

**信号量 P 操作（减 1）**

int sem\_wait(sem\_t \*sem);

功能：

将信号量的值减 1。如果没有信号量（即 <= 0），则阻塞

参数：

**sem**：信号量的地址。

返回值：

成功：0

失败：-1

int sem\_trywait(sem\_t \*sem);

以非阻塞的方式获取信号量

**信号量 V 操作（加 1）**

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（sem\_wait()）。

参数：

**sem**：信号量的地址。

返回值：

成功：0

失败：-1

**获取信号量的值**

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：0

失败：-1

**销毁信号量**

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：0

失败：-1

# 无名信号量应用实例

信号量用于互斥实例：

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

#include <semaphore.h>

sem\_t sem; //信号量

void printer(char \*str)

{

    sem\_wait(&sem);//减一

    while(\*str)

    {

        putchar(\*str);

        fflush(stdout);

        str++;

        sleep(1);

    }

    printf("\n");

    sem\_post(&sem);//加一

}

void \*thread\_fun1(void \*arg)

{

    char \*str1 = "hello";

    printer(str1);

}

void \*thread\_fun2(void \*arg)

{

    char \*str2 = "world";

    printer(str2);

}

int main(void)

{

    pthread\_t tid1, tid2;

    sem\_init(&sem, 0, 1); //初始化信号量，初始值为 1

    //创建 2 个线程

    pthread\_create(&tid1, NULL, thread\_fun1, NULL);

    pthread\_create(&tid2, NULL, thread\_fun2, NULL);

    //等待线程结束，回收其资源

    pthread\_join(tid1, NULL);

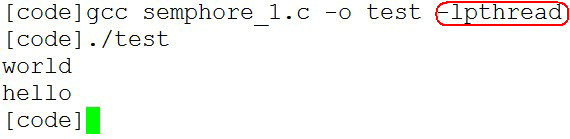
    pthread\_join(tid2, NULL);

    sem\_destroy(&sem); //销毁信号量

    return 0;

}

运行结果如下：



信号量用于同步实例：

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

sem\_t sem\_g,sem\_p;   //定义两个信号量

char ch = 'a';

void \*pthread\_g(void \*arg)  //此线程改变字符ch的值

{

    while(1)

    {

        sem\_wait(&sem\_g);

        ch++;

        sleep(1);

        sem\_post(&sem\_p);

    }

}

void \*pthread\_p(void \*arg)  //此线程打印ch的值

{

    while(1)

    {

        sem\_wait(&sem\_p);

        printf("%c",ch);

        fflush(stdout);

        sem\_post(&sem\_g);

    }

}

int main(int argc, char \*argv[])

{

    pthread\_t tid1,tid2;

    sem\_init(&sem\_g, 0, 0);   //初始化信号量

    sem\_init(&sem\_p, 0, 1);

    pthread\_create(&tid1, NULL, pthread\_g, NULL);

    pthread\_create(&tid2, NULL, pthread\_p, NULL);

    pthread\_join(tid1, NULL);

    pthread\_join(tid2, NULL);

    return 0;

}

运行结果如下：

