

5.2 进程间通讯 - 信号量 (二)_物联网 / 嵌入式工程师 - 慕课网

“ 慕课网慕课教程 5.2 进程间通讯 – 信号量 (二) 涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- 信号量可以进行以下操作:
 - 对信号量的值加 1
 - 对信号量的值减 1
 - 等待信号量的值为 0
- 操作信号量调用 semop 函数

函数头文件

```
#include <sys/types.h>
```

```
#include <sys/ipc.h>
```

```
#include <sys/sem.h>
```

函数原型

```
int semop(int semid, struct sembuf *sops, size_t nsops);
```

函数功能

信号量操作函数，用于占用信号量、释放信号量、设置信号量等待

函数参数

- semid : 信号量集合 id
- sops : 信号量操作结构体指针, 见后面关于 struct sembuf 解释
- nsops : 操作的信号量的数量

函数返回值

- 成功 : 返回 0
- 失败 : 返回 -1, 并设置 errno

struct sembuf 结构体

- unsigned short sem_num;
 - 信号量编号, 从 0 开始, 在 sem_op 的帮助文档中
- short sem_op;
 - 信号量操作
 - -1 : 占用资源
 - +1 : 释放资源
 - 0 : 等待资源
- short sem_flg;

- 信号量操作标志
 - IPC_NOWAIT : 非阻塞, 在信号量的值为 0 时, 会立即返回
 - SEM_UNDO : 在进程终止时, 会自动释放信号量

- 在 semop 函数中关于信号量集合编号的说明

The set of operations contained in sops is performed in array order, and atomically, that is, the operations are performed either as a complete unit, or not at all. The behavior of the system call if not all operations can be performed immediately depends on the presence of the IPC_NOWAIT flag in the individual sem_flg fields, as noted below.

Each operation is performed on the sem_num-th semaphore of the semaphore set, where the first semaphore of the set is numbered 0. There are three types of operation, distinguished by the value of sem_op.

- 信号量集合调用 semctl 函数, 设置命令为 IPC_RMID
 - 注意: 在使用 IPC_RMID 时, 第三个参数会被忽略, 下面是帮助文档中的说明

IPC_RMID Immediately remove the semaphore set, awakening all processes blocked in semop(2) calls on the set (with an error return and setting set to EIDRM). The effective user ID of the calling process must match the creator or owner of the semaphore set, or the caller must be privileged.

The argument semnum is ignored.

- 具体使用方式如下:

```
ret = semctl(semid,IPC_RMID,NULL);
```

- 使用信号量解决父子进程对终端的竞争

- 信号量操作封装:

```
• sem.h

#ifndef __SEM_H_
#define __SEM_H_

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

extern int sem_create(int nsems,unsigned short values[]);
extern int sem_p(int semid,int semnum);
extern int sem_v(int semid,int semnum);
extern int sem_del(int semid);

#endif
```

- sem.c
 - sem_create 函数


```

union semun{
    int val;
    unsigned short *array;
};

int sem_create(int nsems,unsigned short values[])
{
    int semid,ret;
    key_t key;
    union semun s;

    key = ftok(SEM_PATHNAME,SEM_PRO_ID);
    if (key == -1){
        perror("[ERROR] ftok() : ");
        return -1;
    }

    semid = semget(key,nsems,IPC_CREAT|0666);
    if (semid == -1){
        perror("[ERROR] semget() : ");
        return -1;
    }

    s.array = values;

    ret = semctl(semid,0,SETALL,s);
    if (ret == -1){
        perror("[ERROR] semctl() : ");
        return -1;
    }

```

}

return semid;

}

- sem_p 函数

- int sem_p(int semid,int semnum)
 - {
 - struct sembuf sops;
 - sops.sem_num = semnum;
 - sops.sem_op = -1;
 - sops.sem_flg = SEM_UNDO;
 - return semop(semid,&sops,1);

- sem_v 函数

- int sem_v(int semid,int semnum)
 - {
 - struct sembuf sops;
 - sops.sem_num = semnum;
 - sops.sem_op = 1;
 - sops.sem_flg = SEM_UNDO;
 - return semop(semid,&sops,1);

- sem_del 函数

- int sem_del(int semid)
 - {
 - return semctl(semid,0,IPC_RMID,NULL);

- main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

#include "sem.h"

int main(void)
{
    pid_t cpid;

    int semid;
    unsigned short values[] = {1};

    semid = sem_create(1,values);
    if (semid == -1)
        return -1;

    cpid = fork();

    if (cpid == -1){

        perror("[ERROR] fork(): ");
        exit(EXIT_FAILURE);

    }else if (cpid == 0){

        while(1){
            sem_p(semid,0);
            printf("-----\n");
            printf("C Start.\n");
            sleep(1);
            printf("C End.\n");
            printf("-----\n");
            sem_v(semid,0);

        }

    }else if (cpid > 0){

        sleep(1);
        while(1){
```

```
sem_p(semid,0);
printf("-----\n");
printf("P Start.\n");
sleep(1);
printf("P End.\n");
printf("-----\n");
sem_v(semid,0);

}

wait(NULL);

sem_del(semid);
}
```

全文完

本文由 简悦 SimpRead 优化，用以提升阅读体验

使用了 全新的简悦词法分析引擎 beta，[点击查看详细说明](#)

