

Xilinx Zynq FPGA, TI DSP, MCU
기반의 프로그래밍 및 회로 설계
전문가 과정

<리눅스 시스템 프로그래밍>
2018.03.28 - 25 일차

강사 - 이상훈
gcccompil3r@gmail.com

학생 - 안상재
sangjae2015@naver.com

1) 세마포어는 여러 개의 프로세스들이 메모리를 공유할 때, 프로세스들이 공유메모리의 데이터(자원)에 동시에 접근할 수 있는 허용 가능한 counter 의 갯수를 말한다.

```
/* sem.c 파일 */
#include "sem.h"

int main(void)
{
    int sid;    // 세마포어 id
    sid = CreateSEM(0x777); // 0X777 : 세마포어의 권한

    printf("before\n");

    p(sid);    // 세마포어의 값을 1 감소시킴.

    printf("Enter Critical Section\n");

    getchar();
    v(sid);    // 세마포어의 값을 1 증시킴.

    printf("after\n");

    return 0;
}

/////////////////////////////////////////////////////////////////
/* semlib.c 파일 */
#include "sem.h"

int CreateSEM(key_t semkey)    // 세마포어 만듦.
{
    int status = 0, semid;
가
    if((semid = semget(semkey, 1, SEMPERM|IPC_CREAT|IPC_EXCL)) == 1)    /* semkey 로 세마포어
lock 을 걸겠다. 세마포어 1 개 만듦.
                                     IPC : 프로세스간 통신. IPC 생성/ IPC_EXCL : 해당 KEY 값으로 세마포
어가 있으면 실패.*/
        if(errno == EEXIST)
            semid = semget(semkey,1,0);    // 현재 있는 세마포어 가져옴.
        else
            status = semctl(semid,0,SETVAL,2); // semid 의 id 를 갖는 세마포어 집합의 첫번째 세마포어 값을 2 로 만
드름.
    if(semid == -1||status == -1) // 둘 중 하나라도 에러라면 -1 리턴
        return -1;

    return semid;
}

int p(int semid)    // 세마포어 값 1 증가시킴.
{
    struct sembuf p_buf = {0,-1,SEM_UNDO}; // SEM_UNDO : 이 프로세스가 종료되면 원래 값으로 초기화시
```

```

                                                                    김.(0)
if(semop(semid,&p_buf,1) == 1) // 세마포어 값을 1 증가시킴.
    return -1;
return 0;
}

int v(int semid)    // 세마포어 값을 1 감소시킴.
{
    struct sembuf p_buf = {0,1,SEM_UNDO};
    if(semop(semid,&p_buf,1) == -1)
        return -1;
    return 0;    // 정상적이라면 0 리턴
}

```

2-1) 결과 분석

```

before
Enter Critical Section
after

```

3) 프로세스간에 통신을 해야할 경우, 프로세스들간에 공유메모리를 생성해서 데이터를 공유할 수 있다. OS 를 포팅하지 않고 하나의 프로세스 안에서 모든 것을 처리할 수도 있지만 그렇게 하면 CPU 클럭이 느려서 대용량의 데이터를 실시간으로 처리할 수가 없게 된다. 그래서 여러개의 프로세스끼리 데이터를 공유하기 위해서 반드시 운영체제의 도움이 필요한 것이다.

```

/* send.c 파일 */

#include "shm.h"
#include <stdlib.h>

int main(void)
{
    int mid;
    SHM_t *p;    // 공유하려는 메모리번지가 p

    mid = OpenSHM(0x888);    //특정구간의 페이지 프레임을 얻고 id 값을 반환함.(아무나 접근할 수 없게 하기
    //위해)

    p = GetPtrSHM(mid); // 현재의 프로세스를 공유메모리와 연결하고, 공유메모리의 포인터를 얻어옴.

    Getchar();    // 키보드로부터 입력을 기다림.
    strcpy(p->name, "아무개");    // p → name 에 “아무개” 를 복사함.
    p->score = 93;

    FreePtrSHM(p);    // shared memory 해제 (약간 시간 소요)

    return 0;
}

/* recv.c 파일 */
#include "shm.h"

int main(void)
{
    int mid;
    SHM_t *p;

    mid = CreateSHM(0x888);    // 0x888 ID 의 공유메모리 생성

    p = GetPtrSHM(mid);

    getchar();
    printf("이름 : [%s], 점수 : [%d]\n", p->name, p->score);

    FreePtrSHM(p);
    return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* shmlib.c 파일 */
#include "shm.h"    // 공유메모리 관련 함수들을 사용하기 쉽게 커스텀 함수로 만듦.

int CreateSHM(long key)
{

```

```

        return shmget(key, sizeof(SHM_t), IPC_CREAT | 0777);
    }

int OpenSHM(long key)
{
    return shmget(key, sizeof(SHM_t), 0);
}

SHM_t *GetPtrSHM(int shmid)
{
    return (SHM_t *)shmat(shmid, (char *)0, 0); /* 두번째 인자가 0 이면 운영체제가 임의로
                                                공유메모리 주소를 정해줌. */
}

int FreePtrSHM(SHM_t *shmptr)
{
    return shmdt((char *)shmptr);
}

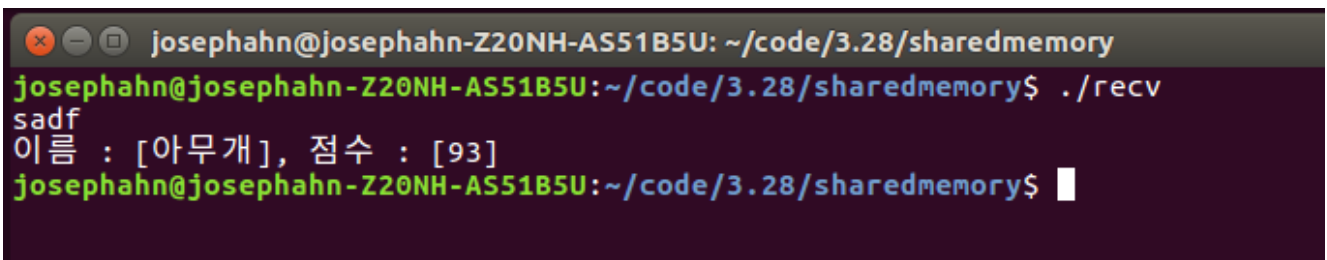
/////////////////////////////////////////////////////////////////
/* shm.h */
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

typedef struct
{
    char name[20];
    int score;
} SHM_t;

int CreateSHM(long key);
int OpenSHM(long key);
SHM_t *GetPtrSHM(int shmid);
int FreePtrSHM(SHM_t *shmptr);

```

3-1) 결과 분석 : 터미널창을 2 개 띄워서 한 쪽에서는 ./send 로 컴파일 하고, 한 쪽에서는 ./recv 로 컴파일 한다. send.c 파일에서 데이터를 입력하면 send.c 와 recv.c 파일의 공유메모리에 “아무개” 와 93 의 데이터가 write 된다. recv.c 파일에서 아래 사진과 같은 결과가 나온다.



```

josephahn@josephahn-Z20NH-AS51B5U: ~/code/3.28/sharedmemory
josephahn@josephahn-Z20NH-AS51B5U:~/code/3.28/sharedmemory$ ./recv
sadf
이름 : [아무개], 점수 : [93]
josephahn@josephahn-Z20NH-AS51B5U:~/code/3.28/sharedmemory$

```