

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

강사 – **Innova Lee(이상훈)**
gcccompil3r@gmail.com

학생 hyungjun Yu(유형준)
love592946@nave.rcm

0328 숙제

구조체의 끝 이후 새로운 시작점을 아는 코드.

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

typedef struct
{
    int count;//4 바이트
    char name[20];//20 바이트
    int score[0];
//인덱스 0 짜리 배열. 주소값.(sizeof(FLEX)를 하면 알수있음) 24 나옴.(앞의 두 크기)그러므로 이 구조체의 끝
이어디인가. 끝이후에 새로운 시작시점이 될수있다. ->score 가 갖는 의미.
} FLEX;

int main(void)
{
    int i;
    FLEX *p = (FLEX *)malloc(4096);
//메모리를 4096 으로 사용하겠다.malloc 의 단점은 할당받는시간과 해제하는 시간이 길다. 그래서 한번에 크게
잡고 배열처럼 쓰는기법. 자료구조나 서버만들때. 속도가 빨라짐.
    for(i = 0 ; i< 1000; i++)
    {
        p->score[i] = i*10;
        printf("%d\n",p->score[i]);
    }
    printf("%d",sizeof(FLEX));
    return 0;
}
```

semaphore

sem.c

```
int main(void)

{
    int sid;
    sid = CreateSEM(0x777);           //권한
    printf("before\n");
    p(sid);//더하기
    printf("Enter Critical Section\n"); //p 를통해서 세마포어를 증가시킴.
```

```

Getchar();           // 문자하나 받음 잠깐 대기하기위해 엔터치면 넘어감
v(sid);             // 빼기
printf("after\n");
return 0;
}

```

semlib.c

```

int CreateSEM(key_t semkey) // 777 로 lock 을 걸겠다.
{
    int status = 0, semid;
    if((semid = semget(semkey, 1, SEMPERM | IPC_CREAT | IPC_EXCL)) == -1) //세마포어에
작업권한주고 IPC 는 인터프로세스커뮤니케이션 - 프로세스간 통신. 그 영역을 공유.
//EXCL -> 해당 키값으로 세마포어가 있으면 씹으라는 소리.

```

if(errno == EEXIST)//세마포어가 존재한다면.

semid = semget(semkey,1,0);/// 세마포어 존재하는걸 현재있는걸 가져오겠다.

else//존재하지않는다면

```

status = semctl(semid,0,SETVAL,2);// semctl 은 SETVAL 현재 sid 를 가운데값(0)으로 해
준다. cnt 값이 0 이된거. 키값 777 가지고.

```

if(semid == -1 || status == -1)

return -1;

```

int p(int semid)
{

```

struct sembuf p_buf = {0,-1,SEM_UNDO};//SEM_UNDO 프로세스 종료할때 초기값으로 되
돌린다.

if(semop(semid,&p_buf,1) == -1)// 세마포어값을 1 증가시켜라. semop 가 실행되면 1 이됨.
--> 이 2 줄이 +일때의 패턴.

return -1;//비정상처리

return 0;//정상처리

}

```

int v(int semid)
{

```

struct sembuf p_buf = {0,1,SEM_UNDO}; // p 의 -1 과 다름.

if(semop(semid,&p_buf,1) == -1); //빼기 패턴.

return -1;

return 0;

```
}
```

```
//shared memory => 메모리 공유한다. 어떤? 페이지를 공유. 물리메모리 공유한다  
//하드웨어단위가 페이지 프레임  
//소프트웨어도 페이지 프레임. 4kb
```

//send.c

```
int main(void)  
{  
    int mid;  
    SHM_t *p;  
  
    mid = OpenSHM(0x888);  
  
    p = GetPtrSHM(mid);  
  
    getchar();  
    strcpy(p->name,"아무개");  
    p->score = 93;  
  
    FreePtrSHM(p);  
  
    return 0;
```

//recv.c

```
#include "shm.h"  
  
int main(void)  
{  
    int mid;  
    SHM_t *p;  
  
    mid = CreateSHM(0x888); // 메모리 주소 , 권한부여  
    p = GetPtrSHM(mid);  
  
    getchar();  
    printf("이름 : [%s], 점수 : [%d]\n", p->name,p->score);  
  
    FreePtrSHM(p);  
    return 0;  
}
```

//shmlib.c

```
#include "shm.h"  
  
int CreateSHM(long key)
```

```
{  
    return shmget(key,sizeof(SHM_t) IPC_CREAT | 0777);      //권한 부여  
int OpenSHM(long key)  
{  
    return shmget(key,sizeof(SHM_t),0);  
}  
SHM_t *GetPtrSHM(int shmid)  
{  
    return (SHM_t *)shmat(shmid,(char*)0,0);           //at 은 장소를 뜻함.  
}  
int FreePtrSHM(SHM_t *shmptr)  
{  
    return shmdf((char *)shmptr);  
}
```

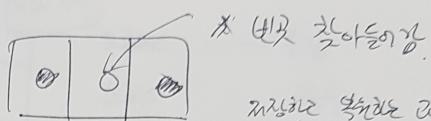
//shm.h

```
#include<sys/ipc.h>  
#include<sys/shm.h>  
#include<string.h>  
#include<errno.h>  
#include<stdlib.h>  
#include<stdio.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);
```

OS Lock 메커니즘

10328

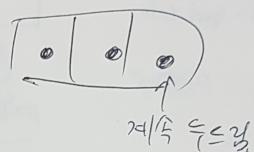
1. Semaphore



* 빙고 칠아들이 많.

제정하고 복잡화는 고려하지
Context switching 한다. clock 속도 떨어짐.

2. Spinlock

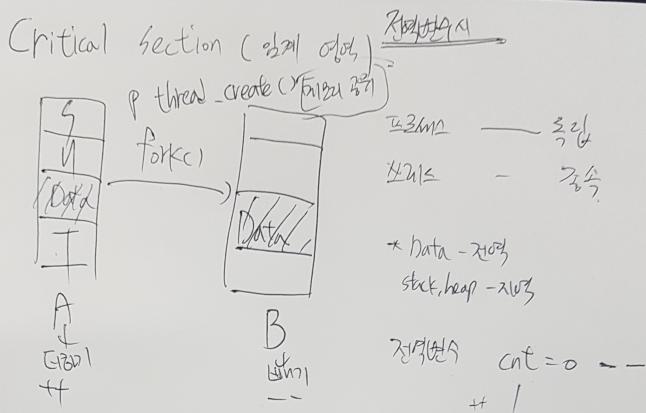


제속 끊드림

1. 대기로 흘러갈 경우 Semaphore

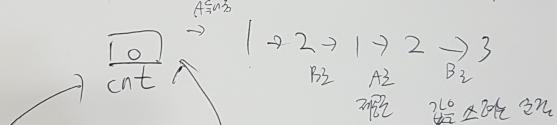
2. 단순, 가벼운 Spinlock. (Semaphore vs context switching, interrupt)

들어온다. 멈춰버린다. Spinlock 사용 예시 일정한 흐름으로 돌아온다.
나오는 경우 멈춰버린다. 멈춰버린 경우 멈춰버린다.

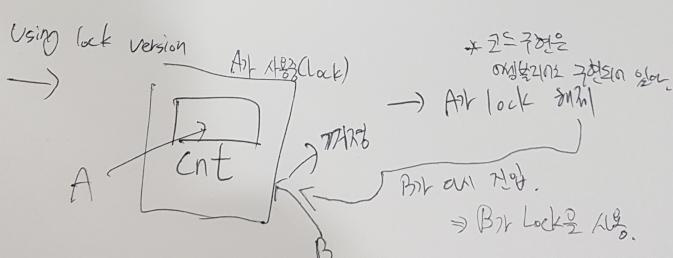


우리가 원하는 것 1 → 0 → 1 → 0 → 1 ...

정답 1 → 2 → 3 → 2 → 3 → 2 → 1 → 2 → 1 ... ?



A2 끝났음.



∴ 여러 Task들이 동시에 접근해 문제를 일으킬 수 있는 시간이 있음
이걸 Lock을 사용해서 막는다.

OS Lock