TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 윤연성
whatmatters@naver.com

Context Switch
= cpu 가 다른프로세스로 교환하려면 이전의 프로세스 상태를 보관하고 새로운 프로세스의 상태를 적재하는 작업이 필요한대 이 작업을 문맥교환이라함

Sem.c
semlib.c

컴파일
gcc semlib.c sem.c

OS lock 메커니즘

1.semaphore // 대기열 . 프로세스 여러개 작업이 가능 (프로세스대기열을 만듬)
lock 이 풀릴때가지 다른애 접근금지 lock 이 없는 곳을 찾아다님
대규모에 적합 ,컨텍스 스위칭
성능을 조금 희생해도 안정적으로 가져감

SEMMSL : 하나의 세마포어 셋트당 세마포어의 최대개수

SEMMNI : 세마포어 세트의 최대개수

2. spinlock // CPU 를 지속적으로 잡고있음 poolling , task 여러개 작업이 불가능
lock 이 풀릴떄까지 계속 기다림
단순간단, 소규모, 빨리끝내는게 이득이다 할때

semaphore 하면 빠지지않는게 있음

critical section (임계 영역)
= 여러 task 들이 동시에 접근해서 정보가 꼬일수 있는 구간

전역변수라고 해서 전부다 critical section 이 아님!

프로세스 → 독립
쓰레드 → 종속

우리가 원하는 것 1 - > 0 - > 1 → 0 . . . . . .
현실 1 →2 →3 → 2 → 3 → 2 → 1 → . . . . . ?

a 와 b 가 +-할때 들어간 애가 락을 걸어서 a 가 사용중이면 b 가 못들어오고
lock 가 해제되고 b 가 들어가면 다시 a 는 못들어감

컴파일방법
gcc -o send shmlib.c send.c
gcc -o recv shmlib.c recv.c

터미널두개 띄우고 ./send 이후에 ./recv

쉐어드 메모리의 용도

sem.h (헤더파일)

```
//////////////////////////////////////////////////////////////////////
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>

#define SEMPERM 0777
```

```
int CreateSEM(key_t semkey);
int p(int semid);
int v(int semid);
////////////////////////////////////////////////////////////////////////


#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);
```

```
#include "shm.h"

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

	mid = CreateSHM(0x888); //쉐어도 메모리 생성 0x888... 메모리아이디값 얻음
	p = GetPtrSHM(mid);

	getchar();
	printf("이름 : [%s], 점수 : [%d]\n", p->name, p->score); //아무개랑 93 이 나옴
```

```
	FreePtrSHM(p);
	return 0;
}
```

```
#include "shm.h"

int CreatSHM(long key)
{
	return shmget(key, sizeof(SHM_t) IPC_CREAT | 0777); //공유메모리
}
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 장소를 찾는거 id 값가지고 0 번지부>터 공유메모리의 물리주소
}

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

int main(void)
{
	int mid;		//메모리아이디
	SHM_t*p;		//공유하고자 아는 아이디의 번지가  p

	mid = OpenSHM(0x888);		//open 쉐어드메모리 ,
	p = GetPtrSHM(mid);		//진짜 쉐어드메모리의 포인터를 얻어와라

	getchar();
	strcpy(p->name, "아무개");	//
	p->score =93;

	FreePtrSHM(p);		//해제 free 는 바로 해제가 안됨
```

```
        return 0;
}
```

//프로세스를 만들고 메모리를 공유할라면 IPC 를 무조건 써야됨

// IPC 쓰는 이유 = 프로세스간에 정보를 공유하기 위해