Xilinx Zynq FPGA, TI DSP MCU 기반의

프로그래밍 및 회로 설계 전문가

강사 이상훈

(Innova Lee)

Gcccompil3r@gmail.com

학생 김민호

minking12@naver.com

IPC 기법중 Shaed Memory :

공유메모리는 여러장치가 공동으로 사용하는 메모리 , 여러개의 프로세스가 공통으로 사용하는 메모리를 의미.

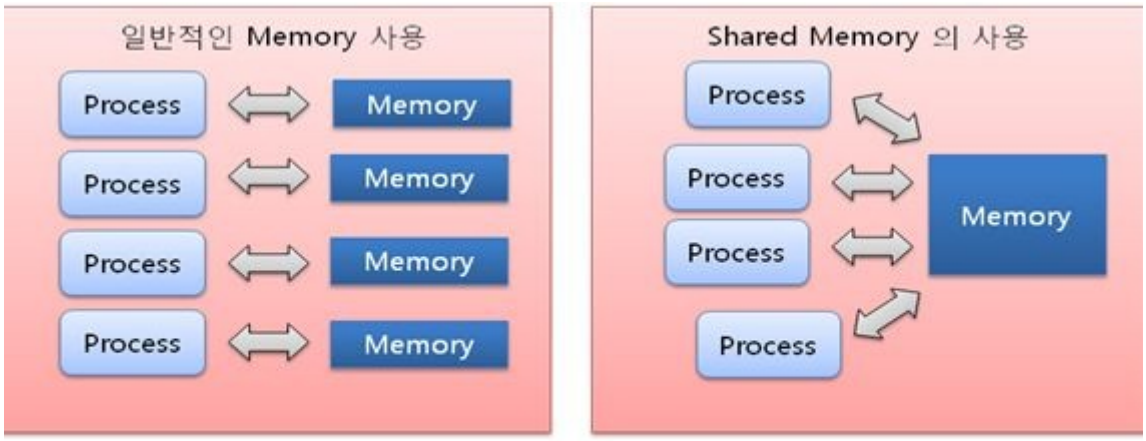

그림 2 일반적인 메모리 사용과 공유메모리 사용의 차이

일반적으로는 프로세스는 각각의 메모리를 독립적으로 사용하지만 ,
공유 메모리는 여러개의 프로세스가 하나의 메모리 영역을 사용한다.

하드웨어에서의 공유 메모리

멀티프로세서 시스템에서 여러 개의 cpu 가 공통적으로 접근 가능한 Ram 영역을 의미한다.
이러한 공유메모리 시스템은 여러개의 CPU 가 쉽게 하나의 View 를 공유 할 수 있고,
데이터를 사용할 때 매우 빠르게 메모리의 접근할 수가 있습니다.

그러나 동시에 많은 CPU 가 접근 함으로 인해, 복잡한 문제도 발생.

캐시 일관성(Cache Coherence) 문제가 대표적인 문제점이라고 할 수 있다.

캐시 일관성 문제란 각 클라이언트 PC 가 자신만의 로컬 캐시(Cache)를 가지고 있을때,

공유 메모리를 사용할 때 마다, 다른 PC 의 캐시와 공유 메모리의 데이터가 불일치하는 문제점을 말한다.

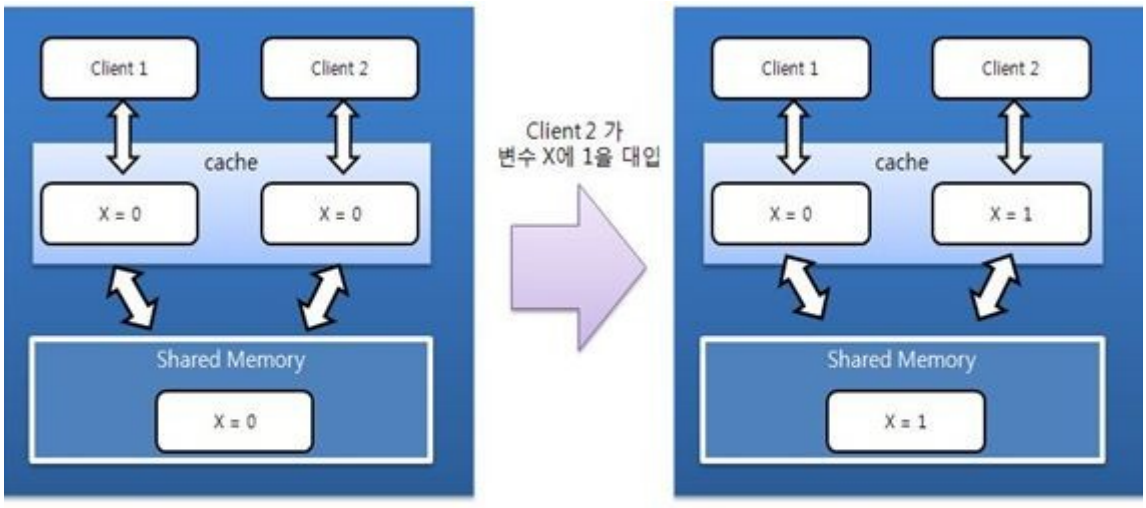

그림 4 Cache Coherence 이슈에 의한 데이터 불일치의 예

소프트웨어에서의 공유메모리

소프트웨에서 공유메모리는 보통 2 가지 용도로 사용.

하나는 지금 살펴보고 있는 IPC 의 한가지 방법이고,

다른 하나는 메모리를 아끼기 위한 보존관리 방법으로 사용한다.

IPC 는 알겠는데, 메모리를 아끼기 위한 보존관리 방법은 무엇일까요?

과거의 프로그래머들은 프로세스가 자주 사용하는 명령이

특정한 코드(Code) 에 집중되어 있다는 것을 깨달았습니다.

그래서 해당 코드 부분을 따로 분리하여,

사용하면 메모리를 절약할 수 있다는 것도 동시에 알게 되었는데,

이것이 바로 라이브러리(Library) 입니다.

라이브러리를 공유메모리 부분에 적재시켜,

메모리 공간을 아껴보자는 것이죠.

하지만 전에 프로세스는 자신의 메모리 영역 밖에는 알수 없다고 하였는데,

어떻게 라이브러리가 적재되어 있는 공유메모리 주소를 찾을 수 있을까요?

그것을 위하여, OS 는 프로세스의 메모리 영역에,

공유 메모리의 해당 라이브러리 부분을 매핑(mapping)합니다.

그림으로 상황을 보면 다음과 같습니다.

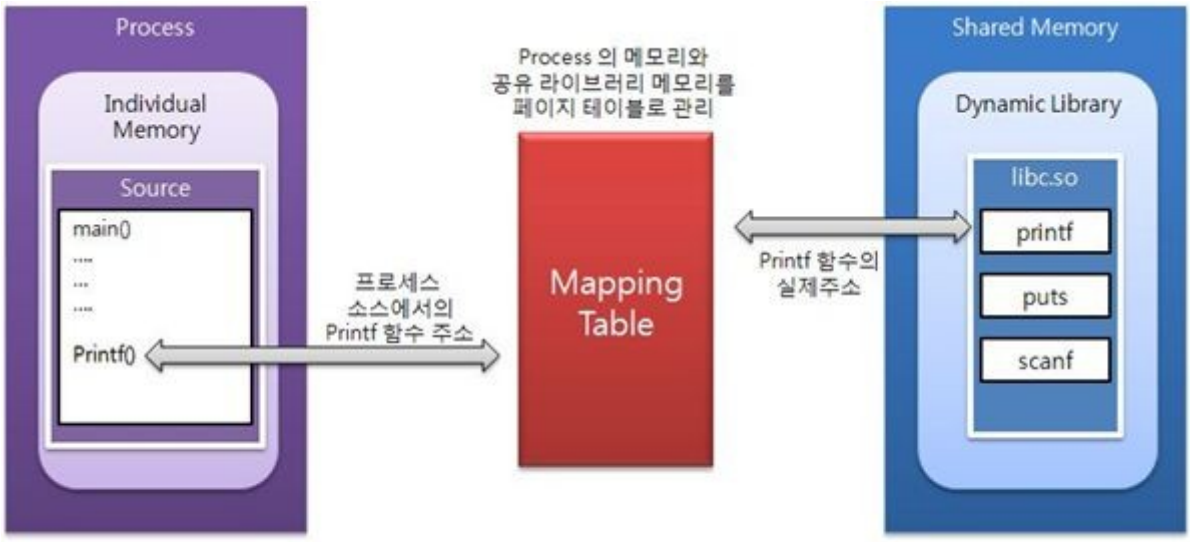

그림 6 동적 라이브러리의 주소 Mapping

프로세스(:12)는 자신의 업무를 수행하기 위해서 필요한 자료를 저장하기 위한 메모리 공간을 가지게 된다. 이러한 메모리공간에는 CPU 에 의해 수행되는 명령어들, 프로그램 시작시 정의되고 초기화

그림처럼 프로세스가 동적 라이브러리 함수를 호출하게 되면,

운영체제가 공유 메모리의 존재하는 printf 의 실제주소를

페이지 테이블로 관리하며 매핑을 해주게 됩니다.

이번에는 **IPC 기법에서의 공유메모리**를 살펴보겠습니다.

사실 이번 포스팅의 주 목적인 부분이죠.

IPC 기법중에서 공유 메모리는 처리속도가 가장 빠르다는 특징이 있습니다.

그 이유는 공유메모리가 다른 IPC 기법과는 달리

메모리 자체를 공유하므로 데이터 복사와 같은, 불필요한 오버헤드가 발생하지 않기 때문입니다.

(물리메모리)

반면 **공유 메모리는 반드시 같은 기계에서만 사용가능 하다**는 단점도 있습니다.

같은 기계라기 보다 같은 RAM 을 사용하는 시스템이라고 하는 편이 더 맞겠죠.

어쨌든 공유 메모리는 네트워크 사용이 불가능하므로, 다른 IPC 기법보다

덜 강력하다고 볼 수 도 있습니다.

또 **같은 메모리를 사용하는 만큼 데이터 불일치 문제를 해결해주어야 한다**는 이슈도 있습니다

https://akj61300.blog.me/80126200460

참고

보통 프로세스에서 사용되는 메모리영역은 해당 프로세스만이사용할수 있다.
하지만 때때로 여러개의 프로세스가 특정 메모리영역을 사용했으면 하는때가 있을것이다.
IPC 설비중의 하나인 "공유메모리"를 통해서 이러한일을 할수있다.

모든 프로세스는 자신의 업무를 수행하기 위해서 필요한 자료를 저장하기 위한 메모리 공간을 가지게 된다.

이러한 메모리공간에는 CPU 에 의해 수행되는 명령어들, 프로그램 시작시 정의되고 초기화된 데이타, 프로그램 시작시 정의되었지만 초기화 되지 않은 데이타, 함수호출에 필요한 정보, 동적할당이 이루어지는 데이타등 이 들어가게 된다.

hjh 프로세스는 시작시 혹은 실행중에 이러한 데이타를 저장하고 사용하기 위한 메모리 공간을 커널에 요구하여서 할당받아 사용하게 되는데, 이러한 메모리공간은 기본적으로 메모리를 요청한 프로세스만이 접근가능하도록 되어있다. 하지만 가끔은 여러개의 프로세스가 특정 메모리 공간을 동시에 접근해야할 필요성을 가질때가 있을것이다.

공유메모리는 이러한 작업을 위한 효율적인 방법을 제공한다.

공유메모리는 여러 IPC 중에서 가장 빠른 수행속도를 보여준다.

그이유는 하나의 메모리를 공유해서 접근하게 되므로, 데이타 복사와 같은 불필요한 오버헤드가 발생하지 않기 때문으로, 빠른 데이타의 이용이 가능하다. 그러나 하나의 프로세스가 메모리에 접근중에 있을때, 또다른 프로세스가 메모리에 접근하는 일이 발생하면 자칫 데이타가 홰손될수 있을것이므로, 한번에 하나의 프로세스가 메모리에 접근하고 있다는걸 보증해줄수 있어야 할것이다.

공유메모리의 생성요청은 최초 공유메모리 영역을 만드는 프로세스가 커널에 공유메모리 공간의 할당을 요청함으로써 이루어지며, 만들어진 공유메모리는 커널에 의해서 관리 되게 된다.

이런 이유로 한번만들어진 공유메모리는 운영체제를 리부팅하거나, 직접 공유메모리 공간을 삭제시켜주지 않은한, 공유메모리를 사용하는 모든 프로세스가 없어졌다고 하더라도, 계속적으로 유지되게 된다.

프로세스가 커널에게 공유메모리 공간을 요청하게 되면, 커널은 공유메모리 공간을 할당시켜주고 이들 공유메모리공간을 관리하기 위한 내부자료구조를 통하여, 이들 공유메모리를 관리하게 된다. 이 자료는 shmid_ds 라는 구조체에 의해서 관리되며 **shm.h** 에 정의되어 있다.

https://www.joinc.co.kr/w/Site/system_programing/IPC/SharedMemory

참고

```
#include<fcntl.h>
#include<unistd.h>
#include<stdio.h>
#include<string.h>
int main()
{
    int fd;
    char buff[1024]={0};

    fd=open("./abcd.txt",O_RDWR|O_CREAT|O_TRUNC,0644);
    read(0,buff,sizeof(buff));
    write(fd,buff,sizeof(buff));
    close(0);
    return 0;

}
```

scanf 를 사용하지 않고 터미널에서 입력한 문자열을 특정파일에 저장 하는 방법 .

0328 오전 큐시간에 ...