Operating Systems LAB 10

Wonpyo Kim
skykwp@gmail.com

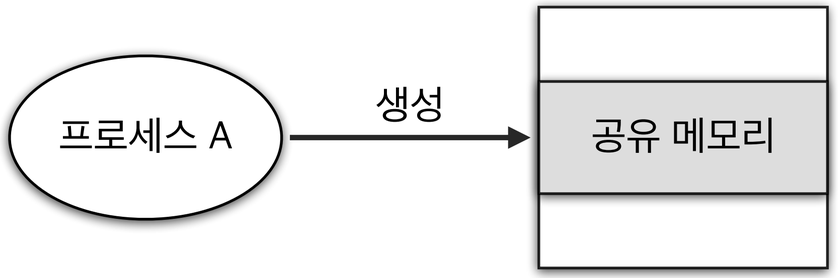
Outline

- Substance
  - **Shared Memory**

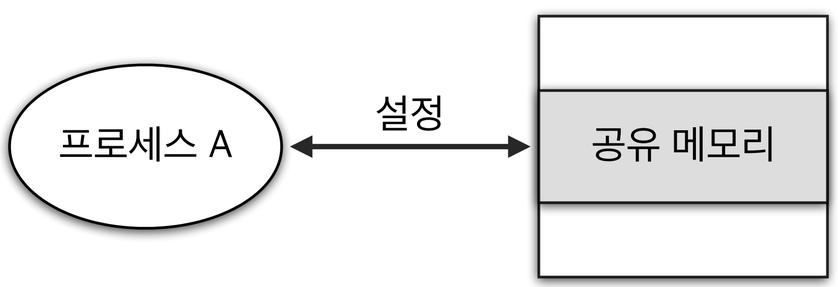
Shared Memory

- 공유 메모리란 여러 프로세스가 함께 이용할 수 있는 메모리를 말한다. 그래서 임의의 프로세스가 이 메모리 영역에 데이터를 쓰면 다른 프로세스가 이 내용을 읽을 수 있으므로 프로세스 간에 통신이 이루어진다. 단계별 동작은 다음과 같다.

  (1) 임의의 프로세스 A가 공유 메모리를 생성한다.

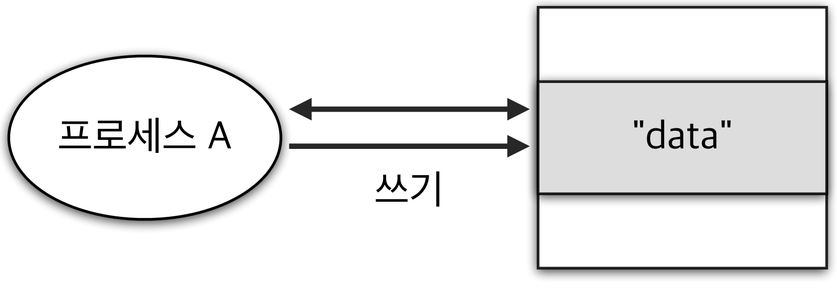


(2) 프로세스 A가 이 메모리 영역을 자신의 메모리 영역으로 임시 설정해야 한다. 이유는 자신의 메모리 처럼 사용하기 위함이다.

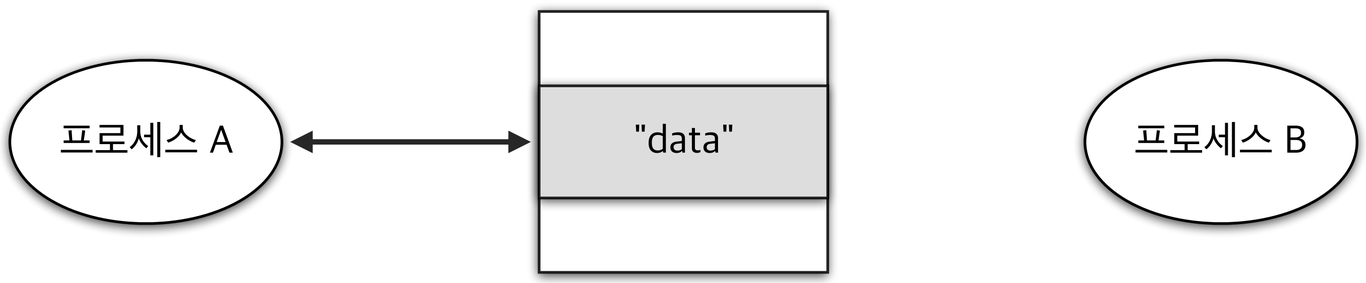


Shared Memory

(3) 데이터를 공유 메모리에 쓴다.

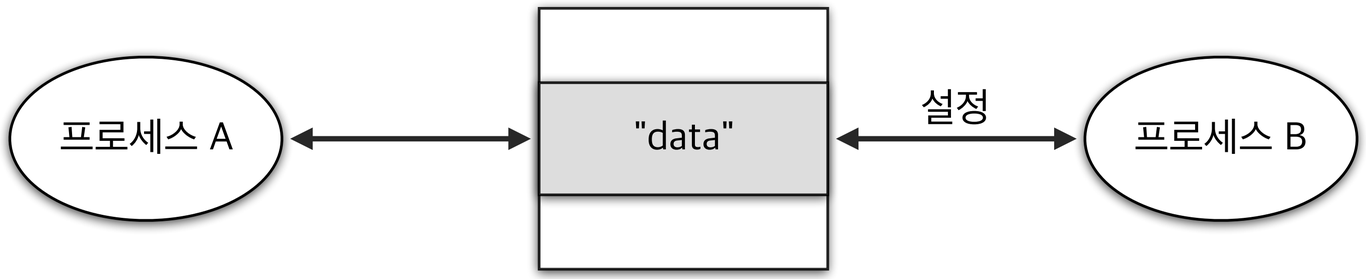


(4) 프로세스 B가 공유 메모리를 찾는다.

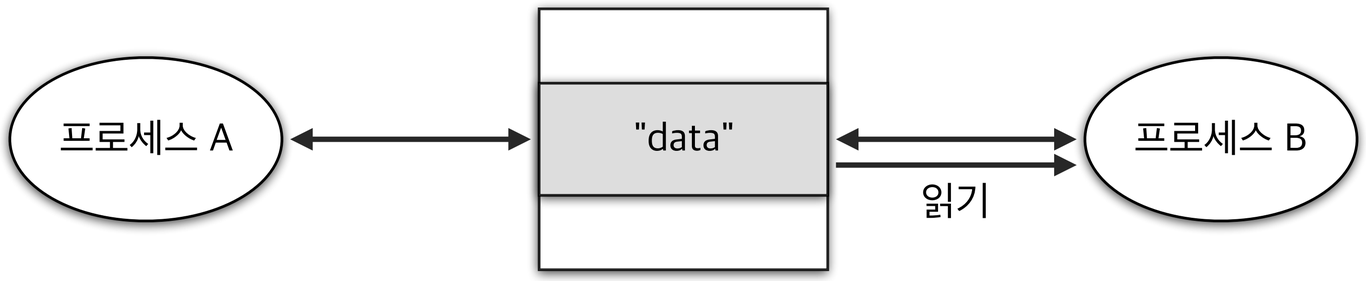


Shared Memory

(5) 찾은 공유 메모리를 프로세스 B의 메모리 영역으로 설정한다.



(6) 공유 메모리에서 데이터를 읽는다.

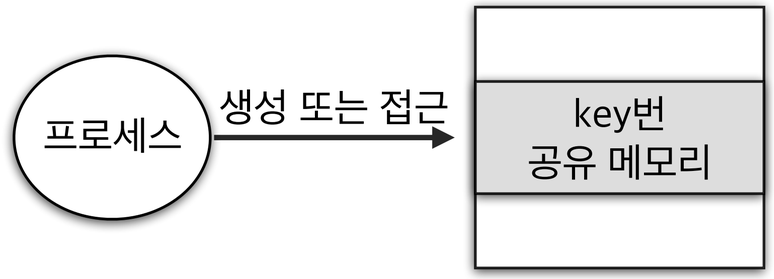


Shared Memory

- 공유 메모리를 이용하기 위해서는 공유 메모리를 생성하여 공유 메모리에 대한 식별자를 받아야 한다. 이 때, 사용하는 함수는 shmget( ) 함수이다. 만약, 공유 메모리가 이미 생성되어 있으면 식별자만을 얻게 된다.

- size 크기의 key 키를 갖는 공유 메모리를 생성하고 식별자를 반환한다.

shmget 함수
기능
　　공유 메모리를 생성한다.
기본형
　　int shmget(key_t key, int size, int shmflg);
　　key: 시스템에서 식별하는 공유 메모리 번호
　　size: 공유 메모리 크기
　　shmflg: 동작 옵션
반환값
　　성공: 공유 메모리 식별자
　　실패: -1
헤더파일
　　<sys/ipc.h>
　　<sys/shm.h>



Shared Memory

- key 란 시스템에서 식별하는 공유 메모리 번호로 사용 예는 다음과 같다. 1234 키를 갖는 공유 메모리를 생성하고 식별자를 반환한다. 만약 시스템에 1234번 공유 메모리가 있다면 기존 공유 메모리에 접근하여 식별자를 반환한다.

```
shmid = shmget((key_t)1234, ... );
```

- 메시지 큐에서와 마찬가지로 IPC_PRIVATE 를 사용할 수 있는데, 이 값으로 설정하면 유일한 공유 메모리를 생성한다.

```
shmid = shmget(IPC_PRIVATE, ... );
```

- 두 번째 인수인 size 는 생성하고자 하는 공유 메모리의 크기를 바이트 단위이다. 그리고 세 번째 인수인 shmflg 에 의해 shmget( ) 함수가 수행해야 할 동작이 설정된다. 관련 값은 다음과 같다.

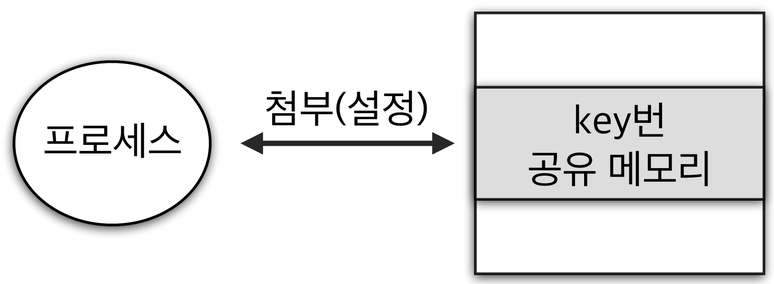
shmflg	의미
IPC_CREAT	key에 해당하는 공유 메모리가 없으면 새로 생성하는데, 이 때 접근 권한도 함께 부여해야 한다. 그러나, 공유 메모리가 이미 존재하면 이 옵션은 무시한다.
IPC_EXCL	공유 메모리가 있으면 실패하라 라는 의미로 -1을 반환한다. 이 값이 설정되지 않으면 기존 공유 메모리에 접근해 식별자를 반환한다.

- 사용 예) `shmid = shmget((key_t)1234, 1024, IPC_CREAT|0666);`

Shared Memory

- 공유 메모리가 생성되면 자신의 메모리 영역으로 설정해야 자신의 메모리 처럼 사용할 수 있다. 이 때, shmat( ) 함수를 사용하며 사용법은 다음과 같다.

shmat 함수
기능
공유 메모리를 호출 프로세스 메모리로 첨부(설정)한다.
기본형
void *shmat(int shmid, const void *shmaddr, int shmflg);
shmid: 공유 메모리 식별자
shmaddr: 공유 메모리 주소, 일반적으로 NULL
shmflg: 동작 옵션
반환값
성공: 공유 메모리 주소 포인터
실패: (void *) -1
헤더파일
<sys/types.h>
<sys/shm.h>



Shared Memory

- 첫 번째 인수인 shmid 는 shmget( ) 함수에 의해 얻어진 공유 메모리 식별자이다.

```
shmid = shmget( ... );
shmat(shmid, ... );
```

- 두 번째 인수인 shmaddr 은 공유 메모리가 호출 프로세스의 메모리 영역에 연결되는 주소로 NULL 을 설정하면 시스템이 알아서 주소를 정해준다. 만약, NULL이 아닌 값을 설정하려면 호출 프로세스가 메모리의 어느 위치에서 동작하는지 알아야 하므로 거의 사용하지 않는다.

```
shmat(shmid, (void *)0, ... );
```

- 세 번째 인수인 shmflg 에 설정 가능한 값은 다음과 같으며, SHM_RND 는 shmaddr 을 NULL 이 아닌 값을 설정할 때만 이용할 수 있다.

shmflg	의미
SHM_RDONLY	공유 메모리를 읽기 전용으로 사용한다.
SHM_RND	shmaddr 을 NULL 이 아닌 값을 설정할 때만 이용할 수 있으며, shmaddr 을 반올림하여 메모리 페이지 경계에 맞춘다.

Shared Memory

- 다음은 shmid 공유 메모리를 호출한 프로세스의 메모리 영역으로 첨부하되 읽기 전용으로 사용한다는 의미이다.

```
shmat(shmid, (void *)0, SHM_RDONLY);
```

- shmat( ) 함수 호출에 성공하면 공유 메모리의 주소에 대한 포인터를 반환하고, 실패하면 (void *) -1 를 반환하므로 호출 실패에 대한 검사에서 제대로 설정해야 한다.

- shmat( ) 함수 호출 성공 후 공유 메모리에 데이터를 쓰는 방법은 다음과 같이 반환된 메모리 주소에 문자열을 저장한다.

```
void *shmaddr;
shmaddr = shmat(shmid, (void *)0, 0);
strcpy((char *)shmaddr, "Hello Shared Memory?");
```

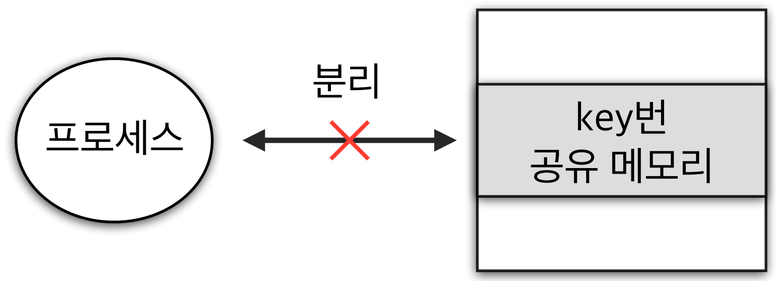
- 공유 메모리에서 저장된 데이터를 읽는 방법은 다음과 같이 메모리 주소가 가리키는 내용을 읽는 것이다.

```
printf("%s", (char *)shmaddr);
```

Shared Memory

- 공유 메모리에 대한 사용이 끝나면 자신이 메모리 영역에 첨부되었던 공유 메모리를 분리해야 한다. 이 때, shmdt( ) 를 쓴다.

shmdt 함수
기능
공유 메모리를 호출 프로세스 메모에서 분리(해제)한다.
기본형
int shmdt(const void *shmaddr);
shmaddr: 삭제할 공유 메모리 주소
반환값
성공: 0
실패:-1
헤더파일
<sys/types.h>
<sys/shm.h>



- 인수인 shmaddr 은 shmat( ) 함수에 의해 반환된 공유 메모리의 주소에 대한 포인터이다. 공유 메모리를 생성, 첨부(설정), 분리(해제)하는 과정을 정리하면 다음과 같다.

```
shmid = shmget((key_t)1234, 1024, IPC_CREAT|0666);
shmaddr = shmat(shmid, (void *)0, 0);
shmdt(shmaddr);
```

Shared Memory

- shmctl( ) 함수는 이용해 공유 메모리 정보를 얻고, 변경하고, 제거할 수 있다.

```
shmctl 함수
기능
        공유 메모리를 제어한다.
기본형
        int shmctl(int shmid, int cmd, struct shmid_ds *buf);
        shmid: 공유 메모리 식별자
        cmd: 제어 종류
        buf: 공유 메모리 정보에 대한 포인터
반환값
        성공: 0
        실패:-1
헤더파일
        <sys/ipc.h>
        <sys/shm.h>
```

Shared Memory

- shmid 공유 메모리를 cmd 에 따라 제어하는데, 세 번째 인수인 buf 는 struct shmid_ds 데이터형에 대한 포인터로 이 데이터형의 내용은 다음과 같다.

```
/* Data structure describing a shared memory segment.  */
struct shmid_ds
  {
    struct ipc_perm shm_perm;       /* operation permission struct */
    size_t shm_segsz;           /* size of segment in bytes */
    __time_t shm_atime;         /* time of last shmat() */
#ifndef __x86_64__
    unsigned long int __glibc_reserved1;
#endif
    __time_t shm_dtime;         /* time of last shmdt() */
#ifndef __x86_64__
    unsigned long int __glibc_reserved2;
#endif
    __time_t shm_ctime;         /* time of last change by shmctl() */
#ifndef __x86_64__
    unsigned long int __glibc_reserved3;
#endif
    __pid_t shm_cpid;           /* pid of creator */
    __pid_t shm_lpid;           /* pid of last shmop */
    shmatt_t shm_nattch;        /* number of current attaches */
    __syscall_ulong_t __glibc_reserved4;
    __syscall_ulong_t __glibc_reserved5;
  };
```

Shared Memory

- 어떠한 동작을 수행할 지를 결정하는 cmd 인수에 들어갈 수 있는 값은 다음과 같이 정의된다.

cmd	의미
IPC_STAT	공유 메모리의 정보를 얻어 buf에 저장한다. shmctl(shmid, IPC_STAT, &buf);
IPC_SET	공유 메모리의 정보를 buf에 저장한 값으로 변경한다. 단, shm_perm과 shm_ctime 멤버에 대한 정보만을 변경할 수 있다. shmctl(shmid, IPC_SET, &buf);
IPC_RMID	공유 메모리를 제거한다. 이 때, 세 번째 인수인 buf는 0으로 지정한다. shmctl(shmid, IPC_RMID, 0);