EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정

제 1기

2022. 08. 20

손표훈

CONTENTS

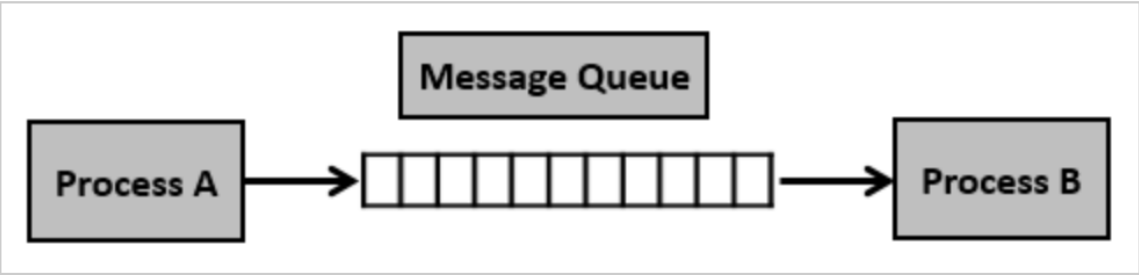
EDDI
Electronic Design
Development Institute

▪ 메시지 큐

EDDI
Electronic Design Development Institute

➢ 메시지 큐란?

→ IPC(Inter Process Communication) 여러 방법 중 하나이며, queue라는 자료구조를 통한 프로세스간 데이터 전달 방법이다.
→ 메시지 큐는 데이터 사이즈가 작은 경우 주로 사용된다.
→ shared memory와 다른 점은 메시지는 메시지 타입과 같은 형태로 공유 자원 접근에 대한 충돌을 처리할 수 있는 반면 공유 메모리는 별도의 메시지 큐와 같은 별도의 충돌 처리 방안이 없어 사용자가 동기화 처리를 해줘야 한다
→ 메시지 큐는 기본적으로 동기화를 지원하지만 플래그 셋팅을 통해 비동기화 설정을 할 수 있다



→ 메시지 큐를 사용하기 위해 다음과 같은 절차가 필요하다
(1) 메시지 큐를 생성 또는 생성된 메시지 큐를 연다-msgget()
(2) 메시지 큐에 데이터를 쓴다-msgsnd()
(3) 메시지 큐에 데이터를 읽는다-msgrcv()
(4) 메시지 컨트롤을 통해 메시지 큐를 삭제하거나 상태정보를 읽는다-msgctl()

■ 메시지 큐

EDDI
Electronic Design Development Institute

➢ msgget 함수

✓ 메시지 큐 생성/열기 시스템 콜 : int msgget(key_t key, int msgflg)

→ 함수 헤더 : sys/msg.h(메시지 큐의 프로토 타입), sys/ipc.h(정의된 ipc 에러 번호), sys/types.h(시스템 관련 구조체 선언)

→ 반환 값 : 메시지 큐 ID(open의 파일 디스크립터와 유사), 메시지 큐가 정상적으로 생성되면 메시지 큐의 ID 값을 반환 한다

✓ 첫 번째 파라미터 key : 각 프로세스들이 메시지 큐에 접근 할 수 있는 값으로 사용자가 임의로 정할 수 있거나 ftok()을 통해 파일 시스템에 존재하는 파일과 사용자 지정 id를 통해 생성 할 수 있다

✓ msgflg : IPC_CREAT를 통해 메시지 큐를 생성, IPC_EXCL를 통해 기존에 생성되어 있으면 에러를 반환한다
그리고 msgflg 파라미터에 OR연산을 통해 권한 값을 설정 할 수 있다

▪ 메시지 큐

EDDI
Electronic Design Development Institute

➢ msgsnd, msgrcv 함수

✓ 메시지 큐 msgsnd 시스템 콜 : int msgsnd(int msqid, const void *msgp, size_t msgsz, long msgtyp, int msgflg);

→ 함수 헤더 : sys/msg.h(메시지 큐의 프로토 타입), sys/ipc.h(정의된 ipc 에러 번호), sys/types.h(시스템 관련 구조체 선언)

→ 반환 값 : 에러 발생시 -1, 성공 시 0을 반환한다

✓ 첫 번째 파라미터 msqid : msgget을 통해 전달된 메시지 큐의 id 값

✓ msgp : 사용자가 정의한 메시지 버퍼를 의미하며 일반적으로 아래와 같은 구조를 가진다

The *msgp* argument is a pointer to caller-defined structure of the following general form:

```
struct msgbuf {
    long mtype;       /* message type, must be > 0 */
    char mtext[1];    /* message data */
};
```

✓ msgsz : 메시지의 사이즈

✓ msgtyp : 메시지 타입으로 같은 메시지 타입끼리 데이터 교환이 가능하다, 하나의 key값을 가지는 메시지 큐에 여러 개의 프로세스가 통신을 하는 경우 메시지 타입이 같은 프로세스 끼리 데이터 교환이 가능하게 설정 할 수 있다.

✓ msgflg : 마스크 비트로 여기서 중요한 것은 IPC_NOWAIT이다, 기본적으로 msgsnd, msgrcv는 blocking 동작을 하게 된다
즉, 보낼 데이터가 있을 때 까지 또는 데이터를 받을 때 까지 다음 동작을 처리안하고 대기하고 있는 것이다
이 때 IPC_NOWAIT을 하게 되면 데이터 송/수신 대기 없이 바로 return되어 다음 처리를 수행 하게 된다.

▪ 메시지 큐

EDDI
Electronic Design Development Institute

➢ msgctl 함수 : 메시지 큐를 삭제하거나 상태 정보를 파악하기 위해 사용

✓ 메시지 큐 msgctl 시스템 콜 : int msgctl(int msqid, int cmd, strut msqid_ds *buf);

→ 함수 헤더 : sys/msg.h(메시지 큐의 프로토 타입), sys/ipc.h(정의된 ipc 에러 번호), sys/types.h(시스템 관련 구조체 선언)

→ 반환 값은 다음 과 같다

RETURN VALUE top

On success, IPC_STAT, IPC_SET, and IPC_RMID return 0. A successful IPC_INFO or MSG_INFO operation returns the index of the highest used entry in the kernel's internal array recording information about all message queues. (This information can be used with repeated MSG_STAT or MSG_STAT_ANY operations to obtain information about all queues on the system.) A successful MSG_STAT or MSG_STAT_ANY operation returns the identifier of the queue whose index was given in *msqid*.

On failure, -1 is returned and *errno* is set to indicate the error.

✓ 첫 번째 파라미터 msqid : msgget을 통해 전달된 메시지 큐의 id 값

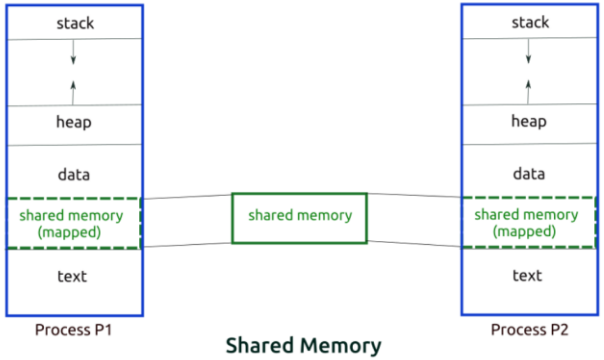
✓ cmd : 메시지 큐를 제어하기 위한 명령으로 다음과 같다
(1) IPC_STAT : 메시지 큐의 현재 상태 정보를 buf에 전달
(2) IPC_SET : 메시지 큐의 상태 정보 중 권한과 메시지 큐 바이트(사이즈)만 buf의 내용에 따라 변경 할 수 있다
(3) IPC_RMID : msqid에 해당하는 메시지 큐를 삭제 한다

▪공유 메모리

EDDI
Electronic Design Development Institute

➢ 공유 메모리란?

→ IPC 방법 중 하나로 커널이 관리하는 여러 프로세스간 공유할 수 있는 메모리를 말한다



➢ shmget 함수

✓ 공유 메모리 shmget 시스템 콜 : int shmget(ket_t key, size_t size, int shmflg) 공유 메모리 파일을 생성하거나 연다

→ 함수 헤더 : sys/shm.h(공유 메모리 API의 프로토 타입, 구조체), sys/ipc.h(정의된 ipc 에러 번호), sys/types.h(시스템 관련 구조체 선언)

→ 반환 값 : 에러 발생시 -1, 성공 시 0을 반환한다

→ key : 메시지 큐와 마찬가지로 공유 메모리에 접근 하기 위한 key 값이다(이 값을 통해 여러 프로세스가 공유 메모리에 접근 할 수 있다)

→ size : 공유 메모리로 설정하기 위한 사용자 버퍼의 사이즈이다. 이 값을 통해 커널에서 공유 메모리에 해당 사이즈 만큼 메모리 할당을 한다

→ shmflg : IPC_CREAT를 통해 메시지 큐를 생성, IPC_EXCL를 통해 기존에 생성되어 있으면 에러를 반환한다
그리고 shmflg 파라미터에 OR연산을 통해 권한 값을 설정 할 수 있다

▪공유 메모리

EDDI
Electronic Design
Development Institute

➢ shmat 함수

✓ 공유 메모리 shmat 시스템 콜 : void *shmat(int shmid, const void *shmaddr, int shmflg) 할당된 공유 메모리의 주소 값을 받는다

→ 함수 헤더 : sys/shm.h(공유 메모리 API의 프로토 타입), sys/ipc.h(정의된 ipc 에러 번호), sys/types.h(시스템 관련 구조체 선언)

→ 반환 값 : 에러 발생시 -1, 성공 시 할당한 공유메모리의 주소 값을 반환한다

→ shmid : 공유 메모리의 ID 값

→ shmaddr : 할당된 공유 메모리(물리 메모리)를 "프로세스"의 메모리 세그먼트(가상 메모리)에 연결하기 위해 프로세스의 특정 메모리를
설정하거나 시스템(커널)이 사용되지 않는 주소로 연결하기 위해 사용 된다- 여러 자료를 참고하여 정리한 내용인데 혹시 잘못된 내용이 있

→ shmflg : SHM_RDONLY, SHM_REMAP과 같은 플래그 비트를 설정하여 프로세스의 공유 메모리 세그먼트의 권한을 설정하거나
메모리 시작 주소를 리맵핑 할 수 있게 한다

▪ 공유 메모리

EDDI
Electronic Design
Development Institute

- shmdt 함수
  - ✓ 공유 메모리 shmdt 시스템 콜 : int shmdt(const void *shmaddr) 할당된 공유 메모리를 해제 한다
    - → 함수 헤더 : sys/shm.h(공유 메모리 API의 프로토 타입), sys/ipc.h(정의된 ipc 에러 번호), sys/types.h(시스템 관련 구조체 선언)
    - → 반환 값 : 에러 발생시 -1, 성공 시 0 반환한다
    - → shmaddr : shmat를 통해 전달 받은 공유 메모리 세그먼트의 시작 주소 값이 된다

■ 세마포어

EDDI
Electronic Design
Development Institute

➢ 트랜잭션(transaction)이란?

→ 어떤 시퀀스 처리의 안정을 보장하는 개념

→ 예시(은행 입출금 시스템)

(1) 은행에서 잔고를 관리하는 프로세스 A에서 10000이 저장 되어 있다
(2) 출금 프로세스 B가 동작하여 5000원을 인출하는 과정에서 출금 프로세스 C가 동작하여 5000원을 인출
(3) 프로세스 A는 5000이라는 잔고 값을 저장하는 도중 C가 발생하여 인출을 종료했다
(4) 프로세스 A에는 결과적으로 0이 되어야 하지만 이미 5000이라는 값이 메모리에 있는 상태 였으므로 5000이 저장된다
여기서 입출금 시스템이 트랜잭션이 되며, 시퀀스 보장을 위해 공유 자원(임계 영역)이라는 메모리에 하나의 프로세스가 동작하면 다른 프로세스가 접근을 못하게 막거나 초기 상태로 rollback을 통해 초기 상태에서 다시 시작하도록 해야 한다

→ 세마포어는 N개의 공유자원에 대한 동기화 한다

→ linux의 세마포에서 트랜잭션의 개념을 적용하기 위해 struct sembuf의 멤버 중 sem_flg에 SEM_UNDO를 설정하여
세마포어 사용시 문제가 발생하면 기존 사항에 적용하지 않은 상태에서 세마포어를 해제 하여 트랜잭션을 적용할 수 있다