IPC(Inter Process Communication)

IPC란?

- 프로세스 간 통신
- 프로세스끼리 서로 데이터를 주고 받는 행위 또는 그에 대한 방법
- 프로세스는 독립적으로 실행됨
    - 다른 프로세스의 영향을 받지 않는 장점이 있지만, 서로간의 통신이 어렵다는 단점 또한 있음

⇒ 독립적인 구조를 가진 프로세스 간의 통신을 해야 하는 상황이 있고, 이를 가능하도록 해주는 것이 **IPC 통신**

IPC 통신은 어디에서 제공?

- **커널** 영역에서 제공함
- 프로세스는 커널이 제공하는 IPC 설비를 이용해서 프로세스간 통신을 할 수 있음

커널이란?

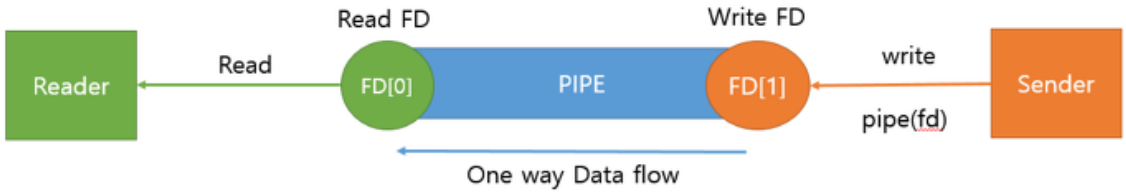
- 운영체제의 핵심적인 부분
- 운영체제 자체도 소프트웨어이기에, 메모리에 올라가야만 사용할 수 있음. 메모리 공간의 제약으로 운영체제 중 필요한 부분만 메모리에 올라가게 되는데, 메모리에 상주하는 운영체제의 부분을 “커널”이라함

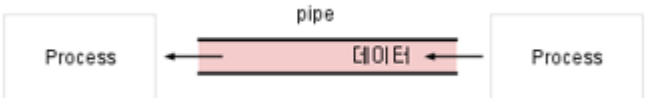
IPC의 종류

- IPC에는 다양한 설비가 존재함.

- 상황에 맞는 IPC 설비를 선택할 필요가 있음

1. **익명 PIPE**

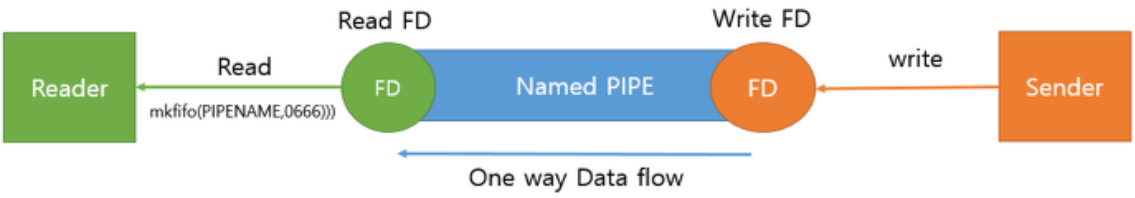




- 파이프는 2개의 프로세스를 연결함
- FIFO 구조
- 하나의 프로세스는 데이터를 쓰기만 하고, 다른 하나는 읽기만 할 수 있음
- 반이중 통신 : 한쪽 방향으로만 통신이 가능
- 송/수신을 모두 하고 싶을 경우, 2개의 파이프를 만들어야 가능해짐
- 통신을 할 프로세스가 명확하게 알 수 있는 경우에 사용
  - ex) 자식과 부모 프로세스간 통신에 사용
- 장점 : 매우 간단
- 단점 : 전이중 통신을 위해서는 2개의 PIPE를 만들어야하는데, 구현이 복잡함.

2. **Named PIPE(FIFO)**

- 부모 프로세스와 무관하게 전혀 다른 프로세스들 사이에서 통신이 가능함
- 프로세스 통신을 위해 이름이 있는 파일을 사용함
- Read only, Write only만 가능
- 익명 PIPE와 마찬가지로, 2개를 만들어야지만 전이중 통신이 가능함



- mkfifo를 통해 생성 가능함. mkfifo가 성공하면 명명된 파일이 생성됨

3. **Message Queue**
    - 메모리를 사용한 PIPE
    - FIFO
    - 커널에서 관리함
    - 입출력 방식으로 보면, Named PIPE와 동일함
    - Named PIPE는 데이터의 흐름이라면, Message Queue는 메모리 공간임
        - 어디에서나 물건을 꺼낼 수 있는 컨테이터 벨트 느낌
    - 메시지 큐에 쓸 데이터에 번호를 붙임으로써 여러 개의 프로세스가 동시에 데이터를 쉽게 다룰 수 있음

4. **공유 메모리(Shared Memory)**

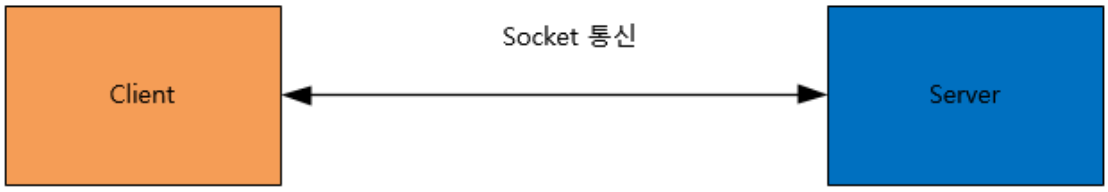
> **데이터를 공유하는 방법**
> 1) 통신을 이용하여 데이터를 주고 받음
> 2) 데이터를 함께 사용(공유)함

- PIPE, Named PIPE, Message Queue는 통신을 위한 설비라면, Shared Memory는 **공유메모리가 데이터 자체를 공유하도록 지원하는 설비**임
- 프로세스 메모리 영역은 독립적이며, 다른 프로세스가 접근하지 못하도록 보호되어야 하지만, 다른 프로세스가 데이터를 사용하도록 해야하는 상황도 필요함
    - 스레드와 같이 메모리를 공유하도록 해주면 매우 편리함
- 공유 메모리 : 프로세스간 메모리 영역을 공유해서 사용할 수 있도록 함
- 프로세스가 공유 메모리 할당을 커널에 요청하면, 커널은 해당 프로세스에 메모리 공간을 할당해주고, 이후 모든 프로세스는 해당 메모리 영역에 접근이 가능함
- IPC 중 가장 빠르게 작동함 → 중개자 없이 바로 메모리에 접근가능하므로

5. **메모리 맵**
    - 공유 메모리처럼 메모리를 공유함

- 메모리맵은 열린 파일을 메모리에 맵핑시켜서 공유하는 방식임
- 공유 매개체 : 파일 + 메모리
- 대용량 데이터를 공유해야 할 때 사용함

6. **소켓**



- 네트워크 소켓 통신을 위해 데이터를 공유함
- 원격에서 프로세스 간 데이터를 공유할때 사용함
- 클라이언트 - 서버 구조로 데이터 통신

2. IPC 별 사용 시기 및 특징

IPC 종류	PIPE	Named PIPE	Mesage Queue	Shared Memory	Memory Map	Socket
사용 시기	부모 자식 간 단 방향 통신 시	다른 프로세스 와 단 방향 통신 시	다른 프로세스와 단 방향 통신 시	다른 프로세스와 양 방향 통신 시	다른 프로세스 와 양 방향 통신 시	다른 시스템 간 양 방향 통신 시
공유 매개체	파일	파일	메모리	메모리	파일+메모리	소켓
통신 단위	Stream	Stream	구조체	구조체	페이지	Stream
통신 방향	단 방향	당 방향	단 방향	양 방향	양 방향	양 방향
통신 가능 범위	동일 시스템	동일 시스템	동일 시스템	동일 시스템	동일 시스템	동일 + 외부 시스템

세마포어, 뮤텍스

- 프로세스 간 메시지를 전송하거나, 공유메모리를 통해 공유된 자원에 여러 개의 프로세스가 동시에 접근하면 임계구역 문제가 발생할 수 있음.
    - 임계구역이란? 여러 프로세스가 동일 자원을 동시에 참조하여 값(공유하는 변수명, 파일 등)이 오염될 위험 가능성이 있는 영역
- 공유된 자원에 한번에 하나의 프로세스만 접근할 수 있도록 제한을 두기 위해 “세마포어”와 “뮤텍스”를 사용함
- IPC 통신에서 프로세스 간 데이터를 동기화하고 보호하기 위해 사용함
- 공유된 자원의 데이터를 여러 스레드/프로세스가 접근하는 것을 막는 역할을함

> 뮤텍스?
>
> - 임계구역을 가진 스레드들의 실행시간이 서로 겹치지 않고 각각 단독으로 실행되도록 하는 기술
> - 동기화 대상이 1개

> 세마포어?
>
> - 멀티 프로그래밍 환경에서 공유된 자원에 대한 접근을 제한하는 방법
> - 동기화 대상이 1개 이상일 때

나올만한 질문

- 뮤텍스, 세마포어가 무엇이고, 차이점은

> ### 뮤텍스
>
> 상호배제, 제어되는 섹션에 **하나의 스레드만 허용**하기 때문에, 해당 섹션에 접근하려는 다른 스레드들을 강제적으로 막음으로써 첫 번째 스레드가 해당 섹션을 빠져나올 때까지 기다리는 것
>
> ### 세마포어

운영체제에서 공유 자원에 대한 접속을 제어하기 위해 사용되는 신호 공유자원에 접근할 수 있는 **최대 허용치만큼만 동시에 사용자 접근 가능**

스레드들은 리소스 접근 요청을 할 수 있고, 세마포어는 카운트가 하나씩 줄어들게 되며 리소스가 모두 사용중인 경우(카운트=0) 다음 작업은 대기를 하게 된다

차이점

세마포어는 뮤텍스가 될 수 있지만, 뮤텍스는 세마포어가 될 수 없음
세마포어는 소유 불가능하지만, 뮤택스는 소유가 가능함
동기화의 개수가 다름

- 프로세스의 특징을 설명하세요에 대한 부가적인 답변

3-1) 프로세스의 특징을 설명하세요.

- 프로세스는 각각 독립된 메모리 영역(Code, Data, Stack, Heap의 구조)을 할당받는다.
- 기본적으로 프로세스당 최소 1개의 스레드(메인 스레드)를 가지고 있다.
- 각 프로세스는 별도의 주소 공간에서 실행되며, 한 프로세스는 다른 프로세스의 변수나 자료구조에 접근할 수 없다.
- 한 프로세스가 다른 프로세스의 자원에 접근하려면 프로세스 간의 통신(IPC, inter-process communication)을 사용해야 한다. (Ex. 파이프, 파일, 소켓 등을 이용한 통신 방법 이용)