● multi process환경에서 process간에 데이터를 어떻게 주고 받을까? (IPC의 예시를 들어 보아라)

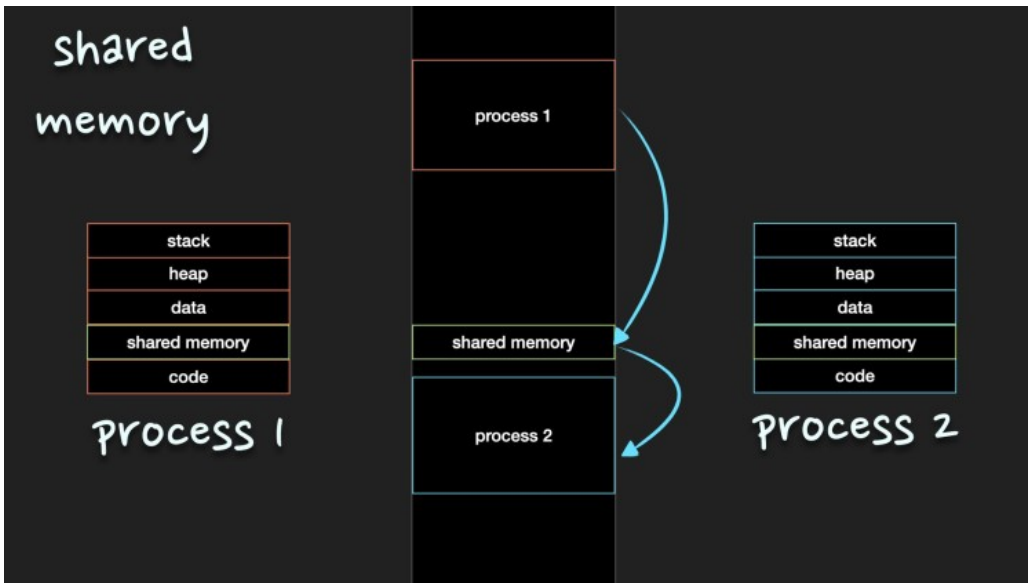
▶ 원칙적으로 process는 독립적인 주소 공간을 갖기 때문에, 다른 process의 주소 공간을 참조할 수 없고, 다른 process와 데이터를 주고받을 수 없습니다. 이를 위해 운영체제는 process 간의 자원 접근을 위한 매커니즘인 프로세스 간 통신(IPC, Inter Process Communication)를 제공하여 프로세스 간에 통신을 가능하게 해 줍니다. 프로세스 간 통신(IPC)은 크게 공유 메모리(shared memory) 모델과 메시지 전달(message passing) 모델로 나눌 수 있습니다. 공유 메모리 모델은 주소 공간의 일부를 공유하며 공유한 메모리 영역에 read/write를 통해 통신하게 되는데, 예시로는 공유메모리와 POSIX가 있습니다. 메시지 전송 모델의 경우에는 kernel을 통해 send/receive 연산을 통해 데이터를 전송합니다. 예시로는 Pipe, socket, message queue 등이 있습니다.

● 공유메모리와 메시지 전달 모델의 장단점

▶ 공유 메모리 모델은 초기에 공유 메모리 할당을 제외하면 kernel의 관여 없이 통신할 수 있기 때문에 속도가 빠른 장점있습니다. 하지만 여러 process가 동시에 메모리에 접근하는 문제가 발생할 수 있어서 별도의 동기화 과정이 필요하다는 단점이 있습니다. 이와 비교하여 메세지 전달 모델은 kernel을 통해서 데이터를 주고받기 때문에 통신 속도가 느리다는 단점이 있는 반면, kernel에서 제어를 해주기 때문에 안전하며 kernel이 동기화를 제공해준다는 장점이 있습니다.

● 공유메모리(shared memory) 모델

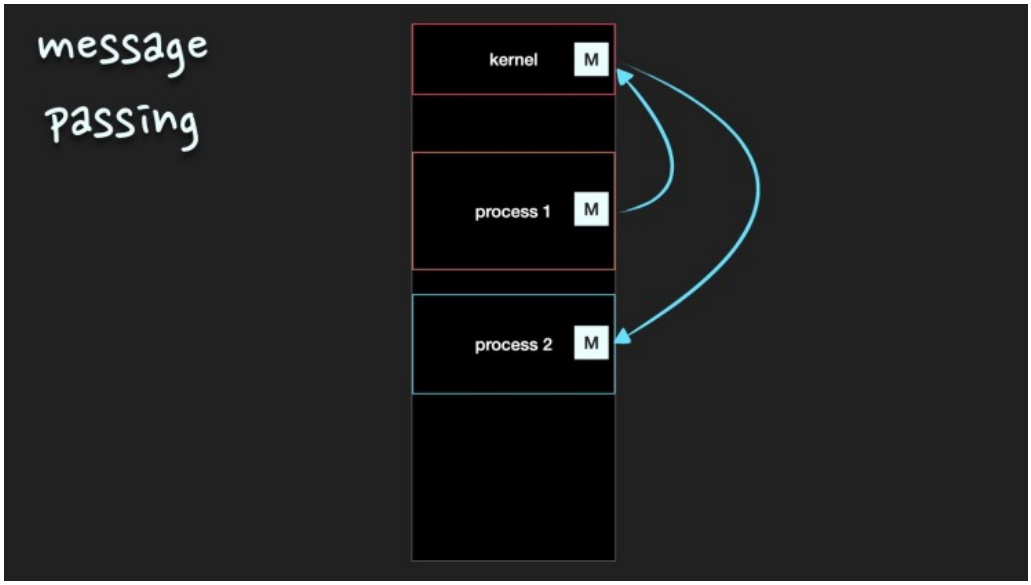
▶ 공유 메모리 방식에서는 process들이 주소 공간의 일부를 공유하고, 공유한 메모리 영역에 읽기/쓰기를 통해서 통신을 수행합니다. 프로세스가 공유 메모리 할당을 kernel에 요청하면 커널은 해당 프로세스 메모리 공간을 할당해 줍니다. 공유 메모리 영역이 구축된 이후에는 모든 접근이 일반적인 메모리 접근으로 취급되기 때문에 더이상 커널의 도움없이도 각 process들이 해당 메모리 영역에 접근할 수 있습니다. 따라서 커널의 관여 없이 데이터를 통신할 수 있기 때문에 IPC속도가 빠르다는 장점이 있습니다. 공유 메모리 방식은 프로세스간의 통신을 수월하게 만들지만 동시에 같은 메모리 위치에 접근하게 되면 일관성 문제가 발생할 수 있습니다. 이에 대해서는 커널이 관여하지 않기 때문에 프로세스들 끼리 직접 공유 메모리 접근에 대한 동기화 문제를 책임져야 합니다.



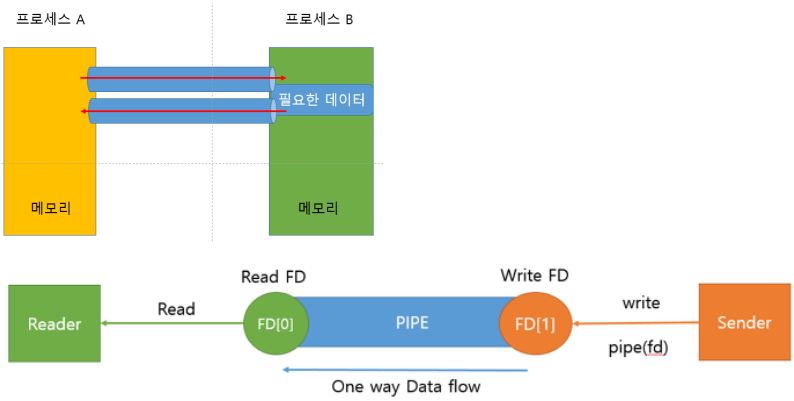
→ POSIX shared memory: shmget으로 공유메모리를 커널에 요청한다. 그럼 커널은 공유메모리객체를 만들기만 할 것이고, shmat을 통해 실제 프로세스 메모리에 공유메모리를 붙일(attach) 수 있다. shmdt는 붙인 공유메모리를 떼어내는데(detach) 사용하고, shmctl은 공유메모리객체를 정리하거나 기타 객체를 조작하는데 잠깐 이용한다 (http://blog.purewell.biz/2008/08/posix-shared-memory.html)

● 메시지 전달(message passing) 모델

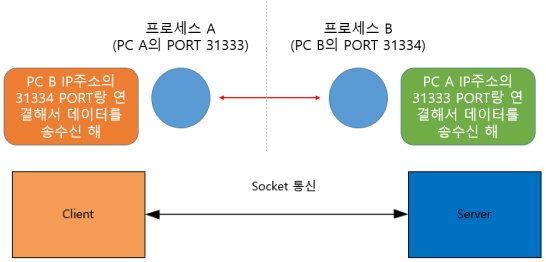
▶ 메시지 전달 방법은 통상 system call을 사용하여 구현됩니다 (시스템이 커널에 요청을 하는 것이 system call). kernel을 통해 send(message)와 receive(message)라는 두 가지 연산을 제공받습니다. 예를 들면, process1이 kernel로 message를 보내면 kernel이 process2에게 message를 보내주는 방식으로 동작합니다. 메모리 공유보다는 속도가 느리지만, 충돌을 회피할 필요가 없기 때문에 적은양의 데이터를 교환하는 데 유용합니다. 또, 구현하기가 쉽다는 장점이 있습니다. 대표적인 예시로는 pipe, socket, message queue등이 있습니다.



→ PIPE 방식: 두 프로세스 사이에 송신 PIPE와 수신 PIPE를 각각 만들어 데이터를 전송하는 방법. 송수진은 동시에 이루어지지 않는다. 익명의 PIPE를 통해서 동일한 PPID를 가진 프로세스들 간에 단방향 통신을 지원하는 것이다. 때문에 쌍방향 통신을 위해서는 read용 PIPE하나 write용 PIPE하나 따로 만들어야 한다.

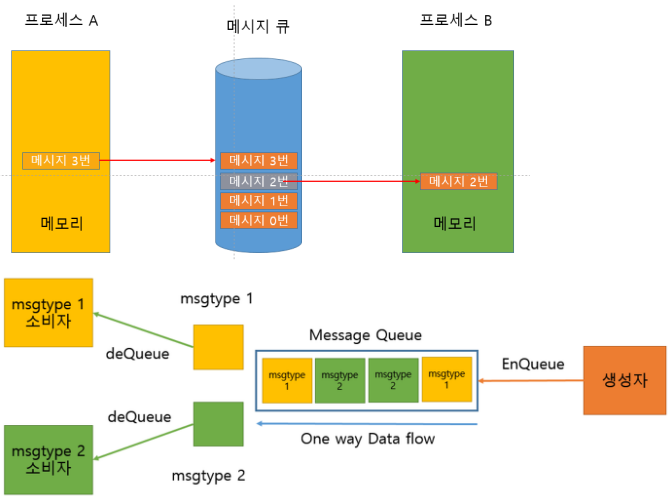


→ Socket: 네트워크 소켓 통신을 사용한 데이터 공유 방식이다. 원격에서 프로세스간 데이터를 공유할 때 사용하고, 네트워크 소켓을 이용하여 Client - Server 구조로 데이터 통신을 한다. 데이터 교환을 위해 양쪽 PC에서 각각 임의의 포트를 정하고, 해당 포트간의 대화를 통해 데이터를 주고받는 방식이다. 네크워크 프로그래밍이 가능해야 하고, 데이터 세그먼트 처리를 잘해야 한다.



→ message queue: FIFO 구조이고, 구조체 기반으로 통신을 한다. 우편함과 같은 방식으로, 통신하는 프로세스는 메시지 큐를 만들어 해당 메시지 함으로 데이터를 전송하거나 메시지함을 확인해서 메시지를 수신하는 방법이다. 커

널에서 제공하는 message queue이기 때문에, enqueue하는데 제한이 존재한다. (https://bluemoon-1st.tistory.com/22) (https://doitnow-man.tistory.com/110)



Resources: inflearn 개발남 노씨, bluemoon-1st, doitnow-man, purewell