운영체제 관점에서의 프로세스에 대한 발표

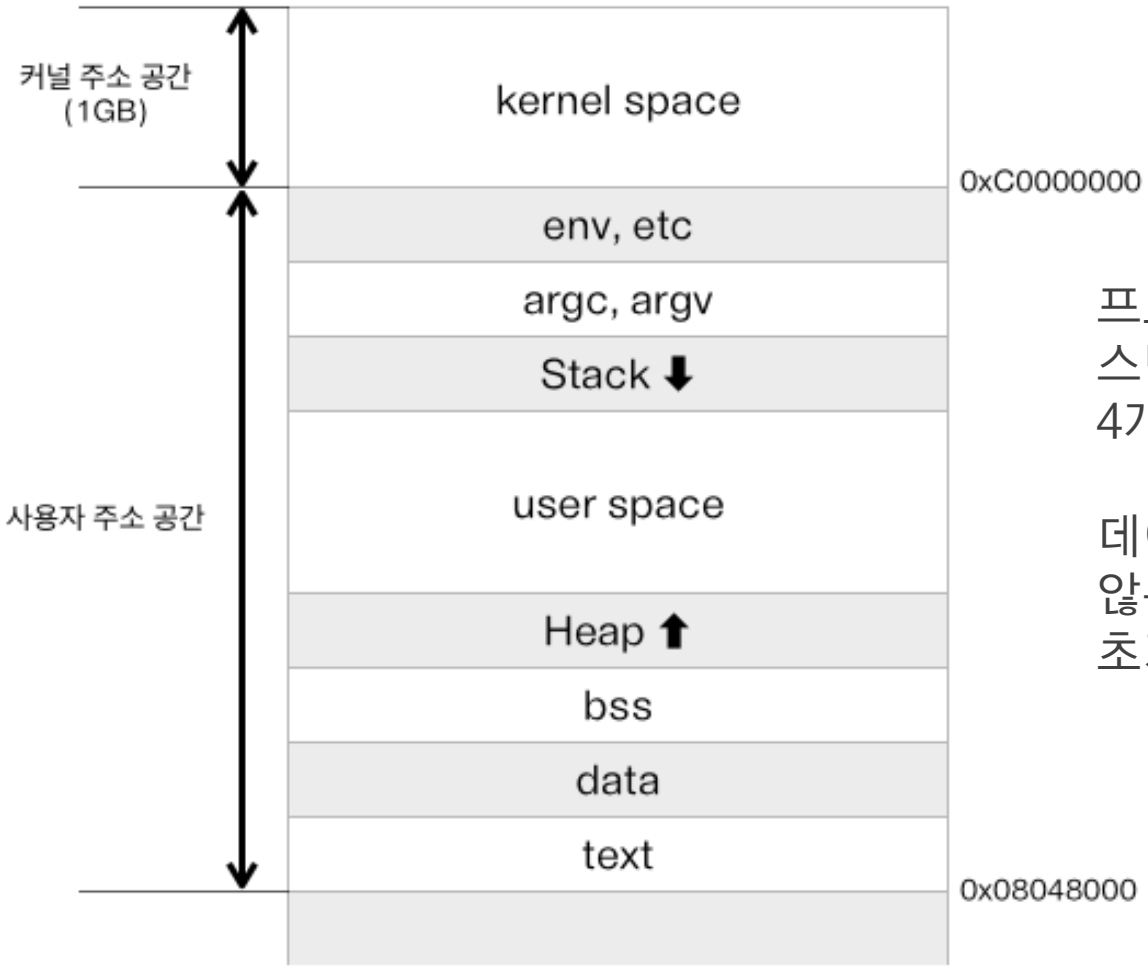
김수정

1. 프로세스의 구조
2. 프로세스 상태 전이
3. 스케줄링
4. 프로세스 간 통신

1. 프로세스

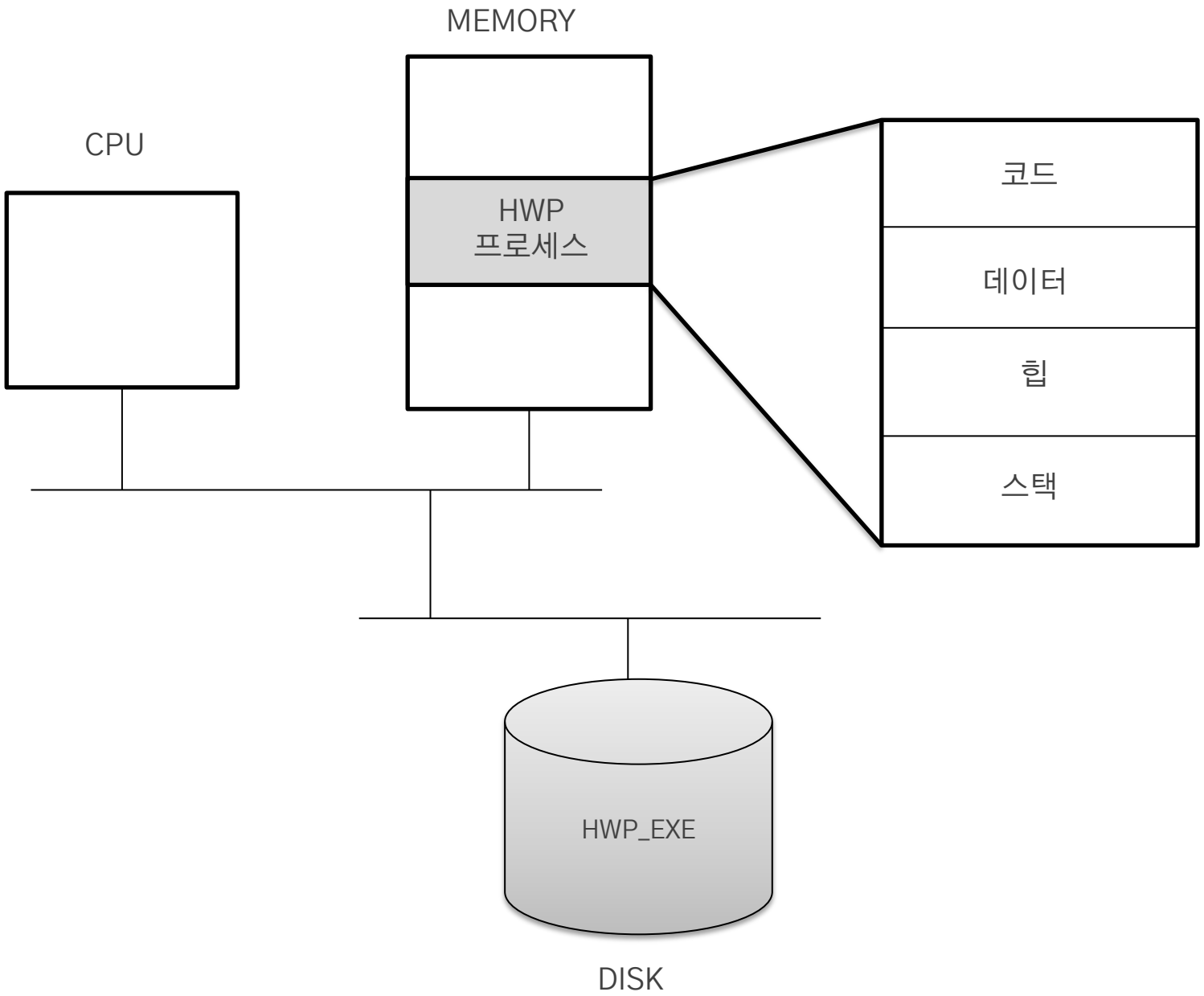
프로세스란?

-A process is program in execution.(실행 중인 프로그램)
-프로그램이 메모리에 적재(Load)되는 순간 프로세스



프로세스 메모리 구조는
스택, 힙, 데이터 및 텍스트
4개의 섹션으로 나뉨

데이터는 초기화되지
않은 데이터(bss)와
초기화된 데이터로 나뉨


MEMORY
CPU
HWP
프로세스
코드
데이터
힙
스택
HWP_EXE
DISK

Stack

임시 데이터가 저장되는 곳 예) 지역변수, 매개변수

컴파일 타임에 크기가 결정되기 때문에 무한히 할당할 수 없고 stack 영역을 초과하면 stack overflow 에러가 발생.

Heap

동적으로 할당되는 데이터가 저장되는 곳으로 런타임에 크기가 결정

Data

Uninitialized data (bss) - 초기화되지 않은 변수(전역 변수, static 변수)가 저장되는 곳

Initialized data - 초기화된 변수(전역변수, static 변수)가 저장되는 곳입니다.

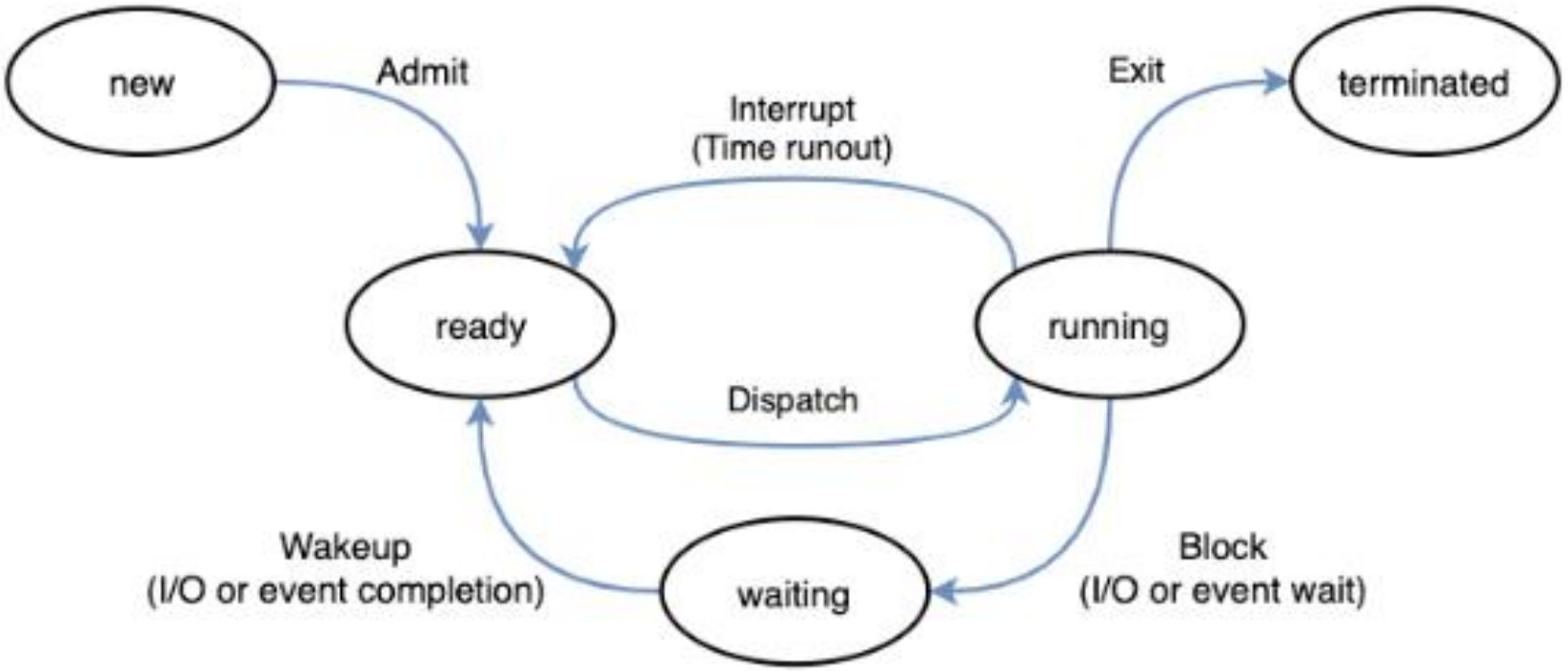
Text(Code)

프로그램의 코드가 저장되는 곳.

컴파일 타임에 결정되고 중간에 코드를 바꿀 수 없게 Read-Only 로 지정돼 있음.

2. 프로세스 상태

프로세스 상태 전이


new
Admit
Interrupt
(Time runout)
Exit
terminated
ready
running
Dispatch
Wakeup
(I/O or event completion)
waiting
Block
(I/O or event wait)

프로세스가 실행되면서 상태 변화

New
프로세스를 생성하기 위해 프로그램이 보조기억장치에 있는 상태

Ready
프로그램이 메인 메모리에 적재되어 프로세스가 된 상태. CPU에 의해 실행되기를 기다리고 있는 상태이며 준비 상태에 있는 프로세스들은 큐에서 대기

Running
CPU가 해당 프로세스를 실행한 상태

Waiting (or Block)
프로세스가 입출력완료, 시그널 수신 등 어떤 사건(event)을 기다리고 있는 상태

Terminated
프로세스가 완전히 종료된 상태.

Dispatch (ready -> running)

여러 프로세스들 중 한 프로세스를 선정하여 CPU에 할당하는 과정.

Interrupt (running -> ready)

할당된 CPU 시간이 지나면 Timeout Interrupt 가 발생하여 CPU를 다른 프로세스에게 양도하고 자신은 ready 상태로 전이되는 과정.

Block (running -> waiting)

자원 요청 후 즉시 할당 받을 수 없어, 할당 받을 때까지 기다리기 위해 running에서 waiting 상태로 전이되는 과정.

I/O 처리는 CPU가 아닌 I/O 프로세스가 담당하기 때문에 block이 발생.

Wakeup (waiting -> ready)

필요한 자원이 할당되면 프로세스는 waiting에서 ready 상태로 전이되는 과정.

3. 프로세스 스케줄링

스케줄링

프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업.

-장기 스케줄링 : 어떤 프로세스가 시스템의 자원을 차지할 수 있도록 할 것인가를 결정하여 준비상태 큐로 보내는 작업

-중기 스케줄링: 어떤 프로세스들이 CPU를 할당 받을 것인지를 결정

-단기 스케줄링: 프로세스가 실행되기 위해 CPU를 할당 받는 시기와 특정 프로세스를 결정하는 작업-> 프로세서 스케줄링과 문맥교환은 스케줄러에 의해 수행

**문맥교환(Context Switching)*: 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것

스케줄링의 목적

- 공정성
- 처리량 증가
- CPU 이용률 증가
- 우선순위 제도
- 오버헤드 최소화
- 응답시간 최소화
- 반환시간 최소화
- 대기 시간 최소화
- 균형 있는 자원의 사용
- 무한 연기 회피

프로세스 제어 블록(Process Control Block, PCB)
운영 체제가 프로세스를 표현한 것

– 운영체제가 프로세스 스케줄링을 위해 프로세스에 관한 모든 정보를 가지고 있는 데이터베이스.

–운영체제에서 프로세스는 PCB로 나타내어지며, PCB는 프로세스에 대한 중요한 정보를 가지고 있는 자료. 각 프로세스가 생성될 때마다 고유의 PCB가 생성, 프로세스가 완료되면 PCB는 제거.

–프로세스는 CPU를 점유하여 작업을 처리하다가도 상태가 전이되면, 진행하던 작업 내용들을 모두 정리하고 CPU를 반환해야 하는데, 이때 진행하던 작업들을 모두 저장하지 않으면 다음에 자신의 순서가 왔을 때 어떠한 작업을 해야 하는지 알 수 없는 사태가 발생.

=> 프로세스는 CPU가 처리하던 작업의 내용들을 자신의 PCB에 저장하고, 다음에 다시 CPU를 점유하여 작업을 수행해야 할 때 PCB로부터 해당 정보들을 CPU에 넘겨와서 계속해서 하던 작업을 진행.

PCB(Process Control Block, 프로세스 제어 블록) 구성 정보

저장정보	설명
프로세스 고유 **식**별자	프로세스를 구분할 수 있는 고유 번호
프로세스의 현재 **상**태	준비, 대기, 실행 등의 프로세스 상태
프로그램 **카**운터	실행될 명령어의 주소를 가지고 있는 레지스터
CPU **레**지스터 정보	누산기, 인덱스 레지스터, 범용 레지스터 등에 대한 정보
스케줄링 및 프로세스 우선순위	스케줄링 정보 및 프로세스가 실행될 우선 순위
계정 정보	CPU 사용 시간, 실제 사용시간, 한정된 시간
입,출력 상태 정보	입/출력장치, 개방된 파일목록
메모리장치 관리 정보	기준 레지스터, 페이지 테이블에 대한 정보
포인터	프로세스가 위치한 메모리 및 할당된 자원에 대한 포인터

4. 프로세스 간 통신

(IPC : Interprocess Communication)

프로세스가 동시에 실행될 때 두 가지 유형

독립적 프로세스(Independent process)
다른 프로세스에게 영향을 주거나 받을 수 없는 프로세스입니다. 독립적 프로세스는 데이터를 공유하지 않습니다.

협력 프로세스(Cooperating process)
다른 프로세스에게 영향을 주거나 받을 수 있는 프로세스입니다. 협력 프로세스는 데이터를 공유합니다

=> 협력을 하기 위해서는 프로세스 간 통신이 필요함.

협력 프로세스의 장점

정보 공유

여러 사용자가 상태나 데이터를 주고받으며 정보를 공유할 수 있음

계산 속도 향상

여러 프로세스가 동시에 작업을 병렬로 처리하기 때문에 속도를 높일 수 있음

모듈성

시스템 기능을 별도의 프로세스 또는 스레드로 분할하여 모듈 식 방식으로 시스템을 구성할 수 있음

편의

한 번에 여러 작업을 수행할 수 있음

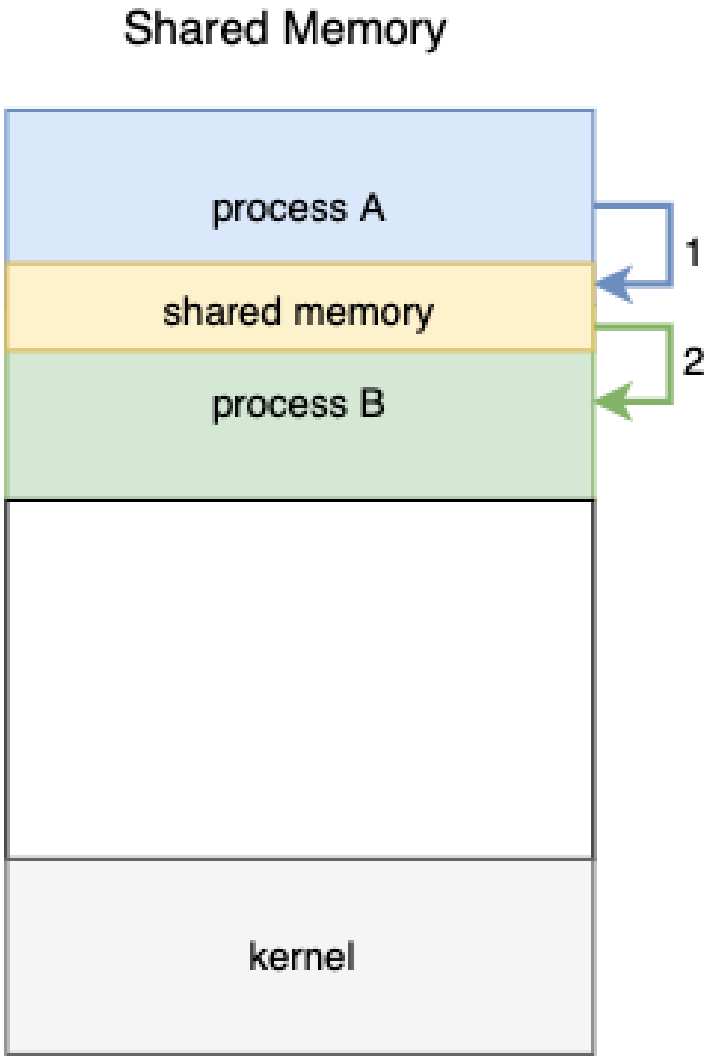
프로세스 간 통신(IPC : Interprocess Communication)

협력 프로세스 사이에서 서로 데이터를 주고받는 방법

IPC의 기본적인 두 가지 모델.

-Shared memory

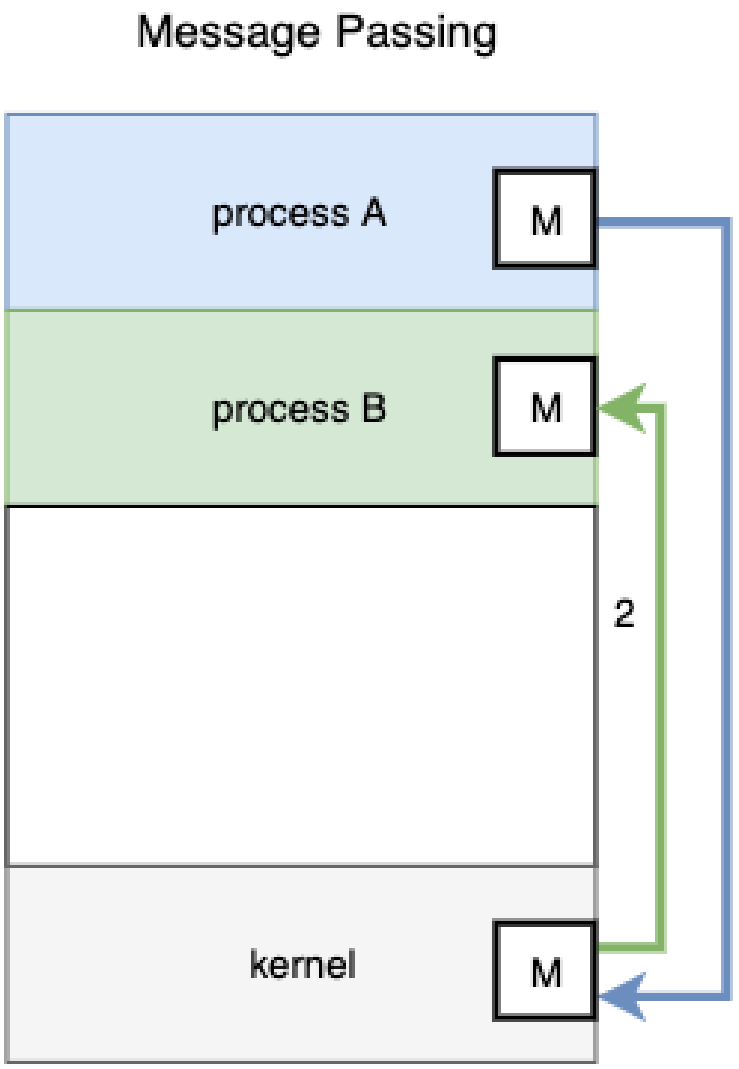
-Message passing



서로 다른 프로세스 간에 일부 주소 공간을 공유하여 데이터를 주고받는 방식.

장점
메모리를 직접 접근하기 때문에 message pasing 모델보다 속도가 빠름.

단점
프로세스 A가 공유 메모리에 데이터 전달해도 프로세스 B가 그것을 알 수 없음
-> 별도의 동기화 기술이 필요함



커널 메모리 영역에 메시지 전달을 위한 채널을 만들어서 협력하는 프로세스들 사이에 메시지 형태로 정보를 Send/Receive 하는 방법.

장점

커널에서 데이터의 주고받음을 컨트롤할 수 있어 별도의 동기화 로직이 없어도 가능.

단점

커널을 통해서 데이터 주고받기 때문에 shared memory 모델보다 느림.

message passing 두 가지 방법

Direct Communication :
통신하려는 프로세스의 이름을 명시적으로 표시하여
메시지를 직접 전달하는 방식

Indirect Communication :
mailbox(또는 port)를 통해 메시지 간접 전달하는 방식