핵심 운영체제

02 프로세스

목차

프로세스란 무엇인지,
어떻게 관리되는지
알아봅니다.

01

프로세스의 개념

프로세스(Process)란?

실행 중인 프로그램

프로그램(Program)이란?

특정 작업을 수행하는 명령어들의 모음

특정 작업을 수행하는 명령어들의 모음인 프로그램이 실행되면 프로세스

스레드(Thread)란?

프로세스 안에서 실행되는 흐름 단위

프로세스에는 하나 이상의 스레드가 존재한다.

프로세스 vs 스레드

프로세스는 독립적인 메모리를 할당 받는다

프로세스는 독립적인 메모리를 할당받고 스레드는

프로세스 속에 있는 것이라서 프로세스가 받은 메모리를 공유

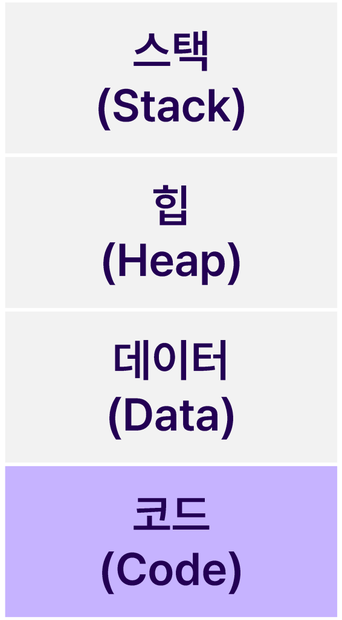
02

프로세스의 구조

프로세스의 구조

	설명
스택 (Stack)	임시 데이터(함수 호출, 로컬 변수 등)이 저장되는 영역
힙 (Heap)	코드에서 동적으로 생성되는 데이터가 저장되는 영역
데이터 (Data)	전역 변수, 정적변수, 배열, 구조체 등이 저장되는 영역
코드 (Code)	CPU에서 실행할 명령어를 저장하는 영역

코드(Code) 영역



- Text 영역이라고도 함
- 컴파일된 코드가 들어가는 영역으로 읽기 전용 영역
- CPU가 실행하게 될 명령어들이 저장됨

용어 해설

컴파일(compile)

컴퓨터가 이해할 수 있는 기계어로 변환해주는 것. 이런 변환을 해주는 프로그램을 컴파일러라고 합니다.

정적과 동적

정적(static)	동적(dynamic)
변하지 않는다.	변한다.

정적과 동적

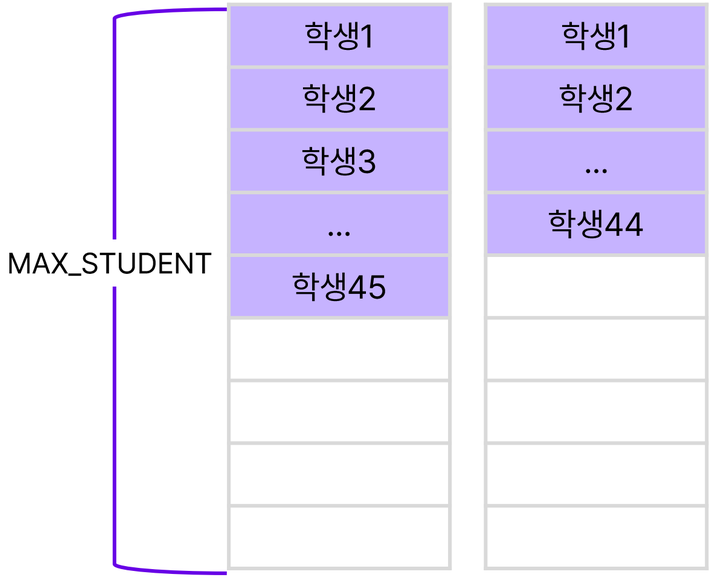
정적(static)	동적(dynamic)
실행 중에 변하지 않는다.	실행 중에 변한다.

정적 할당과 동적 할당

정적 할당	동적 할당
• 실행 전에 미리 메모리를 할당한다. • 할당받을 크기가 미리 정해져 있다.	• 실행 도중에 메모리를 할당한다. • 실행 도중에 할당 받을 크기가 변할 수 있다.

EliceAcademy

정적 할당의 예시



정적 할당의 예시

- 미리 최대 학생 수를 정해두고 메모리를 할당받음
- 학생 수에 비해 많은 메모리가 필요
- 할당받고 사용하지 않는 메모리가 발생
- 예상보다 많은 학생이 있다면 코드를 수정해야 함
- 실행 중에 할당받는 과정이 없기 때문에 속도가 빠름

EliceAcademy

동적 할당의 예시

동적 할당의 예시

- 실행 도중에 학생 수에 따라 메모리를 할당받음
- 정확하게 필요한 만큼만 메모리를 사용
- 학생 수가 늘어나도 추가로 할당받을 수 있음
- 추가로 메모리를 할당받는 과정으로 처리가 지연될 수 있음
- 메모리가 부족한 경우 프로그램이 멈출 수 있음

데이터(Data) 영역

- 전역 변수와 **정적 변수**들을 위한 공간
- 프로그램의 시작과 함께 할당되며, 프로그램이 종료되면 소멸한다.
- **프로그램이 시작되는 시점부터 종료 시점까지 추가되거나 사라지지 않는 공간**

힙(Heap) 영역

스택
(Stack)

힙
(Heap)

데이터
(Data)

코드
(Code)

- 동적 메모리 할당을 위해 사용되는 메모리 영역
- 사용자가 직접 관리할 수 있는 메모리 영역
- 상대적으로 느리며 관리를 잘못할 경우
  메모리 누수가 발생

용어 해설

메모리 누수(Memory Leak)

동적으로 할당된 메모리를 해제하지 않은 채로 사용하지 않는 상태로 남겨두는 현상. 프로그램의 성능 저하와 안정성 문제를 유발할 수 있습니다.

EliceAcademy

스택(Stack) 영역

스택 (Stack)
힙 (Heap)
데이터 (Data)
코드 (Code)

- 함수의 호출과 관련된 지역 변수와 매개변수가 저장됨
- 함수의 호출과 함께 할당되고, 호출이 완료되면 소멸
- 프로세스별로 크기가 제한되어 있음

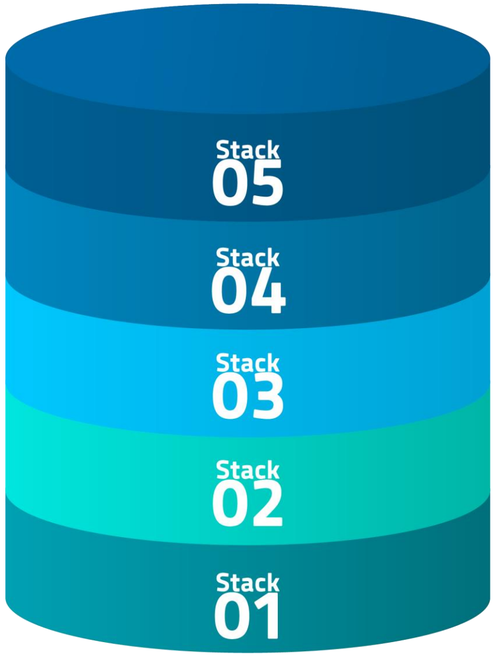
용어 해설

매개변수(parameter)

함수에 전달되는 변수. 함수는 입력값을 받아서 처리를 한 후에 결과값을 반환하는데, 이때 입력값은 함수를 호출할 때 매개변수를 통해 전달됩니다.

EliceAcademy

스택(Stack)



- 데이터를 저장하는 선형 자료 구조
- 자료를 넣는 것은 push라고 하고
  빼는 것을 pop이라고 함
- 후입선출(LIFO, Last-In First-Out)의 방식

용어 해설

후입선출

나중에 들어온 데이터가
먼저 나가는 방식.

EliceAcademy

스택(Stack) 영역

- 함수의 호출과 관련된 지역 변수와 매개변수가 저장됨
- 함수의 호출과 함께 할당되고, 호출이 완료되면 소멸
- 프로세스별로 크기가 제한되어 있음

용어 해설

매개변수(parameter)

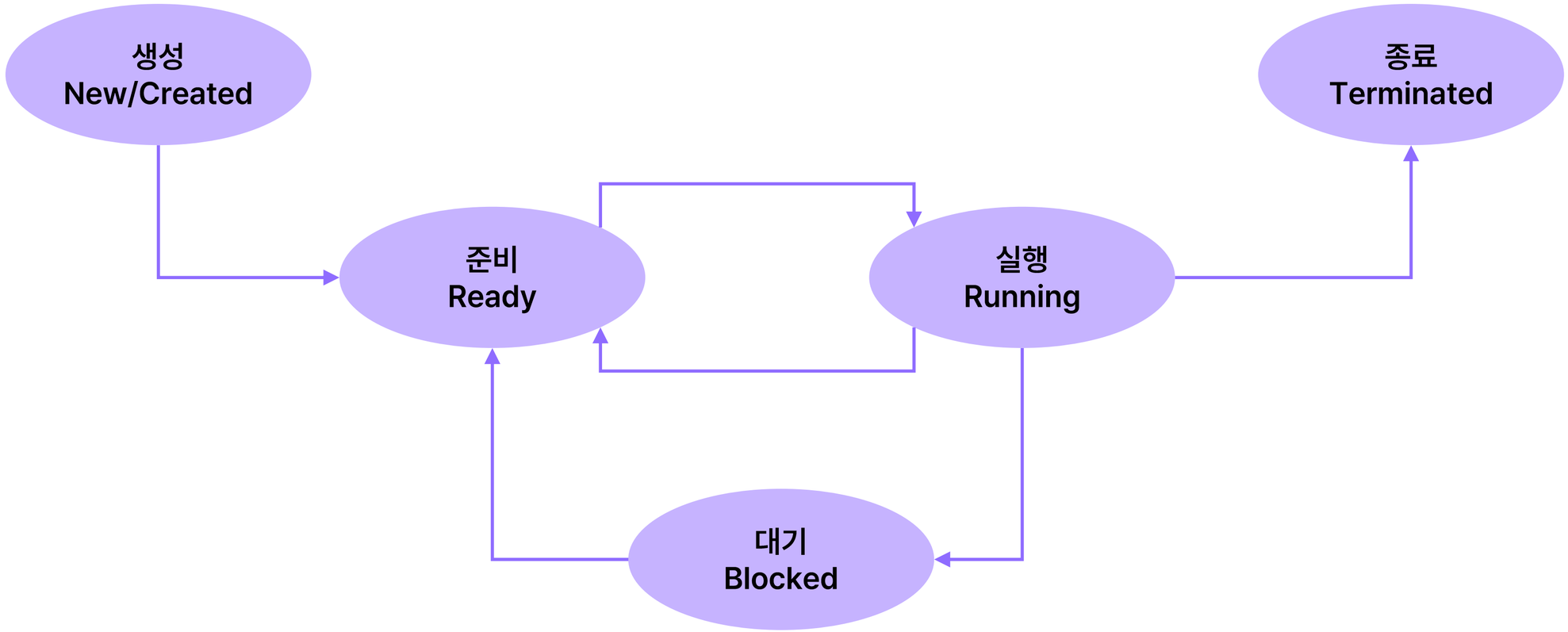
함수에 전달되는 변수. 함수는 입력값을 받아서 처리를 한 후에 결과값을 반환하는데, 이때 입력값은 함수를 호출할 때 매개변수를 통해 전달됩니다.

EliceAcademy

03

프로세스의 상태

프로세스 상태 전이도



생성(new/created)

번호표를 들고 식당밖에서 대기하는 단계

- 준비(Ready) 상태로의 승인을 기다리는 단계
- 아직 메모리에 로드되지 않은 상태
- 대부분 바로 준비상태로 승인되지만, 리소스가 부족하면 이 상태에서 오래 대기할 수 있음

준비(Ready)

식당에 들어가서 앉아 있는 상태

- 프로세스가 CPU를 사용하고 있지 않지만, 언제든 실행 될 수 있도록 대기하는 상태
- 운영체제는 우선순위가 높은 순서대로 CPU를 할당해서 실행(Running)상태로 전이함

실행(Running)

나온 요리를 먹는 단계

- CPU가 할당되어 프로세스의 명령어들을 처리해주는 단계
- 동시에 다른 프로세스들을 처리하기 위해서 일정 시간 이후 다시 Ready상태로 전이함

대기(Blocked/Waiting)

화장실 다녀오는 상태

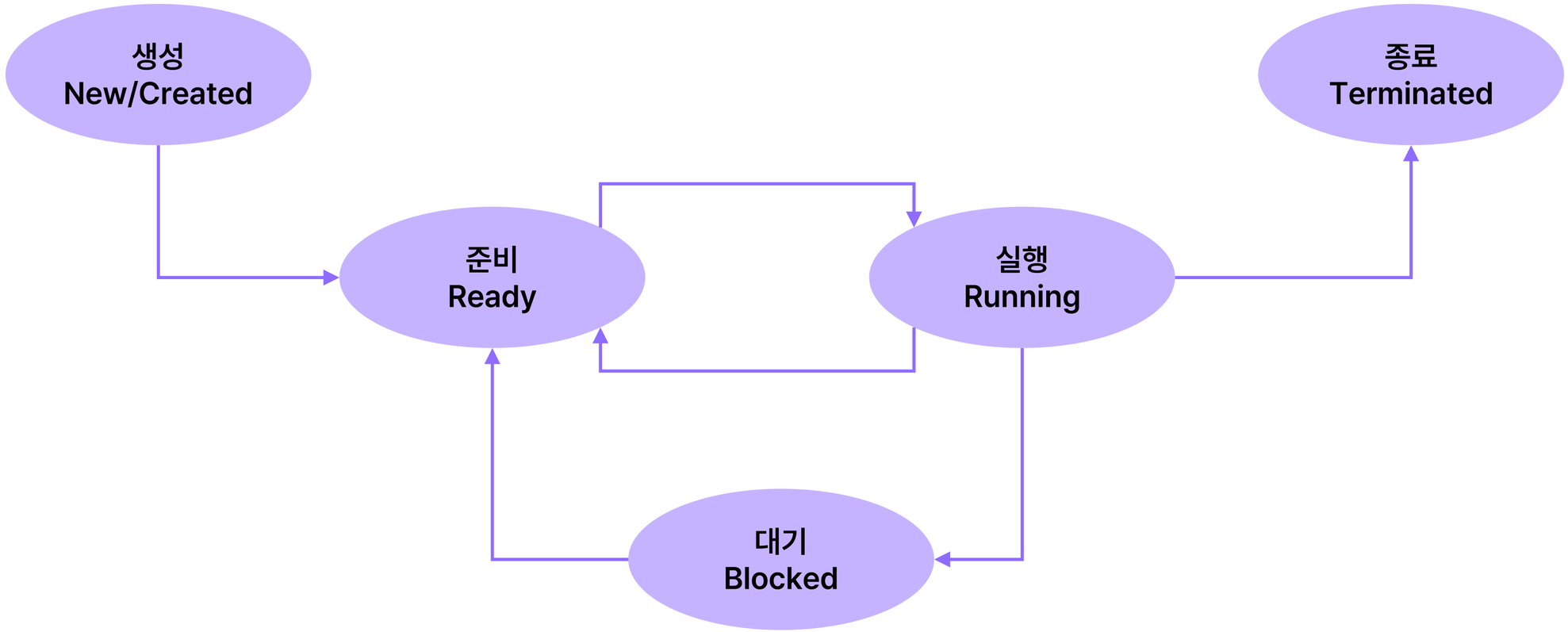
- 프로세스가 특정 자원이나 이벤트를 기다리는 상태
- 입출력 완료 등 특정 이벤트를 기다리는 상태
- 이벤트가 완료되면 Ready 상태로 옮겨와서 계속해서 실행

종료(Terminated)

일어나서 나갈 준비를 하는 단계

- 모든 명령어가 완료된 상태
- 부모 프로세스가 상태를 확인하기 위해 바로 삭제되지 않고 이 상태에서 대기
- 이 상태가 끝나면 프로세스가 메모리에서 삭제됨

프로세스 상태 전이도



EliceAcademy

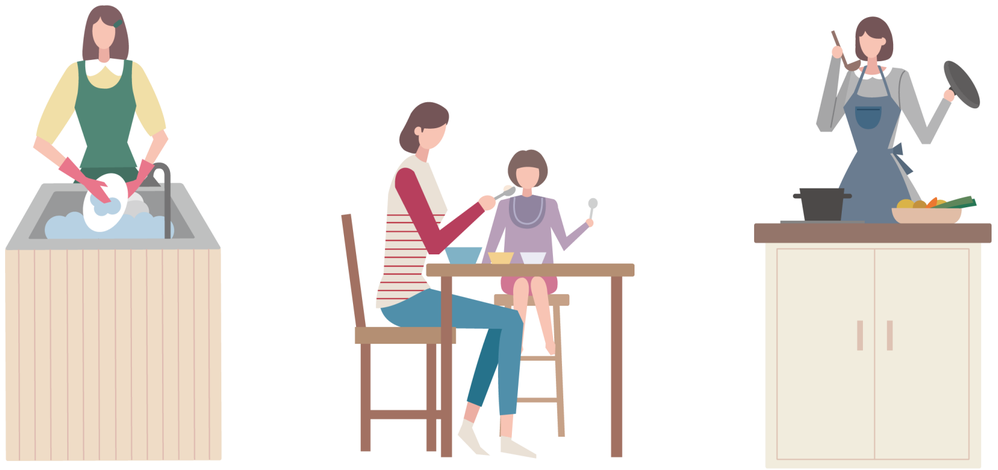
04

멀티 프로세싱

EliceAcademy

동시에 처리하기

동시에 여러가지 집안일 하기



멀티 프로세싱

여러 개의 프로세스를 돌아가면서 조금씩 처리

스레드(Thread)의 정의 돌아보기

프로세스 안에서 실행되는 흐름 단위

프로세스에는 하나 이상의 스레드가 존재한다.

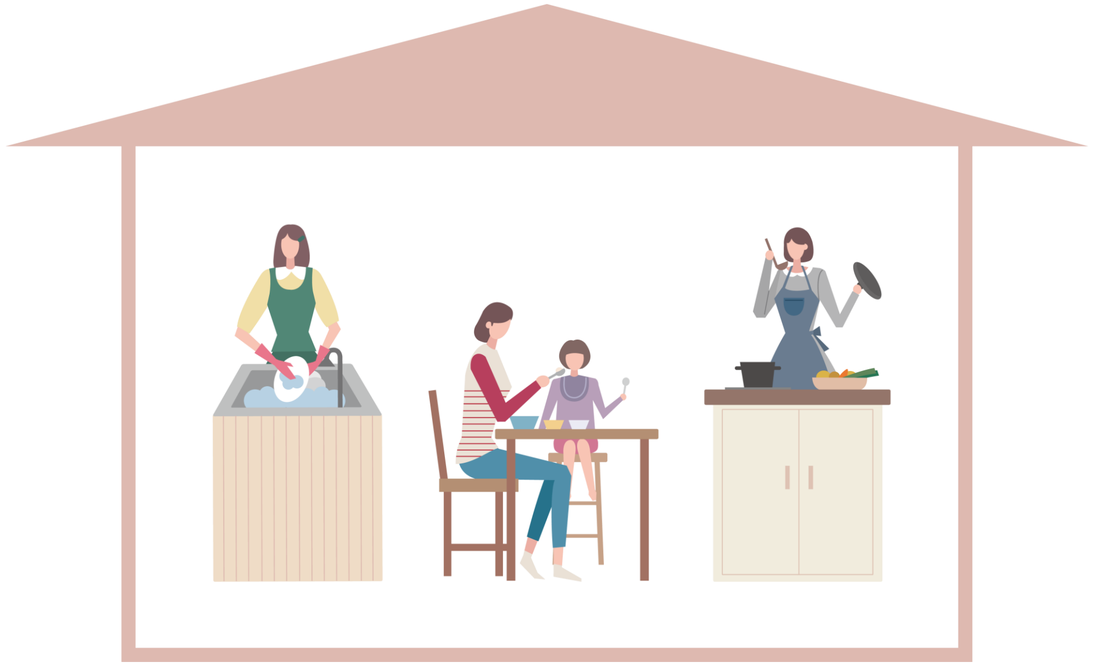
멀티 스레드

한 집안에서 여러 일을 처리하기

- 한 프로세스 내에서 자원을 공유하면서 여러 작업을 처리
- 새로 프로세스를 생성하고 리소스를 할당받는 과정이 없어서 더 빠르게 처리가 가능
- 하나의 스레드가 문제가 생기면 프로세스 전체가 종료될 수 있음

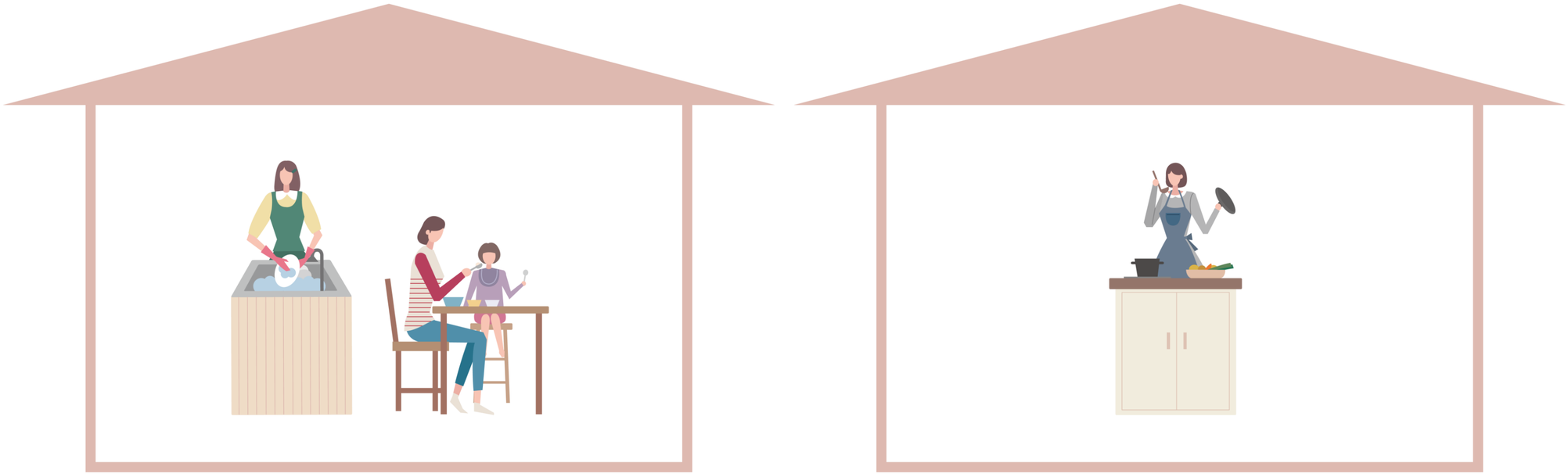
멀티 스레드

한 집안에서 동시에 여러가지 집안일 하기



멀티 프로세스

여러 집에서 동시에 여러가지 집안일 하기



프로세스간 통신(Inter Process Communication, IPC)

다른 집과 접시 효율적으로 주고 받기

- 다른 프로세스의 리소스는 절대 침범하면 안됨
- 운영체제가 허용하는 방법내에서 데이터를 주고 받는 방법들

프로세스간 통신

메시지 큐

- 공용 우편함에 넣어두기
- 데이터를 넣어두고 다른 프로세스들이 사용할 수 있게 하는 방법

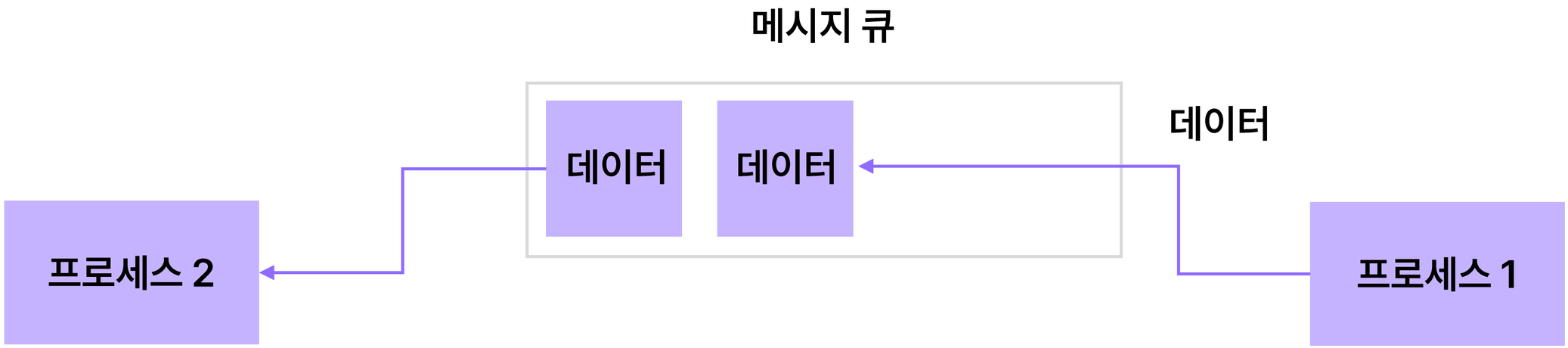
공유 메모리

- 현관문 열어 두기
- 프로세스의 메모리 일부를 공유 메모리로 만들어 다른 프로세스와 공유하는 방법

소켓

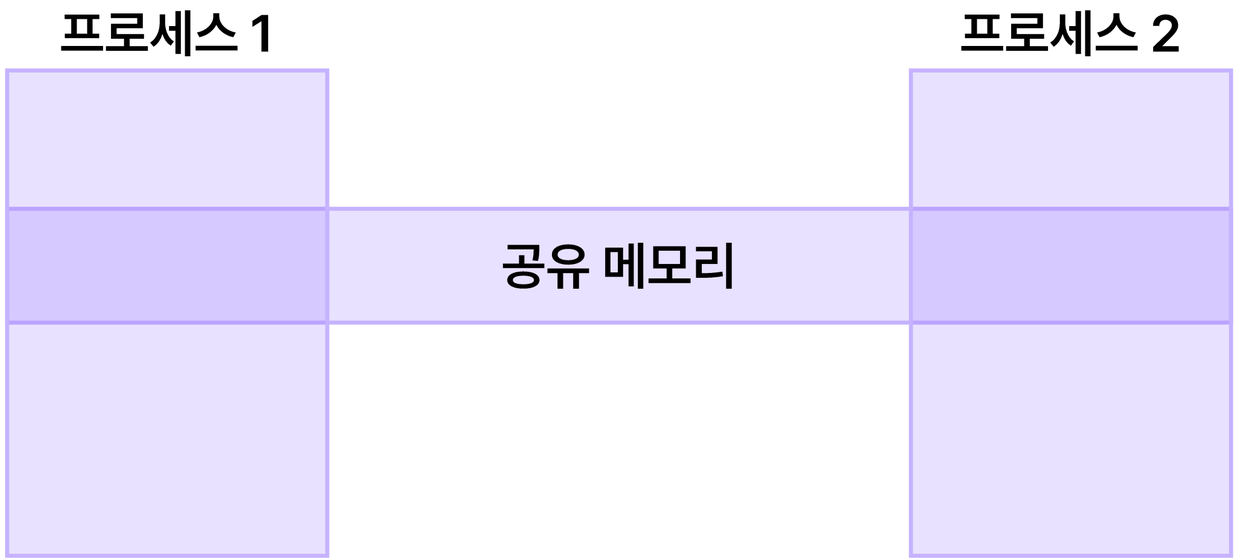
- 택배로 접시 보내기
- 소켓을 생성하여 다른 양방향 통신을 통해 데이터를 전달

메시지 큐(Message Queue)



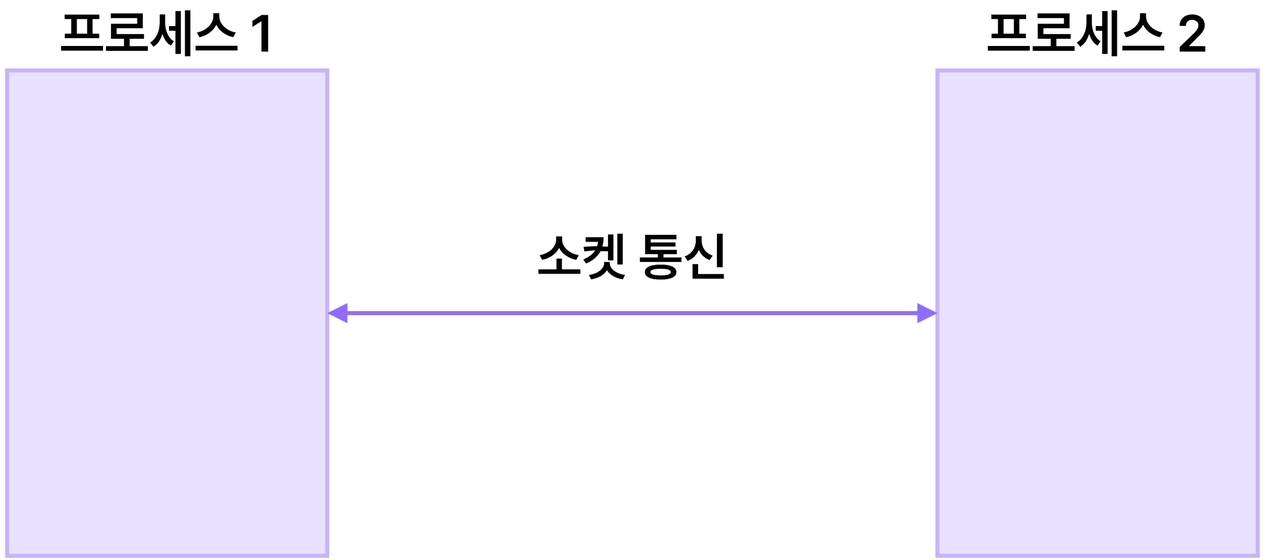
- 운영체제에서 관리하는 메시지 큐를 통해 통신하는 방법
- 데이터를 메시지 큐에 넣어두면 다른 프로세스가 데이터를 찾아서 다룰 수 있음

공유 메모리(Shared Memory)



- 프로세스의 메모리 일부를 공유 메모리로 만들어 다른 프로세스와 공유
- 데이터를 복사하는 과정들이 생략되어 상대적으로 빠르게 동작
- 리소스를 공유하기 때문에 충돌할 위험이 있음

소켓(Socket)



- 다른 네트워크에 있는 프로세스와도 통신할 수 있는 소켓 통신을 이용
- 통신 규약에 따라 통신하며, 데이터를 주고 받는 양방향 통신

05

프로세스 스케줄링

프로세스 스케줄링의 개념

어떤 손님부터 음식을 해줄 것인가

- 대기 시간은 가능한 최소화
- 가능한 공평하게 프로세스를 처리하는 것이 목적

프로세스 스케줄링의 종류

선입선처리

- 먼저 온 손님부터 차례대로 처리

최단 작업 우선

- 음식을 제일 적게 먹는 손님부터 처리

라운드 로빈

- 모두에게 일정 시간만큼만 처리를 해주는 방식

선입선처리

먼저 온 손님부터 순서대로 처리하는 방법

- 가장 먼저 도착한 프로세스부터 순서대로 처리하는 방법
- 구현이 간단하고 일괄처리에 효과적인 방법
- 빠른 응답을 요구하는 환경에서는 부적합

최단 작업 우선

음식을 가장 적게 먹는 손님부터 처리

- 평균 대기시간을 최소로 만들기 위한 알고리즘
- 작업이 얼마나 걸리는지 예측하기 어려움
- 오래 걸리는 작업은 매번 양보하면서 평생 처리가 완료되지 않을 수 있음

라운드 로빈 (Round Robin, RR)

모두 일정한 시간동안 순서대로 요리를 해주는 방식

- 일정 시간을 정해두고 시간이 지나면 대기열의 맨 뒤로 보내고 다음 작업을 처리
- 응답시간이 짧아지는 장점이 있어 실시간 시스템에 유리
- 프로세스 간의 전환이 많아지는 단점이 있음

06

교착상태와 기아상태

교착상태(Deadlock)

- 프로세스가 서로 상대방의 작업이 끝나기 만을 기다리고 있는 상태
- 결과적으로 아무것도 완료되지 않음

교착상태의 발생

상호배제	자원을 동시에 둘 이상이 사용할 수 없다.
점유대기	자원을 이미 가진 상태에서 다른 자원을 사용하려고 기다린다.
비선점	다른 프로세스가 자원을 뺏어올 방법이 없다.
순환대기	각 프로세스가 순환적으로 다음 프로세스가 요구하는 자원을 가지고 있다.

- 위 조건이 모두 만족해야 교착상태가 발생
- 위 조건 중 하나라도 막는다면 교착상태를 예방할 수 있다.

교착상태의 해결

은행원 알고리즘	자원을 할당받기 전에 운영체제가 검사를 하고 할당해준다
교착상태 무시	아무것도 하지 않는다
프로세스 종료	교착상태가 발견되면 해당 관련 프로세스를 종료시켜서 해결한다

- 아직까지 완벽하게 교착상태를 막을 수 있는 방법은 없음

기아상태(Starvation)의 개념

프로세스의 우선순위가 낮아서 원하는 자원을 계속 받지 못하는 상태

기아상태(Starvation)의 해결방법

모든 프로세스가 처리될 수 있도록 순서를 정하는 것이 중요

- 우선순위를 수시로 변경하여 계속해서 낮은 우선순위를 가진 프로세스가 없도록 함
- 오래 기다린 프로세스의 우선순위를 높이기
- 우선순위가 아닌 순서대로 처리