프로세스 vs 스레드

김현수

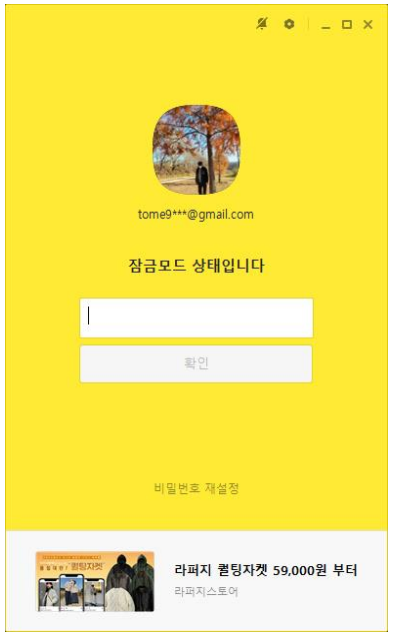
정의

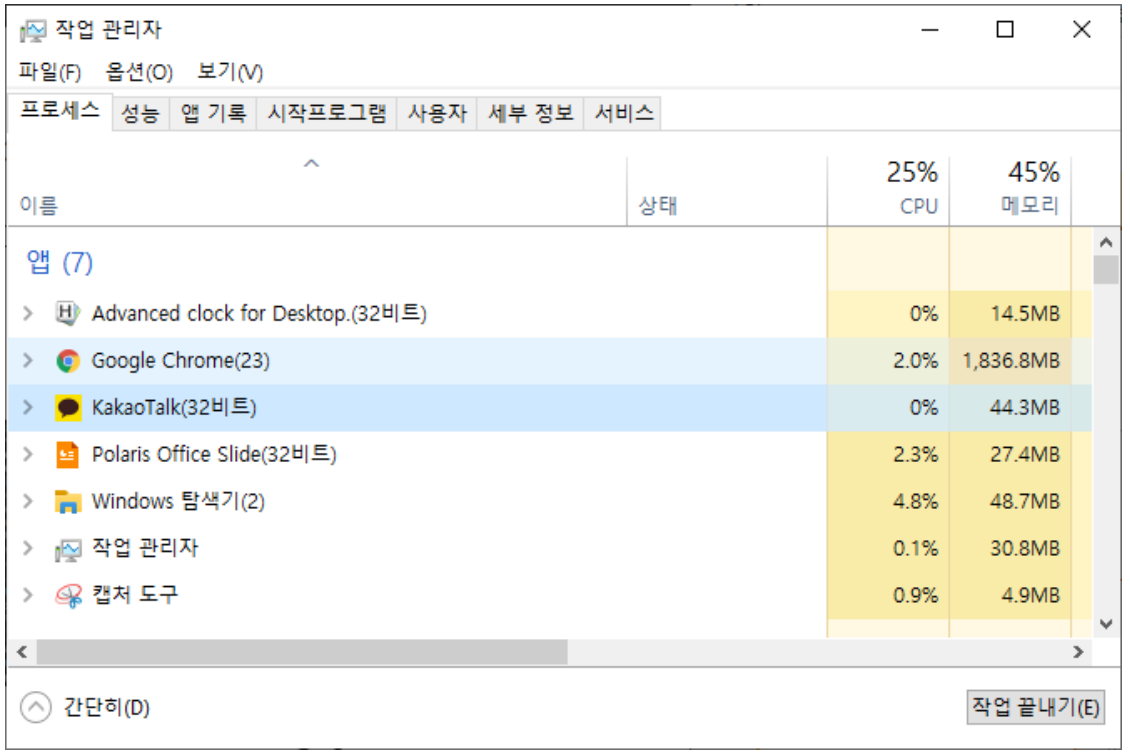
- 프로세스 : 운영체제로부터 자원을 할당받은 **작업**의 단위

- 스레드 : 프로세스가 할당받은 자원의 **실행 흐름**의 단위

프로그램 → 프로세스

- 프로그램 : 파일이 저장 장치에 저장되어 있지만 메모리에는 올라가 있지 않은 정적인 상태
  - 프로그램에 메모리가 할당되면? → **프로세스**





프로세스 → 스레드

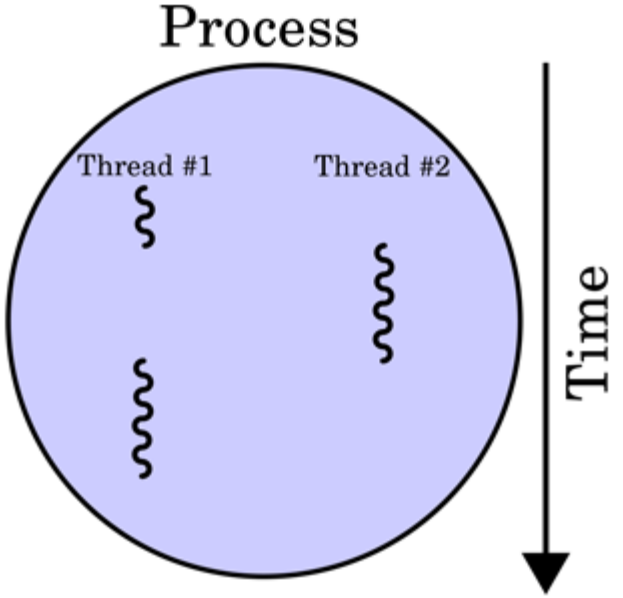
- 1프로그램 - 1프로세스
- 점점 복잡해지는 프로그램

- 1프로그램 -> 여러 개의 프로세스? : 불가능 왜?
- 각 프로세스마다 메모리가 할당되며 다른 프로세스에 접근하면 오류가 발생

- 그래서 생긴 개념이 스레드

스레드

- 스레드는 프로세스의 한계를 극복하기 위해 만들어진 개념
- 스레드는 메모리를 공유함
- 각 스레드는 프로세스의 실행 흐름
- 프로세스 : 프로그램의 코드 덩어리
- 스레드 : 코드 내에 선언된 함수들

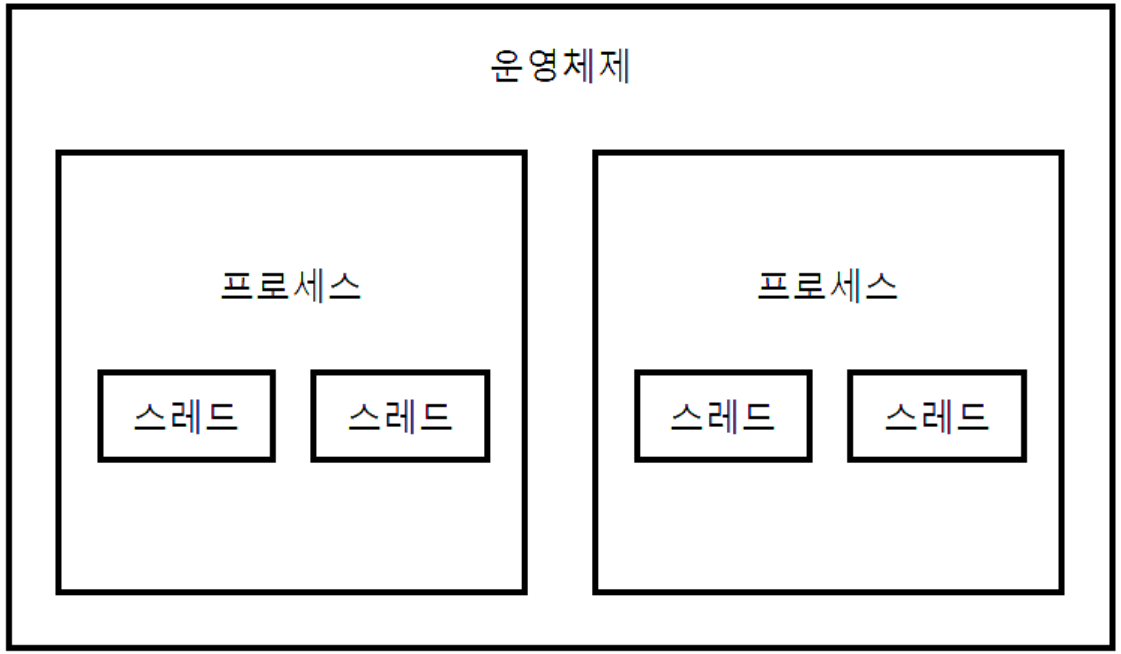


- 스레드는 **프로세스의 코드에 정의**된 절차에 따라 실행되는 특정한 **수행 경로**

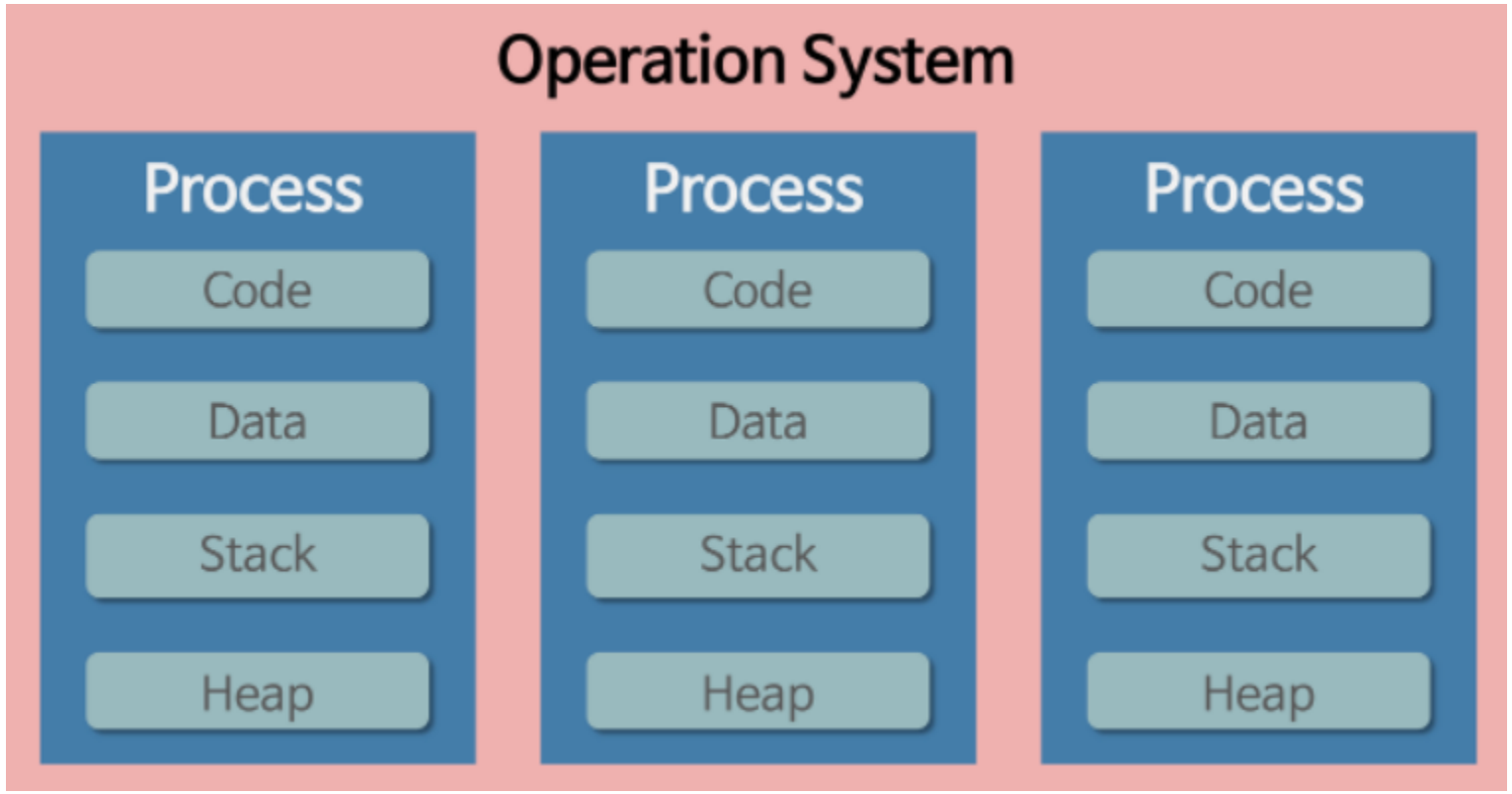
프로세스와 스레드의 차이

- 프로세스와 스레드의 차이를 설명하세요.

- 프로그램 -> 프로세스 -> 스레드

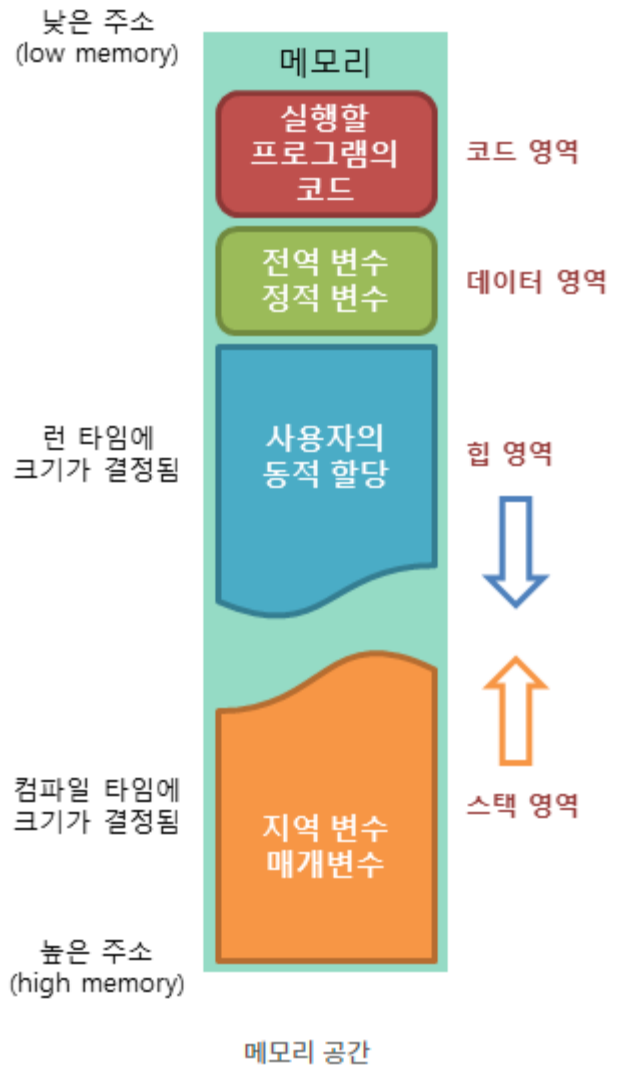


프로세스


Operation System
Process
Code
Data
Stack
Heap
Process
Code
Data
Stack
Heap
Process
Code
Data
Stack
Heap

메모리의 구조

- 코드
- 데이터
- 힙
- 스택



코드

- 실행할 프로그램의 코드가 저장되는 영역
- 텍스트 영역
- CPU는 코드 영역에 저장된 명령을 하나씩 가져가서 처리
- 프로그램이 시작하고 종료될 때 까지 메모리에 남아있음

데이터

- 전역 변수와 정적 변수가 저장되는 영역
- 프로그램의 시작과 함께 할당되며 프로그램이 종료되면 소멸

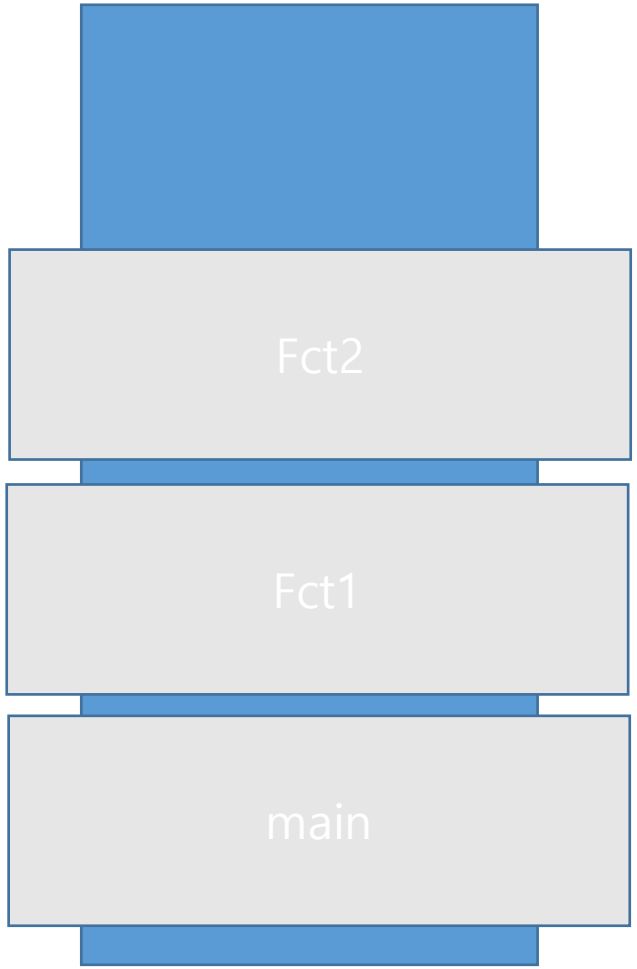
힙

- 프로그래머가 직접 공간을 할당, 해제하는 메모리 공간
- malloc을 통해 메모리 할당
- free를 통해 메모리 해제
- 선입선출 FIFO
- 낮은 주소에서 높은 주소의 방향으로 할당

스택

- 프로그램이 자동으로 사용하는 임시 메모리 영역
- 함수 호출 시 지역 변수와 매개 변수가 저장되는 영역
- 함수 호출이 완료되면 사라짐
- 후입선출 LIFO
- 높은 주소에서 낮은 주소의 방향으로 할당

예제 코드 - 스택




```c
#include <stdio.h>

void fct1(int);
void fct2(int);

int a = 10; // 데이터 영역에 할당
int b = 20; // 데이터 영역에 할당

int main() {

    int i = 100; // 지역변수 i가 스택 영역에 할당

    fct1(i);
    fct2(i);

    return 0;
}

void fct1(int c) {
    int d = 30; // 매개변수 c 와 지역변수 d 가 스택영역에 할당
}

void fct2(int e) {
    int f = 40; // 매개변수 e 와 지역변수 f 가 스택영역에 할당
}
```

스레드

- 스레드는 프로세스 중 하나의 실행 흐름
- 하나의 함수
- 함수끼리 구분하기 위해 스택만 따로 할당
- 그 외 메모리는 공유

강제 종료

- 프로세스 오류 발생 -> 다른 프로세스에 영향을 주지 않음
- 스레드 오류 발생 -> 같은 프로세스 내의 모든 스레드에 영향
- 코드를 짜다가 어떤 한 함수에서 오류가 발생하면 코드가 돌아가지 않음

엥 스레드 왜 씀?

- CPU의 최소 작업 단위 : 스레드
- 운영체제의 최소 작업 단위 : 프로세스
- 프로세스는 1개 이상의 스레드를 갖는다.
  -> 스레드는 반드시 메모리를 공유해야한다.
- 선택권이 없어요 ㅠ.ㅠ

멀티 스레드

- 멀티 프로세서 : 하나의 컴퓨터, 여러 개의 CPU
- 멀티 프로그래밍 : 하나의 CPU, 여러 개의 프로그램
- 멀티 스레드 : 하나의 프로세스, 여러 개의 스레드

- 멀티 스레드의 장점
  - Context switching할 때 공유하고 있는 메모리만큼의 메모리 자원을 아낄 수 있다.
  - 메모리를 공유하기때문에 응답 시간이 빠르다.

다음 시간에

- 멀티 스레드를 운용할 때 반드시 동기화 문제가 발생한다.
- 다음시간엔 동기화 문제에 대해서 알아보아요 ^^

- 앞선 발표에서는 다른 프로세스 메모리에는 접근할 수 없다고 했지만 사실 가능하답니다.
- 어떻게 하면 가능한지 다음시간에 알아보아요 ^^