



# thread와 process의 차이를 설명할 수 있나?

≡ 분류

Java



## [요약]

프로세스는 운영체제로부터 독립된 시간, 공간 자원을 할당 받아 실행된다는 점이고, 스레드는 한 프로세스 내에서 많은 자원을 공유하면서 병렬적으로 (Concurrently) 실행된다는 것이다.

[모든 차이는 근본적으로 이것에 의해 발생하는 것이다.]

## 프로세스(process)

### ▼ 사전적 의미

- 컴퓨터에서 연속적으로 실행되고 있는 컴퓨터 프로그램
- 메모리에 올라와 실행되고 있는 프로그램의 인스턴스(독립적인 개체)
- 운영체제로부터 시스템 자원을 할당받는 작업의 단위
- 동적인 개념으로는 실행된 프로그램을 의미

### ▼ 특징



- 프로세스는 각각 독립된 메모리 영역을 할당받는다.
- 기본적으로 프로세스당 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.

#### ▼ 할당받는 시스템 자원의 예

- CPU 시간
- 운영되기 위해 필요한 주소공간
- Code, Data, Stack, Heap의 구조로 되어 있는 독립된 메모리 영역

## 스레드(Thread)

---

#### ▼ 사전적 의미

- 프로세스 내에서 실행되는 여러 흐름의 단위
- 프로세스의 특정한 수행 경로
- 프로세스가 할당받은 자원을 이용하는 실행의 단위

#### ▼ 특징



- 스레드는 프로세스 내에서 각각 Stack만 따로 할당받고 Code, Data, Heap 영역은 공유한다.
  - 한 프로세스 내에서 동작되는 여러 실행의 흐름으로, 프로세스 내의 주소 공간이나 자원들(힙 공간 등)을 같은 프로세스 내에 스레드끼리 공유하면서 실행된다.
  - 같은 프로세스 안에 있는 여러 스레드들은 같은 힙 공간을 공유한다. 반면에 프로세스는 다른 프로세스의 메모리에 직접 접근할 수 없다.
  - 각각의 스레드는 별도의 레지스터와 스택을 갖고 있지만, 힙 메모리는 서로 읽고 쓸 수 있다.
  - 한 스레드가 프로세스 자원을 변경하면, 다른 이웃 스레드(sibling thread)도 그 변경 결과를 즉시 볼 수 있다.
- 

## 자바 스레드(Java Thread)란

- 일반 스레드와 거의 차이가 없으며, JVM 역할을 한다.
  - 자바에는 프로세스가 존재하지 않고 스레드만 존재하며, 자바 스레드는 JVM에 의해 스케줄되는 실행 단위 코드 블록이다.
  - 자바에서 스레드 스케줄링은 전적으로 JVM에 의해 이루어진다.
  - 아래와 같은 스레드와 관련된 정보도 JVM이 관리한다.
    1. 스레드가 몇 개 존재하는지
    2. 스레드로 실행되는 프로그램 코드의 메모리 위치는 어디인지
    3. 스레드의 상태는 무엇인지
    4. 스레드 우선순위는 얼마인지
  - 개발자는 자바 스레드로 작동할 스레드 코드를 작성하고, 스레드 코드가 생명을 가지고 실행을 시작하도록 JVM에 요청하는 일 뿐이다.
-

# 멀티 프로세스와 멀티 스레드의 차이

---

## 멀티 프로세스

- 정의 : 하나의 응용프로그램을 여러 개의 프로세스로 구성하여 각 프로세스가 하나의 작업(TASK)을 처리하도록 하는 것이다.
- 장점 : 여러 개의 자식 프로세스 중 하나에 문제가 발생하면 그 자식 프로세스만 죽는 것이므로 다른 영향이 확산되지 않는다.
- 단점
  1. Context Switching에서의 오버헤드
    - Context Switching 과정에서 캐쉬 메모리 초기화 등 무거운 작업이 진행되고 많은 시간이 소모되는 등의 오버헤드가 발생하게 된다.
    - 프로세스는 각각의 독립된 메모리 영역을 할당받았기 때문에 프로세스 사이에서 공유하는 메모리가 없어, Context Switching가 발생하면 캐쉬에 있는 모든 데이터를 모두 리셋하고 다시 캐쉬 정보를 불러와야 한다.
  2. 프로세스 사이의 어렵고 복잡한 통신 기법(IPC)
    - 프로세스는 각각의 독립된 메모리 영역을 할당받았기 때문에 하나의 프로그램에 속하는 프로세스들 사이의 변수를 공유할 수 없다.
- <참고> Context Switching 란?
  1. CPU에서 여러 프로세스를 돌아가면서 작업을 처리하는데 이 과정을 Context Switching라 한다.
  2. 구체적으로, 동작 중인 프로세스가 대기를 하면서 해당 프로세스의 상태(Context)를 보관하고, 대기하고 있던 다음 순서의 프로세스가 동작하면서 이전에 보관했던 프로세스의 상태를 복구하는 작업을 말한다.

## 멀티 스레드

- 멀티 스레딩이란?
  1. 하나의 응용프로그램을 여러 개의 스레드로 구성하고 각 스레드로 하여금 하나의 작업을 처리하도록 하는 것이다.
  2. 윈도우, 리눅스 등 많은 운영체제들이 멀티 프로세싱을 지원하고 있지만 멀티 스레딩을 기본으로 하고 있다
  3. 웹 서버는 대표적인 멀티 스레드 응용 프로그램이다.
- 장점
  1. 시스템 자원 소모 감소(자원의 효율성 증대)
    - 프로세스를 생성하여 자원을 할당하는 시스템 콜이 줄어들어 자원을 효율적으로 관리할 수 있다.
  2. 시스템 처리량 증가 (처리 비용 감소)
    - 스레드 간 데이터를 주고 받는 것이 간단해지고 시스템 자원 소모가 줄어들게 된다.
    - 스레드 사이의 작업량이 작아 Context Switching이 빠르다.
  3. 간단한 통신 방법으로 인한 프로그램 응답 시간 단축
    - 스레드는 프로세스 내의 Stack 영역을 제외한 모든 메모리를 공유하기 때문에 통신의 부담이 적다.
- 단점
  1. 주의 깊은 설계가 필요하다.
  2. 디버깅이 까다롭다.
  3. 단일 프로세스에서 스레드를 제어할 수 없다.( 즉, 프로세스 밖에서 각각을 제어할 수 없다.)
  4. 멀티 스레드의 경우 자원 공유의 문제가 발생한다. (동기화 문제)
  5. 하나의 스레드에 문제가 발생하면 전체 프로세스가 영향을 받는다.

---

## 멀티 프로세스 대신 멀티 스레드를 사용하는 이유

- 멀티 프로세스 대신 멀티 스레드를 사용하는 것의 의미

: 프로그램을 여러 개 키는 것보다 하나의 프로그램 안에서 여러 작업을 해결하는 것이다.





- 멀티 프로세스로 할 수 있는 작업들을 하나의 프로세스에서 여러 스레드로 나뉘가며서 하는 이유

#### 1. 자원의 효율성 증대

- 멀티 프로세스로 실행되는 작업을 멀티 스레드로 실행할 경우, 프로세스를 생성하여 자원을 할당하는 시스템 콜이 줄어들어 자원을 효율적으로 관리할 수 있다.
- 프로세스 간의 Context Switching시 단순히 CPU 레지스터 교체 뿐만 아니라 RAM과 CPU 사이의 캐쉬 메모리에 대한 데이터까지 초기화되므로 오버헤드가 크기 때문
- 스레드는 프로세스 내의 메모리를 공유하기 때문에 독립적인 프로세스와 달리 스레드 간 데이터를 주고 받는 것이 간단해지고 시스템 자원 소모가 줄어들게 된다.

#### 2. 처리 비용 감소 및 응답 시간 단축

- 또한 프로세스 간의 통신(IPC)보다 스레드 간의 통신의 비용이 적으므로 작업들 간의 통신의 부담이 줄어든다.
- 스레드는 Stack 영역을 제외한 모든 메모리를 공유하기 때문
- 프로세스 간의 전환 속도보다 스레드 간의 전환 속도가 빠르다.
- Context Switching시 스레드는 Stack 영역만 처리하기 때문

---

## 참고 - 메모리 영역




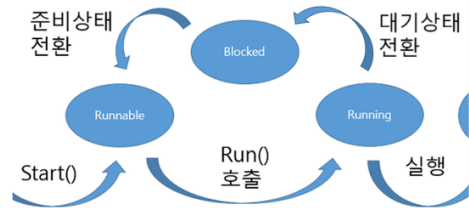
---

## References

### [Java] 자바 Thread(스레드) 사용법 & 예제


Thread란? 하나의 프로세스 내부에서 독립적으로 실행되는 하나의 작업 단위를 말하며, 세부적으로는 운영체제에 의해 관리되는 하나의 작업 혹은 태스크를 의미합니다. 스레드와 태스크(혹은 작업)은 바꾸

 <https://coding-factory.tistory.com/279>



### [OS] 프로세스와 스레드의 차이 - Heee's Development Blog

Step by step goes a long way.

 <https://gmlwjd9405.github.io/2018/09/14/process-vs-thread.html>