



Process & Thread



목차

🧐 Process

[프로그램이란?](#)

[프로세스란?](#)

[프로그램 실행 과정](#)

🐸 Thread

[스레드란?](#)

[특징](#)

🦊 멀티 프로세스 vs 멀티 스레드

[멀티 프로세스](#)

[장점](#)

[단점](#)

[멀티 스레드](#)

[장점](#)

[단점](#)

🔗 [참고](#)



Process

프로그램이란?

- 윈도우의 `*.exe` 파일이나 맥의 `*.dmg` 파일과 같은 컴퓨터에서 실행할 수 있는 파일
- 아직 파일을 실행하지 않은 상태이기 때문에 정적 프로그램이라 부른다
- 일종의 코드 덩어리

프로세스란?

- 프로그램을 실행시켜 정적인 프로그램이 동적으로 변하여 **프로그램이 돌아가고 있는 상태**
 - 즉, 컴퓨터에서 작업 중인 프로그램
- 프로그램을 실행하는 순간 파일은 컴퓨터 메모리에 올라가게 됨
- 운영체제로부터 시스템 자원을 할당받아 프로그램 코드를 실행시켜 사용자가 서비스를 이용

프로그램	프로세스
어떤 작업을 하기 위해 실행할 수 있는 파일	실행되어 작업중인 컴퓨터 프로그램
파일이 저장 장치에 있지만 메모리에는 올라가 있지 않은 정적인 상태	메모리에 적재되고 CPU 자원을 할당받아 프로그램이 실행되고 있는 상태
일종의 코드 덩어리	코드 덩어리를 실행한 것

프로그램 실행 과정

1. 프로그램이 실행되어 프로세스가 되면 메모리 영역에 프로세스의 주 구성요소인 Stack, Heap, Data, Code가 올라간다

Stack 영역 : 지역변수, 함수의 argument, 반환값 등의 일시적인 데이터가 저장되는 영역

Heap 영역: 동적 메모리 호출에 의해 할당되는 메모리 영역, new 연산자로 생성하거나 class 또는 참조 변수들도 힙 영역을 차지한다

Data 영역: 프로그램의 전역 변수와 정적 변수(static)가 저장되어 있는 영역, 프로그램이 끝날 때까지 메모리에 남아있는 변수

Code 영역 : 프로그램 코드 그 자체

2. 프로세스에 대한 정보를 담고 있는 PCB(Process Control Block)가 만들어진다



프로세스 제어 블록 (PCB)

- 운영체제가 프로세스를 제어하기 위해 정보를 저장하는 곳
- 프로세스의 상태 정보를 저장하는 자료구조
- 운영체제가 프로세스를 표현한 것

Pointer
Process State
Process Number (PID)
Program Counter
Registers
Memory Limits
Open File Lists
...

- 포인터 : 프로세스의 현재 위치를 저장하는 포인터 정보
- 프로세스 상태 : 생성, 준비, 실행, 대기, 종료 등의 상태를 저장
- 프로세스 식별자 : 프로세스를 일시적으로 식별할 때 사용
- 프로그램 카운터 : 프로세스를 위해 실행될 다음 명령어의 주소를 포함하는 저장
- 레지스터 : 누산기, 베이스, 레지스터 및 범용 레지스터를 포함하는 CPU 레지스터에 있는 정보



레지스터

레지스터란?

- 컴퓨터의 중앙 처리 장치(CPU) 내부에 있는 매우 빠른 메모리 장치
- 연산과 프로그램의 실행을 효율적으로 수행하기 위해 사용

종류

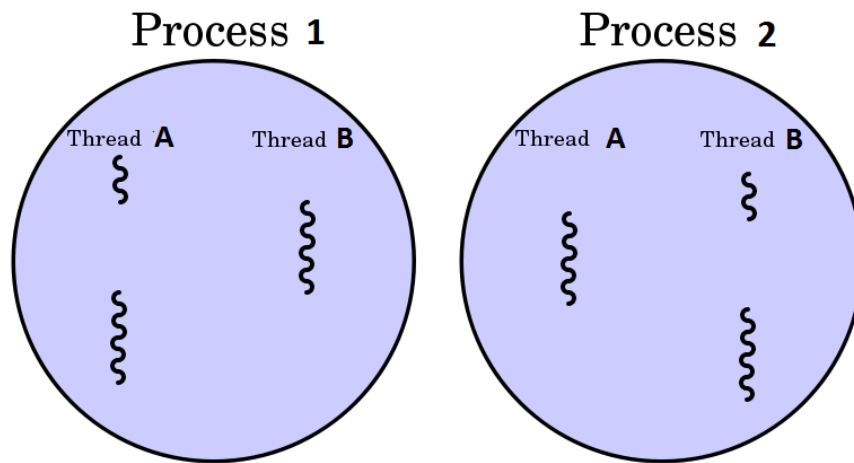
- 누산기 : 연산 장치에 있는 레지스터, 사칙연산 혹은 논리연산 등의 결과를 기억하기 위해 사용
- 베이스 레지스터 : 프로그램이나 데이터가 저장된 위치의 첫 번째 주소를 기억하는 레지스터



Thread

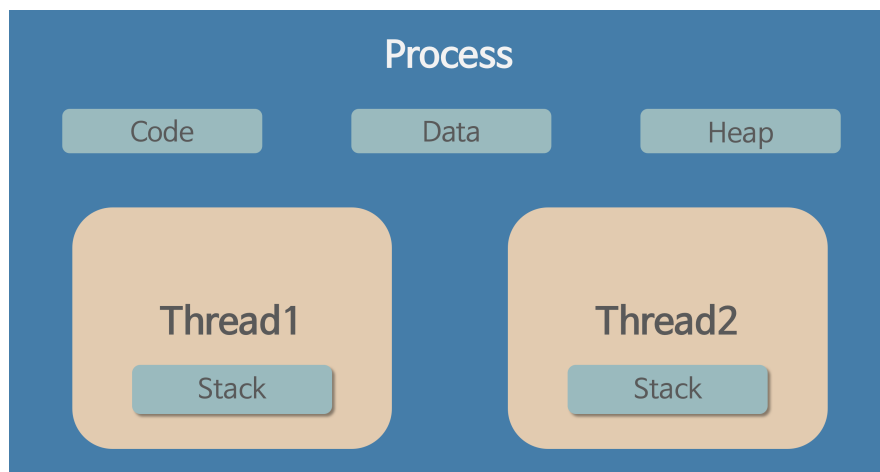
- 프로그램이 점점 복잡해지며 프로세스 작업 하나만을 사용해서 프로그램을 실행하기엔 한계가 존재
- 따라서 스레드라는 개념이 탄생

스레드란?



- 하나의 프로세스 내에서 동시에 진행되는 작업 갈래
- 프로세스가 할당 받은 자원을 이용하는 실행의 단위

특징



- 스레드는 프로세스 내에서 각각 Stack만 따로 할당받고 Code 영역, Data 영역, Heap 영역은 공유한다
- 스레드 또한 프로세스와 유사하게 TCB (Thread Control Block)을 가지며 다음과 같은 정보를 저장한다

Thread ID
Thread state
CPU information : Program counter Register contents
Thread priority
Pointer to process that created this thread
Pointer(s) to other thread(s) that were created by this thread

- ID : 스레드 식별 시 사용
- PCB 포인터 : 이 스레드를 생성한 프로세스를 가르키는 포인터
- 이 스레드가 생성한 다른 스레드를 가르키는 포인터

- 한 스레드가 프로세스 자원을 변경하면 다른 이웃 스레드도 그 변경 결과를 즉시 볼 수 있다
 - 이렇게 같은 자원을 공유하기 때문에 스레드간의 커뮤니케이션이 프로세스간의 커뮤니케이션에 비해 더 효율적이다
 - 또한 스레드간의 Context Switching이 프로세스간의 Context Switching보다 빠르게 이루어질 수 있다



Context Switching?

- 현재 진행하고 있는 Task의 상태를 저장하고 다음에 진행할 프로세스의 상태 값을 읽어 적용하는 과정

진행 과정

1. process P0이 실행되고 있는 상황, CPU 내부엔 P0의 정보가 저장중인 채로 실행
2. 프로세스 교체를 위해 현재 실행중인 P0을 저장
3. 저장을 위해 CPU에서 실행되던 P0과 관련된 Register 값들이 P0의 PCB에 저장됨
4. 다음 실행할 프로세스의 PCB 정보를 읽어 CPU 내부의 Register에 적재하고 CPU가 이전에 진행했던 과정을 연속적으로 실행

멀티 프로세스 vs 멀티 스레드

멀티 프로세스

- 하나의 응용프로그램을 여러 개의 프로세스로 구성해 각 프로세스가 하나의 작업을 처리하도록 하는 것

장점

- 여러 개의 자식 프로세스 중 하나에 문제가 발생하면 그 자식 프로세스만 죽는 것 이상으로 다른 영향이 확산되지 않는다

단점

- Context Switching에서의 오버헤드
 - 캐시 메모리 초기화 등 무거운 작업이 진행 됨
 - 프로세스는 각각의 독립된 메모리 영역을 할당받았기 때문에 프로세스 사이에서 공유하는 메모리가 존재하지 않음
 - 따라서 Context Switching이 발생하면 캐시에 있는 모든 데이터를 모두 리셋하고 다시 캐시 정보를 받아와야 한다
- 프로세스 사이의 어렵고 복잡한 통신 기법

멀티 스레드

- 하나의 응용프로그램을 여러 개의 스레드로 구성하고 각 스레드가 하나의 작업을 처리하도록 하는 것

장점

- 시스템 자원 소모 감소
 - 프로세스를 생성하여 자원을 할당하는 시스템 콜이 줄어든다
- 스레드 사이의 작업량이 작아 Context Switching이 빠름
- 간단한 통신 방법

단점


- 설계와 디버깅의 어려움
- 자원 공유시 동기화 문제가 발생한다
- 하나의 스레드에 문제가 발생하면 전체 프로세스가 영향을 받는다
 - 같은 주소 공간을 공유하기 때문에

참고

▼ 링크

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

Step by step goes a long way.

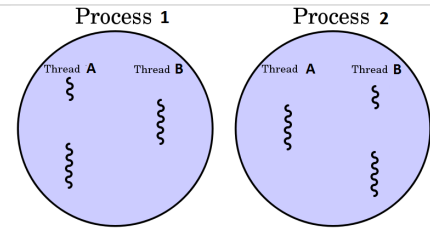
 <https://gmlwjd9405.github.io/2018/09/14/process-vs-thread.html>

<https://inpa.tistory.com/entry/%F0%9F%9F%9F-프로세스-%F0%9F%9F%9F-쓰레드-차이>

프로그램(Program), 프로세스(Process), 쓰레드(Thread)

👉 프로그램, 프로세스, 쓰레드 알아보기 프로세스와 쓰레드의 차이는 운영체제 공부에서도 아주 중요하게 다룬다. 개발자 면접에서도 자주 나오는 주제인 프로세스와 쓰레드에 대해 아는 것은 중요하다. 먼저 프로세스와

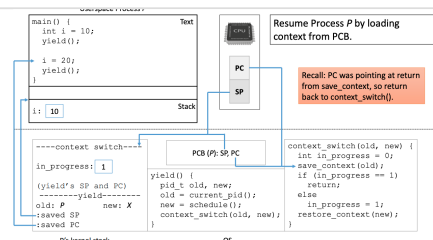
📄 <https://velog.io/@gparkkii/ProgramProcessThread>



Context Switching이란?

Process X가 진행이 완료되고 다시 Context Switch가 발생하게 된다면 이전 Process로 돌아올 준비를 합니다.

👤 <https://nesoy.github.io/articles/2018-11/Context-Switching>



[운영체제(OS)] 프로세스와 컨텍스트 스위칭(Context Switching)이란?

이번 시간에는 운영체제, 그리고 개발자의 상식에서 빼놓을 수 없는 프로세스에 대해서 알아보려고 합니다. 사실 일상생활에서도 많이 쓰이는 단어이기도 하죠. 0. 그럼 프로세스(Process)란 뭡까요? 프로세스는 아주아주 간단하게 설명하자면 '현재

🐢 <https://resilient-923.tistory.com/217>

