● Multi process와 Multi thread를 비교 설명해라. (하나의 프로그램을 one process, multi thread로 할지, multi process로 할지에 대해 - 한 프로세스로 할지, 여러 프로세스로 나누어서 짤지)

▶ Multi process와 multi thread은 동시에 여러 작업을 수행한다는 측면에서 유사한 면이 있지만, 각각의 장단점이 다릅니다. multi thread는 multi process보다 적은 메모리 공간을 차지하고 시스템 자원 소모가 적습니다. 또한 process를 생성하고 자원을 할당하는 등의 system call을 생략할 수 있기 때문에 자원을 효율적으로 관리할 수 있습니다. 뿐만 아니라, context switching 시 캐시 메모리를 초기화 할 필요가 없기 때문에 속도가 빠릅니다. 또한 데이터를 주고받을 때를 비교해 보면, process간의 통신(IPC)보다 multi thread간의 통신 비용이 적기 때문에, 통신으로 인한 오버헤드가 적습니다. 통신 시 별도의 자원을 이용하지 않고, process에 할당된 Heap 영역 등을 이용하여 데이터를 주고받기 때문입니다. 대신 멀티 쓰레드는 thread간 자원을 공유하기 때문에 동기화 문제가 발생할 수 있어서 프로그램 설계 시 주의가 필요합니다. (동기화 문제란, 서로 다른 thread가 메모리 영역을 공유하기 때문에 여러 thread가 동일한 자원에 동시에 접근하여 엉뚱한 값을 읽거나 수정하는 문제입니다.) 또한 하나의 thread에 문제가 생기면 process내의 다른 thread에도 문제가 생길 수 있고 전체 process가 종료될 위험이 있습니다. 반면, multi process는 multi thread보다 많은 메모리 공간과 CPU 시간을 차지하지만, 하나의 process가 죽더라도 다른 process에 영향을 주지 않아 안정성이 높습니다. 정리하자면, 적용할 시스템에 따라 두 방법의 장단점을 고려하여 적합한 방식을 선택해야 합니다. 메모리 구분이 필요할 때는 multi process가 유리합니다 (각 process마다 메모리를 독립적으로 할당받기 때문에). 반면에 context switching이 자주 일어나고 데이터 공유가 빈번한 경우, 그리고 자원을 효율적으로 사용해야 되는 경우에는 multi thread를 사용하는 것이 유리합니다.

	메모리 사용 / CPU 시간	Context Switching	안정성
multi process	많은 메모리 공간 / CPU 시간 차지	느림	높음
multi thread	적은 메모리 공간 / CPU 시간 차지	빠름	낮음

→ 캐시 메모리란? 데이터를 미리 복사해두는 임시 저장공간을 의미한다. 미리 자주 사용하는 데이터를 메모리 또는 디스크로부터 가져와서 CPU 안에 있는 캐시 메모리에 저장해놓고 CPU에서 바로 사용함으로써 더 효율적으로 연산을 처리할 수 있는 것이다. 프로세스 간 Context Switching이 발생하면 공유하는 데이터가 없으므로 캐시에 있는 모든 데이터를 전부 리셋하고 다시 캐시 정보를 불러와야 하기 때문에 시간이 걸린다.

Resources: inflearn 개발남 노씨