* program

: 작성한 코드를 빌드하여 생성된 결과물.

→ program을 실행하기 위해서는 memory를 할당 받아야 함.

이 때 자원을 할당받아 실행되는 프로그램을 process 라고 함.

* process

: 운영체제로부터 작업을 할당받은 작업의 단위.

메모리에 올라와 실행 중인 프로그램.

* process의 특징

- 작업을 수행하기 위해서 CPU / memory 등의 자원이 필요함.
- 프로세스별로 독립된 메모리 공간을 사용함. 여러 개의 프로세스 사이의 통신을 위한 IPC가 필요함.
- 하나의 프로세스 안에는 최소한 하나 이상의 thread가 있음. (main thread)
- 새로운 프로세스를 생성하는 프로세스를 부모 프로세스라고 하고, 새롭게 생성된 프로세스를 자식 프로세스라고 함.

* thread

: 어떤 프로그램 (프로세스) 내에서 실행되는 흐름의 단위.

* thread 의 특징

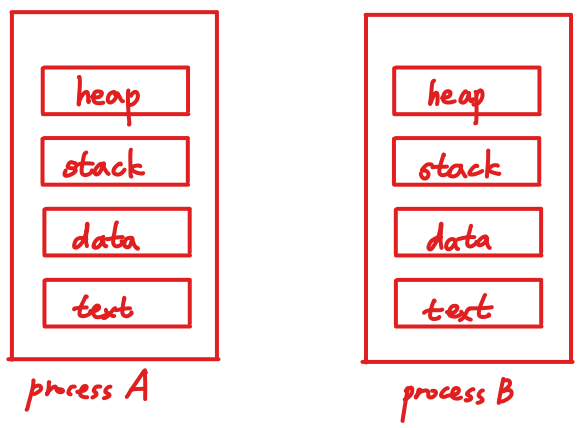
- 생성된 thread들은 메모리를 공유하여 사용함.
- 같은 프로세스 안에서 서로 독립적으로 실행이 됨.
- 부모 thread가 갖고 있는 데이터에 접근 할 수 있어 효율적이고, 메모리 제약이 있는 시스템에 널리 사용.

* memory 구조

1. text 영역 : 프로그램의 machine 명령어, 즉 실행가능한 코드가 들어있는 영역.
2. data 영역 : global 변수나 static 변수가 할당 된 영역.
3. stack과 heap 사이 영역 : stack과 heap에서 사용할 수 있는 공간으로 남겨 둔 영역.
4. stack 영역 : 지역변수, 매개변수, 반환값을 저장하는 공간.
5. heap 영역 : C 언어에서 malloc 함수처럼 동적으로 할당되는 영역.

* process 끼리 서로 독립적으로 메모리를 사용한다는 것의 의미

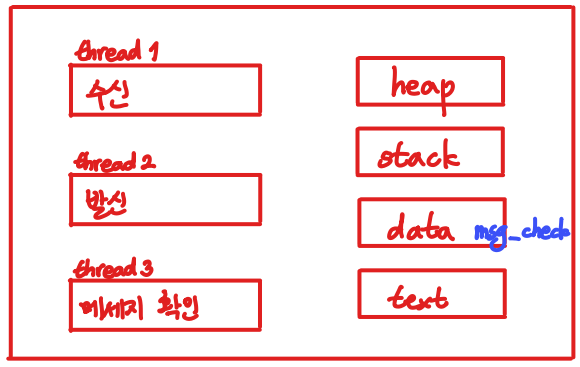
: process A 에서 실행되고 있는 프로그램에서는 process B 에 있는 heap / stack / data / text 에 관여할 수 없음.



→ process A를 크롬 브라우저 실행 프로세스, process B를 카카오톡 실행 프로세스라고 했을 때,
크롬을 조작한다고 해서 카카오톡 실행에 대한 제어를 할 수 없고,
카카오톡을 조작한다고 해서 크롬을 제어하거나 접근하는 것이 불가능함.
(프로세스들은 서로 독립적으로 실행이 되며, 서로의 메모리 영역은 공유되지 않기 때문.)

---

* thread는 하나의 프로세스 안에서 존재. 즉, 메모리 공유할 수 있음.



카카오톡에 메세지 수신 / 발신 / 확인에 대한 thread 만 있다고 가정하자.

thread 1 : 메세지를 받아내는 흐름.

thread 2 : 메세지를 보내는 흐름.

thread 3 : 메세지를 읽었는지 확인하는 흐름.

msg_check : 메세지를 확인했는지에 대한 정보, global 변수로 선언.
(data 영역에 할당)

→ A라는 친구에게 메세지를 받으면 (thread 2), 아직 메세지를 읽지 않으므로 msg_check = False 로 설정.
내가 A라는 친구의 메세지를 읽으면 (thread 3), msg_check = True 로 변경.
그 다음, A에게 답장을 보냄 (thread 1).
(각각의 thread 가 진행되면서 전역변수 msg_check 를 업데이트 할 수 있음.
: 같은 프로세스 안에서 실행되고 있기 때문에 thread 1, 2, 3은 메모리를 공유.)