6장	커널 오브젝트에 대한 이해

-1/10 이창현

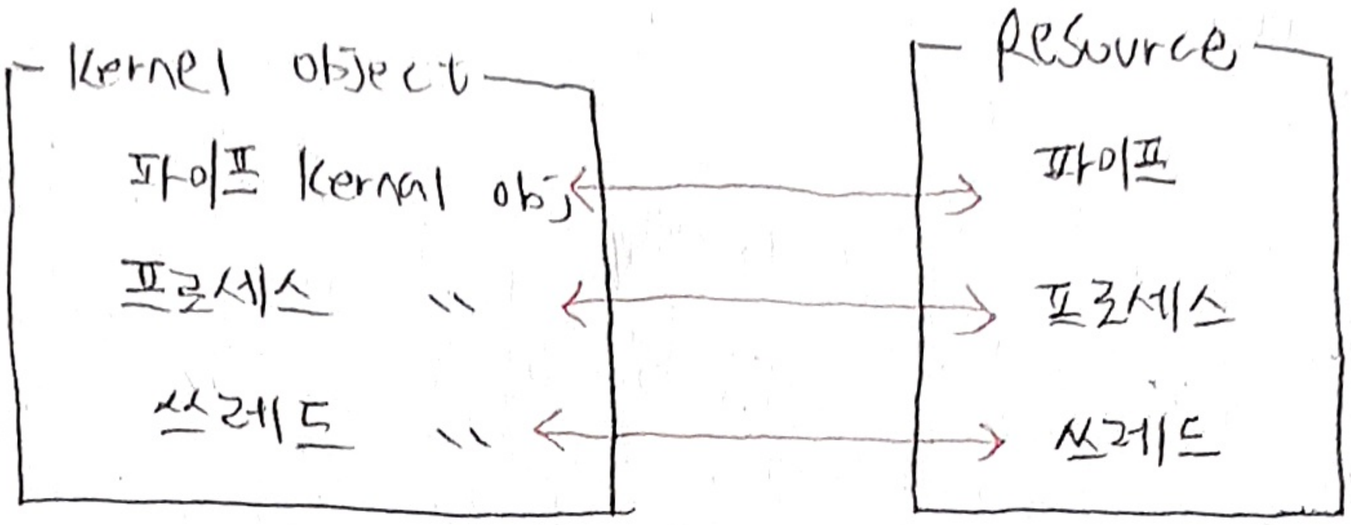
핵심요소!

<커널 오브젝트>

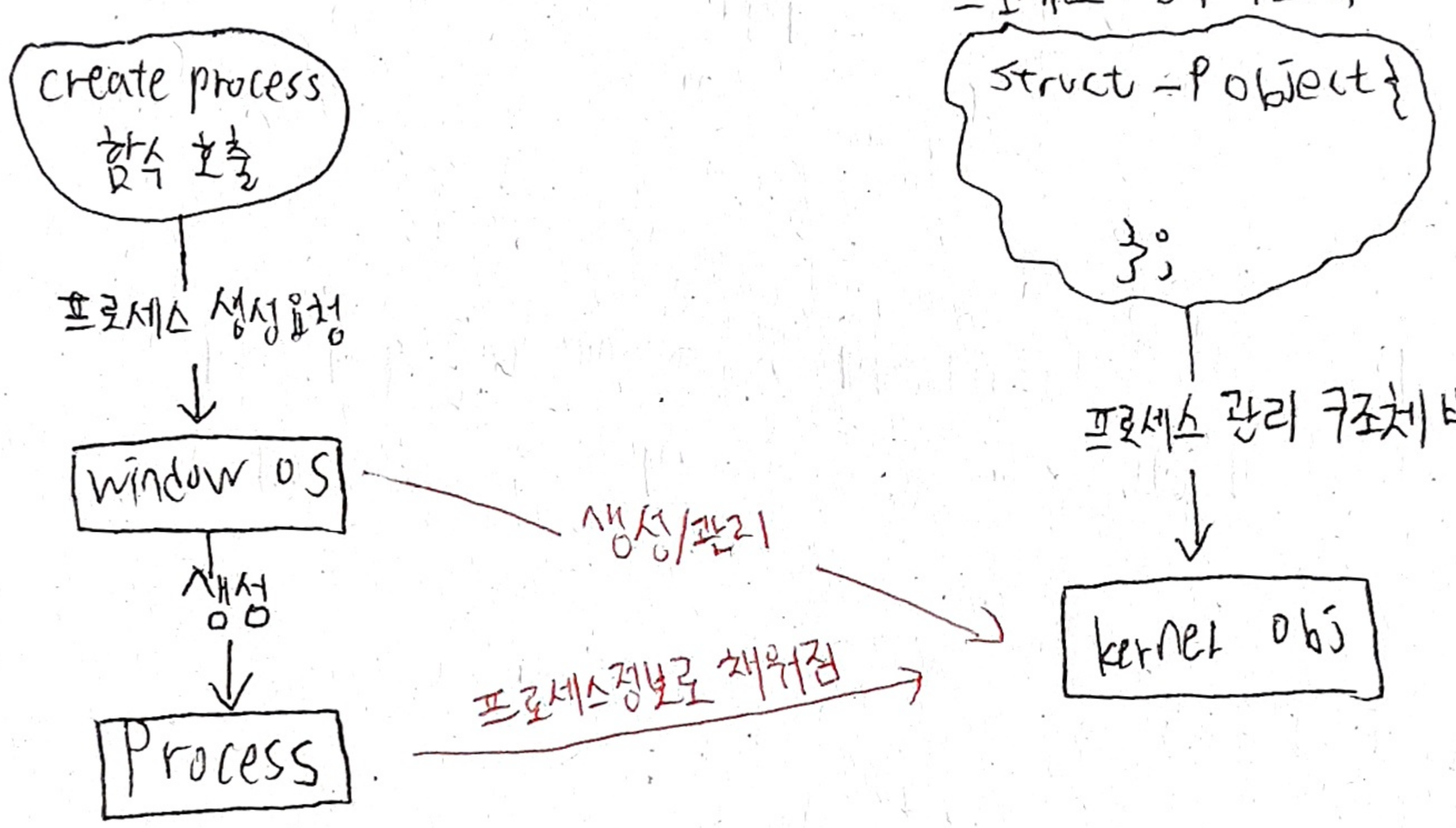
운영체제에 의해 생성/소멸

커널에 의해 관리되는 리소스 정보를 담고있는 데이터 블록

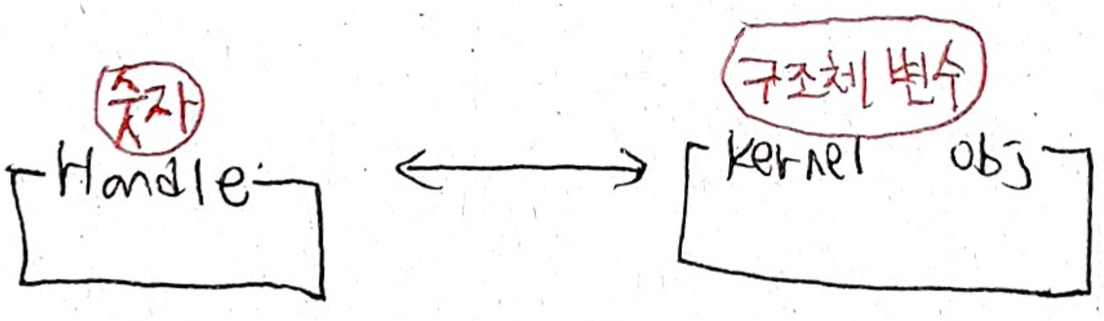
os



① 프로세스 kernel object



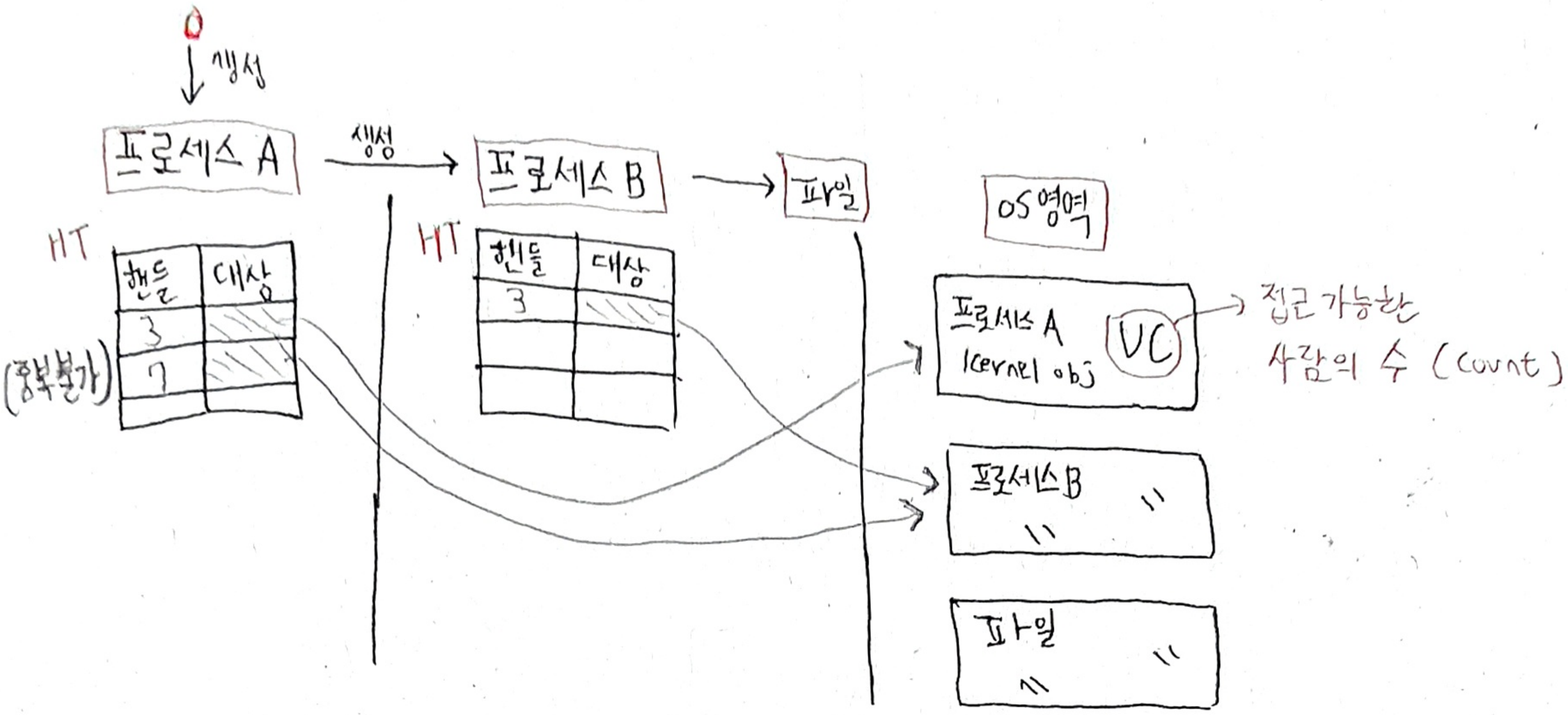
② 커널오브젝트와 핸들의 관계



= kernel obj의 직접적인 접근불가! => Handle 통해 간접적으로 특정 kernel obj 정보를 수정가능!

= 보통 정수로 mapping 되어있다.

<커널 obj와 핸들의 종속관계>



① 모든 일은 OS 내부에서 일어난다

② 프로세스 생성시, 커널 obj와 핸들 Table도 생성된다.

③ 각 프로세스 Kernel obj에 Handle 값 부여

↳ Kernel obj에 직접적 mapping이 아니라 Handle table 가서보니까 핸들의 key값을 참조하는 대상을 찾아가는 접근방식이다.

④ A의 VCC는 현재 자기 자신의 Kernel obj 접근 ⊕ (부모 프로세스인 동시에 자식 프로세스가 된다) = 2

⑤ B의 VCC는 4 자신 ⊕ A에 의해 B가 생성되었으므로 B의 핸들값이 A에게 반환!
= A는 B kerel obj에 접근할 핸들 값을 얻게 된다. ⇒ 따라서 2

⑥ kernel obj에 Handle 값이 정확하게 mapping 되는건 아니다.

⑦ Handle 값은 특정 프로세스 영역에서만 의미를 지닌다. (프로세스랑 종속적!)

⑧ 같은 Handle 값이라도 프로세스의 종속적 관계이므로 의미하는게 다르다.

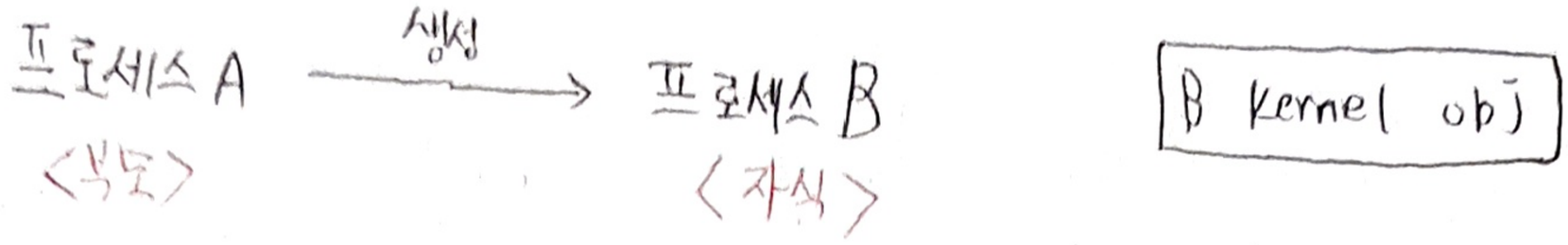
⑨ 파일의 VCC는 접근 가능한 프로세스는 B밖에 없다 = 1
(실제로 생성한 프로세스만 접근가능)

⑩ 만약 B 프로세스가 소멸시?,
프로세스 B kernel obj는 소멸이 안되는 대신 VCC=1로 된다 (아직 A가 관여하므로) → VCC의 필요성

VCC=0일때 그때서야 Kernel obj 소멸한다.

⑪ 파일 Kernel obj가 소멸되도 파일은 물리적으로 존재한다.

Q) 특정 프로세스가 소멸되었는데 굳이 그에 대한 Kernel obj를 두는 이유?



A) 커널까지 소멸시, 부모 Process는 자식 Process가 정상적 종료를 하고 나가는지 확인할 수 없다.
일반적으로 Kernel obj에 종료코드 (return 1/-1, exit --)가 내포되어있다.

Q) VCC=2 라는 뜻이 Handle table에 2군데 등록이 되었다는 뜻일까?

A) No! Handle table은 1군데만 등록되고 나머지는 자기 자신! (GetCurrentProcess)

<CloseHandle>

① VCC Count 줄일때 사용하는 함수, 프로세스가 종료되지 않아도 특정 Kernel에 연관이 없을때 사용됨 = 별개의 프로세스가 됨.

② Handle table에서 해당 handle에 대한 정보를 삭제하는 기능