**[메모리 관리 전략]**

1. **메모리 관리 배경**

: 각각의 프로세스는 독립된 메모리 공간을 가지고, 운영체제나 다른 프로세스의 메모리 공간에 접근할 수 없는 제한이 걸려있다. 오직 운영체제 만이 운영체제 메모리 영역과 사용자 메모리 영역의 접근에 제약을 받지 않는다.

1. Swapping

: 메모리 관리를 위해 사용되는 기법으로 표준 Swapping 방식으로는 Round-Robin과 같은 스케줄링의 다중 프로그래밍 환경에서 CPU 할당 시간이 끝난 프로세스의 메모리를 보조 기억장치로 내보내고 다른 프로세스의 메모리를 불러들일 수 있다. 이 과정을 Swap 이라고 한다. RAM(주 기억장치)로 불러오는 과정을 Swap-in, 보조 기억장치 (e.g.하드디스크)로 내보내는 과정을 Swap-out 이라고 한다. Swap에는 큰 디스크 전송시간이 필요하기 때문에 현재에는 메모리 공간이 부족할 때 시작된다.

1. Fragmentation(단편화)

: 프로세스들이 메모리에 적재되고 제거되는 일이 반복되다보면, 프로세스들이 차지하는 메모리 틈 사이에 사용하지 못할 만큼의 작은 자유공간들이 늘어나는 것으로 2가지 종류로 나뉜다.

1. 외부 단편화 : 메모리 공간 중 사용하지 못하게 되는 일부분으로 물리 메모리(RAM)에서 사이사이 남는 공간들을 모두 합치면 충분한 공간이 되는 부분들이 분산되어 있을 때 발생한다.
2. 내부 단편화 : 프로세스가 사용하는 메모리 공간에 포함된 남는 부분이다.
3. 압축

: 외부 단편화를 해소하기 위해 프로세스가 사용하는 공간들을 한쪽으로 몰아 자유공간을 확보하는 방법론이며, 작업효율이 좋지 않다는 특징이 있다.

1. **Paging(페이징)**

: 하나의 프로세스가 사용하는 메모리 공간이 연속적이어야 한다는 제약을 없애는 메모리 관리 방법이다. 외부 단편화와 압축 작업을 해소하기 위해 생긴 방법론으로, 물리 메모리는 Frame이라는 고정 크기로 분리되어 있고, 논리 메모리(프로세스가 점유하는)는 페이지라 불리는 고정 크기의 블록으로 분리된다.

1. 장점

: 논리 메모리는 물리 메모리에 저장될 때 연속되어 저장될 필요가 없고 물리 메모리의 남는 프레임에 적절히 배치됨으로 외부 단편화를 해결할 수 있다.

: 하나의 프로세스가 사용하는 공간은 논리 메모리에서 여러 개의 페이지로 나누어 관리되고, 개별 페이지는 순서에 상관없이 물리 메모리에 있는 Frame에 Mapping되어 저장된다.

1. 단점

: 내부 단편화 문제의 비중이 늘어나게 된다.

1. **Segmentation(세그멘테이션)**

: 페이징에서처럼 논리 메모리와 물리 메모리를 같은 크기의 블록이 아닌, 서로 다른 크기의 논리적 단위인 세그먼트(Segment)로 분할한다. 사용자가 2개의 주소로 지정하고, 세그먼트 테이블에는 각 세그먼트의 기준과 한계(세그먼트의 시작 물리 주소와 세그먼트의 길이)를 저장한다.

1. 단점

: 서로 다른 크기의 세그먼트들이 메모리에 적재되고 제거되는 일이 반복되다 보면, 자유 공간들이 많은 수의 작은 조각들로 나누어져 못 쓰게 될 수도 있다. (외부 단편화)