가상메모리 & 파일 시스템 & 동기화

운영체제의 메모리 관리란?

운영체제는 프로그램이 실행될 수 있도록 한정된 물리 메모리를 효율적으로 관리해야 한다.

하지만 물리 메모리는 제한되어 있으므로, 이를 보완하기 위해 가상 메모리(Virtual Memory) 개념이 등장했다.

가상 메모리(Virtual Memory)

> 실제 물리 메모리보다 큰 메모리를 사용하는 것처럼 보이게 만드는 기술

목적

- 물리 메모리의 한계를 극복
- 프로세스 간 메모리 보호
- 큰 프로그램 실행 가능

동작 원리

운영체제는 페이지 테이블(Page Table) 을 이용해 가상 주소(Virtual Address)를 물리 주소(Physical Address)로 매핑한다.

프로세스의 가상 주소 공간 → 실제 물리 주소 공간(RAM)

주소 변환(Address Translation)

- **페이지(Page)** : 가상 메모리의 단위
- **프레임(Frame)** : 물리 메모리의 단위
- **페이지 테이블(Page Table)** : 가상 페이지가 어느 프레임에 매핑되는지 관리

가상 페이지	물리 프레임
Page 0	Frame 2
Page 1	Frame 5
Page 2	Frame 1

페이지 교체(Page Replacement)

RAM이 부족할 때, 사용하지 않는 페이지를 디스크로 내보내고 새로운 페이지를 로드한다.

대표 알고리즘

알고리즘	설명
FIFO	가장 먼저 들어온 페이지를 교체
LRU	가장 오랫동안 사용되지 않은 페이지 교체
OPT	미래에 가장 늦게 사용할 페이지 교체 (이론적)
LFU	사용 빈도가 가장 낮은 페이지 교체

페이지 폴트(Page Fault)

> 필요한 페이지가 물리 메모리에 없을 때 발생하는 예외 상황

1. CPU가 가상 주소 접근
2. 페이지 없음 → 페이지 폴트 발생
3. 디스크에서 해당 페이지 로드
4. 페이지 테이블 갱신
5. 명령 재실행

정리

가상 메모리 → 실제보다 큰 메모리처럼 보이게 하는 기술

페이지/프레임 → 가상/물리 멤리의 단위

페이지 테이블 → 주소 매핑 정보 저장

페이지 교체 → 메모리 부족 시 불필요한 페이지 제거

페이지 폴트 → 요청한 페이지가 없을 때 발생하는 예외

파일 시스템(File System)

파일 시스템이란?

> 운영체제가 데이터를 저장·조직·관리하는 방법

프로그램은 파일 단위로 데이터를 저장하고, 운영체제는 이 파일들이 디스크에 어떻게 배치될지 관리한다.

파일 시스템의 역할

- 파일 생성, 삭제, 읽기, 쓰기 관리
- 파일 접근 권한 관리 (권한, 소유자 등)
- 디렉터리 구조 유지
- 디스크 공간 할당 및 회수

구성 요소

구성 요소	설명
파일(File)	데이터의 논리적 단위
디렉터리(Directory)	파일을 계층적으로 관리
인덱스 노드(Inode)	파일의 메타데이터(크기, 위치, 권한 등)
슈퍼블록(Superblock)	파일 시스템 전체 정보를 저장

파일 할당 방식

방식	설명	장점	단점
연속 할당 (Contiguous)	파일을 연속된 블록에 저장	빠른 접근	조각화 발생
연결 할당(Linked)	블록이 포인터로 연결	공간 효율적	임의 접근 느림
인덱스 할당(Indexed)	인덱스 블록이 파일 블록 주소 저장	빠른 접근	인덱스 블록 오버헤드

대표 파일 시스템 예시

OS	파일 시스템	특징
Windows	NTFS	보안, 저널링 지원
Linux	EXT4	안정적, 대용량 파일 지원
macOS	APFS	SSD 최적화
Android	F2FS	Flash Memory 전용 구조

파일 시스템의 계층 구조

응용프로그램 - 파일 시스템 인터페이스 - 논리적 파일 시스템 - 물리적 파일 시스템

동기화(Synchronization)

동기화란?

> 여러 프로세스나 스레드가 공유 자원에 접근할 때의 순서를 제어하는 기술

운영체제는 동시에 실행되는 스레드들이 데이터 불일치를 일으키지 않도록 관리해야 한다.

임계 구역(Critical Section)

> 둘 이상의 스레드가 동시에 접근하면 안 되는 코드 영역

해결 조건 (세 가지)

1. **상호 배제(Mutual Exclusion)**

   → 한 번에 하나의 프로세스만 임계 구역 실행

2. **진행(Progress)**

   → 실행 중인 프로세스가 없으면 대기 중인 프로세스가 진입 가능

3. **한정 대기(Bounded Waiting)**

   → 특정 프로세스가 무한히 대기하지 않음

주요 동기화 도구

도구	설명	특징
뮤텍스(Mutex)	한 스레드만 접근 허용	Lock/Unlock 방식
세마포어(Semaphore)	접근 가능한 스레드 수 제한	카운터 기반 제어
모니터(Monitor)	객체 단위로 동기화	Java의 `synchronized` 키워드 등
스핀락(Spin Lock)	반복 확인(바쁜 대기) 방식	짧은 임계 구역에 적합