1.

(1) 여러 개의 프로세스나 클라이언트가 동일한 데이터에 접근하여 그 데이터를 이용한 연산(수정, 삭제, 쓰기 등)을 수행하는 경우 상호배제(mutual exclusive)적인 실행을 보장하지 않을 때 발생하는 문제이다. 이 문제가 발생하는 경우 데이터를 이용한 연산의 수행 결과가 데이터에 접근한 순서에 따라 달라질 수 있게 되고, 사용자가 의도한 대로 데이터의 최종 상태가 적용되지 않는 등의 문제가 발생할 수 있다. 이를 해결하기 위해서 데이터 연산 수행 및 반영에 대해 상호배제적인 수행을 보장하는 방법을 사용한다.

(2) Semaphore의 wait 연산은 세마포어 값이 통과할 수 있는 값이면 (즉, 아직 프로세스가 진입할 수 있는 여유가 있으면) wait 연산이 수행되지 않고 critical section에 진입하게 된다. 하지만 Monitor 방식에서의 wait 연산은 진입할 수 있는지 여부를 검사하지 않고 무조건적으로 wait 연산을 한 번 거치게 된다. 다시 말해, Semaphore에서의 wait 연산은 상황에 따라 동작 여부가 결정되는 conditional한 함수이고, Monitor에서의 wait 연산은 상황에 상관 없이 무조건적으로 한 번은 실행되게 되는 unconditional 한 함수이다.

2.

RAID의 약자를 풀어보면 Redundant Array of Independent Disk, 즉 '독립된 디스크들의 복수 배열'이라고 할 수 있다. 이 시스템은 과거에 낮은 성능의 저장장치들을 하나로 묶어 하나의 고성능 저장장치처럼 사용하기 위해 도입하게 된 저장소 관리 기법이다. 지금은 이 목표 뿐 아니라 저장소와 관련된 다양한 목표를 달성하기 위해 사용되기도 하며, 0~6까지의 level을 통해 여러 저장 장치를 묶어 속도/안정성을 보장한다.

(1) RAID의 장점은 크게 3가지가 있다.

① 디스크 I/O 성능의 향상

RAID 0과 같이 여러개의 저장소를 병렬로 연결해 하나의 큰 저장소를 구성하면 읽고 쓰는 작업 또한 병렬로 처리되기 때문에 단일 저장장치를 사용해 데이터를 읽거나 쓰는 작업을 하는 것보다 훨씬 빠른 속도로 처리가 가능해진다.

② 안정성 확보

RAID 0을 제외한 다른 레벨의 방식을 사용하는 경우 하나의 데이터가 두 개의 물리적인 디스크에 미러링되어 저장되거나 패리티 데이터가 디스크에 저장된다. 이를 이용해 시스템이나 저장 장치에 고장이 발생했을 때 간편하게 데이터의 복구가 가능하다.

③ 저장 공간 확장/축소의 용이성

단일 저장 장치와는 달리 RAID 기법에서는 여러 개의 저장장치가 병렬적으로 사용되기 때문에 저장 공간의 크기 변경이 필요한 경우 병렬적으로 연결되는 저장 장치의 개수를 변경함으로서 확장과 축소가 가능하다.

(2) RAID 방식에서 신뢰도 (안정성) 향상을 위해 사용한 방법은 위에서 언급한 것처럼 미러링과 패리티의 두 가지로 분류할 수 있다.

- 미러링: RAID 1에서 사용되는 방식으로, 데이터를 기록할 때 두 개 이상의 저장장치에 동일한 내용을 기록하여 하나의 저장 장치가 고장나더라도 다른 저장장치를 이용해 데이터의 복구가 가능하도록 하는 방식이다.

- 패리티: 데이터를 바이트/블록 단위로 나누어 여러 개의 저장장치에 균등하게 저장하고, 오류 검사 및 수정을 위한 패리티 정보를 별도의 저장 장치에 기록함으로써 나중에 하나의 디스크가 고장나거나 오류가 발생하더라도 이를 이용해 데이터를 복구할 수 있도록 하는 방식이다.

3.

log-based 파일 시스템이 등장한 이유는 이전 파일 시스템들의 Data Recovery 방식의 단점을 개선하기 위해서다. System Crash가 발생하고 다시 시스템을 재기동할 때 파일 시스템의 내용에 오류가 있는지를 검사하는 프로세스 (가장 대표적으로 리눅스의 fsck 명령어가 있다.)가 있다. 파일의 양이나 시스템의 크기가 적당히 작은 과거에는 이 방식으로도 충분히 빠른 시간 안에 파일 시스템의 오류를 검사하는데 문제가 없었다. 하지만 현대의 파일 시스템을 그 크기가 방대해져 과거의 방식으로 검사하기에는 많은 시간 자원을 필요로 하게 되었고, 새로운 파일 검사 방식 중 하나로 파일 시스템에서의 변경점을 로그로 남겨 그 로그의 내용과 현재의 파일 시스템 내용이 일치하는지 여부를 검사하는 log-based(또는 journaling) File System이 도입되었다.

Journaling 파일 시스템은 모든 파일의 내용을 검사하는 것이 아니고 파일 시스템의 메타 데이터만 검사한다. 모든 파일 시스템의 변경 기록을 로그로 남기고 오류 검출을 하는 것은 검사 범위와 기록 범위가 너무 커지기 때문에 중요한 파일 변경점에 해당하는 파일 시스템의 메타데이터에 대해서만 로그 기록을 지원한다. 만약 일반적인 파일들도 중요시되는 시스템이라면 블록 단위까지도 Journaling 기능을 지원한다.