1. **Page fault handling**

* **프레임(Frame):**물리 메모리를 사용하는 최소 크기 단위.
* **페이지(Page):**가상 메모리를 사용하는 최소 크기 단위.

1. CPU는 물리 메모리를 확인하여 페이지가 없으면 trap을 발생하여 운영체제에 알린다.

2. 운영체제는 CPU의 동작을 잠시 멈춘다.

3. 운영체제는 페이지 테이블을 확인하여 가상 메모리에 페이지가 존재하는지 확인하고, 없으면 프로세스를 중단한다.

**4. 페이지 폴트이면, 현재 물리 메모리에 비어있는 프레임(Free Frame)이 있는지 찾는다. (Page Replacement Algorithm: FIFO, LRU, LRU Approximation)**

5. 비어있는 프레임에 해당 페이지를 로드하고, 페이지 테이블을 최신화 한다.

6. 중단되었던 CPU를 다시 시작한다.

1. **Process와 Thread의 차이**

**Process**: 컴퓨터에서 연속적으로 실행되고 있는 컴퓨터 프로그램을 말한다. 종종 스케줄링의 대상이 되는 작업(task)이라는 용어와 거의 같은 의미로 쓰인다. CPU 시간이나 메모리 등 시스템 자원이 할당되는 독립적인 개체

**Thread**: 어떠한 프로그램 내에서, 특히 프로세스 내에서 실행되는 흐름의 단위를 말한다.

같은 프로세스 소속의 스레드들은 메모리를 공유. (스택, 레지스터 제외)

* 동시에 여러 작업을 할 때 멀티 스레드가 시스템 자원을 더 적게 사용
* 프로세스는 메모리 공유를 하지 않으므로 프로세스 간 통신은 스레드 간 그것에 비해 복잡 (IPC – Signal, Pipe, Socket..)등의 기술을 사용해야
* 스레드의 경우 공유 자원에 대한 동기화(Synchronization) 문제 발생
* Context switching overhead

1. **Deadlock의 네 가지 조건**
   1. 상호 배제 (Mutual Exclusion) : 프로그램들이 공유 자원을 동시에 쓸 수 없는 상황
   2. 점유 상태로 대기 (Hold and Wait) : 공유 자원을 점유한 상태에서 다른 자원을 기다린다
      1. 자원을 확보할 수 없을 때 확보하고 있는 자원을 놓는다
   3. 선점 불가 (No preemption) : 자원을 어떤 프로세스가 점유 중일 때 다른 프로세스가 그 자원을 뺏을 수 없다
   4. 순환성 대기 (Circular wait) : 각 프로세스는 순환적으로 다음 프로세스가 요구하는 자원을 가지고 있다.
      1. 해결방법: Ordering (자원 유형에 따라 순서를 매김)

* 교착 상태의 예방(Prevention): 네 가지 중 한 조건이라도 제거 (System design)
* 교착 상태의 회피(Avoidance): 시스템이 동적으로 각 요청을 살펴보고 이 요청이 safe한지 아닌지를 결정 -> additional apriori information이 필요(Resource Allocation Graph나 Banker’s Algorithm 이용), 더 높은 Concurrency 확보 가능

1. **GET과 POST의 차이**

* 공통점: 클라이언트가 서버에 데이터를 전달
* 차이점
  + **GET:** 데이터 값이 URL 뒤에 이어붙음 ( ? 를 붙여 파라미터를 구분)

1. 전송하는 데이터량에 제한이 있음
2. 사용자에게 데이터가 그대로 노출
3. URL 형식에 맞지 않는 데이터는 인코딩이 필요
4. 브라우저에서 캐싱이 가능해 웹 크롤러가 그것에 접근하여 마음대로 실행할 가능성도 있음
   1. SQL쿼리의 SELECT와 같은 단순 조회에만 이용하는 것이 좋음
   * **POST:** 데이터를 HTTP body에 넣어 전달
5. 전송량에 제한이 없음
6. 사용자에게 데이터가 그대로 노출되지 않음
7. 암호화는 따로 없음
8. 브라우저에서 캐싱이 불가능
   1. 서버의 값이나 상태를 변경하는 경우에는 POST로 구현하는 것이 더 적절
9. **TCP와 UDP의 차이**
   1. TCP
      1. 애플리케이션이 네트워크 계층 문제를 걱정하지 않고 데이터를 안정적으로 송신할 수 있도록 하는 프로토콜
      2. 연결지향형 서비스 (전송 전에 연결을 먼저 맺어야)
      3. 전송하는 순서대로 데이터가 전달 (연속될 흐름-stream-의 데이터 송수신)
      4. 신뢰성 O 🡪 에러, 손실된 데이터는 자동으로 재전송
         1. 신뢰성을 위해 ACK, checksum 등 사용
      5. 전송되는 데이터의 경계가 존재하지x (한번에 보내지거나 받아질수도, 여러개로 나뉘어 보내지거나 받아질수도)
      6. 혼잡제어 기능 제공 (네트워크가 혼잡할 때 출발징의 전송 속도를 줄이도록 함)
   2. UDP
      1. 연결 수립이 없이 데이터를 송신
      2. 전송되는 순서에 상관없이 가장 빠른 전송을 지향 🡪 실시간 전송에 적합
      3. 데이터가 손실되어도 재전송하지 않음
      4. 전송되는 데이터의 경계 존재
      5. 한번에 전송되는 데이터의 크기는 제한됨
10. **웹 브라우저에서 서버로 어떤 페이지를 요청하면 일어나는 일련의 과정 설명**

* 네트워크 전송시 데이터 표준을 정리한 것이 OSI(Open System Interconnection) 7계층이며, 이 이론을 실제 사용하는 인터넷 표준이 TCP/IP 4계층
* **OSI (Open System Interconnection) 7계층**
  + Physical, Link, Network, Transport: 데이터 전송
  + Session, Presentation, Application: 데이터 생성
* 1. Physical Layer: 5~7계층에서 만든 데이터를 전기신호로 바꿔서 매개체를 통해 전달
* 2. Data Link Layer: MAC address사용, 데이터의 전송 형태 결정
* 3. Network Layer: IP주소 사용, 데이터의 전송 경로 결정
* 4. Transport Layer: 데이터의 분할과 전송 결정, 서비스 구분 (TCP/UDP)
* 5. Session Layer: 사용자 간 포트(세션) 연결이 유효한지 확인하고 설정
* 6. Presentation Layer: 컴퓨터가 인식할 수 있는 형태로 데이터를 변환 (암호화, 압축, 인증)
* 7. Application Layer: 사용자에게 Network service 제공
* **TCP/IP 4계층:** Application Layer(User Data) 🡪 Transport Layer(Segment) 🡪 Internet Layer(Packet) 🡪 Network Access Layer (Frame)
* DNS 서버에 IP 주소를 요청 🡪 수신한 IP주소에 해당하는 웹 서버에 접속 (HTTP Request를 위해 TCP Socket을 개방하고 연결에 성공하면 HTTP Request가 TCP Socket을 통해 보내짐. 이에 대한 응답으로 웹 페이지의 정보가 사용자의 PC로 들어옴)

1. Application Layer에서 데이터를 생성하고, 소켓을 생성
2. Transport Layer에서 호스트 사이의 종점간 연결을 제공. 데이터 전송 조율을 담당. (TCP/UDP에 대한 구분 + 데이터에 대한 제어 정보 포함)
3. Internet Layer에서 논리적 주소인 IP를 이용해 라우팅을 비롯한 발신지로부터 목적지까지의 패킷 전달을 담당
   * + - 대표적 프로토콜: IP / 대표적 장비: 라우터
4. Network Access Layer 에서 물리적 주소인 MAC주소를 사용, IP 패킷의 물리적인 전달을 담당
5. **DB에서의 Commit와 Rollback 이란?**

* Commit: 하나의 논리적 단위(트랜잭션)에 대한 작업이 성공적으로 끝났을 때, 이 트랜잭션이 행한 갱신 연산이 완료된 것을 트랜잭션 관리자에게 알려주는 연산
* Rollback: 하나의 트랜잭션 처리가 비정상적으로 종료되어 DB의 일관성을 깨뜨렸을 때, 모든 연산을 취소(Undo)시키는 연산.

1. **RDBMS란?**

* Relational database management system (관계형 모델을 기반으로 한 데이터베이스 시스템)

1. **.해시충돌 해결방법**
   1. Chaining
      1. 링크드리스트만 사용하면 되므로 복잡한 계산식을 사용할 필요가 Open Addressing에 비해 적다
      2. 링크드리스트를 관리하는 overhead (삽입, 삭제)
      3. 해시테이블이 채워질수록 Lookup 성능저하가 Linear하게 발생한다
   2. Open Addressing
      1. 체이닝처럼 포인터가 필요없고, 지정된 메모리 외 추가적인 저장공간도 필요없다.
      2. 삽입, 삭제시 오버해드가 적다.
      3. 저장할 데이터가 적을 때 유리
2. **대칭키와 공개키**
   1. 대칭키
      1. 암호화와 복호화에 동일한 비밀키 사용
      2. 키는 비밀 통신망 또는 직접전달 등의 안전한 전송방식을 통해 사전에 전달되어야 함
      3. 암복호화 속도가 빠른 이점이 있으나 키 관리에 어려움이 있음
   2. 비대칭키 (공개키)
      1. Public key로 암호화한 정보는 그 쌍이 되는 Private key로 복호화 가능, vice versa
3. **메모리 계층**
   1. 레지스터
   2. 캐쉬
   3. 메모리
   4. 디스크
4. **메모리 할당 알고리즘**
   1. First Fit
   2. Next Fit
   3. Best Fit
5. **Page와 Segmentation**
   1. 둘 다 가상메모리 관리 기법
   2. Page: 같은 크기의 블락 🡪 Internal fragmentation
   3. Segmentation: 서로 다른 크기 🡪 External fragmentation
6. **Mutex와 Semaphore**
   1. 동기화 방법
   2. Mutex는 동기화 자원이 하나, Semaphore는 하나 이상
7. **Context switching이란?**

: 스케줄링에 의해 실행 중인 코드, 자원 등을 저장하고 현재 상태를 대기 상태로 만들고 다른 프로세스를 실행시키는 과정

1. **사용자 수준 Thread와 커널 수준 Thread의 차이**

: 코드가 실행되는 모드의 차이. 커널 수준의 쓰레드는 커널 모드. 사용자 모드에서 커널 쓰레드를 사용하면 context switching이 일어나서 overhead가 발생할 수 있음. 사용자 쓰레드가 여럿 있을 때 하나라도 커널 모드가 되면 다른 쓰레드가 중지됨.

1. **동기화란?**

: 멀티 프로세서에서 자원에 대해 독점적인 접근을 할 수 있도록 사용하는 방식

1. **오버로딩과 오버라이딩**
   1. 오버로딩: 이름이 같게 정의된 메소드에 대해서 다르게 동작하는 것. 같은 메소드일지라도 선언된 매개변수의 타입과 순서에 따라 적절한 메소드가 실행되는 다형성의 하나의 예
   2. 오버라이딩: 상속관계에 있는 두 클래스에서 같은 이름으로 정의된 메소드가 있다면 확장한 클래스의 메소드를 실행시킬 수 있는 것
2. **추상화**

:

1. **캡슐화와 은닉화**
   1. 캡슐화: 클래스의 기능을 하나로 묶어 보호하는 방법
2. **다형성**

: 부모 클래스에서 파생된 객체들의 기능이 서로 다를 수 있음. 오버라이딩과 레퍼런스 형변환, 상속등으로 구현

1. **3-way hands shaking**

: TCP/IP 프로토콜을 이용해서 통신을 하는 응용프로그램이 데이터를 전송하기 전에 먼저 정확한 전송을 보장하기 위해 상대방 컴퓨터와 사전에 세션을 수립하는 과정

* 양쪽 모두 데이터를 전송할 준비가 되었다는 것을 보장하고, 실제로 데이터 전달이 시작하기 전에 한쪽이 다른 쪽이 준비되었다는 것을 알 수 있게 한다
* 양쪽 모두 상대편에 대한 초기 순차일련번호를 얻을 수 있게 한다
* **[STEP 1]**
* A클라이언트는 B서버에 접속을 요청하는 SYN 패킷을 보낸다. 이때 A클라이언트는 SYN 을 보내고 SYN/ACK 응답을 기다리는SYN\_SENT 상태가 되는 것이다.
* **[STEP 2]**
* B서버는 SYN요청을 받고 A클라이언트에게 요청을 수락한다는 ACK 와 SYN flag 가 설정된 패킷을 발송하고 A가 다시 ACK으로 응답하기를 기다린다. 이때 B서버는 SYN\_RECEIVED 상태가 된다.
* **[STEP 3]**
* A클라이언트는 B서버에게 ACK을 보내고 이후로부터는 연결이 이루어지고 데이터가 오가게 되는것이다. 이때의 B서버 상태가 ESTABLISHED 이다.
* 위와 같은 방식으로 통신하는것이 신뢰성 있는 연결을 맺어 준다는 TCP의 3 Way handshake 방식이다.

1. **Transaction의 성질**
   1. 원자성 (Atomicity)

: All-or-Nothing. 트랜잭션의 모든 연산들이 정상적으로 수행완료되거나 어떠한 연산도 수행되지 않은 상태를 보장해야

* 1. 일관성 (Consistency)

: 사용되는 모든 데이터는 일관되어야 (성공적으로 수행된 트랜잭션은 정당한 데이터만을 DB에 반영해야) (Consistency: Referenetial constraint와 같은 무결성 제약 조건 + 비명시적 조건ㅡex.자금이체에서 두 계좌의 잔고 합은 tx이전,이후 동일해야)

* 1. 격리성 (Isolation)

: 접근하고 있는 데이터는 다른 트랜잭션으로부터 격리되어야 (독립적으로 수행

* + - * + 보장하는 방법: 트랜잭션을 순차적으로 수행
  1. 영속성 (Durability)

: 트랜잭션이 성공적으로 완료되어 commit되고 나면 해당 트랜잭션에 의한 모든 변경은 어떤 소프트웨어나 하드웨어 장애가 발생되더라도 보존되어야

1. **DB 정규화**

: 테이블 데이터 간의 종속성, 중복성 등으로 인한 예기치 못한 오류를 제거하는 과정

pros) DB의 일관성 증가. DB의 논리적 구조를 견고하게

cons) 테이블 수 증가, join 비용 증가

1. **무결성 제약조건**

: 데이터의 정확성과 일관성을 보장하기 위해 테이블 생성 시에 각 칼럼에 대해 정의하는 규칙 🡪 데이터 오류 발생 가능성 감소

1. **DBMS:** 데이터의 적절하고 효율적인 관리의 필요성으로 인해 등장한, 체계적으로 데이터를 관리하는 시스템
2. **PK:** 각 row를 유일하게 구분하는 칼럼키

**FK:** 하나의 테이블에 있는 칼럼으로는 그 의미를 표현할 수 없는 경우 다른 테이블의 PK를 참조하는 값

1. **RAID0: 분산 (한 드라이브에 장애 발생 시 손실)**

RAID1: 한 드라이브에 기록되는 모든 내용을 다른 드라이브에 복사, 안정성 증가, 저장용량 차지가 크다. (mirroring)

RAID5: 패러티 정보를 모든 드라이브에 나누어 기록 (패러티 담당 디스크의 병목 x)

1. **DB 시스템 장애시**

* Ckpt부터 따라가며 crash까지 commit, abort되지 않은 tx들을 undo list에, commit, abort된 tx들을 redo list에
* Crash 지점에 도달하면 log를 윗쪽으로 따라가며 undo list에 있는 transaction을 취소시키고, 다시 아래 방향으로 따라가며 redo list에 있는 transaction을 다시 수행