**121. 개발 환경 구축**

SW 및 HW 장비를 구축하는 것, 응용 소프트웨어가 운영될 환경과 유사한 구조로 구축

2. 하드웨어 환경

사용자와의 인터페이스 역할을 하는 클라이언트(Client), 통신하여 서비스를 제공하는 서버(Server)

웹 서버(Web Server) – Client로부터 직접 요청을 받아 처리하는 서버, 저용량의 정적 파일 제공

웹 애플리케이션 서버(WAS) : 동적 서비스 제공, 데이터 가공 작업 수행, 인터페이스 역할 수행

데이터베이스 서버(DB Server) : DB와 이를 관리하는 DBMS를 운영하는 서버

파일 서버(File Server) : 서비스 제공을 목적으로 유지하는 파일들을 저장하는 서버

3. 소프트웨어 환경 – 시스템SW : 운영체제, 웹서버, 서버 운영을 위한 서버 프로 그램, DBMS

개발 SW : 요구사항 관리/설계,모델링(UML)/구현(Eclipse)/빌드(Gradle)/테스트/형상 도구

**122. 서버 개발**

웹 app의 로직을 구현할 서버 프로그램을 제작하여 웹 app 서버(WAS)에 탑재하는 것을 의미

2. 서버 개발 프레임워크

네트워크 설정, 요청/응답 처리, 모델 구현 등을 쉽게 처리할 수 있도록 클래스/인터페이스 제공

Spring(JAVA), Node.js, Django(Python), Codeigniter(PHP), Ruby on Rails(Ruby)

3. 서버 프로그램 구현

응용 SW와 동일하게 모듈 및 공통 모듈을 개발한 후, 모듈들을 통합하는 방식으로 구현

**123. 보안 및 API**

보안 취약점을 최소화하여 안전한 SW를 개발하기 위한 일련의 보안 활동, 기밀성/무결성/가용성

2. 소프트웨어 개발 보안 점검 항목

세션 통제, 입력 데이터 검증/표현, 보안기능, 시간 및 상태, 에러처리, 코드 오류, 캡슐화, API오용

3. API – 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스, 원하는 기능의 쉽고 효율적인 구현, 특정한 작업 수행, 운영체제 기능 활용

**124. 배치 프로그램**

사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리

주기 : 정기/이벤트/On-Demand 배치, 필수 요소 : 대용량 데이터, 자동화, 견고/안정/신뢰성, 성능

2. 배치 스케줄러(Batch Scheduler) – 일괄 처리 작업이 설정된 주기에 맞춰 자동 수행 지원 도구

스프링 배치 : Spring의 특성, DB나 파일의 데이터를 교환하는데 필요한 컴포넌트 제공

구성요소, 역할 : Job(작업 정의), Job Launcher(실행 인터페이스), Step(제어 정보), Job Repository

Quart – 수행 작업과 시간을 관리하는 요소들을 분리하여 일괄 처리 작업에 유연성 제공

Scheduler(실행 환경 관리), Job, JobDetail(Job의 상세 정보), Trigger(Job의 실행 스케줄 정의)

**125. 패키지 소프트웨어**

기업에서 일반적으로 사용하는 여러 기능들을 통합하여 제공하는 SW

2. 패키지 소프트웨어의 특징

장점 : 안정적인 이용 가능, 전문 업체에 의해 품질 검증, 개발 조직의 필요성x비용절감, 이미 개발된 소프트웨어 사용으로 인해 프로젝트 기간 단축,

단점 : 기존 시스템과의 상호연동 및 연계가 어려움, 기능추가/코드 사용에 제약 발생, 결함 발생 시 판매처의 프로세스에 따라 보완되므로 능동적인 대처 기대↓

**128. 연산자**

3. 비트 연산자 - & : and, ^ : xor, | : or, ~ : not, << : 왼쪽 시프트, >> : 오른쪽 시프트

**136. 선언형 언어**

프로그램이 수행해야 하는 문제를 기술하는 언어, 목표 명시, 알고리즘 명시x

함수형 언어 : 수학적 함수를 조합하여 문제를 해결하는 언어, 재귀호출 자주 이용, 병렬처리, LISP

논리형 언어 : 논리 문장 사용 프로그램을 표현 계산, 선언/비절차적 언어, 반복/선택문x, PROLOG

2. 선언형 언어의 장∙단점

가독성,재사용성↑, 구체적 작동순서 작성x 오류↓, 프로그램 동작 변경x 관련 값 대체 가능

3. 선언형 프로그래밍 언어 종류 – HTML, LISP, PROLOG, XML, Haskell

**137. 라이브러리**

프로그램을 효율적으로 개발할 수 있게 자주 사용하는 함수나 데이터를 미리 만들어 놓은 집합체

2. C언어의 대표적인 표준 라이브러리

stdio.h : 데이터 입, 출력 가능 제공 ex) printf, scanf, fclose, fopen 등

math.h : 수학 함수 제공 ex) pow, abs, string.h : 문자열 처리 기능 ex) strlen, strcpy, strcmp

stdlib.h 자료 변환, 난수 할당, 메모리 할당 기능 ex) rand, free, malloc, atoi, time.h : 시간처리

3. JAVA의 대표적인 표준 라이브러리

Java.lang : 자바에 기본적으로 필요한 인터페이스, 자료형, 예외 처리 관련 기능 제공

Java.util : 날짜 처리, 난수 발생, 복잡한 문자열 처리 등에 관련된 기능 제공 ex) Date, Random

Java.io : 파일 입, 출력과 관련된 기능 및 프로토콜 제공 ex) input/Output stream, Reader, Writer,

Java.net : 네트워크 관련 기능 제공 ex) Socket, URL, InetAddress

Java.awt : UI 관련 기능 제공 ex) Frame, Panel, Dialog, Button, Checkbox

**138. 데이터 입 ∙ 출력**

x : 정수형 16진수, f : 소수점 실수, e : 지수형 실수, ld/o/x : long형 10/8/16진수, p : 주소 16진수

\n(new line), \b : 커서 왼쪽 한 칸 이동, \t : 일정 간격 띄움, \r 현재 줄 처음 이동 \0 널 출력

\’ 작은 따옴표 출력, \” 큰 따옴표 출력 \a 벨 소리 \\ 역 슬래시 출력 \f 한 페이지 넘김

**140. 프로토타입**

함수 원형(Function Prototype), 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것

2. C언어에서의 프로토타입 선언 - 메인 함수 바깥쪽, in func(int i, int j); 함수 선언

**141. 운영체제의 개념**

컴퓨터 시스템의 자원들을 효율적으로 관리, 사용자의 편리하고 효과적인 관리를 위해 환경 제공

2. 운영체제의 목적 – 처리능력, 반환 시간, 사용 가능도, 신뢰도

3. 운영체제의 기능

프로세서, 기억장치, 입출력장치, 파일 및 정보 등의 자원 관리, 스케줄링 기능, 인터페이스, 하드웨어와 네트워크 관리 및 제어, 데이터 관리, 데이터 및 자원 공유 기능 제공, 시스템 오류 복구

가상 계산기 : 한 대의 컴퓨터를 여러 대의 컴퓨터처럼 보이게 하는 가상 컴퓨터 운영체제

4. 운영체제의 주요 자원 관리 – 프로세스/기억장치/주변장치/파일 관리

5. 운영체제의 종류 – Windows, UNIX, LINUX, MacOS, MS-DOS

**142. Windows** – GUI, 선점형 멀티 태스킹(Preemptive Multi-Tasking), PnP, OLE

143. UNIX/LUNX/MacOS

1. UNIX.의 개요 및 특징 – 시분할/개방형 시스템, 대화식 운영 체제, 이식성, 호환성↑

2. UNIX 시스템의 구성

커널 : 하드웨어 보호, 프로세스/기억장치/파일/입출력 관리, 프로세스 통신, 데이터 전송 및 변환

쉘 : 명령어 해석, 파일 형태로 존재, 파이프라인, 입 출력 재지정으로 출 입력 방향 변경 가능

Utility Program : 외부 명령어, ex) 에디터, 컴파일러, 인터프리터, 디버거

**144. 기억장치 관리의 개요**

레지스터 → 캐시기억장치 → 주 기억장치 → 보조기억장치

3. 반입(Fetch) 전략 – 데이터를 주 기억장치에 적재하기 위한 전략 / 요구 반입, 예상 반입

4. 배치(Placement) 전략 – 신규 반입 데이터 적재 전략, 최초(First)/최적(Best)/최악(Worst) 적합

5. 교체(Replacement) 전략 – 사용되고 있는 영역 중에서 교체, FIFO, OPT, LRU, LFU, NUR, SCR

**145. 주기억장치 할당 기법**

프로그램, 데이터를 실행시키기 위한 주 기억장치 할당 / 데이터 연속 할당 기법, 분산 할당 기법

2. 단일 분할 할당 기법

주기억장치(사용자 영역/운영체제 영역)한 순간에 사용자 1명이 주 기억장치의 사용자 영역 사용

경계 레지스터 : 사용자 프로그램이 운영체제 영역에 접근하지 못하게 보호하는 레지스터

오버레이(Overlay) 기법 – 주기억장치보다 큰 사용자 프로그램 실행 가능

하나의 프로그램을 여러 개의 조각으로 분할한 후 필요한 조각을 차례로 주 기억장치에 적재 후 실행, 주 기억장치 공간 부족 시 불필요 조각 위치 장소에 새로운 조각 중첩(Overlay)하여 실행

스와핑(Swapping) 기법 – 필요에 따라 프로그램들을 서로 교체하는 기법, 페이징으로 발전

주기억장치 프로그램 보조기억장치 이동 (→ Swap Out), (←Swap In), 작업 완료까지 교체 수행

3. 다중 분할 할당 기법

고정 분할 할당(Multiple Contiguous Fixed partition allocation, MFT) 기법 = 정적 할당 기법

프로그램 할당 전 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할, 준비상태 큐에서 준비중인 프로그램을 각 영역에 할당하여 수행하는 기법, 낭비가 큼

가변 분할 할당(Multiple contiguous Variable partition allocation, MIT) 기법 = 동적 할당 기법

프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할, 효율적, 크기 제약 적음, 영역과 영역사이에 단편화가 발생될 수 있음

**146. 가상기억장치 구현 기법/페이지 교체 알고리즘**

보조기억장치의 일부를 주기억장치처럼 사용하는 것, 프로그램 실행 시 가상기억장치에 보관된 블록 중 요구되는 블록만 주기억장치에 불연속적으로 할당하여 처리

2. 페이징(Paging) 기법 – 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램을 동일하게 나눠진 주기억장치의 영역에 적재 후 실행

프로그램을 일정한 크기로 나눈 단위 – 페이지, 페이지 크기로 나누어진 주기억장치의 단위 – 페이지 프레임, 내부 단편화 발생, 주소 변환을 위해 위치 정보가 실린 페이지 맵 테이블 필요

3. 세그먼테이션(Segmentation) 기법

다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재해 실행시키는 기법, 외부 단편화

4. 페이지 교체 알고리즘 – 페이지 부재 발생 시 페이지 프레임을 선택하고 결정하는 기법

OPT(OPTimal replacement, 최적 교체) – 앞으로 오랫동안 사용하지 않을 페이지 교체

FIFO(First In First Out) – 가장 오래 있었던 페이지 교체 기법, 적재 시간 기억

LRU(Last Recently Used) – 최근 가장 오랫동안 사용하지 않은 페이지 교체 기법, Counter/Stack

LFU(Last Frequently Used) – 사용 빈도 가장 적은 페이지 교체 기법

NUR(Not Used Recently) – 최근에 사용하지 않은 페이지 교체, 참조 비트(호출1)/변형 비트(변경1)

SCR(Second Chance Replacement, 2차 기회 교체) – FIFO 보완, 빈번 사용 페이지 교체 방지

147. 가상기억장치 기타 관리 사항

1. 페이지 크기

작을 경우 : 단편화/이동 시간 감소, 효율↑, 테이블 크기↑, 매핑 속도↓, 전체 입∙출력시간 증가

클 경우 : 테이블 크기 감소, 매핑 속도↑, 단편화↑, 시간↑, 불필요 내용 주기억장치 적재 가능

2. Locality – 프로세스 실행 중 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질

시간 구역성(Temporal Locality) – 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상

기억 장소 : Loop, Stack, Sub Routine, Counting, Totaling에 사용되는 변수(기억장소)

공간 구역성(Spatial Locality) – 일정 위치의 페이지를 집중적으로 액세스하는 현상

어느 하나의 페이지를 참조하면 그 근처의 페이지를 계속 참조할 가능성이 높음을 의미

기억 장소 : Array Traversal, 순차적 코드의 실행, 변수 선언 할당 장소, 같은 영역에 있는 변수

3. 워킹 셋(Working Set) – 프로세스가 일정 시간동안 자주 참조하는 페이지들의 집합

4. 페이지 부재 빈도 방식

프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상, 페이지 부재 빈도(PFF) : 발생 횟수

페이지 부재율에 따라 페이지 프레임의 수를 늘리거나 줄여 부재율을 적정 수준으로 유지함

5. 프리페이징(Prepaging) – 필요할 것 같은 모든 페이지를 한꺼번에 페이지 프레임에 적재

6. 스래싱(Thrashing) – 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간↑ 현상

다중 프로그래밍의 정도 : 얼마나 많은 프로그램을 동시에 수행하느냐의 척도

다중 프로그래밍의 정도가 커질 시 스래싱 발생, 이후 CPU 이용률은 급격하게 감소함

방지 방법 : 다중 프로그래밍의 적정 수준 유지, 페이지 부재 빈도 조절 사용, 워킹 셋 유지, 부족한 자원 증설 및 일부 프로세스 중단, CPU 성능에 대한 자료 관리 및 임계치를 예상하여 운영

**148. 프로세스의 개요**

프로세서(CPU)에 의해 처리되는 사용자/시스템 프로그램을 의미, 작업(Job), 태스크(Task)

2. PCB(Process Control Block, 프로세스 제어 블록)

프로세스에 대한 중요한 정보를 저장해 놓는 곳, 각 프로세스마다 고유 CPB 생성, 완료 시 제거

저장 정보 : 상태, 포인터, 식별자, 우선순위, 스케줄링, 레지스터/입출력 상태/관리/계정 정보

3. 프로세스 상태 전이

프로세스가 시스템 내에 존재하는 동안 상태가 변하는 것을 의미, 상태 전이도로 표시

제출(Submit) : 작업을 처리하기 위해 사용자가 작업을 시스템에 제출한 상태

접수(Hold) : 제출된 작업이 스풀 공간인 디스크의 할당 위치에 저장된 상태

준비(Ready) : 프로세서를 할당받기 위해 기다리고 있는 상태, 큐에서 실행 준비, Job 스케줄러

실행(Run) : 프로세서를 할당받아 실행, 수행 완료 전 프로세서 할당 시간이 종료(Timer Run Out)되면 프로세스는 준비 상태로 전이, 입출력 처리 필요시 대기 상태, CPU 스케줄러에 의해 수행

대기(Wait), 보류, 블록(Block) : 입출력 처리 필요 시 실행중인 프로세스 중단, 완료될 때까지 대기

종료(Terminated, Exit) : 실행이 끝나고 프로세스 할당이 해제된 상태

4. 프로세스 상태 전이 관련 용어

Dispatch : 준비 상태에서 대기중인 프로세스 중 하나가 프로세서를 할당 받아 실행상태로 전이

Wake Up : 입,출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이 되는 과정

Spooling : 입, 출력장치의 공유 및 상대적으로 느린 입, 출력 장치의 처리 속도를 보완하고 다중 프로그래밍 시스템의 성능을 향상시키기 위해 입, 출력할 데이터를 직접 입, 출력장치에 보내지 않고 나중에 한꺼번에 입, 출력하기 위해 디스크에 저장하는 과정

교통량 제어기(Traffic Controller) : 프로세스의 상태에 대한 조사와 통보를 담당

5. 스레드(Thread) : 프로세스 내에서의 작업 단위, 시스템의 여러 자원을 할당받아 실행하는 단위

단일/다중 스레드, 경량 프로세스라고도 함, 스레드 기반 시스템 독립적인 스케줄링의 최소 단위

사용자/커널 수준의 스레드 : 속도 빠름, 구현 어려움/속도 느림, 구현 수월, 운영체제의 커널

장점 : 프로세스1→여러 개의 스레드 생성 = 병행성 증진, 처리율 향상, 응답 시간 단축, 실행 환경 공유 = 기억장소의 낭비 감소, 프로세스들 간 통신 향상, 공통 접근 기억장치로 효율적 통신

**149. 스케줄링**

프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업

장기 스케줄링 : 자원을 차지할 프로세스를 결정해 준비상태 큐로 보냄, 작업 스케줄러가 수행

중기 스케줄링 : CPU 할당을 결정, 프로세스가 많을 경우 일시 보류 후 활성화해 부하 조절

단기 스케줄링 : CPU를 할당받는 시기와 특정 프로세스를 지정, 프로세서 스케줄러에 의해 수행

2. 스케줄링의 목적 – 공정성/처리율(량) 증가, CPU 이용률 증가, 우선순위 제도, 오버헤드/응답 시간(Response Time)/반환 시간/대기 시간 최소화, 무한 연기 회피, 균형 있는 자원의 사용

3. 프로세스 스케줄링의 기법

비선점(Non-Preemptive) 스케줄링 : 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용x, 모든 프로세스에 대한 요구를 공정하게 처리, 응답 시간 예측 용이, 일괄 처리 방식에 적합

선점(Preemptive) 스케줄링 : 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용 가능, 빠른 응답 시간을 요구하는 대화식 시분할 시스템에 사용, 많은 오버헤드 초래, 타이머 클록 필요

**150. 환경 변수**

시스템 SW의 동작에 영향을 미치는 동적인 값들의 모임, 변수와 값들로 구성, 시스템 기본 정보

2. Windows의 주요 환경 변수

%ALLUSERPROFILE%, %APPDATA%, %ComSpec%, %HOMEDRIVE%, %HOMEPATH%, %LOGONSERVER%, %PATH%, %PATHNEXT%, %PROGRAMFILES%, %SYSTEMDRIVE%, %SYSTEMROOT%, %TEMP% 또는 %TMP%, %USERDOMAIN%, %USERNAME%, %USERPROFILE%

3. UNIX / LINUX의 주요 환경 변수

$DISPLAY, $HOME, $LANG, $MAIL, $PATH, $PS1, $PWD, $TERM, $USER

**151. 운영체제 기본 명령어**

운영체제를 제어하는 방법은 크게 CLI(명령어 직접 입력)와 GUI(마우스 조작)로 구분

2. Windows 기본 명령어

CLI 기본 명령어 : 명령 프롬프트(Commend)창에 명령어를 입력하여 작업을 수행하는 것

DIR, COPY, TYPE, REN, DEL, MD, CD, DLS, ATTRIB(파일 속성 변경), FIND, CHKDSK, FORMAT, MOVE

3. UNIX / LINUX 기본 명령어

CLI 기본 명령어 : 쉘(Shell)에 명령어를 입력하여 작업을 수행하는 것

Cat, chdir, chmod, chown, cp, exec, find, fork, fsck, getpid, getppid, ls, mount/unmoun, rm, wait

**152. 인터넷**

TCP/IP 프로토콜을 기반으로 하여 전 세계 컴퓨터와 네트워크들이 연결된 광범위 컴퓨터 통신망

2. IP 주소(Internet Protocol Address) – 인터넷에 연결된 컴퓨터 자원을 구분하기 위한 고유 주소

A~E Class : 국가or대형 통신망, 2^24/중대형, 2^16/소규모, 2^8/멀티캐스트용/실험적 주소, 사용x

3. 서브네팅(Subnetting) – 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용

4. IPv6(Internet Protocol version 6)의 개요 – 주소 부족 문제를 해결하기 위해 개발

128비트의 긴 주소 사용, 자료 전송 속도 빠름, 인증/기밀/데이터 무결성 지원으로 보안문제 해결, IPv4와 호환성↑, 확장/융통/연동성↑, 실시간 흐름 제어, Traffic Class/Flow Label로 품질 보장 용이

5. IPv6의 구성 - 16비트씩 8부분, 각 부분 16진수 표현, :으로 구분

유니/멀티/애니 캐스트 : 1:1통신/1:N통신/1:1통신(단일 수신자와 가장 가까이 있는 단일 수신자)

6. 도메인 네임(Domain Name) – 숫자로 된 IP 주소를 문자 형태로 표현한 것, DNS

**153. OSI 참조 모델**

통신 규약, 개방형 시스템간의 데이터 통신 시 장비 및 처리 방법 등을 7단계로 표준화하여 규정

2. OSI 참조 모델의 목적 – 서로 다른 시스템 간을 상호 접속하기 위한 개념 규정, 공통적 기반

3. OSI 참조 모델에서의 데이터 단위

프로토콜 데이터 단위(PDU, Protocol Data Unit) : 동일 계층 간에 교환되는 정보 단위

물리-비트, 데이터 링크-프레임, 네트워크-패킷, 전송-세그먼트, 세션/표현/응용-메시지

서비스 데이터 단위(SDU, Service Date) : 서비스 접근점(SAP)을 통해 다른 계층끼리 주고받는 정보의 단위, SAP : 상위 계층이 자신의 하위 계층으로부터 서비스를 제공받는 점(Point)

4. 물리 계층(Physical Layer) – 전송에 필요한 두 장치 간의 실제접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의, 관련 장비 : 리피터, 허브

5. 데이터 링크 계층(Data Link Layer) – 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 한다. HDLC, LAPB, LLC, MAC, LAPD, PPP

주요 기능 : 흐름 제어, 프레임 동기화, 오류 제어, 순서 제어, 관련 장비 : 랜카드, 브리지, 스위치

6. 네트워크 계층(Network Layer, 망 계층) – 네트워크 연결 관리, 데이터의 교환 및 중계 기능

네트워크 연결을 설정/유지/해제하는 기능, 경로 설정, 트래픽 제어, 패킷 정보 정송, 라우터

7. 전송 계층(Transport Layer) – 논리적 안정과 균일한 데이터 전송 서비스 제공

상 하위 계층의 인터페이스를 담당, 종단 시스템 간의 전송 연결 설정/데이터 전송/연결 해제 기능, 주소 설정/다중화(분할 및 재 조립)/오류 제어/흐름 제어 수행, TCP/UDP, 게이트 웨이

8. 세션 계층(Session Layer) – 송 수신 측 간의 관련성을 유지하고 대화 제어를 담당

송 수신측 간의 전송 정보의 일정 부분에 체크점을 두어 정보의 수신 상태에 체크

체크점 = 동기점(Synchronization Point), 오류가 있는 데이터의 회복을 위해 사용하는 것

소동기점 : 하나의 대화 단위 내에서 데이터의 전달을 제어하는 역할, 수신쪽에서 ACK 수신x

대동기점 : 처음과 끝에 사용, 전송하는 데이터 단위를 대화 단위로 구성, 확인신호(ACK)를 받음

9. 표현 계층(Presentation Layer)

응용/세션 계층 데이터 : 세션 계층에 보내기 전에 통신에 적당한 형태로/응용 계층에 맞게 변환

서로 다른 데이터 표현 형태를 갖는 시스템 간의 상호 접속을 위해 필요한 계층

코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식(포맷) 변환, 문맥 관리 기능

10. 응용 계층 – 사용자가 OSI 환경에 접근할 수 있도록 서비스를 제공

응용 프로세스 간의 정보 교환, 전자 사서함, 파일 전송, 가상 터미널 등의 서비스 제공

**154. 네트워크 관련 장비**

1. 네트워크 인터페이스 카드(NIC) – com-com or com-net 연결 장치, LAN카드, 네트워크 어댑터

2. 허브(Hub) – 가까운 거리의 컴퓨터들을 연결하는 장치, 신호 증폭 기능, 각 회선 통합적 관리

3. 리피터 – 전송 신호가 원래의 형태와 다르게 왜곡되거나 약해질 경우 재생하여 다시 전송

4. 브리지 – LAN-LAN or LAN 안의 컴퓨터 그룹(세그먼트)을 연결하는 기능을 수행, 병목 현상↓

5. 스위치 – LAN-LAN→Big LAN, 전송 속도↑, 수십-수백개의 포트 제공, 데이트 링크 계층 사용

6. 라우터 – 브리지에 데이터 전송 최적 경로 선택 기능 추가, LAN-WAN 연결 지원

7. 게이트웨이 – 프로토콜 구조가 다른 네트워크의 연결 수행, 세션-표현-응용 계층 연결, 출입구

8. 네트워크 장비 설치 시 고려 사항 – 설치 장비 최대한 활용, 확장 및 증설 고려 설계, HW와 SW 최신 버전 선정, 트래픽 분산 설계, 관리 및 유지 보수 용이 설계, 여유 포트 고려

**155. 프로토콜의 개념**

서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약

2. 프로토콜의 기본 요소

구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정

의미(Semantics) : 두 기기 간의 효율적, 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정

시간(Time) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

3. 프로토콜의 기능

단편화 : 송신 측에서 전송할 데이터를 전송에 알맞은 일정 크기의 작은 블록으로 자르는 작업

재결합 : 수신 측에서 단편화된 블록(PDU)을 원래의 데이터로 모으는 작업

캡슐화 : 단편화된 데이터에 송, 수신지 주소, 오류 검출 코드, 프로토콜 제어 정보 등을 부가

흐름 제어 : 수신 측의 처리 능력에 따라 송신 측에서 송신하는 데이터의 전송량, 전송 속도 조절

오류 제어 : 전송 중 발생 오류를 검출하고 정정해 데이터나 제어 정보의 파손에 대비하는 기능

동기화 : 송, 수신 측이 같은 상태를 유지하도록 타이밍(Timing)을 맞추는 기능

순서 제어 : 전송되는 PDU에 전송 순서를 부여하는 기능, 연결 위주의 데이터 전송 방식에 사용

주소 지정 : 데이터가 목적지까지 정확하게 전송될 수 있도록 목적지 이름, 주소, 경로 부여

다중화 : 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능

경로 제어 : 송, 수신 측 간의 송신 경로 중에서 최적의 패킷 교환 경로를 설정하는 기능

전송 서비스 : 전송 데이터가 사용하도록 하는 별도 부가 서비스, 우선 순위, 서비스 등급, 보안성

**156. TCP/IP**

인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜

TCP : 전송 계층, 연결형 서비스, 스트림 전송 기능, 포트.시퀀스 넘버, 체크섬

IP : 네트워크 계층, 비연결형 서비스, 헤더 길이 20-60, 버전, 패킷 길이, 헤더체크섬, 주소, 목적지

2. TCP/IP의 구조 – 응용(응용 프로그램 송, 수신) / 전송(호스트 간의 통신) / 인터넷(네트워크 / 주소 지정, 경로 설정) / 네트워크 액세스 계층(데이터 링크, 물리 계층 / 실제 데이터 송수신)

3. 응용 계층의 주요 프로토콜

FTP : com-com, com-internet 원격 파일 전송 프로토콜, SMTP : 전자 우편

TELNET : 외부 컴퓨터를 자신의 컴퓨터처럼 사용 가능, 가상 터미널 기능 수행

SNMP : 네트워크 관리 프로토콜, 기기의 net 정보를 네트워크 관리 시스템 에 보내는데 사용

DNS : 도메인 네임 → IP 주소 매핑, HTTP : WWW에서 HTML 문서 송수신 표준 프로토콜

4. 전송 계층의 주요 프로토콜

TCP : 양방향 연결, 가상 회선 연결, 스트림 위주 전달, 메시지 전송 감동, 투명성 보장 통신

UDP : 비연결형, 오버헤드↓, 실시간 전송 유리, 속도 중시, 정기적 전송, 동시 데이터 전달

RTCP : 패킷의 전송 품질 제어, 세션 참여자들에게 주기적 제어 정보 전송, 다중화 제공, 32비트

5. 인터넷 계층의 주요 프로토콜

IP : 전송 데이터 주소 지정, 경로 설정, 비연결형 데이터그램, 신뢰성 보장x

ICMP : IP와 조합, 제어 메시지 관리 역할, 헤더는 8바이트, IGMP : 멀티캐스트 그룹 유저 사용

ARP : Host IP 주소를 물리적 주소인 MAC 주소로 변경, RARP : 물리적 주소를 IP 주소로 변환

6. 네트워크 액세스 계층의 주요 프로토콜 – Ethernet, IEEE 802, HDLC, X.25, RS-232C