4과목 〈프로그래밍 언어 활용〉

1장 서버 프로그램 구현 (출제 비율 0.69%)

섹션127 개발 환경 구축 (C등급)

: 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것을 의미한다

- 개발 환경은 응용 소프트웨어가 운영될 환경과 유사한 구조로 구축
- 개발 프로젝트의 분석 단계의 산출물을 바탕으로 개발에 필요한 하드웨어와 소프트웨어를 선정한다
- 하드웨어와 소프트웨어의 성능, 편의성, 파이선스 등의 비즈니스 환경에 적합한 제품들을 최종적으로 결정하여 구축한다

하드웨어 환경 : 사용자와 인터페이스 역할을 하는 클라이언트 그리고 클라 | 언트와 통신하여 서비스를 제공하는 서버로 구성된다

- 클라이언트는 PC, 스마트폰 등이 있다.
- 서버는 사용 목적에 따라 웹 서버, 웹 어플리케이션 서버, 데이터베이스 서버, 파일 서버등을 나뉜다.

○ 웹 서버 : 클라이언트로부터 직접 요청을 받아 처리하는 서버로, 저용량의 정적 파일들을 제공한다

예) Apache HTTP Server, Microsoft Internet Information Service, Google Web Server 등등

○ 웹 어플리케이션 서버 : 사용자에게 동적 서비스를 제공하기 위해 웹 서버로부터 요청을 받아 데이터 가공 작업을 수행하거나, 웹 서버와 데이터베이스 서버 또는 웹 서버와 파일 서버 사이에서 인터페이스 역할을 수행하는 서비스이다.

예) Apache Tomcat, IBM WebSphere, Oracle WebLogic 등

○ 데이터베이스 서버 : 데이터베이스와 이를 관리하는 DBMS를 운영하는 서버

예) MySQL Server, Oracle Server, Microsoft SQL Server

○ 파일 서버 : 데이터베이스에 저장하기에는 비효율적이거나 서비스 제공을 목적을 유지하는 파일들을 저장하는 서버

예) AWS S3 등

※ 웹 서버의 기능

HTTP/S 지원	브라우저로부터 요청을 받아 응답할 때 사용하는 프로토콜
통신 기록	처리한 요청들을 로그 파일을 기록
정적 파일 관리 (Managing Static Files)	HTML, CSS, 이미지 등의 정적 파일들을 저장하고 관리하는 기능
대역폭 제한 (Bandwidth Throtting)	네트워크 트래픽의 포화를 방지하기 위해 응답속도를 제한
가상 호스팅	하나의 서버로 여러 개의 도메인 이름을 연결하는 기능
인증 (Authentication)	사용자가 합법적인 사용자인지를 확인하는 기능

소프트웨어 환경 : 클라이언트와 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성된다.

- 시스템 소프트웨어에는 운영체제(OS), 웹 서버 및 WAS 운용을 위한 서버 프로그램, DBMS 등이 있다.
- 개발 소프트웨어에는 요구사항 관리 도구, 설계/모델링 도구, 구현 도구_, 빌드 도구, 테스트 도구, 형상관리 도구 등이 있다

○ 요구사항 관리 도구 : 요구사항의 수집과 분석, 추적 등을 편리하게 도와주는 소프트웨어 -〉 JIRA, IBM DOORS, inteGREAT, Reqtify, Trello

○ 설계/모델링 도구 : UML(통합 모델링 언어)를 지원하며, 개발의 전 과정에서 설계 및 모델링을 도와주는 소프트웨어 -〉 DB Designer, PlantUML, ArgoUML

○ 구현 도구 : 개발 언어를 통해 애플리케이션의 실제 구현을 지원하는 소프트웨어 -〉 Eclipse, IntelliJ, Visual Studio, Netbeans, Node.js 등

○ 빌드 도구 : 구현 도구를 통해 작성된 소스의 빌드 및 배포, 라이브러리 관리를 지원하는 소프트웨어 -〉 Ant, Gradle, Maven, Jenkins 등

○ 테스트 도구 : 모듈들이 요구사항에 적합하게 구현되었는지 테스트하는 소프트웨어 -〉 CppUnit, JUnit, SpringTest 등

○ 형상 관리 도구 : 산출물들을 버전별로 관리하여 품질 향상을 지원하는 소프트웨어 -〉 GIT, CVS, Subversion, Mercurial 등

※개발 언어의 산정 기준 : 5가지 특성 고려

적정성	개발하려는 소프트웨어의 목적에 적합해야 한다
효율성	코드의 작성 및 구현이 효율적이어야 한다
이식성	다양한 시스템 및 환경에 적용이 가능해야함
친밀성	개발 언어에 대한 개발자들의 이해도와 활용도가 높아야 한다
범용성	다른 개발 사례가 존재하고 여러 분야에서 활용되고 있어야 함

섹션128 서버 개발 (C등급)

: 웹 애플리케이션의 로직을 구현할 서버 프로그램을 제작하여 웹 애플리케이션 서버에 탑재하는 것을 의미한다

- 웹 애플리케이션 서버에 구현된 서버 프로그램은 웹 서버로부터 받은 요청을 처리하여 결과를 반환하는 역할을 수행한다
- 서버 개발에 사용되는 프로그래밍 언어에는 Java, JavaScript, Python, PHP, Ruby 등이 있다
- 각 프로그래밍 언어에는 해당 언어로 서버 프로그램을 개발할 수 있도록 지원하는 프레임워크가 있다.

서버 개발 프레임워크 : 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스르 제공하는 소프트웨어를 말한다.

- 서버 개발 프레임워크에 따라 지원하는 프로그래밍 언어가 제한적이므로 선정할 수 있는 프레임워크도 제한적이다
- 서버 개발 프레임워크의 대부분은 모델-뷰-컨트롤러(MVC) 패턴을 기반으로 개발되었다.
- 대표적인 서버 개발 프레임웤의 종류는 다음과 같다

프레임워크	특징
Spring	JAVA 기반 프레임워크로, 전자정부 표준 프레임워크의 기반 기술로 사용되고 있다.
Node.js	JavaScript를 기반으로 만들어진 프레임워크로, 비동기 입·출력 처리와 이벤트 위주의 높은 처리 성능을 갖고 있어 실시간으로 입·출력이 빈번한 애플리케이션에 적합하다
Django	Python을 기반으로 만들어진 프레임워크로, 컴포넌트의 재사용과 플러그인화를 강조하여 신속한 개발이 가능하도록 지원한다
Codeigniter	PHP를 기반으로 만들어진 프레임워크로, 인터페이스가 간편할여 서버 자원을 적게 사용한다.
Ruby on Rails	Ruby를 기반으로 만들어진 프레임워크로, 테스트를 위한 웹 서버를 지원하며 데이터베이스 작업을 단순화, 자동화시켜 개발 코드의 길이가 짧아 신속한 개발이 가능하다

서버 프로그램 구현 : 응용 소프트웨어와 동일하게 모듈 및 공통 모듈을 개발한 후, 모듈들을 통합하는 방식으로 구현된다

- 모듈은 모듈화를 통해 분뤼된 시스템의 각 기능들로, 서브 루틴, 서브시스템, 소프퉤어 내의 프로그램, 작업 단위 등과 같은 의미로 사용된다
- 모듈 개발 시 기능적 독립성을 고려하여 다른 모듈과의 과도한 상화작용을 배제함으로써 특정 모듈의 수정이 다른 모듈들에게 영향을 미치지 않아야한다.
- 모듈의 독립성읜 결합도와 응집도에 의해 측정되며, 독립성을 높이려면 모듈의 결합도를 약하게 하고 응집도를 높여 모듈의 크기를 작게 만들어야함
- 공통 모듈은 여러 프로그램이나 재사용할 수 있는 모듈을 의미하며, 자주 사용되는 계산식이나 매번 필요한 사용자 인증 같은 기능들이 공통 모듈로 구성될 수 있다.

섹션129 보안 및 API

: 소프트웨어 개발 보안은 소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하여 보안 위협으로부터 안전한 소프트웨어를 개발하기 위한 일련의 보안 활동을 의미한다

- 소프트웨어 개발 보안은 데이터의 기밀성, 무결성, 가용성을 유지하는 것을 목표로 한다
- 정부에서 제공하는 소프트웨어 개발 보안 가이드를 참고하여 소프트웨어 개발 과정에서 점검해야 할 보안 항목들을 점검한다.

소프트웨어 개발 보안 점검 항목 : 소프트웨어 개발 각 단계에서 점검되어야할 보안 항목들을 말함

--〉 "세션 통제 / 입력 데이터 검증 및 표현 / 보안 기능 / 시간 및 상태 / 에러처리 / 코드 오류 / 캡슐화 / API 오용 "등의 단계에서 보안을 점검해야함

API(Application Programming Interface) : API는 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스를 의미한다.

- API는 프로그래밍 언어에서 특정한 작업을 수행하기 위해 사용되거나, 운영체제의 파일 제어, 화상 처리, 문자 제어 등의 기능을 활용하기 위해 사용된다.
- API는 개발에 필요한 여러 도구를 제공하기 때문에 이를 이용하면 원하는 기능을 쉽고 효울적으로 구현할 수 있다
- API의 종류에는 Windows API, 단일 유닉스 규격(SUS), Java API, 웹 API 등이 있으며 누구나 무료로 사용할 수 있게된 API를 Open API라 한다.

섹션130 배치 프로그램 (B)

: 배치 프로그램은 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것을 말한다

- 배치 프로그램이 자동으로 수행되는 주기에 따라 정기 배치, 이벤트성 배치, On Demand 배치로 구분된다.

정기 배치	일, 주, 월과 같이 정해진 기간에 정기적으로 수행
이벤트성 배치	특정 조건을 설정해두고 조건이 충족될 때만 수행
On-Demand 배치	사용자 요청 시 수행

- 배치 프로그램이 갖추어야할 **필수 요소**는 다음과 같음

대용량 데이터	대량의 데이터를 가져오거나, 전달하거나, 계산하는 등의 처리가 가능해야함
자동화	심각한 오류가 발생하는 상황을 제외하고는 사용자의 개입없이 수행되어야 함
견고성	잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야 함
안정성/신뢰성	오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있어야함
성능	다른 응용 프로그램의 수행을 방해하지 않아야 하고, 지정된 시간 내에 처리가 완료되어야 함

배치 스케쥴러(Batch Scheduler) : 일괄 처리(Batch Processing) 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원해주는 도구

- 배치 스케줄러는 특정 업무(Job)을 원하는 시간에 처리할 수 있도록 지원한다는 특성 때문에 잡 스케줄러 라고도 불린다
- 주로 사용되는 배치 스케줄러에는 스피링 배치, Quartz 등이 있다.

1. 스프링 배치 (Spring batch)

- Spring Source 사와 Accenture사가 2007년 공동 개발한 오픈 소스 프레임웤
- 스프링 프레임워크의 특성을 그대로 가져와 스프링이 가지고 있는 다양한 기능들을 모두 사용할 수 있다
- 데이터베이스나 파일의 데이터를 교환하는데 필요한 컴포넌트를 제공한다
- 로그 관리, 추적, 트랜잭션 관리, 작업 처리 통계, 작업 재시작 등의 다양한 기능들을 제공한다
- 스프링 배치의 주요 구성 요소와 역할

Job	수행할 작업의 정의
Job Launcher	실행을 위한 인터페이스
Step	Job 처리를 위한 제어 정보
Job Repository	Step의 제어 정보를 포함하여 작업 실행을 위한 모든 정보 저장

2. Quartz

- 스프링 프레임워크로 개발되는 응용 프로그램들의 일괄 처리를 위한 다양한 기능을 제공하는 오픈 소스 라이브러리이다
- 수행할 작업과 수행 시간을 관리하는 요소를 분리하여 일괄 처리 작업에 유연성을 제공
- Quartz 구성 요소와 역할

Scheduler	실행 환경 관리
Job	수행할 작업 정의실행을 위한 인터페이스
JobDetail	Job의 상세 정보
Trigger	Job의 실행 스케줄 정의

섹션 131 패키지 소프트웨어

: 기업에서 일반적으로 사용하는 여러 기능들을 통합하여 제공하는 소프트웨어

- 기업에서는 패키지 소프트웨어를 구입하여 기업 환경에 적합하게 커스터마이징하여 사용한다
- 패키지 소프트웨어를 이용하여 시스템을 구축하는 방식을 패키지 개발 방식이라고 한다
- 기능 요구사항을 70% 이상 충족시키는 패키지 소프트웨어가 있을 때만 사용하는 것이 적합하다

패키지 소프트웨어의 특징 : 요구사항을 분석하여 업무 특성에 맞게 전용을 ㅗ개발되는 소프트웨어와 비교하여 안정성, 라이선스, 생산성 등에서 차이가 있다

- 패키지 소프트웨어는 전문 업체에 의해 품질이 검증되었고, 국제/산업계 표준을 준수하고 있어 안정적인 이용이 가능하다
- 소프트웨어에 대한 라이센스가 판매자에게 있기 때문에 시스템 구축 후 기능 추가 및 코드 재사용 등에 제약이 발생한다
- 개발 조직을 갖추어야할 필요성이 없어 비용을 절감할 수 있고, 이미 개발된 소프트웨어를 사용하기 때문에 프로젝트 기간이 단축된다
- 기존에 봉하고 있던 시스템과의 상호 연동 및 연계가 어려울 수 있다
- 결함이 발생한 경우 판매처의 프로세스에 따라 보완되므로 이용자의 사정에 따라 능동적인 대처를 기대하기는 어렵다

※ 패키지 소프트웨어와 전용 개발 소프트웨어의 비교

	패키지 소프트웨어	전용 개발 소프트웨어
기능 요구사항	70% 이상 충족시키는 패키지 소프트웨어가 있는 경우 이용	모든 기능 요구사항 반영 가능
안정성	품질이 검증되었고, 업계 표준 활용	개발자의 역량에 따라 달라짐
라이선스	판매자	회사
생산성	개발을 위한 인력과 시간 절약	개발을 위한 인력과 시간 필요
호환성	보장이 안 됨	설계 단계부터 고려하여 개발
유지보수	결함 발생 시 즉시 대응이 어려움	결함 발생 시 즉시 대응 가능

2장 프로그래밍 언어 활용

섹션 132 데이터 타입

: 데이터타입은 변수에 저장될 데이터의 형식을 나타냄. 변수에 값을 저장하기 전에 문자형/정수형/실수형/ 등 어떤 형식의 값을 지정할지 데이터 타입을 지정하여 변수 선언

- 데이터 타입의 유형 정수 타입/부동 소수점 타입/문자 타입/문자열 타입/불린 타입/ 배열 타입 이 있음

C/C++의 데이터 타입 크기 및 범위

종류	데이터 타입	크기
문자	char	1 Byte
부호없는 문자형	unsigned char	1 Byte
정수	short	2 Byte
	int	4 Byte
	long	4 Byte
	long long	8 Byte
부호없는 정수형	unsigned short	2 Byte
	unsigned int	4 Byte
	unsigned long	4 Byte
실수	float	4 Byte
	double	8 Byte
	long double	8 Byte

Java의 데이터 타입 크기 및 범위

종류	데이터 타입	크기
문자	char	2 Byte
정수	byte	1 Byte
	short	2 Byte
	int	4 Byte
	long	8 Byte
실수	float	4 Byte
	double	8 Byte
논리	boolean	1 Byte

Python의 데이터 타입 크기 및 범위

종류	데이터 타입	크기
문자	str	무제한
정수	int	무제한
실수	float	8 Byte
	complex	16 Byte

※C에서의 구조체 : struct로 선언하며 다양한 타입의 변수들의 모임. 선언 후 마지막에 세미콜론을 붙여야함

※Python의 시퀀스 자료형 : 리스트, 튜플, range, 문자열처럼 값이 연속적으로 이어진 자료형

○ 리스트 list : 다양한 자료의 값을 연속적으로 저장하며, 필요에 따라 개수를 늘리거나 줄일 수 있음

○ 튜플 Tuple : 리스트처럼 연속적인 자료의 값을 저장하지만, 요소의 추가, 삭제, 변경은 불가능

○ range : 연속된 숫자 생성. 특히 반복문에서 많이 사용

섹션 133 변수

: 변수란 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장히기 위한 공간으로, 변할 수 있는 값을 의미

- 저장하는 값에 따라 정수형, 실수형, 문자형, 포인터형 등으로 구분됨

변수명 작성 규칙

- 영문자, 숫자, 언더바(_)를 사용할수 있음
- 첫 글자는 영문자나 언더바로 시작해야하며, 숫자가 올 수 없음
- 글자 수에 자한이 없음
- 공백이나 특수 문자 (*,+,-,/) 등 사용 불가
- 대·소문자 구분함
- 예약어를 변수명으로 사용 불가능
- 변수 선언 시 문장 끝에 반드시 세미콜론을 붙여야함
- 변수 선언 시 변수명에 데이터 타입을 명시하는 것을 헝가리안 표기법(Hungarian Notation)이라 한다

예약어 : 정해진 기능을 수행하도록 이미 용도가 정해져있는 단어

※C에서 헷갈릴 수 있는 예약어들

- enum, signed, typedof, union, unsigned, auto, extern, register, static, const, sizeof, volatile 등등

기억 클래스 : 변수 선언시 메모리 내에 변수의 값을 저장하기 위한 기억영역이 할당되는데, 할당되는 기억영역에 따라 사용 범위에 제한이 있음. 이러한 기억영역을 결정하는 작업들을 기억클래스라 한다.

- C언어에서는 다음과 같이 5가지 종류의 기억클래스를 제공한다.

종류	기억영역	예약어	생존기간	사용 범위
자동 변수	메모리(스택)	**auto**	일시적	지역적
레지스터 변수	레지스터	register		
정적 변수(내부)	메모리(데이터)	static	영구적	
정적 변수(외부)				전역적
외부 변수		extern		

1. 자동 변수 (Automatic Variable) : 함수나 코드의 범위를 한정하는 블록 내에서 선언되는 변수
   - 함수나 블록이 실행되는 동안애만 존재하며, 이를 벗어나면 소멸
   - 초기화하지 않으면 쓰레기값(Garbage Value)가 저장됨
2. 외부 변수 (External Variable) : 현재 파일이나 다른 파일에서 선언된 변수나 함수를 참조하기 위한 변수이다.
   - 외부 변수는 함수 밖에서 선언
   - 함수가 종료된 뒤에도 값이 소멸되지 않음
   - 초기화하지 않으면 자동으로 0으로 초기화 됨
   - 다른 파일에서 선언된 변수를 참조할 경우 초기화할 수 없다.
3. 정적 변수 (Static Variable) : 함수나 블록 내에서 선언하는 내부 정적 변수와 함수 외부에서 선언하는 외부 정적 변수가 있음
   - 내부 정적 변수는 선언한 함수나 블록 내에서만 사용할 수 있고, 외부 정적 변수는 모든 함수에서 사용 가능
   - 두 변수 모두 함수나 블록이 종료된 뒤에도 값이 소멸되지 않는다
   - 초기화는 변수 선언 시 한번만 할 수 있으며, 초기화를 생략하면 자동으로 0을 할당한다
4. 레지스터 변수 (Register Variable) : 메모리가 아닌 CPU 내부의 레지스터에 기억영역을 할당받는 변수들이다
   - 자주 사용되는 변수를 레지스터에 저장하여 처리 속도를 높이기 위해 사용한다
   - 함수나 블록이 실행되는 동안에만 존재하며 이를 벗어나면 자동으로 소멸된다
   - 레지스터의 사용 개수는 한정되어 있어 데이터를 저장할 레지스터가 없는 경우 자동 변수로 취급되어 메모리에 할당된다
   - CPU에 저장되어 메모리 주소 값을 가질 수 없기 때문에 변수의 주소를 구하는 주소 연산자(&)를 사용할 수 없다

변수의 선언 : 자료형 변수명 = 값;

섹션 134 연산자

1. 산술 연산자 : 가감승제 등 산술 계산에 사용되는 연산자. 일반 산순실과 달리 한 변수의 값을 증가시키거나 감소 시키는 증감연산자가 있음
--〉 (+ , - , * , / , % , ++ , --) 등

2. 관계 연산자 : 두 수의 관계를 비교하여 참 또는 거짓을 반환하는 연산자 거짓은 0, 참은 1로 사용되지만 0이 아닌 모든 숫자도 참으로 간주
--〉 (== , != , 〉 , 〉= , 〈 , 〈=) 등

3. **비트 연산자** : 비트별(0,1)로 연산하여 결과를 얻는 연산자

연산자	의미	비고
&	and	모든 비트가 1일 때만 1
^	xor	모든 비트가 같으면 0, 하나로도 다르면 1
|	or	모든 비트중 하나라도 1이면 1
~	not	각 비트의 부정
〈〈	왼쪽 시프트	비트를 왼쪽으로 이동
〉〉	오른쪽 시프트	비트를 오른쪽으로 이동

4. 논리 연산자 : 두 개의 값을 연산하여 참 또는 거짓을 결과를 얻음
--〉 (! , &&, ||)

5. 대입 연산자 : 연산후 결과를 대입하는 연산식을 간략히 입력하도록 제공. 산술·관계·논리·비트 연산자 모두에 적용 가능
--〉 (+= , -= , *= , /= , %= , 〈〈= , 〉〉=) 등

6. 조건 연산자 : 조건에 따라 서로 다른 수식을 수행
--〉 조건 ? 수식 1 : 수식 2 : 조건이 참이면 수식1, 거짓이면 수식2 수행

7. 기타 연산자

연산자	의미
sizeof	자료형의 크기를 반환
.(콤마)	○콤마로 구분하여 한줄에 두 개 이상의 수식을 작성하거나 변수를 정의 ○왼쪽에서 오른쪽 순서대로 수행되며, 순서연산자 라고도 함
(자료형)	○사용자가 자료형을 다른 자료형으로 변환할 때 사용 ○ 캐스트 연산자라고 부름 ○변환할 자료형을 괄호 안에 넣어서 변환할 값이나 변수명 앞에 놓음

연산자 우선순위

대분류	중분류	연산자	결합규칙	우선순위
단항 연산자	단항 연산자	!, ~, ++, --, size of	←	높음 ↕ 낮음
이항연산자	산술 연산자	*, /. %	→	
		+ -		
	시프트 연산자	〈〈, 〉〉		
	관계 연산자	〈, 〈=, 〉, 〉=		
		==, !=		
	비트 연산자	&, ^ , |		
삼항 연산자	조건 연산자	? :	→	
대입 연산자	대입 연산자	=, +=, =+, *=, /=,%=, 〈〈=, 〉〉=	←	
순서 연산자	순서 연산자	.	→	

섹션 135 데이터·입출력

C언어의 표준 입·출력 함수의 개요 : 대표적으로 scanf(), getchar(), gets(), printf(), putchar(), puts() 등이 있음

1. scanf() : 키보드로 입력받아 변수에 저장하는 함수
 - 형식 : scanf(서식 문자열, 변수의 주소)
 - 특징 : ○입력받을 데이터의 자료형, 자릿수 등을 지정할 수 있음
 ○한번에 여러개의 데이터를 입력 받을 수 있음
 ○서식 문자열과, 변수의 자료형은 일치해야 함

예) scanf("%d %f", &i,&j) : i와 j는 각각 정수형 10진수, 실수형 이어야함

서식 문자열

서식 문자열	의미
%d	정수형 10진수
%u	부호없는 정수형 10진수
%o	정수형 8진수
%x	정수형 16진수
%c	문자
%s	문자열

서식 문자열	의미
%f	소수점을 포함하는 실수
%e	지수형 실수
%ld	long형 10진수
%lo	long형 8진수
%lx	long 16진수
%p	주소를 16진수

2. printf() 함수 ★★★ : 인수로 주어진 값을 화면에 출력하는 함수
 - 형식 : printf(서식 문자열, 변수)

예) printf("%-8.2f", 200.2) --〉 200.2__ (_는 빈칸)

 - 주요 제어문자

문자	의미	기능
\n	new line	커서를 다음 줄 앞으로 이동
\b	backspace	커서를 왼쪽으로 한 칸 이동
\t	tab	커서를 일정 간격 띄움
\r	carriage return	커서를 현재 줄의 처음으로 이동
\0	null	널 문자를 출력
\'	single quote	작은 따옴표 출력
\"	double quote	큰 따옴표 출력
\a	alert	스피커로 벨 소리를 출력
\\	backslash	역 슬래시를 출력
\f	form feed	한 페이지를 넘김

※자바에서도 System.out.printf(서식 문자열, 변수)와 기능이 동일하다
※또한 여러개의 제어문자를 이용하여 출력 데이터도 여러개 지정하면서 출력할 수 있다.

기타 표준 입·출력 함수

입력	getchar()	키보드로 한 문자를 입력받아 변수에 저장
	gets()	키보드로 문자열을 입력받아 변수에 저장하는 함수로, 엔터를 누르기 전까지 하나의 문자열로 인식하여 저장
출력	putchar()	인수로 주어진 한 문자를 화면에 출력
	puts()	인수로 주어진 문자열을 화면에 출력한 후 커서를 자동으로 다음 줄 앞으로 이동하는 함수

섹션 136 제어문

: 컴퓨터 프로그램은 명령어가 서술된 순서에 따라 무조건 위에서 아래로 실행되는데, 조건을 지정하여 진행 순서를 변경할 수 있다. 이렇게 프로그램의 순서를 변경할 때 사용하는 명령문을 제어문이라고 함

1. 단순 IF문 : 조건이 참일 때 실행될 때 실행할 문장을 지정 혹은 else를 추가해 거짓일 때 실행할 문장도 지정할 수 있다
2. 다중 if문 : 조건이 여러 개 일 때 사용하며, 각 조건은 위에서 아래로 순서대로 판단한다. 또는 if문 안에 또 다른 if 문을 추가한다
3. switch 문 : 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리가능
 - case문의 레이블에는 한 개의 상수만 지정할 수 있으며, int, char, enum형의 상수만 가능하다
 - case문의 레이블에는 변수를 지정할 수 없음
 - break문은 생략이 가능하나, break문이 생략되면 조건이 해당되는 문장부터 switch가 끝날때까지 모든 문장을 수행됨.
4. goto 문 : 프로그램 실행 중 현재 위치에서 원하는 다른 문장으로 건너뛰어 수행을 계속하기 위해 사용하는 제어문
 - 원하는 문장을 쉽게 이동할 수 있으나 많이 사용하면 프로그램의 유지보수가 어려워져 거의 사용하지 않음

섹션 137 반복문

: 제어문의 한 종류로 일정한 횟수를 반복하는 명령문을 말함. 보통 변수하나를 두고 값을 일정하게 증가시키면서 정해진 횟수가 될 때까지 명령이나 명령 그룹을 반복

1. for문 : 초기값, 최종값, 증가값을 지정하여 정해진 횟수를 반복.
 - 참인 동안만 반복함
 - 초기값, 최종값, 증가값 중 하나 이상을 생략할 수 있고, 각각의 요서에 여러 개의 수식을 지정할 수도 있음
 - for(; ;)와 같이 모두 생략하면 이 for문은 무한 반복함
 - 문자도 for문을 수행할 수 있음 (문자의 아스키코드를 이용하는 방법)
2. while문 : 조건이 참일 경우 실행할 문장을 반복 수행하는 제어문
 - 조건이 거짓이 되면 while문 종료
3. do~while문 : while문과 거의 같지만 do안의 문장을 무조건 한번 수행한 후 wjile조건식으로 판단한 후 반복을 수행하게 됨. 즉 무조건 한번은 실행되는 반복문
4. break, continue
 - break : 바로 블록을 벗어남
 - continue : continue 다음의 문장을 수행하지 않고 제어를 반복문의 처음으로 이동함. 이는 반복문에서만 사용됨

섹션 138 배열과 문자열

: 배열은 동일한 데이터 유형을 여러 개 사용해야 할 경우 이를 손쉽게 처리하기 위해 여러 개의 변수들을 조합해서 하나의 이름으로 정의해 사용하는 것을 말한다
 - 배열은 하나의 이름으로 여러 기억장소를 가리키기 때문에 배열에서 개별적인 요소들의 위치는 첨자를 이용해서 지정한다.
 - 배열은 변수명 뒤에 대괄호를 붙이고, 그안에 사용할 개수를 지정한다
 - 배열은 행 우선으로 데이터가 기억장소에 할당된다
 - C언어에서 배열 위치를 나타내는 첨자 없이 배열 이름을 사용하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같다

1. 1차원 배열 : 변수들을 일직선상의 개념으로 조합한 배열
 - 자료형 변수명[개수];

※자바의 향상된 for 문은 내가아는 for each 문임

2. 2차원 배열 : 변수들을 평면, 즉 행과 열로 조합한 배열이다.
 - 자료형 변수명[행 개수][열 개수];

배열의 초기화 : 배열 선언 시 초기값을 지정할 수 있음
 - 배열 선언 시 배열의 크기를 생략하는 경우 반드시 초기값을 지정해야 초기값을 지정한 개수만큼 배열이 선언됨

배열 형태의 문자열 변수
- C언어에서는 큰따옴표("")로 묶인 글자는 글자 수에 관계 없이 문자열로 처리됨
- C언어에서는 문자열을 저장하는 자료형이 없기 때문에 배열, 또는 포인터를 이용

하여 처리함.
- 형식 : char 배열이름[크기] = "문자열";
- 배열에 문자열을 저장하면 문자열의 끝을 알리기 위한 널문자(\0)이 자동으로 삽입됨. 즉 문자열의 배열 크기는 항상 문자열의 크기보다 1 크게 선언해야함
- 배열에 문자열을 저장할 때는 배열 선언 시 초기값으로 지정해야 하며, 이미 선언된 배열에는 문자열을 저장할 수 없음

섹션 139 포인터

: 포인터는 변수의 주솔르 말하며, C언어에서는 주소를 제어할 수 있는 기능을 제공한다.

- C언어에서 변수의 주소를 지정할 때 사용하는 변수를 포인터 변수라 한다.
- 포인터 변수를 선언할 때는 자료의 형을 먼저 쓰고 변수명 앞에 간접 연산자 *을 붙인다 (예 int *a;)
- 포인터 변수에 주소를 저장하기 위해 변수의 주소를 알아낼 때는 변수 앞에 번지 연산자 &을 붙인다 (예 a = &b;)
- 실행문에서 포인터 변수에 간접 연산자 *을 붙이면 해당 포인터 변수가 기리키는 곳의 값을 말한다 (예 c = *a;)
- 포인터 변수는 필요에 의해 동적으로 할당되는 메모리 영역인 힙 영역에 접근하는 동적 변수이다
- 포인터 변수의 용도
  - ○ 연결된 자료 구조를 구성하기 위해 사용된다.
  - ○ 동적으로 할당된 자료 구조를 지정하기 위해 사용된다
  - ○ 배열을 인수로 전달하기 위해 사용된다
  - ○ 문자열을 표현하기 위해 사용된다
  - ○ 커다란 배열에서 요소를 효율적으로 사용하기 위해 사용된다
  - ○ 메모리에 직접 접근하기 위해 사용된다

※ 같은 주소를 쓰는 변수가 2개면 당연히 두 변수의 변화는 동일하다

포인터와 배열 : 배열을 포인터 변수에 저장한 후 포인터를 이용해 배열의 요소에 접근할 수 있다.

- 배열 위치를 나타내는 첨자를 생략하고 배열의 대표명만 지정하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같다
- 배열 요소에 대한 주소를 지정할 때는 일반 변수와 동일하게 & 연산자를 이용한다.

섹션 140 파이썬의 기초

파이썬의 기본 문법
- 변수의 자료형에 대한 선언이 없다
- 문장의 끝을 의미하는 세미콜론을 쓸 필요가 없다
- 변수에 연속하여 값을 저장하는 것이 가능하다
- if나 for과 같이 코드 블록을 포함하는 명령문을 작성할 때 코드 블록은 콜론(:)과 여백으로 구분한다
- 여백은 일반적으로 4칸 또는 한 개의 탭만큼 띄워야하며, 같은 수준의 코드들은 반드시 동일한 여백을 가져야 한다.

Python의 데이터 입·출력 함수
- input() 함수
  - ○ input()함수는 Python의 표준 입력함수로, 키보드로 입력받아 변수에 저장하는 함수이다
  - ○ 입력되는 값은 문자열로 취급되어 저장한다
  - ○ 형식 1 : 변수 = input(출력문자)
  - ○ 형식 2 : 변수1, 변수2, … = input(출력문자).split(분리문자)
- print() 함수
  - ○ 형식 1 : print(출력값1, 출력값2, …, sep = 분리문자, end = 종료문자)
  - ○ 형식 2 : print(서식 문자열* %(출력값1, 출력값2,…))

입력값의 형변환 : input() 함수는 항상 문자열로 저장하므로 형변환 필요
- 변수 = int(input()) 등
- 변수1, 변수2, … = map(float, input().split()) 등

리스트(List)
- C와 Java에서는 여러 요소들을 하나의 이름으로 처리할 때 배열을 사용했는데 python에서는 리스트를 사용한다
- 리스트는 필요에 따라 개수를 늘이ㅣ거나 줄일 수 있어 리스트를 선언할 때 크기

를 적지 않는다
- 배열과 달리 하나의 리스트에 정수, 실수, 문자열 등 다양한 자료형을 섞어서 저장할 수 있다
- 파이썬에서 리스트의 위치는 0부터 시작한다
- 형식 : ○리스트명 = [값1, 값2,…]
  ○리스트명 = list([값1, 값2, …])

딕셔너리(Dictionary)
- 딕셔너리는 연관된 값을 묶어서 저장하는 용도로 사용한다
- 리스트는 저장된 요소에 접근하기 위한 키로 위치에 해당하는 0,1,2 등의 숫자를 사용하지만 , 딕셔너리는 사용자가 원하는 값을 키로 지정해 사용한다
- 딕셔너리에 접근할 때는 딕셔너리 뒤에 대괄호를 사용하며, 대괄호 안에서 키를 지정한다
- 형식 : ○ 딕셔너리명 { 키1:값1 , 키2:값2,…}
  ○ 딕셔너리명 dict){ 키1:값1 , 키2:값2,…})
- 접근 방법 : 딕셔너리명['키'] --〉 키에 해당하는 값 반환

Range : 연속된 숫자 생성
- 형식 : ○ range(최종값)
  ○ range(초기값,최종값)
  ○ range(초기값, 최종값, 증가값)

슬라이스(Slice) : 문자열이나 리스트와 같은 순차형 객체에서 일부를 잘라 반환하는 기능이다.
- 형식 : ○객체명[초기위치:최종위치]
  ○객체명[초기위치:최종위치:증가값]
- 슬라이스는 일부 인수를 생략하여 사용할 수 있음

섹션 141 파이썬의 활용

1. if문 : 중괄호 대신 콜론으로 표현함
2. for문 : for i in range(n): 또는 for 변수 in 리스트:
   실행할 문장 실행할 문장
3. while 문 : 중괄호 대신 콜론 사용
4. 클래스 : 이것도 중괄호를 콜론으로 대체. 그런데 멤버함수 선언시 def를 이용
   - 객체의 선언 형식 : 변수명 = 클래스명()

※ 클래스 안에서 멤버함수 선언시 형식은 다음과 같다
```
def 메소드명(self, 인수):
        실행할 문장
        return 값
```
이때 self는 메소드에서 자기 클래스에 속한 변수에 접근할 때 사용하는 명칭임
자바에서 this 같은 개념으로, 파이썬에서는 첫 번째 인수로 들어가게 함.
물론 인수를 받지 않는 함수를 구현할때는 필요가 없음

섹션 142 절차적 프로그래밍 언어(C)

: 일련의 처리 절차를 정해진 문법에 따라 순서대로 기술해나가는 언어
- 절차적 프로그래밍 언어는 프로그램이 실행되는 절차를 중요시함
- 절차적 프로그래밍 언어는 데이터 중심으로 프로시저를 구현, 프로그램 전체가 유기적으로 연결되어 있음
- 절차적 프로그래밍 언어는 자연어에 가까운 단어와 문장으로 구성
- 절차적 프로그래밍 언어는 과학 계산이나 하드웨어 제어에 주로 사용

절차적 프로그래밍 언어의 장·단점
- 컴퓨터의 처리 구조와 유사하여 실행 속도가 빠름
- 같은 코드를 복사하지 않고 다른 위치에서 호출하여 사용가능
- 모듈 구성이 용이, 구조적인 프로그래밍 가능
- 프로그램을 분석하기 힘듬
- 유지보수나 코드의 수정이 힘듬

절차적 프로그래밍 언어의 종류
: C / ALGOL / COBOL / FORTRAN 등

섹션 143 객체지향 프로그래밍 언어(C)

: 객체지향 프로그래밍 언어는 현실 세계의 개체를 기계의 부품처럼 하나의 객체로

만들어, 기계적인 부품들을 조립하여 제품을 만들 듯이 소프트웨어를 개발할 때도 객체들을 조립해서 프로그램을 작성할 수 있도록 한 프로그래밍 기법이다
 - 프로시저보다는 명령과 데이터로 구성된 객체를 중심으로 하는 프로그래밍 기법으로, 한 프로그램을 다른 프로그램에서 이용할 수 있도록 함

객체지향 프로그래밍 언어의 장·단점
- 상속을 통한 재사용과 시스템의 확장이 용이
- 코드의 재활용성 높음
- 자연적인 모델링에 의해 분석과 설계를 쉽고 효율적으로 할 수 있음
- 사용자와 개발자 사이의 이해를 쉽게 해줌
- 대형 프로그램의 작성이 용이
- 소프트웨어 개발 및 유지 보수 용이
- 프로그래밍 구현을 지원해 주는 정형화된 분석 및 설계 방법이 없음
- 구현 시 처리 시간이 지연

객체지향 프로그래밍 언어의 종류
 : JAVA / C++ / Smalltalk 등
특히 Smalltalk는 1세대로 최초의 GUI를 제공한 언어

객체지향 프로그래밍 언어의 구성 요소 : 개체, 클래스, 메시지
※메시지 : 객체들간의 상호작용을 하는데 사용하는 수단으로 객체의 메소드(동작, 연산)를 일으키는 외부의 요구 사항

객체지향 프로그래밍 언어의 특징
 : 캡슐화 / 정보 은닉 / 상속성 / 다형성 / 추상화

섹션 144 스크립트 언어

 : 스크립트 언어는 HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용하는 것으로, 기계어로 컴파일 되지 않고 별도의 번역기가 소스를 분석하여 바로 동작하게 하는 언어이다.
 - 게시판 입력, 상품 검색, 회원 가입과 같은 데이터베이스 처리 작업을 수행하기 위해 주로 사용한다
- 스크립트 언어는 클라이언트의 웹 브라우저에서 해석되어 실행되는 클라이언트용 스크립트 언어와 서버에서 해석되어 실행된 후 결과만 클라이언트에게 보내는 서버용 스크립트 언어가 있다
 ○ 서버용 스크립트 언어 : ASP, JSP, PHP, 파이썬
 ○ 클라이언트용 스크립트 언어 : 자바 스크립트, VB 스크립트(Visual Basic Script)

스크립트 언어의 장·단점
 - 컴파일 없이 바로 실행하므로 결과를 바로 확인할 수 있음
 - 배우고 코딩하기 쉬움
 - 개발시간이 짧음
 - 소스 코드를 쉽고 빠르게 수정할 수 있음
 - 코드를 읽고 해석해야 하므로 실행 속도가 느림
 - 런타임 오류가 많이 발생함

스크립트 언어의 종류

자바 스크립트 (JAVA Script)	○ 웹 페이지의 동작을 제어하는 데 사용되는 클라이언트용 스크립트 언어이다. ○ 클래스 기반의 객체 상속을 지원하여 객체지향 프로그래밍 언어의 성격도 갖고 있다. ○ Prototype Link 와 Prototype Object를 통해 프로토타입 개념을 활용 가능
VB 스크립트 (Visual Basic Script)	○ 마이크로소프트 사에서 자바 스크립트에 대응하기 위해 제작한 언어로, Active X를 사용하여 마이크로소프트 사의 애플리케이션들을 컨트롤할 수 있음
ASP (Active Server Page)	○ 서버 측에서 동적으로 수행되는 페이지를 만들기 위한 언어로 마이크로 소프트 사에서 제작함 ○ Windows 계열에서만 수행 가능한 프로그래밍 언어
JSP (Java Server Page)	○ Java로 만들어진 서버용 스크립트로, 다양한 운영체제에서 사용이 가능
PHP (Professional Hypertext Preprocessor)	○서버용 스크립트 언어로 Linux, Unix, Windows 운영체제에서 사용 가능 ○C,Java와 문법이 유사해 배우기 쉬어 웹 제작에 많이 사용

파이썬	○ 귀도 반 로섬이 발표한 대화영 인터프리터 언어 ○ 객체지향 기능을 지원하고 플랫폼에 독립적이며 문법이 간단하여 배우기 쉬움
쉘 스크립트	○ 유닉스/리눅스 계열의 쉘(Shell)에서 사용되는 명령어들의 조합으로 구성된 스크립트 언어 ○ 컴파일 단계가 없어 실행 속도가 빠름 ○ 저장시 확장자로 .sh가 붙음 ○ 쉘의 종류 : Bash shell, Bourne Shell, C shell, Korn Shell ○ 쉘 스크립트에서 사용되는 제어문 : if, case / for, while, until
basic	절차지향 기능을 지원하는 대화형 인터프리터 언어로, 초보자도 쉽게 사용할 수 있는 문법 구조를 가짐

섹션 145 선언형 언어 (C)

: 선언형 언어는 명령형 언어와 반대되는 개념의 언어로, 명령형 언어가 문제를 해결하기 위한 방법을 기술한다면 선언형 언어는 프로그램이 수행해야 하는 문제를 기술하는 언어

- 선언형 언어는 목표를 명시하고 알고리즘은 명시하지 않음
- 선언형 언어에는 함수형 언어와 논리형 언어가 있음

함수형 언어	○ 수학적 함수를 조합하여 문제를 해결하는 언어로, 알려진 값을 함수에 적용하는 것을 기반으로 함 ○ 적용형 언어라고도 함 ○ 재귀호출이 자주 이용 ○ 병렬처리에 유리 ○ 종류 : LISP
논리형 언어	○ 기호 논리학에 기반을 둔 언어로,'논리 문장을 이용하여 프로그램을 표현하고 계산을 수행한다 ○ 선언적 언어라고도 한다 ○ 반복문이나 선택문을 이용하지 않음 ○ 비절차적 언어 ○ 종류 : PROLOG

※명령형 언어는 우리가 흔히 아는 절차적 언어와 객체지향 언어가 해당됨

선언형 언어의 장·단점

- 가독성이나 재사용성이 좋음
- 작동 순서를 구체적으로 작성하지 않기 때문에 오류가 적음
- 프로그램 동작을 변경하지 않고도 관련 값을 대체할 수 있음

선언형 프로그래밍 언어 종류

HTML	인터넷의 표준 문서인 하이퍼텍스트 문서를 만들기 위해 사용하는 언어로, 특별한 데이터 타입이 없는 단순한 텍스트이므로 호환성이 좋고 사용이 편리하다
LISP	인공지능 분야에 사용되는 언어이다 기본 자료 구조가 연결 리스트 구조이며, 재귀 호출을 많이 사용함
PROLOG	논ㄹ학을 기초로 한 고급 언어로, 인공 지능 분야에서 논리적인 추론이나 리스트 처리등에 주로 사용된다
XML	기존 HTML의 단점을 보완하여 웹에서 구조화된 폭넓고 다양한 문서들을 상호 교환할 수 있도록 설계된 언어 HTML에 사용자가 새로운 태그를 정의할 수 있으며, 문서의 내용과 이를 표현하는 방식이 독립적
Haskell	함수형 프로그래밍 언어로 부작용이 없음 코드가 간결하고 에러 발생 가능성이 낮음

섹션 146 라이브러리

: 라이브러리는 프로그램을 효율적으로 개발할 수 있도록 자주 사용하는 함수나 데이터들을 미리 만들어 모아 놓은 집합체

- 자주 사용하는 함수들의 반복적인 코드 작성을 피하기 위해 미리 만들어 놓은 것으로 , 필요할 때는 언제든지 호출하여 사용 가능
- 프로그래밍 언어에 따라 일반적으로 도움말, 설치 파일, 샘플 코드 등을 제공
- 라이브러리는 모듈과 패키지 모두를 의미함
  - ○ 모듈 : 하나의 기능이 한 개의 파일로 구현된 형태
  - ○ 패키지 : 하나의 패키지 폴더 안에 여러 개의 모듈을 모아둔 형태
- 라이브러리에는 표준 라이브러리와 외부 라이브러리가 있음
- 표준 라이브러리 : 프로그램이 언어에 기본적으로 포함되어 있는 라이브러리로, 여러 종류의 모듈이나 패키지 형태
- 외부 라이브러리 : 개발자들이 필요한기능들을 만들어 인터넷에 공유해 놓은 것으로, 외부 라이로러리를 다운받아 설치한 후 이용

C언어의 대표적인 표준 라이브러리 : C언어는 라이브러리를 헤더파일로 제공하는데, 각 헤더 파일에는 응용 프로그램 개발에 필요한 함수들이 정리되어 있음

- C언어에서 헤더 파일을 사용하려면 include문 선언 필요

헤더 파일	기능
stdio.h	○ 데이터 입출력 제공 ○ printf, scanmf, fprintf,fscanf, fclose,fopen등 제공
math.h	○ 수학함수 제공 ○ sqrt, pow, abs 등
string	○ 문자열 처리에 사용되는 기능들 제공 ○ strlen, strcpy, strcmp등
stdlib.h	○ 자료형 변환 , 난수발생, 메모리 할당에 사용되는 기능들을 제공 ○ atoi, atof ,srand ,rand ,malloc ,free 등
time.h	○ 시간처리에 사용되는 기능들 제공 ○ time, check 등

Java언어의 대표적인 표준 라이브러리 : 임포트문 사용해야함

패키지	기능
java.lang	○ 자바에 기본적으로 필요한 인터페이스, 자료형, 예외 처리 등에 관련된 기능 제공 ○ 임포트 문 없이 사용 가능 ○ String, System, Process, Runtime, Math, Error 등
java.util	○ 날짜 처리, 난수 발생, 복잡한 문자열 처리 등의 기능 제공 ○ Date, Calender, Random, String Tokenizer 등
java.io	○ 파일 입출력과 관련된 기능 제공 ○ InputStream, OutputStream, Reader, Writer 등
java.net	○ 네트워크 관련된 기능 제공 ○ Socket, Url, InetAddress 등
java.awt	○ 사용자 인터페이스와 관련된 기능 제공 ○ Frame, panel, Dialog, Button, Checkbox 등

섹션 147 예외 처리 (C)

: 프로그램의 정상적인 실행을 방해하는 조건이나 상태를 예외라고 하며, 이러한 예외가 발생을 때 프로그래머가 해당 문제에 대비해 작성해 놓은 처리 루틴을 수행하도록 하는 것을 예외 처리라고 함

- 예외가 발생했을 때 일반적인 처리 루틴은 프로그을 종료시키거나 로그를 남기도록 하는 것이다.
- C++, ada, JAVA, 자바 스크립트와 같은 언어에는 예외 처리 기능이 내장되어 있으며, 그 외의 언어에서는 필요한 경우 조건문을 이용하여 예외처리 루틴을 작성
- 예외의 원인에는 하드웨어 문제, 운영체제의 설정 실수, 라이브러리 손상, 사용자의 입력 실수, 받아들일 수 없느 연산, 할당하지 못하는 기억장치 접근 등 다양함

자바의 예외 처리 : try catch 문 적용.
try-〉 catch -〉finally 순으로 진행됨

자바의 주요 예외 객체는 알아서 판단할 것

섹션 148 프로토타입 (C)

: 프로그래밍 언어에서 프로토타입이란 함수 원형(Function Prototype)이라는 의미로, 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것이다.

- 함수가 호출되기 전에 함수가 미리 정의되는 경우에는 프로토타입을 정의하지 않아도 됨
- 프로토타입은 본문이 없다는 점을 제외하고 함수 정의와 형태가 동일
- 프로토타입에 정의된 반환 형식은 함수 정이에 지정된 반환 형식과 받느시 일치해야함

※즉 파일 초반부에 함수의 선언만 하고 밑에 함수에 구현을 하는 것이다.

3장 응용 SW 기초 기술 활용

섹션 149 운영체제의 개념

: OS는 컴퓨 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 관리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램들의 모임이다

- 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로, 다른 응용 프로그램이 유용한 적업을 할 수 있도록 환경을 제공해준다

운영 체제의 목적 : 처리 능력 향상, 사용 가능도 향상, 신뢰도 향상, 반환 시간 단 축 등이 있다.

처리능력 / 반환 시간 /사용 가능도/신뢰도 는 운영체제의 성능을 평가하는 기준

운영체제의 구성

1. 제어 프로그램 : 컴퓨터 전체의 작동 상태 감시, 작업의 순서 지정, 작업에 사용되는 데이터 관리 등의 역할을 수행하는 것으로 다음과 같이 구분할 수 있음

감시 프로그램 (Supervisor Prog.)	제어 프로그램 중 가장 핵심적인 역할을 하는 것으로, 자원의 할당 및 시스템 전체의 작동 상태를 감시하는 프로그램
작업 관리 프로그램 (Job Management Prog.)	작업이 정상적으로 처리될 수 있도록 작업의 순서와 방법을 관리하는 프로그램
데이터 관리 프로그램 (data Management Prog)	작업에 사용되는 데이터와 파일의 표준적인 처리 및 전송을 관리하는 프로그램

2. 처리 프로그램 : 제어 프로그램의 지시를 받아 사용자가 요구한 문제를 해결하기 위한 프로그램으로 다음과 같이 구분할 수 있음

언어 번역 프로그램	사용자가 고급언어로 작성한 원시 프로그램을 기계어 형태의 목적 프로그램으로 변환시키는 것으로, 컴파일러, 어셈블러, 인터프리터 등이 있음
서비스 프로그램	○사용자가 컴퓨터를 더욱 효율적으로 사용할 수 있도록 제작된 프로그램 ○분류/병합, 유틸리티 프로그램이 여기에 해당됨

운영체제의 기능

- 프로세서(처리기), 기억장치, 입·출력 장치, 파일 및 정보 등의 자원을 관리한다
- 자원을 효율적으로 관리하기 위해 자원의 스케줄링 기능을 제공한다
- 사용자와 시스템 간의 편리한 인터페이스 제공
- 시스템의 각종 하드웨어와 네트워크를 관리·제어함
- 데이터를 관리하고, 데이터 및 자원의 공유 기능을 제공
- 시스템의 오류를 검사하고 복구함
- 자원 보호 기능을 제공
- 입·출력에 대한 보조 기능을 제공
- 가상 계산기 기능을 제공 (Virtual Computer)

섹션 150 Windows 운영체제 (C)

: 윈도우즈는 1990년대 마이크로소프트 사가 개발한 운영체제

- 주요 특징에는 GUI, 선점형 멀티태스킹, OLE , PnP등이 있다.

1. 그래픽 사용자 인터페이스(GUI) : 초보자도 쉽게 사용할 수 있는 그래픽 사용자 인터페이스를 채용하였음

2. 선점형 멀티태스킹(Preemptive Mulit-tasking) : 선점형 멀티태스킹은 동시에 여러 개의 프로그램을 실행하는 멀티태스킹을 하면서 운영체제가 각 작업의 CPU 이용시간을 제어하여 으용 프로그램 실행중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환하는 방식이다.

- 하나의 응용 프로그램이 CPU를 독점하는 것을 방지할 수 있어 시스템 다운 현상 없이 더욱 안정적인 작업을 할 수 있다.

3. PnP(Plug and Play) : PnP는 컴퓨터 시스템에 프린터나 사운드 카드 등의 하드웨어를 설치하였을 때 , 해당 하드웨어를 사용하는 데 필요한 시스템 환경을 운영체제가 자동으로 구성해주는 기능

- 운영체제가 하듸웨어의 규격을 자동으로 인식하여 동작하게 해주므로 PC 주변 장치를 연결할 때 사용자가 직접 환경을 설정하지 않아도 된다.

- PnP 기능을 활용하기 위해서는 하드웨어와 소프트웨어 모두 PnP를 지원해야 한다

4. OLE(Object Linking and Embedding) : OLE는 다른 여러 응용 프로그램에서 작성된 문자나 그림 등의 개체를 현재 작성 중인 문서에 자유롭게 연결하거나 삽입하여 편집할 수 있게 하는 기능이다.

- OLE로 연결된 이미지를 원본 프로그램에서 수정하거나 편집하면 그 내용이 그대로 해당 문서에 반영된다.

5. 255자의 긴 파일명 : Windows에서는 파일 이름을 지정할 때 VFAT(Virtual File Allocation Table)를 이용하여 최대 255자 까지 지정할 수 있다.

- 파일 이름으로 ₩ , / , : , * , ? , " , 〈 , 〉 , |를 제외한 모든 문자 및 공백을 사용할 수 있으며, 한글의 경우 127자까지 지정할 수 있다

6. Single-User 시스템 : 컴퓨터 한 대를 한 사람만이 독점해서 사용한다.

섹션 151 UNIX / LINUX / MacOS

UNIX의 개요 및 특징 : UNIX는 1960년대 AT&A 벨 연구소, MIT, General Electric이 공동 개발한 운영체제 이다

- 시분할 시스템을 위해 설계된 대화식 운영체제로, 소스가 공개된 개방형 시스템이다.

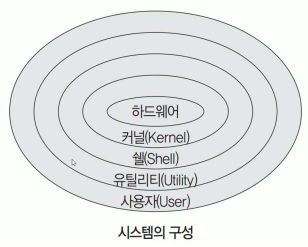
- 대부분 C언어로 작성되어 있어 이식성이 높으며 장치, 프로세스 간의 호환성이

높다
- 크기가 작고 이해하기 쉽다
- 다중 사용자, 다중 작업(멀티 태스킹)을 지원한다
- 많은 네트워킹 기능을 제공하므로 통신망(네트워크) 관리용 운영체제로 적합
- 트리 구조의 파일 시스템을 갖는다
- 전문적인 프로그램 개발에 용이하다
- 다양한 유틸리티 프로그램들이 존재한다

※다중 사용자는 여러 사용자가 동시에 시스템을 사용한다는 것이고, 다중 작업은 여러 개의 작업이나 프로그램을 동시에 사용하는 것을 말한다

UNIX 시스템의 구성

하드웨어와 사용자 사이의
커널, 쉘, 유틸리티로 이루어져 있음



시스템의 구성

1. 커널
- 유닉스의 가장 핵심적인 부분
- 컴퓨터가 부팅될 때 주기억장치에 적재된 후 상주하면서 실행된다
- 하드웨어를 보호하고, 프로그램과 하드웨어 간의 인터페이스 역할을 담당한다
- 프로세스(cpu 스케줄링) 관리, 기억장치 관리, 파일관리, 입·출력 관리 , 프로세스 간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행한다

2. 쉘(Shell)
-사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기
- 시스템과 사용자 간의 인터페이스르 담당한다
- DOS의 COMMAND.COM과 같은 기능을 수행한다
- 주기억장치에 상주하지 않고, 명령어가 포함된 파일 형태로 존재하여 보조 기억장치에서 교체 처리가 가능하다
- 파이프라인 기능을 지원하고 입·출력 재지정을 통해 출력과 입력의 방향을 변경할 수 있다
- 공용 Shell(Bourne Shell, C Shell, Korn Shell)이나 사용자 자신이 만든 Shell 을 사용할 수 있다

3. Utility Program
- 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용한다
- DOS에서의 외부 명령어에 해당된다
- 유틸리티 프로그램에는 에디터, 컴파일러, 인터프리터, 디버거 등이 있다

※ UNIX에서의 프로세스 간 통신 : 각 프로세스는 시스템 호출을 통해 커널의 기능을 사용하며, 프로세스 간 통신은 시그널, 파이프, 소켓등을 사용한다
1. 시그널 : 간단한 메시지를 이용하여 통신하는 것으로 초기 UNIX에서 사용됨
2. 파이프 : 프로세스의 출력이 다른 프로세스의 입력으로 사용되는 단방향 통신
3. 소켓 : 프로세스 사이의 대화를 가능하게 하는 쌍방향 통신 방식

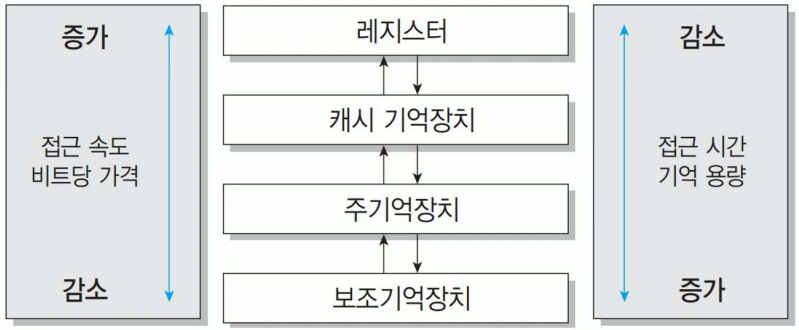
LINUX의 개요 및 특징 : LINUX는 1991년 리누스 토발즈가 UNIX 기반으로 개발한 체제이다.
- 프로그램 소스 코드가 무료로 공개되어 있기 때문에 프로그래머가 원하는 기능을 추가할 수 있고, 다양한 플랫폼에 설치하여 사용이 가능하며, 재배포가 가능하다
- UNIX와 완벽하게 호환된다
- 대부분의 특징이 UNIX와 동일하다

MacOS의 개요 및 특징 : MacOS는 1980년대 애플사가 UNIX를 기반으로 개발한 운영체제이다.
- 아이맥과 맥북 등 애플 사에서 생산하는 제품에서만 사용이 가능하다
- 드라이버 설치 및 install과 uninstall의 과정이 단순하다

※파일 디스크립터 (File Descriptor, 파일 서술자)
: 파일을 관리하기 위한 시스템(운영체제)이 필요로 하는 파일에 대한 정보를 가진 제어 블록을 의미하며, 파일 제어 블록(FCB : File Control Block)이라고 함
- 파일 디스크립터는 파일마다 독립적으로 존재하며, 시스템에 따라 다른 구조를 가질 수 있음
- 보통 파일 디스크립터는 보조기억장치 내에 저장되어 있다가 해당 파일이 Open될 때 주기억장치로 옮겨짐
- 파일 디스크립터는 파일 시스템이 관리하므로 사용자가 직접 참조할 수 없음

섹션 152 기억장치 관리의 개요

기억장치 계층 구조의 특징 : 레지스터, 캐시 기억장치, 주기억장치, 보조기억장치로 분류할 수 있다.



- 계층 구조에서 상위의 기억장치일수록 접근 속도와 접근 시간이 빠르지만, 기억 용량이 적고 고가이다
- 주기억장치는 각기 자신의 주르 갖는 워드 또는 바이트들로 구성되어 있으며 주소를 이용하여 액세스할 수 있다
- 레지스터, 캐시, 주기억장치와 프로그램 데이터는 CPU가 직접 액세스할 수 있으나 보조기억장치에 있는 프로그램이나 데이터는 직접 액세스할 수 없다
- 보조기억장치에 있는 데이터는 주기억장치에 적재된 후 CPU에 의해 액세스될 수 있다.

기억장치의 관리 전력의 개요 : 보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 적재 위치 등을 지정하여 한정된 주기억장치의 공간을 효율적으로 사용하기 위한 것으로, 반입(Fetch)전략, 배치(Placement) 전략, 교체(Replacement) 전략이 있다.

1. 반입(Fetch) 전략 : 보조기억장치에 보관중인 프로그램이나 데이터를 언제 주기억장치에 적재할 것인지 결정하는 전략이다
   - **요구 반입 (Demand Fetch) :** 실행중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법이다
   - **예상 반입 (Anticipatiory Fetch) :** 실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법이다

2. 배치 (Placement) 전략 : 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략이다
   - **최초 적합 (First Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법
   - **최적 적합 (Best Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 적게 남기는 분할 영역에 배치시키는 방법
   - **최악 적합 (Worst Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

3. 교체(Replacement) 전략 : 주기억장치의 모든 영역이 이미 사용중인 상태엣 ㅓ 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략
   - 교체 전략에는 FIFO, OPT, LRU, LFU, NUR, SCR 등이 있다.

섹션 153 주기억장치 할당 기법 (C)

: 프로그램이나 데이터를 실행시키기 위해 주기억장치에 어떻게 할당할 것인지에 대한 내용이며, 연속 할당 기법과 분산 할당 기법으로 분류할 수 있다

연속 할당 기법	프로그램을 주기억장치에 연속적으로 할당하는 기법으로, 단일 할당 기법과 다중 분할 할당 기법이 있다 - **단일 분할 할당 기법** : 오버레이, 스와핑 - **다중 분할 할당 기법** : 고정 분할 할당 기법, 동적 분할 할당 기법
분산 할당 기법	프로그램을 특정 단위의 조각으로 나누어 주기억장치 내에 분산하여 할당하는 기법으로, 페이징 기법과 세그먼테이션 기법으로 나눌 수 있음

(연속 할당 기법)

1. 단일 분할 할당 기법 : 주기억장치를 운영체제 영역과 사용자 영역으로 나누어 한 순간에는 오직 한 명의 사용자만이 주기억장치의 사용자 영역을 사용하는 기법이다.
   - 가장 단순한 기법으로 초기의 운영체제에서 많이 사용하던 기법
   - 운영체제를 보호하고, 프로그램이 사용자 영역만을 사용하기 위해 운영체제 영역과 사용자 영역을 구분하는 경계 레지스터가 사용됨 (Boundary Register)
   - 프로그램의 크기가 작을 경우 사용자 영역이 낭비될 수 있음
   - 초기에는 주기억장치보다 큰 사용자 프로그램은 사용될 수 없었으나 오버레이

기법을 사용하면서 이문제가 해결됨

○ 오버레이(Overlay) 기법 : 주기억장치보다 큰 사용자 프로그램을 실행하기 위한 기법

- 보조기억장치에 저장된 하나의 프로그램을 여러 조각으로 분할한 후 필요한 조각을 차례로 주기억장치에 적재하여 프로그램을 실행한다
- 프로그램이 실행되면서 주기억장치의 공간이 부족가면 주기억장치에 적재된 프로그램 조각 중 불필요한 조각이 위치한 장소에 새로운 프로그램의 조각을 중첩(Overlay)하여 적재한다
- 프로그램을 여러 개의 조각으로 분할하는 작업은 프로그래머가 수행해야 하므로 프로그래머는 시스템 구조나 프로그램 구조를 알아야한다.

○ 스와핑(Swapping) 기법 : 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다 필요에 따라 다른 프로그램과 교체하는 기법이다

- 주기억장치에 있는 프로그램이 보조기억장치로 이동되는 것을 Swap Out, 보조기억장치에 있는 프로그램이 주기억장치로 이동되는 것을 Swap In.
- 하나의 사용자 프로그램이 완료될 때까지 교체 과정을 여러 번 수행할 수 있음
- 가상기억장치의 페이징 기법으로 발전되었다.

2. 다중 분할 할당 기법 :

○ 고정분할 할당 (Multiple contiguous Fixed parTition allocation, MFT) 기법 = 정적 할당 (Static Allocation) 기법 : 고정 분할 할당은 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고 준비상태 큐에서 준비중인 프로그램을 각 영역에 할당하여 수행하는 기법

- 프로그램을 실행하려면 프로그램 전체가 주기억장치에 위치해야 한다
- 프로그램이 분할된 영역보다 커서 영역 안에 못들어갈 수 있는 경우가 발생할 수 있다
- 일정한 크기의 분할 영역에 다양한 크기의 프로그램이 할당되므로 내부단편화 및 외부 단편화가 발생하여 주기억장치의 낭비가 많다
- 실행할 프로그램의 크기를 미리 알고 있어야 한다
- 다중 프로그래밍을 위해 사용되었으나 현재는 사용되지 않는다

※ 절대 번역과 적재, 재배치 번역과 적재 p137

○ 가변분할 할당 (Multiple contiguous Variable parTition allocation, MVT) 기법 = 동적 할당 (Dynamic Allocation) 기법 : 고정 분할 할당 기법의 단편화를 줄이기 위한 것으로, 미리 주기억장치를 분할해 놓는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법이다

- 주기억장치를 효율적으로 사용할 수 있고, 다중 프로그래밍의 정도를 높임
- 고정 분할 할당 기법에 비해 실행될 프로세스 크기의 제약이 적음
- 단편화를 상당 부분 해결할 수 있으나 영역과 영역 사이에 단편화가 발생할 수 있음

섹션 154 가상 기억 장치 구현 기법 / 페이지 교체 알고리즘

가상 기억 장치의 개요 : 가상 기억 장치는 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법이다

- 프로그램을 여러 개의 작은 블록 단위로 나누어서 가상기억장치에 보관해 놓고, 프로그램 실행 시 요구되는 블록만 주기억장치에 불연속적으로 할당하여 처리
- 주기억장치의 용량보다 큰 포르갬을 실행하기 위해 사용한다
- 주기억장치이 이용률과 다중 프로그래밍의 효율을 높일 수 있다
- 가상기억장치에 저장된 프로그램을 실행하려면 가상기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환 작업이 필요하다
- 블록 단위로 나누어 사용하므로 연속 할당 방식에서 발생할 수 있는 단편화를 해결할 수 있다
- 가상기억장치의 일반적인 구현 방법에는 블록의 종류에 따라 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다

1. 페이징 기법 : 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램(페이지)을 동일하게 나눠진 주기억장치의 영역(페이지 프레임)에 적재시켜 실행하는 방법이다.

- 프로그램을 일정한 크기로 나눈 단위를 페이지라고 하고, 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임(Page Frame)이라고 한다
- 외부 단편화는 발생하지 않지만, 내부 단편화는 발생할 수 있다
- 주소 변환을 위해서 페이지의 위치 정보를 가지고 있는 페이지 맵 테이블이 필요하다.
- 페이지 맵 테이블 사용으로 비용이 증가하고, 처리 속도가 감소한다.

2. 세그먼테이션(Segmentation) 기법 : 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 방법
 - 프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖는다
 - 기억장치의 사용자 관점을 보존하는 기억장치 관리 기법이다.
 - 세그먼테이션 기법을 이용하는 궁극적인 이유는 기억공간을 절약하기 위해서이다.
 - 주소 변환을 위해서 세그먼트가 존재하는 위치 정보를 가지고 있는 세그먼트 맵 테이블이 필요하다.
 - 세그먼트가 주기억장치에 적재될 때 다른 세그먼트에게 할당된 영역을 침범할 수 없으며, 이를 위해 기억장치 보호키(Storage Protection Key)가 필요하다
 - 내부 단편화는 발생할 수 없으나 외부 단편화는 발생할수 있다.

세그먼테이션 기법의 일반적인 주소 변환
 - 가상주소는 세그먼트 번호를 나타내는 s와 세그먼트 내에 실제 내용이 위치하는 곳까지의 거리를 나타내는 변위값 d로 구성된다
 - 이를 세그먼트 테이블에 가서 세그먼트 번호로 데이터를 확인하는데, 세그먼트 테이블에 있는 레이블은 세그먼트 번호, 크기, 기준번지를 갖고 있다.
 - 그렇다면 세그먼트 번호 s를 통해 그 세그먼트의 기준번지 b를 얻고 b + d를 하여 원하는 데이터가 있는 실기억 주소를 얻을 수 있다.

페이지 교체 알고리즘 ★★★★
 : 페이지 부재(page fault)가 발생했을 때 가상기억장치의 필요한 페이지를 주기억장치에 적재해야 하는데, 이때 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프로임을 선택하여 교체할 것인지 결정하는 기법이다.
 - 페이지 교체 알고리즘에는 FIFO, OPT, LRU, LFU, NUR, SCR 등이 있다.

1. OPT(OPTimal repalcement, 최적 교체)
 - OPT는 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 방법이다
 - 벨레이디(Belady)가 제안한 것으로, 페이지 부재 회수가 가장 적게 발생하는 효율적인 알고리즘이다.

2. FIFO (First In First Out)
 - 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 방법
 - 이해하기 쉽고 프로그래밍 및 설계가 간단하다

3. LRU (Least Recently Used)
 - LRU는 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법이다
 - 각 페이지마다 계수기(Counter)나 스택(Stack)을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래 전에 사용된 페이지를 교체한다

4. LFU (Least Frequently Used)
 - LFU는 사용빈도가 가장 적은 페이지를 교체하는 기법이다
 - 활발하게 사용되는 페이지는 사용 횟수가 않아 교체되지 않고 사용된다.

5. NUR(Not Used Recently)
 - NUR은 LRU와 비슷한 알고리즘으로, 최근에 사용하지 않은 페이지를 교체하는 기법이다
 - 최근에 사용하지 않은 페이지는 향후에도 사용되지 않을 가능성이 높다는 것을 전제로, LRU에서 나타나는 시간적인 오버헤드를 줄일 수 있다.
 - 최근에 사용 여부를 확인하기 위해서 각 페이지마다 두 개의 비트, 즉 참조 비트(Reference Bit)와 변형 비트(Modified Bit, Dirty Bit)가 사용된다.
 - 참조가 앞 비트, 변형비트가 뒷 비트로 구성되어 2진법의 수가 작을수록 우선순위가 증가한다

6. SCR (Second Chance Replacement, 2차 기회 교체)
 - SCR은 아 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법이다

섹션 155 가상기억 장치 기타 관리 사항

1. 페이지 크기 : 페이징 기법을 사용하면 프로그램을 페이지 단위로 나누게 되는데, 페이지의 크기에 따라 시스템에 미치는 영향이 다르다.

○ **페이지 크기가 작은 경우 :**

- 페이지 단편화가 감소하고, 한 개의 페이지를 주기억장치에 이동하는 시간이 감소한다
- 불필요한 내용이 주기억장치에 적재될 확률이 적으므로 효율적인 워킹 셋을 유지할 수 있다.
- Locality에 더 일치할 수 있기 때문에 기억장치의 효율이 높아진다.
- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 커지고, 매핑 속도가 늦어진다.
- 디스크 접근 횟수가 많아져서 전체적인 입·출력 시간은 늘어난다

○ **페이지 크기가 큰 경우 :**

- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 작아지고, 매핑 속도가 빨라짐
- 디스크 접근 횟수가 줄어들어 전체적인 입·출력의 효율성이 증가한다
- 페이지 단편화가 증가하고, 한 개의 페이지를 주기억장치로 이동하는 시간이 늘어난다
- 프로세스(프로그램) 수행에 불필요한 내용까지도 주기억장치에 적재될 수 있다.

2. Locality(국부성, 지역성, 구역성, 국소성)은 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론이다.

- 스래싱을 방지하기 위한 워킹 셋 이론의 기반이 되었다
- 프로세스가 집중적으로 사용하는 페이지를 알아내는 방법 중 하나로, 가상기억장치 관리의 이론적인 근거가 된다
- 데닝(Denning) 교수에 의해 구역성의 개념이 증명되었으며, 캐시 메모리 시스템이 이론적 근거이다
- Locality의 종류에는 시간 구역성(Temporal Locality)과 공간 구역성(Spatial Locality)가 있다.

○시간 구역성(Temporal Locality)

- 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상
- 한 번 참조한 페이지는 가까운 시간 내에 계속 참조할 가능성이 높음을 의미한다
- **시간구역성이 이루어지는 기억장소** : Loop(반복, 순환), 스택, 부 프로그램, 카운팅, 집계(Totaling)에 사용되는 변수(기억장소)

○ 공간 구역성(Spatial Locality)

- 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상
- 어느 하나의 페이지를 참조하면 그 근처의 페이지를 계속 참조할 가능성이 높음을 의미한다
- **공간 구역성이 이루어지는 기억 장소** : 배열 순환, 배열 순례, 순차적 코드의 실행, 프로그래머들이 관련 변수(기억 장소)들을 서로 근처에 선언하여 할당되는 기억장소, 같은 영역에 있는 변수들을 참조할 때 사용

3. 워킹 셋(Working set) : 프로세스가 일정 시간 동안 자주 참조하는 페이즈들의 집합이다.

- 데닝이 제안한 프로그램의 움직임에 대한 모델로, 프로그램의 Locality 특징을 이용한다
- 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상이 줄어들어 프로세스의 기억장치 사용이 안정된다.
- 시간이 지남에 따라 자주 참조하는 페이지들의 집합이 변화하기 때문에 워킹 셋은 시간에 따라 변경된다.

4. 페이지 부재 빈도 방식 : 페이지 부재는 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 것을 위미하고, 페이지 부재 빈도(PFF: Page Fault Frequency)는 페이지 부재가 일어나는 횟수를 의미함

- 페이지 부재 빈도 방식은 페이지 부재율(Page Fault Rate)에 따라 주기억장치에 있는 페이지 프레임 수를 늘리거나 줄여 페이지 부재율을 적정 수준으로 유지하는 방식이다.
- 운영체제는 프로세스 실행 초기에 임의의 페이지 프레임을 할당하고, 페이지 부재율을 지속적으로 감시하고 있다가 부재율이 상한선을 넘어가면 좀 더많은 페이지 프레임을 할당하고, 부재율이 하한선을 넘어가면 페이지 프레임을 회수하는 방식을 사용한다.

5. 프리페이징(Prepaging) : 처음의 과도한 페이지 부재를 방지하기 위해 필요할 것 같은 모든 페이지를한꺼번에 페이지 프레임에 적재하는 기법이다

- 기억장치에 들어온 페이지들 중에서 사용되지 않는 페이지가 많을 수도 있다

6. 스래싱(Thrashing) : 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상이다

- 다중 프로그래밍 시스템이나 가상기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정중 자주 페이지 부재가 발생함으로써 나타나는 현상으로, 전체 시스템의 성능이 저하된다.
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU 이용률은 어느 특정 시점까지는 높아지지만, 다중 프로그래밍 정도가 너무 커지면 스레싱이 나타나고, CPU의 이용률은 급격히 감소하게 된다.

○ 스래싱 현상 방지 방법

- 다중 프로그래밍의 정도를 적정 수준으로 유지한다
- 페이지 부재 빈도를 조절하여 사용한다
- 워킹 셋을 유지한다
- 부족한 자원을 증설하고 , 일부 프로세스를 중단시킨다
- CPU 성능에 대한 자료의 지속적인 관리 및 분석으로 임계치를 예상하여 운영한다.

섹션 156 프로세스의 개요

: 프로세스는 일반적으로 프로세서(처리기,CPU)에 의해 처리되는 사용자 프로그램, 시스템 프로그램, 즉 실행중인 프로그램을 의미하며, 작업(JOB), 태스크(Task)라고도 한다.

※프로세스는 다음과 같이 여러 형태로 정의할 수 있다

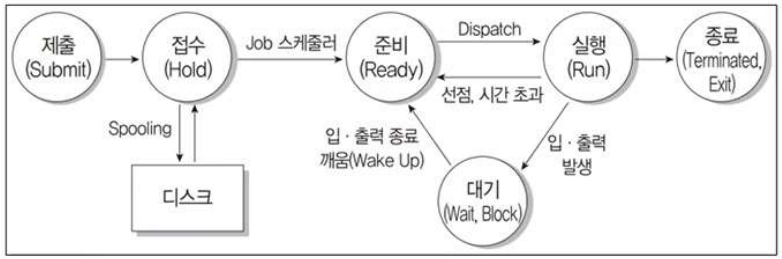
- PCB를 가진 프로그램
- 실기적장치에 저장된 프로그램
- 프로세서가 할당되는 실체로서, 디스패치가 가능한 단위
- 프로시저가 활동중인 것
- 비동기적 행위를 일으키는 주체
- 지정된 결과를 얻기 위한 일련의 계통적 동작
- 목적 또는 결과에 따라 발생되는 사건들의 과정
- 운영체제가 관리하는 실행 단위

PCB(Process Control Box, 프로세스 제어 블록) : PCB는 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓은 곳으로, Task Control Block 혹은 Job Control block이라고도 한다

- 각 프로세스가 생성될 때마다 고유의 PCB가 생성되고, 프로세스가 완료되면 PCB는 제거된다
- PCB에 저장되어 있는 정보

저장 정보	설명
프로세스의 현재 상태	준비, 대기, 실행 등의 프로세스 상태
포인터	○부모 프로세스에 대한 포인터 : 부모 프로세스의 주소 기억 ○자식 프로세스에 대한 포인터 : 자식 프로세스의 주소 기억 ○프로세스가 위치한 메모리에 대한 포인터 : 현재 프로세스가 위치한 주소 기억 ○할당딘 자원에 대한 포인터 : 프로세스에 할당된 각 자원에 대한 주소 기억
프로세스 고유 식별자	프로세스를 구분할 수 있는 고유의 번호
스케줄링 및 프로세스의 우선 순위	스케줄링 정보 및 프로세스가 실행될 우선순위
CPU 레지스터 정보	Accumulator, 인덱스 레지스터, 범용 레지스터, 프로그램 카운터에 대한 정보
주기억장치 관리 정보	기준 레지스터, 페이지 테이블에 대한 정보
입·출력 상태 정보	입·출력 장치, 개방된 파일 목록
계정 정보	CPU 사용 시간, 실제 사용 시간, 한정된 시간

프로세스 상태 전이 : 프로세스가 시스템 내에 존재하는 동안 프로세스의 상태가 변하는 것을 의미



- **제출** : 작업을 처리하기 위해 사용자가 작업을 시스템에 제출한 상태
- **접수 :** 제출된 작업이 스풀 공간인 디스크의 할당 위치에 저장된 상태
- **준비** : ○프로세스가 프로세서를 할당받기 위해 기다리는 상태
  ○프로세스는 준비상태 큐에서 실행을 준비하고 있음
  ○접수 상태에서 준비 상태로의 전의는 Job 스케줄러에 의해 수행

- **실행** : ○준비상태 큐에 있는 프로세스가 프로세스를 할당받아 실행되는 상태
  ○프로세스 수행이 완료되기 전에 프로세스에게 주어진 프로세서 할당 시간이 종료(Time Run Out)되면 프로세스는 준비 상태로 전이함
  ○실행중인 프로세스에 입출력 처리가 필요하면 실행중인 프로세스는 대기 상태로 전이함
  ○준비 상태에서 실행 상태로의 전이는 CPU스케줄러에 의해 수행
- **대기, 보류, 블록** : 프로세스에 인터럽트가 걸리면, 실행중이 프로세스가 중단되고 완료될 때 까지 대기함
- **종료** : 프로세스의 실행이 끝나고 프로세스 할당이 해제된 상태

프로세스 상태 전이 관련 용어
- Dispatch : 준비 상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정
- Wake Up : 입출력 작업이 완료되어 대기 상태에서 준비 상태로 전이되는 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·출력 장치에 보내지 않고 나중에 한꺼번에 입·출력하기 위해 디스크에 저장하는 어
- 교통량 제어기(Traffic Controller) : 프로세스의 상태에 대한 조사와 통보를 담당

스레드(Thread) : 프로세스 내의 작업 단위로서 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위
- 하나의 프로세스에 하나의 스레드가 존재하면 단일 스레드, 여러 개의 스레드가 존재하면 다중 스레드라고 한다
- 프로세스의 일부 특성을 갖고 있기 때문에 경량 프로세스라고도 한다
- 스레드 기반 시스템에서는 독립적인 스케줄링의 최소 단위로서 프로세스의 역할을 담당한다
- 동일 프로세스 환경에서 서로 독립적인 다중 수행이 가능

사용자 수준의 스레드	○사용자가 만든 라이브러리를 이용해 스레드를 운용 ○커널 모드로의 전환이 없어 오버헤드가 줄어듬 ○속도는 빠르지만 구현이 어려움
커널 수준의 스레드	○운영체제의 커널에 의해 스레드를 운용 ○한 프로세스가 운영체제를 호출할 때 전체 프로세스가 대기하지 않으므로 시스템의 성능을 높일 수 있음 ○여러 스레드가 커널에 동시에 접근할 수 있음 ○스레드의 독립적인 스케줄링이 가능 ○구현이 쉽지만 속도가 느림

스레드 사용의 장점
- 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진
- 하드웨어 운영체제의 성능과 응용 프로그램의 처리율 향상
- 응용프로그램의 응답시간 단축
- 실행환경을 공유시켜 기억장소의 낭비가 줄어듬
- 프로세스들 간의 통신이 향상
- 스레드는 공통적을 접근 가능한 기억장치를 통해 효율적으로 통신

섹션 157 스케줄링 (C)

: 스케줄링은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프롯스에게 할당하는 작업을 의미
- 프로세스가 생성되어 완료될 때까지 프로세스는 여러 종류의 스케줄링 과정을 거침

장기 스케줄링	○어떤 프로세스가 시스템의 자원을 차지할 수 있도록 할것인가를 결정하여 준비 상태 큐로 보내는 작업을 의미 ○ 작업 스케줄링(Job Sceduling), 상위 스케줄링이라고도 하며, 작업 스케줄러에 의해 수행
중기 스케줄링	○어떤 프로세스들이 CPU를 할당받을 것인지 결정하는 작업 ○CPU를 할당받으려는 프로세스가 많은 경우 프로세스를 일시 보류시킨 후 활성화 해서 일시적으로 부하 조절
단기 스케줄링	○프로세스가 실행되기 위해 CPU를 할당받는 시기와 특정 프로세스를 지정하는 작업 ○프로세서 스케줄링, 하위 스케줄링 이라고도 함 ○프로세서 스케줄링 및 문맥 교환은 프로세서 스케줄러에 의해 수행

프로세스의 스케줄링 기법

1. 비선점형 스케줄링 (Non-Preemptive)

- 이미 할당된 CPU를 다른 프로세스가 강제로 뺴앗아 사용할 수 없는 스케줄링 기법
- 프로세스가 CPU를 할당받으면 해당 프로세스가 완료될 때까지 CPU를 사용
- 모든 프로세스에 대한 요구를 공정하게 처리할 수_ 있음
- 프로세스 응답 시간의 예측이 용이하며, 일괄 처리 방식에 적합
- 중요한 작업(짧은 작업)이 중요하지 않은 작업(긴 작업)을 기다리는 경우가 발생할 수 있음
- 비선점형 스케줄링의 종류로는 **FCFS, SJF, 우선순위 , HRN, 기한부** 등의 알고리즘이 있음

2. 선점형 스케줄링

- 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 뺴앗아 사용할 수 있는 스케줄링 기법
- 우선순위가 높은 프로세스를 빠르게 처리핤 수 있음
- 주로 빠른 응답 시간을 요구하는 대화식 시분할 시스템에 사용
- 많은 오버헤드를 초래함
- 선점이 가능하도록 일정 시간 배당에 대한 인터럽트용 타이머 클록이 필요
- 선점형 스케줄링의 종류로는 **Round Robin, SRT, 선점 우선순위, 다단계 큐, 다단계 피드백 큐**등의 알고리즘이 있음

섹션 158 주요 스케줄링 알고리즘 ★★★★★★★

1. FCFS(First Come First Served, 선입선출)

- 준비상태 큐에 도착한 순서대로 CPU를 할당하는 기법임

※여기서 중요한 것은 평균 실행 시간과 평균 대기 시간, 평균 반환 시간을 계산하는 것이다. 평균 반환 시간 = 평균 실행 시간 + 평균 대기 시간

2. SJF(Shortest Job First,단기 작업 우선)

- 준비상태 큐에 있는 프로세스 중에서 실행시간이 가장 짧은 프로세스에게 먼저 CPU 할당
- 만약 제출 시간이 다른 경우에서, 이 기법은 비선점형 기법이므로 먼저 들어온 것이 끝날 때까지 방해를 받지 않는다.

3. HRN (Highest Responce-ratio Next)

- 실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로, 대기 시간과 서비스(실행) 시간을 이용하는 기법
- 우선순위 계산 공식을 이용하여 서비스(실행) 시간이 짧은 프로세스나 대기 시간이 긴 프로세스에게 우선순위를 주어 CPU를 할당한다
- 서비스 실행 시간이 짧거나 대기 시간이 긴 프로세스일 경우 우선순위가 높아짐
- 우선순위를 계산하여 그 숫가자 가장 높은 것부터 낮은 순으로 우선순위가 부여
- **우선순위 계산식 = (대기 시간 + 서비스 시간) / 서비스 시간**

섹션 159 환경 변수

: 환경 변수(Environment Variable)란 시스템 소프트웨어의 동작에 영향을 미치는 동적인 값들의 모임이다

- 환경 변수는 변수명과 같은 값으로 구성된다
- 환경 변수는 시스템의 기본 정보를 저장한다
- 환경 변수는 자식 프로세스에 상속된다
- 환경 변수는 시스템 전반에 걸쳐 적용되는 시스템 환경 변수와 사용자 계정 내에서만 적용되는 사용자 환경 변수로 구분된다

Windows의 주요 환경 변수 : 윈도우즈에서 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞뒤에 %를 붙여야함

- 윈도우즈에서 set을 입력하면 모든 환경 변수와 값을 출력함

환경 변수	용도
%ALLUSERPROFILE%	모든 사용자의 프로필이 저장된 폴더
%APPDATA%	설치된 프로그램의 필요 데이터가 저장된 폴더
%COMSPEC%	기본 명령 프롬프트로 사용할 프로그램명
%HOMEDRIVE%	로그인한 계정의 정보가 저장된 드라이브
%HOMEPATH%	로그인한 계정의 기본 폴더
%LOGONSERVER%	로그인한 계정이 접속한 서버명

%PATH%	실행 파일을 찾는 경로
%PATHEXT%	cmd에서 실행할 수 있는 파일의 확장자 목록
%PROGRAMFILES%	기본 프로그램의 설치 폴더
%SYSTEMDRIVE%	Windows가 부팅된 드라이브
%SYSTEMROOT%	부팅된 운영체제가 들어 있는 폴더
%TEMP% 또는 %TMP%	임시 파일이 저장되는 폴더
%USERDOMAIN%	로그인한 시스템의 도메인명
%USERNAME%	로그인한 계정 이름
%USERPROFILE%	로그인한 유저의 프로필이 저장된 폴더명

UNIX / LINUX의 주요 환경 변수

- UNIX / LINUX에서는 변수명 앞에서 $를 입력해야 한다
- UNIX / LINUX에서는 set, env, printenv, setenv 중 하나를 입력하면 모든 환경 변수와 값을 표시함

환경 변수	용도
$DISPLAY	현재 X 윈도 디스플레이 위치
$HOME	사용자의 홈 디렉터리
$LANG	프로그램 사용 시 기본적으로 지원되는 언어
$MAIL	메일을 보관하는 경로
$PATH	실행 파일을 찾는 경로
$PS1	쉘 프롬프트 정보
$PWD	현재 작업하는 디렉터리
$TERM	로긴 터미널 타입
$USER	사용자의 이름

섹션 160 운영체제 기본 명령어

: 운영체제를 제어하는 방법은 크게 CLI와 GUI로 구분할 수 있다.

- CLI는 키보드로 명령어를 직접 입력하여 작업을 수행하는 사용자 인터페이스를 의미한다
- GUI는 키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 작업을 수행하는 그래픽 사용자 인터페이스로 의미한다

Windows 기본 명령어

- CLI 기본 명령어

명령어	기능
DIR	파일 목록을 표시함.
COPY	파일을 복사함.
TYPE	파일의 내용을 표시함.
REN	파일의 이름을 변경함.
DEL	파일을 삭제함.
MD	디렉터리를 생성함.
CD	디렉터리의 위치를 변경함.
CLS	화면의 내용을 지움.
ATTRIB	파일의 속성을 변경함.
FIND	파일을 찾음.
CHKDSK	디스크 상태를 점검함.
FORMAT	디스크 표면을 트랙과 섹터로 나누어 초기화함.
MOVE	파일을 이동함.

-GUI는 아이콘으로 실행하는 작업

UNIX / LINUX 기본 명령어

- CLI 기본 명령어 : 쉘에 명령어를 입력하여 작업을 수행

명령어	기능
cat	파일 내용을 화면에 표시함.
chdir	현재 사용할 디렉터리의 위치를 변경함.
chmod	파일의 보호 모드를 설정하여 파일의 사용 허가를 지정함.
chown	소유자를 변경함.
cp	파일을 복사함.
exec	새로운 프로세스를 수행함.
find	파일을 찾음.
fork	새로운 프로세스를 생성함. (하위 프로세스 호출, 프로세스 복제 명령)
fsck	파일 시스템을 검사하고 보수함.
getpid	자신의 프로세스 아이디를 얻음.
getppid	부모 프로세스 아이디를 얻음.
ls	현재 디렉터리 내의 파일 목록을 확인함.
mount/unmount	파일 시스템을 마운팅한다/ 마운팅 해제함.
rm	파일을 삭제한다.
uname	시스템의 이름과 버전, 네트워크 호스트명 등의 시스템 정보를 표시함.
wait	fork 후 exec에 의해 실행되는 프로세스의 상위 프로세스가 하위 프로세스 종료 등의 event를 기다림.

※리눅스에서 중요한 명령어는 fork와 uname이다.

섹션 161 인터넷 ★★★★★

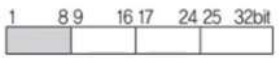
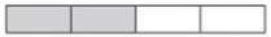
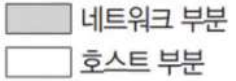
: 인터넷에서 TCP/IP 프로토콜을 기반으로 하여 전 세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 컴퓨터 통신망이다

- 인터넷은 미 국방성의 ARPANET에서 시작되었다.
- 인터넷은 유닉스 운영체제를 기반으로 한다
- 통신망과 컴퓨터가 있는 곳이라면 시간과 장소에 구애받지 않고 정보를 교환할 수 있다
- 인터넷에 연결된 모든 컴퓨터는 고유한 IP주소를 갖는다
- 컴퓨터 또는 네트워크를 서로 연결하기 위해서는 브리지, 라우터, 게이터웨이가 사용된다.
- 다른 네트워크 또는 같은 네트워크를 연결하여 중추적 역할을 하는 네트워크로, 보통 인터넷의 주가 되는 기간망을 일컫는 용어를 백본(Backbone)이라고 한다

IP 주소 (Internet Protocol Addreess)

: 인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소

- 숫자로 8비트씩 4부분, 총 32비트로 구성되어 있다
- IP주소는 네트워크 부분의 길이에 따라 다음과 같이 A~E 클래스로 구성되어있다.

A Class	• 국가나 대형 통신망에 사용(0~127로 시작) • 2^{24} = 16,777,216개의 호스트 사용 가능	
B Class	• 중대형 통신망에 사용(128~191로 시작) • 2^{16} = 65,536개의 호스트 사용 가능	
C Class	• 소규모 통신망에 사용(192~223으로 시작) • 2^{8} = 256개의 호스트 사용 가능	
D Class	멀티캐스트용으로 사용(224~239로 시작)	
E Class	실험적 주소이며 공용되지 않음	

서브네팅(Subnetting) : 서브네팅은 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용하는 것을 말한다

- 4바이트의 IP주소 중 네트워크 주소와 호스트 주소를 구분하기 위한 비트를 서브넷 마스크(Subnet Mask)라고 하며, 이를 변경하여 네트워크 주소를 여러 개로 분할하여 사용한다
- 서브넷 마스크는 각 클래스마다 다르게 사용된다.

※서브네팅의 예 p.166 필시 확인!!!

IPv6의 개요 : 현재 사용하고 있는 IP 주소 체계인 IPv4의 주소 문제를 해결하기 위해 개발되었다

- 128비트의 긴 주소를 사용하여 주소 부족 문제를 해결할 수 있으며, IPv4에 비해 자료 전송 속도가 빠르다
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제를 해결할 수 있다.
- IPv4와 호환성이 뛰어나다
- 주소의 확장성, 융통성, 연동성이 뛰어나며, 실시간 흐름 제어로 향상된 멀티미디어 기능을 지원한다
- Traffic Class, Flow Label을 이용하여 등급별, 서비스별로 패킷을 구분할 수 있어 품질 보장이 용이하다
- 패킷 크기를 확장할 수 있으므로 패킷 크기에 제한이 ㅇ벗음
- 기본 헤더 뒤에 확장 헤더를 더함으로써 더욱 다양한 정보의 저장이 가능해져 네트워크 기능 확장이 용이하다
- 미리 예약된 알고리즘을 통해 고유성이 보장된 주소를 자동으로 구성할 수 있다. 즉 자동을 네트워크 환경 구성이 가능하다

IPv6의 구성

- 16비트 씩 8부분, 총 128비트로 구성되어 있다
- 각 부분을 16진수로 표현하고, 콜론으로 구분한다
- IPv6은 다음과 같이 세가지 주소 체계로 나누어진다.

유니캐스트(Unicast)	단일 송신자와 단일 수신자간의 통신 (1 대 1 통신)
멀티캐스트(Multicast)	단일 통신자와 다중 수신자간의 통신 (1대 다 통신)
애니캐스트(Anycaset)	단일 송신자와 가장 가까이 있는 단일 수신자간의 통신

도메인 네임(Domain name) : 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 푫현한 것
 - 호스트 컴퓨터 이름, 소속 기관 이름, 소속 기관의 종류, 소속 국가명 순으로 구성되며, 왼쪽에서 오른쪽으로 갈수록 상위 도메인을 의미한다
 - 문자로 된 도메인을 아이피 주소로 변환은 DNS 서버가 해준다

섹션 162 OSI 참조 모델 ★★★★★

: 다른 시스템 간의 원할한 통신을 위해 ISO에서 제안한 통신 규약
- 개방형 시스템 간의 데이터 통신 시 필요한 장비 및 처리 방법 등을 7단계로 표준화하여 규정하였음
- OSI 7계층은 1~3계층을 하위계층, 4~7계층을 상위계층이라고 한다

OSI 참조 모델의 목적
 - 서로 다른 시스템 간을 상호 접속하기 위한 개념 규정
 - OSI 규격을 개발하기 위한 범위를 정함
 - 관련 규정의 적합성을 조절하기 위한 공통적 기반을 제공

OSI 참조 모델에서의 데이터 단위
- 프로토콜 데이터 단위(PDU; Protocol Data Unit)
- 서비스 데이터 단위(SDU; Service Data Unit)

1. **물리 계층(Physical layer)** : 전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의
 - 물리적 전송 매체와 전송 신호 방식을 정의하며,RS-232C,X.21 등의 표준있음
 - **관련 장비 : 리피터, 허브**

2. 데이터 링크 계층 (DataLink Layer) : 두 개의 인접한 개방 시스템들 간의 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 시스템 간 연결 설정과 유지 및 종료를 담당
 - 송신 측과 수신 측의 속도 차이를 해결하기 위한 흐름 제어 기능 제공
 - 프레임의 시작과 끝을 구분하기 위한 프레임의 동기화 기능을 함
 - 오류의 검출과 회복을 위한 오류 제어 기능 제공
 - 프레임의 순서적 전송을 위한 순서 제어 기능 제공
 - HDLC, LAPB, LLC, MAC, LAPD, PPP 등의 표준이 있음
 - **관련 장비 : 랜카드, 브리지, 스위치**

3. 네트워크 계층 (망 계층) : 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터 교환 및 중계 기능
 - 네트워크 연결을 설정, 유지, 해제하는 기느을 함
 - 발신자와 목적지의 논리 주소가 추가된 패킷을 최종 목적지까지 전달하는 책임을 진다
 - 경로 설정(Routing), 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송을 함
 - X.25, IP 등의 표준이 있음
 - **관련 장비 : 라우터**

4. 전송 계층 (Transport Layer) : 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 함
 - OSI 7계층 중 하위 3계층과 상위 3계층의 인터페이스를 담당한다
 - 종단 시스템 간의 전송 연결 설정, 데이터 전송, 연결 해제 기능을 한다
 - 주소 설정, 다중화(분할 및 재조립), 오류 제어, 흐름제어등을 수행한다
 - TCP, UDP 등 표준이 있음
 - **관련 장비 : 게이트웨이**

5. 세션 계층 : 송·수신 측 간의 관련성을 유지하고 대화 제어를 담당
 - 대화(회화) 구성 및 동기 제어, 데이터 교환 관리 기능을 함
 - 송·수신 측 간의 데이터 전송, 연결 해제, 동기 처리 등의 대화를 관리하기 위해 토큰이 사용됨
 - 송·수신 측 간의 대화 동기를 위해 전송하는 정보의 일정한 부분에 체크점을 두어 정보의 수신 상태를 체크하며, 이떄의 체크점을 동기점이라고 한다
 - 동기점은 오류가 있는 데이터의 회복을 위해 사용하는 것으로, 종류에는 소동기점과 대동기점이 있음

6. 표현 계층 : 응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능을 함
 - 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 기능을 한다

7. 응용 계층 (Application) : 응용 계층은 사용자(응용 프로그램)가 OSI 환경에 접

근할 수 있도록 서비스를 제공한다
 - 응용 프로그램 간의 정보 교환, 전자 사서함, 파일 전송, 가상 터미널 등의 서비스를 제공

섹션 162 네트워크 관련 장비

1. 네트워크 인터페이스 카드 (NIC) : 네트워크 인터페이스 카드는 컴퓨터와 컴퓨터 또는 컴퓨터와 넽트워크를 연결하는 장치로, 정보 전송 시 정보가 케이블을 통해 전송될 수 있도록 정보 형태를 변경 한다
 - 이더넷 카드(LAN 카드) 혹은 네트워크 어뎁터라고도 한다

2. 허브 : 한 사무실이나 가까운 거리의 컴퓨터를 연결하는 장치로, 각 회선을 통합적으로 관리하며, 신호 증폭 기능을 하는 리피터의 역항도 포함한다
 - 허브의 종류에는 더미 허브, 스위칭 허브가 있다
 ○ 더미 허브 (Dummy hub)
  - 네트워크에 흐르는 모든 데이터를 단순히 연결하는 기능만 제공
  - LAN이 보유한 대역폭을 컴퓨터 수만큼 나누어 제공
  - 네트워크에 연결된 각 노드를 물리적인 성형 구조로 연결
 ○ 스위칭 허브 (Switching hub)
  - 네트워크상에 흐르는 데이터의 유무 및 흐름을 제어하여 각각의 노드가 허브의 최대 대역폭을 사용할 수 있는 지능형 허브이다
  - 최근에 사용되는 허브는 대부분 스위칭 허브이다

3. 리피터(Repeater) : 전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행한다
 - OSI 참조 모델의 물리 계층에서 동작하는 장비이다
 - 근접한 네트워크 사이에 신호를 전송하는 역할로, 전송 거리의 연장 또는 배선의 자유도를 높이기 위해 사용한다

4. 브리지(Bridge) : LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹(세그먼트)를 연결하는 기능을 수행
 - 데이터 링크 계층 중 MAC(Media Access Control) 계층에서 사용되므로 MAC브리지라고도 한다
 - 네트워크 상의 많은 단말기들에 의해 발생되는 트래픽 병목 현상을 줄임
 - 네트워크를 분산적으로 구성할 수 있어 보안성을 높일 수 있다
 - 브리지를 이용한 서브넷 구성 시 전송 가능한 회선 수는 브리지가 N개일 때, n(n-1)/2 개 이다.

5. 스위치 : 브리지와 같이 LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치이다
 - 하드웨어를 기반으로 처리하므로 전송 속도가 빠름
 - 포트마다 각기 다른 전송 속도를 지원하도록 제어할 수 있고, 수십에서 수백개의 포트를 제공한다
 - OSI 참조 모델의 데이터 링크 계층에서 사용된다

6. 라우터 : 브리지와 같이 LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능이 추가된 것으로, 서로 다른 LAN이나 LAN과 WAN 연결도 수행한다
 - OSI 참조 모델의 네트워크 계층에서 동작하는 장비이다
 - 접속 가능한 경로에 대한 정보를 라우팅 제어표에 저장하여 보관한다
 - 3계층(네트워크 계층) 까지의 프로토콜 구조가 다른 네트워크 간의 연결을 위해 프로토콜 변환 기능을 수행한다

※브라우터(Brouter) : 브리지와 라우터의 기능을 모두 수행하는 장치로, 브리지 기능은 내부 네트워크를 분리하는 용도로, 라우터 기능은 외부 네트워크에 연결하는 용도로 사용

7. 게이트 웨이 : 전 계층 (1~7 계층)의 프로토콜 구조가 다른 네트워크의 연결을 수행한다
 - 세션 계층, 표현 계층, 응용 계층 간을 연결하여 데이터 형식 변환, 주소 변환, 프로토콜 변환 등을 수행된다
 - LAN에서 다른 네트워크에 데이터를 보내거나 다른 네트워크로부터 데이터를 받아들이는 출입구 역할을 한다

※ 전처리기 (FEP; Front End Processor)
- 통신 회선 및 단말장치 제어, 메시지의 조립과 분해, 전송 메시지 검사등을 미리 수행하여, 컴퓨터의 부담을 줄여주는 역할을 함
- 호스트 컴퓨터와 단말장치에 고속 통신 회선으로 설치;

섹션 164 프로토콜의 개념 (C)

: 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약
- 통신을 제어하기 위한 표준 규칙과 절차의 집합으로 하드웨어와 소프트웨어, 문서를 모두 규정한다

프로토콜의 기본 요소
1. 구문 (Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등 규정
2. 의미(Semantics) : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
3. 시간(Timing) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

프로토콜의 기능

단편화와 재결합	○ 송신 측에서 전송할 데이터를 전송에 알맞은 일정 크기의 작은 블록으로 자르는 작업을 단편화(Fragmentation)라 하고, 수신 측에서 단편화된 블록을 원래의 데이터로 모으는 것을 재결합(Reassembly)라 한다 ○ 단편화를 통해 세분화된 데이터 블록을 프로토콜 데이터 단위 PDU라고 한다 ○ 데이터를 단편화하여 전송하면, 전송 시간이 빠르고, 통신중의 오류를 효과적으로 제어할 수 있음 ○ 너무 작은 블록을 단편화할 경우 재결합 시 처리 시간이 길어지고, 실제 데이터 외에 부수적인 데이터가 많아지므로 비효율적이다.
캡슐화	○ 캡슐화는 단편화된 데이터에 송·수신지 주소, 오류 검출 코드, 프로토콜 기능을 구현하기 위한 프로토콜 제어 정보 등의 정보를 부가하는 것으로, 요약화라고 한다. ○ 대표적인 예가 데이터 링크 제어 프로토콜의 HDLC 프레임이다 ○ 정보 데이터를 오류 없이 정확하게 전송하기 위해 캡슐화를 수행함
흐름 제어	○흐름제어는 수신 측의 처리 능력에 따라 송신 측에서 송신하는 데이터의 전송량이나 전송 속도를 조절하는 기능이다 ○정지-대기(Stop-and-Wait), 슬라이딩 윈도우 방식을 이용한다
오류 제어	○ 오류제어는 전송중에 발생하는 오류를 검출하고 정정하여 데이터나 제어 정보의 파손에 대비하는 기능이다
동기화	송·수신 측이 같은 상태를 유지하도록 타이밍을 맞추는 기능이다
순서 제어	○전송되는 데이터 블록(PDU)에 전송 순서를 부여하는 기능으로, 연결 위주의 데이터 전송 방식에만 사용된다 ○송신 데이터들이 순서적으로 전송되도록 함으로써 흐름 제어 및 오류 제어를 용이하게 하는 기능을 한다
주소 지정	○데이터가 목적지까지 정확하게 전송될 수 있도록 목적지 이름, 주소, 경로를 부여하는 기능이다
다중화	○ 한 개의 통신의 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	○전송하려는 데이터가 사용하도록 하는 별도의 부가 서비스 ○우선순위 : 특정 메시지를 최대한 빠른 시간 안에 목적지로 전송하기 위해서 메시지 단위에 우선순위를 무여하여 우선순위가 높은 메시지가 먼저 도착하도록 함 ○서비스 등급 : 데이터의 요구에 따라 서비스 등급을 부여하여 서비스 ○ 보안성 : 액세스 제한과 같은 보안 체제를 구현

섹션 165 TCP/IP ★★★★★

: TCP/IP는 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜이다.
- TCP/IP는 1960년대 말 ARPA에서 개발하여 ARPANET에서 사용하기 시작
- TCP/IP는 UNIX의 기본 프로토콜로 사용되었고, 현재 인터넷 범용 프로토콜로 사용된다

TCP (Transmission Control Protocol)	○OSI 7계층의 전송 계층에 해당 ○신뢰성 있는 연결형 서비스 제공 ○패킷의 다중화, 순서 제어, 오류제어, 흐름 제어를 제공 ○스트림(Stream) 전송 기능 지공 ○TCP 헤더에는 송수신자포트 번호, 순서번호, Ack, 체크섬 포함
IP (Internet Protocol)	○OSI 7계층의 네트워크 계층에 해당 ○데이터그램을 기반으로 하는 비연결형 서비스 제공 ○Best Effort 원칙에 따른 전송 기능 제공 ○패킷의 분해/조립, 주소 지정, 경로 선택 기능 제공 ○헤더의 길이는 최소 20Byte에서 60byte ○IP헤더에는 송수신자 IP주소, 헤더 길이, 버전, 패킷 길이, 체크섬이 포함됨

TCP/IP의 구조

OSI	TCP/IP	기능
응용 계층 표현 계층 세션 계층	응용 계층	○응용 프로그램 간의 데이터 송수신 제공 ○TELNET, FTP, SMTP, SNMP, DNS, HTTP 등
전송 계층	전송 계층	○호스트들 간의 신뢰성 있는 통신 제공 ○TCP,UDP
네트워크 계층	인터넷 계층	○데이터 전송을 위한 주소 지정, 경로 설정 제공 ○IP, ICMP, IGMP, ARP, RARP
데이터링크 계층 물리 계층	네트워크 액세스 계층	○실제 데이터(프레임)을 송수신 하는 역할 ○Ethernet, IEEE 802, HDLC, X.25, RS-232C, ARQ등

응용 계층의 주요 프로토콜★★

FTP (File Transfer Protocol)	컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜
SMTP (Simple Mail Transfer Protocol)	전자 우편을 전송하는 프로토콜
TELNET	○원격 컴퓨터 접속 서비스 ○프로그램을 실행하는 등 시스템 관리 작업을 할 수 있는 가상의 터미널 기능을 수행
SNMP (Simple Network Management	TCP/IP의 네트워클 관리 프로토콜로, 라우터나 허브 등 네트워크 기기의 네트워크 정보를 네트워크 관리 시스템에 보내는 데 사용하는 표준 통신 규약
DNS (Domain Name System)	도메인 네임을 IP주소로 매핑해주는 시스템
HTTP (HyperText Transfer Protocol)	월드 와이드 웹에서 HTML 문서를 송수신하기 위한 표준 프로토콜
MQTT (Message Queuing Telemety Transport)	발행-구독 기반의 메시징 프로토콜로, IoT환경에서 주로 사용

전송 계층의 주요 프로토콜 ★★★

TCP (Transmission Control Protocol)	○양방향 연결 (Full Duplex Connection) 서비스를 제공 ○스트림 위주의 전달(패킷 단위)을 한다 ○가상 회선 연결(Virtual Circuit Connection) 형태의 서비스를 제공 ○신뢰성 있는 경로를 확립하고 메시지 전송ㅇ을 감독 ○순서 제어, 오류제어, 흐름 제어 기능을 함 ○패킷의 분실, 손상, 지연이나 순서가 틀린 것 등이 발생할 때 투명성이 보장되는 통신을 제공 ○ TCP 프로토콜의 헤더는 기본적으로 20~60 바이트까지 사용하고, 선택적으로 40을 추가할 수 있어 최대 100까지 크기 확장 가능
UDP (User Datagram Protocol)	○데이터 전송 전에 연결을 설정하지 않는 비연결형 서비스 제공 ○TCP에 비해 간단한 헤더 구조를 가지므로 오버헤드가 적고, 흐름 제어나 순서 제어가 없어 전송 속도도 빠름 ○고속의 안정성 있는 전송 매체를 사용하여 빠른 속도를 필요로 하는 경우, 동시에 여러 사용자에게 데이터를 전달할 경우, 정기적으로 반복해서 전송할 경우 사용 ○실시간 전송에 유리하며, 신뢰성보다는 속도가 중요시되는 네트워크에 사용 ○UDP헤더에는 송수신 포트 번호, 길이, 체크섬 등이 포함됨
RTCP (Real Time Control Protocol)	○RTP 패킷의 전송 품질을 제어하기 위한 프로토콜 ○세션에 참여한 각 참여자들에게 주기적으로 제어 정보 전송 ○하위 프로토콜은 데이터 패킷과 제어 패킷의 다중화(Multiflexing)을 제공 ○데이터 전송을 모니터링하고 최소한의 제어와 인증 긴으만을 제공 ○RTCP 패킷은 항상 32비트의 경계로 끝남

인터넷 계층의 주요 프로토콜 ★★★

IP	○전송할 데이터에 주소를 지정하고, 경로를 설정하는 기능 ○비연결형인 데이터그램 방식 이용으로 신뢰성 보장이 안됨
ICMP	IP와 조합하여 통신중에 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리하는 역할을 하며 헤더는 8바이트
IGMP (Internet Group Management Protocol)	멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용
ARP (Address Resolution Protocol)	호스트의 IP주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소 (맥 주소)로 바꿈
RARP (Reverse Address Resolution Protocol)	ARP와 반대로 맥주소를 IP주소로 바꿈

네트워크 액세스 계층의 주요 프로토콜★★

Ethernat(IEEE 802.3)	CSMA/CD 방식의 LAN
IEEE 802	LAN을 위한 표쥰 프로토콜
HDLC	비트 위주의 데이터 링크 제어 프로토콜
X.25	패킷 교환망을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜
RS-232C	공중 전화 교환망(PSTN)을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜