49. 공통 모듈 구현
재사용(Software Reusability)
소프트웨어 재사용의 개요

이미 개발되어 그 기능 및 성능, 품질을 인정받았던 소프트웨어의 전체 또는 일부분을 다시 사용하여 새롭게 개발하는 기법
재사용의 이점

개발 시간 및 비용 감소, 소프트웨어의 품질 향상, 소프트웨어의 생산성 향상
구축 방법에 대한 지식의 공유, 프로젝트 실패 위험 감소
재사용 범위에 따른 분류

함수와 객체 재사용, 컴포넌트 재사용, 애플리케이션 재사용
모듈화

모듈화는, 거대한 문제를 작은 조각의 문제로 나누어 다루기 쉽도록 하는 과정으로, 작게 나누어진 각 부분을 모듈이라고 한다.
소프트웨어의 모듈은 프로그래밍 언어에서 Subroutine, Function 등으로 표현될 수 있다.
모듈화는 시스템을 지능적으로 관리할 수 있도록 해주며, 복잡도 문제를 해결하는데 도움을 준다.
모듈화는 시스템의 유지보수와 수정을 용이하게 한다.
모듈(Module)의 개념

하나의 프로그램을 몇 개의 작은 부분으로 분할한 단위이다.
모듈의 독립성은 결합도와 응집도에 의해 측정된다.
각 모듈은 논리적 또는 기능적으로 분리되어 격리되고 독립적인 일을 수행한다.
특성 : Unity(한 가지 일만 수행), Smallness(간단 명료), Simplicity(단순성), Independency(독십성) 등
결합도(Coupling)

한 모듈과 다른 모듈 간의 상호 의존도 또는 두 모듈 사이의 연관 관계를 의미한다.

모듈 간의 결합도를 약하게 하면 모듈 독립성이 향상된다.

인터페이스가 정확히 설정되어 있지 않을 경우 불필요한 인터페이스가 나타나 모듈 사이의 의존도는 높아지고 결합도가 증가한다.

다른 모듈과 데이터 교류가 필요한 경우 전역 변수(Global Variable)보다는 매개 변수(Parameter)를 사용하는 것이 결합도를 낮추는데 도움이 된다.

결합도 정도 : 데이터 결합도 < 스탬프 결합도 < 제어 결합도 < 외부 결합도 < 공통 결합도 < 내용 결합도

데이터 결합도(Data Coupling)

한 모듈이 파라미터나 인수로 다른 모듈에 데이터를 넘겨주고 호출받은 모듈은 받은 데이터에 대한 처리 결과를 다시 돌려주는 경우의 결합도.
스탬프 결합도(Stamp Coupling)

두 모듈이 동일한 자료 구조를 조회하는 경우의 결하본
제어 결합도(Control Coupling)

한 모듈이 다른 모듈의 내부 논리 조직을 제어하기 위한 목적으로 제어 신호를 이용하여 통신하는 경우의 결합도
외부 결합도(External Coupling)

한 모듈에서 외부로 선언한 변수를 다른 모듈에서 참조할 경우의 결합도
공통 결합도(Common Coupling)

한 모듈이 다른 모듈에 제어 요소를 전달하고 여러 모듈이 공통 자료 영역을 사용하는 경우의 결합도
내용 결합도(Content Coupling)

한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 참조하는 경우의 결합도
응집도(Cohesion)

한 모듈 내부의 처리 요소 간의 기능적 연관도를 의미한다.

모듈 내부 요소는 명령어, 명령어의 모임, 호출문, 특정 작업 수행 코드 등이다.

응집도 정도 : 기능적 응집도 > 순차적 응집도 > 교환적 응집도 > 절차적 응집도 > 시간적 응집도 > 논리적 응집도 > 우연적 응집도

기능적 응집도(Functional Cohesion)

한 모듈 내부의 한 기능 요소에 의한 출력 자료가 다음 기능 원소의 입력 자료로써 제공되는 경우의 응집도

순차적 응집도(Sequential Cohesion)

모듈의 구성 요소가 하나의 활동으로부터 나온 출력 자료를 그다음 활동의 입력 자료로 사용하는 같은 모듈 내에서의 응집도
교환적 응집도(Communicational Cohesion)

동일한 입력과 출력을 사용하는 소 작업들이 모인 모듈에서 볼 수 있는 응집도.
절차적 응집도(Procedural Cohesion)

모듈이 다수의 관련 기능을 가질 때 모듈 내부의 기능 요소들이 그 기능을 순차적으로 수행할 경우의 응집도
시간적 응집도(Temporal Cohesion)

특정 시간에 처리되는 여러 기능을 모아 한 개의 모듈로 작성할 경우의 응집도
논리적 응집도(Logical Cohesion)

유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도
우연적 응집도(Coincidental Cohesion)

서로 간에 어떠한 의미 있는 연관 관계도 지니지 않은 기능 요소로 구성되는 경우의 응집도
효과적인 모듈화 설계 방법

응집도는 강하게, 결합도는 약하게 설계한다.
복잡도와 중복성을 줄이고 일관성을 유지할 수 있도록 설계한다.
유지보수가 용이하도록 설계한다.
모듈 크기는 시스템의 전반적인 기능과 구조를 이해하기 쉬운 크기로 설계한다.
모듈 기능은 예측이 가능해야 하며 지나치게 제한적이어서는 안 된다.
50. C 언어의 개요
C 언어의 기초
C 언어의 개념

1972 년 미국 벨 연구소의 데니스 리치에 의해 개발되었다.
컴파일러 방식의 언어이다.
시스템 프로그래밍에 가장 적합한 언어이다.
포인터에 의한 번지 연산 등 다양한 연산 기능을 가진다.
이식성이 뛰어나 컴퓨터 기종에 관계없이 프로그램을 작성할 수 있다.
UNIX 운영체제를 구성한다.

C 언어의 기본 구조

main 함수를 반드시 포함해야 하며, main 함수에서 실행이 시작된다.
영문 대/소문자를 엄격하게 구별한다.
문장을 끝마칠 때는 세미콜론(;)을 사용한다.
여러 개의 문장을 묶어 하나의 블록으로 구성할 때 중괄호({})를 사용한다.
주석문은 /_ ~ _/로 표기한다.
기본 자료형

자료형 예약어 크기
정수형 int 2Byte
long 4Byte
실수형 float 4Byte
double 8Byte
문자형 char 1Byte
기억 클래스

자동 변수(Automatic Variables)
레지스터 변수(Register Variables)
정적 변수(Static Variables)
외부 변수(External Variables)
입/출력 함수

표준 입/출력 함수
scanf() : 표준 입력 함수
printf() : 표준 출력 함수
getchar() : 문자 입력 함수
putchar() : 문자 출력 함수
gets() : 문자열 입력 함수
puts() : 문자열 출력 함수
변환 문자(출력 형식 지정 문자)
%d : 10 진 정수
%o : 8 진 정수
%x : 16 진 정수
%f : 실수형
%e : 지수형
%c : 문자

%s : 문자열
이스케이프 시퀀스(Escape Sequence)

문자	의미	기능
\n	new line	커서를 다음 줄 처음으로 이동한다.
\r	carriage return	커서를 현재 줄 처음으로 이동한다.
\t	tab	커서를 일정 간격만큼 띄운다.
\b	backspace	커서를 뒤로 한 칸 이동한다.
\f	form feed	한 페이지를 넘긴다.
\0	null character	널 문자를 출력한다.
'	single quote	작은 따옴표를 출력한다.
"	double quote	큰따옴표를 출력한다.
\ \	backslash	역슬래시를 출력한다.
\a	alert	벨 소리를 발생한다.

C 언어 변수명 작성 규칙

영문 대소문자(AZ, az), 숫자(0~9), '_'를 혼용하여 사용할 수 있다.
첫 글자는 숫자로 시작할 수 없고, 영문자나 "_"로 시작해야 한다.
영문자는 대소문자를 구분한다.
공백을 포함할 수 없다.
auto, beak, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register,return, short, signed, sizeof, static, struct, switch, typedef, union, unsigend, void, volatile, while 32 개 예약어(Reserved Word)를 사용할 수 없다.
포인터 변수

다른 변수의 주소값을 저장할 수 있다.
표인터 변수는 자료형에 상관없이 메모리 크기가 동일하다.
가리키고 있는 변수값을 읽기 위해서는 포인터 연산자를 사용한다.
포인터 변수에 일반 변수의 주소를 대입하기 위해서는 &(주소) 연산자를 사용한다.
포인터 변수를 가리키는 포인터 변수를 선언할 수 있다.
표준 라이브러리 함수

stdio.h : C 언어의 표준 입/출력 라이브러리(Standard Input and Output Library)이다.
stdlib.h : C 표준 유틸리티 함수를 모아놓은 헤더 파일이다. 문자형 변환, 수치를 문자형으로 변환, 동적 할당 관련 함수, 난수 생성 함수, 정수의 연산 함수, 검색 및 정렬 함수 등이다.

stdlib.h 함수 종류
atoi() : 문자열을 정수형으로 변환
atof() : 문자열을 실수형으로 변환
atol() : 문자열을 log 형 정수로 변환
itoa() : 숫자를 문자열로 변환
ceil() : 자리 올림
floor() : 자리 버림
rand() : 난수 발생
div() : 정수 나눗셈
51. C 언어의 연산자
C 언어 연산자의 종류
연산자의 종류 및 우선순위

연산자 종류	결합 방향	우선순위
단항 연산자	+, -, !, ~, ++, --, &, *, sizeof	<- 높음
산술 연산자	*, / ,%	->
	+, -	->
시프트 연산자	<<, >>	->
관계 연산자	<, <=, >, >=	->
	==, !=	->
비트 연산자	&, |, ^	->
논리 연산자	&&, ||	->
조건 연산자	? :	<-
할당 연산자	=, +=, -=, *=, /=, %=, <<=, >>=	<-
콤마 연산자	,	-> 낮음

C 언어 연산자의 특징
단항 연산자

! : 부정(NOT)
~ : 1 의 보수(0->1, 1->0)를 구한다.
++ : 1 씩 증가를 의미한다.
-- : 1 씩 감소를 의미한다.
& : 변수의 주소를 의미한다.
* : 변수의 내용을 의미한다.
sizeof : 변수, 변수형, 배열의 저장 장소의 크기를 Byte 단위로 구한다.
산술 연산자

이항 연산자는 +, -는 *, /, % 보다 우선순위가 낮다.
% : 정수 나눗셈 연산 후 나머지를 구한다.
시프트(shift) 연산자

<<는 비트를 왼쪽으로 이동시킨다.
>>는 비트를 오른쪽으로 이동시킨다.
관계 연산자

< : ~보다 작다.
> : ~보다 크다.
<= : ~보다 작거나 같다.
>\ : ~보다 크거나 같다.
== : ~와 같다.
!= : ~와 같지 않다.
비트 연산자

& : 논리곱(AND)
| : 논리합(OR)
^ : 배타적 논리합(XOR)

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    int a = 4;
    int b = 7;
    int c = a|b;
    printf("%d", c);
    return 0;
}
```

// 1. 변수 a 와 b 의 4, 7 을 (2 진수)비트 연산자 |(OR)로 연산한다.
// 2. 비트 연산자는 2 진수로 변환 후 계산하낟.
// 3. OR 연산자는 두 비트 중 1 개라도 1 이면 1 이 출력된다.
//   0100 (10 진수: 4)
// OR 0111 (10 진수: 7)
// =   0111(둘 중 1 개라도 1 이면 1 이기 때문에 10 진수: 7)
// 4. 0111 는 "%d" 출력 형식 지정 문자에 의해 10 진수로 변환하면 7 이 되어 출력된다.
//
// 2 와 5 를 가정할때
//   0010
// OR 0101
// =   0111

논리 연산자

! : 논리 부정(NOT)
&& : 논리곱(AND)
|| : 논리합(OR)
조건 연산자

C 언어에서 유일하게 3 개의 피연산자를 갖는 삼항 연산자이다.
조건식 ? 참일 경우 값 : 거짓일 경우 값
ex. big = a > b ? a:b; -> a 와 b 중에서 큰 수가 big 에 저장됨
할당 연산자

= : a = b -> b 를 a 에 할당
+= : a += b -> a = a+b
-= : a -= b -> a = a-b
= : a=b -> a = a*b
/= : a/=b -> a = a/b
%= : a%=b -> a = a%b
<<= : a<<=b -> a = a<<b
>>= : a>>=b -> a = a>>b
콤마(4 열)연산저

성격이 동일한 자료형을 나열할 때 사용된다.
라이브러리
라이브러리의 개념과 구성

라이브러리란 필요할 때 찾아서 쓸 수 있도록 모듈화되어 제공되는 프로그램을 말한다.
프로그래밍 언어에 따라 일반적으로 도움말, 설치 파일, 샘플 코드 등을 제공한다.
라이브러리는 모듈과 패키지를 총칭하며, 모듈이 개별 파일이라면 패키지는 파일들을 모아 놓은 폴더라고 볼 수 있다.
표준 라이브러리는 프로그래밍 언어가 기본적으로 가지고 있는 라이브러리를 의미하며,
외부 라이브러리는 별도의 파일 설치를 필요로 하는 라이브러리를 의미한다.
52. Java 언어
Java 언어의 기초
Java 언어의 개념

객체지향 언어이다.
추상화, 상속화, 다형성과 같은 특징을 가진다.
네트워크 환경에서 분산 작업이 가능하도록 설계되었다.
특정 컴퓨터 구조와 무관한 가상 바이트 머신 코드를 사용하므로 플랫폼이 독립적이다.

Garbage Collector

S/W 개발 중 유효하지 않은 가비지 메모리가 발생한다. Java 에서는 C 언어와 달리 JVM 가비지 컬렉터가 불필요 메모리를 알아서 정리해준다.

Java 언어의 기본 자료형

분류	예약어	바이트 수	비고
정수형	byte	1byte	-127 ~ +128
	short	2byte	-32,768 ~ +32,767
	int	4byte	-2,147,483,648 ~ +2,147,483,648
	long	8byte	-9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,808
실수형	float	4byte	단정도 실수형 (유효 자리는 7 정도임)
	double	8byte	배정도 실수형 (유효 자리는 15 정도)
문자형	char	2byte	유니코드 문자열 1 자
논리형	boolean	1byte	true,false

이스케이프 시퀀스(Escape Sequence)

문자	의미	기능
\n	new line	커서를 다음 줄 처음으로 이동한다.
\r	carriage return	커서를 현재 줄 처음으로 이동한다.
\t	tab	커서를 일정 간격만큼 띄운다.
\b	backspace	커서를 뒤로 한 칸 이동한다.
\f	form feed	한 페이지 넘긴다.
'	single quote	작은따옴표를 출력한다.
"	double quote	큰따옴표를 출력한다.
\ \	backslash	역슬래시를 출력한다.

Java 접근 제한자(접근 제어자)

public : 모든 접근을 허용한다.

private : 같은 패키지에 있는 객체와 상속 관계의 객체들만 허용한다.

default : 같은 패키지에 있는 객체들만 허용한다.

protected : 현재 객체 내에서만 허용한다.

Java 의 출력 함수

System.out.print() : 괄호 안을 출력하고 줄 바꿈을 안 한다.

System.out.printIn() : 괄호 안을 출력하고 줄 바꿈을 한다.

System.out.printf() : 변환 문자를 사용하여 출력한다.
변환 문자(출력 형식 지정 문자)
%d : 10 진 정수
%o : 8 진 정수
%x : 16 진 정수
%f : 실수형
%e : 지수형
%c : 문자
%s : 문자열
Java 언어 변수명 작성 규칙

영문 대소문자(AZ, az), 숫자(0~9), '_', '$'를 혼용하여 사용할 수 있다.
첫 글자는 영문자나 '_', '$'로 시작해야 한다.
영문자는 대소문자를 구분한다.
공백을 포함할 수 없다.
예약어(Reserved Word)를 사용할 수 없다.
Java 언어의 연산자
연산자의 종류 및 우선순위

연산자 종류 결합 방향 우선순위
단항 연산자 +, -, !, ~, ++, -- <- 높음
산술 연산자 *, / ,% ->
+, - ->
비트 시프트 <<, >>, >>> ->
관계 연산자 <, <=, >, >=, instanceof ->
==, != ->
논리 연산자 &, |, ^ ->
비트 논리 &&, || ->
조건 연산자 ? : <-
할당 연산자 =, +=, -=, *=, /=, %=, <<=, >>= <- 낮음
오버로딩(Overloading)과 오버라이딩(Overriding)
오버로딩(Overloading - 과적, 과부하)

한 클래스 내에서 같은 메서드를 사용하는 것이다.
같은 이름의 메서드를 여러 개 정의하면서 매개 변수의 유형과 개수가 달라지도록 하는 기술이다.
오버라이딩(Overriding - 가장 우선되는, 최우선으로 되는, 다른 것보다 우선인)

상속 관계의 두 클래스의 상위 클래스에서 정의한 메서드를 하위 클래스에서 변경(재정의)하는 것이다.
Java 언어에서는 static 메ㅓㅅ드의 오버라이딩을 허용하지 않는다.
오버라이딩의 경우 하위 객체의 매개 변수 개수와 타입은 상위 객체와 같아야 한다.
53. 제어문
조건문
if 문

if 문

```
    if(조건식) ~ : 조건식의 결과가 참일 때 실행하는 명령문;
```

if / else 문

```
    if(조건식 1) 조건식 1 이 참일 때 실행하는 명령문;
    else if(조건식 2) 조건식 2 의 결과가 참일 때 실행하는 명령문;
    else 조건식 1 과 조건식 2 의 결과가 거짓일 때 실행하는 명령문;
```

삼항 연산자에 의한 조건문

```
조건식 ? 참일 때 명령문 : 거짓일 때 명령문
```

switch ~ case 문

```
switch(조건값) {
    case 값 1:
        조건값이 값 1 일 때 실행하는 명령문;
        break;
    case 값 2:
        조건값이 값 2 일 때 실행하는 명령문;
        break;
    ...
    default:
        조건값이 모든 case 에 해당되지 않을 때 실행하는 명령문;
        break;
}
```

반복문
while 문

조건식의 결과가 참이면 while 문 내의 명령을 실행하고 다시 조건식을 검사한다.
조건의 초기값이 거짓이면 while 문 내의 명령문은 한 번도 실행되지 않는다.

```
while(조건식){
```

```
    명령문 1;
    ...
    명령문 n;
}
```

do ~ while 문

명령문을 일단 실행하고 나서 조건식을 검사하여 반복 실행 여부를 결정한다.
명령문이 적어도 한 번은 실행된다.

```
do {
    명령문 1;
    ...
    명령문 n;
} while(조건식);
```

for 문

반복 변수를 초기화하는 초기식은 한 번만 수행되고 조건식을 만족하면 하위 명령문을 수행한 후 증감식을 수행하고 조건식을 검사하면서 반복한다.

```
for(초기식; 조건식; 증감식){
    명령문 1;
    ...
    명령문 n;
}
```

54. 스크립트 언어와 Python

스크립트 언어

스크립트 언어(Script Language)의 개념

소스 코드를 컴파일 과정을 거치지 않고 실행할 수 있는 프로그래밍 언어이다.
스크립트 언어에 내장된 번역기에 의해 번역되어 실행된다.
실행 단계에서 구분을 분석한다.
스크립트 언어의 종류

서버 측 스크립트 언어

ASP(Active Server Page)

서버 측에서 동적으로 수행되는 페이지를 만들기 위한 언어로 Windows 계열의 운영체제에서 실행 가능하다.

JSP(Java Server Page)

Java 를 기반으로 하고 서버 측에서 동적으로 수행하는 페이지를 만드는 언어이다.

PHP(Professional Hypertext Preprocessor)

소스 코드가 HTML 파일에 포함되는 언어이다.
데이터베이스와의 연동이 매우 용이하다.

Linux, UNIX, Window 등의 다양한 운영 체제에서 사용 가능하다.
PHP 연산자
산술 연산자 : +, -, *, /, %, **
할당 연산자 : =, +=, -=, *=, /=, %=
증감 연산자 : ++, --
관계 연산자 : ==, ===, !=, <>, !==, >, <, >=, <=
논리 연산자 : and, or, xor, &&, ||, !
파이썬(Python)
인터프리터 방식의 객체지향 언어이다.
실행 시점에 데이터 타입을 결정하는 동적 타이핑 기능을 갖는다.
클라이언트 측 스크립트 언어
JavaScript
HTML 문서에서 HTML 이나 CSS 로 표현하기 어렵거나 불가능한 작업을 수행하기 위해 개발되었다.
소스 코드가 HTML 문서에 포함되어 있다.
클래스가 존재하지 않으며 변수 선언도 필요 없다.(?)
사용자의 웹 브라우저에서 직접 번역되고 시랳ㅇ된다.
VBScript
마이크로소프트가 개발한 액티브 스크립트 언어이다.
VBScript 의 구분은 비주얼 베이직(Visual Basic) 프로그래밍 언어를 일부 반영한다.
파이썬(Python)
파이썬의 개요

1991 년 귀도 반 로섬(Guido van Rossum)이 개발한 고급 프로그래밍 언어이다.
플랫폼에 독립적이고 인터프리터식, 객체지향적, 동적 타이핑(Dynamicall Typed) 대화형 언어이다. 매우 쉬운 문법 구조로 초보자들도 쉽게 배울 수 있다.
파이썬 변수명 작성 규칙

영문 대소문자(AZ, az), 숫자(0~9), '_'를 혼용하여 사용할 수 있다.
첫 글자는 영문자나 '_'로 시작해야 한다.
영문자는 대소문자를 구분한다.
공백을 포함할 수 없다.
예약어(Reserved Word)를 사용할 수 없다.
문자열 추출하기

하나의 문자를 추출하려면 추출하려는 문자의 인덱스(0 부터 시작)을 지정한다.
```
string = 'hello python'
```

```
s1 = string[0]
print(s1) # 'h'

s2 = string[4]
print(s2) # 'o'

# 역순으로 맨 오른쪽의 인덱스는 -1 이다.

s3 = string[-1]
print(s3) # 'n'

s4 = string[-6]
print(s4) # 'p'

# [:]는 처음부터 끝까지 추출한다.

s5 = string[:]
print(s5) # 'hello python'

# [x:] 인덱스 x 부터 끝까지 추출한다.

s6 = string[6:]
print(s6) # 'python'

s7 = string[-6:]
print(s7) # 'python'

# [:y] 처음부터 인덱스 y-1 까지 추출한다.

s8 = string[:5]
print(s8) # 'hello'

# [x:y] 인덱스 x 부터 y-1 까지 추출한다.
s9 = string[4:7]
print(s9) # 'o p'

# [x:y:z] 인덱스 x 부터 y-1 까지 z 만큼 건너뛰면서 추출한다.
s10 = string[:10:2]
print(s10) # 'hlopt'

s11 = string[1:10:2]
print(s11) # 'el yh'
```

55. 운영체제의 개요

운영체제의 개요

운영체제(OS : Operating System)의 개념

운영체제는 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어이다.

운영체제는 컴퓨터를 편리하게 사용하고 컴퓨터 하드웨어를 효율적으로 사용할 수 있도록 한다.
운영체제는 스스로 어떤 유용한 기능도 수행하지 않고 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 마련해준다.
운영체제의 종류로는 MS-DOS, Windows 10, Linux, UNIX, OS/2, 안드로이드, IOS 등이 있다.
운영체제의 기능

사용자와 시스템 간의 편리한 인터페이스를 제공한다.
컴퓨터의 시스템의 성능을 최적화시킨다.
자원의 효과적인 경영을 위해 스케줄링 기능을 제공한다.
자원 보호 기능을 제공한다.
시스템에서 발생하는 오류로부터 시스템을 보호한다.
사용자들 간에 데이터를 공유할 수 있도록 한다.
운영체제의 목적

처리 능력(Throughput) 향상
처리 능력은 일정 시간 내에 시스템이 처리하는 일의 양이다.
처리 능력이 높을수록 처리하는 일의 양이 많아진다.
반환 시간(Turnaround Time)감소
반환 시간은 컴퓨터 센터에 작업을 지시하고 나서부터 결과를 받을 때까지의 경과 시간이다.
반환 시간이 감소될수록 처리 속도가 빨라진다.
신뢰도(Reliability) 향상
신뢰도는 시스템이 주어진 문제를 정확하게 해결하는 정도이다.
신뢰도가 높을수록 일을 정확하게 처리한다.
사용 가능도(Availability) 향상
사용 가능도는 한정된 자원을 여러 사용자가 요구할 때, 어느 정도 신속하고 충분히 지원해 줄 수 있는지의 정도이다.
사용 가능도가 높을수록 반환 시간이 감소한다.
운영체제의 운영 방식

일괄 처리 시스템(Batch Processing System)
일정량 또는 일정 기간 동안 데이터를 한꺼번에 모아서 처리하는 방식이다.
운영체제 운용 방식 중 시대적으로 가장 먼저 생겨났다.
ex. 수도요금 계산 업무, 월급 계산 업무 등

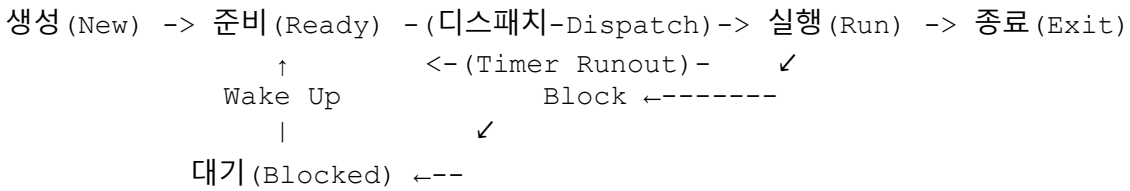
다중 프로그래밍 시스템(Multi programming System)
컴퓨터 시스템 자원 활용률을 극대화하기 위해 2 개 이상의 프로그램을 주기억 장치에 기억시키고 CPU 를 번갈아 사용하면서 처리하는 방식이다.
시분할 시스템(Time Sharing System)
CPU 의 전체 사용 시간을 작은 작업 시간량(Time Slice)으로 나누어서 그 시간량 동안만 번갈아 가면서 CPU 를 할당하여 각 작업을 처리하는 방식이다.
실제로 많은 사용자가 하나의 컴퓨터를 공유하고 있지만 마치 자신만이 컴퓨터 시스템을 독점하여 사용하고 있는 것처럼 느끼게 된다.
다중 처리 시스템(Multi-Processing System)
동시에 프로그램을 수행할 수 있는 CPU 를 두 개 이상 두고 각각 그 업무를 분담하여 처리할 수 있는 방식이다.
실시간 처리 시스템(Real Time Processing System)
데이터 발생 즉시, 또는 데이터 처리 요구가 있는 즉시 처리하여 결과를 산출하는 방식이다.
정해진 시간에 반드시 수행되어야 하는 작업들을 처리할 때 가장 정합하다.
ex. 항공기 예약 업무, 은행 창구 업무, 조회 및 질의 업무 등
다중 모드 시스템(Multi-mode System)
일괄 처리 + 시분할 + 다중 처리 + 실시간 처리
분산 처리 시스템(Distributed Processing System)
여러 대의 컴퓨터로 작업을 나누어 처리하여 그 내용이나 결과를 통신망을 이용하여 상호 교환되도록 연결하는 방식이다.
운영체제의 구성
제어 프로그램(Control Program)

감시 프로그램(Supervisor Program)
자원의 할당 및 시스템 전체의 작동 상태를 감시/감독하는 프로그램이다.
제어 프로그램에서 가장 핵심이 된다.
작업 제어 프로그램(Job Control Program)
어떤 업무를 처리하고 다른 업무로의 이행을 자동적으로 수행하기 위한 준비 및 그 처리 완료를 담당하는 기능을 수행한다.
작업의 연속 처리를 위한 스케줄 및 시스템 자원 할당 등을 담당한다.
데이터 관리 프로그램(Data Management Program)
주기억 장치와 보조 기억 장치 사이의 자료 전송, 파일의 조작 및 처리, 입/출력 자료와 프로그램 간의 논리적 연결 등 시스템에서 취급하는 파일과 데이터를 표준적인 방법으로 처리할 수 있도록 관리한다.
처리 프로그램(Processing Program)

언어 번역 프로그램(Language Translator Program)
프로그래머가 작성한 원시 프로그램을 컴퓨터가 이해할 수 있는 형식으로 번역한다.
종류 : 컴파일러, 어셈블러, 인터프리터 등
서비스 프로그램(Service Program)
사용자의 편의를 위해 사용 빈도가 높은 프로그램을 시스템 제공자가 미리 작성하여 사용자에게 제공해 주는 프로그램이다.
종류 : 연계 편집, 유틸리티, 정렬, 병합 등
문제 프로그램(Problem Program)
특정 업무를 처리하기 위해 사용자가 작성한 프로그램이다.
56. 프로세스 관리
프로세스
프로세스(Process)의 정의

실행 중인 프로그램이다.
실행 가능한 PCB 를 가진 프로그램이다.
프로세서가 할당되는 실체이다.
프로시저가 활동 중인 것이다.
비동기적 행위를 일으키는 주체이다.
프로세스 제어 블록(PCB: Process Control Block)

운영체제가 프로세스를 관리하기 위해 프로세스에 대한 중요한 정보를 저장해 놓은 곳이다.
프로세스가 생성될 때마다 고유의 PCB 가 생성되며, 프로세스가 소멸되면 PCB 도 소멸된다.
PCB 에 저장되어 있는 정보 : 프로세스의 현재 상태, 프로세스의 우선순위, 프로세스에 할당된 자원에 대한 정보, CPU 레지스터 정보
프로세스 상태 전이



준비 상태(Ready State)
프로세스가 CPU 를 할당받기 위해 준비하고 있는 상태이다.
실행 상태(Running State)
준비 상태의 프로세스가 CPU 를 할당받아 실행 중인 상태이다.

디스패치(Dispatch) : 우선순위가 가장 높은 프로세스가 준비 상태에서 실행 상태로 전환되는 것이다.
할당 시간 종료(Time Runout) : 실행 상태의 프로세스가 할당 시간(타이머)이 종료되어 준비 상태로 전환되는 것이다.
대기 상태(Blocked State)
실행 상태의 프로세스가 종료되기 전에 입/출력 등의 다른 작업이 필요할 경우 CPU 를 반납하고 작업의 완료를 기다리는 상태이다.
블록(Block) : 실행 상태에서 대기 상태로 전환되는 것이다.
웨이크 업(Wake Up) : 대기 상태의 프로세스가 웨이크업(조건 만족)되면 준비 상태로 전환된다.
스레드(Thread)

프로세스 내에서의 작업 단위로서 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위를 의미한다.
하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상시킬 수 있다.
한 개의 프로세스는 여러 개의 스레드를 가질 수 있다.
병행 프로세스와 교착상태
병행 프로세스(Concurrent Process)

두 개 이상의 프로세스들이 동시에 실행 상태에 있는 것이다.
병행 프로그래밍 기법 하에서 발생할 수 있는 오류에 대한 오류 방지 방법에는 임계구역, 상호배제, 동기화 기법이 있다.
임계 영역(Critical Section)
어느 한 시점에서 하나의 프로세스가 자원 또는 데이터를 사용하도록 지정된 공유 영역이다.
임계영역에서의 작업은 신속하게 이루어져야 한다.
임계영역 내의 프로그램에서는 교착상태가 발생하지 않도록 해야 한다.
임계영역 내의 프로그램에서는 무한 반복이 발생하지 않도록 해야 한다.
상호 배제(Mutual Exclusion)
공유 변수를 접근하고 있는 하나의 프로세스 외에는 다른 모든 프로세스들이 공유 변수를 접근하지 못하도록 제어하는 기법이다.
상호배제 구현 기법 : 데커 알고리즘, 피터슨 알고리즘, Lamport 의 빵집 알고리즘, Test and set 명령어 기법, Swap 명령어 기법
동기화 기법(Synchronization)
세마포어(Semaphore) : Dijkstra 가 제안한 방법으로, 연산 P 와 V 를 통해서 프로세스 사이의 동기를 유지하고 상호배제의 원리를 보장한다.

모니터(Monitor) : 공유 데이터와 이 데이터를 처리하는 프로시저를 포함하는 병행성 구조이다. 모니터의 경계에서 상호배제가 시행되며, 모니터 외부에서는 모니터 내부의 데이터를 직접 액세스할 수 없다.
교착상태(Deadlock)

둘 이상의 프로세스들이 서로 다른 프로세스가 차지하고 있는 자원을 요구하며 무한정 기다리게 되어 해당 프로세스들의 진행이 중단되는 현상이다.
교착상태의 발생 조건
상호배제(Mutual Exclusion)
한 번에 한 프로세스만이 어떤 자원을 사용할 수 있다.
점유 및 대기(Hold and Wait)
프로세스는 다른 자원이 할당되기를 기다리는 동안 이미 확보한 자원을 계속 보유하고 있다.
비선점(Non-preemption)
자원을 보유하고 있는 프로세스로부터 다른 프로세스가 강제로 그 자원을 빼앗을 수 없다.
환형 대기(Circular Wait)
이미 자원을 가진 프로세스가 앞이나 뒤의 프로세스의 자원을 요구한다.
교착상태의 해결 방법
예방(Prevention)
교착상태가 발생하지 않도록 사전에 시스템을 제어하는 방법이다.
일반적으로 자원의 낭비가 가장 심한 것으로 알려진 기법이다.
회피(Avoidance)
교착상태 발생 가능성을 인정하고 교착상태가 발생하려고 할 때, 교착상태 가능성을 피해 가능 방법이다.
주로 은행가 알고리즘(Banker's Algorithm)을 사용한다.
발견(Detection)
교착상태가 발생했는지 검사하여 교착상태에 빠진 프로세스와 자원을 발견하는 방법이다.
회복(Recovery)
교착상태에 빠진 프로세스를 종료하거나 해당 프로세스가 점유하고 있는 자원을 선점하여 다른 프로세스에게 할당하는 기법이다.
57. 프로세스 스케줄링
프로세스 스케줄링의 개요
프로세스 스케줄링(Process Scheduling)의 개념

프로세스의 생성 및 실행에 필요한 시스템의 자원을 해당 프로세스에 할당하는 작업이다.

다중 프로그래밍 운영체제에서 자원의 성능을 향상시키고 효율적인 프로세서의 관리를 위해 작업 순서를 결정하는 것이다.
프로세스 스케줄링의 목적

모든 작업들에 대한 공평성 유지, 단위 시간당 처리량 최대화, 응답 시간 및 반환 시간 최소화, 운영체제의 오버헤드 최소화
바람직한 스케줄링 정책

CPU 이용률 최대화, 응답 시간 및 반환 시간 최소화, 대기 시간 최소화
프로세스 스케줄링 기법
비선점(Non-Preemptive) 스케줄링

한 프로세스가 일단 CPU 를 할당받으면 다른 프로세스가 CPU 를 강제로 빼앗을 수 없고, 사용이 끝날 때까지 기다리는 방식이다.
모든 프로세스들에 대한 요구를 공정히 처리하여 응답 시간의 예측이 용이하다.
CPU 의 사용 시간이 짧은 프로세스들이 사용 시간이 긴 프로세스들로 인하여 오래 기다리는 경우가 발생할 수 있다.
FIFO(First in Fist Out)

준비 상태 큐에 도착한 순서대로 CPU 를 할당하는 기법이다.
FCFS(Fist Come First Service)라고도 한다.
ex. FIFO 스케줄링에서 다음과 같은 3 개의 작업에 대하여 모든 작업들의 평균 대기 시간 및 평균 반환 시간은?

작업	도착 시간	실행 시간
P1	0	13
P2	3	35
P3	8	10

실행 순서 : P1 -> P2 -> P3
대기 시간 : P1(0), P2(10), P3(40)
평균 대기 시간 : (0 + 10 + 40) / 3 = 16.66
반환 시간 : P1(13), P2(45), P3(50)
평균 반환 시간 : (13 + 45 + 50) / 3 = 36
SJF(Shortest Job First)

준비 상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에게 먼저 CPU 를 할당하는 스케줄링 기법이다.
평균 대기 시간을 최소화한다.
ex SJF 스케줄링에서 다음과 같이 4 개의 작업이 준비 상태 큐에 있을 때 모든 작업들의 평균 대기 시간 및 평균 반환 시간은?

작업 실행시간
P1 6
P2 3
P3 8
P4 7
실행 순서 : P2 -> P1 -> P4 -> P3
대기 시간 : P2(2), P1(3), P4(9), P3(16)
평균 대기 시간 : (0 + 3 + 9 + 16) / 4 = 7
반환 시간 : P2(3), P1(9), P4(16), P3(24)
평균 반환 시간 : (3 + 9 + 16 + 24) / 4 = 13
HRN(Highest Responseratio Next)

어떤 작업이 서비스받을 시간과 그 작업이 서비스를 기다린 시간으로 결정되는 우선순위에 따라 CPU 를 할당하는 기법이다.
우선순위 계산식 = (대기 시간 + 서비스를 받을 시간) / 서비스를 받을 시간
ex. HRN 방식으로 스케줄링할 경우, 입력된 작업이 다음과 같을 때 처리되는 작업 순서는?

작업	대기 시간	서비스(실행) 시간	우선순위
P1	5	20	5/4
P2	40	20	3
P3	15	45	4/3
P4	20	20	2

우선순위(Priority)

준비 상태 큐에서 대기하는 프로세스에게 부여된 우선순위가 가장 높은 프로세스에게 먼저 CPU 를 할당하는 기법이다.
우선순위가 낮은 프로세스는 무한 정지(Indefinite Blocking)가 발생할 수 있으며, 에이징(Aging) 기법으로 이를 해결할 수 있다.
선점(Preemptive) 스케줄링

한 프로세스가 CPU 를 할당받아 실행 중이라도 우선순위가 높은 다른 프로세스가 CPU 를 강제적으로 빼앗을 수 있는 방식이다.
긴급하고 높은 우선순위의 프로세스들이 빠르게 처리될 수 있다.
선점을 위한 시간 배당에 대한 인터럽트용 타이머 클럭(Clock)이 필요하다.
온라인 응용에 적합한 스케줄링이다.
종류
RR(Round Robin)
주어진 시간 할당량(Time Slice) 안에 작업을 마치지 않으면 준비 상태 큐의 가장 뒤로 배치된다.
시분할 시스템(Time-sharing System)을 위해 고안된 방식이다.

시간 할당량이 커지면 FCFS 스케줄링과 같은 효과를 얻을 수 있다.
시간 할당이 작아지면 프로세스 문맥 교환이 자주 일어난다.
SRT(Shortest Remainig Time)
작업이 끝나기까지의 실행시간 추정치가 가장 작은 작업을 먼저 실행시키는 기법이다.
FIFO 기법보다 평균 대기 시간이 감소된다.
작업 시간이 큰 경우 오랫동안 대기하여야 한다.
다단계 큐(Multi-Level Queue)
프로세스들을 우선순위에 따라 상위, 중위, 하위 단계의 단계별 준비 상태 큐를 배치하는 기법이다.
다단계 피드백 큐(Multi-Level Feedback Queue)
각 준비 상태 큐마다 부여된 시간 할당량 안에 완료하지 못한 프로세스는 다음 단계의 준비 상태 큐로 이동하는 기법이다.
58. 기억 장치 관리
기억 장치 관리 전략
반입 전략

보조 기억 장치에 보관 중인 프로그램이나 데이터를 주기억 장치로 언제 가져올 것인지 결정하는 전략으로 요구 반입, 예살 반입이 있다.
배치 전략

보조 기억 장치에 보관 중인 프로그램이나 데이터를 주기억 장치 내의 어디로 가져올 것인지 결정하는 전략이다.
최초 적합(First-Fit) : 적재 가능한 공간 중에서 첫 번째 분할 영역에 배치
최적 적합(Best-Fit) : 적재 가능한 공간 중에서 가장 작은 공백이 남는 부분에 배치
최악 적합(Worst-Fit) : 적재 가능한 공간 중에서 가장 큰 공백이 남는 부분에 배치
기억 장치 교체 전략

주기억 장치의 모든 페이지 프레임이 사용 중일 때 어떤 페이지 프레임을 교체할 것인지 결정하는 전략이다.
OPT(OPTimal Replacement)
이후에 가장 오랫동안 사용되지 않을 페이지를 먼저 교체하는 기법이다.
실현 가능성이 희박하다.
FIFO(First in Fist Out)
가장 먼저 적재된 페이지를 먼저 교체하는 기법이다.
구현이 간단하다.
LRU(Least Recently Used)
각 페이지마다 계수기나 스택을 두어 현시점에서 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법이다.

LFU(Least Frequently Used)
참조된 횟수가 가장 적은 페이지를 먼저 교체하는 기법이다.
NUR(Not Used Recently)
각 페이지당 두 개의 하드웨어 비트를 두어서 가장 최근에 사용하지 않은 페이지를 교체하는 기법이다.
SCR(Second Chance Replacement)
FIFO 의 단점을 보완하는 기법으로, 가장 오랫동안 주기억 장치에 상주했던 페이지 중에서 자주 참조되는 페이지의 교체를 예방한다.
가상 기억 장치 구현 기법
가상 기억 장치(Virtual Memoty)

주기억 장치의 부족한 용량을 해결하기 위해 보조기억 장치를 주기억 장치처럼 사용하는 기법이다.
가상 기억 장치의 일반적인 구현 방법에는 프로그램을 고정된 크기의 일정한 블록으로 나누는 페이징 기법과 가변적인 크기의 블록으로 나누는 세그멘테이션 기법이 있다.
페이징(Paging) 기법

가상 기억 장치에 보관된 프로그램과 주기억 장치의 영역을 동일한 크기로 나누어진 것이 페이징이다. 나눠진 프로그램을 동일하게 나눠진 주기억 장치의 영역에 적재시켜 실행하는 기법이다.
가상 기억 장치에서 주기억 장치로 주소를 조정(매핑)하기 위해 페이지의 위치 정보를 가진 페이지 맵 테이블이 필요하다.
페이지의 크기가 클수록 페이지 맵 테이블의 크기가 작아지고, 단편화가 증가하고, 디스크 접근 횟수가 감소하며, 전체 입/출력 시간이 감소한다.
세그먼테이션(Segmentation) 기법

가상 기억 장치에 보관된 프로그램을 다양한 크기로 나눈 후, 나눠진 프로그램을 주기억 장치에 적재시켜 실행하는 기법이다.
세그먼트(Segment) : 큰 프로그램을 보다 작은 프로그램으로 분할해서 하나의 논리적 단위로 묶어서 주기억 장치에 읽어들일 수 있는 최소 단위이다.
구역성(Locality)

프로세스가 실행되는 동안 일부 페이지만 집중적으로 참조되는 경향을 의미한다.
시간 구역성(Temporal Locality) : 순환(Looping), 스택(Stack), 부프로그램(Subprogram), 집계(Totaling) 등에 사용되는 변수 등이 있다.
공간 구역성(Spatial Locality) : 프로세스가 어떤 페이지를 참조했다면 이후 가상 주소 공간상 그 페이지와 인접한 페이지들을 참조할 가능성이 높음을 의미한다. 배열 순례(Array Traversal). 프로그램의 순차적 수행 등이 있다.

워킹 셋(Working Set)

운영체제의 가상 기억 장치 관리에서 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합이다.
스래싱(Thrashing)

하나의 프로세스가 작업 수행 과정에 수행하는 기억 장치 접근에서 지나치게 페이지 부재가 발생하여 프로세스 수행에 소요되는 시간보다 페이지 이동에 소요되는 시간이 더 커지는 현상이다.
페이지 부재(Page Fault)

참조할 페이지가 주기억 장치에 없는 현상이다.
59. 디스크 스케줄링
디스크 스케줄링(DiskScheduling)
사용할 데이터가 디스크의 여러 곳에 저장되어 있을때 데이터를 액세스하기 위해 디스크 헤드의 이동 경로를 결정하는 기법이다.
디스크 스케줄링의 종류
FCFS(First Come First Service)

디스크 대기 큐에 먼저 들어온 트랙에 대한 요청을 먼저 서비스하는 기법이다.
FIFO(Fist In First Out) 방식이라고도 한다.
구현은 쉬운 반면 부하가 크면 응답 지연이 발생한다.
SSTF(Shortest Seek Time First)

탐색 거리가 가장 짧은 트랙에 대한 요청을 먼저 서비스하는 기법이다.
안쪽이나 바깥쪽 트랙이 가운데 트랙보다 서비스를 적게 받아 탐색 패턴이 편중된다.
응답 시간 편차로 인해 대화형에는 부적합하다.
SCAN

현재 헤드의 위치에서 진행 방향의 모든 요청을 서비스하면서 끝까지 이동한 후 반대 방향의 요청을 서비스하는 기법이다.
바깥쪽 트랙이 안쪽 트랙보다 서비스를 적게 받게 된다.
C-SCAN(Circular SCAN)

헤드가 항상 바깥쪽에서 안쪽으로 움직이며 모든 요청을 서비스하면서 끝까지 이동한 후 다시 바깥쪽에서 안쪽으로 이동하면서 요청을 서비스하는 기법이다.
N-step SCAN

어떤 방향의 진행이 시작될 당시에 대기 중이던 요청들만 서비스하고, 진행 도중 도착한 요청들은 한데 모아서 다음의 반대 방향 진행 때 최적으로 서비스하는 기법이다.

에센바흐(Eschenbach) 스케줄링

헤드가 진행하는 과정에서 각 실린더에 대해 한 번의 디스크팩 회전 시간 동안만 입/출력 요구들을 처리하는 기법이다. 즉, 한 회전 동안 서비스를 받지 못하는 요구들에 대한 처리는 다음으로 미루는 기법이다.

60. 정보 관리

파일 시스템

파일 시스템(File System)의 개념

파일(File)은 연관된 데이터들의 집합이다.
파일은 각각의 고유한 이름을 갖고 있다.
파일은 주로 보조 기억 장치에 저장하여 사용한다.
파일 시스템은 보조 기억 장치와 그 안에 저장된 파일을 관리하는 시스템이다.

파일 시스템의 기능

사용자가 파일을 생성, 수정, 제거할 수 있도록 해준다.
파일에 대한 여러 가지 접근 제어 방법을 제공한다.
사용자와 보조 기억 장치 사이에서 인터페이스를 제공한다.
정보의 백업(Backup) 및 복구(Recovery) 기능을 제공한다.
정보의 암호화(Encryption) 및 해독(Decryption)기능을 제공한다.
적절한 제어 방식을 통해 타인의 파일을 공동으로 사용할 수 있도록 해준다.

파일 디스크립터(File Descriptor)의 개념

파일을 관리하기 위해 필요한 파일에 대한 정보를 갖고 있는 제어 블록이다.
파일 제어 블록(FCB : File Control Block)이라고도 한다.
파일마다 독립적으로 존재하며, 시스템에 따라 다른 구조를 가질 수 있다.
대개 보조 기억 장치에 저장되어 있다가 해당 파일이 열릴(Open) 때 주기억 장치로 옮겨진다.
파일 시스템이 관리하므로 사용자가 직접 참조할 수 없다.

파일 디스크립터의 내용*

파일의 구조, 유형
파일의 크기, 이름
파일의 생성 시간, 수정 시간
파일에 대한 접근 횟수
포조 기억 장치 정보, 접근 제어 정보
파일 구조

파일 구조의 종류

순차 파일(Sequential File)
레코드들이 논리적인 순서에 따라 물리적인 연속 공간에 순차적으로 저장되는 파일 구조이다.
주기적으로 처리하는 경우에 시간적으로 속도가 빠르며, 처리 비용이 절감된다.
순차적으로 실제 데이터만 저장되므로 기억 공간의 활용률이 높다.
특정 레코드를 검색할 때, 순차적 검색을 하므로 검색 효율이 낮다.
색인 순차 파일(Indexed Sequential File)
키 값에 따라 순차적으로 정렬된 데이터를 저장하는 데이터 구역(Data Area)과 이 구역에 대한 포인터를 가진 색인 구역(Index Area)으로 구성된 파일 구조이다.
순차 처리와 직접 처리가 모두 가능하다.
레코드의 삽입, 삭제, 갱신이 용이하다.
인덱스를 이용하여 해당 데이터 레코드에 접근하기 때문에 처리 속도가 랜덤 편성 파일보다 느리다.
인덱스를 저장하기 위한 공간과 오버플로우 처리를 위한 별도의 공간이 필요하다.
색인 순차 파일의 구성
기본 구역(Prime Area)
레코드가 기록되는 영역이다.
색인 구역(Index Area)
기본 구역의 레코드의 위치를 찾는 색인이 기록된 영역이다.
색인 구역의 구성
트랙 색인 구역(Track Index Area)
실린더 색인 구역(Cylinder Index Area)
마스터 색인 구역(Master index Area)
오버플로우 구역
기본 구역에 레코드를 삽입하지 못하는 오버플로우 처리를 위한 별도의 영역이다.
직접 파일(Direct File)
키에 일정한 함수를 적용하여 상대 레코드 주소를 얻고, 그 주소를 레코드에 저장하는 파일 구조이다.
해싱 등의 사상 함수를 사용하여 레코드 키(Record Key)에 의한 주소 계산을 통해 레코드를 접근할 수 있도록 구성된다.
디렉토리 구조

1 단계 디렉토리 구조
같은 디렉토리에 시스템이 보관된 모든 파일 정보를 포함하는 구조이다.

모든 파일들이 유일한 이름을 가진다.
2 단계 디렉토리 구조
각각의 사용자에 대한 MFD 와 각 사용자별로 만들어지는 UFD 로 구성된다.
MFD 는 각 사용자의 이름이나 계정번호 및 UFD 를 가리키는 포인터를 갖고 있으며, UFD 는 오직 한 사용자가 갖고 있는 파일들에 대한 파일 정보만 갖고 있다.
트리 디렉토리 구조
UNIX 에서 사용하는 디렉토리 구조이다.
각 디렉토리는 서브 디렉토리나 파일을 가질 수 있다.
디렉토리의 생성과 파괴가 비교적 용이하다.
디렉토리의 탐색은 포인터를 사용하며, 절대 경로명과 상대 경로명을 사용한다.
비순환 그래프 디렉토리 구조
부디렉토리의 공동 사용이 가능하다.
디스크 공간을 절약할 수 있다.
하나의 파일이나 디렉토리가 여러 개의 경로 이름을 가질 수 있다.
공유하고 있는 파일 제거 시 떨어진 포인터(Dangling Pointer) 문제가 발생할 수 있다.
일반적인 그래프 디렉터리 구조
사이클이 허용되고 불필요한 파일 제거를 위해 참조 카운터가 필요한 디렉토리 구조이다.
61. 분산 운영체제
다중 처리기
다중 처리기(Multi-Processor)의 개념

하나의 시스템에 2 개 이상의 프로세서를 가지고 동시에 여러 개의 작업을 처리하는 장치이다.
프로세서 중 하나가 고장나도 다른 프로세서들에 의해 고장난 프로세서의 작업을 대신 수행하는 장애 극복이 가능하다.
프로세서 간의 통신은 공유 기억 장치를 통하여 입/출력 채널, 주변 장치들을 공유한다.
대칭적 다중 처리 방식과 비대칭적 다중 처리 방식이 있다.
성능 개선 목표 : 유연성, 신뢰성, 수행 속도
다중 처리기의 상호 연결 방법

시분할 공유 버스(Time Sharing Shared Bus)
프로세서, 기억 장치, 입/출력 장치 간에 하나의 버스 통신로만을 제공하는 구조이다.
어느 한 시점에 단지 하나의 전송만이 가능하다.
버스에 이상이 생기면 전체 시스템에 장애가 발생한다.
크로스바 교환 행렬(Crossbar Switch Matrix)
공유 버스 시스템에서 버스의 수를 기억 장치의 수만큼 증가시킨 구조이다.

두 개의 서로 다른 저장 장치를 동시에 참조할 수 있다.
하드웨어가 복잡해지는 단점이 있다.
하이퍼 큐브(Hyper Cube)
10 개 이상의 프로세서를 병렬로 동작시키는 구조이다.
하나의 프로세서에 연결되는 다른 프로세서의 수(연결점)가 n 개의 경우 총 2n 개의 프로세서가 필요하다.
다중 포트 메모리(Multiport Memoty)
하나의 프로세서에 하나의 버스가 할당되어 버스를 이용하려는 프로세서 간 경쟁이 적은 구조이다.
다중 처리기 운영체제의 구조

주/종(Master/Slave)처리기
하나의 주 프로세서와 나머지 종 프로세서로 구성된다.
주 프로세서만이 운영체제를 수행한다.
주 프로세서는 입/출력과 연산을 수행한다.
종 프로세서는 입/출력 발생 시 주 프로세서에게 서비스를 요청한다.
비대칭 구조를 갖는다.
분리 수행(Separate Execution) 처리기
주/종 처리기의 비대칭성을 보완하여 각 프로세서가 별도의 운영체제를 가진다.
프로세서별 자신만의 파일 및 입/출력 장치를 제어한다.
프로세서별 인터럽트는 독립적으로 수행된다.
한 프로세서의 장애는 전 시스템에 영향을 미치지 않는다.
대칭적(Symmetric) 처리기
모든 프로세서가 하나의 운영체제를 공유해서 수행한다.
가장 복잡하지만 가장 강력한 구조이다.
여러 개의 프로세서가 동시에 수행될 수 있다.
다중 처리기의 구조

약결합 시스템(Loosely Coupled System)
둘 이상의 시스템을 통신 링크를 이용하여 연결한 시스템이다.
각 시스템마다 별도의 운영체제를 가진다.
각 프로세서마다 독립된 메모리를 가진다.
프로세스 간의 통신은 메시지 전달이나 원격 프로시저 호출을 통하여 이루어진다.
분산 처리 시스템이라고도 한다.
강결합 시스템(Tightly Coupled System)
하나의 운영체제가 모든 처리기와 시스템 하드웨어를 제어한다.

프로세서 간 통신은 공유 메모리를 통하여 이루어진다.
메모리에 대한 프로세서 간의 경쟁 최소화가 고려되어야 한다.
분산 처리 시스템
분산 처리 시스템(Distributed Processing System)의 개념

여러 대의 컴퓨터들에 의해 작업한 결과를 통신망을 이용하여 상호 교환할 수 있도록 연결되어 있는 시스템으로 시스템의 점진적 확장이 용이하다.
투명성(Transparency)

분산 처리 운영체제에서 구체적인 시스템 환경을 사용자가 알 수 없도록 하며, 또한 사용자들로 하여금 이에 대한 정보가 없어도 원하는 작업을 수행할 수 있도록 지원하는 개념이다.
투명성의 종류
위치(Location) 투명성 : 하드웨어와 소프트웨어의 물리적 위치를 사용자가 알 필요가 없다.
이주(Migration) 투명성 : 사용자나 응용 프로그램의 동작에 영향을 받지 않고 자원들을 한 곳에서 다른 곳으로 이동할 수 있다.
복제(Replication) 투명성 : 사용자에게 통지할 필요 없이 시스템 안에 파일들과 자원들의 부가적인 복사를 자유로이 할 수 있다.
병행(Concurrency) 투명성 : 다중 사용자들이 자원들을 자동으로 공유할 수 있다.
분산 운영체제 구조

성형(Star) 구조
모든 사이트는 하나의 호스트에 직접 연결된 구조이다.
중앙 컴퓨터 장애 시 모든 사이트 간 통신이 불가능하다.
통신 시 최대 두 개의 링크만 필요하고 통신 비용이 저렴하다.
링형(Ring) 구조
각 사이트는 정확히 다른 두 사이트와 물리적으로 연결된 구조이다.
정보 전달 방향은 단방향 또는 양방향일 수 있다.
기본 비용은 사이트의 수에 비례한다.
기본 비용은 사이트의 수에 비례한다.
메시지가 링을 순환할 경우 통신비용은 증가한다.
다중 접근 버스(Multi Access Bus)구조
모든 사이트는 공유 버스에 연결된 구조이다.
기본 비용은 사이트 수에 비례한다.

사이트의 고장은 다른 사이트 간의 통신에 영향을 주지 않지만, 링크의 고장은 전체 시스템에 영향을 준다.
사이트의 추가와 삭제가 용이하다.
계층 연결
각 사이트들이 트리(Tree) 형태로 연결된 구조이다.
상위 사이트 장애 시 하위 사이트들은 통신이 불가능하다.
완전 연결(Fully Connection) 구조
모든 사이트는 다른 모든 사이트와 직접 연결된 구조이다.
사이트 간의 연결은 여러 회선이 존재하므로 신뢰성이 높다.
사이트 간의 메시지 전달이 매우 빠르다.
하나의 링크가 고장나더라도 통신은 단절되지 않는다.
사이트 설치 시 소요되는 기본 비용은 많이 든다.

62. UNIX

UNIX 의 개요
UNIX 의 특징

시분할(Time-sharing) 시스템을 위해 설계된 대화식 운영체제이다.
소스가 공개된 개방형 시스템(Open System)이다.
트리 구조의 파일 시스템을 갖는다.
멀티 유저(Multi-user), 멀티태스킹(Multi-tasking)을 지원한다.
하나 이상의 작업에 대하여 백그라운드에서 수행이 가능하다.
90% 이상이 고급 언어인 C 로 구성되어 있어서 이식성이 높다.
UNIX 시스템의 구성

커널(Kernel)
UNIX 시스템의 가장 핵심적인 부분이다.
프로세스 관리, 메모리 관리, 파일 관리, 입/출력 관리 등의 기능을 수행한다.
쉘(Shell)
사용자가 지정한 명령들을 해석하여 커널로 전달하는 명령어 해석기이다.
시스템과 사용자 간의 인터페이스를 담당한다.
종류 : C Shell, Bourn Shell, Korn Shell 등
유틸리티(Utility)
사용자의 편의를 위한 프로그램이다.
종류 : 편집기, 컴파일러, 일터프리터 등
UNIX 파일 시스템의 구조

부트 블록(Boot Block)

부팅에 필요한 코드를 저장하고 있는 블록이다.
슈퍼 블록(Super Block)
전체 파일 시스템에 대한 정보를 저장하고 있는 블록이다.
I-node 블록(Index Node Block)
각 파일에 대한 정보를 저장하고 있는 블록이다.
파일 소유자의 식별번호, 파일 크기, 파일의 최종 수젖ㅇ시간, 파일 링크 수 등의 내용을 가지고 있다.
데이터 블록(Data Block)
실제 데이터를 저장하고 있는 블록이다.
UNIX 명령어
시스템 관련 명령어

login : UNIX 시스템에 접속한다.
logout : UNIX 시스템 접속을 종료한다.
finger : 시스템에 등록된 사용자의 정보를 표시한다.
who : 현재 로그인 중인 각 사용자에 관한 정보를 표시한다.
ping : 네트워크상의 문제를 진단한다.
fsck : 파일 시스템의 무결성을 검사한다.
mount : 기존 파일 시스템에 새로운 파일 시스템을 서브 디렉토리에 연결한다.
unmae : 현시 시스템 정보를 확인하는 명령어이다.(옵션 -a : 시스템 모든 정보 출력)
프로세스 관련 명령어

fork : 새로운 프로세스를 생성한다.

exec : 새로운 프로세스를 수행한다.

exit : 프로세스 수행을 종료한다.

wait : 자식 프로세스 중 하나가 종료될 때까지 부모 프로세스를 임시로 중지시킨다.

kill : 현재 실행 중인 프로세스를 종료하거나 한 줄 전체를 지운다.

ps : 현재 실행 중인 프로세스의 상태를 표시한다.

getpid : 자신의 프로세스 아이디를 구한다. -getppid : 부모 프로세스 아이디를 구한다.

디렉토리 관련 명령어

pwd : 현재 작업 중인 디렉토리의 경로를 표시한다.

ls : 현재 디렉토리 내의 모든 파일을 표시한다.

mkdir : 디렉토리를 생성한다.

rd : 디렉토리를 삭제한다.

cd : 디렉토리의 위치를 변경한다.

파일 관련 명령어

create : 파일을 생성한다.
open : 파일을 사용 가능한 상태로 준비시킨다.
cp : 파일을 복사한다.
rm : 파일을 삭제한다.
mv : 파일의 이름을 바꾼다.
cat : 파일의 내용을 화면에 표시한다.(cat/etc/release : 파일의 사용 권한을 지정한다.)
chmod : 파일의 사용 권한을 지정한다.
chown : 파일의 소유자를 변경한다.
UNIX 환경 변수
환경 변수(Environment Variables)의 개념

쉘(Shell)이 프로그램들 사이에서 값을 전달해 주는 역할을 하는 변수이다.
프로세스가 컴퓨터에 동작하는 방식에 영향을 미치는 값들의 집합이다.
기본적으로 환경 변수는 대문자를 사용한다.
환경 변수 관련 명령어

env : 전역 환경 변수를 설정하거나 출력한다.
set : 사용자 환경 변수를 설정한다.
printenv : 현재 설정되어 있는 환경 변수의 값을 모두 출력한다.
echo : 특정 환경 변수의 값을 출력한다.
setenv : 환경 변수의 값을 설정한다.
BASH Shell

LINUX, MAC OSX 등 다양한 운영체제에서 사용되며 LINUX 표준 쉘이다.
LINUX 에서 환경 변수를 설정하는 명령어에는 env, set, export 등이 있다.
env : 전역 변수 설정, 조회, 삭제

set : 사용자 환경 변수 설정 및 조회
export : 사용자 환경 변수 전역 변수로 설정
declare : 변수 타입을 설정
63. OSI 7 계층과 오류 제어 방식
OSI 참조 모델
OSI(Open Systems Interconnection) 참조 모델의 개념

국제표준화기구(ISO)에서 개발한 모델이다.
컴퓨터 네트워크에서 여러 시스템이 데이터를 주고 받고 서로 연동할 수 있는 표준화된 인터페이스를 제공하기 위해 프로토콜을 기능별로 나눈 것이다.
시스템 연결을 위한 표준 개발을 위하여 공통적인 기법을 제공한다.
시스템 간의 정보 교환을 위한 표준 설정을 가질 수 있도록 한다.
각 계층에 대해 서로 표준을 생산적으로 발전시킬 수 있도록 개념적, 기능적인 골격을 제공하는 역할을 한다.
일반적으로 OSI 7 계층이라고 한다.
OSI 참조 모델에서 계층을 나누는 목적

시스템 간의 통신을 위한 표준 제공
시스템 간의 정보 교환을 하기 위한 상호 접속점의 정의
관련 규격의 적합성을 조성하기 위한 공통적인 기반 조성
OSI 7 계층
물리 계층(Physical Layer)
물리적인 장치와 인터페이스가 전송을 위해 필요한 기계적, 전기적, 기능적, 절차적 기능을 정의하는 계층이다.
장치와 전송 매체 간의 인터페이스 특성 규정, 전송 매체의 유형 규정, 전송로의 연결, 유지 및 해제를 담당한다.
프로토콜 종류 : RS-232C, V.24, X.21
데이터 링크 계층(Data Link Layer)
인접한 두 개의 통신 시스템 간에 신뢰성 있는 효율적인 데이터를 전송하는 계층이다.
링크의 설정과 유지 및 종료를 담당한다.
전송 데이터의 흐름 제어, 프레임 동기, 오류 제어 등을 수행한다.
링크의 효율성을 향상시킨다.
프로토콜 종류 : HDLC, PPP, LLC, LAPB, LAPD, ADCCP
네트워크 계층(Network Layer)
통신망을 통해 패킷을 목적지까지 전달하는 계층.
경로 설정 및 네트워크 연결 관리를 수행.

과도한 패킷 유입에 대한 폭주 제어 기능을 한다.
프로토콜 종류 : X.25, IP, ICMP, IGMP
전송 계층(Transport Layer)
통신 종단 간(End-to-End) 신뢰성 있고 효율적인 데이터를 전송하는 계층이다.
투명한 데이터 전송을 제공한다.
에러 제어 및 흐름 제어를 담당한다.
프로토콜 종류 : TCP, UDP
세션 계층(Session Layer)
프로세스 간에 대한 연결을 확립, 관리, 단절시키는 수단을 제공한다.
논리적 동기 제어, 긴급 데이터 전송, 통신 시스템 간의 회화 기능 등을 제공한다.
표현 계층(Presentation Layer)
응용 간의 대화 제어(Dialogue Control)을 담당한다.
응용 계층과 세션 계층 사이에서 데이터 변환을 담당한다.
정보의 형식 설정, 암호화, 데이터 압축, 코드 변환, 문맥 관리 등의 기능을 수행한다.
긴 파일 전송 중에 통신 상태가 불량하여 트랜스포트 연결이 끊어지는 경우 처음부터 다시 전송하지 않고 어디까지 전송이 진행되었는지를 나타내는 동기점을 이용하여 오류를 복구한다.
응용 계층(Application Layer)
사용자에게 서비스를 제공한다.
응용 프로세스와 직접 관계하여 일반적인 응용 서비스를 수행한다.
프로토콜 종류 : HTTP, FTP, SMTP, Telnet, DNS
오류 제어 방식
자동 반복 요청(ARQ : Automatic Repeat reQuest)

통신 경로에서 오류 발생 시 수신측은 오류의 발생을 송신측에 통보하고, 송신측은 오류가 발생한 프레임을 재전송하는 오류 제어 방식이다.
정지-대기 ARQ(Stop-and-Wait-Arq)

송신측이 한 블록 전송 후 수신측에서 오류의 발생을 점검 후 에러 발생 유무 신호(ACK/NAK 신호)를 보내올 때까지 기다리는 방식이다.
수신측에서 에러 점검 후 제어 신호를 보내올 때까지 오버헤드가 효율면에서 가장 부담이 크다.
연속 ARQ(Continuous ARQ)

Go-Back-N ARQ : 수신측으로부터 NAK 수신 시 오류 발생 이후의 모든 블록을 재전송하는 방식이다.

선택적 재전송 ARQ(Selective-Repeat ARQ) : 수신측으로부터 NAK 수신 시 오류가 발생한 블록만 재전송하는 방식이다.
적응적 ARQ(Adaptive ARQ)

채널 효율을 최대로 하기 위해 데이터 블록의 길이를 채널의 상태에 따라 동적으로 변경하는 방식이다.

64. TCP/IP 프로토콜

TCP/IP 프로토콜의 개념

TCP/IP(Transmission Control Protocol/Internet Protocol)

인터넷에 연결된 서로 다른 기종의 컴퓨터 간에 데이터 송/수신이 가능하도록 도와주는 표준 프로토콜이다.
TCP 프로토콜과 IP 프로토콜의 결합적 의미로서 TCP 가 IP 보다 상위층에 존재한다.
접속형 서비스, 전이중 전송 서비스, 신뢰성 서비스를 제공한다.
네트워크 환경에 따라 여러 개의 프로토콜을 허용한다.
TCP 프로토콜의 기본 헤더 크기는 20byte 이고 60byte 까지 확장 가능하다.
OSI 표준 프로토콜과 가까운 네트워크 구조를 가진다.
OSI 7 계층 : 물리, 데이터, 네트워크, 전송, 세션, 표현, 응용 계층
TCP/IP 4 계층 : 링크, 인터넷, 전송, 응용 계층
TCP(Transmission Control Protocol)

OSI 7 계층의 전송 계층의 역할을 수행한다.
서비스 처리를 위해 Multiplexing 과 DeMultiplexing 을 이용한다.
전이중 서비스와 스트림 데이터 서비스를 제공한다.
IP(Internet Protocol)

OSI 7 계층의 네트워크 계층에 해당하며 비신뢰성 서비스를 제공한다.
TCP/IP 의 구조
링크 계층(Link Layer)

프레임을 송/수신한다.
프로토콜의 종류 : Ethernet, IEEE 802, HDLC, X.25, RS-232C 등
인터넷 계층(Internet Layer)

주소 지정, 경로 설정을 제공한다.
네트워크 계층이라고도 한다.
프로토콜 종류 : IP, ICMP, IGMP, ARP, RARP 등
IP(Internet Protocol)
비연결형 및 비신뢰성 전송 서비스를 제공한다.

라우팅과 단편화 기능을 수행한다.
데이터그램(Datagram)이라는 데이터 전송 형식을 가진다.
각 데이터그램이 독립적으로 처리되고 목적지까지 다른 경로를 통해 전송될 수 있어 데이터그램은 전송 순서와 도착 순서가 다를 수 있다.
비연결성이기 때문에 송신지가 여러 개인 데이터 그램을 보내면서 순서가 뒤바뀌어 도달할 수 있으며 IP 프로토콜의 헤더 길이는 최소 20~60byte 이다.
ICMP(Internet Control Message Protocol)
IP 프로토콜에서는 오류 보고와 수정을 위한 메커니즘이 없기 때문에 이를 보완하기 위해 설계된 프로토콜이다.
메시지는 크게 오류 보고(Error-Reporting) 메시지와 질의(Query) 메시지로 나눌 수 있다.
메시지 형식은 8 바이트의 헤더와 가변 길이의 데이터 영역으로 분리된다.
에코 메시지는 호스트가 정상적으로 동작하는지를 결정하는데 사용할 수 있다.
수신지 도달 불가 메시지는 수신지 또는 서비스에 도달할 수 없는 호스트를 통지하는데 사용한다.
IGMP(Internet Group Management Protocol)
시작지 호스트에서 여러 목적지 호스트로 데이터를 전송할 때 사용되는 프로토콜이다.
멀티캐스트 그룹에 가입한 네트워크 내의 호스트를 관리한다.
ARP 20.9(Address Resolution Protocol)
논리 주소(IP 주소)를 물리 주소(MAC 주소)로 변환하는 프로토콜이다.
네트워크에서 두 호스트가 성공적으로 통신하기 위해 각 하드웨워의 물리적인 주소 문제를 해결해 줄 수 있다.
RARP(Reverse Address Resolution Protocol)
호스트의 물리 주소(MAC 주소)로부터 논리 주소(IP 주소)를 구하는 프로토콜이다.
IP 호스트가 자신의 물리 네트워크 주소(MAC)는 알지만 IP 주소를 모르는 경우, 서버에게 IP 주소를 요청하기 위해 사용한다.
전송 계층(Transport Layer)

호스트 간 신뢰성 있는 통신을 제공한다.
프로토콜 종류 : TCP, UDP
TCP 20.8(Transmission Control Protocol)
신뢰성 있는 연결 지향형 전달 서비스를 제공한다.
순서 제어, 에러 제어, 흐름 제어 기능을 제공한다.
전이중 서비스와 스트림 데이터 서비스를 제공한다.
메시지를 캡슐화(Encapsulation)와 역캡슐화(Decapsulation)한다.
서비스 처리를 위해 다중화(Multiplexing)와 역다중화(Demultiplexing)를 이용한다.
UDP 20.9(User Datagram Protocol)
비연결형 및 비신뢰성 전송 서비스를 제공한다.

흐름 제어나 순서 제어가 없어 전송 속도가 빠르다.
수신된 데이터의 순서 재조정 기능을 지원하지 않는다.
복구 기능을 제공하지 않는다.

65. IP 주소

IPv4(Internet Protocol version 4)

IPv4 의 개념

32 비트 길이의 IP 주소이다.
주소의 각 부분을 8 비트씩 4 개로 나눠서 10 진수로 표현한다.
IP 주소 = 네트워크 주소(Netid) + 호스트 주소

IPv4 주소 체계

클래스 A
0.0.0.0 ~ 127.255.255.255
기본 서브넷 마스크 : 255.0.0.0
국가나 대형 통신망에서 사용한다.

클래스 B
128.0.0.0 ~ 191.255.255.255
기본 서브넷 마스크 : 255.255.0.0
중대형 통신망에서 사용한다.

클래스 C
192.0.0.0 ~ 223.255.255.255
기본 서브넷 마스크 : 255.255.255.0
소규모 통신망에서 사용한다.

클래스 D
224.0.0.0 ~ 239.255.255.255
멀티캐스트용으로 사용한다.

클래스 E
240.0.0.0 ~ 255.255.255.255
실험용으로 사용한다.

서브넷 마스크(Subnet Mask)

네트워크를 작은 내부 네트워크로 분리하여 효율적으로 네트워크를 관리하기 위한 수단이다.
서브넷 마스크는 32 비트의 값으로 IP 주소를 네트워크와 호스트 IP 주소를 구분하는 역할을 한다.
네트워크 ID 에 해당하는 모든 비트를 1 로 설정하며, 호스트 ID 에 해당하는 모든 비트를 0 으로 설정한다.

CIDR 표기 형식 : 10 진수의 IP/네트워크 ID 의 1 비트의 개수
IPv6(Internet Protocol version 6)
IPv6 의 개념

IPv4 의 주소 부족 문제를 해결하기 위해 개발되었다.
128 비트 길이의 IP 주소이다.
16 비트씩 8 개의 필드로 분리 표기된다.
IPv6 의 장점

인증 보안 기능을 포함하고 있어 IPv4 보다 보안성이 강화되었다.
IPv6 확장 헤더를 통해 네트워크 기능 확장이 용이하다.
임의의 크기의 패킷을 주고받을 수 있도록 패킷 크기 제한이 없다.
멀티미디어의 실시간 처리가 가능하다.
자동으로 네트워크 환경 구성이 가능하다.
주소체계는 유니캐스트(Unicast), 애니캐스트(Anycast), 멀티캐스트(Multicast) 등 세 가지로 나뉜다.
IPv6 통신 방식 20.6

유니캐스트(Unicast)
하나의 호스트에서 다른 하나의 호스트에게 전달하는 1:1 통신 방식이다.
애니캐스트(Anycast)
하나의 호스트에서 그룹 내의 가장 가까운 곳에 있는 수신자에게 전달하는 '1: 가장 가까운 1' 통신 방식이다.
멀티캐스트(Multicast)
하나의 호스트에서 네트워크상의 특정 그룹 호스트들에게 전달하는 1:N 통신 방식이다.
IEEE 802 의 표준 규격

802.1 : 상위 계층 인터페이스
802.2 : 논리 링크 제어(LLC)
802.3 : CSMA/CD
802.4 : 토큰 버스(Token Bus)
802.5 : 토큰 링(Token Ring)
802.6 : MAN
802.8 : 고속 이더넷(Fast Ethernet)
802.11: 무선 LAN
802.15 : 블루투스
CSMA/CA

무선 랜에서 데이터 전송 시, 매체가 비어있음을 확인한 뒤 충돌을 회피하기 위해 임의 시간을 기다린 후 데이터를 전송하는 방법이다.
에트워크에 데이터의 전송이 없는 경우라도 동시 전송에 의한 충돌에 대비하여 확인 신호를 전송한다.
CSMA/CD(Carrier Sensing Multiple Access/Collision Detection)

전송 중에 충돌이 감지되면 패킷의 전송을 즉시 중단하고 충돌이 발생한 사실을 모든 스테이션들이 알 수 있도록 간단한 통보 신호를 송신한다.
스테이션의 수가 많아지면 충돌이 많아져서 효율이 떨어진다.
어느 한 기기에 고장이 발생해도 다른 기기의 통신에 전혀 미치지 않는다.