

4과목

Chapter 1. 서버프로그램 구현

070 서버개발 프레임워크 ★

• 서버개발 프레임워크의 특징

특징	설명
모듈화 (Modularity)	- 캡슐화를 통해 모듈화를 강화해 설계의 변경에 따른 영향을 최소화함 - 유지 보수가 용이함
재사용성 (Reusability)	반복적으로 사용되는 모듈들을 재사용 가능한 모듈로 제공함으로써 생산성 향상, 품질 보증함
확장성 (Extensibility)	다형성을 통한 인터페이스 확장이 가능함
제어 반전 (Inversion of Control)	프레임워크에서 제어를 하며 외부 라이브러리의 코드를 호출해 이용함

071 보안 취약성 식별

• 소프트웨어 취약점

버퍼 오버플로 (Buffer Overflow)	메모리를 다루는 데 오류가 발생하여 잘못된 동작을 하는 프로그램 취약점
허상 포인터 (Dangling Pointer)	메모리 보안 위반의 특별한 경우로 적절한 타입의 유효한 객체를 가리키고 있지 않은 포인터를 말함
FTP 바운스 공격 (FTP Bounce Attack)	FTP 프로토콜 구조의 허점을 이용한 공격
SQL 삽입 (SQL Injection)	웹 URL 등에 SQL 문법 삽입 등을 통한 에러를 이용한 공격기법
디렉토리 접근 공격 (Directory Traversal)	웹 루트 이외의 디렉토리에 있는 데이터에 접근하여 특정 동작을 수행함
포맷 스트링 버그 (Format String Attack)	포맷팅을 수행하는 printf() 같은 특정한 함수에서 검사되지 않은 사용자 입력을 통한 공격
코드 인젝션 (Code Injection)	유효하지 않은 데이터를 실행하는 공격

Chapter 2. 프로그래밍 언어 활용

072 데이터 타입과 변수★★★

• 데이터 타입의 유형

데이터 타입 유형	설명	언어별 사용 타입		
		C언어	JAVA	파이썬
정수형 (Integer)	- 정수(소수점이 없는 숫자)를 저장할 때 사용 - 1, 2, -1	int, short	int, short, long	int
부동 소수형 (Float Point)	- 실수(소수점이 있는 숫자)를 저장할 때 사용 - 1.0, -3.14, 4.24e-10	float, double	float, double	float, complex
문자형 (Character)	- 문자 하나를 저장할 때 사용 - 작은 따옴표(' ')안에 표기 - 'A', '1', '%'	char	char	string
문자열 (String)	- 여러 개의 문자를 저장할 때 사용 - 큰 따옴표(" ")안에 표기 - "ABC", "안녕하세요", "A+B"	char 배열	string	string
논리형 (Boolean)	- 조건의 참(True), 거짓(False) 여부를 판단하고자 할 때 사용	미지원	boolean	boolean

• 변수 작성 규칙

변수 작성 규칙	사용 가능 예시	사용 불가능 예시
영문 대문자/소문자, 숫자, 밑줄(_) 사용	A, abc, a1004, _a	!a, ?a, %a
첫글자는 영문자, 밑줄(_) 사용, 숫자 사용 불가	ABC123, _a123	1, 1004, 3abc
변수의 중간에 공백이나 특수문자(밑줄 제외) 사용 불가	A_123	A/BC, H.ellow, A 123, text-color
예약어 변수 사용 불가		if, else, while, for, break, int, short, long, string

073 연산자★★★

• 연산자 종류

연산자 종류	기호	설명
산술 연산자	+, -, *, /, %	더하기, 빼기, 곱하기, 나누기, 나머지 연산자
증감 연산자	++, --	변수의 값을 증가하거나 감소시키는 연산자
관계 연산자	>, <, >=, <=, ==, !=	두 변수의 관계를 비교하여 참(True), 거짓(False)을 반환하는 연산자
논리 연산자	!, &&,	두 개의 논리값을 연산하여 참(True), 거짓(False)을 반환하는 연산자
비트 연산자	&, , ^, ~,	비트 단위(0, 1)로 연산하는 연산자
시프트 연산자	<<, >>	비트 값을 왼쪽, 오른쪽으로 이동하여 연산하는 연산자
대입 연산자	+=, -=, *=, /=, %= <<=, >>=	변수에 값을 대입하는 연산자
삼항 연산자 (조건 연산자)	(조건)?(참):(거짓):	조건이 참일 경우, (참)부분을 반환하고, 조건이 거짓인 경우, (거짓)부분을 반환하는 연산자

• 산술 연산자

연산자 종류	기호	설명
산술 연산자	+	더하기
	-	빼기
	*	곱하기
	/	나누기(나누기 후 몫)
	%	나머지(나누기 후 나머지 값)

• 증감 연산자

연산자 종류	기호	설명
증감 연산자	++A, --A	전치 연산자로, 변수 A의 값을 1씩 증가 또는 감소시킨 후 연산에 사용
	A++, A--	후치 연산자로, 변수 A를 연산에 사용한 후, 1씩 증가 또는 감소

• 관계 연산자

연산자 종류	기호	설명
관계 연산자	>	두 값 중 왼쪽의 값이 오른쪽의 값보다 크면 참
	<	두 값 중 왼쪽의 값이 오른쪽의 값보다 작으면 참
	>=	두 값 중 왼쪽의 값이 오른쪽의 값보다 크거나 같으면 참
	<=	두 값 중 왼쪽의 값이 오른쪽의 값보다 작거나 같으면 참
	==	두 값이 같으면 참
	!=	두 값이 다르면 참

• 논리 연산자

연산자 종류	기호	설명
논리 연산자	&&	논리값이 모두 참이면, 참을 반환하고, 그렇지 않은 경우, 거짓 반환(AND 조건)
		논리값 중 하나 이상이 참이면, 참을 반환하고, 그렇지 않으면 거짓 반환(OR 조건)
	!	- NOT 조건으로 0 이외의 수 - 앞에 ! 붙이면 거짓(0) 반환

• 비트 연산자와 시프트 연산자

연산자 종류	기호	설명
비트 연산자	&	모든 비트값이 1일 때, 1을 반환, 아닌 경우, 0 반환(AND 조건)
		비트값 중 하나 이상이 1인 경우, 1을 반환, 아닌 경우 0 반환(OR 조건)
	^	- 모든 비트값이 같으면 0을 반환, 하나라도 다른 경우 1 반환 - XOR 조건으로 bit끼리 더하고 올림을 제거함
	~	비트의 값을 반대로 부정하는 연산
시프트 연산자	<<	비트를 왼쪽으로 이동, 빈 칸에는 0을 채움
	>>	비트를 오른쪽으로 이동

• 대입 연산자

연산자 종류	기호	설명
대입 연산자	=	왼쪽의 변수에 오른쪽 값을 대입
	+=	왼쪽 변수 값과 오른쪽 값을 더하고 왼쪽 변수에 대입
	-=	왼쪽 변수 값에 오른쪽 값을 빼고 왼쪽 변수로 대입
	*=	왼쪽 변수 값에 오른쪽 값을 곱하여 왼쪽 변수로 대입
	/=	왼쪽 변수 값에 오른쪽 값을 나누어 왼쪽 변수로 대입
	%=	왼쪽 변수 값에 오른쪽 값의 나머지를 구한 후, 왼쪽 변수로 대입
	<<=	왼쪽 변수 값에 오른쪽 값만큼 왼쪽으로 비트 이동 후, 왼쪽 변수로 대입
	>>=	왼쪽 변수 값에 오른쪽 값만큼 오른쪽으로 비트 이동 후, 왼쪽 변수로 대입

- 삼항 연산자(조건 연산자)

연산자 종류	기호	설명
삼항 연산자	(조건) ? (참) : (거짓);	조건이 참인 경우, (참)의 수식 실행, 변수 반환, 조건이 거짓인 경우, (거짓)의 수식, 변수 반환

074 데이터 입·출력 ★

- 표준 입력 함수와 표준 출력 함수

scanf(서식 문자열, 변수의 주소)	
<ul style="list-style-type: none"> - 서식 문자열: 입력받을 데이터의 자료형 - 변수의 주소: 데이터를 입력받을 변수의 주소로 &변수명으로 표기 	
printf(서식 문자열, 변수)	
<ul style="list-style-type: none"> - 서식 문자열: 입력받을 데이터의 자료형 - 변수: 출력할 변수 	

- 서식 문자열 유형

유형	문자	설명
문자형	%c	문자 입·출력
	%s	문자열 입·출력
정수형	%d	정수형 10진수 입·출력
	%u	부호없는 정수형 10진수 입·출력
	%o	정수형 8진수 입·출력
	%x	정수형 16진수 입·출력
실수형	%f	소수점을 포함하는 실수 입·출력
	%e	지수형 실수 입·출력

- 이스케이프 문자

종류	의미	설명
\n	New Line	커서를 다음 줄 앞으로 이동(개행)
\t	Tab	커서를 일정 간격만큼 띄움
\b	Backspace	커서를 왼쪽으로 한 칸 이동

- JAVA의 입출력 함수

출력 함수	설명
System.out.printf(서식 문자열, 변수)	C언어 출력처럼 서식 문자열을 사용하여 출력
System.out.print()	변수의 값을 형식 없이 출력
System.out.println()	변수의 값을 형식 없이 출력 후 자동 개행

- 파이썬의 입출력 함수

출력 함수	설명
print(문자열)	문자열 출력 후 자동 개행
print(문자열, end='')	문자열 출력 후 자동 개행하지 않음

075 배열 및 포인터★★★

• 1, 2차원 배열 선언 방법

자료형 변수명[개수];	- 초깃값이 없는 경우 - 자료형: 배열에 저장할 자료의 형 지정 - 변수명: 사용할 배열의 이름으로 사용자가 임의로 지정 - 개수: 배열의 크기를 지정하는 것으로 생략 가능
자료형 변수명[개수] = {초깃값};	- 초깃값이 있는 경우
자료형 변수명[행개수][열개수]	- 자료형: 배열에 저장할 자료의 형 지정 - 변수명: 사용할 배열의 이름으로 사용자가 임의로 지정 - 행 개수: 배열의 행 크기 지정 - 열 개수: 배열의 열 크기 지정
자료형 변수명[행개수][열개수] = {초깃값}	- 초깃값이 있는 경우

076 명령문(1) - 조건문★★★

• C언어, JAVA if문

if (조건식){	if문의 조건식이 참일 경우 if 안에 있는 문장1 실행
문장1;	
}	
else if (조건문){	- if문의 조건이 거짓이면서 else if문의 조건이 참일 경우 else if 안에 있는 문장2 실행
문장2;	- else if는 여러 개 사용 가능
}	
else {	- else는 if문의 조건문이 거짓이고 여러 개의 else if 조건문이 모두 거짓일 때 else 안에 있는 문장3 실행 (else는 사용하지 않거나 한 번만 사용)
문장3;	
}	

• 파이썬 if문

if 조건식:	if의 조건식이 참이면 문장1 실행
문장1	
elif 조건식:	if 조건식은 거짓이고, elif의 조건식이 참이면 문장2 실행
문장2	
else:	if, elif 조건식이 모두 거짓이면 문장3 실행
문장3	

• C언어, JAVA if문의 삼항 연산자 표기법

if문 사용	삼항 연산자 사용
if (조건식){	(조건식)? 문장1 : 문장2
문장1;	
}	
else {	
문장2;	
}	

• 파이썬에서 if문의 삼항 연산자 표기법

if문 사용	삼항 연산자 사용
if 조건식: 문장1	문장1 if 조건식 else 문장2
else 조건식: 문장2	

• C언어, JAVA에서의 switch문

switch (식){	- switch 문의 식을 계산해서 일치하는 값을 가진 case 문장을 실행하고 없으면 default 문장을 실행 - break를 만나면 switch 문 탈출
case 값:	'switch(식)' 식을 계산한 값과 'case 값'이 같으면 진입하여 문장1 실행
문장1:	
break:	break를 만나면 switch 문 탈출 (break가 없으면 밑에 문장2도 실행)
default:	switch 문에 식이 어떠한 case의 값도 만족하지 않으면 default로 진입해 문장2 실행(default는 없어도 됨)
문장2:	
}	

077 명령문(2) - 반복문★★★★

• C언어, JAVA의 for문

for (초기식; 조건식; 증감식){	- for 문을 첫 번째 실행할 때, 초기식 확인, 조건식 확인 후 조건식이 참이면 반복문 내의 명령문 실행 - for 문을 두 번째 실행할 때부터는, 증감식 확인, 조건식 확인 후 조건식이 참이면 반복문 내의 명령문 실행 - 조건식이 거짓이 되면 for 문을 탈출
문장:	
}	

• 파이썬 for문

for 변수 in range(시작값, 끝값+1):	range에 (시작값)과 (끝값+1)로 정의할 경우 변수는 (시작값)부터 (끝값)까지 1씩 증가
문장	
for 변수 in range(반복횟수):	range에 (반복횟수)를 정의한 경우 0부터 (반복횟수-1)까지 변수가 1씩 증가
문장	

• while문

while(조건문){	- 조건문이 참이면 해당 분기를 반복해서 실행 - 조건문이 거짓이 되면 while 문을 탈출 - JAVA의 경우 while의 조건문이 boolean 타입이 아닌 경우 오류 발생 (즉, C언어의 경우 while(1) 등도 가능하지만 JAVA의 경우 불가)
문장:	
}	

• C언어, JAVA에서 do~while문

do{ 문장;	참, 거짓과 관련 없이 무조건 한 번은 실행
} while(조건문);	조건문이 참이면 해당 분기를 반복해서 실행하고, 조건문이 거짓이 되면 do~while문을 탈출

078 사용자 정의 함수와 클래스★★

• 사용자 정의 함수

언어	사용법 예시	설명
C언어	<pre>#include <stdio.h> int add(int, int); main() { int a = 3, b = 4; printf("%d", add(a, b)); }</pre>	<ul style="list-style-type: none"> - 프로토타입 자료형만 표시 - add 함수가 있음을 선언 - a와 b의 변수에 3과 4를 할당 - add 함수에 int형 파라미터 a, b를 할당해 호출 - 반환된 int형 값을 출력
	<pre>int add(int a, int b) { int sum; sum = a + b; return sum; }</pre>	<ul style="list-style-type: none"> - 함수명 앞부분에 반환할 자료형을 명시 - 반환할 값이 없는 경우 void - 입력받은 파라미터와 변수명을 정의 - 입력받은 변수 a와 b를 더한 값을 sum에 대입 - sum 값을 리턴
JAVA	<pre>public class Operator{ public static void main(String[] args) { int a = 3, b = 4; System.out.print(add(a, b)); } }</pre>	<ul style="list-style-type: none"> - JAVA의 main 함수 부분이다. - a와 b의 변수에 3과 4를 할당 - add 함수에 int형 파라미터 a, b를 할당해 호출
	<pre>public static int add(int a, int b) { int sum; sum = a + b; return sum; }</pre>	<ul style="list-style-type: none"> - 함수명(자바는 메소드) 앞에 static 선언 시 메모리에 자동으로 올라가서 Operator 클래스 객체를 생성하지 않고 호출 가능 - static은 선언 시 메모리에 자동으로 올려 객체를 생성하지 않고 호출 가능 - 반환할 변수형과 입력받은 파라미터값을 명시 - 입력받은 파라미터 a와 b를 더한 값을 반환
Python	<pre>def add(a, b): sum = a + b return sum a, b = 3, 4 print(add(a, b))</pre>	<ul style="list-style-type: none"> - 입력 파라미터 a, b를 명시하여 add의 함수를 선언 - 함수 선언 시 마지막에 ":"을 붙여야 함 - a와 b 변수에 3과 4를 할당 - add 함수에 a, b 파라미터를 할당해 호출 후 반환 값을 출력

• JAVA 접근제어자

접근 제어자	설명
public	모든 접근을 허용함
protected	같은 패키지(같은 폴더)에 있는 객체와 상속관계 객체들만 허용
default	같은 패키지(같은 폴더)에 있는 객체들만 허용
private	객체 내에서만 허용

• 클래스

언어	사용법 예시	설명
JAVA	<pre>public class Client { public int age; public String name; }</pre>	<ul style="list-style-type: none"> - Client의 클래스를 정의 (클래스명 대문자로 시작) - 클래스 내부에 선언한 변수를 필드(Field)라고 하며 객체의 데이터를 저장 - age와 name의 필드형과 필드명 정의 (필드명 소문자로 시작)
	<pre>Client(String name, int age){ this.name = name; this.age = age; }</pre>	<ul style="list-style-type: none"> - 생성자(Constructor)로 클래스 인스턴스 생성 시 초기화하여 실행 - this는 인수나 변수가 필드와 같은 이름 일 경우 필드를 구분하기 위해 사용 - this를 사용하면 필드를 의미
	<pre>public void clientInfo() { System.out.println("이름:" + name); System.out.println("나이:" + age); }</pre>	<ul style="list-style-type: none"> - 객체 내의 메소드(Method) - 객체의 동작을 정의
	<pre>public class Main { public static void main(String[] args) { Client client = new Client("홍길동", 50); client.clientInfo(); } }</pre>	<ul style="list-style-type: none"> - Client 클래스의 객체 client를 생성 - 생성자가 있는 경우 파라미터를 함께 전달 - 생성과 동시에 생성자 함수가 호출됨 - 생성자가 없는 경우 Client()로 생성 - client의 clientInfo 메소드 실행
Python	<pre>class Client: def __init__(self, name, age): self.name = name self.age = age def client_info(self): print('이름:', self.name) print('나이:', self.age)</pre>	<ul style="list-style-type: none"> - Client 클래스를 정의 - 파이썬에서는 '__init__(self)' 매직 메소드 함수로 생성자를 정의 - self는 객체의 인스턴스 그 자체로 자기 자신을 참조하는 매개변수 - 객체 내의 client_info 메소드 정의
	<pre>clinet = Client('홍길동', 50) clinet.client_info()</pre>	<ul style="list-style-type: none"> - Client 클래스의 객체 client를 생성 - 생성자가 있는 경우 파라미터를 함께 전달 - 생성과 동시에 생성자 함수가 호출됨 - 생성자가 없는 경우 Client()로 생성 - client의 cinet.client_info() 메소드 실행

079 절차지향 프로그래밍 언어 / 객체지향 프로그래밍 언어 / 스크립트 언어★★★

• 절차적 프로그래밍 언어와 객체지향 프로그래밍 언어

구분	설명								
절차적 프로그래밍 언어 (Procedural Programming Language)	<ul style="list-style-type: none">- 프로그램을 프로시저(Procedure)의 단위로 나누어 문제를 해결하는 방식으로 작성하는 프로그래밍 언어- 복사해서 붙이지 않고 같은 코드를 다른 위치에서 사용 가능- 모듈화를 하거나 구조화 가능- 절차적 프로그래밍 언어는 아래 표 외에도 PL/I, 모듈라-2, 에이다, 베이직 등이 있다.								
	<table><tr><th>언어</th><th>설명</th></tr><tr><td>C</td><td><ul style="list-style-type: none">- C는 1972년 켄 톰슨과 데니스 리치가 벨 연구소에서 개발- 수많은 플랫폼에서의 이식성을 제공</td></tr><tr><td>FORTTRAN</td><td><ul style="list-style-type: none">- 1954년 IBM 704에서 과학적인 계산을 하기 위해 시작된 컴퓨터 프로그램 언어- 산술 기호(+, -, *, / 등)와 기초적인 수학 함수들, 벡터, 행렬계산기능 등의 과학 기술 전문언어</td></tr><tr><td>ALGOL</td><td>알고리즘의 연구개발, 수치 계산과 논리 연산에 이용하기 위한 목적으로 만들</td></tr></table>	언어	설명	C	<ul style="list-style-type: none">- C는 1972년 켄 톰슨과 데니스 리치가 벨 연구소에서 개발- 수많은 플랫폼에서의 이식성을 제공	FORTTRAN	<ul style="list-style-type: none">- 1954년 IBM 704에서 과학적인 계산을 하기 위해 시작된 컴퓨터 프로그램 언어- 산술 기호(+, -, *, / 등)와 기초적인 수학 함수들, 벡터, 행렬계산기능 등의 과학 기술 전문언어	ALGOL	알고리즘의 연구개발, 수치 계산과 논리 연산에 이용하기 위한 목적으로 만들
	언어	설명							
	C	<ul style="list-style-type: none">- C는 1972년 켄 톰슨과 데니스 리치가 벨 연구소에서 개발- 수많은 플랫폼에서의 이식성을 제공							
	FORTTRAN	<ul style="list-style-type: none">- 1954년 IBM 704에서 과학적인 계산을 하기 위해 시작된 컴퓨터 프로그램 언어- 산술 기호(+, -, *, / 등)와 기초적인 수학 함수들, 벡터, 행렬계산기능 등의 과학 기술 전문언어							
ALGOL	알고리즘의 연구개발, 수치 계산과 논리 연산에 이용하기 위한 목적으로 만들								
객체지향 프로그래밍 언어 (Object Oriented Programming Language)	<ul style="list-style-type: none">- 데이터와 그 데이터를 처리할 메소드를 묶어 객체를 만드는 객체 중심의 프로그래밍 언어								
	<table><tr><th>특징</th><th>설명</th></tr><tr><td>캡슐화 (Encapsulation)</td><td><ul style="list-style-type: none">- 변수와 함수를 하나의 단위로 묶는 것- 접근 제어를 통해서 자료형의 정보를 은닉- 모듈 내에서의 응집도를 높이며, 외부로의 노출을 최소화하여 모듈 간의 결합도를 떨어트림</td></tr><tr><td>상속 (Inheritance)</td><td><ul style="list-style-type: none">- 자식 클래스가 부모 클래스의 특성을 그대로 물려 받음- 부모 클래스의 기능을 일부 변경할 경우 이를 오버라이딩(overriding)이라 함</td></tr><tr><td>다형성 (Polymorphism)</td><td>한 요소에 다양한 자료형에 속하는 것이 허가되는 것</td></tr></table>	특징	설명	캡슐화 (Encapsulation)	<ul style="list-style-type: none">- 변수와 함수를 하나의 단위로 묶는 것- 접근 제어를 통해서 자료형의 정보를 은닉- 모듈 내에서의 응집도를 높이며, 외부로의 노출을 최소화하여 모듈 간의 결합도를 떨어트림	상속 (Inheritance)	<ul style="list-style-type: none">- 자식 클래스가 부모 클래스의 특성을 그대로 물려 받음- 부모 클래스의 기능을 일부 변경할 경우 이를 오버라이딩(overriding)이라 함	다형성 (Polymorphism)	한 요소에 다양한 자료형에 속하는 것이 허가되는 것
	특징	설명							
	캡슐화 (Encapsulation)	<ul style="list-style-type: none">- 변수와 함수를 하나의 단위로 묶는 것- 접근 제어를 통해서 자료형의 정보를 은닉- 모듈 내에서의 응집도를 높이며, 외부로의 노출을 최소화하여 모듈 간의 결합도를 떨어트림							
	상속 (Inheritance)	<ul style="list-style-type: none">- 자식 클래스가 부모 클래스의 특성을 그대로 물려 받음- 부모 클래스의 기능을 일부 변경할 경우 이를 오버라이딩(overriding)이라 함							
	다형성 (Polymorphism)	한 요소에 다양한 자료형에 속하는 것이 허가되는 것							
	<ul style="list-style-type: none">- 객체지향 언어는 C++, C#, smaltalk, JAVA 등이 있다.								
	<table><tr><th>언어</th><th>설명</th></tr><tr><td>C++</td><td><ul style="list-style-type: none">- c 문법에 객체지향 프로그래밍 개념을 적용해 만든 언어- c언어와 호환성을 유지함</td></tr><tr><td>JAVA</td><td><ul style="list-style-type: none">- JAVA는 Sun Microsystems에서 개발한 객체지향 언어- 힙에 남아 있으나 변수가 가지고 있던 참조값이나 변수 등 메모리 관리를 위한 가비지 컬렉터(Garbage Collector) 사용</td></tr></table>	언어	설명	C++	<ul style="list-style-type: none">- c 문법에 객체지향 프로그래밍 개념을 적용해 만든 언어- c언어와 호환성을 유지함	JAVA	<ul style="list-style-type: none">- JAVA는 Sun Microsystems에서 개발한 객체지향 언어- 힙에 남아 있으나 변수가 가지고 있던 참조값이나 변수 등 메모리 관리를 위한 가비지 컬렉터(Garbage Collector) 사용		
언어	설명								
C++	<ul style="list-style-type: none">- c 문법에 객체지향 프로그래밍 개념을 적용해 만든 언어- c언어와 호환성을 유지함								
JAVA	<ul style="list-style-type: none">- JAVA는 Sun Microsystems에서 개발한 객체지향 언어- 힙에 남아 있으나 변수가 가지고 있던 참조값이나 변수 등 메모리 관리를 위한 가비지 컬렉터(Garbage Collector) 사용								

• 스크립트 언어

언어	설명	
PHP (Professional Hypertext Preprocessor)	서버용 스크립트 언어로 DB 연동을 편리하게 할 수 있음	
	연산자	설명
	비트 연산자	&(AND), (OR), XOR(^), Shift(<<, >>), Not(~)
	관계 연산자	==, ===(같은), !=, !==(다름), >, <, >=, <= (비교), <=> (같으면 0, 왼쪽이 크면 1, 작으면 -1 반환), <> (같으면 False, 같지 않으면 True)
	에러 연산자	@를 사용해 에러를 무시할 수 있음
	논리 연산자	(OR), &&(AND), xor(XOR), !(NOT)

자바스크립트 (JavaScript)	- 클래스 대신 프로토타입을 이용하지만 ECMAScript6 이상부터 클래스 기반 과 객체 상속 모두 지원 - 객체 기반의 스크립트 언어												
파이썬 (Python)	귀도 반 로섬(Guido van Rossum)이 발표한 언어로 인터프리터 방식이자 객체 지향적이며, 배우기 쉽고 이식성이 좋은 것이 특징인 스크립트 언어												
배시 (Bash)	본셀을 확장한 기능의 쉘로 리눅스에 기본 탑재됨 <table border="1"> <thead> <tr> <th>명령어</th><th>설명</th></tr> </thead> <tbody> <tr> <td>awk</td><td>입력을 주어진 분리자로 분리하여 처리</td></tr> <tr> <td>grep</td><td>지정한 문자열을 포함하고 있는 행을 반환</td></tr> <tr> <td>sort</td><td>텍스트를 정렬</td></tr> <tr> <td>제어문</td><td>조건문(if) 반복문(until, for, while) 등이 있음</td></tr> <tr> <td>sleep</td><td>프로그램을 지정한 시간만큼 일시적으로 정지</td></tr> </tbody> </table>	명령어	설명	awk	입력을 주어진 분리자로 분리하여 처리	grep	지정한 문자열을 포함하고 있는 행을 반환	sort	텍스트를 정렬	제어문	조건문(if) 반복문(until, for, while) 등이 있음	sleep	프로그램을 지정한 시간만큼 일시적으로 정지
명령어	설명												
awk	입력을 주어진 분리자로 분리하여 처리												
grep	지정한 문자열을 포함하고 있는 행을 반환												
sort	텍스트를 정렬												
제어문	조건문(if) 반복문(until, for, while) 등이 있음												
sleep	프로그램을 지정한 시간만큼 일시적으로 정지												

080 라이브러리★★★

• 라이브러리

라이브러리 종류	설명										
표준 라이브러리	<ul style="list-style-type: none"> - 프로그래밍 언어가 기본적으로 포함되어 있는 라이브러리 - c언어 라이브러리 <table border="1"> <thead> <tr> <th>헤더 파일</th><th>설명</th></tr> </thead> <tbody> <tr> <td>stdio.h</td><td>- c언어의 표준 라이브러리 - printf(), scanf()등을 포함</td></tr> <tr> <td>stdlib.h</td><td>- 자료형 변환 제공 - atoi()(char를 int로), atof()(char를 double로), itoa()(int를 char로), ceil()(소수점이 나오면 올림) 등을 포함</td></tr> <tr> <td>string.h</td><td>- 문자열 처리 기능을 제공 - strlen(s)(문자열의 길이를 구함), strcpy(s1, s2) (s2를 s1에 복사), strcmp(s1, s2)(s1과 s2 비교), strcat(s1, s2)(s2를 s의 끝에 연결), strrev(s)(s를 거꾸로 변환) 등을 포함</td></tr> <tr> <td>math.h</td><td>- 제곱근, 삼각함수 등 수학적인 함수를 제공 - sqrt()(제곱근을 구함), acos()(역코사인을 구함) 등을 포함</td></tr> </tbody> </table>	헤더 파일	설명	stdio.h	- c언어의 표준 라이브러리 - printf(), scanf()등을 포함	stdlib.h	- 자료형 변환 제공 - atoi()(char를 int로), atof()(char를 double로), itoa()(int를 char로), ceil()(소수점이 나오면 올림) 등을 포함	string.h	- 문자열 처리 기능을 제공 - strlen(s)(문자열의 길이를 구함), strcpy(s1, s2) (s2를 s1에 복사), strcmp(s1, s2)(s1과 s2 비교), strcat(s1, s2)(s2를 s의 끝에 연결), strrev(s)(s를 거꾸로 변환) 등을 포함	math.h	- 제곱근, 삼각함수 등 수학적인 함수를 제공 - sqrt()(제곱근을 구함), acos()(역코사인을 구함) 등을 포함
헤더 파일	설명										
stdio.h	- c언어의 표준 라이브러리 - printf(), scanf()등을 포함										
stdlib.h	- 자료형 변환 제공 - atoi()(char를 int로), atof()(char를 double로), itoa()(int를 char로), ceil()(소수점이 나오면 올림) 등을 포함										
string.h	- 문자열 처리 기능을 제공 - strlen(s)(문자열의 길이를 구함), strcpy(s1, s2) (s2를 s1에 복사), strcmp(s1, s2)(s1과 s2 비교), strcat(s1, s2)(s2를 s의 끝에 연결), strrev(s)(s를 거꾸로 변환) 등을 포함										
math.h	- 제곱근, 삼각함수 등 수학적인 함수를 제공 - sqrt()(제곱근을 구함), acos()(역코사인을 구함) 등을 포함										
외부 라이브러리	프로그래밍 언어에 기본적으로 포함되어 있지 않아 외부에서 다운받아 설치해야 하는 라이브러리										

081 예외처리

• 예외처리

언어	설명
JAVA	<ul style="list-style-type: none"> - 사용 방법 <pre> try{ 실행 코드; } catch (예외 객체 매개변수){ 예외 발생 시 실행하는 코드; } ... (catch 여러 번 가능) finally { 예외가 발생 여부에 상관없이 처리하는 코드 } </pre>

	- 예외 객체 종류																	
	<table> <tr> <th>예외 객체</th><th>설명</th></tr> <tr> <td>Exception</td><td>- 모든 시스템 종료 외의 내장 예외 - 다른 예외는 해당 예외에서 파생됨</td></tr> <tr> <td>ArithmeticException</td><td>- 예외적인 산술 조건이 발생한 경우 - 0으로 나눈 경우</td></tr> <tr> <td>BufferOverflowException</td><td>버퍼의 제한에 도달한 경우</td></tr> <tr> <td>IndexOutOfBoundsException</td><td>배열, 문자열 등에서 인덱스 범위를 넘어간 경우</td></tr> <tr> <td>NullPointerException</td><td>존재하지 않는 객체를 참조하려고 하는 경우</td></tr> <tr> <td>RuntimeException</td><td>정상 작동 중에 발생할 수 있는 예외의 상위 클래스</td></tr> <tr> <td>FileNotFoundException</td><td>존재하지 않는 파일을 읽으려고 하는 경우</td></tr> </table>	예외 객체	설명	Exception	- 모든 시스템 종료 외의 내장 예외 - 다른 예외는 해당 예외에서 파생됨	ArithmeticException	- 예외적인 산술 조건이 발생한 경우 - 0으로 나눈 경우	BufferOverflowException	버퍼의 제한에 도달한 경우	IndexOutOfBoundsException	배열, 문자열 등에서 인덱스 범위를 넘어간 경우	NullPointerException	존재하지 않는 객체를 참조하려고 하는 경우	RuntimeException	정상 작동 중에 발생할 수 있는 예외의 상위 클래스	FileNotFoundException	존재하지 않는 파일을 읽으려고 하는 경우	
예외 객체	설명																	
Exception	- 모든 시스템 종료 외의 내장 예외 - 다른 예외는 해당 예외에서 파생됨																	
ArithmeticException	- 예외적인 산술 조건이 발생한 경우 - 0으로 나눈 경우																	
BufferOverflowException	버퍼의 제한에 도달한 경우																	
IndexOutOfBoundsException	배열, 문자열 등에서 인덱스 범위를 넘어간 경우																	
NullPointerException	존재하지 않는 객체를 참조하려고 하는 경우																	
RuntimeException	정상 작동 중에 발생할 수 있는 예외의 상위 클래스																	
FileNotFoundException	존재하지 않는 파일을 읽으려고 하는 경우																	
Python	- 사용 방법																	
	<pre>try: 실행 코드 except 예외 객체 as 매개변수: (예외 발생 시 실행하는 코드 ... except 여러 번 가능) finally: 예외가 발생 여부에 상관없이 처리하는 코드</pre>																	
	- 예외 객체 종류																	
	<table> <tr> <th>예외 객체</th><th>설명</th></tr> <tr> <td>Exception</td><td>- 모든 시스템 종료 외의 내장 예외 - 다른 예외는 해당 예외에서 파생됨</td></tr> <tr> <td>ImportError</td><td>import 문이 모듈을 로드하는데 문제가 있는 경우</td></tr> <tr> <td>IndexError</td><td>시퀀스가 인덱스 범위를 벗어난 경우</td></tr> <tr> <td>KeyError</td><td>딕셔너리의 키가 존재하지 않는 경우</td></tr> <tr> <td>KeyboardInterrupt</td><td>사용자가 인터럽트 키(Control-c나 delete 등)를 누른 경우</td></tr> <tr> <td>MemoryError</td><td>메모리가 부족한 경우</td></tr> <tr> <td>NameError</td><td>지역 또는 전역 이름을 찾을 수 없는 경우</td></tr> <tr> <td>ZeroDivisionError</td><td>0으로 나눈 경우</td></tr> </table>	예외 객체	설명	Exception	- 모든 시스템 종료 외의 내장 예외 - 다른 예외는 해당 예외에서 파생됨	ImportError	import 문이 모듈을 로드하는데 문제가 있는 경우	IndexError	시퀀스가 인덱스 범위를 벗어난 경우	KeyError	딕셔너리의 키가 존재하지 않는 경우	KeyboardInterrupt	사용자가 인터럽트 키(Control-c나 delete 등)를 누른 경우	MemoryError	메모리가 부족한 경우	NameError	지역 또는 전역 이름을 찾을 수 없는 경우	ZeroDivisionError
예외 객체	설명																	
Exception	- 모든 시스템 종료 외의 내장 예외 - 다른 예외는 해당 예외에서 파생됨																	
ImportError	import 문이 모듈을 로드하는데 문제가 있는 경우																	
IndexError	시퀀스가 인덱스 범위를 벗어난 경우																	
KeyError	딕셔너리의 키가 존재하지 않는 경우																	
KeyboardInterrupt	사용자가 인터럽트 키(Control-c나 delete 등)를 누른 경우																	
MemoryError	메모리가 부족한 경우																	
NameError	지역 또는 전역 이름을 찾을 수 없는 경우																	
ZeroDivisionError	0으로 나눈 경우																	

Chapter 3. 응용 SW 기초 기술 활용

082 운영체제 종류★★★

• 운영체제 기능

운영체제 기능	내용	
제어 프로그램 (Control Program)	종류	설명
	감시 프로그램	- 운영체제를 제어하고 동작을 감독 및 감독함 - 커널이라고도 함
	작업 제어 프로그램	- 작업의 연속 처리를 위한 스케줄 및 시스템 자원을 할당 - 운영체제의 각종 제어 루틴의 수행 순서를 관리
	데이터 관리 프로그램	주기억장치와 보조기억장치 간 자료 전송과 논리적인 연결, 파일 조작 및 처리
처리 프로그램 (Processing Program)	종류	설명
	언어 번역 프로그램	프로그램 언어를 기계로 번역
	서비스 프로그램	사용 빈도가 높은 프로그램을 시스템 제공자가 미리 작성해 사용자에게 제공
	문제 프로그램	시스템의 문제 해결을 위한 프로그램

• 셸(Shell)과 커널(Kernel)

종류	설명
셸 (Shell)	<ul style="list-style-type: none"> - 자체의 내장 명령어 제공 - 보조기억장치에 상주 - 사용자의 명령을 해석하고 커널로 전달하는 기능을 제공 - 반복적인 명령 프로그램을 만드는 프로그래밍 기능을 제공 - 초기화 파일을 이용해 사용자 환경을 설정하는 기능을 제공 - 파이프라인 기능을 제공 - 사용자 인터페이스를 제공 - 입출력 방향지정 - 여러 종류의 셸이 존재
커널 (Kernel)	<ul style="list-style-type: none"> - 프로세스와 메모리를 관리 - 기억장치, 파일, 입출력 장치를 관리 - 프로세스간 통신 및 데이터 전송 및 변환 등을 수행

• 운영체제의 종류

운영체제	특징
윈도우 (Windows)	<ul style="list-style-type: none"> - 마이크로소프트(Microsoft)에서 개발한 운영체제로 Windows 95, 98, ME, XP, Vista, 7, 8, 10 등으로 버전이 계속 출시 - GUI(Graphic User Interface)를 지원 - 한사람이 한 대를 독점하는 방식인 Single-User 시스템 - 운영체제가 작업의 CPU 시간을 제어하는 선점형 멀티태스킹 제공 - Multi-Tasking을 지원함 - 하드웨어 설치 시 자동으로 감지하는 PnP(Plug and Play)기능을 제공함 - 프로그램 간의 작성 중인 개체를 연결 또는 삽입하여 편집할 수 있는 기능을 제공함
리눅스/유닉스 (Linux/Unix)	<ul style="list-style-type: none"> - 리눅스는 소스가 공개된 개방형(Open) 시스템으로 대부분 무료로 지원하며 유료도 존재 - Multi-Tasking 및 Multi-User를 지원함 - 계층적 파일 시스템으로 트리 구조를 가짐 - 이식성이 높으며 장치 간 호환성이 높음 - 하나 이상의 작업을 백그라운드에서 수행할 수 있어 여러 개의 작업을 병행처리 할 수 있음 - 파일 시스템은 디렉토리과 파일을 쉽게 찾고 유지관리 하며 디스크 블록을 가짐

블록	설명
부트 블록 (boot block)	파일 시스템으로부터 UNIX 커널을 적재시키기 위한 코드를 저장하고 있는 영역
슈퍼 블록 (Super Block)	파일 시스템을 기술하는 블록의 수, 블록 크기 등의 정보를 저장
아이노드 블록 (i-node block)	파일, 디렉토리에 대한 저장공간 등 모든 정보를 가짐
데이터 블록 (Data Block)	실제 데이터가 저장됨

083 메모리 관리 기법★★

• 메모리 배치 전략

기법	설명
최초 적합 (First fit)	- 할당할 수 있는 가장 처음 만나는 빈 메모리 공간에 프로세스를 할당 - 할당이 빠른 장점이 있음
최적 적합 (Best fit)	할당할 수 있는 메모리 공간 중 자원 낭비가 가장 적은 공간에 할당
최악 적합 (Worst fit)	- 할당할 수 있는 메모리 공간 중 자원 낭비가 가장 많은 공간에 할당 - 남은 메모리 공간에 다른 프로세스를 할당할 수 있는 장점이 있음

084 페이지 교체 알고리즘★★★

• 페이지 크기에 따른 현상

페이지 크기	설명
작을 경우	- 더 많은 페이지징 사상 테이블이 필요 - 내부 단편화가 감소 - 페이지의 집합을 효율적으로 운영 가능 - 기억장소 이용 효율이 향상 - 입·출력 시간이 증가
클 경우	- 페이지 사상 테이블의 크기가 작아져 주기억장치 공간을 절약하고 매핑 속도가 빨라짐 - 페이지의 단편화가 증가함

• 페이지 교체 알고리즘

기법	설명
FIFO(First In First Out)	선입 선출로 가장 오래 있었던 페이지를 교체
OPT(OPTimal replacement)	최적 교체로 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체
LRU(Least Recently Used)	가장 오랫동안 사용되지 않은 페이지를 교체
LFU(Least Frequently Used)	참조 횟수가 가장 작은 페이지를 교체
MFU(Most Frequently Used)	참조 횟수가 가장 많은 페이지를 교체
NUR(Not Used Recently)	최근에 사용하지 않은 페이지를 교체
SRC(Second Chance Replacement)	가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지를 교체

• 페이지 교체 관련 개념

개념	설명				
지역성 (Locality)	기억장치로부터 정보가 참조될 때 시간적, 공간적, 순차적으로 분포가 집중되는 성질 <table border="1"> <tr> <th>지역성</th><th>설명</th></tr> <tr> <td>공간 지역성 (Spatial Locality)</td><td>- 기억 장소들에 대해 참조가 집중적으로 이루어지는 경향을 보임</td></tr> </table>	지역성	설명	공간 지역성 (Spatial Locality)	- 기억 장소들에 대해 참조가 집중적으로 이루어지는 경향을 보임
지역성	설명				
공간 지역성 (Spatial Locality)	- 기억 장소들에 대해 참조가 집중적으로 이루어지는 경향을 보임				

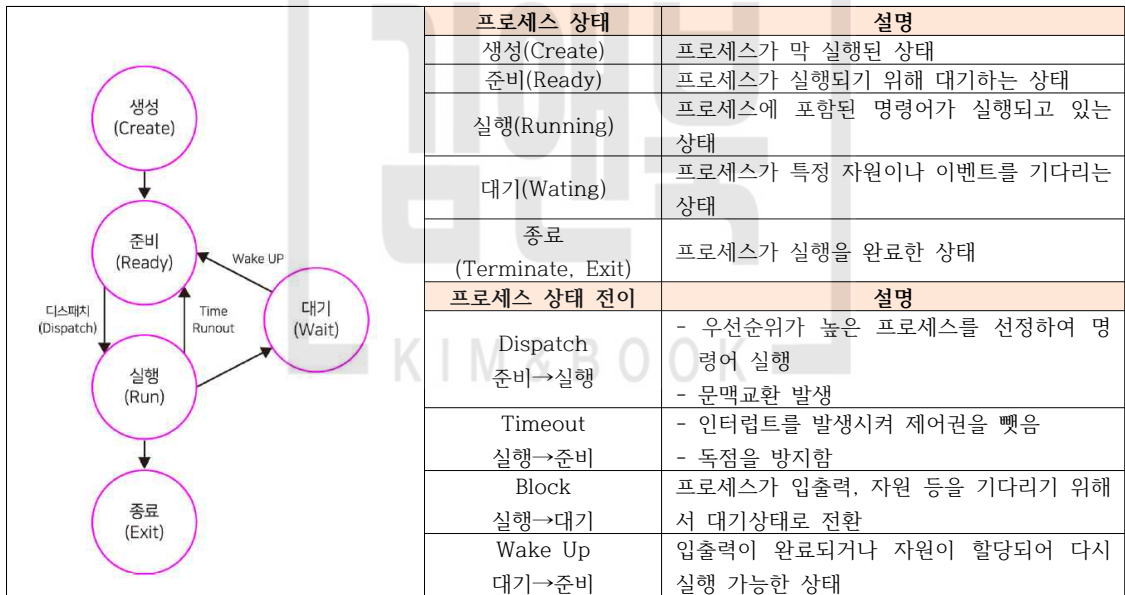
	<table> <tr> <td></td><td>- 참조된 메모리 근처 메모리를 참조</td></tr> <tr> <td>시간 지역성 (Temporal Locality)</td><td>- 최근 사용되었던 기억 장소들이 집중적으로 액세스 되는 현상 - 참조했던 메모리는 빠른 시간에 다시 참조될 확률이 높음</td></tr> <tr> <td>순차 지역성 (Sequential Locality)</td><td>- 데이터가 순차적으로 액세스 되는 현상 - 프로그램 내의 명령어가 순차적인 구성에 기인함</td></tr> </table>		- 참조된 메모리 근처 메모리를 참조	시간 지역성 (Temporal Locality)	- 최근 사용되었던 기억 장소들이 집중적으로 액세스 되는 현상 - 참조했던 메모리는 빠른 시간에 다시 참조될 확률이 높음	순차 지역성 (Sequential Locality)	- 데이터가 순차적으로 액세스 되는 현상 - 프로그램 내의 명령어가 순차적인 구성에 기인함
	- 참조된 메모리 근처 메모리를 참조						
시간 지역성 (Temporal Locality)	- 최근 사용되었던 기억 장소들이 집중적으로 액세스 되는 현상 - 참조했던 메모리는 빠른 시간에 다시 참조될 확률이 높음						
순차 지역성 (Sequential Locality)	- 데이터가 순차적으로 액세스 되는 현상 - 프로그램 내의 명령어가 순차적인 구성에 기인함						
워킹 셋 (Working Set)	운영체제의 가상기억장치 관리에서 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합						
스레싱 현상 (Thrashing)	- 페이지 수행 시간보다 교환시간이 커질 때 발생함 - 프로세스 처리 도중, 참조할 페이지가 주기억장치에 없어 프로세스 처리시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상						

• 세그멘테이션 주소 변환

주소	설명
물리 주소	(세그먼트 번호, 변위 값)으로 표기함
논리 주소	세그먼트 시작 주소 + 변위 값으로 계산함

085 프로세스/스레드★★

• 프로세스 상태



• 프로세스 관련 개념

개념	설명
프로세스 제어 블록 (PCB; Process Control Block)	- 프로세스 상태, 고유 식별자, 스케줄링 정보, 소유자, 실시간 통계, 스레드, 관련 프로세스 리스트, 자식 프로세스 리스트, 주소 공간 자원 스택 등의 정보로 구성 - 운영체제가 프로세스를 스케줄링하고 관리하는 데 필요한 모든 정보를 유지
문맥교환	CPU가 현재 실행하고 있는 프로세스의 상태를 PCB에 저장하고 다음 프로세스의 PCB로부터 문맥을 복원

• 스레드의 분류

스레드의 종류	설명
커널 수준 스레드	<ul style="list-style-type: none"> - 운영체제 커널에 의해 스레드를 운영 - 구현이 쉬우나 속도가 느림 - 여러 스레드가 커널에 동시에 접근 가능
사용자 수준 스레드	<ul style="list-style-type: none"> - 사용자가 만든 라이브러리를 사용해 스레드를 운용 - 속도가 빠르나 구현이 어려움 - 커널 모드로 전환이 없어 오버헤드가 줄어듦

086 프로세스 스케줄링과 교착상태★★★

• 스케줄링

스케줄링	개념
선점 스케줄링	<ul style="list-style-type: none"> - 우선순위가 높은 프로세스 위주로 빠르게 처리 - 빠른 응답시간을 요구하는 대화식 시분할 시스템이 주로 사용 - 선점 시간 배당을 위한 타이머 클럭이 필요하며, 선점으로 인한 오버헤드가 발생
비선점 스케줄링	<ul style="list-style-type: none"> - 모든 프로세스에 대한 요구를 공정하게 처리할 수 있음 - 일괄처리방식에 적합함 - 중요한 짧은 작업이 긴 작업을 기다리는 비효율이 발생할 수 있음 - 응답 시간 예측이 용이함

• 스케줄링 기법

구분	기법	설명
선점 스케줄링	FCFS (First Come First Service)	먼저 들어온 프로세스를 먼저 처리함
	SJF (Shortest Job First)	처리시간이 짧은 프로세스부터 처리함
	HRN (Highest Response-ratio Next)	우선순위가 높은 순서로 처리함 $\text{우선순위} = \frac{\text{대기한 시간} + \text{서비스를 받을 시간}}{\text{서비스를 받을 시간}}$
비선점 스케줄링	라운드 로빈 (Round Robin)	<ul style="list-style-type: none"> - 먼저 들어온 순서대로 일정 시간만큼 처리함 - 시간 할당이 커지면 FCFS 스케줄링과 같아짐
	SRT (Shortest Remaining Time First)	<ul style="list-style-type: none"> - 남은 시간이 짧은 프로세스부터 처리함 - 실행시간을 추적해야 하므로 오버헤드 발생
	MLQ (Multi Level Queue)	우선순위별로 큐를 분리하여 다양한 스케줄링 적용함
	MLFQ (Multi Level Feedback Queue)	MLQ에서 큐 간 이동하여 우선순위 조정

• 교착상태 필요 충분 조건

필요 충분 조건	설명	
상호배제 (Mutual Exclusion)	한 리소스는 한 번에 한 프로세스만 사용할 수 있음	
	알고리즘	설명
	Dekker Algorithm	프로세스가 두 개일 때 Flag와 Turn 변수를 조정해 상호 배제를 보장
	Lamport Algorithm	프로세스에게 고유한 번호를 부여하고, 번호를 기준으로 우선순위를 정해 높은 프로세스가 먼저 임계 구역에 진입하도록 구현
	Peterson Algorithm	프로세스가 두 개일 때 상대방에게 진입 기회를 양보해 상호 배제를 보장

	<table> <tr> <td>Semaphore Algorithm</td><td>공유된 자원의 데이터 혹은 임계영역 등에 따라 여러 Process 혹은 Thread가 접근하는 것을 막아줌</td></tr> </table>	Semaphore Algorithm	공유된 자원의 데이터 혹은 임계영역 등에 따라 여러 Process 혹은 Thread가 접근하는 것을 막아줌
Semaphore Algorithm	공유된 자원의 데이터 혹은 임계영역 등에 따라 여러 Process 혹은 Thread가 접근하는 것을 막아줌		
점유와 대기 (Hold and Wait)	<ul style="list-style-type: none"> - 리소스를 점유하고 있는 프로세스가 있으면 다른 프로세스는 해당 리소스를 사용하기 위해 기다림 - 프로세스가 수행되기 전 모든 자원을 할당함 - 자원이 점유되지 않은 상태에서만 자원을 요구함 		
비선점 (Non Preemption)	프로세스의 자원 사용을 마친 후 리소스를 자발적으로 반환할 때까지 기다림		
환형 대기 (Circular Wait)	두 개 이상의 프로세스 간 자원의 점유와 대기가 원형을 구사해 대기 중인 상태로 Hold and Wait 관계의 프로세스가 서로를 기다림		

• 교착상태 해결 방법

해결 방법	설명
예방 (Prevention)	<ul style="list-style-type: none"> - 교착상태의 필요 조건을 부정함으로써 교착상태가 발생하지 않도록 미리 예방 - 점유 및 대기, 비선점, 환형대기를 부정함 - 모든 자원을 미리 선점해 두기 때문에 자원 낭비가 심함
회피 (Avoidance)	<ul style="list-style-type: none"> - 교착상태가 발생할 가능성이 있는 자원을 할당하지 않음 - 대표적으로 은행원 알고리즘, 자원 할당 그래프가 있음
발견 (Detectin)	<ul style="list-style-type: none"> - 시스템에서 교착상태가 발생했는지 감시 - 교착상태 발생을 허용하고 발생 시 해결
복구 (Recovery)	<ul style="list-style-type: none"> - 교착상태 발견 후 프로세스를 하나씩 종료해 자원을 회복 - 프로세스 종료(중지) 시 희생자를 선택해야 해 기아 상태 발생

087 환경변수 / 셸 스크립트(Shell Script)★★

• 환경변수 명령어

명령어	설명
printenv	변수 이름을 명령어에 단일 변수로 주면 하나의 단일 변수를 반환
env	환경 변수들을 출력하거나, 환경 변수를 설정하고 적용된 내용을 출력
set	<ul style="list-style-type: none"> - 환경 변수, 옵션을 확인 - 위치 파라미터를 조작
setenv	환경 변수를 추가 또는 업데이트
export	<ul style="list-style-type: none"> - 사용자 환경 변수를 export 시키면 전역(Global) 변수로 변경 - 사용자가 생성한 변수는 export 명령어에 표시하지 않으면 현재 셸에 국한됨

• 셸 스크립트 명령어

운영체제	명령어	설명
Windows	dir	현재 디렉토리의 파일 목록을 표시
	copy	파일을 복사
	del	파일을 삭제
	ren	파일 이름을 변경
	md	디렉토리를 생성
	cd	디렉토리의 위치를 변경
	attrib	파일의 속성을 변경
	find	파일에서 문자열을 찾음
	chkdsk	디스크의 상태를 점검
	format	디스크 표면을 트랙과 섹터로 나누어 초기화
	move	파일을 이동
Linux/Unix	alias	별칭을 정의
	awk	패턴 검사 및 문자열을 처리
	batch	명령어를 배치 대기열에서 실행하도록 스케줄링
	bg	프로세스를 백그라운드에서 실행하도록 함
	fg	프로세스를 포그라운드에서 실행하도록 함

cat	파일을 연결하거나 출력
cd	디렉토리의 위치를 변경
chmod	파일 모드, 특성, 권한을 변경
chown	파일 소유자를 변경
cksum	파일 체크섬 및 크기 기록
cmp	두 파일을 비교
cp	파일을 복사
crontab	백그라운드 작업을 주기적으로 스케줄링
date	날짜 및 시간을 표시
df	남아 있는 디스크 공간을 보여줌
du	파일 공간 사용량을 측정해 줌
echo	인수를 표준 출력에 기록
grep	패턴에 따른 문자열을 검색
kill	프로세스 종료 또는 신호 전송
ls	디렉토리 및 내용을 확인
man	시스템 문서를 표시
mkdir	디렉토리를 생성
mv	파일을 이동함
ps	프로세스 상태를 보고
pwd	현재 디렉토리를 출력
rm	파일을 제거
rmdir	디렉토리를 제거
fork	새로운 프로세스를 생성
who	시스템에 누가 있는지를 표시
tar	여러 개의 파일을 하나로 묶거나 풀 때 사용

088 인터넷 구성과 네트워크★★★

• IEEE 802 표준 규약

IEEE 표준	접속제어 방식
802.3	Ethernet(CSMA/CD)
802.4	Token Bus
802.5	Token RING
802.8	Fiber optic LANS
802.9	Integrated Service LAN
802.11	Wireless LAN & Wi-Fi(음성/데이터 통합 LAN)
표준	접속제어 방식
802.11a	5GHz 대역의 전파를 사용
802.11d	지역 간 로밍용 확장
802.11e	QoS 강화를 위해 MAC 지원 기능 채택
802.11h	장비와의 간섭 문제를 해결하는 방식 지원

• 네트워크 7계층(OSI: Open System Interconnection)-7 Layer

계층	설명
응용 계층 (Application Layer) 7계층	- 단위(PDU): 데이터(Data) - 사용자와 밀접한 계층으로 인터페이스 역할을 함 - 응용 프로세스 간의 정보 교환을 담당함 - TELNET, FTP, SMTP, HTTP, POP3, IMAP, SSH, SNMP, DNS 등의 프로토콜이 있음
표현 계층 (Presentation Layer) 6계층	- 단위(PDU): 데이터(Data) - 상이한 부호체계 간의 변화에 대해 규정 - 인코딩과 디코딩, 압축과 해제, 암호화와 복호화 등의 역할을 수행
세션 계층 (Session Layer) 5계층	- 단위(PDU): 데이터(Data), 메시지(Message) - 응용 프로그램 간의 논리적인 연결 생성 및 제어를 담당
전송 계층	- 단위(PDU): 세그먼트(Segment)

(Transport Layer) 4계층	<ul style="list-style-type: none"> - 종단 간 신뢰성 있는 전송을 담당 - 구체적인 목적지까지 데이터가 도달할 수 있도록 함 - process를 특정하기 위한 주소로 port number를 사용함 - 주요 장비: L4 Switch - 프로토콜은 TCP와 UDP가 있다. 														
네트워크 계층 (Network Layer) 3계층	<ul style="list-style-type: none"> - 단위(PDU): 패킷(Packet) - 종단간 전송을 위한 경로 설정을 담당(End-to-End) - 호스트로 도달하기 위한 최적의 경로를 라우팅 알고리즘을 통해 선택하고 제어함 - 주요 장비: 라우터(Router), L3 Switch <table border="1"> <thead> <tr> <th>프로토콜</th><th>설명</th></tr> </thead> <tbody> <tr> <td>IP (Internet Protocol)</td><td>인터넷이 통하는 네트워크에서 어떤 정보를 수신하고 송신하는 통신에 대한 규약</td></tr> <tr> <td>ARP (Address Resolution Protocol)</td><td> <ul style="list-style-type: none"> - IP address를 이용해 mac address를 알아냄 - MAC address를 통해 IP를 알아내는 RARP(Reverse Address Resolution Protocol)도 있음 </td></tr> <tr> <td>ICMP (Internet Control Message Protocol)</td><td> <ul style="list-style-type: none"> - 오류처리와 전송 경로의 변경 - IP의 동작 과정에서의 전송 오류가 발생하는 때에 대비해 오류 정보를 전송하는 목적으로 사용함 </td></tr> <tr> <td>IGMP (Internet Group Management Protocol)</td><td>호스트 컴퓨터가 멀티캐스트 그룹을 주위의 라우터에 알림</td></tr> <tr> <td>RIP (Routing Information Protocol)</td><td>거릿값에 근거한 알고리즘으로, 목적지까지의 경로(Hop)를 알 수 있음</td></tr> <tr> <td>OSPF (Open Shortest Path First)</td><td> <ul style="list-style-type: none"> - 대규모 네트워크에 적합 - Area 개념을 사용해 전체 OSPF 네트워크를 작은 영역으로 나누어 관리 </td></tr> </tbody> </table>	프로토콜	설명	IP (Internet Protocol)	인터넷이 통하는 네트워크에서 어떤 정보를 수신하고 송신하는 통신에 대한 규약	ARP (Address Resolution Protocol)	<ul style="list-style-type: none"> - IP address를 이용해 mac address를 알아냄 - MAC address를 통해 IP를 알아내는 RARP(Reverse Address Resolution Protocol)도 있음 	ICMP (Internet Control Message Protocol)	<ul style="list-style-type: none"> - 오류처리와 전송 경로의 변경 - IP의 동작 과정에서의 전송 오류가 발생하는 때에 대비해 오류 정보를 전송하는 목적으로 사용함 	IGMP (Internet Group Management Protocol)	호스트 컴퓨터가 멀티캐스트 그룹을 주위의 라우터에 알림	RIP (Routing Information Protocol)	거릿값에 근거한 알고리즘으로, 목적지까지의 경로(Hop)를 알 수 있음	OSPF (Open Shortest Path First)	<ul style="list-style-type: none"> - 대규모 네트워크에 적합 - Area 개념을 사용해 전체 OSPF 네트워크를 작은 영역으로 나누어 관리
프로토콜	설명														
IP (Internet Protocol)	인터넷이 통하는 네트워크에서 어떤 정보를 수신하고 송신하는 통신에 대한 규약														
ARP (Address Resolution Protocol)	<ul style="list-style-type: none"> - IP address를 이용해 mac address를 알아냄 - MAC address를 통해 IP를 알아내는 RARP(Reverse Address Resolution Protocol)도 있음 														
ICMP (Internet Control Message Protocol)	<ul style="list-style-type: none"> - 오류처리와 전송 경로의 변경 - IP의 동작 과정에서의 전송 오류가 발생하는 때에 대비해 오류 정보를 전송하는 목적으로 사용함 														
IGMP (Internet Group Management Protocol)	호스트 컴퓨터가 멀티캐스트 그룹을 주위의 라우터에 알림														
RIP (Routing Information Protocol)	거릿값에 근거한 알고리즘으로, 목적지까지의 경로(Hop)를 알 수 있음														
OSPF (Open Shortest Path First)	<ul style="list-style-type: none"> - 대규모 네트워크에 적합 - Area 개념을 사용해 전체 OSPF 네트워크를 작은 영역으로 나누어 관리 														
데이터 링크 계층 (Data Link Layer) 2계층	<ul style="list-style-type: none"> - 단위(PDU): 프레임(Frame) - 인접한 노드간의 신뢰성 있는 데이터 전송을 제어함 - 네트워크 카드 MAC(Media Access Control) 주소를 통해 목적지를 찾음 - 신뢰성 있는 전송을 위해 흐름 제어(Flow Control), 오류 제어(Error Control), 회선 제어(Line Control)를 수행 - 주요 장비: 브리지(Bridge), L2 Switch <table border="1"> <thead> <tr> <th>프로토콜</th><th>설명</th></tr> </thead> <tbody> <tr> <td>HDLC (High-level Data Link Control)</td><td> <ul style="list-style-type: none"> - 점대점 링크 및 멀티포인트 링크를 위해 개발됨 - 에어 제어를 위해 Go-Back-N ARQ를 사용 - 슬라이딩 윈도우 방식에 의해 흐름 제어를 제공 </td></tr> <tr> <td>PPP (Point-to-Point Protocol)</td><td>네트워크 분야에서 두 통신 노드 간의 직접적인 연결을 위해 일반적으로 사용되는 프로토콜</td></tr> <tr> <td>LLC (Logical link control)</td><td>다양한 매체접속제어 방식 간의 차이를 보완함</td></tr> <tr> <td>X.25</td><td>DTE(Data Terminal Equipment)와 DCE(Data Circuit-terminating Equipment)간의 인터페이스 제공</td></tr> </tbody> </table>	프로토콜	설명	HDLC (High-level Data Link Control)	<ul style="list-style-type: none"> - 점대점 링크 및 멀티포인트 링크를 위해 개발됨 - 에어 제어를 위해 Go-Back-N ARQ를 사용 - 슬라이딩 윈도우 방식에 의해 흐름 제어를 제공 	PPP (Point-to-Point Protocol)	네트워크 분야에서 두 통신 노드 간의 직접적인 연결을 위해 일반적으로 사용되는 프로토콜	LLC (Logical link control)	다양한 매체접속제어 방식 간의 차이를 보완함	X.25	DTE(Data Terminal Equipment)와 DCE(Data Circuit-terminating Equipment)간의 인터페이스 제공				
프로토콜	설명														
HDLC (High-level Data Link Control)	<ul style="list-style-type: none"> - 점대점 링크 및 멀티포인트 링크를 위해 개발됨 - 에어 제어를 위해 Go-Back-N ARQ를 사용 - 슬라이딩 윈도우 방식에 의해 흐름 제어를 제공 														
PPP (Point-to-Point Protocol)	네트워크 분야에서 두 통신 노드 간의 직접적인 연결을 위해 일반적으로 사용되는 프로토콜														
LLC (Logical link control)	다양한 매체접속제어 방식 간의 차이를 보완함														
X.25	DTE(Data Terminal Equipment)와 DCE(Data Circuit-terminating Equipment)간의 인터페이스 제공														
물리 계층 (Physical Layer) 1계층	<ul style="list-style-type: none"> - 단위(PDU): 비트(Bit) - 주요 장비: 허브(HUB), 리피터(Repeater) 네트워크 카드(NIC) - 물리적인 장치의 전기적, 전자적 연결에 대한 정의 - 디지털 데이터를 아날로그적인 전지적 신호로 변환해 물리적인 전송을 가능하게 함 														

• TCP와 UDP

프로토콜	설명
TCP (Transmission Control Protocol)	<ul style="list-style-type: none"> - 인접한 노드 사이의 프레임 전송 및 오류를 제어 - 흐름 제어(Flow Control)의 기능을 수행 - 전이 중(Full Duplex) 방식의 양방향 가상회선을 제공 - 패킷의 전송 및 오류를 제어해 신뢰성 있는 연결을 지향
UDP (User Datagram Protocol)	<ul style="list-style-type: none"> - 송신자가 수신자에게 일방적으로 데이터그램을 전송하는 통신 방식으로, 비 연결형 프로토콜 - 흐름 제어나 순서제어가 없어 속도가 빠름 - 비 연결형 및 비 신뢰성 전송 서비스를 제공 - 복구 기능을 제공하지 않음 - 신뢰성보다는 속도가 중요한 실시간 전송에 유리함

• TCP의 헤더

필드	설명
Source Port	출발지 포트 번호
Destination Port	목적지 포트 번호
Sequence Number	<ul style="list-style-type: none"> - 바이트 단위의 순서번호 지정하여 데이터의 순위를 유지 - 신뢰성 및 흐름제어 기능제공
Acknowledgment Number	수신하기를 기대하는 다음 바이트 번호
Offset	헤더 길이
Reserved	예약된 필드로 사용하지 않음
Window	<ul style="list-style-type: none"> - 수신 버퍼 여유 용량 크기를 통보해 데이터를 얼마나 받을 수 있는지 상대방에게 알려줌 - 흐름제어를 수행하게 되는 필드
TCP 제어 플래그	U(Urgent) 긴급 비트, A(Ack) 승인 비트, P(Push) 밀어 넣기 비트, R(Reset) 초기화 비트, S(Syn) 동기화 비트, F(Fin) 종료 비트가 있음
Urgent Pointer	<ul style="list-style-type: none"> - 어디서부터 긴급 값인지 알려줌 - TCP 제어 플래그의 U(Urgent) 긴급 비트와 같이 사용

• UDP의 헤더

필드	설명
Source Port	출발지 포트 번호
Destination Port	목적지 포트 번호
Total Length	8 Byte ~ 65507 Byte 사이의 값으로 헤더와 데이터를 합한 전체 길이를 정의
Checksum	오류를 탐지하기 위해 사용

• TCP의 흐름제어 기법

흐름제어 기법	설명
Stop and Wait	<ul style="list-style-type: none"> - 프레임이 손실되었을 때, 손실된 프레임 1개를 전송하고 수신자의 응답을 기다림 - 한 번에 프레임 1개만 전송
Slow Start	패킷을 하나씩 보내고 문제없이 도착하면 윈도우 크기를 패킷마다 1씩 증가시킴
Sliding Window	<ul style="list-style-type: none"> - 확인받지 않고도 여러 패킷을 보낼 수 있어 Stop and Wait보다 효율적 - 윈도우에 포함되는 모든 패킷을 전송하고, 패킷들의 전달이 확인되면 윈도우를 옮겨(slide) 다음 패킷을 전송
Congestion Avoidance	패킷의 흐름을 제어해 네트워크가 혼잡해지지 않게 조절

• TCP의 오류제어 기법

오류제어 기법(ARQ)	설명
Stop and Wait	- 프레임이 손실되었을 때, 손실된 프레임 1개를 전송하고 수신자의 응답을 기다림 - 한 번에 프레임 1개만 전송
Go Back n	- 한 번에 여러 개를 보낸 후 하나의 긍정 확인응답(ACK)을 받고, 후속 데이터 전송 - 부정 확인응답(NAK)을 수신할 때까지 계속하여 데이터를 송신 - 전이중방식에서 동작
Selective Repeat	Go Back n 방식과 비슷하게 동작하지만 오류가 발생한(NACK) 프레임 이후 혹은 오류 프레임만을 재전송
Adaptive	ARQ 횟수를 줄여 전송 효율을 높임

089 IP와 서브네팅★★★

• IP의 특징

- 인터넷 프로토콜(Internet Protocol)의 약자로, 인터넷이 통하는 네트워크에서 어떤 정보를 수신하고 송신하는 통신규약을 의미
- 패킷을 분할, 병합하는 기능을 수행
- 비 연결형, 비 신뢰성 서비스를 제공
- 데이터그램 전송 서비스를 제공
- IP 주소는 논리적 주소이며 MAC(Media Access Control) 주소는 인터넷 가능한 장비가 가지고 있는 물리적 주소
- Bert Effort 원칙에 따른 전송 기능을 제공

• IPv4의 헤더 필드

필드	설명
Version	- 4비트 - IPv4의 버전4를 사용
Header Length	- 4비트 - 헤더의 길이를 워드 단위로 표시
Type of Service	- 8비트 - 요구되는 서비스 품질
Total Packet Length	- 16비트 - IP 패킷 전체 길이를 바이트 단위로 표시
Fragment Identifier	- 16비트 - 각 조각이 동일한 데이터그램에 속하면 같은 일련번호 공유
Fragmentation Flag	- 3비트 - 분열의 특성을 나타냄
Fragmentation Offset	- 13비트 - 조각나기 전 원래의 데이터그램의 8바이트 단위의 위치
Time To Live	- 8비트 - IP 패킷의 수명
Protocol Identifier	- 8비트 - 어느 상위계층 프로토콜이 데이터 내에 포함되었는지 나타냄
Header Checksum	- 8비트 - 헤더 오류검출
Source Address	- 32비트 - 출발지 IP 주소
Destination Address	- 32비트 - 목적지 IP 주소

• IPv4의 Class

CLASS	설명	중요 사설 IP 주소
A Class	- 앞 비트 0으로 시작 - 0.0.0.0 ~ 127.255.255.255	- Zero 주소 0.0.0.0~0.255.255.255 - 사설망 10.0.0.0~10.255.255.255 - 로컬호스트 127.0.0.0~127.255.255.255
B Class	- 앞 비트 10으로 시작 - 128.0.0.0~191.255.255.255	사설망 127.16.0.0~127.31.255.255
C Class	- 앞 비트 110으로 시작 - 192.0.0.0~223.255.255.255	- IPv4에서 IPv6로 애니캐스트 릴레이 192.88.99.0~192.88.99.255 - 사설망 192.168.0.0~192.168.255.255
D Class	- 앞 비트 1110으로 시작 - 224.0.0.0~239.255.255.255	멀티캐스트 224.0.0.0~239.255.255.255
E Class	- 앞 비트 1111로 시작 - 240.0.0.0~255.255.255.255	예약됨 240.0.0.0~255.255.255.255

• IPv6의 기본 및 확장 헤더 필드

필드	설명
Version	- 4비트 - IPv4의 버전 4를 사용
Traffic	- 8비트 - IPv4의 TOS와 유사하며 요구되는 서비스 품질
Flow Label	- 20비트 - 연결 지향적 프로토콜을 사용할 수 있게 우선권을 주기 위해 특정 트래픽에 대한 라벨링
Payload Length	- 16비트 - IPv4의 Total Packet Length와 유사함 - 확장헤더와 상위계층 데이터의 길이로 최대 65536을 가짐
Next Header	- 16비트 - 기본 헤더 다음에 오는 확장 헤더의 종류를 나타냄
Hop Limit	- 8비트 - IPv4의 TTL과 같이 패킷의 수명
Source Address	- 32비트 - 출발지 IP 주소
Destination Address	- 32비트 - 목적지 IP 주소
IP Header Option	선택적 옵션으로 가변길이

• IPv4와 IPv6 차이점

차이점	설명
IP 주소 확장	IPv4의 32비트에서 128비트로 확장
호스트 주소 자동설정	IPv6 네트워크에 접속하는 순간 자동으로 네트워크 주소를 부여받음
패킷 크기 확장	IPv4에서 패킷 크기는 64킬로바이트로 제한에서 제한이 사라짐
라우팅	- IP 패킷의 처리를 신속하게 할 수 있도록 고정 크기의 단순한 헤더를 사용 - 네트워크 기능에 대한 확장 및 옵션기능의 확장이 용의
플로 레이블링 (Flow Labeling)	특정 트래픽은 별도의 특별한 처리(실시간 통신 등)를 통해 높은 품질의 서비스를 제공할 수 있도록 함
인증 및 보안	패킷 출처 인증과 데이터 무결성 및 비밀 보장 기능을 IP 프로토콜 체계에 반영
이동성	네트워크의 물리적 위치에 제한받지 않고 같은 주소를 유지하면서도 자유롭게 이동할 수 있음

• 데이터 전송 방법

데이터 전송 방법	설명
유니캐스트 (Unicast)	<ul style="list-style-type: none"> - IPv4, IPv6 둘 다 사용 - 출발지와 목적지가 하나로 정해져 있는 1대1 통신
멀티캐스트 (Multicast)	<ul style="list-style-type: none"> - IPv4, IPv6 둘 다 사용 - 여러 명에게 보내야 할 경우에 사용하는 방식 - 특정 그룹을 지정해 그룹원에게 보내는 방식
브로드캐스트 (Broadcast)	<ul style="list-style-type: none"> - IPv4에서만 사용 - 같은 네트워크에 있는 모든 장비에게 보내는 통신 - 상대 IP는 알지만, MAC 정보를 모르는 경우 주로 사용 - 목적지가 전체이기 때문에 과도하게 사용 시 네트워크 성능이 저하될 수 있다.
애니캐스트 (Anycast)	<ul style="list-style-type: none"> - IPv6에서만 사용 - 네트워크에 연결된 수신 가능한 노드 중 한 노드에만 데이터 전송 - 트래픽을 분산하고 네트워크 이중화함

