**4과목 프로그래밍 언어 활용**

1장 서버 프로그램 구현

127 개발 환경 구축

개발 환경 구축 – 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것을 의미

하드웨어 환경 – 사용자와의 인터페이스 역할을 하는 클라이언트와 클라이언트와 통신하여 서비스를 제공하는 서버로 구성됨

서버 – 웹 서버 / 웹 애플리케이션 서버 / 데이터베이스 서버 / 파일 서버

웹 서버 기능 – HTTP/HTTPS 지원 / 통신 기록 / 정적 파일 관리 / 대역폭 제한 / 가상 호스팅 / 인증

소프트웨어 환경 – 클라이언트와 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성

시스템 소프트웨어 – 운영체제, 웹 서버, WAS 운용을 위한 서버 프로그램, DBMS

개발 소프트웨어 – 요구사항 관리 도구 / 설계/모델링 도구 / 구현 도구 / 빌드 도구 / 테스트 도구 / 형상 관리 도구

개발 언어 선정 기준 – 적정성 / 효율성 / 이식성 / 친밀성 / 범용성

128 서버 개발

- 웹 애플리케이션의 로직을 구현할 서버 프로그램을 제작하여 웹 애플리케이션 서버에 탑재하는 것을 의미

Java / JavaScropt / Python / PHP / Ruby

서버 개발 프레임워크 – 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스를 제공하는 소프트웨어

Spring / Node.js / Django / Codeniter / Ruby on Rails

서버 프로그램 – 응용 소프트웨어와 동일하게 모듈 및 공통 모듈을 개발한 후 모듈들을 통합하는 방식으로 구현

모듈의 독립성은 결합도와 응집도에 의해 측정

독립성을 높이려면 모듈의 결합도를 약하게 하고 응집도를 강하게 하여 모듈의 크기를 작게 만들어야 함

129 보안 및 API

소프트웨어 개발 보안 - 소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하여 보안 위협으로부터 안전한 소프트웨어를 개발하기 위한 일련의 보안 활동을 의미

데이터의 기밀성 / 무결성 / 가용성을 유지하는 것이 목표

소프트웨어 개발 보안 점검 항목

세션 통제 / 입력 데이터 검증 및 표현 / 보안 기능 / 시간 및 상태 / 에러처리 / 코드 오류 / 캡슐화 / API 오용

API – 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스

130 배치 프로그램

- 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것

정기 배치 / 이벤트성 배치 / On-Demand 배치

배치 프로그램이 갖추어야 하는 필수 요소 – 대용량 데이터 / 자동화 / 견고성 / 안정성/신뢰성 / 성능

배치 스케줄러 – 일괄 처리 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원해주는 도구

스프링 배치 – Job / Job Launcher / Step / Job Repository

Quartz – Scheduler / Job / JobDetail /Trigger

131 패키지 소프트웨어

- 기업에서 일반적으로 사용하는 여러 기능들을 통합하여 제공하는 소프트웨어를 의미

요구사항을 분석하여 업무 특성에 맞게 전용으로 개발되는 소프트웨어와 비교하여 안정성, 라이선스, 생산성 등에서 차이가 있음

문제정리

소프트웨어 환경 – 시스템 소프트웨어 / 개발 소프트웨어

가상 호스팅 – 하나의 서버로 여러 개의 도메인 이름을 연결하는 기능

대역폭 제한 – 네트워크 트래픽의 포화를 방지하기 위해 응답 속도를 제한하는 기능

개발 언어 선정 기준 – 적정성 / 효율성 / 이식성 / 친밀성

패키지 소프트웨어 – 결함이 발생했을 때 판매처의 프로세스에 따라 보안되므로 이용자의 사저엥 따라 즉각적이고 능동적인 대처가 어려움

전용 개발 소프트웨어 – 사업 환경에 맞추어 직접 개발한 소프트웨어

라이브러리 – 개발 편의를 위해 자주 사용되는 코드

통합 개발 환경 도구 – 개발에 필요한 편집기, 컴파일러, 디버거 등의 다양한 툴을 하나의 인터페이스로 제공하는 소프트웨어

소프트웨어 개발 보안 점검 항목 – 세션 통제 / 입력 데이터 검증 및 표현 / 보안 기능 / 시간 및 상테 / 에러처리 / 코드 오류 / 캡슐화 / API 오용

입력 데이터 검증 및 표현 – 입력 데이터에 대한 유효성 검증체계를 갖추고 검증 실패 시 이를 처리할 수 있도록 코딩하는 것

캡슐화 – 데이터와 데이터를 처리하는 함수를 하나의 객체로 묶어 코딩하는 것

스프링 배치 주요 구성 요소 – Job / JobLauncher / Step / JobRepository

Quartz – 오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있는 안정성, 신뢰성을 필수적으로 갖추고 있음

API – 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스

응용 프로그램 – 특정 업무를 처리하기 위해 만들어진 프로그램

ORM – 객체지향 프로그래밍의 객체와 관계형 데이터베이스의 데이터를 연결하는 기술

프로그래밍 언어의 선정 기준

친밀감 / 언어의 능력 / 처리의 효율성 / 프로그램 구조 / 프로그램 길이 / 이식성 / 과거의 개발 실적 / 알고리즘과 계산상의 난이도 / 자료 구조의 난이도 / 성능 고려 사항들 / 대상 업무의 성격 / 소프트웨어의 수행 환경 / 개발 담당자의 경험과 지식 / 사용자의 요구사항 / 컴파일러의 이용 가능성

키워드 : 1. 웹 서버 2. 웹 애플리케이션 서버 3. 개발 언어 선정 기준 4. 프레임워크 5. 소프트웨어 개발 보안 6. API 7. 배치 프로그램 8. 스프링 배치 9. Quartz 10. 패키지 소프트웨어

2장 프로그래밍 언어 활용

132 데이터 타입

- 변수에 저장될 데이터의 형식을 나타내는 것

변수에 값을 저장하기 전에 문자형, 정수형, 실수형 등 어떤 형식의 값을 저장할지 데이터 타입을 지정하여 변수를 선언해야 함

데이터 타입 유형

정수 타입 – 정수, 즉 소수점이 없는 숫자를 저장할 때 사용

부동 소수점 타입 – 소수점 이하가 있는 실수를 저장할 때 사용

문자 타입 – 한 문자를 저장할 때 사용 / 작은따옴표 안에 표시

문자열 타입 – 문자열을 저장할 때 사용 / 큰 따옴표 안에 표시

불린 타입 – 조건의 참, 거짓 여부를 판단하여 저장할 때 사용 / 기본 값은 거짓

배열 타입 – 같은 타입의 데이터 집합을 만들어 저장할 때 사용 / 데이터는 중괄호 안에 콤마로 구분하여 값들을 나열

C/C++의 데이터 타입 크기 및 기억 범위

문자 char 1Byte / 부호없는 문자형 unsigned char 1Byte / 정수 short 2Byte / 정수 int 4Byte / 정수 long 4Byte / 정수 long long 8Byte / 부호 없는 정수형 unsigned short 2Byte / 부호 없는 정수형 unsigned int 4Byte / 부호 없는 정수형 unsigned ling 4Byte / 실수 float 4Byte / 실수 double 8Byte / 실수 long double 8Byte

JAVA의 데이터 타입 크기 및 기억 범위

문자 char 2Byte / 정수 byte 1Byte / 정수 short 2Byte / 정수 int 4Byte / 정수 long 8Byte / 실수float 4Byte / 실수 double 8Byte / 논리 Boolean 1Byte

Python의 데이터 타입 크기 및 기억 범위

문자 str 무제한 / 정수 int 무제한 / 실수 float 8Byte / 실수 complex 16Byte

133 변수

- 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로 변할 수 있는 값을 의미

변수명 작성 규칙

영문자, 숫자, \_ 사용 가능

첫 글자 영문자 or \_ / 숫자X

글자수 제한 X

공백이나 특수문자 사용 X

대 소문자 구분

예약어 변수명으로 사용X

변수 선언 시 문장 끝 ;

기억 클래스 – 자동 변수 / 레지스터 변수 / 정적 변수(내부) / 정적 변수(외부) / 외부 변수

자동 변수 – 함수나 코드의 범위를 한정하는 블록 내에서 선언되는 변수

외부 변수 – 현재 파일이나 다른 파일에서 선언된 변수나 함수를 참조하기 위한 변수

정적 변수 – 함수나 블록 내에서 선언하는 내부 정적 변수와 함수 외부에서 선언하는 외부 정적 변수

레지스터 변수 – 메모리가 아닌 CPU 내부의 레지스터에 기억영역을 할당받는 변수

변수 선언

자료형 : 변수에 저장될 자료의 형식을 지정

변수형 : 사용자가 원하는 이름을 임의로 지정 / 변수명 작성 규칙에 맞게 지정

값 : 변수를 선언하면서 초기화할 값을 지정

134 연산자

산술 연산자 – 산술 계산에 사용되는 연산자

+ 덧셈 / - 뺄셈 / \* 곱셈 / / 나눗셈 / % 나머지 / ++ 증가 연산자 / -- 감소연산자

관계 연산자 – 두 수의 관계를 비교하여 참 또는 거짓을 결과로 얻는 연산자

== 같다 / != 같지 않다 / > 크다 / >= 크거나 같다 / < 작다 / <= 작거나 같다

비트 연산자 – 비트별(0,1)로 연산하여 결과를 얻는 연산자

& and / ^ xor / ! or / ~ not / << 왼쪽 시프트 / >> 오른쪽 시프트

논리 연산자 – 두 개의 논리 값을 연산하여 참 또는 거짓을 결과로 얻는 연산자

! not / && and / || or

대입 연산자 – 연산 후 결과를 대입하는 연산식을 간략하게 입력할 수 있도록 대입 연산자를 제공

+= / -= / \*= / /= / %= / <<= / >>=

조건 연산자 – 조건에 따라 서로 다른 수식을 수행

조건 ? 수식1:수식2;

135 데이터 입•출력

scanf() – 키보드로 입력받아 변수에 저장하는 함수

scanf(서식 문자열, 변수의 주소)

서식 문자열 - %d / %u/ %o / %x / %c / %s / %f / %e / %d / %e / %ld / %lo / %lx / %p

printf() – 인수로 주어진 값을 화면에 출력하는 함수

printf(서식 문자열, 변수)

주요 제어 문자 - \n / \b / \t / \r / \0 / \’ / \” / \a / \\ / \f

기타 표준 입•출력 함수

getchar() / gets() – 입력

putchar() / puts() – 출력

136 제어문

- 프로그램의 순서를 변경할 때 사용하는 명령문

단순 if – 조건이 한 개일 때 사용하는 제어문

형식1: 조건이 참일 때만 실행

if(조건)

실행할 문장;

if(조건) {

실행할 문장1;

실행할 문장 2 ;

… }

형식2 : 조건이 참일 대와 거짓일 때 실행할 문장이 다르다

if(조건)

실행할 문장1;

else

실행할 문장2;

다중if – 조건이 여러 개 일 때 사용하는 제어문

if(조건1)

실행할 문장1;

else if(조건2)

실행할 문장2;

…

else

실행할 문장3;

형식2: if문 안에 if문이 포함된다

if(조건1) {

if(조건2)

실행할 문장1;

else

실행할 문장2; }

else

실행할 문장3;

switch – 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문

switch(수식)

{

case 레이블1:

실행할 문장1;

break;

case 레이블2:

실행할 문장2;

break;

default:

실행할 문장3;

}

goto - 프로그램 실행 중 현재 위치에서 원하는 다른 문장으로 건너뛰어 수행을 계속하기 위해 사용하는 제어문

goto 레이블;

레이블:

실행할 문장

137 반복문

- 한 종류로 일정한 횟수를 반복하는 명령문

for – 초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문

for(식1; 식2; 식3)

실행할 문장;

while – 조건이 참인 동안 실행할 문장을 반복 수행하는 제어문

while(조건)

실행할 문장;

do~while – 실행할 문장을 우선 실행한 후 조건을 판별하여 조건이 참이면 실행할 문장을 계속 반복 수행하고 거짓이면 do~while문을 끝낸 후 다음 코드를 실행

do

실행할 문장;

while(조건);

break : switch문이나 반복문 안에서 break가 나오면 블록을 벗어남

continue – continue 이후의 문장을 실행하지 않고 제어를 반복문의 처음으로 옮김

138 배열과 문자열

배열 – 동일한 데이터 유형을 여러 개 사용해야 할 경우 이를 손쉽게 처리하기 위해 여러 개의 변수들을 조합해서 하나의 이름으로 정의해 사용하는 것

1차원 배열 – 변수들을 일직선상의 개념으로 조합한 배열

자료형 변수명[개수];

2차원 배열 – 변수들의 평면 / 행과 열로 조합한 배열

자료형 변수명[행개수][열개수]

배열은 선언 시 초기값을 지정할 수 있음

char 배열이름[크기]=”문자열”

“ “ 사용하면 문자열로 지정

139 포인터

- 변수의 주소를 말하며 C언어에서는 주소를 제어할 수 있는 기능을 제공함

-> 연결된 자료 구조를 구성하기 위해 사용

-> 동적으로 할당된 자료 구조를 지정하기 위해 사용

-> 배열을 인수로 전달하기 위해 사용

-> 문자열을 표현하기 위해 사용

-> 커다란 배열에서 요소를 효율적으로 저장하기 위해 사용

-> 메모리에 직접 접근하기 위해 사용

배열은 포인터 변수에 저장한 후 포인터를 이용해 배열의 요소에 접근할 수 있음

140 Python의 기초

- 변수의 자료형에 대한 선언이 없음

- ; 을 사용할 필요가 없음

- 변수에 연속하여 값을 저장하는 것이 가능

input() 함수

- 키보드로 입력받아 변수에 저장하는 함수

형식1

변수 = input(출력문자)

형식2

변수1, 변수2, … = input(출력문자).split(분리문자)

print() 함수

형식1

print(출력값1, 출력값2, … ,sep = 분리문자, end = 종료문자)

형식2

print(서식 문자열 \* % (출력값1, 출력값2, … ))

리스트

형식

리스트명 = [값1, 값2, …]

리스트명 = list([값1,값2, … ])

딕셔너리

형식

딕셔너리명 = {키1:값1, 키2:값2, … }

딕셔너리명 = dict({키1:값1, 키2:값2,…})

range – 연속된 숫자를 생성하는 것으로 리스트, 반복문에서 많이 사용

형식

range(최종값)

range(초기값, 최종값)

range(초기값, 최종값, 증가값)

슬라이스 – 문자열이나 리스트와 같은 순차형 객체에서 일부를 잘라 반환하는 기능

객체명[초기위치:최종위치]

객체명[초기위치:최종위치:증가값]

객체명[:] or 객체명[::]

객체명[초기위치:]

객체명[:초기위치]

객체명[::증가값]

141 Python 활용

if문

if 조건:

실행할 문장

- 조건이 참일 때만 실행

if 조건:

실행할 문장1

else:

실행할 문장2

- 조건이 참일 때와 거짓일 때 실행할 문장

if 조건1:

실행할 문장1

elif 조건2:

실행할 문장2

else:

실행할 문장3

- 조건이 여러 개이고 조건마다 실행할 문장이 다름

if 조건1:

if 조건2:

실행할 문장1

else:

실행할 문장2

else:

실행할 문장3

- if문 안에 if문 포함

for문

for 변수 in range(최종값):

실행할 문장

- range를 이용하는 방식

for 변수 in 리스트

실행할 문장

- 리스트를 이용하는 방식

while문

while 조건:

실행할 문장

142 절차적 프로그래밍 언어

- 일련의 처리 절차를 정해진 문법에 따라 순서대로 기술해 나가는 언어

-> 프로그램이 실행되는 절차를 중요시 함

-> 데이터를 중심으로 프로시저를 구현, 프로그램 전체가 유기적으로 연결되어 있음

-> 자연어에 가까운 단어와 문장으로 구성

-> 과학 계산이나 하드웨어 제어에 주로 사용됨

절차적 프로그래밍 언어의 종류

C / ALGOL / COBOL / FORTRAN

143 객체지향 프로그래밍 언어

- 현실 세계의 개체를 기계의 부품처럼 하나의 객체로 만들어 기계적인 프로그램을 작성할 수 있도록 한 프로그래밍 기법

객체지향 프로그래밍 언어의 종류

JAVA / C++ / Smaltalk

객체지향 프로그래밍 언어의 구성 요소

- 객체 / 클래스 / 메시지

객체지향 프로그래밍 언어의 특징

캡슐화 / 정보 은닉 / 추상화 / 상속성 / 다형성

144 스크립트 언어

- HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용하는 것으로 기계어로 컴파일 되지 않고 별도의 번역기가 소스를 분석하여 동작하게 하는 언어

서버용 스크립트 언어 : ASP / JSP / PHP / 파이썬

클라이언트용 스크립트 언어 : 자바 스크립트, VB 스크립트

스크립트 언어의 종류

자바스크립트 / VB 스크립트 / ASP / JSP / PHP / 파이썬 / 쉘 스크립트 / Basic

145 선언형 언어

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

함수형 언어 – 수학적 함수를 조합하여 문제를 해결하는 언어로 알려진 값을 함수에 적용하는 것을 기반으로 함 / LSP

논리형 언어 – 기호 논리학에 기반을 둔 언어로 논리 문장을 이용하여 프로그램을 표현하고 계산을 수행 / PROLOG

선언형 프로그래밍 언어 종류

HTML / LSP / PROLOG / XML / Haskell

146 라이브러리

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

- 모듈과 패키지 모두를 의미

표준 라이브러리 / 외부 라이브러리

C언어의 대표적인 표준 라이브러리

-> stdio.h / math.h / string.h / stdlib.h / time.h

JAVA의 대표적인 표준 라이브러리

-> java.lang / java.util / java.io / java.net / java.awt

147 예외 처리

- 예외가 발생했을 때 프로그래머가 해당 문제에 대비해 작성해 놓은 처리 루틴을 수행하도록 하는 것

JAVA의 예외처리

try { 예외가 발생할 가능성이 있는 코드; }

catch (예외객체1 매개변수) { 예외객체1에 해당하는 예외 발생 시 처리코드; }

catch (예외객체 2 매개변수) { 예외객체2에 해당하는 예외 발생 시 처리코드; }

catch (예외객체n 매개변수) { 예외객체n에 해당하는 예외 발생 시 처리 코드; }

catch (Exception 매개변수) { 예외객체1~n에 해당하지 않는 예외 발생 시 처리 코드; }

finally { 예외의 발생 여부와 관계없이 무조건 처리되는 코드; }

ClassNotFoundException / NoSuchMethException / FileNotFoundException / InterruptedlOException / ArithmeticException / lllegalArgumentException / NumberFormathException / ArrayIndexOutOfBOundsException / NegativeArraySizeException / NullPointerException

148 프로토타입

- 프로그래밍 언어에서 프로토타입이란 함수 원형이라는 의미 / 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것

C언어에서의 프로토타입 선언

int func(int I, int j);

문제 정리

for문 : 초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문

while문 : 조건이 참인 동안 실행할 문장을 반복 수행하는 제어문, 조건이 처음부터 거짓이면 한 번도 수행하지 않음

switch문 : 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문

|| -> 하나라도 참이면 참을 반환하는 연산자

&& -> 둘 다 참이면 참을 반환하는 연산자

!= -> 같지 않음을 뜻하는 연산자

switch(a) -> a에는 int, char, enum 형의 자료만 사용가능

malloc() 함수는 입력한 byte만큼 메모리를 할당하는 함수

C언어는 컴파일 과정을 거쳐야 실행할 수 있는 컴파일러 언어임

메소드 – 객체의 상태를 참조하거나 변경하기 위한 수단

메시지 – 객체들 사이에서 정보를 교환하기 위한 수단

속성 – 객체가 가지고 있는 정보로 객체의 상태, 분류 등을 나타냄

인스턴스 – 클래스에 속한 각각의 객체

서버용 스크립트 언어 : ASP, JSP, PHP

클라이언트용 스크립트 언어 : 자바 스크립트, VB 스크립트

명령형 언어 – 폰노이만 구조에 개념적 기초를 두고 있음

getchar() 함수는 문자열을 사용할 수 없음

# -> 특수 기호로 코드에 주석을 첨가할 때 사용

@ -> 오류 메시지를 숨겨주는 오류 제어 연산자

<> -> 같지 않음을 의미하는 관계 연산자

=== -> 두 항의 값과 형식이 모두 일치하는지 비교할 때 사용하는 관계 연산자

export – 기존 환경 변수의 값을 변경하거나 새로운 환경 변수를 설정할 때 사용하는 명령어

strcmp는 s1과 s2에 저장된 문자열이 동일한지 비교하는 함수

키워드 : 1. 변수명 작성 규칙 2. 연산자 3. printf 4. 다중 if문 5. for문 6. while문 7. 배열 8. 포인터 9. 스크립트 언어 10. C언어의 라이브러리

3장 응용 sw 기초 기술 활용

149 운영체제의 개념

운영체제 – 컴퓨터 시스템의 자원들을 효율적으로 관리하며 사용자가 컴퓨터를 관리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임

사용자

응용 프로그램

유틸리티

운영체제

하드웨어

운영체제의 목적 – 처리 능력 / 반환 시간 / 사용 가능도 / 신뢰도

제어 프로그램 – 컴퓨터 전체의 작동 삼태 감시, 작업의 순서 지정 작업에 사용되는 데이터 관리 등의 역할을 수행하는 것

처리 프로그램 – 제어 프로그램의 지시를 받아 사용자가 요구한 문제를 해결하기 위한 프로그램

언어 번역 프로그램 / 서비스 프로그램

150 Windows

그래픽 사용자 인터페이스(GUI)

-> 키보드로 명령어를 직접 입력하지 않고 마우스로 아이콘이나 메뉴를 선택하여 모든 작업을 수행하는 방식

선점형 멀티태스킹(Preemptive Multi-Tasking)

-> 동시에 여러 개의 프로그램을 실행하는 멀티태스킹을 하면서 운영체제가 각 작업의 CPU 이용 시간을 제어하여 응용 프로그램 실행 중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환하는 방식

PnP(Plug and Play, 자동 감지 기능)

-> 하드웨어를 사용하는 데 필요한 시스템 환경을 운영체제가 자동으로 구성해주는 기능

OLE(Object Linking and Embedding)

-> 개체를 현재 작성 중인 문서에 자유롭게 연결하거나 삽입하여 편집할 수 있는 기능

151 UNIX/LINUX/MACOS

UNIX – 시분할 시스템을 위해 설계된 대화식 운영체제로 소스가 공개된 개방형 시스템

하드웨어

커널

쉘

유틸리티

사용자 -> 순으로 구성되어 있음

커널 – UNIX의 가장 핵심 부분으로 프로세스 관리, 기억장치 관리, 파일 관리, 입출력 관리, 프로세스간 통신, 데이터 전송 및 변환 등의 기능을 수행

쉘 – 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기

Utility Program – 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용

-> 에디터 / 컴파일러 / 인터프리터 / 디버거

LINUX – UNIX를 기반으로 개발한 운영체제

MacOS - 애플사가 UNIX를 기반으로 개발한 운영체제

152 기억장치 관리의 개요

기억장치 – 레지스터, 캐시 기억장치, 주기억장치, 보조기억장치 계층 구조로 이루어져 있음

상위의 기억장치일수록 접근 속도와 접근 시간이 빠르지만 기억용량이 적고 고가임

기업장치의 관리 전략 – 반입 전략 / 배치 전략 / 교체 전략

반입 전략 – 보조기억장치에 보관중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략

요구 반입 : 실행 중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법

예상 반입 : 실행 중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법

배치 전략 – 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략

최초 적합 – 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법

최적 적합 – 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 작게 남기는 분할 영역에 배치시키는 방법

최악 적합 – 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

교체 전략 – 주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략

- FIFO / OPT / LRU / LFU / NUR / SCR

153 주기억장치 할당 기법

- 프로그램이나 데이터를 실행시키기 위해 주기억장치에 어떻게 할당할 것인지에 대한 내용

연속 할당 기법 / 분산 할당 기법

연속 할당 기법 – 프로그램을 주기억장치에 연속으로 할당하는 기법 ->

단일 분할 할당 기법 / 다중 분할 할당 기법

분산 할당 기법 – 프로그램을 특정 단위의 조각으로 나누어 주기억장치 내에 분산하여 할당하는 기법 -> 페이징 기법 / 세그먼테이션 기법

단일 분할 할당 기법 – 주기억장치를 운영체제 영역과 사용자 영역으로 나누어 한 순간에는 오직 한 명의 사용자만이 주기억장치의 사용자 영역을 사용하는 기법

-> 오버레이 기법 / 스와핑 기법

다중 분할 할당 기법

- 고정 분할 할당 = 정적 할당 기법

-> 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고 준비상태 큐에서 준비중인 프로그램을 각 영역에 할당하여 수행하는 기법

- 가변 분할 할당 = 동적 할당 기법

- 고정 분할 할당 기법의 단편화를 줄이기 위한 것으로 미리 주기억장치를 분활해 놓는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법

154 가상기억장치 구현 기법 / 페아자 교체 알고리즘

가상기억장치 – 보조기억장치의 일부를 주기억장치처럼 사용하는 것으로 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법

페이징 기법 – 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램을 동일하게 나눠진 주기억장치의 영역에 적재시켜 실행하는 기법

세그먼테이션 기법 – 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법

가상주소 형식 : 세그먼트 번호 | 변위값

실기억주소 형식 : 실기억주소(세그먼트 기준번지 + 변위값)

세그먼트 맵 테이블 : 세그먼트 번호 | 세그먼트 크기 | 기준번지

페이지 교체 알고리즘 – 페이지 부재가 발생했을 때 가상기억장치의 필요한 페이지를 주기억장치에 적재해야 하는데 이떄 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프레임을 선택하여 교체할 것인지를 결정하는 기법

OPT : 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법

FIFO : 가장 오래 있었던 페이지를 교체하는 기법

LRU : 최근에 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법

LFU : 사용 빈도가 가장 적은 페이지를 교체하는 기법

NUR : 최근에 사용하지 않은 페이지를 교체하는 기법

155 가상기억장치 기타 관리 사항

페이지 크기

작을 경우 – 페이지 단편화가 감소되고 한 개의 페이지를 주기억장치로 이동하는 시간이 줄어듬

페이지 맵 테이블의 크기가 커지고 매핑 속도가 늦어짐

디스크 접근 횟수가 많아져 전체적인 입출력 시간이 늘어남

클 경우 – 페이지 단편화가 증가되고 한 개의 페이지를 주기억장치로 이동하는 시간이 늘어남

페이지 맵 테이블의 크기가 작아지고 매핑 속도가 빨라짐

디스크 접근 횟수가 줄어들어 전체적인 입출력 효율성 증가

Locality – 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론

시간 구역성 – 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상

공간 구역성 – 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상

워킹 셋 – 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합

페이지 부재 – 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상

프리페이징 – 과도한 페이지 부재를 방지하기 위해 필요할 것 같은 모든 페이지를 한꺼번에 페이지 프레임에 적재하는 기법

스래싱 – 프로세스 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상

156 프로세스의 개요

프로세스 – 일반적으로 프로세스에 의해 처리되는 사용자 프로그램, 시스템 프로그램

PCB – 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓은 곳

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 기다리고 있는 상태

실행 : 준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태

대기 : 프로세스에 입출력 처리가 필요하면 현재 실행중인 프로세스가 중단되고 입출력 처리가 완료될 때까지 대기하고 있는 상태

종료 : 프로세스의 실행이 끝나고 프로세스 할당이 해제된 상태

Dispatch : 준비 상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정

Wake up : 입출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이 되는 과정

Spooling : 입출력장치의 공유 및 상대적으로 느린 입출력장치의 처리 속도를 보안하고 다중 프로그래밍 시스템의 성능을 향상시키기 위해 입출력 할 데이터를 직접 입출력장치에 보내지 않고 나중에 한꺼번에 입출력하기 위해 디스크에 저장하는 과정

쿄통량 제어기 : 프로세스의 상태에 대한 조사와 통보를 담당함

스레드 : 프로세스 내에서의 작업 단위로서 시스템의 여러 자원을 할당받아 실행하는 프로그램 단위

157 스케줄링

- 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미

장기 스케줄링 / 중기 스케줄링 / 단기 스케줄링

비선점 스케줄링 – 이미 할당된 CPU가 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법

선점 스케줄링 – 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법

-> Round Robin / SRT / 선점 우선순위 / 다단계 큐 / 다단계 피드팩 큐

158 주요 스케줄링 알고리즘

FCFS(선입 선출 = FIFO)

- 준비 상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법

SJF(단기 작업 우선)

- 준비 상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법

HRN

- 실행 시간이 긴 프로세스에 불리한 SJF를 보안하기 위한 것으로 대기 시간과 서비스 시간을 이용하는 기법

우선순위 계산식 = 대기시간+서비스 시간 / 서비스 시간

159 환경 변수

- 시스템 소프트웨어의 동작에 영향을 미치는 동적인 값들의 모임

-> 환경변수는 변수명과 값으로 구성

-> 시스템의 기본 정보를 저장

-> 자식 프로세스에 상속

-> 시스템 전반에 걸쳐 적용되는 시스템 환경변수와 사용자 계정 내에서만 적용되는 사용자 환경 변수로 구분

UNIX/LINUX의 주요 환경 변수

- 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞에 ‘$’를 입력해야함

-> set / env / printenv / setenv 중 하나를 입력하면 모든 환경 변수 값을 표시

160 운영체제 기본 명령어

CLI – 키보드로 명령어를 직접 입력하여 작업을 수행하는 사용자 인터페이스

GUI – 마우스로 아이콘이나 메뉴를 선택하여 작업을 수행하는 그래픽 사용자 인터페이스

Windows 기본 명령어

CLI -> 명령 프롬포트 창에 명령어를 입력하여 작업을 수행하는 것

GUI -> 바탕 화면이나 Windows 탐색기에서 마우스로 아이콘을 더블클릭하여 작업하는 것

UNIX/LINUX 기본 명령어

CLI -> 쉘에 명령어를 입력하여 작업을 수행하는 것

GUI -> X Window라는 별도의 프로그램을 설치하여 GUI 방식으로 운영할 수 있음

161 인터넷

- TCP/IP 프로토콜을 기반으로 하여 전 세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 컴퓨터 통신망

백본 – 인터넷의 주가 되는 기간망

IP 주소 – 인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소

A Class – 0~127 국가나 대형 통신망에 사용

B Class – 128~191 중대형 통신망에 사용

C Class – 192~223 소규모 통신망에 사용

D Class – 224~239 멀티캐스트용으로 사용

E Class – 실험적 주소이며 공용되지 않음

서브네팅 – 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용하는 것

IPv6 – IPv4의 주소 부족 문제를 해결하기 위해 개발

유니캐스트 / 멀티캐스트 / 애니캐스트

도메인 네임 – 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현한 것

162 OSI 참조 모델

- 다른 시스템 간의 원활한 통신을 위해 ISO에서 제안한 통신 규약

하위 계층 : 물리 계층 -> 데이터 링크 계층 -> 네트워크 계층

상위 계층 : 전송 계층 -> 세션 계층 -> 표현 계층 -> 응용 계층

물리 계층 – 전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의 / 리피터 허브 / 비트

데이터 링크 계층 – 두 개의 인접한 개방 시스템들 간의 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 시스템 간 연결 설정과 유지 및 종료를 담당 / 랜카드 브리지 스위치 / 프레임

네트워크 계층 – 개방 시스템들 가느이 네트워크 연결을 관리하는 기능과 데이터 교환 및 중계 기능 / 라우터 / 패킷

전송 계층 – 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템 간에 튜명한 데이터 전송을 가능하게 함 / 게이트웨이 / 세그먼트

세션 계층 – 송수신 측 간의 관련성을 유지하고 데이터 교환 관리 기능을 함 / 메시지

표현 계층 – 응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능 / 메시지

응용 계층 – 사용자가 OSI 환경에 접근할 수 있도록 서비스를 제공 / 메시지

163 네트워크 관련 장비

네트워크 인터페이스 카드 – 컴퓨터와 컴퓨터 또는 컴퓨터와 네트워크를 연결하는 장치로 정보 전송 시 정보가 케이블을 통해 전송될 수 있도록 정보 형태를 변경함

허브 – 가까운 거리의 컴퓨터들을 연결하는 장치로 각 회선을 통합적으로 관리하며 신호 증폭 기능을 하는 리피터의 역할도 포함

더미 허브 / 스위칭 허브

리피터 – 전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행

브리지 – LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹을 연결하는 기능을 수행

스위치 – 브리지와 같이 LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치

라우터 – 브리지와 같이 LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능이 추가된 것으로 서로 다른 LAN이나 LAN과 WAN의 연결도 수행

게이트웨이 – 전 계층의 프로토콜 구조가 다른 네트워크의 연결을 수행함

164 프로토콜의 개념

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

구문 : 전송하고자 하는 데이터 형식, 부호화, 신호 레벨 등을 규정

의미 : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정

시간: 두 기기 간의 통신 속도 메시지의 순서 제어 등을 규정

프로토콜의 기능 – 단편화와 재결합 / 캡슐화 / 흐름 제어 / 오류 제어 / 동기화 / 순서 제어 / 주소 지정 / 대중화 / 경로 제어 / 전송 서비스