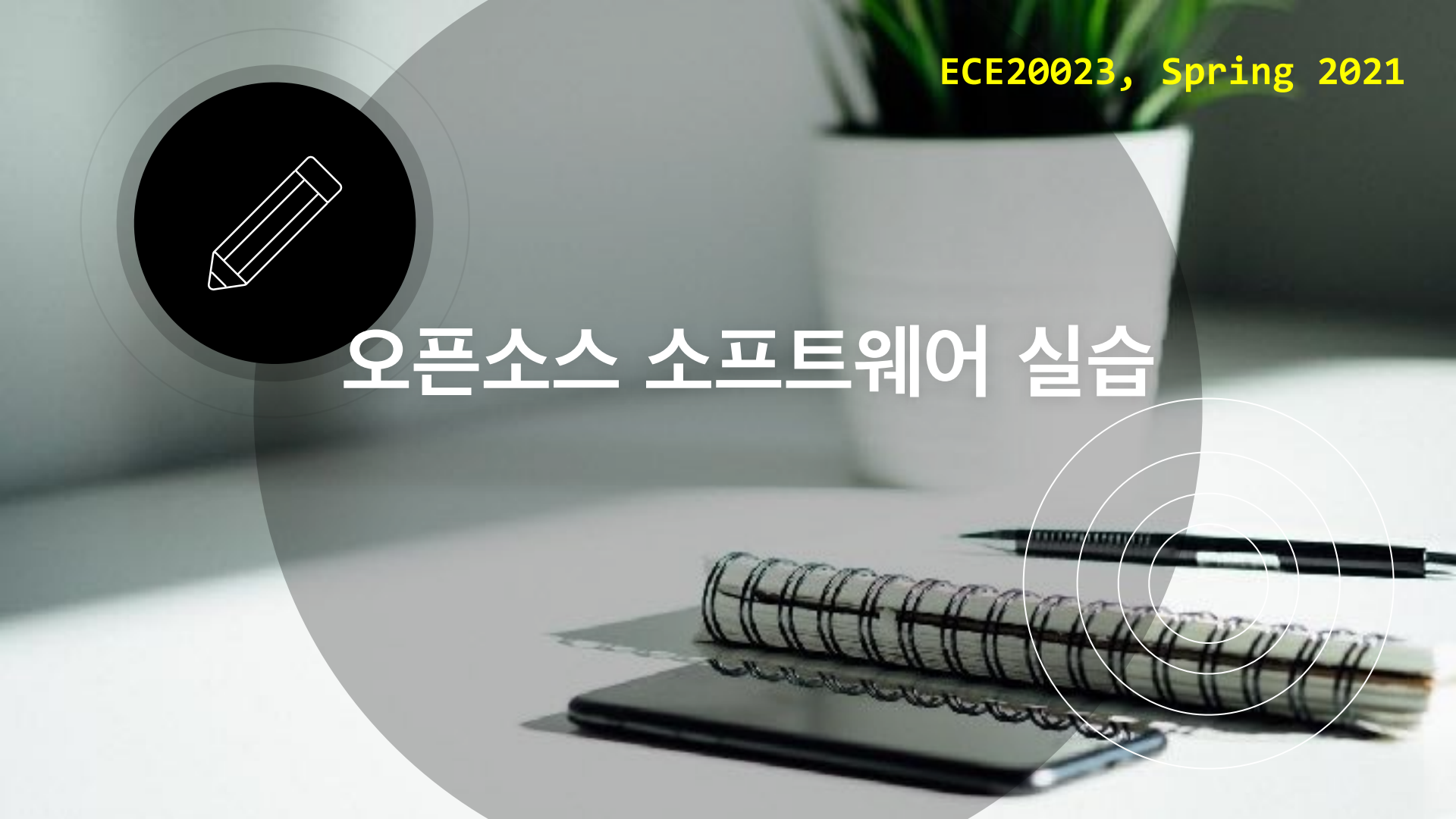
ECE20023, Spring 2021



오픈소스 소프트웨어 실습

10

Debugging

gdb, macro

C 컴파일 과정

01	02 전처리	03 컴파일	04 어셈블	05 링킹
코딩	**전처리기 (PREPROCESSOR)**	**컴파일러 (COMPILER)**	**어셈블러 (ASSEMBLER)**	**링커 (LINKER)**
소스코드 작성 Hello.c	전처리된 파일 **Hello.i**	어셈블리어 파일 생성 **Hello.s**	오브젝트 파일 생성 **Hello.o**	실행파일 생성 **a.out**

```
% gcc -c hello.c
% gcc hello.c
% gcc hello.c -o hello
```

C 컴파일 과정

단계	처리명	입력	출력파일
1	소스코딩		Hello.c
2	Preprocess	Hello.c	Hello.i
3	Compile	Hello.i	Hello.s
4	Assemble	Hello.s	Hello.o
5	Link	Hello.o + library	Hello

```
$ gcc -E hello.c
```

```
$ gcc -S hello.c
```

```
$ gcc -c hello.c
```

```
$ gcc -o hello hello.c
```

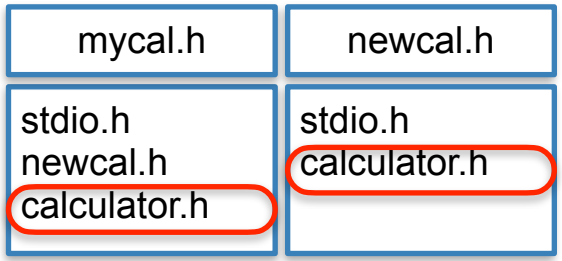
전처리 과정

- C언어 컴파일 과정에서 처음 처리하는 과정
- 헤더 파일 삽입
- 매크로 치환 및 적용
- #define
- #if, #ifdef, #ifndef, #else, #elif, #endif
- 헤더파일 중복 방지

```c
#include <stdio.h>
#define PI 3.14
#define PRINT printf
#define DEBUG

int main(){
#ifdef DEBUG
        printf("DEBUGMODE => Hello World!!!\n");
#endif
        PRINT("Hello World!!!\n");
        return 0;
}
```

헤더파일 중복 방지

- 매크로 상수를 이용하여 헤더파일 중복 방지
- #ifndef, #define, #endif 사용 방법
- #pragma once : gcc, MS c/c++에서 지원, 지원되지 않는 컴파일러도 있음




```
//calculator.h
#ifndef _CALCU_H
#define _CALCU_H
#include <stdio.h>

int plus(int n1, int n2);
 ....

#endif
```

조건부 컴파일

- 매크로 상수를 이용하여 조건부 컴파일이 가능
- #define, #if, #ifdef, #ifndef, #else, #elif, #endif

Hello World!!!

```c
#include <stdio.h>
#define DEBUG   // DEBUG 매크로 정의

int main(){
#ifdef DEBUG
    printf("Debug: %s %s %s %d\n", __DATE__, __TIME__, __FILE__, __LINE__);
#endif

    printf("Hello World!!!\n");
    return 0;
}
```

```
walab-HGU:~/2021OSS/lab6:> gcc -o lab6_1 lab6_1.c
walab-HGU:~/2021OSS/lab6:> ./lab6_1
Debug: Apr  5 2021 12:51:46 lab6_1.c 6
Hello World!!!
```

조건부 컴파일

- 매크로 상수를 컴파일 할 때 정의할 수 있음

```c
#include <stdio.h>
#define DEBUG   // DEBUG 매크로 정의

int main(){
#ifdef DEBUG
    printf("Debug: %s %s %s %d\n", __DATE__, __TIME__, __FILE__, __LINE__);
#endif

    printf("Hello World!!!\n");
    return 0;
}
```

```
walab-HGU:~/2021OSS/lab6:> gcc -o lab6_2 lab6_2.c
walab-HGU:~/2021OSS/lab6:> ./lab6_2
Hello World!!!
```

```
walab-HGU:~/2021OSS/lab6:> gcc -DDEBUG -o lab6_2_debug lab6_2.c
walab-HGU:~/2021OSS/lab6:> ./lab6_2_debug
Debug: Apr  5 2021 13:12:49 lab6_2.c 5
Hello World!!!
```

Makefile(조건부 컴파일)

- 디버깅용과 배포용으로 실행파일 생성하도록 Makefile 작성

```
CC = gcc
CFLAGS = -W -Wall
TARGET = shop
DTARGET = shop_debug
OBJECTS = main.c menu.o guest.o
all : $(TARGET)
$(TARGET) : $(OBJECTS)
        $(CC) $(CFLAGS) -o $@ $^

$(DTARGET) : $(OBJECTS)
        $(CC) $(CFLAGS) -DDEBUG -o $@ $^
clean:
        rm *.o shop
```

```
walab-HGU:~/2021OSS/lab5:> make shop_debug
gcc -W -Wall   -c -o menu.o menu.c
gcc -W -Wall   -c -o guest.o guest.c
gcc -W -Wall -DDEBUG -o shop_debug main.c menu.o guest.o
walab-HGU:~/2021OSS/lab5:> ./shop_debug
=> DEBUGMODE
**********************
1. Pizza : 20000
2. Spaghetti: 12000
**********************
```

실습과제

GDB(GNU Debugger)

- Debugging or Debugger
  - 프로그램 개발 단계 중에 발행하는 시스템의 논리 오류나 비정상적인 연산 등의 원인을 찾아 수정하는 작업 과정
- C, C++언어등을 지원하며 디버깅할 때 사용하는 프로그램
- gdb shell 지원
- gdb사용
  - 컴파일(-g 옵션포함)

```
walab-HGU:~/2021OSS/lab6:> gcc -g lab6_3.c -o lab6_3
```

  - gdb 실행파일명

```
walab-HGU:~/2021OSS/lab6:> gdb lab6_3
```

GDB(GNU Debugger) – 명령어

list	현재 위치에서 소스파일 보여줌	list, 2,5
run	프로그램을 시작	run arg0 arg1
break	특정라인이나 함수에 breakpoint 설정	break main break 3
watch	breakpoint를 변수에 걸어 변수의 값이 변경될 때 break됨	watch i
clear	breakpoint 삭제	
bt	현재 프로그램의 스택을 보여줌(backtrace)	
display	현재 display된 명령의 목록을 보여줌	
next	현재 파일에서 다음행을 수행	n 10
step	한줄씩 수행, 함수가 있으면 내부로 들어가 한줄씩 실행	
print	수식의 값을 보여줌	print i
kill	현재 실행중인 프로그램의 실행을 취소	
cont	Continue, 현재 위치에서 프로그램을 계속 실행	
quit	gdb 종료	

Gdb 실습

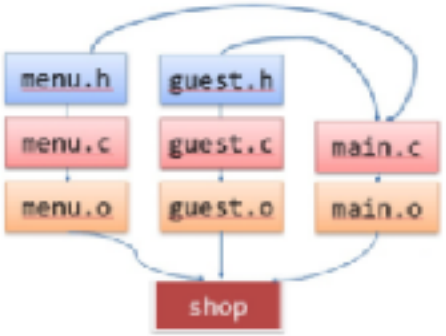
```c
#include <stdio.h>

int main(void) {
  int size,scnt, bcnt;
  printf("size? ");
  scanf("%d", &size);
  scnt = 1;
  bcnt = (size -1)*2;
  for(int i = 0; i < size*2-1; i++){
    for(int j = 0; j < scnt; j++) printf("*");
    for(int j = 0; j < bcnt; j++) printf(" ");
    for(int j = 0; j < scnt; j++) printf("*");
    if(i < size){
      scnt++;bcnt-=2;
    } else{
      scnt--;bcnt+=2;
    }
    printf("\n");
  }
  return 0;
}
```


```
walab-HGU:~/2021OSS/lab6:> gcc -g -o lab6_3 lab6_3.c
walab-HGU:~/2021OSS/lab6:> gdb lab6_3
(gdb) list
(gdb) l
(gdb) b main
(gdb) b 9
(gdb) b 18
(gdb) info break
(gdb) r
(gdb) c
(gdb) p size
(gdb) c
(gdb) l 15, 18
(gdb) display scnt
(gdb) display bcnt
(gdb) display I
(gdb) watch I
(gdb) c
(gdb) bt
(gdb) info locals
(gdb) q
```

실습과제

과제설명

1. 지난 수업시간에 만들었던 소스를 이용하여 make를 이용하여 컴파일 및 빌드 할 수 있도록 Makefile 만들어 보세요.



- menu.h, menu.c, guest.h, guest.h, main.c 소스(수업시간에 작성한 소스 그대로 사용 가능) - **소스 제출**
- Makefile 작성 - **내용 제출**
- make shop 실행 - **실행화면+ 설명**
- make clean 실행 - **실행화면+ 설명**
- make 실행 - **실행화면+설명**
- ./shop 실행 - **실행화면**

2. 조건부 컴파일 실습과제와 gdb 실습과제를 실행하고 결과를 제출하세요.
    - 조건부 컴파일 하는 방법 정리
    - gdb 사용방법 간단히 정리
    - 각 변경된 소스 및 실행결과 화면캡처 제출

MINI Project
3. 다중데이터 처리

Walab 서버에 접속해서 코딩할 것!

제출내용 :

- 현재 mini_project 개발하고 있는 폴더의 절대 주소 (예, /home/s학번/2021OSS/mini_project)
- main.c 소스
- manager.c, manager.h
- Makefile 소스
- mini_project 폴더에서 ls -al 실행결과 및 make 컴파일, 빌드 화면
- 프로그램 실행 결과화면 텍스트 복사 및 붙여넣기(데이터 2개 추가 및 조회프함)

* 과제제출시 반드시 하나의 파일로 작성하여 학번_이름_mini3.pdf 로 제출

CRUD를 구현한 본인의 mini project의 소스를 이용하여 다음 조건에 따라 코딩하세요.

1. 다중 데이터를 처리할 수 있도록 (배열 or 포인터배열 버전) main.c 수정
2. 다중 데이터 추가로 인해 필요한 함수들은 manager.h, manager.c 파일을 추가하여 코딩
3. 조건부 컴파일을 이용하여 debugging 코드 추가
4. Makefile 생성
    a. macro 이용
    b. suffix rule 적용
    c. special macro 적용

- walab.handong.edu 서버에 접속(make를 사용하여 컴파일/실행)
- product.c , product.h 에서 코딩한 CRUD관련 함수는 library이므로 수정할 필요가 없음