

# C 프로그램 개발 환경 및 GCC 컴파일러

---

# 학습 목표

---

- C 프로그램 개발 환경 익히기
- GCC 컴파일러 익히기
- Archive 명령 익히기 (ar)

# Computer Languages

---

- 기계어 (Machine Language)
  - 컴퓨터가 직접 이해 할 수 있는 이진수 집합
  - 소프트웨어 개발자가 기계어로 프로그램 작성하기 어려움
  - 특별한 CPU 만 이해할 수 있는 이진수 코드 (표준화 되지 않음)
- 어셈블리어 (Assembly Language)
  - 기계어 명령어에 일치하는 기억 하기 쉬운 코드
  - 컴퓨터 연산과 데이터 저장 변수들을 이진수보다 기억하기 쉬운 코드와 심볼을
  - 기계어와 같은 단점
- 고급 언어 (High-level Language)
  - 대수 표현과 영어기호를 결합시킨 기계 독립적인 프로그래밍언어
  - FORTRAN, COBOL, LISP, C, Prolog, Ada, Smalltalk, C++, Java

# Programming Language 계층 구조

---

High level Language

C = A + B;



Assembly Language

```
movl 0x8049388, %eax
addl 0x8049384, %eax
movl %eax, 0x804946c
```

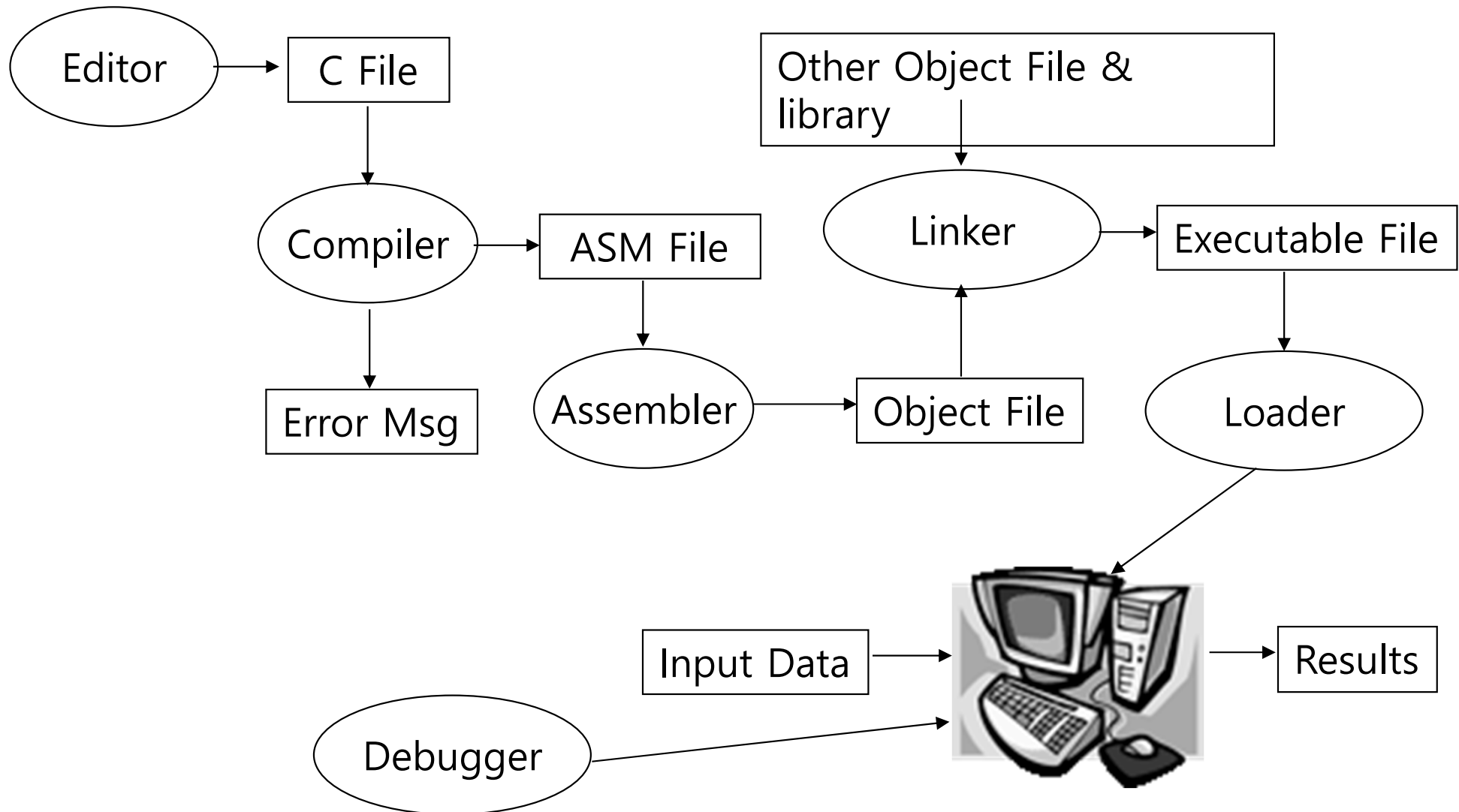


Machine Language  
Binary code (intel)

```
...
00a1 8893 0408
0305 8493 0408
00a3 6c94 0408
...
```

# Program의 작성/번역/실행 절차

---



# Program의 작성/번역/실행 절차

---

- 프로그램 작성
  - 문서 편집기 사용(vi 또는 IDE editor)
  - 파일 이름 : *filename.c*
- 컴파일
  - gcc filename.c 또는 IDE에서 컴파일 명령 사용
  - 컴파일 결과(목적 파일)
    - UNIX : a.out
    - Windows : *filename.exe*
- 실행
  - UNIX : a.out
  - Windows : *filename*
- IDE(Integrated Development Environment)
  - 워드프로세서, 컴파일러, 링커, 로더, 디버깅 작업 도구들을 결합한 통합 개발 환경(예) Turbo-C IDE, MS Visual Studio)

# Program 작성/번역/실행 예

---

- “from sea to shining C”를 출력하는 프로그램 작성
1. 문자 편집기를 사용하여 다음과 같은 내용을 가지는 파일을 작성하고 파일 확장자가 .c인 파일이름을 줌 (예, sea.c)

```
#include <stdio.h>
int main(void)
{
    printf("from sea to shining C\n");
    return 0;
}
```

(참고) 파일 이름은 프로그램 성격에 맞는 것으로 선택해야 함

# Program 작성/번역/실행 예

---

- “from sea to shining C”를 출력하는 프로그램 작성 (계속)

## 2. 앞에서 작성한 프로그램을 컴파일

```
$ gcc sea.c
```

(참고) 코드에 오류가 없다면 이것의 결과로 실행 파일(목적 파일)인 a.out 이 생성됨

## 3. 프로그램 실행

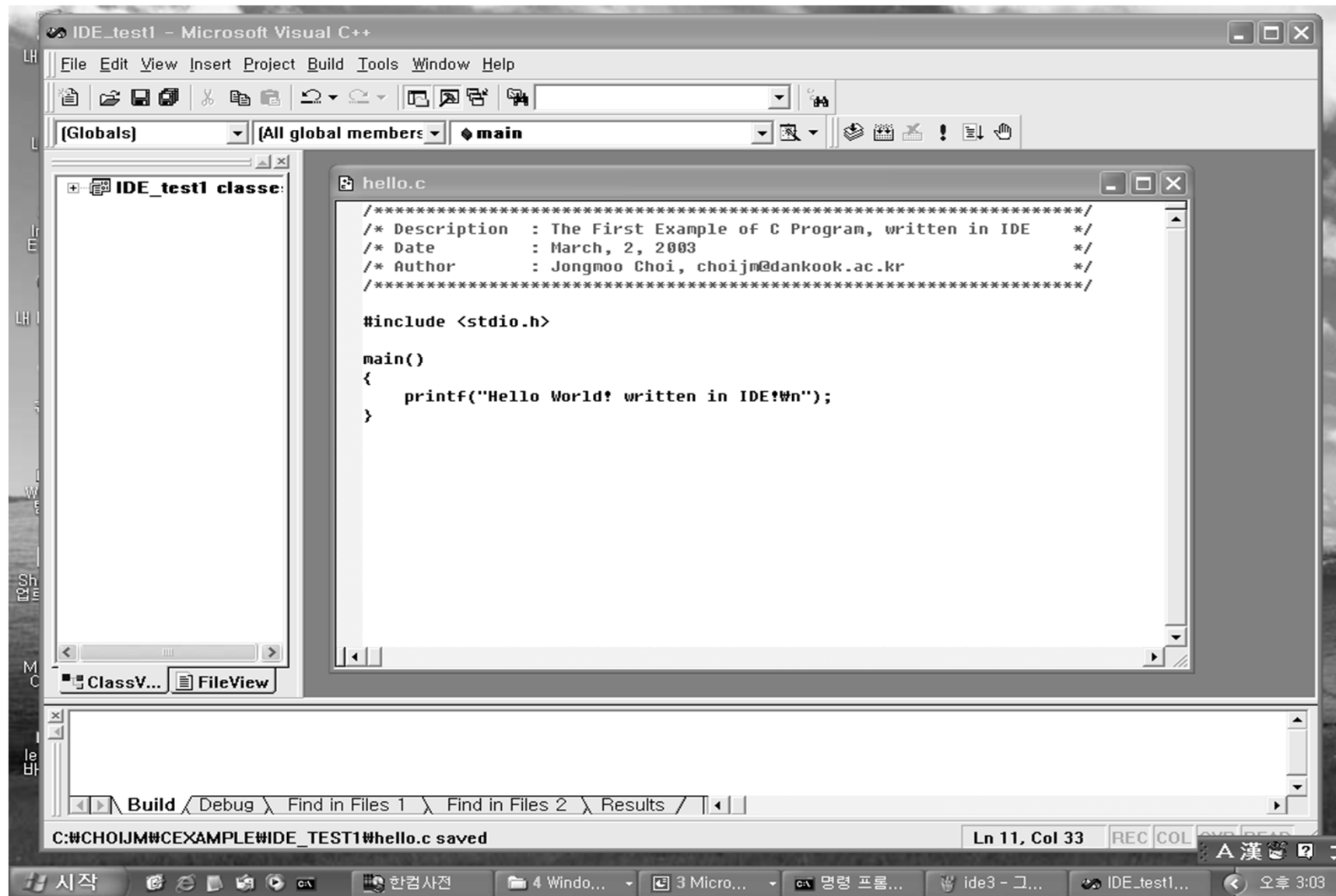
```
$ a.out
```

```
From sea to shining C
```

← 프로그램 수행 결과

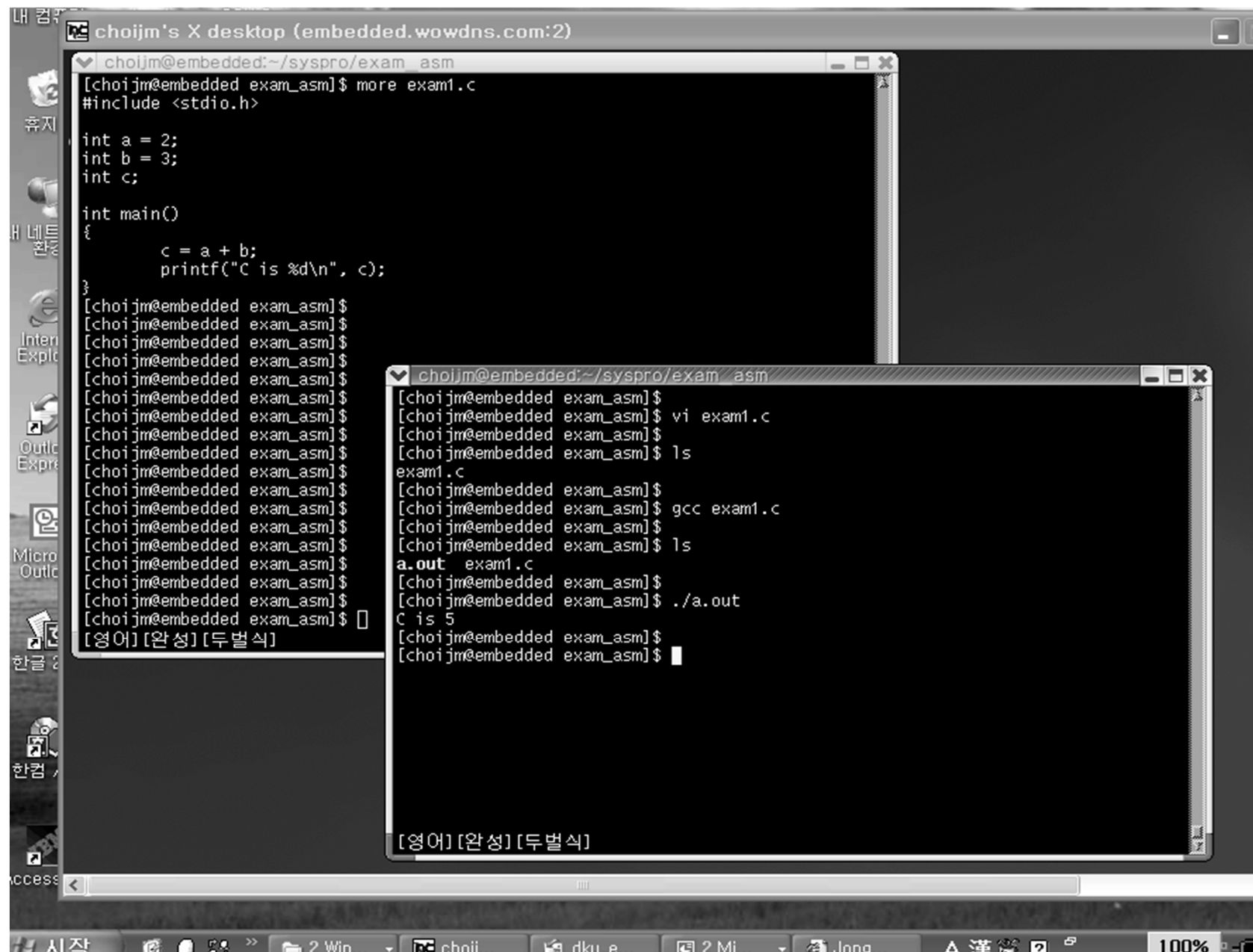


# Compile in MS Windows



☞ IDE (Integrated Development Environments) 사용

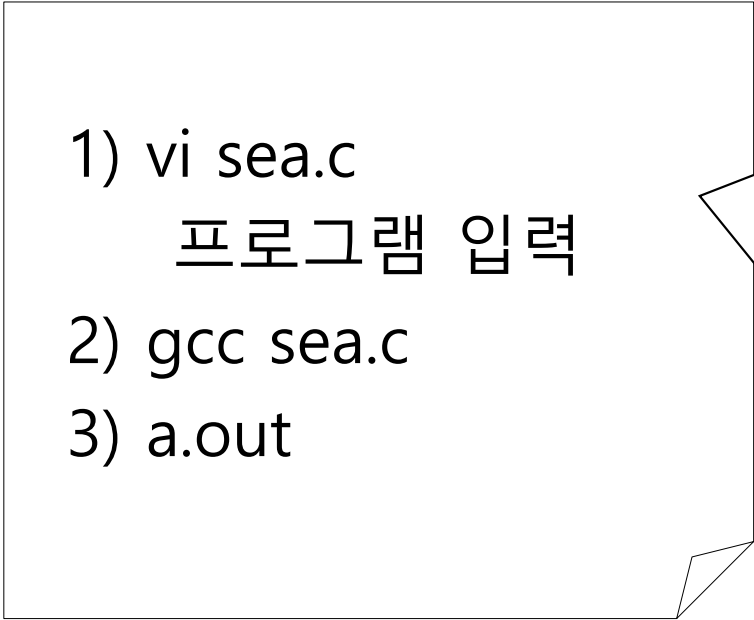
# Compile in UNIX

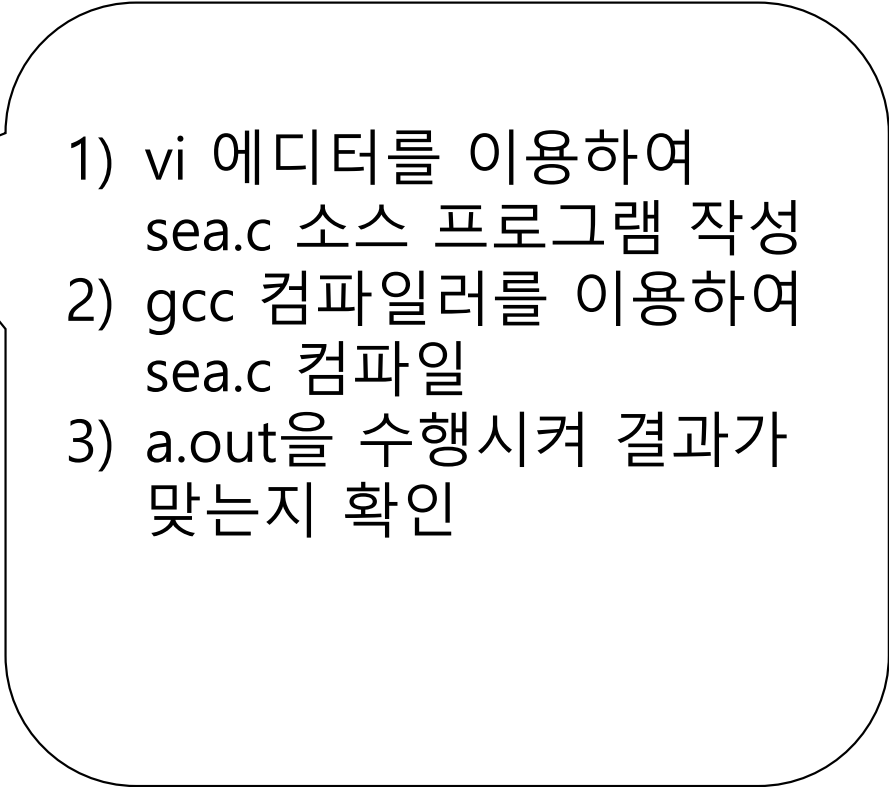


# [실습하기] C 프로그램 작성 및 컴파일

---

- 실습하기

- 
- 1) vi sea.c  
    프로그램 입력
  - 2) gcc sea.c
  - 3) a.out

- 
- 1) vi 에디터를 이용하여  
    sea.c 소스 프로그램 작성
  - 2) gcc 컴파일러를 이용하여  
    sea.c 컴파일
  - 3) a.out을 수행시켜 결과가  
    맞는지 확인

# [실습하기] gcc 사용시 주의 사항

---

- gcc가 제대로 수행되지 않는 경우
  - `cp /etc/ske1/.profile ~` 을 수행한 후 재 로그인
- vi가 제대로 동작이 안되는 경우
  - .profile에 아래 내용 추가할 것
  - `stty -istrip -parenb cs8`

# gcc - C compiler

---

- SYNOPSIS
  - gcc [option | filename] ...
- DESCRIPTION
  - process input files through one or more of four stages
    - preprocessing, compilation, assembly, and linking.
  - Source filename suffixes identify the source language

# gcc - C compiler

---

- 사용되는 파일의 suffix 및 가능한 process
  - .c
    - C source; preprocess, compile, assemble
  - .C
    - C++ source; preprocess, compile, assemble
  - .cc
    - C++ source; preprocess, compile, assemble
  - .i
    - preprocessed C; compile, assemble
  - .ii
    - preprocessed C++; compile, assemble

# gcc - C compiler

---

- 사용되는 파일의 suffix 및 가능한 process
  - .s
    - Assembler source; assemble
  - .S
    - Assembler source; preprocess, assemble
  - .h
    - Preprocessor file; not usually named on command line

# gcc - C compiler

---

- Linker에게 전달되는 파일의 suffix
  - .o
    - Object file
  - .a
    - Archive file
  - .SO
    - shared object file



# gcc - C compiler

---

- EXAMPLES

- gcc hello.c

- hello.c를 compile하여 a.out 실행 파일을 생성.

- gcc part1.c part2.c part3.c

- part1.c, part2.c, part3.c를 각각 compile한 후에 link하여 실행파일 a.out을 생성.

- gcc part1.c part2.o part3.o

- part1.c를 compile 한 후 part2.o, part3.o를 link하여 실행파일 a.out을 생성.

- gcc part1.c part2.o libtmp.a

- part1.c를 compile 한 후 part2.o와 libtmp.a 중 필요한 object file을 link하여 실행파일 a.out 생성.

# gcc - C compiler

---

- OPTIONS: overall options
  - -C
    - compile or assemble the source files, but do not link.
    - suffix가 .o 인 object file을 생성.
    - gcc -c hello.c
      - hello.o 가 생성된다.
  - -S
    - stop after the stage of compilation; do not assemble.
    - suffix가 .s 인 assembler code file을 생성.
    - gcc -S hello.c
      - hello.s 가 생성된다.

# gcc - C compiler

---

- OPTIONS: overall options
  - -E
    - Stop after the preprocessing stage
    - The output is preprocessed source code, which is sent to the standard output
  - -V
    - print the commands executed to run the stages of compilation.

# gcc - C compiler

---

- OPTIONS: overall options
  - -o file
    - place output in file.
    - gcc -o hello hello.c
      - 실행파일 "hello"가 생성된다.
    - gcc -o hi.o -c hello.c
      - object file "hi.o" 가 생성된다.
  - This applies regardless to whatever sort of output GCC is producing, whether it be an executable file, an object file, an assembler file or preprocessed C code.

# gcc - C compiler

---

- OPTIONS: overall options
  - -o file
    - Since only one output file can be specified, it does not make sense to use '-o' when compiling more than one input file, unless you are producing an executable file as output.
    - If you do not specify '-o', the default is to put an executable file in 'a.out', the object file for 'source.suffix' in 'source.o', its assembler file in 'source.s', and all preprocessed C source on standard output.

# gcc - C compiler

---

- OPTIONS: preprocessor options
  - -Dmacro
    - define macro with the string '1' as its definition.
  - -Dmacro=defn
    - define macro as defn. (기존의 macro를 override하지는 않음)
    - gcc -DMAX=10 test.c
  - -Umacro
    - Undefine macro macro.
    - '-U' options are evaluated after all '-D' options

# gcc - C compiler

---

- OPTIONS: linker options
  - -llib
    - library lib를 link한다.
    - link되는 library는 liblib.a 형태의 이름을 가져야 한다.
    - gcc 명령의 인자로 library 파일의 이름을 직접 쓰는 것과의 차이는 여러 directory를 찾는다는 점이다.
  - -static
    - static으로 linking 한다.
    - default는 dynamic linking이다.

# gcc - C compiler

---

- OPTIONS: directory options
  - -Idir
    - include file search path에 dir 추가
  - -Ldir
    - library file search path에 dir을 추가
    - gcc hello.c -L. -lmylib
      - 현재 directory에 있는 libmylib.a 파일을 link 한다



# gcc - C compiler

---

- OPTIONS: warning options
  - -w
    - warning message를 출력하지 않도록 한다.
  - -Wall
    - 모든 warning message를 출력한다.

# gcc - C compiler

---

- OPTIONS: debugging options
  - -g
    - 생성되는 파일에 debugging 정보를 포함시킨다.
    - gdb debugger를 사용하려면, 이 옵션을 사용하여 compile 해야 한다.

# gcc - C compiler

---

- OPTIONS: optimization options
  - -O, -O1
    - Optimize.
    - Optimizing compilation takes somewhat more time, and a lot more memory for a large function.
  - Without '-O', the compiler's goal is to reduce the cost of compilation.
  - Without '-O', only variables declared register are allocated in registers.
  - With '-O', the compiler tries to reduce code size and execution time.

# gcc - C compiler

---

- OPTIONS: optimization options
  - -O2
    - Optimize even more.
    - Nearly all supported optimizations that do not involve a space-speed tradeoff are performed.
    - Loop unrolling and function inlining are not done
  - -O3
    - Optimize yet more.
    - This turns on everything -O2 does, along with function inlining.
  - -O0
    - do not optimize.

# ar - Library Archives

---

- SYNOPSIS

- ar {dmpqrtx} [abiuv] [pos\_name] archive\_name files ...

- DESCRIPTION

- archive (library)를 생성하고 관리

- 필수 option

- d
    - archive에서 지정된 file을 삭제
    - ar d libmylib.a test.o
      - libmylib.a 에서 test.o를 제거

# ar - Library Archives

---

## - p

- 지정된 file의 내용을 출력
- file이 지정되지 않으면 archive의 모든 내용 출력
- `ar p libmylib.a test.o`

## - m [abi] [pos\_name]

- 지정된 file의 위치를 바꾼다.
- 위치 지정을 안 하면 archive의 맨 끝으로 이동
- `ar mb test1.o libmylib.a test2.o`
  - libmylib.a 에 있는 test2.o를 test1.o 앞으로 이동시킨다.
- 같은 symbol이 여러 file에 있을 경우 순서가 중요.

# ar - Library Archives

---

- r [abiku] [pos\_name]
  - archive에 새로운 file을 추가하거나, 같은 이름의 file이 있을 경우 기존의 file을 새로운 file로 대체한다.
  - 위치가 지정되지 않으면 맨 끝에 추가한다.
  - u option을 사용하면, 대체되는 file의 변경시간이 더 나중일 때에만 기존의 file을 새 file로 바꾼다.
  - ar r libmylib.a test3.o
  
- q
  - quick append.
  - archive 내에 같은 이름의 file이 존재하는지 검사하지 않고 추가한다.

# ar - Library Archives

---

- t [v]
  - archive에 속한 file의 목록을 출력.
  - v option을 사용하면 각 file에 대한 정보를 자세히 출력한다.
  - ar tv libmylib.a
  
- x
  - extract. 지정된 file을 archive에서 추출한다.
  - 추출한 file이 archive에서 제거되는 것은 아니다.



# ar - Library Archives

---

- 기타 option
  - a pos\_name
    - 위치 지정. pos\_name의 뒤를 나타낸다.
  - b pos\_name
    - 위치 지정. pos\_name의 앞을 나타낸다.
  - i
    - b와 동일
  - u
    - replace 시 새로운 file일 경우에만 replace

# Example

---

- test.c

```
/* gcc linking example */
```

```
int main(void)
{
    f();
    return 0;
}
```

# Example

---

- f.c

```
#include <stdio.h>
```

```
void f(void)
```

```
{
```

```
    printf("function f()\n");
```

```
    g();
```

```
}
```

# Example

---

- g.c

```
#include <stdio.h>
```

```
void g(void)
```

```
{
```

```
    printf("function g()\n");
```

```
}
```

# Example

---

- make archive

```
$ gcc -c f.c
```

```
$ ar r libf.a f.o
```

```
$ gcc -c g.c
```

```
$ ar r libg.a g.o
```

- compile

```
$ gcc test.c -L. -lf -lg
```