운영체제 및 실습
GCC and Make

한예진, 김수창, 이승준

Dankook University

Index

- GCC
  - 설치
  - 첫 번째 예제 - 단일 파일 컴파일
  - 두 번째 예제 - 다수 파일 컴파일
- Make
  - 예제
  - 매크로

GCC

- GCC (GNU Compiler Collection)
  - GNU 프로젝트의 오픈 소스 컴파일러 컬렉션
  - 1987년 리처드 스톨만 개발
  - 유닉스/리눅스의 표준 컴파일러

* GNU: GNU's not UNIX 의 재귀약자로, 리처드 스톨먼이 각종 자유 소프트웨어들이 돌아가고 번영할 수 있는 기반 생태계를 구축하기 위해 시작한 프로젝트
* Compiler: 어떤 프로그래밍 언어로 쓰여진 소스 코드 파일을 다른 언어로 바꾸어 주는 번역기

GCC

- GCC 설치

```
[choi_gunhee@localhost gcc_practice]$ sudo yum install gcc
```

  - sudo yum install gcc

```
Last metadata expiration check: 3:41:02 ago on Mon 15 Aug 2022 05:29:54 PM KST
.
Dependencies resolved.
================================================================================
 Package             Arch        Version                 Repository       Size
================================================================================
Installing:
 gcc                 x86_64      11.3.1-2.1.el9          appstream        32 M
Installing dependencies:
 glibc-devel         x86_64      2.34-40.el9             appstream        43 k
 glibc-headers       x86_64      2.34-40.el9             appstream       543 k
 kernel-headers      x86_64      5.14.0-142.el9          appstream       3.2 M
 libxcrypt-devel     x86_64      4.4.18-3.el9            appstream        29 k
 make                x86_64      1:4.3-7.el9             baseos          538 k

Transaction Summary
================================================================================
Install  6 Packages

Total download size: 36 M
Installed size: 94 M
Is this ok [y/N]: y
```

GCC

- 첫 번째 예제 파일 만들기

```
#include <stdio.h>

int main() {

    printf("Hello, GCC\n");

    return 0;
}
```

1) cd ~ (Home으로 이동)
2) mkdir gcc_practice (dir 생성)
3) Cd gcc_practice (dir 이동)
4) Vim main.c (main.c 작성)
5) 코드 작성
6) 저장 후 종료

GCC

- 첫 번째 예제 실행 - 단일 파일 컴파일

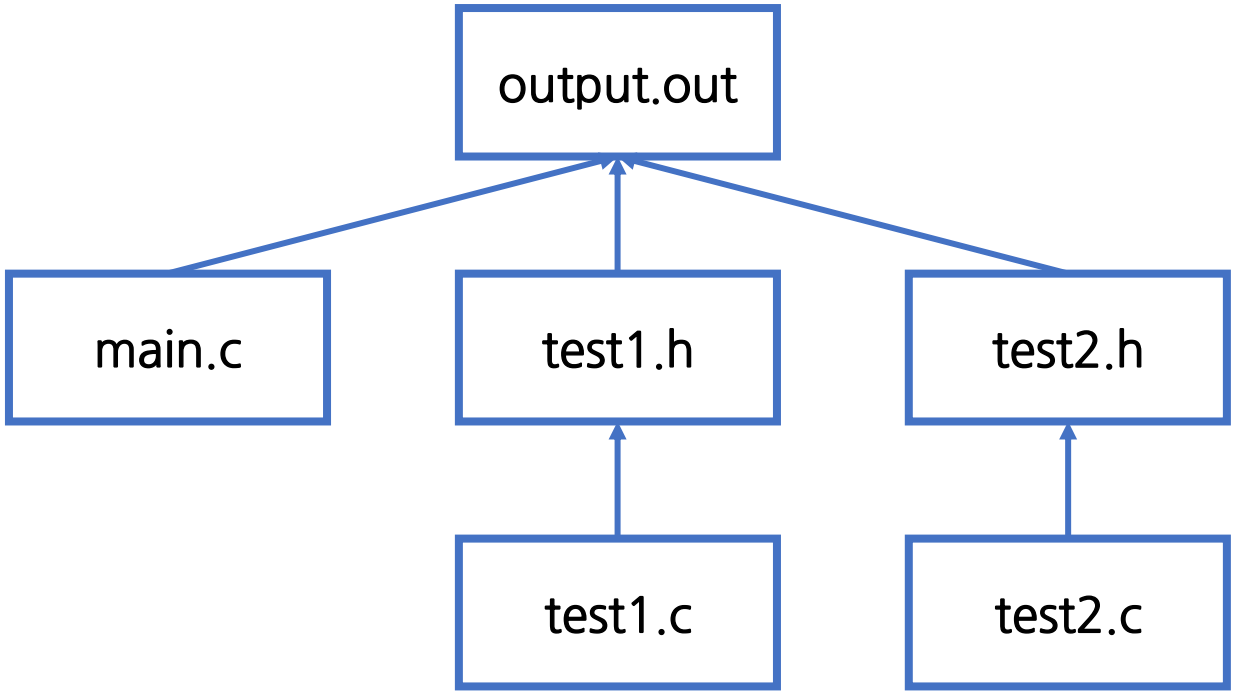
```
[choi_gunhee@localhost gcc_practice]$ ls
main.c
[choi_gunhee@localhost gcc_practice]$ gcc main.c
[choi_gunhee@localhost gcc_practice]$ ls
a.out  main.c
[choi_gunhee@localhost gcc_practice]$ ./a.out
Hello, GCC
[choi_gunhee@localhost gcc_practice]$
```

GCC

- 첫 번째 예제 실행 - 단일 파일 컴파일
  - gcc 옵션
    - o : 지정된 이름으로 실행파일 생성 (지정을 안 할 시, a.out으로 생성)
    - c : 오브젝트 파일 (.o) 생성
    - l(소문자 L) : 같이 링크할 라이브러리 지정
    - v : 컴파일 수행 메세지 표시
    - S : 어셈블리 파일 생성
    - g : 디버깅 옵션, gdb에서 제공하는 정보를 삽입

GCC

- 두 번째 예제 - 다수 파일 컴파일



- 두 번째 예제 - 다수 파일 컴파일

main.c

```
#include <stdio.h>
#include "test1.h"
#include "test2.h"

int main() {
        printf("Hello, I am Main\n");

        print_function1();
        print_function2();

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

- 두 번째 예제 - 다수 파일 컴파일

test1.h > vim test1.h

```
#include <stdio.h>

void print_function1();
```

test1.c > vim test1.c

```
#include "test1.h"

void print_function1() {
        printf("Hello, I am Function1 in test1.c\n");
}
```

- 두 번째 예제 - 다수 파일 컴파일

test2.h > vim test2.h

```
#include <stdio.h>

void print_function2();
```

test2.c > vim test2.c

```
#include "test2.h"

void print_function2() {
        printf("Hello, I am Function2 in test2.c\n");
}
```

GCC

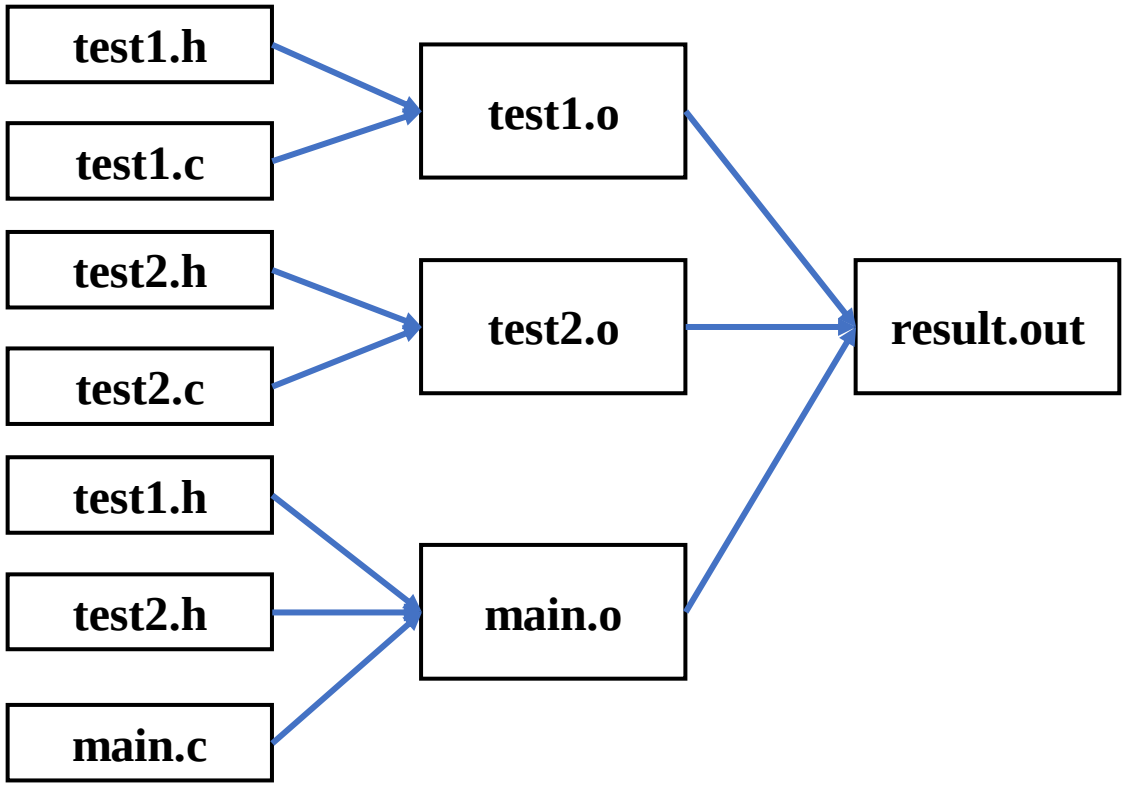
- 두 번째 예제 - 다수 파일 컴파일
  - GCC 순서

    ✓ gcc -c test1.c test1.h
    ✓ gcc -c test2.c test2.h
    ✓ gcc -c main.c test1.h test2.h
    ✓ gcc -o result.out main.o test1.o test2.o

GCC

- 두 번째 예제 - 다수 파일 컴파일
  - GCC 순서

✓ gcc -c test1.c test1.h
✓ gcc -c test2.c test2.h
✓ gcc -c main.c test1.h test2.h
✓ gcc -o result.out main.o test1.o test2.o



GCC

- 두 번째 예제 - 다수 파일 컴파일
  - GCC 순서

```
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  test1.c  test1.h  test2.c  test2.h
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ gcc -c test1.c test1.h
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  test1.c  test1.h  test1.h.gch  test1.o  test2.c  test2.h
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ gcc -c test2.c test2.h
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  test1.c  test1.h  test1.h.gch  test1.o  test2.c  test2.h  test2.h.gch  test2.o
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ gcc -c main.c test1.h test2.h
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  main.o  test1.c  test1.h  test1.h.gch  test1.o  test2.c  test2.h  test2.h.gch  test2.o
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ gcc -o result.out main.o test1.o test2.o
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  main.o  result.out  test1.c  test1.h  test1.h.gch  test1.o  test2.c  test2.h  test2.h.gch  test2.o
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ./result.out
Hello, I am Main
Hello, I am Function1 in test1.c
Hello, I am Function2 in test2.c
Bye!
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$
```

Make

- Make
  - 1976년 개발된 GNU 프로젝트의 빌드 자동화 소프트웨어
  - UNIX 계열 운영 체제를 대상으로 개발

  - 여러 파일들의 각 의존성과 각 파일에 대한 명령을 정의
  - Make를 위한 특정 형식 파일인 Makefile 존재

  - Makefile을 Make가 해석하여 프로그램 빌드를 자동화해준다.
  - 소스코드의 일부만 변경된 경우, 변경된 부분만 다시 컴파일하여 링크한다.

* Makefile: 목적파일, 의존성, 명령어, 매크로 등을 활용하여 컴파일을 쉽게 하기 위해 사용하는 make파일의 설정 파일

Make

- Make 설치

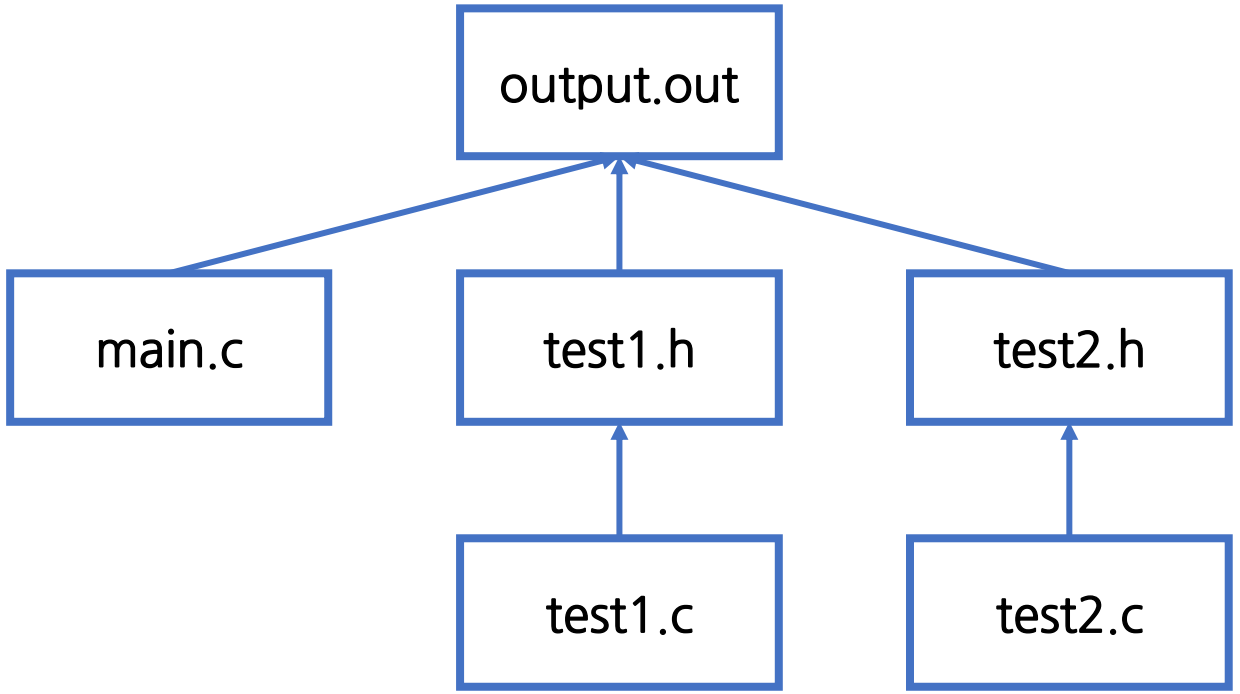
```
[choi_gunhee@localhost gcc_practice]$ sudo yum install make
```

```
[choi_gunhee@localhost gcc_practice]$ sudo yum install make
[sudo] password for choi_gunhee:
Last metadata expiration check: 2:02:19 ago on Mon 15 Aug 2022 09:13:55 PM KST
.
Package make-1:4.3-7.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[choi_gunhee@localhost gcc_practice]$
```

위 명령어로 설치가 가능하다.
하지만, CentOS의 경우 기본적으로 설치 되어 있다.

Make

- Make 예제



- ✓ gcc –c test1.c test1.h
- ✓ gcc –c test2.c test2.h
- ✓ gcc –c main.c test1.h test2.h
- ✓ gcc –o result.out main.o test1.o test2.o

Make

- Make 예제

✓ gcc –c test1.c test1.h

✓ gcc –c test2.c test2.h

✓ gcc –c main.c test1.h test2.h

✓ gcc –o result.out main.o test1.o test2.o

Makefile > vim Makefile

```
result.out : main.o test1.o test2.o
        gcc -o result.out main.o test1.o test2.o

main.o : main.c test1.h test2.h
        gcc -c main.c test1.h test2.h
test1.o : test1.c test1.h
        gcc -c test1.c test1.h
test2.o : test2.c test2.h
        gcc -c test2.c test2.h
```

Make

- Make 예제

```
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  Makefile  test1.c  test1.h  test2.c  test2.h
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ make
gcc -c main.c test1.h test2.h
gcc -c test1.c test1.h
gcc -c test2.c test2.h
gcc -o result.out main.o test1.o test2.o
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  main.o  Makefile  result.out  test1.c  test1.h  test1.h.gch  test1.o  test2.c  test2.h  test2.h.gch  test2.o
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ./result.out
Hello, I am Main
Hello, I am Function1 in test1.c
Hello, I am Function2 in test2.c
Bye!
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$
```

> rm *.o
> rm *.gch
> rm *.out
> make

Make

- Make 매크로
  - 변수 이름들은 $사용
    - $(변수) : 변수
    - $@ : 현재 목표 파일 (target)
    - $< : 현재 목표 파일보다 더 최근에 갱신된 파일

```
CC=gcc
CFLAGS=-g -Wall
OBJS=main.o test1.o test2.o
LIBS = -lpthread
TARGET=result.out

$(TARGET): $(OBJS)
        $(CC) -o $@ $(OBJS) $(LIBS)

main.o : main.c test1.h test2.h
        gcc -c main.c
test1.o : test1.c test1.h
        gcc -c test1.c
test2.o : test2.c test2.h
        gcc -c test2.c

clean :
        rm -f *.o
        rm -f $(TARGET)
```

Make

- Make 매크로
  - 변수 이름들은 $사용
    - $(변수) : 변수
    - $@ : 현재 목표 파일 (target)
    - $< : 현재 목표 파일보다 더 최근에 갱신된 파일

```
CC=gcc
CFLAGS=-c -g -Wall
OBJS=main.o test1.o test2.o
SRCS=test1.c test2.c
HEARS=test1.h test2.h
LIBS = -lpthread
TARGET=result.out

$(TARGET): $(OBJS)
	$(CC) -o $@ $(OBJS) $(LIBS)

%.o: %.c %.h
	$(CC) $(CFLAGS) -c $(SRCS) $(HEARS)

clean :
	rm -f *.o
	rm -f *.gch
	rm -f $(TARGET)
```

Make

- Make 매크로

```
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  Makefile  test1.c  test1.h  test2.c  test2.h
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ make
gcc -c -g -Wall   -c -o main.o main.c
gcc -c -g -Wall -c test1.c test2.c test1.h test2.h
gcc -o result.out main.o test1.o test2.o -lpthread
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ls
main.c  main.o  Makefile  result.out  test1.c  test1.h  test1.h.gch  test1.o  test2.c  test2.h  test2.h.gch  test2.o
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$ ./result.out
Hello, I am Main
Hello, I am Function1 in test1.c
Hello, I am Function2 in test2.c
Bye!
choi_gunhee@gunhee-linux-95:~/test/TABA_1st_example/complete$
```

Makefile


```
CC=gcc
CFLAGS=-c -g -Wall
OBJS=main.o test1.o test2.o
SRCS=test1.c test2.c
HEARS=test1.h test2.h
LIBS = -lpthread
TARGET=result.out

$(TARGET): $(OBJS)
	$(CC) -o $@ $(OBJS) $(LIBS)

%.o: %.c %.h
	$(CC) $(CFLAGS) -c $(SRCS) $(HEARS)

clean :
	rm -f *.o
	rm -f *.gch
	rm -f $(TARGET)
```

```
CC=gcc
CFLAGS=-g -Wall
OBJS=main.o test1.o test2.o
LIBS = -lpthread
TARGET=result.out

$(TARGET): $(OBJS)
	$(CC) -o $@ $(OBJS) $(LIBS)

main.o : main.c test1.h test2.h
	gcc -c main.c
test1.o : test1.c test1.h
	gcc -c test1.c
test2.o : test2.c test2.h
	gcc -c test2.c

clean :
	rm -f *.o
	rm -f $(TARGET)
```