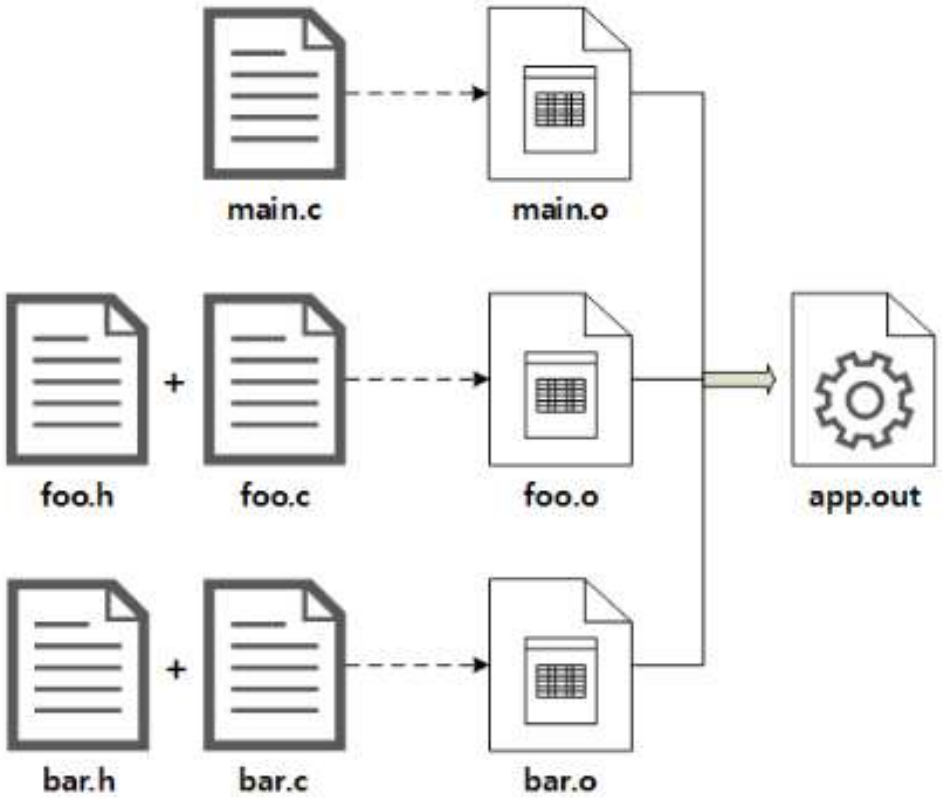
System Software Experiment 1
Lecture 12 – Makefile

Spring 2020

Hwansoo Han (hhan@skku.edu)
Advanced Research on Compilers and Systems, ARCS LAB
Sungkyunkwan University
http://arcs.skku.edu/

Build Example




```
$ gcc –o main main.c foo.c bar.c
$ ./main
```

VS

```
$ gcc –c main.c
$ gcc –c foo.c
$ gcc –c bar.c
$ gcc –o main main.o foo.o bar.o
$ ./main
```

Make & Makefile

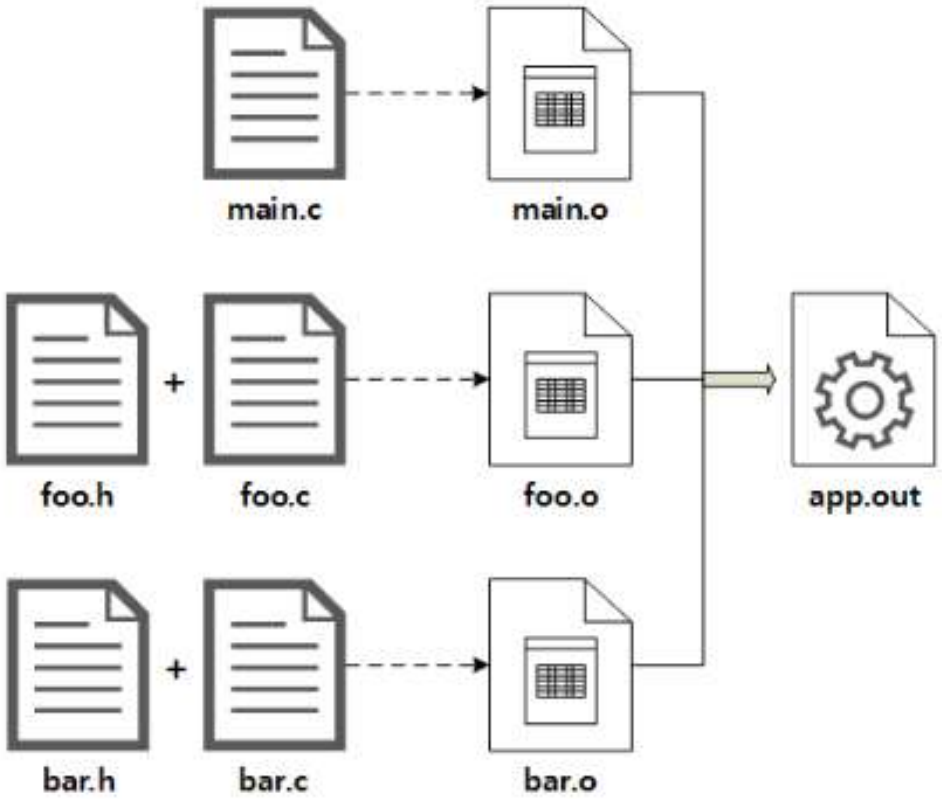
- Make
  - 파일 간의 종속관계를 파악하여 Makefile에 적힌 대로 컴파일러에 명령하여 shell 명령을 순차적으로 실행
  - 각 파일에 대한 반복적 명령의 자동화
  - 프로그램의 종속 구조를 빠르게 파악할 수 있으며 관리가 용이
  - 단순 반복 작업 및 재작성을 최소화

Makefile

- <Target>: 빌드 대상 이름, 최종적으로 생성해내는 파일명
- <Dependencies>: 빌드 대상이 의존하는 target이나 파일 목록, 여기에 나열된 대상들을 먼저 만들고 빌드 대상을 생성
- <Recipe>: 빌드 대상을 생성하는 명령, 여러 줄로 작성할 수 있으며, 각 줄 시작에 반드시 tab문자로 된 indent가 필요

```
<Target>: <Dependencies>
        <Recipe>
```

Makefile



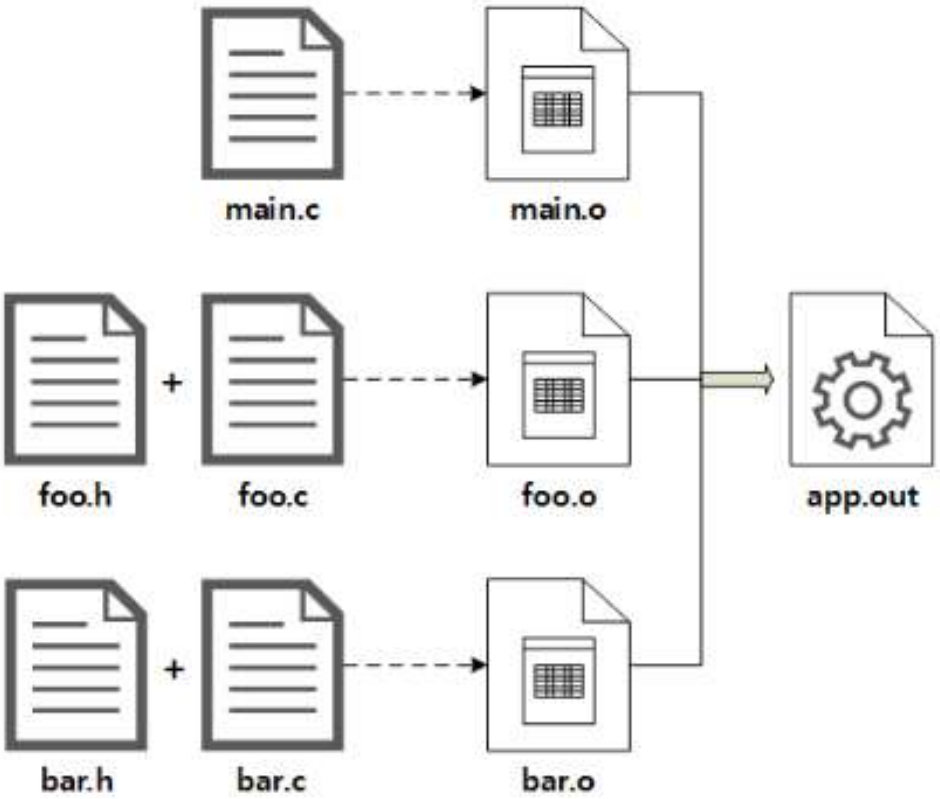
Rule== <Target>: <Dependencies>
<Recipe>

```
$ vi Makefile
---------------------------------
main: main.o foo.o bar.o
        gcc –o main main.o foo.o bar.o
```

Tab

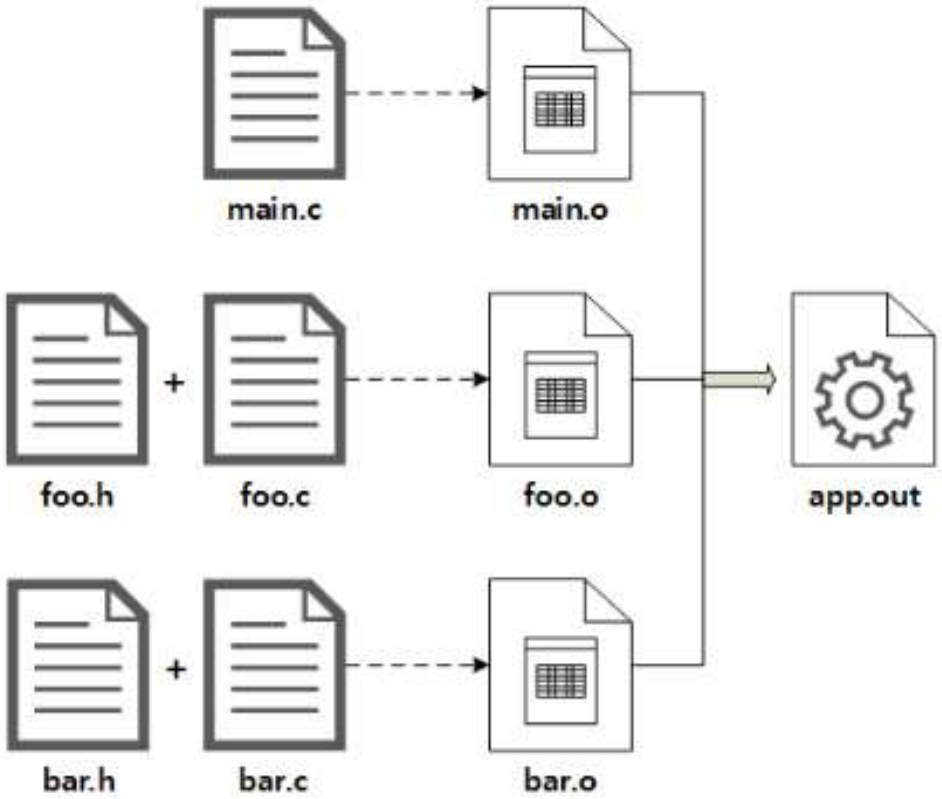
```
main.o: main.c
        gcc –c main.c

foo.o: foo.c
        gcc –c foo.c

bar.o: bar.c
        gcc –c bar.c
---------------------------------
$ make
$ ./main
```

Makefile (+매크로)



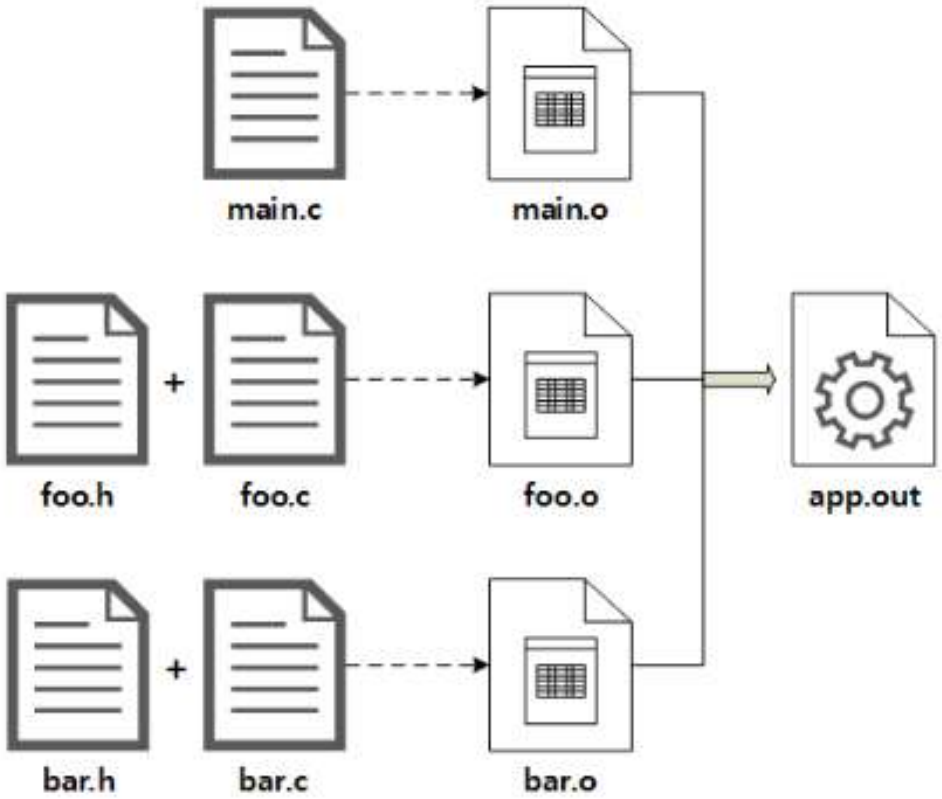

```
$ vi Makefile
--------------------------------
OBJECTS = main.o foo.o bar.o

main: $(OBJECTS)
        gcc –o main $(OBJECTS)

main.o: foo.h bar.h main.c
        gcc –c main.c

foo.o: foo.h foo.c
        gcc –c foo.c

bar.o: bar.h bar.c
        gcc –c bar.c
--------------------------------
$ make
$ ./main
```

Makefile (+매크로)



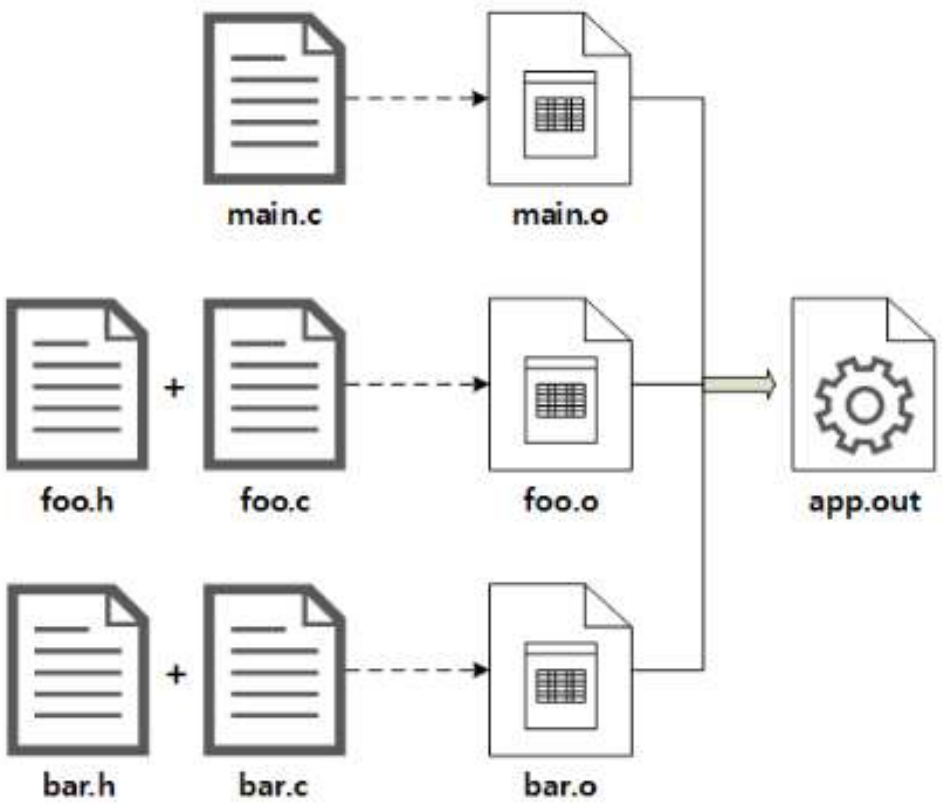

```
$ vi Makefile
--------------------------------
CC = gcc
CFLAGS = -W -Wall
OBJECTS = main.o foo.o bar.o

main: $(OBJECTS)
        $(CC) $(CFLAGS) –o $@ $^

main.o: foo.h bar.h main.c
        gcc –c main.c

foo.o: foo.h foo.c
        gcc –c foo.c

bar.o: bar.h bar.c
        gcc –c bar.c
--------------------------------
$ make
$ ./main
```

Makefile (+매크로)



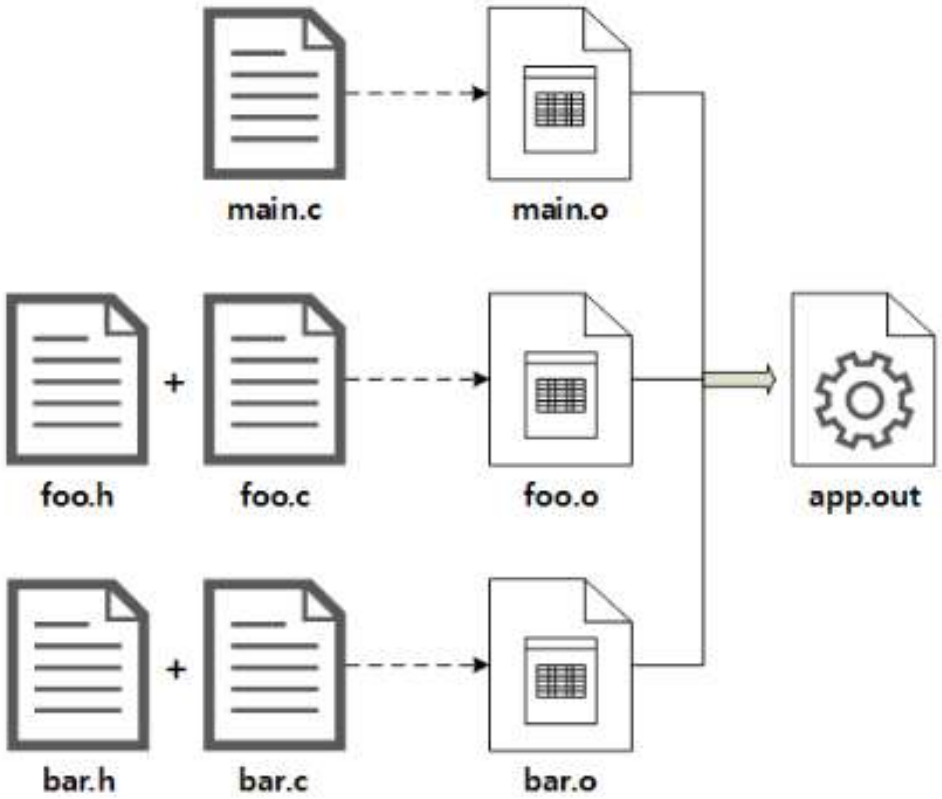

```
$ vi Makefile
--------------------------------
CC = gcc
CFLAGS = -W -Wall
OBJECTS = main.o foo.o bar.o

main: $(OBJECTS)
        $(CC) $(CFLAGS) –o main $(OBJECTS)

main.o: foo.h bar.h main.c
        gcc –c main.c

foo.o: foo.h foo.c
        gcc –c foo.c

bar.o: bar.h bar.c
        gcc –c bar.c
--------------------------------
$ make
$ ./main
```

Makefile (+dep)




```
$ vi Makefile
---------------------------------
CC = gcc
CFLAGS = -W -Wall
OBJECTS = main.o foo.o bar.o
SRCS = $(OBJECTS:.o=.c)

main: $(OBJECTS)
        $(CC) $(CFLAGS) –o main $(OBJECTS)

dep:
        gccmakedep $(SRCS)
---------------------------------
$ make dep
```

Makefile (+dep)




```
$ vi Makefile
---------------------------------
CC = gcc
CFLAGS = -W -Wall
OBJECTS = main.o foo.o bar.o
SRCS = $(OBJECTS:.o=.c)

main: $(OBJECTS)
        $(CC) $(CFLAGS) –o main $(OBJECTS)

dep:
        gccmakedep $(SRCS)
# DO NOT DELETE
main.o: main.c /usr/include/stdio.h …
foo.o: foo.c …
bar.o: bar.c …
---------------------------------
$ make
$ ./main
```

Makefile 변수

- Make 내장 변수
  - CC: 컴파일러
  - CFLAGS: 컴파일 옵션
  - OBJS: 중간 산물 object 파일 목록
  - TARGET: 빌드 대상(실행 파일) 이름
  - LDFLAGS: 링커 옵션
  - LDLIBS: 링크 라이브러리
- 자동 변수
  - $@: target 이름
  - $^: Dependencies 전체 목록
  - $?: Dependencies 중 변경된 것들의 목록
  - $<: Dependiecies 중 첫번째

Makefile 기본 패턴

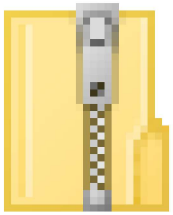
```
CC=<컴파일러>
CFLAGS=<컴파일 옵션>
LDFLAGS=<링크 옵션>
LDLIBS=<링크 라이브러리 목록>
OBJS=<Object 파일 목록>
TARGET=<빌드 대상 이름>

all: $(TARGET)

clean:
        rm –f *.o
        rm –f $(TARGET)

$(TARGET): $(OBJS)
        $(CC) –o $@ $(OBJS)
```

Makefile Clean

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

Exercise – Makefile 수정

- 주어진 makefile에 변수를 세 개 이상 추가 하여 추후 수정이 용이한 makefile로 만들기.
- Makefile 내용과 “make clean && make” 를 실행하여 빌드를 완료한 터미널 화면을 함께 스크린샷을 찍어 아이캠퍼스에 업로드.



make.zip

-c
-shared –o libmylibname.so | -L -l

강의 노트

#include “./path/to/my/header”

터미널 변수
~/.bashrc 파일에 아래 내용 추가하기
export LD_LIBRARY_PATH = $LD_LIBRARY_PATH:/my/library/path