



make

Written by 조진제
2006. 07. 28



make

- What is make?

make – GNU make utility to maintain groups of programs

: make utility 의 목적 – 프로그램 그룹 중에서 어느 부분이 새롭게 컴파일되어야 하는지를 자동적으로 판단하여 gcc와 같은 명령어를 이용하여 그들을 재컴파일하여 프로그램의 일관성을 유지토록 하는 것.

- Makefile의 내부구성(1)

- 목표(target), 의존관계(dependency), 명령(command)의 세 가지로 이루어진 기본적인 규칙(rule)이 나열
- You tell make what targets you want to build and then give rules explaining how to build them. You also specify dependencies that indicates when a particular target should be rebuilt.

```
target : dependency...  
        command...
```

```
...
```

```
...
```



make

- Makefile의 내부구성 (2)

- target : 명령 (gcc, ar, objcopy ...)이 수행되어져서 나온 결과 파일을 지정.
object file / execution file
- 명령부분에 정의된 명령들은 의존관계부분에 정의된 파일의 내용이 변경되었거나 목표부분에 정의된 파일이 없을 때 이곳에 정의된 것들이 차례대로 실행된다.
- 명령부분은 반드시 TAB글자로 시작한다. (make는 명령어인지 아닌지를 TAB으로 구분)

- Makefile의 예

```
# gcc -c main.c
# gcc -c read.c
# gcc -c write.c
# gcc -o test main.o read.o
write.o
```

```
test : main.o read.o write.o
        gcc -o test main.o read.o
        write.o

main.o : io.h main.c
        gcc -c main.c

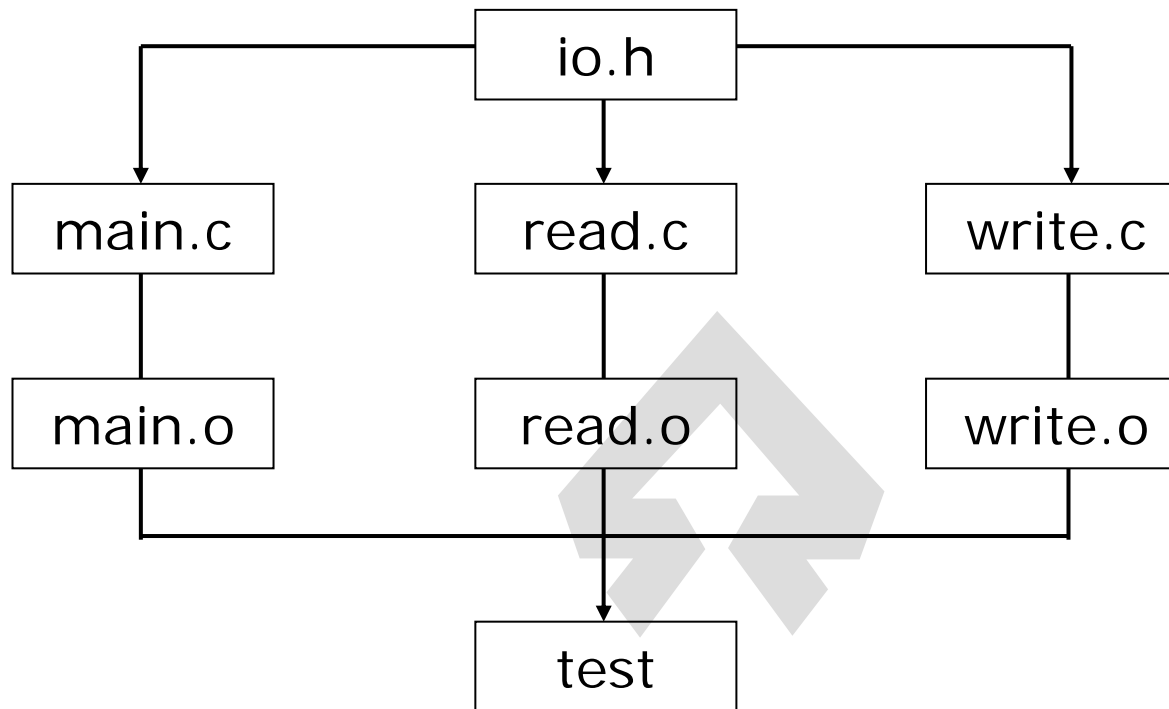
read.o : io.h read.c
        gcc -c read.c

write.o : io.h write.c
        gcc -c write.c
```



make

- Makefile의 의존성 관계





make

- Macro

: 특정 내용을 치환하여 프로그램을 단순화시키고자 할 때 이용.
반드시 \$(...)안에 넣어서 사용.

```
OBJECTS = main.o read.o write.o
```

```
test : $(OBJECTS)  
      gcc -o test $(OBJECTS)
```

```
main.o : io.h main.c  
        gcc -c main.c  
read.o : io.h read.c  
        gcc -c read.c  
write.o : io.h write.c  
         gcc -c write.c
```

- Pre-defined Macro

: make -p

: ASFLAG, AS, CFLAG, CC, CPPFLAGS, CXX, LDFLAGS, LD, ...



make

- Label

: 목표부분에 해당되는 부분이 레이블로 처리되어 간략화 될 수 있음.

```
OBJECTS = main.o read.o write.o
```

```
test : $(OBJECTS)  
      gcc -o test $(OBJECTS)
```

```
main.o : io.h main.c  
        gcc -c main.c
```

```
read.o : io.h read.c  
        gcc -c read.c
```

```
write.o : io.h write.c  
         gcc -c write.c
```

```
Clean :  
      rm $(OBJECTS)
```



make

- 확장자 규칙(Suffix Rule)

: 파일의 확장자를 보고 그에 따라 적절한 연산 동작을 수행시키는 규칙

```
.SUFFIXES = .c .o /* .c .o : $(CC) $(CFLAG) -c $< -o $@ */
```

: .c와 .o 확장자를 가진 파일들을 확장자 규칙에 의해서 처리될 수 있도록 해준다.
즉, 미리 정의된 .c 파일을 컴파일해서 .o파일을 만들어 내도록 자동적으로 동작.

```
.SUFFIXES = .c .o
OBJS = main.o read.o write.o
TARGET = test
test : $(OBJS)
    $(CC) -o $(TARGET) $(OBJS)

Clean :
    rm $(OBJS) $(TARGET)
```



make

- Internal Macro

: 사용자가 임의로 정할 수 없으며 매크로를 연산, 처리하는데 쓰이는 매크로.

\$* : 확장자가 없는 현재의 목표파일(target)

\$@ : 현재의 목표파일(target)

\$< : 현재의 목표파일(target)보다 더 최근에 갱신되어진 파일이름

\$? : 현재의 목표파일(target)보다 더 최근에 갱신되어진 파일이름

```
main.o : main.c io.h
        gcc -c $*.c

test : $(OBJS)
        gcc -o $@ $(OBJS)

.c.o :
        gcc -c $<
```




make

- Automatic dependency

target : dependency

....

....

-> target이 어느 파일에 의존하고 있는지 표시해 주는 정보의 역할

- gccmakedep(or gcc -M)

: 어떤 파일의 의존관계를 자동으로 조사해서 makefile의 뒷부분에 추가하는 유틸리티

```
.SUFFIXES = .c .o
CFLAGS = -O2 -g

OBJS = main.o read.o write.o
SRCS = $(OBJS:.o=.c)
TARGET = test

test : $(OBJS)
    $(CC) -o $(TARGET) $(OBJS)

dep :
    gccmakedep $(SRCS)
```



make

- Recursive make

: 여러 개의 서브시스템(서브 디렉토리)이 전체 시스템을 구성할 경우 각 서브 디렉토리마다

Makefile이 존재

```
.SUFFIXES = .c .o
CC = gcc
CFLAGS = -O2 -g

all : DataBase Test

DataBase :
    cd db ; $(MAKE) /* db로 이동하여 make 실행 $(MAKE) -C db 도 가능*/

Test :
    cd test ; $(MAKE) /* test로 이동하여 make 실행 $(MAKE) -C test 도 가능
*/
```



make

- Make 중요 옵션 정리

- C dir : Makefile을 계속 읽지 말고 우선 dir로 이동하라. 순환 make에 이용
- d : Makefile을 수행하면서 나오는 각종 정보를 모두 출력
- f file : file에 해당하는 파일을 Makefile로서 취급
- h : option에 관한 도움말 출력
- p : make에서 내부적으로 설정되어 있는 값들을 출력
- k : make 실행도중 에러가 발생해도 멈추지 말고 계속 진행.