부록 8: Makefile 만들기

Makefile 만들기 목차

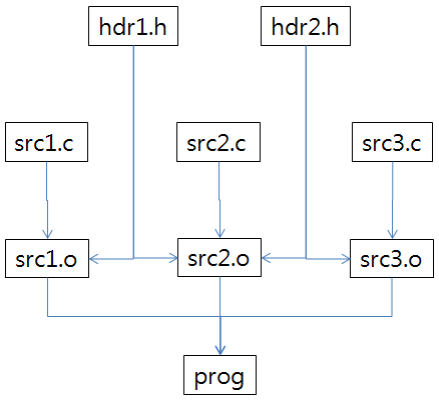
1. Makefile 내부 구조 385
2. 매크로(Macro) 385
3. 레이블(Label) 386
4. 미리 정해져 있는 매크로(Pre-defined macro) 386
5. 확장자 규칙(Suffix rule) 387
6. 긴 명령어를 여러 줄에 걸쳐 표시하기 388
7. 매크로 치환 (Macro substitution) 389
8. Makefile 작성의 가이드라인 389

1. Makefile 내부 구조

```
목표 ... : 의존관계 ...
        명령
        ...
```

- 목표(target): 명령이 수행이 되어서 나온 결과 파일 지정한다. 레이블로도 사용 가능하다. Makefile을 작성하고 그 디렉토리상에서 쉘에 그냥 make만 입력하면 Makefile의 내용을 살펴보다가 첫 번째 목표파일에 해당되는 명령을 실행한다.
- 명령(command): 의존관계 부분에 정의된 파일의 내용이 바뀌었거나, 목표 부분에 해당하는 파일이 없을 때 차례대로 실행되는 부분이다. 쉘에서 사용할 수 있는 모든 명령어 사용 가능하다. (명령 부분은 꼭 tab 글자로 시작해야 한다.)

의존관계(dependecy): 다음 그림과 같이 목표 부분에 해당하는 파일을 만들 때 꼭 필요한 파일들이다.



위 그림에서 prog을 만들기 위해선 src1.o, src2.o, src3.o가 필요하고, src1.o를 만들기 위해선 src1.c, hdr1.h가 필요하다. 이러한 관계를 의존관계라고 한다.

2. 매크로(Macro)

매크로의 정의는 프로그램을 작성할 때 변수를 지정하는 것처럼 하면 된다. 그리고, 매크로를 사용하기 위해서는 $(...)을 이용하면 된다.

정의	매크로이름 = 대체표현 ...
사용	$(매크로이름)

매크로를 사용하는 Makefile	매크로를 사용하지 않는 Makefile
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	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

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

실행결과

```
$ make(혹은 make test)
gcc -c main.c
gcc -c read.c
gcc -c write.c
gcc -o test main.o read.o write.o
```

3. 레이블(Label)

목표 부분이 레이블로 사용될 때는 의존관계 부분이 없어도 된다.

Makefile

```
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 $(OBECTS) test
```

실행결과

```
$ make clean
rm main.o read.o write.o test
```

4. 미리 정해져 있는 매크로(Pre-defined macro)

```
CC = cc (= gcc)
CFLAGS =
```

CC는 C 컴파일러에 대한 매크로이고, CFLAGS는 C 컴파일러의 옵션에 대한 매크로이다. 이들은 사용자에 의해 재정의 가능하다.

Makefile

```
CC = gcc
CFLAGS = -g -c

OBJECTS = main.o read.o write.o
TARGET = test

$(TARGET) : $(OBJECTS)
        $(CC) -o $(TARGET) $(OBJECTS)

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

실행결과

```
$ make
gcc -g -c main.c -o main.o
gcc -g -c read.c -o read.o
gcc -g -c write.c -o write.o
gcc -o test main.o read.o write.o
```

위의 예에서, main.o와 read.o, write.o 각각에 대해 .c 파일을 .o 파일로 바꿔주는 코드가 빠져있지만 실행결과에선 정상적으로 작동하고 있음을 확인할 수 있다. 미리 정해져 있는 매크로들은 make 파일 내부에서 위 실행결과에서와 같은 작업을 하는데 이용되기 때문에, CFLAGS를 설정하는 것만으로도 .c 파일을 .o 파일로 바꿔주는 코드를 작성할 필요가 없어진다.

5. 확장자 규칙(Suffix rule)

확장자 규칙이란 파일의 확장자를 보고 그에 따라 적절한 연산을 수행시키는 규칙을 말한다. 앞서 .c 파일을 .o 파일로 자동으로 바꿔주는 부분이 바로 확장자 규칙 덕분이다. 이를 직접 명시하면 다음과 같다.

```
.SUFFIXES : .c .o
```

위의 표현은 .c 파일과 .o 파일 확장자 규칙에 따라 처리하도록 즉, .c 파일을 .o 파일로 바꾸는 표현이다. 이 처리는 다음의 작업을 내부적으로 처리하여 동작하게 된다.

```
.c.o :
        $(CC) $(CFLAGS) -c $< -o $@
```

또는

```
%.o : %.c
        $(CC) $(CFLAGS) -c $< -o $@
```

여기서 %는 매치되는 텍스트로 대치된다. 또한 $<, $@는 내부 매크로(internal macro)들로, 일반 매크로처럼 임의로 정해서 사용할 수는 없다.

내부 매크로(Internal macros)	
$*	확장자가 없는 현재의 목표 파일.
$@	현재의 목표 파일. 콜론의 왼쪽에 오는 패턴.
$<	현재의 목표 파일보다 더 최근에 갱신된 파일 이름. 콜론의 오른쪽에 오는 패턴.

Makefile

```
.SUFFIXES : .c .o

OBJECTS = main.o read.o write.o
SRCS = main.c read.c write.c

CC = gcc
CFLAGS = -g -c

TARGET = test

$(TARGET) : $(OBJECTS)
	$(CC) -o $(TARGET) $(OBJECTS)

clean :
	rm -rf $(OBJECTS) $(TARGET)

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

실행결과

```
% make
gcc -g -c main.c -o main.o
gcc -g -c read.c -o read.o
gcc -g -c write.c -o write.o
gcc -o test main.o read.o write.o
```

6. 긴 명령어를 여러 줄에 걸쳐 표시하기

'\' 문자를 사용하여 한 줄의 명령어를 여러 줄에 걸쳐 표시할 수 있다. 예를 들어, 아래 두 명령어는 같다.

```
OBJS = shape.o rectangle.o circle.o line.o bezier.o
```

```
OBJS = shape.o \
rectangle.o \
circle.o \
```

```
line.o \
bezier.o
```

7. 매크로 치환 (Macro substitution)

매크로의 일부 내용을 수정하여 사용가능하다. 방법은 다음과 같다.

```
$(매크로이름:치환대상=치환내용)
```

다음의 매크로 치환을 사용한 경우와 사용하지 않은 경우 모두 SRCS는 내용이 같다.

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

```
OBJS = main.o read.o write.o
SRCS = main.c read.c write.c
```

8. Makefile 작성의 가이드라인

Makefile

```
# 매크로 정의 부분
.SUFFIXES : .c .o

CC = gcc

INC =
LIBS =
CFLAGS = -g $(INC)

SRCS =
OBJS = $(SRCS:.c=.o)

TARGET =

# 명령어 정의 부분
all : $(TARGET)

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

clean :
	rm -rf $(OBJS) $(TARGET)

# 의존관계 부분
```

- INC: 헤더 파일의 경로
- LIBS: 링크할 때 필요한 라이브러리
- CFLAGS: 컴파일에 필요한 옵션들
- SRCS: 소스 파일들
- TARGET: 링크 후 생성될 실행 파일의 이름