2009년 06월 26일

컴파일을 더더더 쉽게. MACRO와 SUFFIX

컴파일을 더더더 쉽게 하려면, MACRO를 이용한 방법이 있는데, 잘 살펴보면 재미있습니다. MACRO는 컴퓨터에서 이용되는 분야라면 어디든 사용되는 개념이니, Makefile에서도 예외라고 할 수 없습니다. Makefile에서는 MACRO를 이용하여, Makefile을 만드는 작업조차 간단하게 만들었는데, 이 것을 잘 모르면, Makefile을 분석하거나, 더더더 편하게 만들기
힘드니까, 이번 기회에 깔끔하게 정리할 필요가 있겠습니다.

우선 MACRO는 다음과 같이 선언합니다. 예)를 들어,

CC = tcc

선언은 이렇게 하고 사용은 어떻게 하느냐!

$(CC) 로 사용하는 거죠. 이렇게 하면, $(CC)라고 표현되어 있는 부분은 모두 tcc로 바뀌게 됩니다. 이외 여러 가지 ?=, :=, += 로 선언하는 방법들도 있긴 합니다만, 사족으로 넘깁시다.

자, 그러면 이 MACRO를 이용하여, spaghetti.mak를 조금 더 간단하게 수정해 보겠습니다.

1. 간단한 MACRO로 간단하게 만들기

spaghetti.mak ----------

```
TARGET = spaghetti.bin
ELF_TARGET = spaghetti.elf
BINTOOL = fromelf
CC = tcc

$(TARGET) : $(ELF_TARGET)
 $(BINTOOL) -bin -o $(TARGET) $(ELF_TARGET)

$(ELF_TARGET) : spaghetti.o manupulation.o
 armlink -elf -o $(ELF_TARGET) spaghetti.o manupulation.o

spaghetti.o : spaghetti.c
 $(CC) -c spaghetti.c

manupulation.o : manupulation.c
 $(CC) -c manupulation.c
```

오, 먼가 조금 깜찍하게 깔끔해 진 것 같기도 합니다. 내친 김에 조금 더 간추려 볼까요?

2. Link 없애기

spaghetti.mak ----------

```
TARGET = spaghetti.bin
ELF_TARGET = spaghetti.elf
BINTOOL = fromelf
CC = tcc

$(TARGET) : $(ELF_TARGET)
 $(BINTOOL) -bin -o $(TARGET) $(ELF_TARGET)

$(ELF_TARGET) : spaghetti.o manupulation.o
 $(CC) -o $(ELF_TARGET) spaghetti.o manupulation.o

spaghetti.o : spaghetti.c
 $(CC) -c spaghetti.c

manupulation.o : manupulation.c
 $(CC) -c manupulation.c
```

이번엔 Link 과정이 없어졌습니다요. 하지만, Link과정이 없어졌다기 보다는 tcc가 -o optoin을 통해서 곧바로 link까지 진행하니까 마치 없어진 것처럼 보이지만, 우리 한가지를 더 간단하게 했네요. 진전입니다. 진전 더 간단히 갈 때까지 가 볼 것입니다.

조금 더 간추려 봅시다. 점점 더 암호화 해서 가장 간단하게 만들어 내는 것이 우리 목적입니다. 자 이제부터 이용할 것이 우리 중복해서 사용하는 것들이 너무 많습니다. file name들인데, 이 녀석들도 어떻게든 줄이고 싶습니다. 특히나 만들고 싶은 output을 만들기 위해 시키고 싶은 일에는 반드시 file 이름들이 중복되고 있습니다.

이럴 때 이용하는 것이 바로 $@, $^ 들인데, 미리 약속된 Macro들입니다. $@은 현재의 만들고 싶은 output이름이고, $^는 재료들의 이름입니다. 이제부터 정신 똑바로 차리고 가보시죠. 길 잃으시면 안 되요.

3. 재료와 Output 이름도 간단하게 $@, $^

spaghetti.mak ----------

```
TARGET = spaghetti.bin
ELF_TARGET = spaghetti.elf
BINTOOL = fromelf
CC = tcc

$(TARGET) : $(ELF_TARGET)
 $(BINTOOL) -bin -o $@ $^

$(ELF_TARGET) : spaghetti.o manupulation.o
 $(CC) -o $@ $^

spaghetti.o : spaghetti.c
 $(CC) -c @^

manupulation.o : manupulation.c
 $(CC) -c @^
```

어, 점점 더 간단해져 가고 있습니다. 아까 보다 훨씬 짧아지고 간단해 졌지요? 자, 우리는 make가 대단히 smart하다는 것을 익히 느껴가고 있으니까, 더 간단하게 한번 해볼까요? 재료들 이름 쓰는 것도 귀찮으니까, 자자~ 이렇게 가보겠습니다.

spaghetti.mak ----------

```
TARGET = spaghetti.bin
ELF_TARGET = spaghetti.elf
BINTOOL = fromelf
CC = tcc
OBJECTS = spaghetti.o manupulation.o

$(TARGET) : $(ELF_TARGET)
 $(BINTOOL) -bin -o $@ $^

$(ELF_TARGET) : $(OBJECTS)
 $(CC) -o $@ $^
```

어? 더 암호화 된 모습니다. 이걸로도 정말 makefile이 제대로 작동할 까요? 제대로 작동합니다. 정말로 멋진 모습입니다. 결국 OBJECTS (재료들)만 선언해 주고, compile은 Macro 한 줄로 모든 걸 해결해 버렸네요.

여기에서 마지막줄의 $(CC) -o $@ $^ 요 부분에서 보면 어쨌거나 .o로 된 녀석들은 CC를 이용해서 .c를 만들어 내는 건 공통이겠죠? 그러니까 확장자 규칙이란 걸 쓰면 더욱 간단하게 할 수 있습니다.

4. 반복되는 확장자 관계도 간략하게~ (.c → .o), SUFFIX rule 이용하기

spaghetti.mak ----------

```
TARGET = spaghetti.bin
ELF_TARGET = spaghetti.elf
BINTOOL = fromelf
CC = tcc
OBJECTS = spaghetti.o manupulation.o

$(TARGET) : $(ELF_TARGET)
	$(BINTOOL) -bin -o $@ $^

$(ELF_TARGET) : $(OBJECTS)

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

오호라 마지막에 처음 보는 %가 나왔지요? 이런 건 .o가 나오면 같은 이름의 .c를 (CC)를 이용해서 컴파일 해라 뭐 이런 의미로 사용되는 겁니다. 확장자규칙이라고 부르는데요. 이렇게 하면 심지어. 다른 확장자를 가진 녀석들도 한꺼번에 OBJECT에 넣어놓고 확장자 규칙만 추가하면 한꺼번에 $(ELF_TARGET)을 만들어 낼 수 있는 게지요. 3번 예에서 실은 요 rule이 숨어 있는 거지요. 일반적인 make는 .c를 .o로 만드는 일이 대부분 이니까, Default make 설정으로 이걸 갖고 있어요. 그러니까, 이런 식으로 다시 선언해 주면 좀 더 명확해 질 수 있으니까, 만드는 사람들에 따라서 이걸 명확하게 선언해 주는 사람들도 많이 있습니다.

이런걸 이용해서, .c뿐만 아니라, 자신만의 확장자 rule을 이용해서, 다른 명령을 만들 수도 있는 거지요. 예를 들어 자신만의 script 파일을 만들었는데, 그 script를 해석하는 tool을 자기가 만들었다고 칩시다. script file의 확장자는 .m 이고, tool의 이름은 mytool이라고 치고요. 장난친 후의 file의 확장자가 .out이라고 한다면요.

Compile중에 이런 일을 좀 했으면 좋겠는데 싶은 상황에

```
%.out : %.m
	mytool $@ $<
```

요렇게 만들어 놓으면 재료 중에 .m file을 만나면 mytool을 이용해서 .m을 .out으로 만들어 준다니깐요. 음화화. 한가지 유의할 사항은 Suffix rule에서의 재료는 @^대신 @<를 쓴다는 거 잊지 마세요~

5. 어떤 한 디렉토리를 통째로 한꺼번에 컴파일 하고 싶을 때

한가지 더, 이제는 OBJECTS를 선언해 주는 것도 귀찮아져 버렸습니다. 이럴 때는 또 더 간단하게 해결할 수 있는 방법이 있습니다.

```
SRC = $(wildcard *.c)
OBJECTS = $(SRC:.c=.o)
```

마지막으로 Spaghetti.mak의 재료들이 모두 .c이고, output이 모두 .o라면, 이렇게 make위에 선언만 해준다면, 현재 Directory의 모든 c file을 SRC로 등록을 한 후, MACRO 안에서 .c만 .o로 모두 바꾸어 OBJECTS로 선언해 줍니다. MACRO의 새로운 용법이네요. 그렇다면~

spaghetti.mak ----------

```
TARGET = spaghetti.bin
ELF_TARGET = spaghetti.elf
BINTOOL = fromelf
CC = tcc
SRC = $(wildcard *.c)
OBJECTS = $(SRC:.c=.o)

$(TARGET) : $(ELF_TARGET)
 $(BINTOOL) -bin -o $@ $^

$(ELF_TARGET) : $(OBJECTS)
 $(CC) -o $@ $^
```

더 더욱이 간단해 졌습니다. 우리가 이렇게 까지 달려올 줄 누가 알았겠습니까. 이런 기법을 이용한다면, source file이 수만 개가 되더라도 간단하게 compile할 수 있겠습니다. 자, 다 덤벼 보시지.

그 결과는 다음과 같이 모두 같습니다. 냐하~

```
E:₩>make
tcc     -c -o manupulation.o manupulation.c
tcc     -c -o spaghetti.o spaghetti.c
tcc -o spaghetti.elf manupulation.o spaghetti.o
fromelf -bin -o spaghetti.bin spaghetti.elf
```

makefile, 컴파일, macro, 만들기, 간단

Linked at at 2009/06/26 01:16

... ocator ⓖ Makefile은 뭘하는 녀석일까~ ⓡ 컴파일을 더더더 쉽게. MACRO와 SUFFIX ⓢ 조금 더 Make 테크닉들 ⓣ Make option들 4) ... more