Centos 7 의 \*.c파일과 mysql연결파일을 갖은 \*.c를 한번에 실행하기 위한 make파일 만들기

mysql컴파일 방법

gcc –o 실행파일 $(mysql\_config –-cflags) \*.c $(mysql\_config –-libs)

make 와 makefile은?

shell에서 컴파일을 해봤다면, make명령어로 컴파일을 실행하는 경우를 자주 봤다. Makefile이 있는 DIR에서 make만 치면 컴파일이 실행됨.

Make는

파일 간의 종속관계를 파악하여 makefile(기술파일)에 적힌 대로 컴파일러에 명령하여 shell 명령이 순차적으로 실행될 수 있게 해줌

Makefile도 같은 역할을 함.

Make, makefile을 사용하는 이유는 3가지 장점 존재

각 파일에 대한 반복적 명령의 자동화로 인한 시간 절약

프로그램의 종속 구조를 빠르게 파악 할 수 있으며 관리가 용이

단순 반복 작업 및 재작성을 최소화

예제

|  |
| --- |
|  |
| diary\_exe라는 실행파일 생성 |

1. diary.h 헤더파일 만들기

3개의 c파일이 include 할 헤더파일을 생성해보자.

vi diary.h = \*.c …과 같은 c파일이 참조할 헤더파일을 생성

|  |
| --- |
| #include <stdio.h>  void memo();  void calendar(); |
| diary.h |

2. diary.h를 사용할 재료파일(memo.c, calendar.c, main.c)만들기

|  |  |  |
| --- | --- | --- |
| #inlcude “diary.h”  void memo() {  printf(“Func memo!\n”);  } | #inlcude “diary.h”  void calendar() {  printf(“Func calendar!\n”);  } | #inlcude “diary.h”  void main() {  printf(“Func main!\n”);  } |
| memo.c | calendar.c | main.c |

3.생성된 파일 확인

위의 모든 파일을 생성했다면 ls로 확인

이제부터 2가지 방법으로 컴파일

기본적인 컴파일

gcc –c main.c

gcc –c memo.c

gcc –c calendar.c

obj파일 확인하고 diary\_exe파일 실행

gcc –o diary\_exe main.o memo.o calendar.o

./diary\_exe

Make를 이용한 컴파일 과정

Makefile의 구조

-목적파일(target) : 명령어가 수행되어 나온 결과를 저장하는 파일

-의존성(dependency) : 목적파일을 만들기 위해 필요한 재료

-명령어(command) : 실행되어야 할 명령어들

-매크로(macro) : 코드를 단순화 시키기 위한 방법

Makefile의 기본구조

|  |
| --- |
|  |
|  |

Makefile의 작성규칙

|  |
| --- |
|  |

Makefile 예제 따라하기

1. 기존에 \*.h, \*.c(3개 파일)을 따라함.

2. Makefile작성

vi Makefile

|  |
| --- |
|  |

3. make 명령어 입력

자동적으로 Makefile을 실행

advanced Makefile

지금까지는 그냥 컴파일 과정을 묶은 것 외에 특별한 것이 없음.

이제 매크로(macro)를 사용할 것임

Makefile 매크로 사용 예제

위의 코드에서 중복되는 파일 이름을 특정 단어로 치환

마치 C언어의 #define과 비슷한 원리

|  |
| --- |
|  |
| 메크로를 참조 할 때는 소괄호나 중괄호 둘러싸고 앞에 ‘$’를 붙임.  탭으로 시작해서는 안되고 ,:,=,#,”” 등은 매크로 이름에 사용할 수 없음.  매크로는 반드시 치환될 위치보다 먼저 정의 되어야 함.  여기에서 –W –Wall는 컴파일 시 컴파일이 되지 않을 정도의 오류라도 출력되게 하는 옵션  Make clean  vi Makefile //매크로 사용 예제처럼 수정  ./diary\_exe  로 전과 같은 결과가 나오는지 확인하기 |

Makefile 개선2 : 내부 매크로 사용

|  |
| --- |
|  |
| 내부 매크로를 사용하여 코드가 굉장히 단순해짐  내부 매크로 해석  “$@” : 현재 타겟의 이름  “$^” : 현재 타겟의 종속 항목 리스트  1. gcc 컴파일러 사용  2. 사소한 오류까지 출력  3. 최종 타겟 파일은 diary\_exe  4. OBJECT로 정의할 파일들은 memo.o, main.o, calendar.o  5. all은 현재는 사용하지 않았지만 타겟 파일이 여러 개 일 때 사용됨.  6. 타겟 파일을 만들기 위해 OBJECT들을 사용(단 OBJECT파일이 없다면 OBJECT파일과 이름이 동일한 C파일을 찾아 OBJECT파일을 생성)  7. gcc –o diary\_exe memo.o main.o calendar.o 와 동일  8. 더미타겟 |

참조

[**http://bowbowbow.tistory.com/12**](http://bowbowbow.tistory.com/12)