1. 컴파일 실습(실습 3) 시간에 작성한 모든 소스파일이 있어야 하고 컴파일 에러가 없어야 이번 실습을 할 수 있다. 못한 학생들은 컴파일 실습을 먼저 끝내고 이번 실습을 하기 바란다. 앞으로는 터미널 창을 항상 두 개 실행하여 각각에서 로그인 한 후 한쪽 터미널에서는 vi 만 실행하고(향후 vi 창이라 함), 다른 터미널(명령창이라 함)에서는 명령어들만 실행하도록 한다.
2. 명령창에서 자신의 홈 디렉토리 밑에 up 밑에 pr4 디렉토리를 만든 후, pr4 디렉토리에서 아래의 모든 실습을 실시하라.

   $ cd [enter] **$ cd up** **$ pwd**
   $ ls (이미 pr4가 있으면 mkdir 하지 마라.)
   $ mkdir pr4 **$ ls** (pr4 존재해야 함)
   $ cd pr4 **$ pwd** (pr4여야 함)

3. 명령창에서 pr4에서 pr3에 있는 모든 소스 파일을 pr4로 복사하라.

   $ cp ../pr3/*.c . (.을 정확히 줄 것)
   (상위 디렉토리 밑에 pr3 밑에 *.c를 현재 디렉토리 .로 복사)
   $ ls -l (모든 소스 파일이 존재해야 함)

   경고: 위 파일들을 정상적으로 복사하지 않으면 향후 모든 실습이 정상적으로 수행되지 않는다.

   ---

4. 다른 터미널인 vi창(명령창 아님)에서 강의노트의 [Makefile 예제 1]과 동일한 "Makefile" 파일을 만들어라.

   $ cd ~/up/pr4 (pr4로 이동) **$ pwd** (pr4여야 함)
   $ vi Makefile ([Makefile 예제 1]을 보고 입력하라.**)**
   // 입력이 끝난 후 :w[enter]로 저장만 하라.

5. 명령창에서 강의노트의 [실행 예제 1]을 보고 동일하게 실행시켜 보라. make가 처음부터 제대로 실행되지 않으면 Makefile의 내용을 다시 확인하라.

6. 명령창에서 다시 make를 실행 해 보라.

   $ make
   make: 'all'을 위해 할 일이 없습니다.
   (어떤 파일도 수정되거나 삭제되지 않았기에 수행할 것이 없다는 메시지다. 에러가 아님.)

7. vi창에서 [Makefile 예제 2]의 내용을 Makefile에 추가하고, 명령창에서 [실행 예제 2]와 동일하게 실행시켜라.

8. vi창에서 [Makefile 예제 3]의 내용을 Makefile에 추가하고, 명령창에서 [실행 예제 3]과 동일하게 실행시켜라.

9. vi창에서 [Makefile 예제 4]의 내용을 Makefile에 추가하고,
   명령창에서 [실행 예제 4]과 동일하게 실행시켜라.

10. make는 Makefile에 링크 룰이 있을 경우 링크 룰에 기술된 각각의 목적어 파일에 대해 링크 룰 아래 쪽을 찾아 해당 목적어 파일을 생성하는 컴파일 룰을 찾아 컴파일을 먼저 한다. 예를 들어, 만약 add.c 소스파일이 수정되었다면 add.o : add.c 컴파일 룰에 의해 컴파일이 먼저 될 것이다. 모든 목적어 파일에 대해 컴파일 룰이 모두 실행되고 나면 링크 룰이 실행된다. 이때 앞 전의 컴파일 룰들에 의해 목적어 파일들 중 하나라도 수정되었다면, 링크 룰에 의해 실행파일이 새로 생성될 것이다.

11. vi창에서 [Makefile 예제 5]의 내용을 Makfile에 추가하고, 명령창에서 [실행 예제 5]와 동일하게 실행시켜라.

12. 명령창에서 지금까지의 실습 내용을 담은 Makefile을 아래처럼 복사하라. Makefile.old은 추후 검사 대상이니 지금까지의 내용을 잘 보관하기 바란다. **이후 Makefile에 변경되는 내용은 Makefile.old에 반영하지 마라**.

    $ cp Makefile Makefile.old

13. 이제 매크로 상수의 활용법을 실습해 볼 것이다. vi창에서 [Makefile 예제 6]을 참조하여 기존의 Makfile을 수정하고, 명령창에서 [실행 예제 6]과 동일하게 실행시켜라.

14. 이제부턴 새로운 head 디렉토리를 만들어 기존의 모든 소스 파일과 Makefile 파일을 복사할 것이다. 명령창에서 다음 명령을 실행하라.

    $ pwd (pr4여야 함, 아니면 $ cd ~/up/pr4 하라.)
    $ ls (이미 head가 있으면 mkdir 하지 마라.)
    $ mkdir head **$ ls** (head 존재해야 함)
    $ cd head **$ pwd** (head여야 함)
    $ cp ../*.c ../Makefile . (.을 정확히 줄 것)

15. 기존의 vi 창에서 실행하던 vi에서 빠져 나와 아래처럼 head 디렉토리로 이동하라. 이제 이 디렉토리에서 이후의 모든 파일 수정 작업을 수행하라.

    $ cd head **$ pwd** (head여야 함)
    처음 로그인 했을 때 또는 어느 디렉토리에 있든 아래처럼 입력하여 head로 옮겨 갈 수 있다.
    $ cd ~/up/pr4/head **$ pwd** (head여야 함)

16. 이제 **head** 디렉토리에서 헤드 파일을 만들고 소스 파일에서 이를 include하도록 하자. 강의노트의 [헤드 파일을 만들고 include하기]를 참조하라.

1) vi 창에서 $ vi t1.c하여 t1.c에서 아래의 함수 선언을 삭제하라.

```
extern int add(int, int);
extern int sub(int, int);
extern int mul(int, int);
extern double dvd(int, int);
```

이 문장들은 헤드 파일로 옮겨지고, 대신에 그 헤드 파일을 include할 예정이다.

2) vi 창에서 add.h, sub.h, mul.h, dvd.h 헤드 파일을 생성하고, 위에서 삭제된 각 extern 함수 선언문을 해당 헤드 파일에 추가하라. 예를 들어, add.h에는 아래 extern 함수 선언문 하나만 있어야 한다.

```
extern int add(int, int);
```

3) 각 함수를 **정의하는** 소스 파일은 해당 헤드 파일을 include해야 한다. 예를 들어, add.c에는 아래처럼 add.h를 include해야 한다. 사용자가 만든 헤드 파일을 include할 때는 <add.h>대신 "add.h"를 사용해야 한다.

```
#include "add.h"
```

마찬가지로 각 함수를 **정의하고** 있는 sub.c, mul.c, dvd.c 소스 파일에 각자의 헤드 파일을 include 시켜라.

4) 또한 헤드 파일에 있는 함수를 **호출하는** 소스 파일에서도 그 헤드 파일을 include해야 한다. 여기선 t1.c에서 모든 함수들을 호출하므로, t1.c의 #include <stdio.h> 다음 행에 4개의 헤드 파일을 include하라.

5) 이상을 정리하면 하나의 함수를 정의하는 소스파일을 만들면(예를 들어, add.c), 그와 짝을 이루는 헤드파일(add.h)을 만들고 그 파일 안에 함수 add()에 대한 extern 선언문을 저장한다. 그리고 해당 함수(add())를 정의한 add.c와 이 함수를 호출하는 소스파일(t1.c)에 헤드파일을 include 시킨다. 그러면 호출하는 쪽(t1.c)에선 별도의 extern 함수선언 없이 바로 함수를 호출할 수 있다. 굳이 함수를 정의한 곳(add.c)에서도 헤드파일을 include하는 이유는 헤드파일에 있는 함수선언과 add.c에 정의되어 있는 함수의 리턴 값과 함수인자의 데이터 타입이 일치하는지를 체크하기 위함이다. 만약 헤드파일에 다른 상수나 자료구조체가 정의되어 있다면, 이 헤드파일만 include 함으로써 굳이 add.c에 상수나 구조체를 새로 정의할 필요가 없어진다. 즉, 함수 호출하는 쪽과 정의하는 쪽에서 상수 및 구조체를 공유할 수 있게 된다.

17. 위의 헤드 파일들이 수정되었을 때, 영향을 받는 소스 파일들이 자동으로 컴파일 되어야 한다. 이를 위해 vi 창에서 [Makefile 예제 7]을 참조하여 기존의 Makfile을 수정하고, 명령창에서 [실행 예제 7]을 실행하라.

18. vi창에서 다시 t1과 t2도 test1처럼 10 % 3(10을 3으로 나눈 나머지 값)과 같은 수식을 처리하는 기능을 삽입하여라.

1) mod.c 라는 소스 파일을 만들고 mod() 함수를 구현하라. test1.c의 mod() 함수와 동일하게 만들어라. 또한 mod.h를 include 시켜라.

2) 다른 헤드 파일을 참조하여 mod.h 라는 헤드 파일을 만들어라.

3) t1.c에 앞 부분에 mod.h 헤드파일을 include 시켜라. 또한 t1.c의 main() 함수에 mod() 함수를 호출하는 코드를 삽입하라. test1.c의 main() 함수 참조하라.

4) Makefile 내에 맨 앞 부분에 있는 매크로 상수인 **SOURCES**, **OBJECTS**, **HEADS**의 행 끝에 각각 mod.c, mod.o, mod.h를 추가하라. 또한 컴파일하는 룰인 dvd.o: dvd.c dvd.h 룰 아래 쪽에 mod.c를 컴파일하여 mod.o를 생성하는 컴파일 룰을 삽입하라.

5) 명령창에서 **$make**를 실행하여 t1과 t2를 새로 생성하라.

6) 생성된 실행파일 t1 또는 t2를 아래처럼 실행시켜 본다. 명령창에서

```
$ t1 [enter] 또는 $ t2 [enter]
Expression: 7 % 3[enter]
```

// 결과 값은 1이어야 함

19. 모든 실습이 완료 되었다면, 명령창에서 make clean 후 새로 make하여 모든 목적어 파일 및 실행 파일을 다시 생성시켜라.

20. 명령 창에서 다음을 실행하여 모든 것이 정상인지 확인하라.

```
$ eshw 4      // 에러가 발생한 항목에 대해서만 보여주면서 확인
$ fshw 4      // 모든 항목을 보여 주면서 확인
$ progtest 4
```

// 실습 4에 대해 실행파일인 test1, test2, t1, t2를 대신 실행하고, 다양한 입력을 대신 입력하여(정상적인 입력과 잘못된 입력) 프로그램이 정상적으로 작동하는지 테스트함