Open Source Software(ECE20019)

Makefile

장 소 연 교수 (전산전자학부)

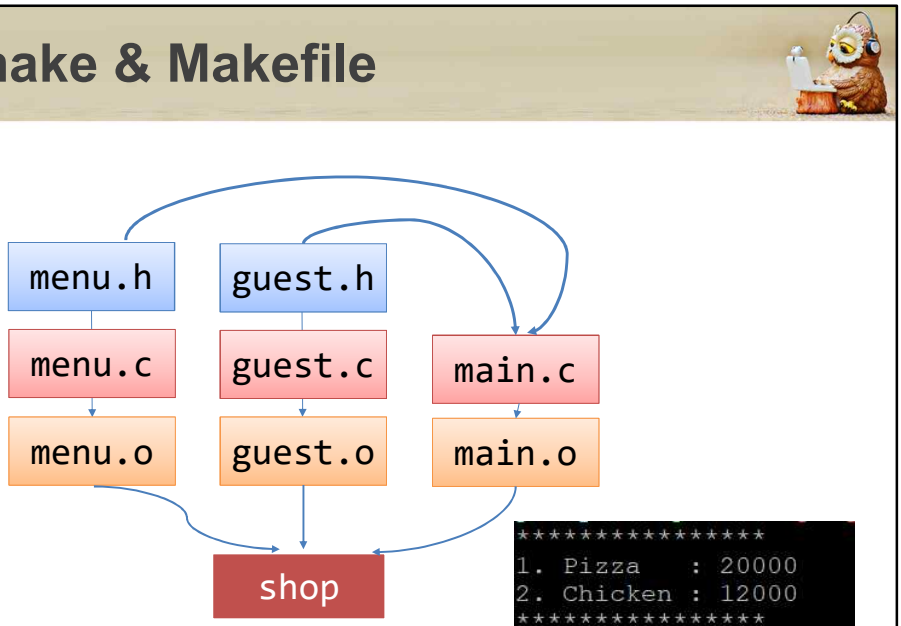
1

make vs Makefile

- **make** : 컴파일을 실행하기 위해 사용하는 파일 관리 유틸리티
- **Makefile** : make를 실행하면 Makefile 을 실행하게 됨
- **make를 사용할 때 장점**
  - ✓ 파일에 대한 반복적 명령 자동화로 인한 시간 절약
  - ✓ 프로그램의 종속 구조를 빠르게 파악하며 관리가 용이
  - ✓ 단순 반복 작업 및 재 작성을 최소화

2

make & Makefile



3

make & Makefile

- Create header files

menu.h

```
1 //menu.h
2
3 #include <stdio.h>
4
5 void displayMenu();
6
```

guest.h

```
// guest.h

#include <stdio.h>

int addGuest();
void displayGuest(int menu);
```

4

make & makefile

- Create C files

menu.c

```
// menu.c
#include "menu.h"

void displayMenu(){
        printf("****************\n");
        printf("1. Pizza    : 20000\n");
        printf("2. Chicken  : 12000\n");
        printf("****************\n");
}
```

guest.c

```
// guest.c
#include "guest.h"

int  addGuest(){
        int menu;
        printf("원하는 메뉴? ");
        scanf("%d", &menu);
        return menu;
}

void displayGuest(int menu){
        if(menu == 1)
                printf("Pizza 선택");
        else
                printf("Chicken 선택");
}
```

main.c

```
#include "menu.h"
#include "guest.h"

int main(){
        int  menu;

        displaymenu();
        menu = addGuest();
        displayGuest(menu);
        return 0;
}
```

5

make & makefile

- ls

```
jerry1004@peace:~/project/shop:> ls
guest.c  guest.h  main.c  menu.c  menu.h
```

- 각 파일 Compile

```
jerry1004@peace:~/project/shop:> gcc -c -o guest.o guest.c
jerry1004@peace:~/project/shop:> gcc -c -o menu.o menu.c
jerry1004@peace:~/project/shop:> gcc -c -o main.o main.c
```

- 하나의 실행파일 생성

```
jerry1004@peace:~/project/shop:> gcc -o shop main.o guest.o menu.o
jerry1004@peace:~/project/shop:> ls
guest.c  guest.h  guest.o  main.c  main.o  menu.c  menu.h  menu.o  shop
```

6

make & makefile

- **Makefile**
  - ✓ 목적파일(target):명령어가 수행되어 나온 결과 저장 파일(예외, dummy target)
  - ✓ 의존성(Dependency):목적파일을 만들기 위해 필요한 파일
  - ✓ 명령어 (Command):실행해야 할 명령어
  - ✓ 매크로 (Macro):코드를 단순화하기 위한 방법

```
CC = gcc

target1 : dependency1 dependency2
        command1
        command2
        …
```

7

make & makefile

- **Makefile 만들기 (Makefile)**
  - Makefile 생성 : **vi Makefile**

```
shop : menu.o guest.o main.o
        gcc -o shop menu.o guest.o main.o

menu.o : menu.c
        gcc -c -o menu.o menu.c

guest.o : guest.c
        gcc -c -o guest.o guest.c

main.o : main.c
        gcc -c -o main.o main.c

clean :
        rm *.o shop
```

```
make [enter]
ls  [enter]
make clean[enter]
```

8

make & makefile

- **Makefile2 만들기(with macro)**

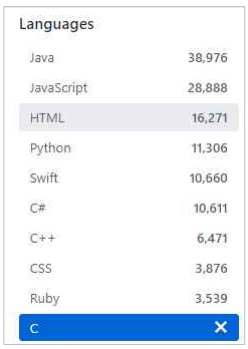
```
CC = gcc
CFlags = -W -Wall
TARGET = shop
OBJECTS = menu.o guest.o main.o

all : $(TARGET)

$(TARGET) : $(OBJECTS)
        $(CC) $(CFLAGS) -o $@ $^

clean :
        rm *.o shop
```

```
/shop:> make -f Makefile2
```

9

Lab1 : clone a project in Github

1. **Sign up/ Sign in "Github" : http://www.github.com**
2. **Search "calculator"**
3. **Select C language.**



4. **Click on selected Project.**
   https://github.com/btmills/calculator

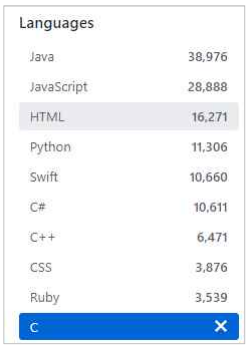
5. **Clone**

$git clone https://github.com/btmills/calculator

10

Lab1 : Makefile(1/2)

1. Sign up/ Sign in "Github" : http://www.github.com
2. Search "calculator"
3. Select C language.



4. Click on selected Project.
   https://github.com/btmills/calculator

5. Clone (위치 : cd ~/project/)

   $git clone https://github.com/btmills/calculator

11

Lab1 : Makefile(2/2)

1. tree calculator

```
jerry1004@peace:~/project:> tree cal*
calculator
├── calculator.c
├── LICENSE
├── Makefile
├── README.md
├── stack.c
├── stack.h
└── stack_test.c
```

2. 실행파일 만들기
3. Makefile 각 라인별 comment 작성

12