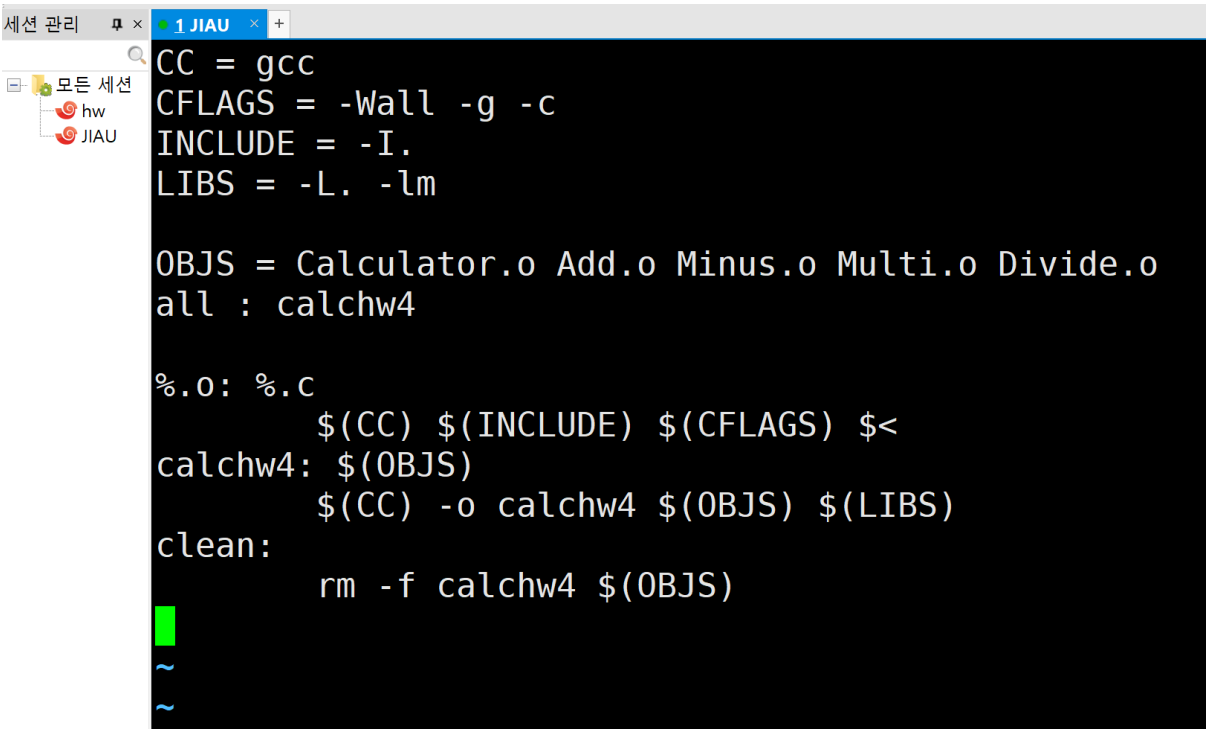
HW4. 분할 컴파일과 Makefile 실습

060분반 부산대학교 정보컴퓨터공학부
2020-55565
여지수
제출일: 2021-04-09

1. 수정한 C 코드 설명 (30점)

main함수가 들어있는 Calculator.c는 아래와 같이 수정한다.
<stdlib.h>와 "Caculator.h"를 include 해준다.
enum에서 Minus=2, Multi=3, Divide=4, Exit=5로 설정한다.
int main(void)는 int main(int argc, char *argv[])로 바꾸어 준다. 그 이유는 리눅스 환경에서 Calculator.c를 컴파일해서 실행파일을 만들고, 실행할 때 main 함수에 argc, argv라는 매개변수를 전달할 것이기 때문이다. argc는 매개변수의 개수로, 실행프로세스 명도 포함이 되어있고, argv는 매개변수가 저장되어 있는 포인터 배열이다. 그렇기에 scanf로 val1,val2,select값을 받는 것이 아니라 리눅스에서 실행할 때 매개변수를 전달받아 프로그램을 동작하도록 한다. 그래서 scanf는 없애준다. **참고로 while loop문은 필요없으니 제거한다.**
따라서 실행파일을 실행할 때 주어진 매개변수의 개수가 4개(실행프로세스명, select, val1, val2) 라면 계산을 수행하고, 개수가 4개보다 적거나 많으면 다른 메시지를 출력하도록 하는 것이다. 그래서 if-else 문을 이용해 if (argc<4)이면 더 많은 요소가 필요하다는 메시지를, else if (argc>4)이면 너무 많은 요소가 들어왔다는 메시지를, else 면 아래와 같은 동작을 수행하다록 한다.
매개변수는 문자열로 argv에 저장된다. else문 안에서 select는 정수로 받고 val1과 val2는 문자열로 받아야하기 때문에 atoi()와 atof()함수를 사용한다. 따라서 select=atoi(argv[1]) 로 atoi함수로 문자열을 정수로 변환하고, val1=atof(argv[2]); val2=atof(argv[3]) 로 atof함수로 문자열을 실수로 변환한다. 이후 switch문에서 select가 5일때 프로그램을 종료하는 메시지를 예제와 동일하게 변경해준다.
Calculator.h에서는 #ifndef 후 더하기, 빼기, 곱하기, 나누기 함수를 선언해주고, #endif를 해준다. 이러한 전처리기를 사용한 이유는 header의 중복을 막기 위해서이다. Add.c, Minus.c, Multi.c, Divide.c에서는 "Calculator.h"를 include하고 각각 더하기, 빼기, 곱하기, 나누기가 수행되도록 함수 코드를 적어준다. 함수들은 두개의 double을 매개변수로 두고 두 매개변수를 계산한 값을 return 하도록 구현했다.

2. Makefile 설명 (30점)



CC=gcc

➔ 매크로 CC를 정의한다. 컴파일러를 세팅해서 환경변수로 지정해주는 것이다. 버전을 바꿔서 컴파일할 때 유용하게 쓸 수 있다.

CFLAGS = -Wall -g -c

➔ 매크로 CFLAGS를 정의한다. gcc의 옵션을 추가해주는 용도이다. -g는 디버그 정보를 추가하라는 것이다.

INCLUDE = -I.

➔ 디렉토리를 추가한다.

LIBS = -L. -lm

➔ 링크할 때 필요한 라이브러리를 추가한다.

OBJS = Calculator.o Add.o Minus.o Multi.o Divide.o

➔ 오브젝트 파일들을 정의해준다.

<Target>: <Dependencies> (룰 rule)
<Recipe> (명령 command)

➔ Target 에는 빌드 대상 이름이 온다. 이 룰에서 최종적으로 생성해내는 파일명을 쓴다. Dependencies 는 빌드 대상이 의존하는 Target 이나 파일 목록이다. 여기에 나열된 대상들을 먼저 만들고 빌드 대상을 생성한다. Recipe 는 빌드 대상을 생성하는 명령이다. 여러 줄로 작성할 수 있으며, 각 줄 시작에 반드시 Tab 문자로 된 Indent 가 있어야 한다. 아래는 위와 같은 룰과 명령이 사용된다.

all : calchw4

➔ make 는 Makefile 을 순차적으로 읽어서 가장 처음에 나오는 규칙을 수행하게 된다. all 은 더미타겟(dummy target)이 바로 첫 번째 타겟으로써 작용하게 된다. 결과 파일이 많을 때 all 의 의존 관계(dependency)로써 정의해 두면 편리하다.

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

➔ 내부적으로 확장자 규칙이 적용되었다. $(...)은 매크로의 사용을 의미한다. 매크로를 통해 복잡한 것을 간단하게 나타낼 수 있다. $<는 입력파일로 콜론의 오른쪽에 오는 패턴을 치환하다는 의미를 담고있다. %는 일치하는 확장자를 제외한 파일명을 의미한다. 그래서 %.c를 컴파일하여 같은 이름의 오브젝트 파일을 만들겠다는 것이다. 즉 어떤 .o 파일이 필요할 경우 이 규칙에 의해서 해당 .c 파일을 찾고 아랫줄의 명령을 실행시켜서 .o파일을 생성해내는 것이다.

```
calchw4: $(OBJS)
	$(CC) -o calchw4 $(OBJS) $(LIBS)
clean:
	rm -f calchw4 $(OBJS)
```

➔ 이러한 확장자 규칙을 통해 확장자가 같은 여러 개의 파일에 똑같은 규칙을 적용할 수 있는 것이다.

이후 이 make를 실행하면 사용자가 내리는 명령을 알아서 해석해 컴파일을 자동화할 수 있는 것이다.

```
duwltn1301@ubuntu:~$ vi makefile
duwltn1301@ubuntu:~$ make clean
rm -f calchw4 Calculator.o Add.o Minus.o Multi.o Divide.o
duwltn1301@ubuntu:~$ make
gcc -I. -Wall -g -c Calculator.c
gcc -I. -Wall -g -c Add.c
gcc -I. -Wall -g -c Minus.c
gcc -I. -Wall -g -c Multi.c
gcc -I. -Wall -g -c Divide.c
gcc -o calchw4 Calculator.o Add.o Minus.o Multi.o Divide.o -L. -lm
duwltn1301@ubuntu:~$ ./calchw4
More arguments are needed.
------------The first argv[0] is filename:"./calc"
Please provide argv[1], Argv[2] and argv[3] for this program.
argv[1]: the type of operator
type: 1 for +, 2 for -, 3 for *, 4 for /, 5 for quit
argv[2] and argv[3]: input values

duwltn1301@ubuntu:~$ ./calchw4 1 1.0 4.0
Result value: 5.000000
duwltn1301@ubuntu:~$ ./calchw4 2 1.0 4.0
Result value: -3.000000
duwltn1301@ubuntu:~$ ./calchw4 3 1.0 4.0
Result value: 4.000000
duwltn1301@ubuntu:~$ ./calchw4 4 1.0 4.0
Result value: 0.250000
duwltn1301@ubuntu:~$ ./calchw4 5 1.0 4.0
Quit the program...
argv[2] and argv[3]: input values
duwltn1301@ubuntu:~$ ./calchw4 5 1.0 4.0 5
Too many arguments supplied.
------------The first argv[0] is filename:"./calc"
Please provide argv[1], Argv[2] and argv[3] for this program.
argv[1]: the type of operator
type: 1 for +, 2 for -, 3 for *, 4 for /, 5 for quit
duwltn1301@ubuntu:~$ ./calchw4 1
More arguments are needed.
------------The first argv[0] is filename:"./calc"
Please provide argv[1], Argv[2] and argv[3] for this program.
argv[1]: the type of operator
type: 1 for +, 2 for -, 3 for *, 4 for /, 5 for quit
argv[2] and argv[3]: input values

duwltn1301@ubuntu:~$
```

3. 프로그램 실행 방법 (30점)

비디오 제출 했습니다.

4. 논의 사항 (10점)

과제를 하면서 어려웠던 점은 공유라이브러리 부분이었다. 처음에 어떻게 포함해야할질 잘

몰라 오브젝트 파일들 각각에 대하여 공유라이브러리를 하나씩 생성했다. 알고보니 코드 옆에 오브젝트 파일들을 모두 나열하면 되는 것이었다. 그리고 1번 과제를 수행할 때 c파일을 어떻게 수정했는지 꼼꼼하게 설명 하려다보니 시간이 굉장히 오래걸렸다. 그 점이 조금 힘들었다.

새롭게 알게 된 점은 메인함수의 명령인수 argc, argv에 대해서이다. C언어를 처음배우고 윈도우 상에서 dev c++를 사용해 프로그램을 작성했을 때는 한번도 사용해본 적이 없는 명령인수였다. 이번에 유닉스 기초 과목을 하면서 처음 리눅스를 접했고, 리눅스에서 main함수에 정보를 전달할 때 공백을 기준으로 문자열이 argc에 차례대로 들어간다는 것을 알게 되었다.