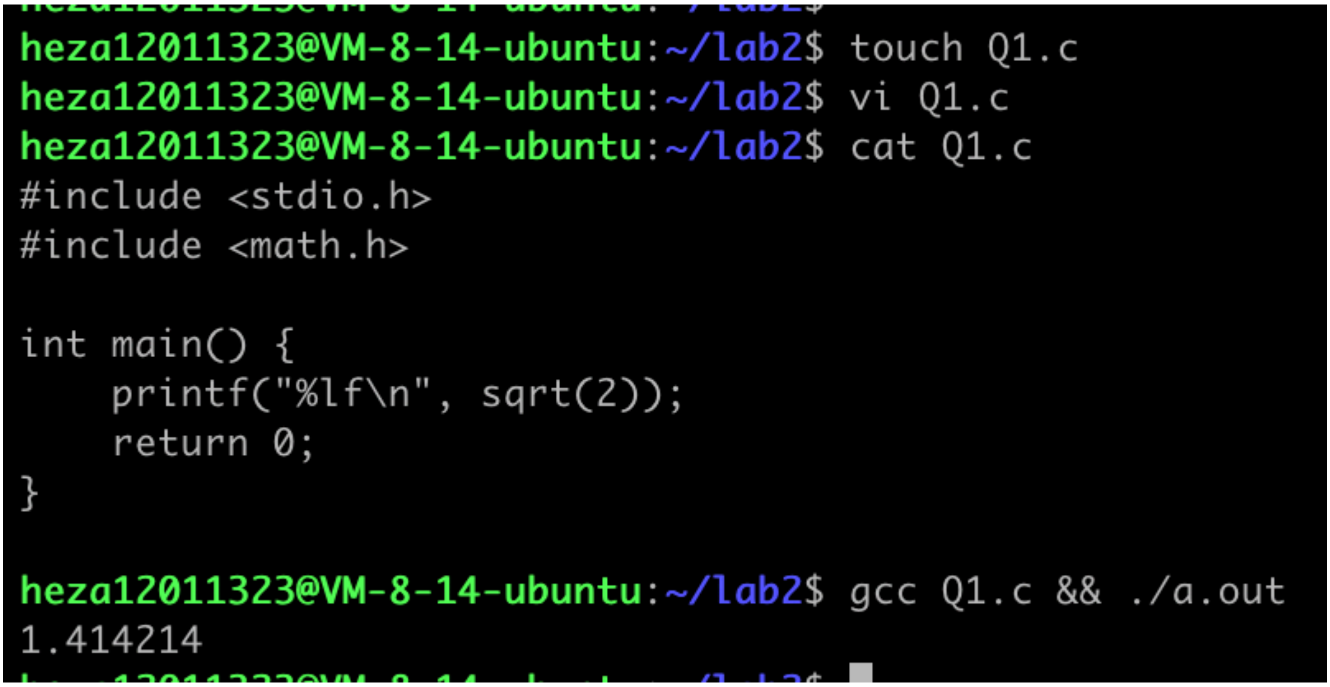
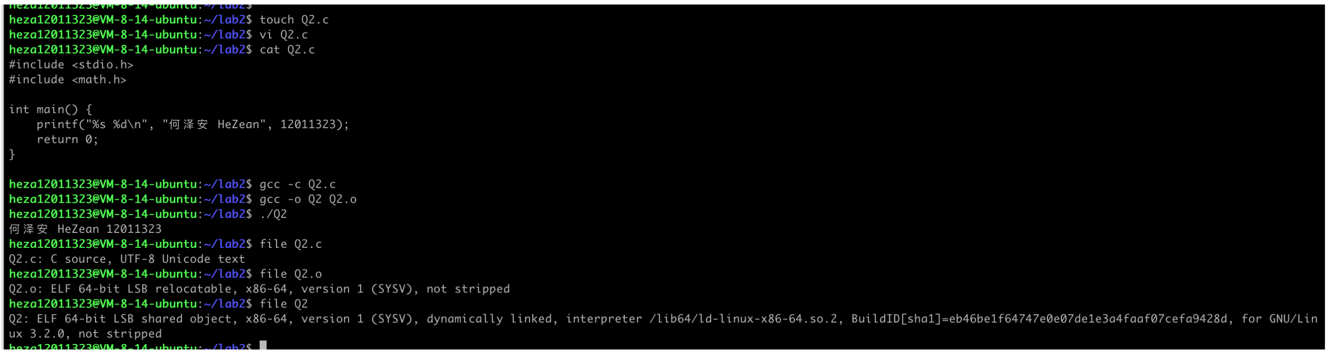
CS302 Lab2 Report

> 何泽安 12011323
> 2023.2.24

1. 

```
heza12011323@VM-8-14-ubuntu:~/lab2$ touch Q1.c
heza12011323@VM-8-14-ubuntu:~/lab2$ vi Q1.c
heza12011323@VM-8-14-ubuntu:~/lab2$ cat Q1.c
#include <stdio.h>
#include <math.h>

int main() {
    printf("%lf\n", sqrt(2));
    return 0;
}

heza12011323@VM-8-14-ubuntu:~/lab2$ gcc Q1.c && ./a.out
1.414214
```

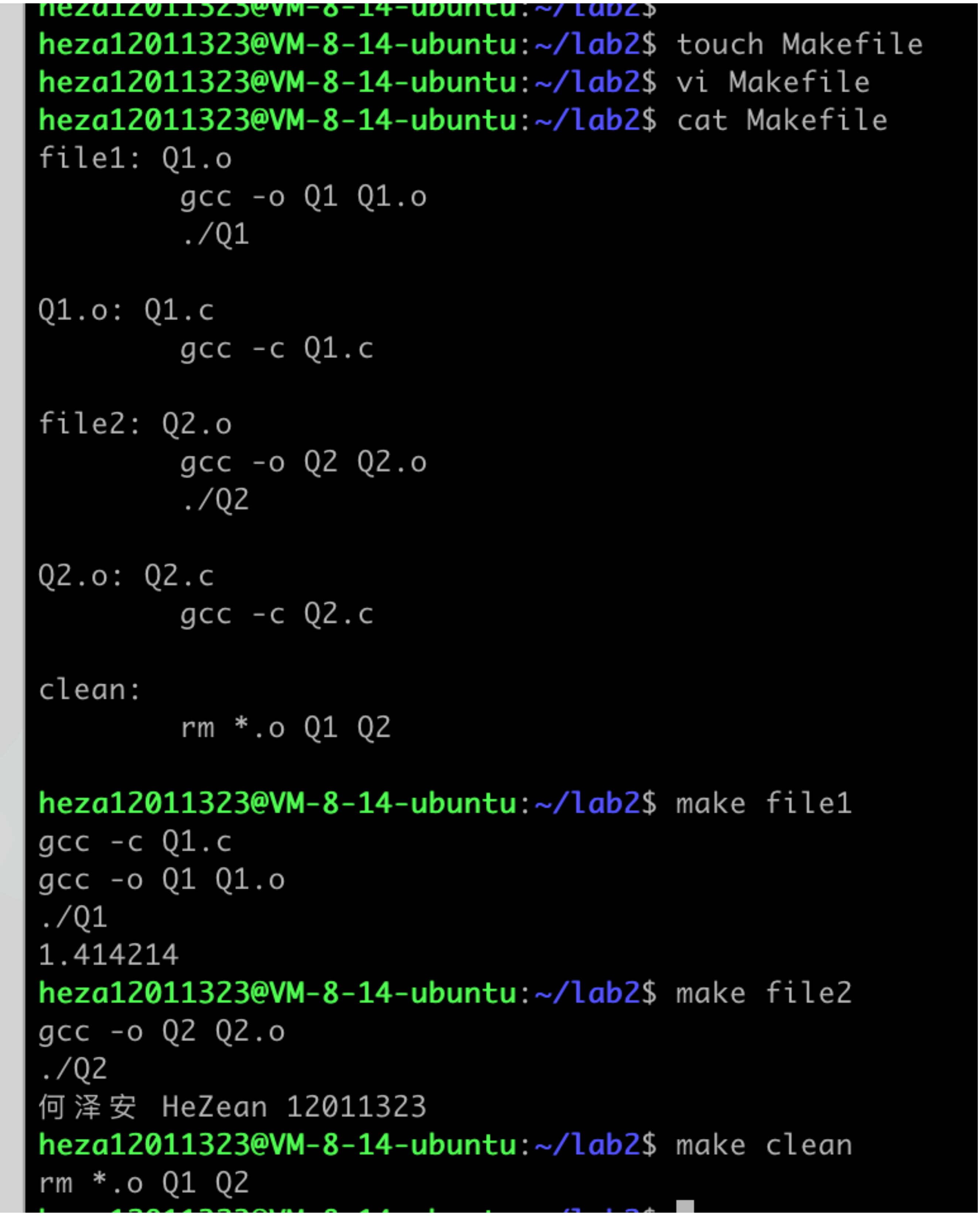
2. 

```
heza12011323@VM-8-14-ubuntu:~/lab2$ touch Q2.c
heza12011323@VM-8-14-ubuntu:~/lab2$ vi Q2.c
heza12011323@VM-8-14-ubuntu:~/lab2$ cat Q2.c
#include <stdio.h>
#include <math.h>

int main() {
    printf("%s %d\n", "何泽安 HeZean", 12011323);
    return 0;
}

heza12011323@VM-8-14-ubuntu:~/lab2$ gcc -c Q2.c
heza12011323@VM-8-14-ubuntu:~/lab2$ gcc -o Q2 Q2.o
heza12011323@VM-8-14-ubuntu:~/lab2$ ./Q2
何泽安 HeZean 12011323
heza12011323@VM-8-14-ubuntu:~/lab2$ file Q2.c
Q2.c: C source, UTF-8 Unicode text
heza12011323@VM-8-14-ubuntu:~/lab2$ file Q2.o
Q2.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped
heza12011323@VM-8-14-ubuntu:~/lab2$ file Q2
Q2: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=eb46be1f64747e0e07de1e3a4faaf07cefa9428d, for GNU/Lin
ux 3.2.0, not stripped
```

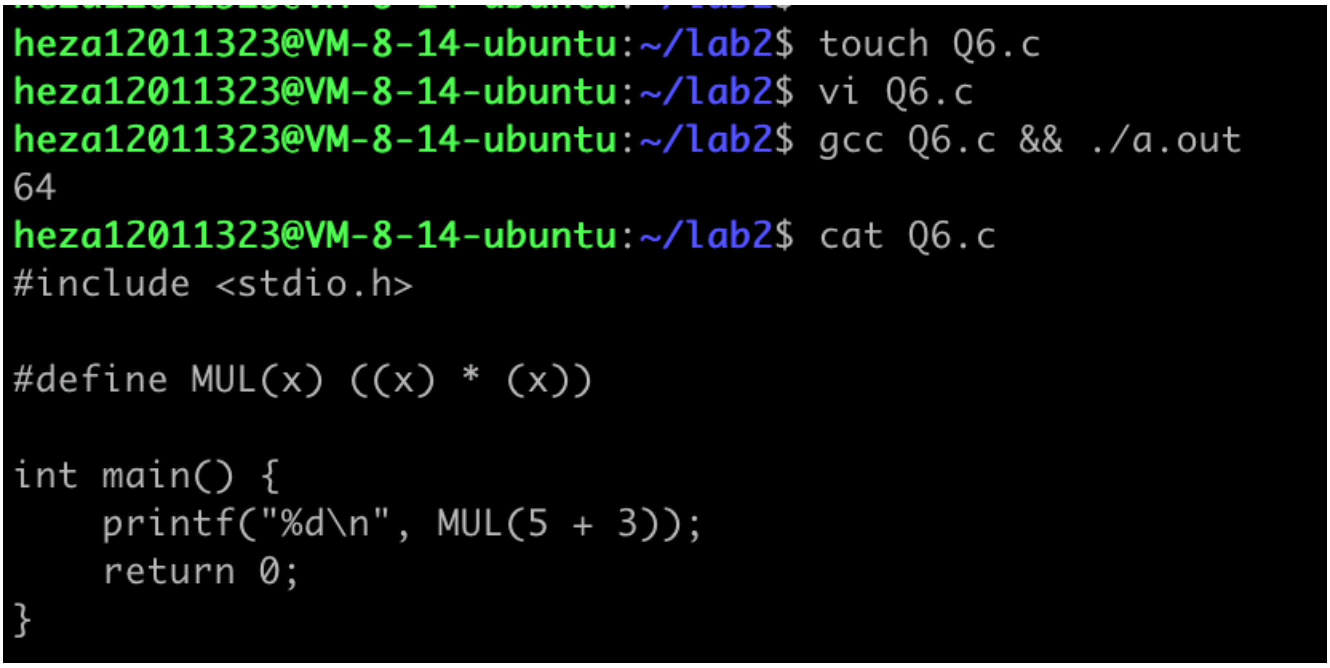
3. The process of compiling a `.c` file into an executable file contains three steps:
    1. **Preprocess**: the preprocessor find and process some macros, such as `#define`, `#include`, `#if` and 'expand' or some other process (like remove the code between the unsatisfied `#if` and `#else`) the C code.
    2. **Compile & Optimize**: the compiler first checks if there exists any syntex error, and do analyzes, if not, it process the C code into some inner code like assembly code. The optimizer performs optimizations on the code.
    3. **Assemble & Link**: The assembler convert the asm file into machine/object code, and finally we use the linker to link the used functions from the static/dynamic libraries, then generate the executable file.
4. From the screenshot above in Q2, the executable files' format is `ELF` (Executable Linkable Format). While the executable file in Windows is `PE` (Portable Executable).

> I don't have a Windows PC, the above statement about `PE` is referenced from CSDN

5.



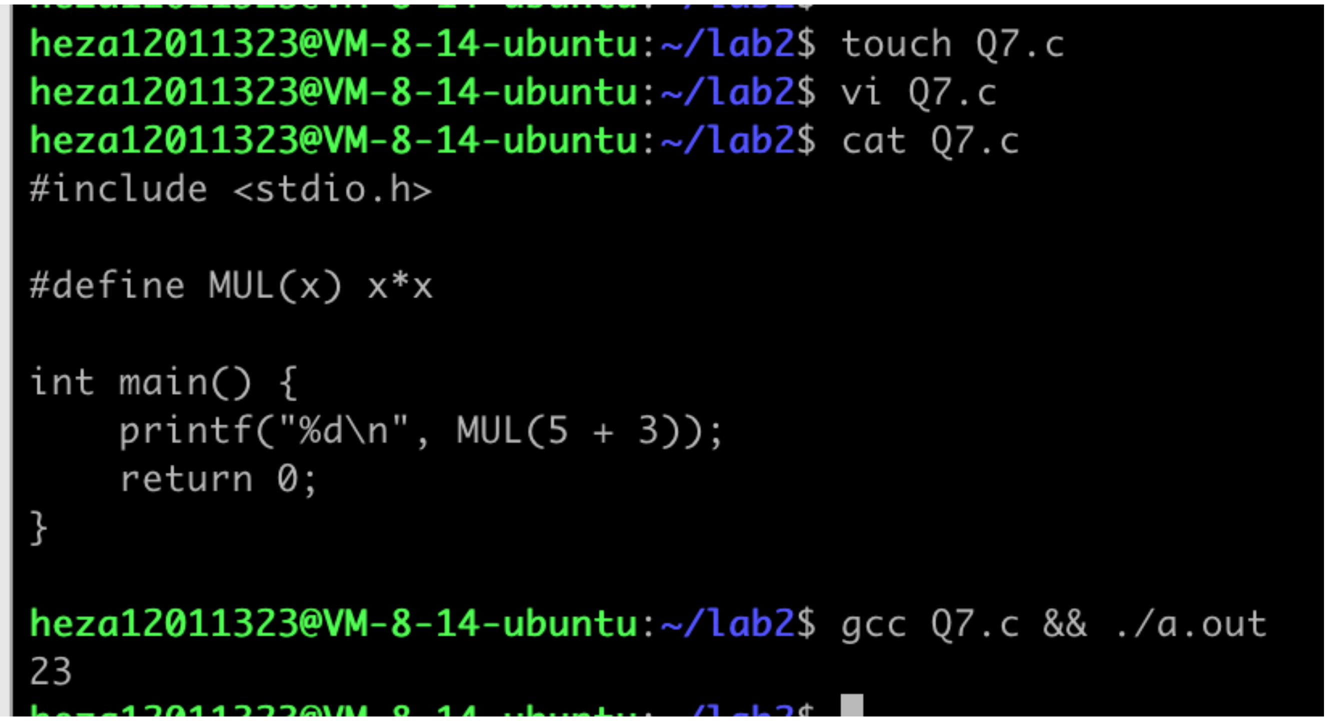
```
heza12011323@VM-8-14-ubuntu:~/lab2$
heza12011323@VM-8-14-ubuntu:~/lab2$ touch Makefile
heza12011323@VM-8-14-ubuntu:~/lab2$ vi Makefile
heza12011323@VM-8-14-ubuntu:~/lab2$ cat Makefile
file1: Q1.o
        gcc -o Q1 Q1.o
        ./Q1

Q1.o: Q1.c
        gcc -c Q1.c

file2: Q2.o
        gcc -o Q2 Q2.o
        ./Q2

Q2.o: Q2.c
        gcc -c Q2.c

clean:
        rm *.o Q1 Q2

heza12011323@VM-8-14-ubuntu:~/lab2$ make file1
gcc -c Q1.c
gcc -o Q1 Q1.o
./Q1
1.414214
heza12011323@VM-8-14-ubuntu:~/lab2$ make file2
gcc -o Q2 Q2.o
./Q2
何泽安 HeZean 12011323
heza12011323@VM-8-14-ubuntu:~/lab2$ make clean
rm *.o Q1 Q2
```

6. 

```
heza12011323@VM-8-14-ubuntu:~/lab2$ touch Q6.c
heza12011323@VM-8-14-ubuntu:~/lab2$ vi Q6.c
heza12011323@VM-8-14-ubuntu:~/lab2$ gcc Q6.c && ./a.out
64
heza12011323@VM-8-14-ubuntu:~/lab2$ cat Q6.c
#include <stdio.h>

#define MUL(x) ((x) * (x))

int main() {
    printf("%d\n", MUL(5 + 3));
    return 0;
}
```

The result is correctly 64. Since we all know that macro is a simple textual replacement during preprocessing. The replaced code is as below, which gives `8*8=64`

```
((5 + 3) * (5 + 3))
```

7. 

```
heza12011323@VM-8-14-ubuntu:~/lab2$ touch Q7.c
heza12011323@VM-8-14-ubuntu:~/lab2$ vi Q7.c
heza12011323@VM-8-14-ubuntu:~/lab2$ cat Q7.c
#include <stdio.h>

#define MUL(x) x*x

int main() {
    printf("%d\n", MUL(5 + 3));
    return 0;
}

heza12011323@VM-8-14-ubuntu:~/lab2$ gcc Q7.c && ./a.out
23
```

The result is 23. Since we all know that macro is a simple textual replacement during preprocessing. The replaced code is as below, which gives `5+15+3=23`

```
5 + 3*5 + 3
```