The following is a simple C language program example, showing the input and output of the compiler, as well as the compilation process.

#### 1. \*\*Introduce\*\*

- Source Code: Code written in C language, stored in a file with the '.c' extension.

- Sample file: 'hello.c'

```c

#include <stdio.h>

int main() {

printf("Hello, World!\n");

return 0;

}

```

- \*\*Compiler Options\*\*: When using a compiler like 'gcc', options such as output file name ('-o') may be specified.

- Example command:

```bash

gcc hello.c -o hello

```

#### 2. \*\*Compilation Process\*\*

In the case of 'gcc' (GNU C Compiler), the compiler performs the following steps:

1. \*\*Preprocessing\*\*: Expand '#include <stdio.h>' to include the definitions of standard input and output libraries.

2. \*\*Compilation\*\*: Converting C code into assembly language.

3. \*\*Assembly\*\*: Convert the assembly language into machine code and generate the target file ('.o').

4. \*\*Linking\*\*: Link the target file with the C standard library to generate the final executable file.

#### 3. \*\*Export\*\*

- \*\*Executable File\*\*:

- On Linux, the generated file is 'hello' (or specify a name).

- On Windows, it might be 'hello.exe'.

- \*\*Execution Result\*\*: After running the executable file, the output is as follows:

```bash

./hello

```

Screen display:

```

Hello, World!

```

- \*\*Error Message (if there is an error)\*\*: If there is an error in the code, such as missing semicolons, the compiler outputs an error message:

```bash

hello.c:3: error: expected ‘; ’ before ‘return’

```

#### 4. \*\*Example Operation\*\*

Let's say you have the 'hello.c' file, run the following command:

```bash

gcc hello.c -o hello

./hello

```

- \*\*Input\*\*: 'hello.c' (C source code).

- \*\*Output\*\*: 'hello' (executable file) that displays 'Hello, World!' when executed.

#### 5. \*\*Compiler Used\*\*

- \*\*gcc\*\*: GNU C compiler for Linux, Windows (via MinGW), and more.

- Other common C compilers include 'clang' and 'MSVC'.