COMPILING A C++ PROGRAM

Tsykal I.O., *igorcikal2260622@gmail.com*
Institute of Special Communications and Information Protection
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

The procedure for compiling a C++ program is the same as for a C program, but uses the command g++ instead of gcc. Both compilers are part of the GNU Compiler Collection. To demonstrate the use of g++, here is a version of the Hello World program written in C++:

```
#include
      int main () {
std::cout << "Hello, world!" << std::endl;
return 0;    }
```

The program can be compiled with the following command line:

```
g++ -Wall hello.cc -o hello
```

The resulting executable can be run in exactly same way as the C version, simply by typing its filename:

```
./hello
Hello, world!
```

The C++ frontend of GCC uses many of the same the same options as the C compiler gcc. Note that C++ source code should be given one of the valid C++ file extensions '.cc', '.cpp', '.cxx' or '.C' rather than the '.c' extension used for C programs.[1]

Compiling a C++ program involves taking the source code we have written (.cpp, .c, .h, .hpp files) and converting them into an executable or library that can run on a specified platform.[2]

This process can be divided into three key stages:

1.Pre-processing
2.Compilation
3.Linking

C++ has pre-processor directives that are identified in the code by the prefix #, which defines behaviors that are to be carried out on the source code before it's compiled. The exact nature of what the pre-processor does depends on the pre-processor directive. For example, we often split code into separate files to make it easier to organize and read. To link code in one file with that in another, we use the #include directive.

Once the preprocessor finishes creating that (sometimes huge) translation unit, the compiler starts the compilation phase and produces the object file. We can see that the compiler must compile a much larger file than the simple source file that we see. This is because of the included headers.

An object file has the .obj or .o file extension and is created for each source code file. The object file contains all of the machine level instructions for that file. It is referred to as an intermediary file because it's not until the final stage, linking, that an actual executable or library that we can use is created.[3]

Linking as the name suggests, refers to creation of a single executable file from multiple object files. The file created after linking is ready to be loaded into memory and executed by the system . There is difference in linking and compilation when it comes to understanding errors. After the compiler creates one or more object files, then another program called the linker kicks in. The job of the linker is three fold.First, to take all the object files generated by the compiler and combine them into a single executable program.

Each object file can be viewed as binary storage of individual source file contributions to the program memory map sections of all kinds (code, initialized data, uninitialized data, debugging information, etc.).[4]

References

1. Brian Gough An Introduction to GCC 2005. 55 p.

2. Learn How To Compile a C++ Program: website. URL: https://betterprogramming.pub/learn-how-to-compile-a-c-program-382c4c690bdc (Last accessed 31.03.2021)

3. C++ Compiler Operation: website. URL: https://icarus.cs.weber.edu/~dab/cs1410/textbook/1.Basics/compiler_op.html (Last accessed 31.03.2021)

4 Milan Stevanovic C and C++ compiling 2014. 29 p.

Scientific adviser: *Sokyrska O.S., PhD in Philology, senior lecturer, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"*