Hello World and Why It's Important

Charles Martin

November 14, 2017

Contents

1 Introduction

One of the problems that faces beginning programmer – and every programmer ever when they start a new project – is what to do with that new blank project directory. The traditional way to deal with that is to build a simple program that demonstrates you can indeed write some simple program that runs.

Since (at least) the original version of Kernighan and Ritchie's [[The C Programming Language][]] that first simple program is called "Hello, world!". In C, the hello, world program is:

```
#include <stdio.h>

int main(int argc, char** argv){
    printf("Hello, world!\n");
    return 0;
}
```

Compiling and running it, simplest case is like this:

```
$ cc hello.c
$ ./a.out
Hello, world!
$
```

In Java, it looks like:

```
/**
 * Canonical hello world in Java. Note this *must* be in a file named
 * 'Hello.java'
 */
public class Hello {
    public static void main(String[] argv){
        System.out.println("Hello, world!");
    }
}
```

```
$ javac Hello.java
$ java Hello
Hello, world!
$
```

In C++ it's:

```
#include <iostream>

using namespace std;

int main(int argc, char ** argv){
    cout << "Hello, world!" << endl;
    return 0;
}
```

And sure enough:

```
$ g++ hello.cpp
$ ./a.out
Hello, world!
$
```

That's enough examples for now. If you want to see "Hello, world!" in every computer language you can name, there's an online repository of "hello, world" programs at The Hello World Collection.

This is obviously not a very interesting program in itself, but writing it and running it means that you must:

- find an editor or IDE that you can use for the particular language;
- successfully write the code and save it;
- compile and run, or invoke the interpreter to run the program.

These are, not coincidentally, all the problems you must solve to start writing a program in any language or environment. This is even true in an old-fashioned punch-card environment – you needed to punch cards more or less like this:

```
//FORT     EXEC PGM=IEYFORT,REGION=100K                                    00020000
//SYSPRINT DD  SYSOUT=A                                                    00040000
//SYSPUNCH DD  SYSOUT=B                                                    00060000
//SYSLIN   DD  DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,                X00080000
//             SPACE=(80,(200,100),RLSE),DCB=BLKSIZE=80                    00100014
//LKED EXEC PGM=IEWL,REGION=96K,PARM=(XREF,LET,LIST),COND=(4,LT,FORT)      00120000
//SYSLIB   DD  DSNAME=SYS1.FORTLIB,DISP=SHR                                00140000
//SYSLMOD  DD  DSNAME=&GOSET(MAIN),DISP=(NEW,PASS),UNIT=SYSDA,            X00160000
//             SPACE=(1024,(20,10,1),RLSE),DCB=BLKSIZE=1024                00180014
//SYSPRINT DD  SYSOUT=A                                                    00200000
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),DCB=BLKSIZE=1024,    X00210018
//             DSNAME=&SYSUT1                                              00220018
//SYSLIN   DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)                           00240000
//         DD  DDNAME=SYSIN                                                00260000
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((4,LT,FORT),(4,LT,LKED))                00280000
//FT05F001 DD  DDNAME=SYSIN                                                00300000
//FT06F001 DD  SYSOUT=A                                                    00320000
//FT07F001 DD  SYSOUT=B
C     Hello World in Fortran

      PROGRAM HELLO
      WRITE (*,100)
      STOP
  100 FORMAT (' Hello World! ' /)
```

```
      END
/*
```

(Warning, don't take that JCL – the stuff at the top – very seriously. It's an example JCL that will compile and run a FORTRAN program, but it's utterly untested.)

Now, this looks pretty simple – super-simple, frankly. There are a lot of potential complexities you could run into – and we'll talk about them more in the next section – but even this simple a "Hello, world!" can expose surprises.

For example, my original version of the C++ program was:

```
#include <iostream.h>

int main(int argc, char ** argv){
    cout << "Hello, world!" << endl;
    return 0;
}
```

The results of compiling it were:

```
541 $ g++ hello.cpp
hello.cpp:1:10: fatal error: 'iostream.h' file not found
#include <iostream.h>
         ^~~~~~~~~~~~
1 error generated.
```

"What?" I wondered. (Actually I used three words, but the meaning was the same.) Now, I haven't written much C++ in a long time, but it looked reasonable. But C++ has gone through a lot of modifications since I was using it in the late 90's, and I'd missed the change from '<iostream.h>' to '<iostream>'.

In this case, it wouldn't have been a really major issue even if I'd written lots of code before trying an initial compile (but don't do that! More on this in another note.) On the other hand, making this program compile and run both exhibited the problem and reminded me to grab my copy of Stroustrup's The C++ Programming Language and refresh myself before I try anything too radical.

2 Beyond Hello, World

So far, we've looked at some trivial "Hello, world" sorts of programs. While they *are* trivial, they're not useless – every one of them means you've solved whatever problems there are before you can start writing a program.

But sometimes it's not that simple. In the Java world, the "hello, world" above demonstrates that you've successfully installed both the Java runtime and a Java Development Kit (or installed an IDE for Java, like Net Beans.)

If you intend to write a significant program in Java, you have to choose a build tool and set up the environment it requires, or you have to become enough a wizard in that build tool to make up your own set of conventions (pro tip: don't.)

Other JVM languages have similar problems because of some design decisions made early in the life of Java that turn out to be perhaps less wise than they seemed. (Which ones, you ask? Some of them: tying the names of classes to their file names, tying packaging to the directory structure, which makes the directory layout be reflected in the source code: move a file or rename a package, and you break all the code associated with that. But that's another article.)

2.1 Making a Java Hello, World project using Maven

The most common environment for Java programming these days is using Maven. The Maven tool, `mvn`, needs to be downloaded and installed, which can be a project in itself. Then you need to build a Maven environment, using an incantation like:

```
$ mvn -B archetype:generate \
    -DarchetypeGroupId=org.apache.maven.archetypes \
    -DgroupId=com.mycompany.app \
    -DartifactId=my-app
```

```
$ mvn -B archetype:generate \
      -DarchetypeGroupId=org.apache.maven.archetypes \
      -DgroupId=com.salveteomnis.hello \
      -DartifactId=hello
```

After which you get:

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.4:generate (default-cli) > generate-sources @ stand
[INFO]
[INFO] <<< maven-archetype-plugin:2.4:generate (default-cli) < generate-sources @ stand
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:2.4:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archety
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: /Users/chasrmartin/Dropbox (Personal)/Tutoring/HowToM
[INFO] Parameter: package, Value: com.salveteomnis.hello
[INFO] Parameter: groupId, Value: com.salveteomnis.hello
[INFO] Parameter: artifactId, Value: hello
[INFO] Parameter: packageName, Value: com.salveteomnis.hello
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: /Users/chasrmartin/Dropbox (Per
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5.839 s
[INFO] Finished at: 2017-11-14T10:52:00-07:00
[INFO] Final Memory: 17M/320M
[INFO] ------------------------------------------------------------------------
$
```

This generates a "Hello, world" program for you, which is nice:

```
package com.salveteomnis.hello;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

It also generates mvn's own project description file, called the POM file ("project object model", named in the days when to be a good product it had to have the word "object" somewhere in it.) That is an XML-formatted file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.salveteomnis.hello</groupId>
  <artifactId>hello</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>hello</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Which builds the whole directory tree Maven expects, all 20 directories and six files of it.

```
$ tree
.
 pom.xml
 src
    main
       java
           com
               salveteomnis
                   hello
                       App.java
    test
        java
            com
                salveteomnis
                    hello
                        AppTest.java
 target
     classes
        com
            salveteomnis
                hello
                    App.class
     maven-status
         maven-compiler-plugin
             compile
                 default-compile
                     createdFiles.lst
                     inputFiles.lst

20 directories, 6 files
```

After that, you can finally compile your program:

```
$ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building hello 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hello ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. bu
[INFO] skip non existing resourceDirectory /Users/chasrmartin/Dropbox (Personal)/Tutor
```

```
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ hello ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is
[INFO] Compiling 1 source file to /Users/chasrmartin/Dropbox (Personal)/Tutoring/HowToI
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.566 s
[INFO] Finished at: 2017-11-14T11:04:01-07:00
[INFO] Final Memory: 14M/255M
[INFO] ------------------------------------------------------------------------
$
```

If this sounds like I'm not a major fan of Maven, well, good. But at the same time, Maven solves a lot of problems in Java programming, like retrieving 3rd-party packages and making them available. The real point, though, is that you are going to have to solve all your Maven issues to get started anyway, and you don't know you have until you can run a "Hello, world" Maven project.

3 Conclusion

I do a lot of training and mentoring real beginners in programming. Conssitently, one of the hardest problems for them is what to do when you're looking at that first, empty project. But anyone can set up a "Hello, world" program, and make it run, and that first step means solving your first problems with any new project.