Hello World: Special Edition

Max Meldrum
max@meldrum.se

Abstract

Since being introduced in the book "The C Programming Language", Hello World has become the typical way of how people are introduced to the world of programming. We observe that the classical approach is conservative, all characters are displayed solely through one programming language. In this paper, we propose an improved version which enables the execution to transpire through multiple languages.

1 Introduction

Instructing the computer to display text on the screen is the most common first program that any developer will encounter. The concept of Hello World became popular through[1]. Many years has past since its introduction, and still there has not been any further research into the language exposure aspect of Hello World.

The remainder of the paper introduces the special edition algorithm (Section 2), dissects the performance and language exposure (Section 3), discusses next generation Hello World programs (Section 4) and draws the conclusion (Section 5).

2 Algorithm

In our reference implementation[1], we elected to go for "Hello World!" rather than "Hello, World!". In total, including the white space, we have 12 characters. It is possible to connect any amount of languages. However, it is not guaranteed that the 12 selected languages will be unique. The algorithm will pick randomly from a basket.

Special Edition can be understood by describing its two phases. In phase 1 (initialization phase), there is a confirmation step that compilers of the chosen languages are available. If the examination goes through without any issues, all languages are inserted in to a List, which is then used to generate another list with 12 random languages. One important thing to note is that as long as you have one language available, the algorithm will be able to run. If Python, C and Java are selected, the algorithm will simply pick randomly from the three until it has a total of 12.

A prerequisite for Special Edition is the capability of executing system calls. Executor's (see 2.1) utilises system calls to pass on the next task to the Controller (see 2.2).

Algorithm 1 Initialization Phase

```
1: procedure GENERATELANGUAGES
2:     N ← 12
3:     L ← Languages
4:     for N iterations do
5:         S ← Random(L)
6:     return S
```

2.1 Executor

The Executor system has two tasks. PartialPrint and Notify. The former checks if the given character position is valid, if it is, it then prints the character that corresponds to the position. The latter reviews the current position and decides whether to pass on the task via the Controller or exit the system.

For the final output to be outputted correctly, the buffered stdout has to be flushed after every print operation.

[1] https://github.com/Max-Meldrum/hello-world-special-edition/

Algorithm 2 Executor PP

```
procedure PARTIALPRINT(POS)
    if pos == 0 then
        print "H"
    else if pos == 1 then
        print "e"
    else if pos == 2 then
        print "l"
    else if pos == 3 then
        print "l"
    else if pos == 4 then
        print "o"
    else if pos == 5 then
        print "␣"
    else if pos == 6 then
        print "W"
    else if pos == 7 then
        print "o"
    else if pos == 8 then
        print "r"
    else if pos == 9 then
        print "l"
    else if pos == 10 then
        print "d"
    else if pos == 11 then
        print "!"
    else if pos == 12 then
        print "\n"
```

Algorithm 3 Executor Notify

```
procedure NOTIFY(POS, NLANGS)
    if pos >= 0 and pos < 12 then
        target ← nlangs.head
        executors ← nlangs.dropHead
        Controller(target, pos+1, executors)
    else
        shutdown()
```

2.2 Controller

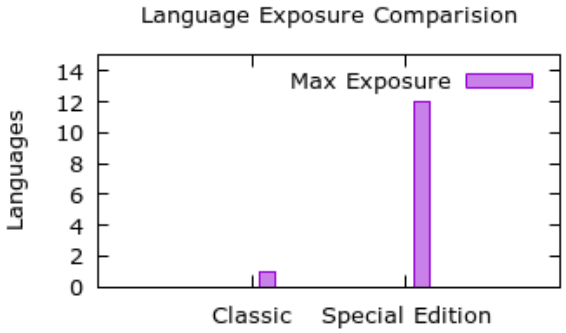
The controller acts as a bridge between Executor's. After an Executor has printed a character, it notifies the controller of the next target language, position and the list that contains the Executor order. With this information, the bridge acts like a proxy and runs the new Executor. The controller can be implemented in any language with basic functionalities. A shell script is used in the reference implementation.

Algorithm 4 Controller Algorithm

```
procedure BRIDGE(TARGET, POS, NLANGS)
2:      Execute(Target).with(pos, nlangs)
```

3 Evaluation

The traditional Hello World program outpeforms the Special Edition version in runtime performance as there will be delays between each Executor instance. However, what we are interested in is the maximum amount of languages that can be utilised.



4 Future Work

Special Edition runs locally on the computer. One could improve the system to be capable of executing Hello World in a peer-to-peer fashion. A peer would be able to act as an Executor and a Controller. This would increase the language exposure immensely as the more computers that are connected to the network, the bigger chance we have of being exposed to more languages.

5 Conclusion

The traditional approach to Hello World is only executed with one language in mind. In this paper, we describe an improved version named Special Edition which supports the execution to occur through multiple languages. We draw the conclusion that our version increases the level of language exposure at cost of performance. We believe that this is just the first step into the next generation of Hello World programs.

References

[1] Brian W. Kernighan. *The C Programming Language*. Prentice Hall Professional Technical Reference, 2nd edition, 1988.