DIRE

DIFERENÇA ENTRE LIGUAGEM COMPILADA E LIGUAGEM INTERPRETADA

Com a popularização de linguagens como [Java](http://www.java.com/) e [C#](http://msdn.microsoft.com/en-us/vcsharp), e sua forte adoção no mercado de TI, é comum nos depararmos com debates sobre as diferenças entre **linguagens interpretadas** e **linguagens compiladas**. Mas na hora de classificar uma linguagem como interpretada ou compilada, a coisa esquenta e ninguém entra em acordo! Mas afinal, o que é uma **linguagem interpretada** e o que é uma **linguagem compilada**?

Antes de mais nada, vamos definir nosso glossário:

O dicionário da língua portuguesa define **Compilar**:

do Latim *compilare*  
v. tr.  
reunir; ajuntar

Enquanto a definição de **Interpretar**, é:

do Latim *interpretare*  
v. tr.  
tornar claro o sentido de; explicar; traduzir; fazer juízo a respeito de.

Pelas definições desses dois verbos, já podemos perceber que seus significados não se opõe, mas se complementam. Então como classificar uma linguagem de programação como sendo de um jeito **ou** de outro? Bem, a resposta é simples, definindo o contexto ou ponto de vista! E como estamos analisando linguagens de programação, nosso contexto é **arquitetura de linguagens de programação**.

Na computação, a compilação é o processo que reúne o código fonte e o transforma em algo que faça mais sentido para o computador. Do ponto de vista do código fonte, toda linguagem de programação é compilada.

O produto final do processo de compilação de uma linguagem diz muito sobre seu design. Linguagens como [C](http://en.wikipedia.org/wiki/C_(programming_language)) e [C++](http://www.research.att.com/~bs/C++.html) são *compiladas estaticamente*, e seus códigos fontes são transformados diretamente em *linguagem de máquina*. Enquanto as linguagens mais modernas como [Java](http://www.java.com/), [C#](http://msdn.microsoft.com/en-us/vcsharp) e [Python](http://www.python.org/) têm seus códigos fontes transformados em uma *linguagem intermediária* (específica de cada linguagem), que será **interpretada** pela *máquina virtual da linguagem* quando o programa for executado.

Este processo de **interpretação** da *linguagem intermediária* durante a execução do programa, consiste na tradução dos comandos da *linguagem intermediária* para *linguagem de máquina*. Sendo assim, em tempo de execução, o código intermediário pode ser encarado como um “código fonte” que será *compilado dinamicamente* pelo *interpretador da linguagem* em código de máquina.

Obviamente, ter este processo de compilação embutido na execução do programa tem um custo. E esse custo não é barato! Por isso, nos últimos anos muito foi investido para otimizar este processo, resultando em todas as técnicas de [Just In Time Compiling](http://en.wikipedia.org/wiki/Just-in-time_compilation) e [Ahead of Time Compiling](http://en.wikipedia.org/wiki/AOT_compiler" \o "Veja a definição de AOT no wikipedia." \t "_blank) que permitem as **linguagens interpretadas** alcançarem performance excepcionais.

Finalmente, com base nestas definições, podemos dizer que C e C++ são **linguagens compiladas**. Enquanto Java, C# e Python, mesmo com as técnicas de JIT e AOT, são **linguagens interpretadas**, afinal, esta é uma definição da *arquitetura da linguagem de programação*.