Interpretação

Baseado no Capítulo 8 de Programming Language Processors in Java, de Watt & Brown

Atualizado em 29/03/2018

- Execução imediata do programa-fonte
- Não há tradução para uma representação de baixo-nível

VANTAGENS:

- Não há consumo de recursos para a tradução;
- Resposta imediata para o usuário;
- Ideal para ambientes interativos;
- Programas geralmente são descartados após o uso.

- DESVANTAGENS:

- Interpretador deve ser executado junto com o código;
- Mais tempo e mais memória são necessários;
- Análise sintática e de contexto são feitas durante a excução do programa;
- Execução mais lenta.

TIPOS DE INTERPRETAÇÃO:

Interpretação iterativa

- Quando todas as instruções do programa-fonte são primitivas;
- A interpretação de uma instrução não depende da interpretação de uma outra instrução interna.
- Ciclo de busca, análise e execução;
- Adequada para máquinas reais e abstratas (emuladores), linguagens de programação muito simples (Basic) e linguagens de comando (shell scripts);
- Mais rápida;
- Mais simples.

TIPOS DE INTERPRETAÇÃO:

Interpretação iterativa

```
do      {
        fetch the next instruction
        analyze this instruction
        execute this instruction
        }
While (still running);
```

- A instrução é recuperada da memória ou digitada diretamente pelo usuário;
- São identificadas as partes componentes da instrução;
- A instrução é executada;
- O ciclo é repetido.

TIPOS DE INTERPRETAÇÃO:

Interpretação iterativa

- Para linguagens de baixo-nível e linguagens de comando a análise sintática e a análise de contexto são muito simples (poucos operadores e operandos, e tabela de símbolos para resolução de rótulos);
- Para linguagens de programação simples, as análises sintática e de contexto não são triviais e se assemelham com as que são feitas em compiladores;
- Programas podem ser representados de três formas:
    - Arquivo-texto: as análises léxica e sintática são feitas em tempo de execução – aumenta o tempo de resposta;
    - Seqüência de tokens: a análise léxica é feita no momento do carregamento e a análise sintática é feita no momento da execução – solução de compromisso;
    - AST: análise léxica e sintática são feitas no momento do carregamento – execução rápida porém carregamento lento.

TIPOS DE INTERPRETAÇÃO:

Interpretação recursiva

- A interpretação de uma instrução pode depender da interpretação de uma outra instrução interna;
- Adequada para linguagens de programação de alto-nível com instruções compostas (comandos, expressões, declarações etc) e query languages;
- Análises e execução precisam ser feitas de forma recursiva;
- Mais lenta;
- Mais complexa.

TIPOS DE INTERPRETAÇÃO:

Interpretação recursiva

```
fetch and analyze the program
execute the program
```

- A interpretação ocorre em duas etapas;
- A análise abrange sintaxe e contexto;
- As duas são feitas de forma recursiva;
- AST costuma ser usada como representação intermediária entre as duas etapas;
- Não é muito comum, por causa do tempo de execução;
- É mais comum compilar para alguma linguagem de baixo-nível, e depois interpretar a mesma (maior velocidade).

- Apesar do ônus de uma execução mais lenta, a interpretação ainda é bastante usada;
- É mais fácil construir um interpretador do que um compilador;
- Os processos internos de análise sintática e de contexto, no entanto, são idênticos;
- Muitas vezes a interpretação é usada como primeira implementação, ou implementação experimental, de uma nova linguagem;
- BASIC foi uma linguagem interpretada muito popular, e que contribuiu de forma decisiva para a disseminação da computação entre leigos;
- Outras linguagens bastante diferentes e que se mostraram muito adequadas à interpretação (recursiva) foram LISP e PROLOG;
- JAVA utiliza um esquema combinado, com compilação e interpretação.