Ano V – Número 08 – Fevereiro de 2008 – Periódicos Semestral

UMA BREVE INTRODUÇÃO AO ESTUDO E IMPLEMENTAÇÃO DE COMPILADORES

[1]BRANCO; Guido Aparecido Junior, [2]TAMAE, Rodrigo Yoshio
1-Discente do Curso Sistemas de Informação – FAEG/Garça
2-Docente do Curso Sistemas de Informação – FAEG/Garça
Guido_email@email.com.br; rytamae@yahoo.com.br

RESUMO

Este artigo tem o objetivo de demonstrar as etapas e os fundamentos necessários para a concepção de um compilador (parsers e gerador de código), bem como abordar, de forma introdutória, duas ferramentas baseadas em Java utilizadas na geração de projeto de compiladores: JavaCC e Jasmin.
Palavras-chave: Parser, JavaCC e Jasmin.

ABSTRACT

This paper has the objective to demonstrate the stages and the necessary beddings for the conception of a compiler (parsers and code generator), as well as approaching, in introductory form, two tools based on Java used in generation compilers project's: JavaCC and Jasmin.
Keywords: Parser, JavaCC and Jasmin.

1 – INTRODUÇÃO

Desde os primórdios da computação, desenvolver meios eficientes que se obter o controle e otimizar o uso dos recursos computacionais disponíveis constituem-se em objetos de estudo dos cientistas da computação.

A concepção de uma linguagem pode ocorrer por meio das mais diversas justificativas, a exemplo da linguagem Java, que surgiu a partir da necessidade de se criar uma linguagem para atender a demanda de dispositivos embarcados, nos anos 90. Neste contexto, às vezes, necessidades como resolver problemas de limitação de hardware ou melhorar o desempenho dos recursos computacionais no nível do software ou mesmo para resolver algum problema em especial, fizeram surgir os compiladores.

Todo este esforço e a variedade de linguagens de programação e de compiladores disponíveis justifica-se pelo fato de que programar em linguagem de baixo nível tornou-se cada vez mais complicado, na mesma proporção em que surgem novos recursos computacionais.

Abordar os aspectos mais relevantes sobre o tema compiladores é o foco deste trabalho, que aponta para a concepção de um novo compilador através das ferramentas Java JavaCC e Jasmin.

2 – PRINCÍPIOS DE UM COMPILADOR

Um compilador é um software que traduz um programa descrito em uma linguagem de alto nível para um programa equivalente em código de máquina para um processador. Em geral, um compilador não produz diretamente o código de máquina, mas sim, um programa em linguagem simbólica *(assembly)* semanticamente equivalente ao programa em linguagem de alto nível. O programa em linguagem simbólica é, então, traduzido para o programa em linguagem de máquina através de montadores.

Para realizar esta tarefa o compilador executa a análise léxica, sintática e semântica do código-fonte do programa que esta sendo executada em linguagem abstrata para depois gerar o código de máquina.

Com o advento do computador de programa armazenado de John Von Neumann no final da década de 1940, tornou-se necessário escrever seqüências de código (ou programas) para que os computadores efetuassem as computações desejadas. Inicialmente, estes programas foram escritos em linguagem de Máquina, ou seja, códigos numéricos representando as operações de máquina e serem efetivamente executadas (LOUDEN, 2004). Ao longo dos anos 50, os compiladores foram considerados programas notoriamente difíceis de escrever, pois, o primeiro compilador FORTRAN, por exemplo, consumiu 18 homens-ano para programar, levando a descobertas técnicas sistemáticas para o tratamento de muitas das mais importantes tarefas que ocorreram durante a compilação, e seguindo esta evolução, as técnicas de implementação em linguagens abstratas também se tornaram cada vez mais avançadas (AHO, 1996). O nome compilador faz referência ao processo de composição de um programa pela reunião de varias rotinas de bibliotecas, a tradução de linguagem abstrata para linguagem de baixo nível que é executada pelo compilador. O compilador tem duas tarefas básicas: Análise, que examina, verifica e compreende o texto de entrada (no código-fonte); Geração de código, em que o

texto de saída (código de maquina) é gerado de forma correspondente ao texto de entrada (código-fonte da linguagem abstrata).

2.1 – ANALISADORES: LÉXICO, SINTÁTICO E SEMÂNTICO

A análise léxica é o processo de analisar a entrada de linhas de caracteres (tal como o código-fonte de um programa de computador) e produzir uma seqüência de símbolos chamados "símbolos léxicos" (*lexical tokens*), ou somente "símbolos" (*tokens*), que podem ser manipulados mais facilmente por um parser (leitor de saída). A Análise Léxica é a forma de verificar determinado alfabeto, quando analisamos uma palavra podemos definir através da análise léxica se existe ou não algum caracter que não faz parte do nosso alfabeto, ou um alfabeto inventado por nós. O analisador léxico é a primeira etapa de um compilador, logo após virá a análise sintática (LOUDEN, 2004)..

O analisador sintático é considerado o coração do compilador, sendo responsável por verificar se a sequencia de simbolos contida no código-fonte compõe um programa válido ou não. A sintaxe de uma linguagem de programação pode ser descrita por uma gramática independente de contexto e representada gráficamente através da notação BNF (*Backus-Naur Form* ou Forma de Backus-Naur). Esta gramática descreve recursivamente a combinatória de *tokens* possíveis de uma linguagem. Em termos práticos, pode também ser usada para decompor um texto em unidades estruturais para serem organizadas dentro de um bloco. O analisador sintático recebe do analisador léxico um grupo de *tokens* e usa um conjunto de regras para construir uma árvore sintática da estrutura. A vasta maioria dos analisadores sintáticos implementados em compiladores aceitam alguma linguagem livre de contexto para fazer a análise (LOUDEN, 2004)..

A análise semântica é a terceira fase da compilação, que recebe este nome pois requer a computação de informações que estão alem da capacidade das gramaticas livres de contexto e dos algaritmos padrão de analise sintática, portanto estas informações não podem ser consideradas sintaxe. A informação computada também esta fortemente relacionada com o significado, ou seja, a semântica do programa traduzido (LOUDEN, 2004). É neste analisador que se verifica os erros semânticos,

(por exemplo, uma multiplicação entre tipos de dados diferentes) no código-fonte, ou seja, se não exitem incoerências com os signficados das construções utilizadas pelo programador, e coleta as informações necessárias para a próxima fase da compilação que é a geração de código-fonte. A verificação realizada pelo analisador semântico deve procurar por erros como: Se os tipos de operadores são compatíveis com os da linguagem; Variáveis não declaradas; Chamadas de funções ou métodos com um número incorreto de parâmetros; Comandos colocados fora de contexto.
Para que estas verificações sejam executadas o analisador depende de uma tabela de simbolos, onde estão armazenados informações de variáveis declaradas, funções ou métodos, tipos ou classes. Somente com a tabela de simbolos é possivel realizar a validação semântica do programa.

2.2 – GERADORES DE CÓDIGO

Uma vez verificado que não existem erros sintáticos ou semânticos, o compilador pode realizar sua tarefa, que é a criação do código-objeto. Em geral, o código-objeto é armazenado num arquivo que pode ser posteriormente linkeditado com outros códigos-objeto ou simplesmente carregado na memória do computador e executado pelo sistema operacional (DELEMARO, 2004). Dentro do diversificado leque de categorias de ferramentas que prestam apoio às atividades da Engenharia de Software, uma específica vem ganhando cada vez mais destaque e, sobre ela, tem-se aplicado muito investimento nos últimos tempos: as Ferramentas de Geração de Código, ou simplesmente Geradores de Código. Dessa forma, Gerador de Código é aquela ferramenta que possui a capacidade de gerar código a partir de um determinado modelo de software. Inclusive, de acordo com alguns pontos de vista e a partir das características específicas do tipo de Gerador de Código, ele passa a ser conversor de códigos de linguagens distintas. Isso acontece, por exemplo, com o compilador, que transforma um código escrito através de uma linguagem de programação para código de máquina ou código-objeto .

3 – AS ETAPAS PARA A CRIAÇÃO DE UM COMPILADOR

O objetivo de se criar um compilador vem da nessecidade de se resolver um determinado problema através de uma nova linguagem abstrata (linguagem de alto nivel) especifica, identificar os propósitos desta linguagem, saber se esta vai ser utilizada para fins comercial, ciêntifico ou de propósitos gerais. Tendo definido isto o proximo passo é a contrução de uma gramática BNF para a nova linguagem e, conseqüentemente, escrever os analisadores e gerador de código.

Atualmente, um dos meios para se projetar um compilador é a utilização das ferramentas JavaCC (que é um gerador de analisador sintático que gera código em linguagem Java) e o Jasmin que gera o código de maquina semanticamente ao código-fonte das linguagens abstratas.

O JavaCC é um programa gerador de compiladores ou mais precisamente um gerador de analisador sintático. Ele toma como entrada uma gramática e transforma-a num programa descrito em linguagem Java capaz de analisar um arquivo e dizer se satisfaz ou não as regras especificadas nesta gramática. Oferece facilidades para a construção da árvore sintática. O programa gerado pelo JavaCC realiza um tipo de análise sintática chamada de top down ou análise descendente (DELAMARO, 2004).

O Jasmin é uma interface de montagem para Java (Java ASseMbler INterface), que toma como entrada um arquivo ASCII com instruções JVM (Java Virtual Machine) e produz um arquivo executável para a JVM (arquivo ".class"). Assim pode-se verificar se código gerado esta correto ou não. Porém, para que possamos executar o código, precisamos executar mais uma tarefa, que é transformá-lo num arquivo executável Java, utilizando um programa Jasmin (DELAMARO, 2004).

4 – CONSIDERAÇÕES FINAIS

Percebe-se que a construção de um compilador não é uma tarefa trivial, mas cujos estudos não geram apenas um conhecimento técnico em específico. A rigor, da nova tendência do uso de técnicas de programação linear nos ambientes corporativos como ferramentas de tomada de decisão que leva, não só ao apontamento da solução ótima de um problema, mas também a uma melhor compreensão do problema em questão, assim também ocorre com os estudos das técnicas de implementação de um compilador, pois além dos conhecimentos

técnicos, leva a um entendimento apurado sobre as limitações de máquinas e como estas podem ser resolvidas pelo software.

Atualmente, existem diversas formas de se implementar um compilador, mas ressalta-se neste trabalho o uso de duas ferramentas Java: JavaCC e Jasmin. Dotadas de vasta biblioteca sobre a gramática de linguagens como C, SQL e Java, estas ferramentas possibilitam a uma imersão direta no mundo dos compiladores, abrindo novos horizontes, em nível de compreensão das arquiteturas computacionais, melhorando a qualidade de software produzido.

Pode-se afirmar que o conhecimento gerado através da implementação de um compilador torna-se um grande diferencial ao profissional de tecnologia de informação que se propõe a desenvolver um software de qualidade, através de boas práticas de programação.

5 – REFERÊNCIAS BIBLIOGRÁFICAS

LOUDEN, K.C. **Compiladores – Princípios e Praticas**. Ed. Thomson Pioneira, 2004.

DELAMARO, M.E. **Com construir compilador utilizando ferramentas Java.** Ed. Novatec, São Paulo. 2004

AHO, S. **Compiladores: Princípios, técnicas e ferramentas**, Ed. LTC. 1996.