Construcción de Compiladores - Laboratorio Guiado No. 1

Gabriel Brolo, Bidkar Pojoy

Julio de 2024

1. Introducción

En este laboratorio, exploraremos el proceso de construcción de un compilador utilizando herramientas como Lex y Yacc. Utilizaremos un lenguaje de programación simple que soporta expresiones aritméticas y asignaciones de variables. **Lea todo el documento antes de comenzar a realizar las actividades**.

2. Gramática del Lenguaje

La gramática de nuestro lenguaje es la siguiente:

```
program: statement_list

statement_list: statement
              | statement_list statement

statement: assignment
         | expression

assignment: ID '=' expression

expression: NUMBER
          | ID
          | expression '+' expression
          | expression '-' expression
          | expression '*' expression
          | expression '/' expression

ID: [a-zA-Z][a-zA-Z0-9]*
NUMBER: [0-9]+
```

3. Instrucciones para usar Lex y Yacc

Puede seguir estos pasos para utilizar Lex y Yacc, pero se recomienda que consulte el material asociado que contiene una guía de uso más robusta para Lex y Yacc:

1. Guarde el código Lex en un archivo llamado `simple_language.lex` y el código Yacc en un archivo llamado `simple_language.y`.
2. Ejecute Lex para generar el código del analizador léxico: `lex simple_language.lex`.

3. Compile el código del analizador léxico: `gcc lex.yy.c -o lexer`.
4. Ejecute Yacc para generar el código del analizador sintáctico: `yacc -d simple_language.y`.
5. Compile el código del analizador sintáctico: `gcc y.tab.c -o parser`.
6. Ejecute el analizador sintáctico: `./parser`.
7. Ingrese expresiones o asignaciones según la gramática definida.

4. Instrucciones para usar Docker

Este laboratorio viene con un Dockerfile y docker-compose que puede utilizar para ejecutar el entorno utilizando Docker, sin necesidad de instalar Lex y Yacc en su máquina. Se recomienda que siga los siguientes pasos y que consulte la documentación oficial de Docker o Rancher Desktop para correr su entorno local:

1. Abra una terminal en el directorio donde se encuentra el archivo Dockerfile y el código fuente del compilador.
2. Ejecute el siguiente comando para construir la imagen del contenedor:

   ```
   docker build -t compiler_env .
   ```

3. Una vez que la imagen se haya construido correctamente, puede ejecutar el contenedor con el siguiente comando:

   ```
   docker run -it compiler_env
   ```

4. También puede correr el dockerfile adjuntado:

   ```
   docker compose up --build -d
   ```

5. Y luego eliminarlo:

   ```
   docker compose down --volumes
   ```

5. Aterrizando conceptos

Cuando utilizamos Lex y Yacc para construir un compilador, estamos trabajando en dos etapas importantes del proceso de compilación: análisis léxico y análisis sintáctico.

Análisis Léxico (Lex):

- **¿Qué es el análisis léxico?**: El análisis léxico es la primera fase del proceso de compilación, donde el programa fuente se escanea caracter por caracter para identificar los tokens válidos del lenguaje.
- **¿Qué hace Lex en este proceso?**: Lex toma una especificación de expresiones regulares y genera un analizador léxico en C que identifica los tokens en el programa fuente.
- **¿Por qué es importante?**: Es crucial porque proporciona una estructura básica para el análisis del código fuente y facilita la identificación de tokens como identificadores, números, operadores, etc.

Análisis Sintáctico (Yacc):

- **¿Qué es el análisis sintáctico?**: El análisis sintáctico es la segunda fase del proceso de compilación, donde se verifica si el programa fuente cumple con la gramática del lenguaje.
- **¿Qué hace Yacc en este proceso?**: Yacc toma una especificación de la gramática del lenguaje y genera un analizador sintáctico en C que verifica la estructura y el orden de las construcciones del lenguaje.
- **¿Por qué es importante?**: Es fundamental porque garantiza que el programa fuente tenga una estructura sintácticamente válida, lo que facilita la generación del código objeto y su posterior ejecución.

6. Actividades por completar

Utilice el lenguaje definido y las herramientas Lex y Yacc para realizar las siguientes tareas:

1. Cree un programa que asigne un valor a una variable.
2. Cree un programa que realice una operación aritmética simple.
3. Experimente con expresiones más complejas y verifique que el compilador las procese correctamente.
4. Modifique el lenguaje para incluir la asignación de variables con expresiones aritméticas.
5. Agregue manejo de errores al compilador para detectar tokens inválidos en el programa fuente.
6. Experimente con la precedencia de operadores en el lenguaje y observe cómo afecta la generación del árbol sintáctico.

7. Entregables

- Documento PDF con capturas de pantalla de la ejecución del ambiente en Docker de Lex y Yacc, o de su propio entorno, para realizar las actividades solicitadas.
- El documento debe de incluir el enlace a un repositorio privado de Github con todo el código utilizado para la realización de este laboratorio guiado. Recuerde que debe de compartir todos los repositorios utilizados en este curso con el catedrático y los auxiliares. La información de los usuarios se encuentra en Canvas.

8. Rúbrica

La siguiente rubrica será utilizada para asignar puntos a las tareas anteriores:

Tarea	**Puntos**
Crear un programa de asignación	10
Crear un programa de operación	15
Agregar manejo de errores	15
Experimentar con la precedencia	15
Completar todas las tareas	10
Respuestas comprensibles y claras	10
Seguimiento de instrucciones	10
Organización y presentación adecuada	10