

ITCR – IC5701.COMPILADORES E INTÉRPRETES

TAREA #1 - HERRAMIENTA: ANTLR

Estudiantes:

Antonio Fernández García

Josué Mena González

Marco Rodríguez Vargas

Profesor:

Emmanuel Ramírez S.

II Semestre 2024



ANTLR

- **Descripción**

ANTLR (ANother Tool for Language Recognition) es una herramienta que se utiliza para generar analizadores (parsers) a partir de gramáticas definidas en un formato específico. Su principal uso es en el desarrollo de lenguajes de programación, traductores de lenguajes, y en la creación de herramientas de análisis de código. Fue desarrollada por Terence Parr, un profesor de Ciencias de la Computación en la Universidad de San Francisco, esta herramienta es Open Source bajo la licencia BSD, actualmente sigue siendo mantenida y actualizada, principalmente por Parr y la comunidad, su última versión es la 4.13.2



ANTLR

- **Enlaces de Descarga**

ANTLR puede ser descargada desde su sitio oficial en
<https://www.antlr.org/download.html>

Así mismo y al ser Open Source, su código está disponible en
<https://github.com/antlr/antlr4/tree/master>

Para descargar versiones anteriores se pueden conseguir en
<https://github.com/antlr/website-antlr4/tree/gh-pages/download>



ANTLR

Lenguajes

Entre los lenguajes con los que se puede integrar se tienen los siguientes:

- *Java*
- *C#*
- *JS*
- *TS*
- *Go*
- *C++*
- *PHP*



ANTLR

- **Etapas de generación de procesadores de lenguaje**

ANTLR genera un parseador (analizador sintáctico) específicamente y también puede generar un analizador sintáctico en conjunto. Su funcionalidad es muy útil ya que ayuda a construir de forma automática el AST. Por otra parte, también implementa los "tree walkers" que se usan para recorrer el AST y visitar sus nodos.

El algoritmo utilizado es el conocido $LL(*)$ o $LL(K)$ (visto en clase) para generar el analizador sintáctico. Sus funcionalidades le permiten transformarse en un intérprete también.



ANTLR

- **Cómo se utiliza?**

La utilidad de esta herramienta se basa en dos puntos importantes:

- Generar el lexer y parser para procesar gramáticas.
- Generar el árbol de parseo para una entrada específica.

Con esta herramienta es bastante sencillo generar el código para procesar las gramáticas y posteriormente pasarle entradas que generen un árbol de forma visual con la interfaz gráfica. La interfaz gráfica para visualizar el árbol viene con la instalación de una vez.

Para lograr esto basta con simplemente escribir una gramática en un archivo.g4

Posteriormente se ejecuta el comando “`antlr4 archivo.g4`” para generar el código del Lexer y Parser.

Si se quiere probar algún ejemplo se deben compilar los archivos y ejecutar “`grun archivo primera regla -gui`”



ANTLR

- **Compatibilidad con otras herramientas**

- ANTLR es una herramienta muy compatible con otros lenguajes y plataformas, como Java (utilizado en este ejemplo).
- Al trabajar con el JDK, ANTLR permite generar analizadores léxicos y sintácticos que se pueden integrar en proyectos de lenguajes como como Java, C#, JavaScript, TypeScript, Go, C++, y PHP.
- ANTLR depende de Java para su ejecución y se puede integrar fácilmente en flujos de trabajo automatizados con herramientas como Maven o Gradle, lo que facilita la generación y compilación de los analizadores de manera automática.
- También permite la visualización de árboles sintácticos mediante una interfaz gráfica en Java, lo que lo hace muy útil en entornos de desarrollo.



ANTLR

- Ejemplo. Crear un archivo Expr.g4 con el siguiente contenido.

```
Expr.g4
~/Documents/2024-S2/Compi/Tarea1

1 grammar Expr;
2
3 prog:    expr EOF ;
4 expr:    expr ('*' | '/') expr
5         | expr ('+' | '-') expr
6         | INT
7         | '(' expr ')'
8         ;
9
10 NEWLINE : [\r\n]+ -> skip;
11 INT     : [0-9]+ ;
```




ANTLR

- Ejemplo. Este comando descarga el archivo `antlr-4.8-complete.jar` en el directorio `~/Documents/2024-S2/Compi/Tarea1`.

The screenshot displays a terminal window on the left and a file manager on the right. The terminal window shows the execution of the `wget` command to download `antlr-4.8-complete.jar` from `https://www.antlr.org/download/antlr-4.8-complete.jar` into the directory `~/Documents/2024-S2/Compi/Tarea1`. The output indicates a successful download of 2,089,101 bytes (2.0 MB) at a speed of 6.52 MB/s. The file manager on the right shows the directory `~/Documents/2024-S2/Compi/Tarea1` containing the downloaded file `antlr-4.8-complete.jar` and another file `Expr.g4`.

```
marco@marco-B660M-AORUS-PRO-AX-DDR4: ~/Documents/2024-S2/Compi/Tarea1
marco@marco-B660M-AORUS-PRO-AX-DDR4:~/Documents/2024-S2/Compi/Tarea1$ wget https://www.antlr.org/download/antlr-4.8-complete.jar -O ~/Documents/2024-S2/Compi/Tarea1/antlr-4.8-complete.jar
--2024-10-08 22:51:50-- https://www.antlr.org/download/antlr-4.8-complete.jar
Resolving www.antlr.org (www.antlr.org)... 185.199.110.153, 185.199.111.153, 185.199.109.153, ...
Connecting to www.antlr.org (www.antlr.org)|185.199.110.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2089101 (2.0M) [application/java-archive]
Saving to: '/home/marco/Documents/2024-S2/Compi/Tarea1/antlr-4.8-complete.jar'

/home/marco/Documents/2024-S2/Compi/Tarea1/antlr-4.8-complete.jar 100%[=====>] 1,99M 6,52MB/s in 0,3s

2024-10-08 22:51:50 (6,52 MB/s) - '/home/marco/Documents/2024-S2/Compi/Tarea1/antlr-4.8-complete.jar' saved [2089101/2089101]
```



ANTLR

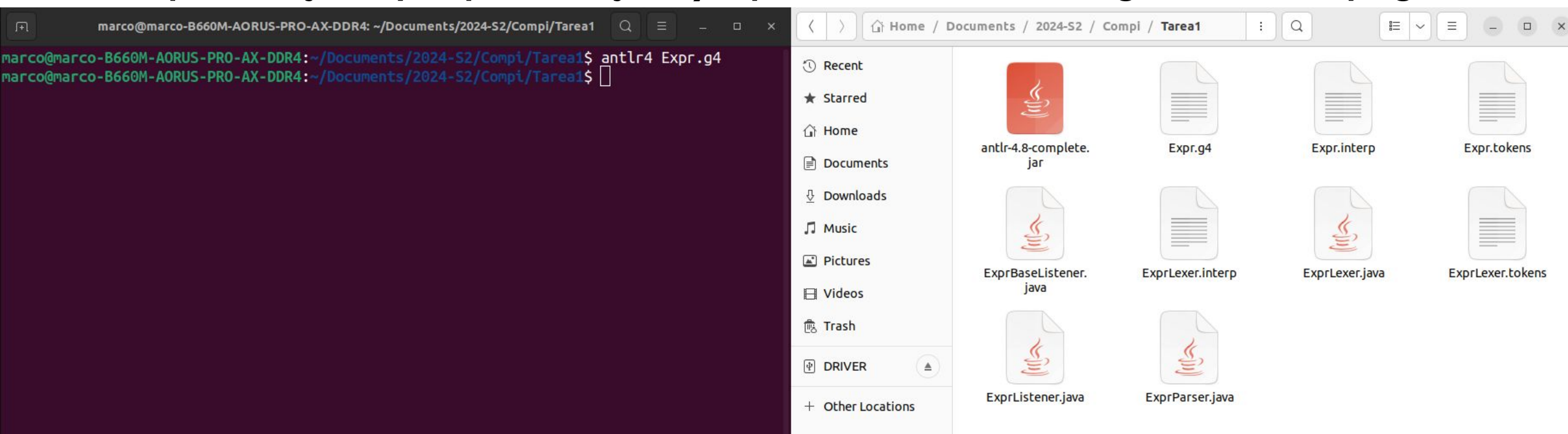
- Ejemplo. Configurar con un alias para facilitar la ejecución de antlr4 y grun.

```
marco@marco-B660M-AORUS-PRO-AX-DDR4: ~/Documents/2024-S2/Compi/Tarea1
marco@marco-B660M-AORUS-PRO-AX-DDR4:~/Documents/2024-S2/Compi/Tarea1$ alias grun='java -cp
.:~/home/marco/Documents/2024-S2/Compi/Tarea1/antlr-4.8-complete.jar org.antlr.v4.gui.Test
Rig'
marco@marco-B660M-AORUS-PRO-AX-DDR4:~/Documents/2024-S2/Compi/Tarea1$ source ~/.bashrc
marco@marco-B660M-AORUS-PRO-AX-DDR4:~/Documents/2024-S2/Compi/Tarea1$
```



ANTLR

- Ejemplo. Este comando genera los archivos .java necesarios (como ExprLexer.java y ExprParser.java) a partir del archivo de gramática Expr.g4.





ANTLR

- Ejemplo. Este comando compila los archivos .java generados por ANTLR utilizando el archivo JAR de ANTLR.

The screenshot displays a terminal window on the left and a file explorer on the right. The terminal window shows the execution of the `javac` command to compile the generated Java files using the ANTLR JAR file.

```
marco@marco-B660M-AORUS-PRO-AX-DDR4: ~/Documents/2024-S2/Compi/Tarea1
marco@marco-B660M-AORUS-PRO-AX-DDR4:~/Documents/2024-S2/Compi/Tarea1$ javac -cp "../home/marco/Documents/2024-S2/Compi/Tarea1/antlr-4.8-complete.jar" Expr*.java
marco@marco-B660M-AORUS-PRO-AX-DDR4:~/Documents/2024-S2/Compi/Tarea1$
```

The file explorer on the right shows the contents of the `~/Documents/2024-S2/Compi/Tarea1` directory. It contains the following files:

- `antlr-4.8-complete.jar` (JAR file)
- `Expr.g4` (ANTLR grammar file)
- `Expr.interp` (ANTLR intermediate representation file)
- `Expr.tokens` (ANTLR token definitions file)
- `ExprBaseListener.class` (Compiled Java class)
- `ExprBaseListener.java` (Generated Java source file)
- `ExprLexer.class` (Compiled Java class)
- `ExprLexer.interp` (ANTLR intermediate representation file)
- `ExprLexer.java` (Generated Java source file)
- `ExprLexer.tokens` (ANTLR token definitions file)
- `ExprListener.class` (Compiled Java class)
- `ExprListener.java` (Generated Java source file)
- `ExprParser.class` (Compiled Java class)
- `ExprParser.java` (Generated Java source file)
- `ExprParser$ExprCo` (Generated Java source file)
- `ExprParser$ProgCo` (Generated Java source file)



ANTLR

- Ejemplo. Después de ejecutar el comando, ingresa una expresión matemática (por ejemplo, $10+20*30$), presiona Enter, y luego Ctrl + D para indicar el final de la entrada.

The screenshot shows a terminal window on the left and a Parse Tree Inspector window on the right. The terminal window displays the command `grun Expr prog -gui` and the input `10+20*30`. The Parse Tree Inspector window shows the resulting parse tree for the expression `10+20*30`.

Terminal Window:

```
marco@marco-B660M-AORUS-PRO-AX-DDR4: ~/Documents/2024-S2/Compi/Tarea1
marco@marco-B660M-AORUS-PRO-AX-DDR4:~/Documents/2024-S2/Compi/Tarea1$ grun Expr prog -gui
10+20*30

```

Parse Tree Inspector:

The parse tree structure is as follows:

```
graph TD
    prog --> expr1[expr]
    prog --> EOF["<EOF>"]
    expr1 --> expr2[expr]
    expr1 --> plus["+"]
    expr1 --> expr3[expr]
    expr2 --> 10["10"]
    expr3 --> expr4[expr]
    expr3 --> star["*"]
    expr3 --> expr5[expr]
    expr4 --> 20["20"]
    expr5 --> 30["30"]
```

The Parse Tree Inspector window also includes buttons for `OK`, `Export as PNG`, and `Export as SVG`.



ANTLR

- **Bibliografía**

<https://www.antlr.org/>

<https://github.com/antlr/antlr4/blob/master/doc/index.md>