

Trabajo Práctico Integrador



Materia: Sintaxis y Semántica de los Lenguajes
Trabajo Práctico Integrador

Intérprete de lenguaje RSS

Primer Cuatrimestre 2022

Integrantes:

- **Arduña Zago**, Agustín Juan Luis (Delegado)
- **Schefer**, Mauricio Nicolás
- **Segnana**, Juan Franco
- **Velazco Gez Schegtél**, Juan Ignacio

Primera entrega: 24/04/2022

Segunda entrega: 28/05/2022

Tercera entrega: 03/07/2022

Versión del documento:3

Introducción	3
Lenguaje RSS	3
Componentes léxicos o tokens	3
Observaciones	3
Etiquetas	3
Reglas	3
Gramática	6
Descripción de la gramática	6
Producciones	7
Terminales	9
Analizador Léxico	10
Ejecución del lexer	10
Línea por línea	10
Analizar un archivo de texto	11
Analizador Sintáctico	13
Definición de Tokens	13
Funciones auxiliares	14
Modo de obtención del intérprete	14
Ejecutar intérprete	15
Ejemplos	19
Básico con elementos obligatorios	19
Completo con elementos opcionales	19
Conclusión	20
Bibliografía	21

Introducción

El presente trabajo, realizado para la cátedra de Sintaxis y Semántica de los Lenguajes, en la Facultad Regional Resistencia, tiene como objetivo la creación de un Lexer y Parser que pueda interpretar el lenguaje RSS. En esta sección se definirán las reglas gramaticales a tener en cuenta para la elaboración de intérprete que pueda reconocer el lenguaje RSS. Se definió que, para el intérprete a desarrollar en próximas ediciones, se usará el lenguaje Python debido a su sencillez y rapidez en cuanto al uso y su curva de aprendizaje. Además de utilizar la dependencia [PLY](#), ya que permite fácilmente integrar la gramática definida.

Lenguaje RSS

Componentes léxicos o tokens

Observaciones

- Las etiquetas se distinguirá entre mayúsculas y minúsculas.
- Se podrán usar etiquetas opcionales, como `<category>`, `<copyright>` o `<image>`.

Etiquetas

- Las etiquetas reservadas, al inicializarse, estarán encerradas por los símbolos `<>` y al finalizar su uso se cerrarán con `</etiqueta>`.
- Cada etiqueta puede contener atributos del formato: `nombreAtributo="contenidoAtributo"`.
- Las etiquetas se usarán en minúscula.

Reglas

Versión XML:

- Aparece una sola vez y no tiene etiqueta de cierre.
- Atributos: *version* (1.0), *encoding* (UTF-8).

Versión RSS:

- Aparece una sola vez, utiliza *version* 2.0.

channel:

El elemento `<channel>`, aparece por única vez, tiene tres elementos secundarios obligatorios:

- `<title>`: define el título del canal (p. ej., página de inicio de la cátedra)
- `<link>`: define el hipervínculo al canal, ver etiqueta link más abajo.
- `<description>`: descripción del canal (por ejemplo, RSS de la cátedra de Sintaxis y Semántica de lenguajes)
- Además posee otros elementos secundarios opcionales, de los cuales vamos a destacar e implementar los siguientes:
 - `<category>`: define una categoría para agrupamiento
 - `<copyright>`: informa sobre los autores y derechos o restricciones.

- <image>: una imagen que describe al canal. Ver etiqueta de image más abajo.
- Nota: Otros elementos opcionales no se solicitan en el trabajo , pero si quieren pueden implementarlos.

item:

Cada elemento <channel> puede tener uno o más elementos <item>.

- Cada elemento <item> define un artículo o "historia" en el feed RSS.
- El elemento <item> tiene tres elementos secundarios obligatorios:
 - <title>: define el título del elemento (por ejemplo, Enunciado TPI)
 - <link>: define el hipervínculo al elemento (por ejemplo, <https://www.>) etiqueta link más abajo.
 - <description>: describe el elemento (p. ej., Enunciado del TPI 2022)
- Además posee otros elementos secundarios opcionales, de los cuales vamos a destacar e implementar los siguientes:
 - <category>: define una categoría para agrupamiento
- Nota: Otros elementos opcionales no se solicitan en el trabajo , pero si quieren pueden implementarlos.

Image:

- Se puede incluir una imagen por canal.
- El elemento <image> tiene tres elementos secundarios obligatorios:
 - <url> : Define la URL a la imagen, (GIF, JPEG or PNG) ver etiqueta link más abajo.
 - <title> : Define el texto a mostrar si la imagen no puede ser cargada
 - <link> : Define el hipervínculo al elemento. ver etiqueta link más abajo.
- Además posee otros elementos secundarios opcionales, de los cuales vamos a destacar e implementar los siguientes:
 - <height>: define la altura de la imagen. máximo es 400
 - <width>: define ancho de la imagen. máximo es 144

Link:

- El elemento <link> permite incluir una URL.
- Una URL (Uniform Resource Location) es la dirección concreta de un recurso en Internet.
- La sintaxis completa de una URL es la siguiente:

protocolo://dominio:puerto/ruta#fragmento

Se detalla el significado de todos los elementos

- Los únicos caracteres permitidos en URL son letras, números, guión medio, guión bajo y punto. además de los caracteres reservados: # , / , :
- El protocolo o esquema de red. Hace referencia al nombre de protocolo de red necesario para poder alcanzar el recurso al que hace referencia la URL. Protocolos habituales son:
 - http:// (para recursos de la web)
 - https:// (para recursos de la web contenidos en un servidor seguro)
 - ftp:// (recursos contenidos en un servidor de ficheros)
 - ftps:// (recursos contenidos en un servidor de ficheros seguro)

Tras el protocolo se indican dos puntos, tras los cuales normalmente se indican dos barras para indicar la máquina.

- Nombre de dominio. Nombre completo en Internet de la máquina (o la red) que posee el recurso en forma de nombre de dominio.no distinguen entre mayúsculas y minúsculas.
- Puerto. Opcional. Puerto por el que se debe conectar con el servidor para obtener el recurso. Si no se indica (que es lo habitual) se toma el puerto por defecto. Por ejemplo en http el puerto por defecto es el 80. Si queremos usar uno en particular se indica tras el servidor poniendo dos puntos y el puerto.
- Ruta. Opcional. Indica el recorrido dentro de la máquina remota que hay que hacer a través de los directorios para llegar al recurso que queremos. Se pone después del servidor. Ejemplos:
 - /index.html Accede a la página index.html situada en el directorio raíz.
 - /imagenes/paisajes/foto001.jpg Accede a la imagen foto001.jpg dentro del directorio paisajes dentro, a su vez, del directorio imágenes.
- Localizador interno. Opcional. Va detrás del símbolo # y sirve para indicar un identificador (que puede ser más o menos complejo) que permita localizar o seleccionar una parte concreta del recurso destino de la URL.

Gramática

Descripción de la gramática

VERSION → indica versión del archivo XML

RSS → indica versión del RSS y permite producciones de tipo Channel

CHANNEL → conjunto de producciones de tipo Channel, permite todas las etiquetas obligatorias y las opcionales. Además se incluye el conjunto de producción de tipo ITEM (1 o más ítems).

ENTERO → numérico

ET_OBL → conjunto de etiquetas obligatorias y opcionales de tipo channel.

ET_TITLE → sentencia de tipo title

ET_LINK → sentencia de tipo link

ET_URL → sentencia de tipo url

ET_DESC → sentencia de tipo descripcion

ET_CATEGORY → sentencia de tipo category

ET_COPYRIGHT → sentencia de tipo copyright

CONT_TEXTO → tipo de dato cadena. Permite que el contenido de la etiqueta sea alfanumérico.

CONT_IMG → conjunto de etiquetas obligatorias y opcionales para IMAGE

ET_IMG_OBL → conjunto de etiquetas obligatorias para IMAGE

ET_HEIGHT → sentencia de tipo height

ET_WIDTH → sentencia de tipo width

ET_IMG_OP → conjunto de etiquetas opcionales para IMAGE

CONT_LINK → conjunto de producciones que permite formar un LINK

PROTOCOLO → conjunto de posibles protocolos en una URL

DOMINIO_GRAL → permite inclusión de producciones alfanuméricas y numéricas para tanto el dominio como el puerto (opcionalmente)

FINAL_URL → conjunto de producciones que acepta el final de un LINK

RUTA → cadena alfanumérica

LOCALIZADOR → cadena alfanumérica

DOMINIO → cadena alfanumérica (permite cualquier dirección, ejemplo "fre.utn.com")

PUERTO → numérico

ITEM_REC → permite recursividad para aceptar uno o más ítems.

ET_ITEM → conjunto de etiquetas de tipo item

ET_OBL_ITEM → conjunto de etiquetas opcionales y obligatorias de tipo ITEM. En las próximas entregas se eliminarán todas las posibles combinaciones para una etiqueta item y se harán verificaciones desde el código.

Producciones

(!) **Aclaración:** Mayúsculas = No Terminales; Minúsculas = Terminales.

SIGMA → VERSION RSS

VERSION → <?xml version="1.0" encoding="UTF-8" ?>

RSS → <rss version="2.0">**CHANNEL**</rss>

CHANNEL → <channel>**ET_OBL ITEM_REC**</channel>

ET_OBL → **ET_TITLE ET_LINK ET_DESC ET_CATEGORY ET_COPYRIGHT ET_IMG**

ET_TITLE → <title>**CONT_TEXTO**</title>

ET_LINK → <link>**CONT_LINK**</link>

ET_URL → <url>**CONT_LINK**</url>

ET_DESC → <description>**CONT_TEXTO**</description>

ET_CATEGORY → <category>**CONT_TEXTO**</category> | Lambda

ET_COPYRIGHT → <copyright>**CONT_TEXTO**</category> | Lambda

CONT_TEXTO → Cadena **CONT_TEXTO** | Cadena

CONT_IMG → **ET_IMG_OBL ET_IMG_OP** | **ET_IMG_OBL**

ET_IMG → <image>**CONT_IMG**</image> | Lambda

ET_IMG_OBL → **ET_TITLE ET_LINK ET_URL** | **ET_TITLE ET_URL ET_LINK** | **ET_URL**

ET_TITLE ET_LINK | **ET_URL ET_LINK ET_TITLE** | **ET_LINK ET_TITLE ET_URL** |

ET_LINK ET_URL ET_TITLE

ET_IMG_OP → **ET_HEIGHT ET_IMG_OP** | **ET_WIDTH ET_IMG_OP** | **ET_HEIGHT** | **ET_WIDTH**

ET_HEIGHT → <height>Numero</height>

ET_WIDTH → <width>Numero</width>

Tipo de dato: URL

CONT_LINK → **PROTOCOLO://DOMINIO_GRAL FINAL_URL** |

PROTOCOLO://DOMINIO_GRAL

FINAL_URL → **/RUTA#LOCALIZADOR** | **/RUTA**

PROTOCOLO → http | https | ftp | ftps

DOMINIO_GRAL → **DOMINIO:PUERTO** | **DOMINIO**

LOCALIZADOR → Cadena alfanumérica

DOMINIO → Cadena alfanumérica

RUTA → Cadena alfanumérica

PUERTO → Numérico

ENTERO → Numérico

ITEM_REC → ET_ITEM ITEM_REC | ET_ITEM

ET_ITEM → <item>ET_OBL_ITEM</item>

ET_OBL_ITEM →

ET_TITLE ET_DESC ET_LINK ET_CATEGORY
ET_TITLE ET_DESC ET_CATEGORY ET_LINK
ET_TITLE ET_CATEGORY ET_DESC ET_LINK
ET_TITLE ET_CATEGORY ET_LINK ET_DESC
ET_TITLE ET_LINK ET_CATEGORY ET_DESC
ET_TITLE ET_LINK ET_DESC ET_CATEGORY

ET_DESC ET_TITLE ET_LINK ET_CATEGORY
ET_DESC ET_TITLE ET_CATEGORY ET_LINK
ET_DESC ET_CATEGORY ET_TITLE ET_LINK
ET_DESC ET_CATEGORY ET_LINK ET_TITLE
ET_DESC ET_LINK ET_CATEGORY ET_TITLE
ET_DESC ET_LINK ET_TITLE ET_CATEGORY

ET_CATEGORY ET_DESC ET_TITLE ET_LINK
ET_CATEGORY ET_TITLE ET_DESC ET_LINK
ET_CATEGORY ET_DESC ET_LINK ET_TITLE
ET_CATEGORY ET_LINK ET_TITLE ET_DESC
ET_CATEGORY ET_LINK ET_DESC ET_TITLE
ET_CATEGORY ET_DESC ET_LINK ET_TITLE

ET_LINK ET_CATEGORY ET_TITLE ET_DESC
ET_LINK ET_CATEGORY ET_DESC ET_TITLE
ET_LINK ET_TITLE ET_DESC ET_CATEGORY
ET_LINK ET_TITLE ET_CATEGORY ET_DESC
ET_LINK ET_DESC ET_CATEGORY ET_TITLE
ET_LINK ET_DESC ET_TITLE ET_CATEGORY

Terminales

- <
- >
- /
- http
- https
- ftp
- ftps
- height
- width
- category
- image
- copyright
- description
- link
- title
- url
- channel
- rss
- =
- “
- .
- ?
- :
- xml
- version
- encoding
- UTF-8

Analizador Léxico

En esta parte del trabajo, como se aclaró en la introducción se usará el lenguaje *Python* y el módulo **PLY** para la realización de tanto el analizador léxico como el sintáctico. Realizamos una lectura de la [documentación del módulo PLY](#), el cual nos ayudó a tener una base de la estructura principal de un lexer.

Lo primero a realizar fue el traspaso de los terminales descritos en la gramática a **expresiones regulares** dentro del lexer. Para la generación de las expresiones regulares, se utilizó, como soporte, a la herramienta [regex101](#), la cual fue de gran ayuda ya que brinda una explicación de las palabras reservadas propias de `Regex`. Además que se pueden agregar textos de entrada, y la página hará la función de un aceptor, indicándonos si la expresión regular brindada servirá o no en el lexer.

Cada terminal o **token** es definido en forma de variable, empezando con la palabra clave `t_nombre`, esto es necesario debido a que el módulo PLY identifica a los tokens de esta manera. Existen algunos tokens que fueron definidos en forma de función, debido a que necesitaban un tratamiento o lógica especial.

En esta entrega, se realizaron pequeñas modificaciones en la gramática, debido a que nos encontramos con algunas dificultades en el momento de, por ejemplo, detectar texto entre dos etiquetas.

Ejecución del lexer

Nuestro lexer cuenta con dos modos de ejecución. Ambos deben ejecutarse desde una terminal. Con el comando: `./bin/ejecutableSO`, según su sistema operativo.

Línea por línea

En este modo, se puede ingresar una sentencia específica y el programa indicará al usuario si es una expresión **correcta**. En caso de que no lo sea, notificará de manera específica el carácter que denota la **ilegalidad** de la expresión.

Capturas

- Funcionamiento esperado

```
Lexer de RSS | Grupo 1. SSL 2022.  
Para salir pulse: [ctrl] + [C] | 0 escriba _salir  
>> <title>Hola mundo</title>  
Tipo: titulo | Valor: <title>  
Tipo: contenido_texto | Valor: Hola mundo  
Tipo: cerrartitulo | Valor: </title>  
>> █
```

- Carácter ilegal

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS  COMMENTS

Lexer de RSS | Grupo 1. SSL 2022.
Para salir pulse: [ctrl] + [C] | 0 escriba _salir
>> <title>No cierra etiqueta!</title>
Tipo: titulo | Valor: <title>
Caracter ilegal! : 'N'.
En línea: 1. Posición: 7
Caracter ilegal! : 'o'.
En línea: 1. Posición: 8
Caracter ilegal! : 'c'.
En línea: 1. Posición: 10
Caracter ilegal! : 'i'.
En línea: 1. Posición: 11
Caracter ilegal! : 'e'.
En línea: 1. Posición: 12
Caracter ilegal! : 'r'.
En línea: 1. Posición: 13
Caracter ilegal! : 'r'.
```

Analizar un archivo de texto

En este modo, cuando se llama al lexer por terminal, se pasa por argumento (con el símbolo `-f` de *file*) la ruta donde se encuentra el archivo a analizar. Luego de que el lexer lo analice, creará un archivo `.txt` donde se detalla cada token analizado, con su correspondiente valor. De igual forma que el modo anterior, también notificará al usuario si hay un carácter ilegal.

Ejemplo de cómo ejecutarlo: `./bin/ejecutableSO -f "./pruebas/clases.rss"`

Capturas

- Funcionamiento esperado

```
Lexer de RSS | Grupo 1. SSL 2022.
(✓) El lexer ACEPTA este archivo.
(!) Se exportó un .txt con los tokens analizados.

~/Documents/grupo-1-ssl-2022 main !3 ?6 .....
> clear && python3 src/lexer.py -f "src/ejemplos/clases.rss"
```

```
tokens-analizados-2022-05-28T15:59:41.765595.txt
1  TOKEN | VALOR
2  -----
3  1- xml: <?xml version="1.0" encoding="UTF-8"?>
4  2- rss: <rss version="2.0">
5  3- channel: <channel>
6  4- titulo: <title>
7  5- contenido_texto: RSS de las clases virtuales
8  6- cerraritulo: </title>
9  7- link: <link>
10 8- contenido_texto: ht=tps://misclases.live/
119- cerrarlink: </link>
1210- description: <description>
1311- contenido_texto: RSS con las clases virtuales de las materias
1412- cerrardescription: </description>
```

```
64 62- contenido_texto: Clase virtual de AM. Practica.  
65 63- cerrardescription: </description>  
66 64- category: <category>  
67 65- contenido_texto: Clase  
68 66- cerrarcategory: </category>  
69 67- cerraritem: </item>  
70 68- cerrarchannel: </channel>  
71 69- cerrarrss: </rss>  
72 -----  
73 Total de tokens válidos analizados: 69.  
74
```

Analizador Sintáctico

Para el analizador sintactico, continuamos el uso del módulo PLY, donde la principal tarea fue importar los tokens, o expresiones regulares, al parser. Así luego comenzamos a transcribir las producciones descritas en este documento. PLY identifica como producciones a aquellas variables o funciones que comienzan con `p_NOMBRE`.

En la realizacion del parser, se modifiko la logica para iniciar la ejecucion del programa. Anteriormente, se debia pasar por argumentos en la terminal si el usuario deseaba analizar un archivo de texto, en esta edición se implementó un menú, donde el usuario podrá elegir entre ejecutar el modo `línea a línea` ó `analizar un archivo de texto`, indicando la ruta.

Ademas, para esta entrega, se realizó la logica para la creacion de un archivo HTML, según las etiquetas que el usuario ingresa. Esto se logró desde el parser, ya que al ingresar a una produccion, PLY permite la manipulación de lo “matcheado”, por lo tanto, en caso de por ejemplo tener una sentencia `<title>Hola mundo</title>`, es posible obtener el contenido (Hola Mundo) y así asignarle su etiqueta de HTML, según lo requerido por la cátedra. La lista es la siguiente:

- Títulos de tipo Channel: `<h1>`
- Descripción de Channel `<p>`
- Enlace de Channel `<a>`
- Título de ítem `<h3>`
- Descripción de ítem `<p>`

Definición de Tokens

```
# Etiquetas
def t_xml(t): r'<?xml\s+version="1\.\0"\s+encoding="UTF-8"\s*\?>'; return(t)

def t_category(t): r'<category>'; return (t)
def t_cerrarcategory(t): r'<\/category>'; return (t)

def t_description(t): r'<description>'; return (t)
def t_cerrardescription(t): r'<\/description>'; return (t)

def t_link(t): r'<link>'; return (t)
def t_cerrarlink(t): r'<\/link>'; return (t)

def t_titulo(t):
    r'<title>'
    return t;
def t_cerrartitulo(t): r'<\/title>'; return(t)

def t_url(t): r'<url>'; return(t)
def t_cerrarurl(t): r'<\/url>'; return(t)

def t_rss(t): r'<rss\s*version="2.0"\s*>'; return(t)
def t_cerrarrss(t): r'<\/rss>'; return(t)

def t_channel(t): r'<channel>'; return(t)
def t_cerrarchannel(t): r'<\/channel>'; return(t)

def t_item(t): r'<item>'; return(t)
def t_cerraritem(t): r'<\/item>'; return(t)
```

```

# Protocolos
def t_protocolo(t): r'(https|http|ftp|ftps):\//'; return (t)

# Etiquetas opcionales
def t_height(t): r'<height>'; return(t)
def t_cerrarheight(t): r'</height>'; return(t)

def t_width(t): r'<width>'; return(t)
def t_cerrarwidth(t): r'</width>'; return(t)

def t_image(t): r'<image>'; return(t)
def t_cerrarimage(t): r'</image>'; return(t)

def t_copyright(t): r'<copyright>'; return(t)
def t_cerrarcopyright(t): r'</copyright>'; return(t)

# Resto
t_digito = r'\d+'
t_dospuntos = r'\:'
t_slash = r'\/'
t_numeral = r'\#'

# PLY ignorará espacios, saltos de líneas y tabs.
t_ignore = ' \t'

```

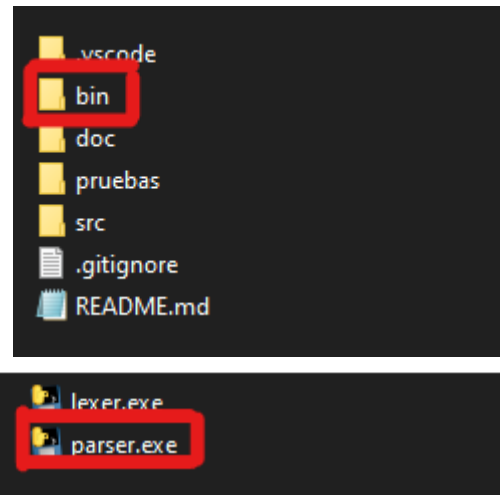
Funciones auxiliares

Las funciones auxiliares a destacar son:

- ``obtenerHtml.py``: se encarga de leer el archivo RSS de entrada y lo traduce a un nuevo archivo html con las etiquetas que se solicitó en la consigna.
- ``logicaMenu.py``: se encarga de mostrar por pantalla un menú de 3 opciones, donde se puede pasar por argumento la función de cada opción, de esta forma, se puede reutilizar el menú visual tanto en el lexer como en el parser.
- ``helpers.py``: se encarga de pedir al usuario la ruta donde se encuentra el archivo .rss a analizar. Luego de que el usuario introduzca la ruta, se remueven comillas del string que puedan impedir que se lea correctamente el archivo.

Modo de obtención del intérprete

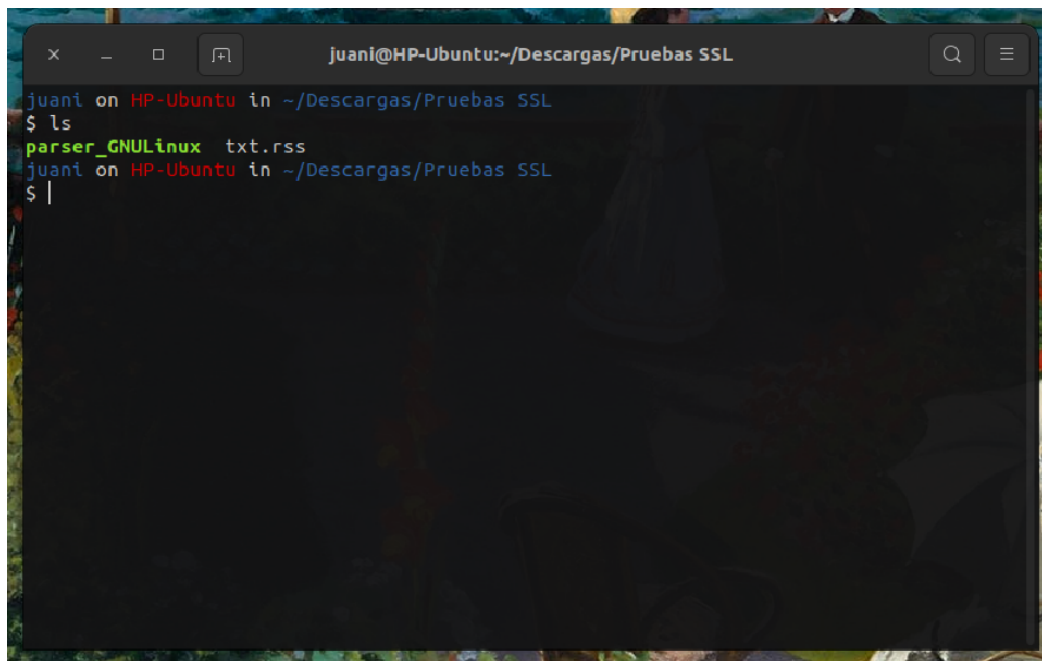
Para poder abrir el intérprete, una vez que estamos en la carpeta principal, entramos a bin y ejecutamos el parser según el SO que se esté utilizando.



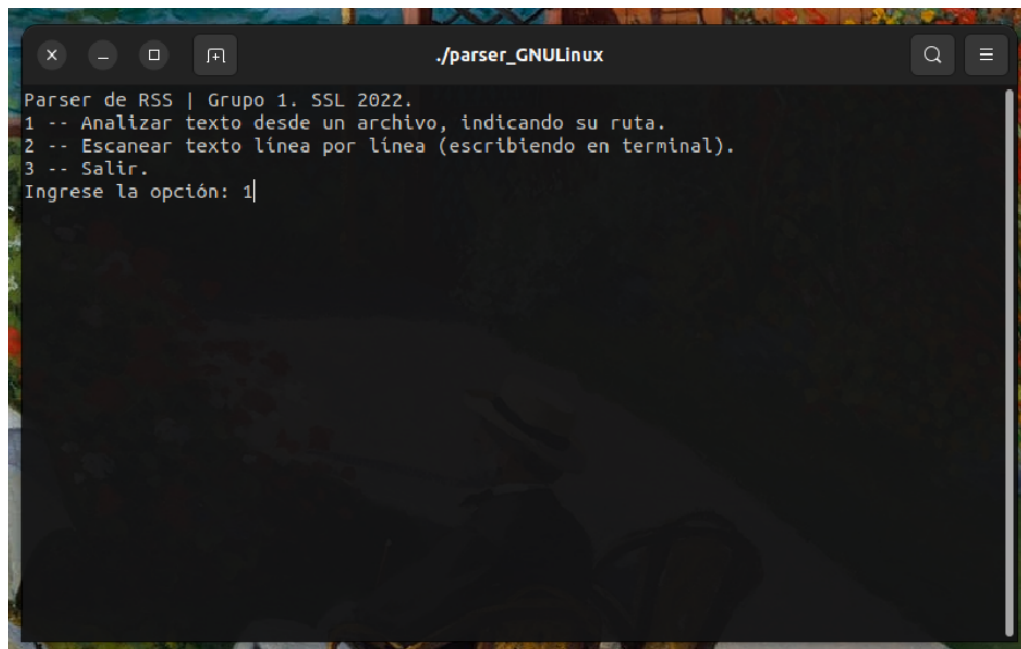
Ejecutar intérprete

- Ir a la carpeta `bin` e iniciar ejecutable según su sistema operativo
- Se mostrará por pantalla un menú con tres opciones,
 1. Elegir un archivo a analizar
 2. Entrar en modo “línea a línea”
 3. Salir

Si el archivo analizado por el parser es correcto sintácticamente, se generará automáticamente un archivo html.



```
juani@HP-Ubuntu:~/Descargas/Pruebas SSL
$ ls
parser_GNULinux.txt.rss
juani@HP-Ubuntu:~/Descargas/Pruebas SSL
$ |
```

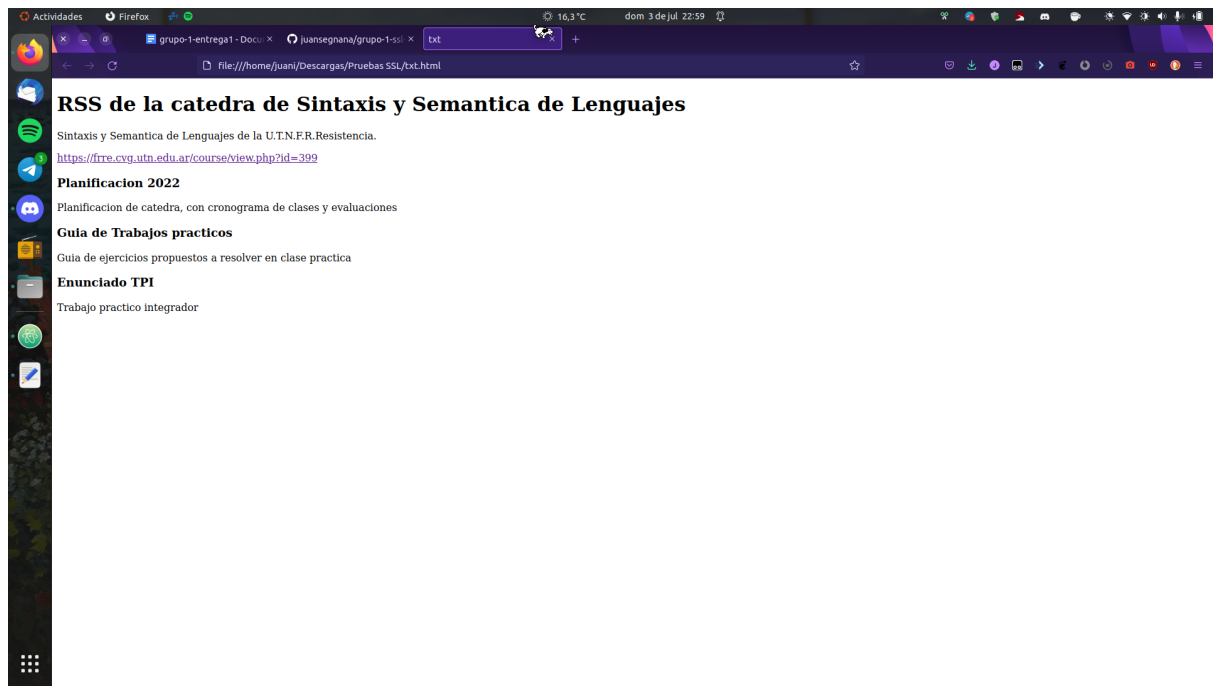


```
./parser_GNULinux
Parser de RSS | Grupo 1. SSL 2022.
1 -- Analizar texto desde un archivo, indicando su ruta.
2 -- Escanear texto línea por línea (escribiendo en terminal).
3 -- Salir.
Ingrese la opción: 1|
```

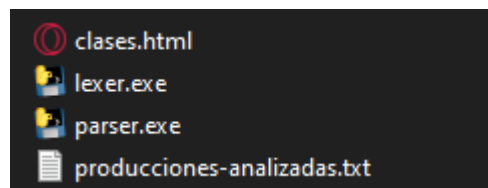
- Previo al análisis del archivo .rss


```
./parser_GNULinux
Ingrese la ruta del archivo a analizar: '/home/juani/Descargas/Pruebas SSL/txt.rss'
Ejecución completa!
✓ El archivo es sintacticamente correcto!
(✓) Sintacticamente correcto. Se exportó un .html con los comentarios.
(!) Se exportó un .txt con las producciones analizadas.
1 -- Analizar texto desde un archivo, indicando su ruta.
2 -- Escanear texto línea por línea (escribiendo en terminal).
3 -- Salir.
Ingrese la opción: |
```

```
juani@HP-Ubuntu:~/Descargas/Pruebas SSL
$ ls
parser_GNULinux      txt.html
producciones-analizadas-2022-07-03T21:50:33.480917.txt  txt.rss
juani on HP-Ubuntu in ~/Descargas/Pruebas SSL
$ |
```



- Después del análisis, con los archivos de producciones analizadas y el .html creados y guardados en la misma carpeta de ejecución.



Ejemplos

Básico con elementos obligatorios

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
  <title>
    RSS de la cátedra de Sintaxis y Semántica de Lenguajes</title>
  <link> https://frre.cvg.utn.edu.ar/course/view.php?id=399 </link>
  <description> Sintaxis y Semántica de Lenguajes de la U.T.N. F.R.Resistencia. </description>
  <item>
    <title>Planificacion 2022</title>
    <link>https://aa</link>
    <description>Planificacion de catedra, con cronograma de clases y evaluaciones
  </description>
  </item>
  <item>
    <title>Guia de Trabajos practicos</title>
    <link>https://frre.cvg.utn.edu.ar/mod/resource/view.php?id=43544</link>
    <description>Guía de ejercicios propuestos a resolver en clase practica</description>
  </item>
  <item>
    <title>Enunciado TPI</title>
    <link>https://wl</link>
    <description>Enunciado del Trabajo práctico integrador</description>
  </item>
</channel>
</rss>
```

Completo con elementos opcionales

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title> RSS de las clases virtuales - Probando html </title>
    <link> https://misclases.live/ </link>
    <description> RSS con las clases virtuales de las materias </description>
    <category> Educacion </category>
    <copyright> 2022 </copyright>
    <image>
      <url>
```

https://www.compartirpalabramaestra.org/sites/default/files/styles/articulos/public/field/image/el-aula-vacia.jpg?itok=BoQ5B1_c

```
    </url>
    <title>Aula</title>
    <link> https://misclases.live/ </link>
    <height> 400 </height>
    <width> 144 </width>
  </image>
  <item>
    <link> https://us04web.zoom.us/j/999999 </link>
    <description> Clase virtual de AM. Teoria </description>
    <title> Analisis Matematico: Teoria </title>
    <category> Clase </category>
  </item>
  <item>
    <title> Analisis Matematico: Practico </title>
    <link> https://us04web.zoom.us/j/999999 </link>
    <description> Clase virtual de AM. Practica. </description>
```

```

        <category> Clase </category>
    </item>
    <item>
        <title> Otro tiem </title>
        <link> https://us04web.zoom.us/j/9999999 </link>
        <description> De prueba </description>
        <category> Clase </category>
    </item>
</channel>
</rss>

```

Conclusión

Uno de los inconvenientes que tuvimos es que el análisis sintáctico se indicaba como incorrecto al tener las etiquetas en la misma línea que su contenido. Luego de investigar en diversos foros, hallamos que la solución más práctica a esto era modificar la expresión regular para el “contenido_texto”, cambiándola a la forma “([\\w\\W])+?(?=<\\V)”. En dicha expresión se utiliza “\\w” para reconocer letras y “\\W” para otros símbolos como puntos o “slashes”. La última parte de la expresión exceptúa el comienzo de la etiqueta de cierre.

También, se realizaron cambios en la gramática debido a ciertos errores que alertamos en etapas más avanzadas del desarrollo del programa.

En las versiones preliminares del programa no contábamos con un menú, lo que contribuía a que el programa sea poco intuitivo en su funcionamiento. Luego de desarrollarlo, se presentaron algunos errores en los avisos acerca de la aceptación o rechazo de los archivos analizados, pero la solución fue relativamente sencilla, ya que se trataba de un contador de errores que estaba posicionado de manera incorrecta en el código.

El programa se desarrolló y probó principalmente en sistemas MacOS y GNU/Linux, por lo que otro de los conflictos que encontramos surgió de manera tardía al crear y ejecutar los archivos binarios de Windows. Una de las funciones encargadas de crear un archivo con las producciones analizadas por el parser arrojaba un error debido a la función de Python llamada “datetime”, la cual tiene como objeto nombrar parte del archivo con la fecha actual del sistema. Una vez removida esta bandera el programa se ejecutó sin problema alguno, pero se perdió esta característica.

Bibliografía

- Librería PLY. <https://ply.readthedocs.io/en/latest/ply.html#lex-example>.
- Regex101: <https://regex101.com>.
- ¿Cómo anidar una expresión regular con una variable?
<https://stackoverflow.com/a/12217856>