Documentación de la Práctica: Analizador de Peticiones y Respuestas HTTP

Rubén Fernández Farelo
ruben.fernandez.farelo@udc.es

Samuel Otero Agraso
s.agraso@udc.es

December 17, 2024

1 Objetivos

El objetivo principal de la práctica es la creación de un analizador de peticiones y respuestas HTTP, capaz de validar tanto sintáctica como léxicamente su correcta estructura. Este analizador procesará mensajes HTTP para verificar que cumplen con las especificaciones estándar del protocolo HTTP, validando:

- Peticiones HTTP (requests).
- Respuestas HTTP (responses).

El analizador es capaz de identificar y reportar errores relacionados con el formato y la coherencia de los mensajes, proporcionando mensajes claros sobre los puntos de fallo.

2 Ficheros que la componen

Esta práctica se ha desarrollado utilizando los siguientes ficheros:

- `proyecto.l`: Contiene el código del analizador léxico. Este archivo define las expresiones regulares necesarias para identificar los tokens que forman parte de una petición o respuesta HTTP. Utiliza la biblioteca de Flex para la generación de un lexer.
- `proyecto.y`: Contiene el código del analizador sintáctico. Este archivo define las reglas de gramática de las peticiones y respuestas HTTP, implementando una serie de acciones para verificar la correcta estructura de los mensajes HTTP. Utiliza la herramienta Bison para la generación del parser.
- `test.sh`: Script de pruebas que ejecuta todos los casos de prueba de forma automática. El script se encarga de analizar los ficheros ubicados en los directorios `tests/pass` y `tests/fail`. Para los ficheros en `tests/pass`, se espera que la estructura del mensaje sea válida, mientras que en los de `tests/fail`, se espera que la estructura sea inválida.

El analizador léxico (definido en `proyecto.l`) recibe un archivo de texto como entrada. Si las secuencias de caracteres coinciden con alguna de las expresiones regulares, los tokens correspondientes se envían al analizador sintáctico (definido en `proyecto.y`) a través de la variable `yylval`.

3 Funcionamiento

El analizador léxico y sintáctico ha sido diseñado para validar el formato de las peticiones y respuestas HTTP. El proceso de análisis se realiza en dos fases:

1. **Análisis léxico**: El analizador léxico (Flex) es responsable de identificar los tokens de un mensaje HTTP, como el método, la versión HTTP, las cabeceras, el cuerpo, etc. Se utilizan expresiones regulares para identificar patrones en el texto de entrada.
2. **Análisis sintáctico**: El analizador sintáctico (Bison) organiza los tokens en una estructura jerárquica según las reglas de la gramática de HTTP. Verifica que el orden de los tokens sea el adecuado, y valida que las cabeceras y el cuerpo del mensaje estén bien formados.

3.1 Comandos para compilar y ejecutar

- `make all`: Para compilar el proyecto.
- `make run`: Para realizar pruebas manuales con un archivo de prueba.
- `make run2`: Si se desea cambiar el archivo de prueba, se debe modificar la variable `PRUEBA` en el archivo `Makefile`.
- `make test`: Ejecuta todos los tests automáticamente, aplicando el analizador a los ficheros en los directorios `tests/pass` y `tests/fail`.

4 Tratamiento de errores

Cuando se detecta un error en el formato de una petición o respuesta HTTP, el analizador proporciona detalles sobre el fallo, incluyendo la línea aproximada donde ocurrió el error. Algunos ejemplos de errores que se pueden detectar son:

- **Líneas de inicio mal formadas**: La línea de inicio no comienza con un método HTTP o una versión del protocolo.
- **URI incorrecta**: La URI en una petición no sigue el formato esperado.
- **Falta de versión HTTP**: En las peticiones, la versión del protocolo HTTP no está presente.
- **Código de estado incorrecto**: En las respuestas, el código de estado no es uno de los valores válidos.
- **Cabeceras mal formadas**: Las cabeceras no siguen el formato esperado, como un valor incorrecto o un campo desconocido.

El analizador ofrece mensajes claros, como por ejemplo:

- `Request [INVALID]`
- `Response header [INVALID]`

Además, en el caso de los errores relacionados con el cuerpo del mensaje, si el cuerpo es de tipo HTML o JSON, el analizador verifica la correcta apertura y cierre de las etiquetas HTML o los objetos JSON.

5 Peculiaridades

- **Verificación de etiquetas HTML y JSON**: El analizador también verifica si el cuerpo del mensaje (en el caso de respuestas o peticiones que incluyan un cuerpo) está correctamente formado, ya sea como HTML o JSON. Para el HTML, el analizador valida las etiquetas de apertura y cierre utilizando una pila de tags. Para el JSON, valida que las llaves {} y los corchetes `[]` estén balanceados.
- **Soporte para múltiples métodos HTTP**: El analizador es compatible con los siguientes métodos HTTP: `GET`, `POST`, `PUT`, `DELETE`, `HEAD`, `OPTIONS`, `TRACE`, y `CONNECT`.
- **Soporte para diferentes versiones de HTTP**: Se soportan las versiones HTTP/0.9, HTTP/1.0, HTTP/1.1, HTTP/2 y HTTP/2.0.
- **Detección de errores en cabeceras comunes**: El analizador verifica la presencia y el formato de las cabeceras estándar como `Host`, `Cache-Control`, `Accept`, `Content-Type`, `Content-Length`, entre otras.

6 Estructura del código

6.1 Archivo `proyecto.l` (Flex)

Este archivo contiene las definiciones de las expresiones regulares que definen los tokens de un mensaje HTTP. Algunas de las expresiones importantes son:

- **Métodos HTTP**: `GET`, `POST`, `PUT`, `DELETE`, `HEAD`, `OPTIONS`, `TRACE`, `CONNECT`.
- **Versión HTTP**: Define las versiones de HTTP que se pueden encontrar en las peticiones.
- **URI**: La estructura de la URI que puede aparecer en las peticiones.
- **Códigos de estado HTTP**: Como `200 OK`, `404 Not Found`, etc.

Las reglas de Flex se encargan de hacer coincidir estas expresiones regulares en el texto de entrada y devolver los tokens correspondientes.

6.2 Archivo `proyecto.y` (Bison)

Este archivo define la gramática para las peticiones y respuestas HTTP, detallando las reglas sintácticas para la estructura completa de un mensaje HTTP. Las principales reglas incluyen:

- **Request line**: Validación de la línea de solicitud, como el método, URI y versión HTTP.

- **Response line**: Validación de la línea de respuesta, como el código de estado y la frase.
- **Headers**: Validación de las cabeceras HTTP y sus valores.
- **Body**: Procesamiento del cuerpo de la petición o respuesta, validando su formato como HTML o JSON.