INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

Práctica WGET

Profesor: Axel Ernesto Moreno Cervantes

Alumnos:

Jiménez Rodríguez Alejandro Martin

Núñez Ramírez Valery Aylin

Grupo: 6CM1

Introducción

El comando **wget** es una herramienta ampliamente utilizada en sistemas operativos para la descarga de recursos desde la red, como páginas web, imágenes, documentos y otros archivos, utilizando protocolos como HTTP. Su funcionamiento se basa en la comunicación cliente–servidor, donde el cliente realiza una petición HTTP y el servidor responde con el recurso solicitado.

El objetivo de esta práctica es comprender el funcionamiento interno de un cliente HTTP implementando un **Mini Wget** en el lenguaje de programación **Java**, utilizando **sockets bloqueantes**. A través de esta implementación se busca reforzar conceptos fundamentales como el uso de sockets, el manejo del protocolo HTTP, la interpretación de encabezados (headers), la detección del tipo de contenido (MIME) y la descarga correcta de archivos binarios y documentos HTML.

Además, se implementa un mecanismo de **descarga recursiva**, que permite analizar documentos HTML y descargar automáticamente los recursos adicionales que estos referencian, evitando procesar una misma URL más de una vez.

Desarrollo de la práctica

Recepción y gestión de URLs

El programa recibe como argumento una URL inicial desde la línea de comandos. Para manejar la descarga recursiva, se utilizan dos estructuras de datos principales:

- Un **Map<String, Boolean>** para almacenar las URLs ya visitadas y evitar ciclos infinitos.
- Una **cola (Queue<String>)** para gestionar las URLs pendientes de procesar.

Este enfoque permite recorrer los recursos de forma controlada y eficiente.

Creación de sockets bloqueantes

Para la comunicación con el servidor, se utiliza la clase **Socket**, configurada para trabajar de manera bloqueante. Esto implica que las operaciones de lectura y escritura se detienen hasta que haya datos disponibles, lo cual simplifica el manejo del flujo de información para esta práctica.

Se obtiene:

- El **host**
- El **puerto** (por defecto 80 para HTTP)
- El **recurso solicitado**

a partir de la URL utilizando la clase URI.

Envío de la petición HTTP

Se construye manualmente una petición HTTP usando el protocolo **HTTP/1.0**, lo cual evita el uso del mecanismo Transfer-Encoding: chunked, simplificando la lectura de datos binarios.

Ejemplo de la petición enviada:

GET /ruta HTTP/1.0

Host: servidor

User-Agent: MiniWget/1.0

El uso del encabezado **User-Agent** permite que el servidor identifique al cliente y evita errores comunes como respuestas 403 (Forbidden).

Lectura y procesamiento de headers

La respuesta del servidor se divide en dos partes:

- **Encabezados (headers)**
- **Cuerpo (body)**

Los headers se leen byte a byte hasta detectar la secuencia \r\n\r\n, que indica el final de los encabezados. A partir de ellos se extrae información clave como:

- Código de estado HTTP (200, 301, 403, etc.)
- Tipo de contenido (Content-Type)

Si el código HTTP no es **200 (OK)**, la descarga se cancela.

Detección del tipo de contenido (MIME)

El encabezado **Content-Type** permite identificar si el recurso es:

- **HTML (text/html)**
- **Archivo binario** (imágenes, PDF, ZIP, etc.)

Esta decisión es fundamental para determinar cómo se procesará la respuesta.

Procesamiento de documentos HTML

Cuando el contenido es HTML:

- Se lee el cuerpo como texto.
- Se analiza usando expresiones regulares para encontrar enlaces en atributos href y src.
- Los enlaces relativos se convierten en URLs absolutas utilizando el método resolve de la clase URI.

Las nuevas URLs encontradas se agregan a la cola siempre que no hayan sido visitadas previamente, permitiendo así la **descarga recursiva**.

Descarga de archivos binarios

Si el recurso no es HTML:

- Se lee directamente el flujo de entrada (InputStream) en modo binario.
- El contenido se guarda en un archivo local con el mismo nombre que el recurso solicitado.

Este proceso garantiza que archivos como imágenes o documentos se descarguen sin corrupción.

Prevención de ciclos infinitos

Para evitar que el programa entre en ciclos, se utiliza la tabla de URLs visitadas. Antes de procesar cualquier enlace, se verifica si ya fue descargado anteriormente. Esto asegura que cada recurso se procese una sola vez.

Conclusiones

Nunez Ramirez Valery Aylin

Durante la realización de esta práctica pude entender mejor cómo funciona el protocolo HTTP a un nivel más bajo del que normalmente se ve al usar un navegador o herramientas ya hechas. Al implementar el Mini Wget utilizando sockets bloqueantes, aprendí a construir manualmente una petición HTTP y a interpretar la respuesta del servidor, especialmente los encabezados y el tipo de contenido. También me ayudó a identificar errores comunes, como la corrupción de archivos al leer incorrectamente los datos. En general, considero que esta práctica fue muy útil para reforzar mis conocimientos sobre redes y la comunicación cliente–servidor.

Jimenez Rodriguez Alejandro Martin

Al principio esta práctica me resultó complicada, ya que nunca había trabajado directamente con sockets ni con el manejo manual de peticiones HTTP. Sin embargo, conforme fui avanzando, comprendí cómo una página web puede contener múltiples recursos y cómo estos pueden descargarse de forma automática. La implementación de la descarga recursiva y el control de URLs visitadas me permitió entender la importancia de evitar ciclos y descargas repetidas. Esta práctica me ayudó a mejorar mi lógica de programación y a tener una visión más clara de cómo funcionan internamente herramientas como wget.