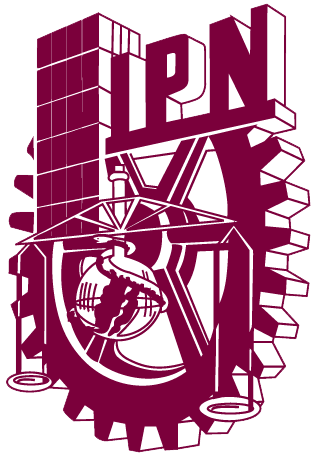
**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

**Aplicaciones para Comunicaciones de Red**

**Práctica 5**

**Wget**

**Miembros:**

Benítez Ramírez Sergio

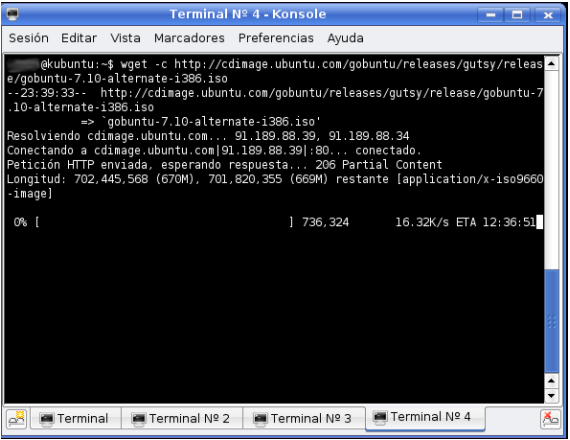
Monteros Cervantes Miguel Angel



3CM15

Introducción

GNU Wget es una herramienta libre que permite la descarga de contenidos desde servidores web de una forma simple. Su nombre deriva de World Wide Web (w), y de «obtener» (en inglés get), esto quiere decir: obtener desde la WWW. Fue escrito originalmente por Hrvoje Nikšić y por ser un proyecto de software libre tiene una gran cantidad de colaboradores directos e indirectos. Actualmente admite descargas mediante los protocolos HTTP, HTTPS y FTP. Entre las características más destacadas que ofrece Wget está la posibilidad de fácil descarga de mirrors (repositorios) complejos de forma recursiva, conversión de enlaces para la visualización de contenidos HTML localmente, soporte para proxies, etc. Su primera versión se lanzó en 1996, coincidiendo con el boom de popularidad de la web. Es un programa utilizado a través de línea de comandos, principalmente en sistemas tipo UNIX, especialmente en GNU/Linux. Escrito en el lenguaje de programación C, Wget puede ser fácilmente instalado en sistemas derivados de UNIX, y ha sido portado a muchas interfaces gráficas de usuario (GUI) y aplicaciones gráficas de descarga como Gwget2​ para GNOME, wGetGUI3​ y VisualWget4​ para Microsoft Windows, Wget 1.10.2r25​ para Mac OS X.



**Hilo**

Un hilo (en inglés “thread”) es la menor de las estructuras lógicas de programación que se ejecuta de forma secuencial por parte del planificador del sistema operativo (nota: forma secuencial no quiere decir que no haya bucles, solo que es una línea secuencial de ejecución: una sentencia después de otra).

Los hilos son más “ligeros” que sus hermanos mayores (los procesos) ya que muchos de los recursos que es necesario reservar para lanzar y ejecutar un proceso, son compartidos entre distintos hilos dentro de un mismo proceso.

Los hilos existían como recurso de programación desde mucho tiempo antes de que empezaran a hacerse comunes los procesadores de varios núcleos y otras estructuras de paralelización habituales en las arquitecturas de hoy día (pipelines, grids, GPUs, etc.). Pero no nacieron como forma de acelerar la ejecución de los programas, sino como forma de “simplificar” la forma en la que se diseñan los programas más complejos.

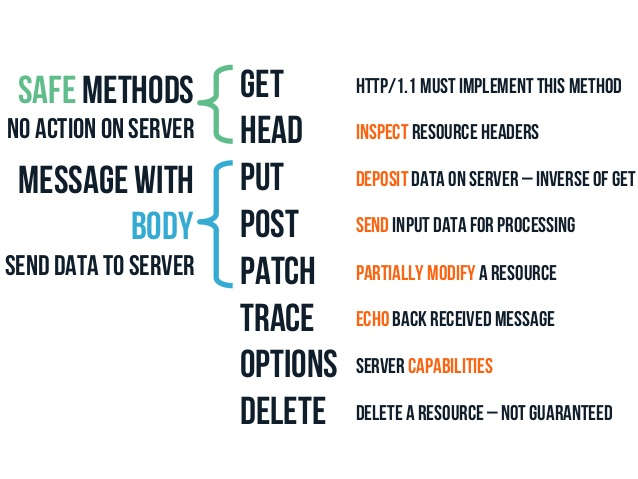
**Peticiones HTTP**

HTTP contiene un grupo de peticiones HTTP (también llamadas HTTP verbs por, el tipo de nombre que manejan casi todos ellos -pues si bien algunos son sustantivos, la gran mayoría no-) que nos ayudan a especificar la acción que se requiere realizar en un elemento determinado y aunque estas peticiones tienen distintas semánticas, también tienen muchas similitudes en las mismas que evitan que este grupo se extienda demasiado.

**Clasificación de las peticiones HTTP**

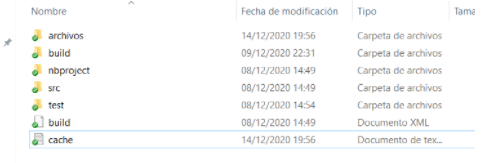
Estas peticiones las podemos clasificar en Safe e Idempotent:

* Peticiones HTTP Safe
* Peticiones HTTP Idempotent

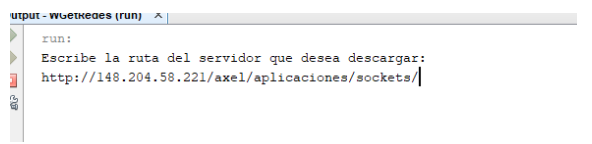


Desarrollo

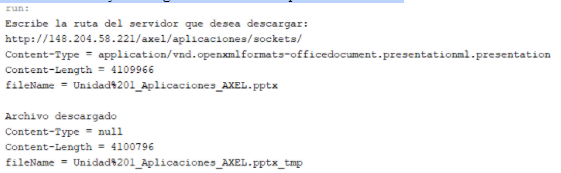
Tenemos una carpeta llamada “archivos”, que nos permitirá alojar todas las descargar que se hagan a cualquier servidor que brinde archivos. El archivo cache.txt, permitirá alojar todos los nombres de los archivos descargados, para evitar que se vuelvan a descargar nuevamente



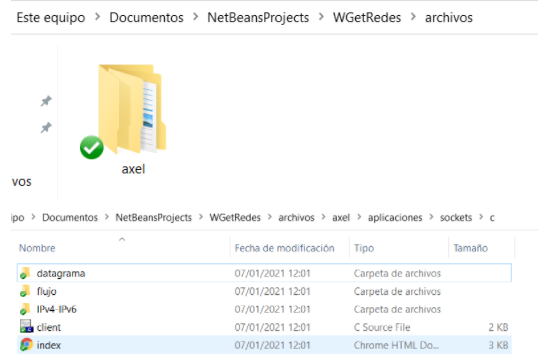
Primeramente, le pedimos la ruta del servidor que desea descargar sus archivos y carpetas.



Una vez indicada la ruta, nuestro cliente comenzará a mapear todos las carpetas y archivos que ofrece esta dirección descargando los archivos, carpetas y subcarpetas de manera recursiva y las irá guardando en la carpeta “archivos”.

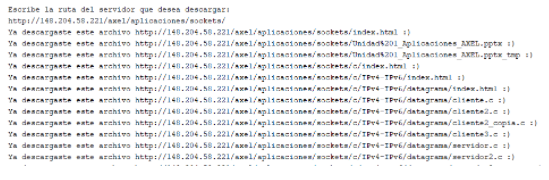


Podemos observar que efectivamente, ha descargado todos los archivo de la ruta indicada en nuestra carpeta “archivos” y construye todo la ruta:





En nuestro caché, ha guardado todas las rutas que se han descargado en nuestra carpeta para que la próxima vez que se vuelva a querer descargar cualquiera de estos recursos, se avise que ya está descargado y evitar que se vuelva a descargar

Si volvemos a correr el proyecto y volvemos a proporcionar la misma ruta, se revisará en el caché y se avisará en caso de que ya esté descargado: 

**Conclusiones**

En esta práctica, pudimos entender un poco como funciona de manera profunda los programas que incluyen sistemas de descargas dada una ruta. La implementación de un hilo para las descargas y la conexión mediante URL Connection, fueron algunos tópicos que fueron utilizados para lograr realizar una conexión con el servidor.

Fue complicado en un principio lograr descargar los archivos que estaban enlistados en la página de los servidores, porque era necesario realizar un manejo de strings de forma profunda para obtener si lo que ofrecía era un archivo o una carpeta, o en su defecto algún otro link de escape. De esta manera, de manera similar a la práctica del Drive, pudimos mapear todos los recursos que ofrecía el servidor de manera recursiva y poder ir descargando y creando las carpetas con mkDir() el árbol que se iba generando.

**Referencias**

<https://es.wikipedia.org/wiki/GNU_Wget#/media/Archivo:Konsole_1.6.6-wget_1.10.2-es.png>

<https://www.cs.buap.mx/~iolmos/redes/3_Hilos.pdf>

<https://es.quora.com/Qu%C3%A9-es-el-hilo-en-t%C3%A9rminos-de-programaci%C3%B3n>

<https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/http-request/>