Dumb DNS

The project contains a basic DNS proxy, with caching, blocking and filtering capabilities, made on the context of CC4303 Course "Redes" (Networks) at the University of Chile. We are based on examples given on the course and on RFC 1035 from IETF.

Basic usage

```
dumbdns.py [-h] [-P PORT] [-R RESOLVER] [-C CACHE] [-F FORWARD]
               [-B BLOCKED]
```

a fucking damn dns that receives the following parameters:

```
optional arguments:
  -h, --help            show this help message and exit
  -P PORT, --port PORT  Port used for the request. Default port is 1025
  -R RESOLVER, --resolver RESOLVER
                        Resolver to get an actual answer
  -C CACHE, --cache CACHE
                        Cache Timeout in [s], default value is 3600.
  -F FORWARD, --forward FORWARD
                        JSON containing names/domains used to be forwarded to
                        another IP with format {"page": "ip"}
  -B BLOCKED, --blocked BLOCKED
                        JSON containing names/domains that will not receive a
                        response with format {"names": [page1, page2, ...]}
```

all this are optionals, but by default comes with `-P 1025 -R 1.1.1.1 -C 3600`, forward and blocked JSONs are not requiered to run this proxy.

Explanation and assumptions (from spanish subtitles)

En la tarea el servidor atiende todas las consultas en el mismo thread, y atiende preguntas en nombre de un resolvera elección seteado al momento de correr el servidor.

Al recibir una consulta se realiza un parseo de esta, extrayendo información relevante de la misma, como la dirección a la que se realiza la consulta y el tipo de consulta.

Una vez parseada la consulta se revisa si la respuesta está en caché, de manera que si está y aún no ha pasado el timeout de la última respuesta guardada, se responde la misma (cambiando el ID de la respuesta cacheada por el ID de la request actual). Si la respuesta no está guardada se procede a buscar el dominio en la lista de bloqueados, ignorando la request si está contenido en ella, sino, procede a realizar la consulta.

Una vez obtenida una respuesta, se procede al parsing de la misma guardando la IP de respuesta. Si el dominio se encuentra en la lista de forwarding se cambia la IP de la respuesta por la IP otorgada por el usuario del script, guardándola en caché y respondiendo a la request.

Si no estaba en la lista de forwarding, simplemente se guarda en caché y se responde a la request.

Caché

Una vez parseados los elementos de la consulta se revisa si existe alguna respuesta a la consulta en caché, el que por simplicidad es guardado en un archivo `cache.json` que por cada nueva URL consultada guarda un string representando la IP de la respuesta, un string representando el tipo de la pregunta, un binario de un objeto de la clase `datetime` representando el momento de realización de la consulta y finalmente otro binario conteniendo la request completa. La actualización del caché se realiza de manera lazy, por lo que solo se actualizará una respuesta del mismo cuando se realice una nueva consulta y haya pasado el timeout otorgado por el usuario (si no se consulta, y pasó el timeout no se actualiza). Se dió un timeout de 3600 segundos, pudiendo el usuario cambiar dicho valor al momento de lanzar el servidor utilizando la flag `-C` entregando un entero representando los segundos de duración de cada respuesta a algún dominio en caché.

Dominios bloqueados

El usuario puede otorgar al programa utilizando la flag `-B` un string conteniendo el path a algún archivo JSON conteniendo los dominios bloqueados por el servidor, de manera que todas las request realizadas a alguno de los dominios contenidos en aquel archivo serán ignoradas por el servidor (no serán respondidas). A continuación se presenta un ejemplo de JSON que hace que las consultas a facebook.com, google.com y bancodechile.cl sean ignoradas:

blocked.js

```
{
  "names": ["facebook.com", "google.com", "bancodechile.cl"]
}
```

El servidor no verifica que el formato del JSON sea correcto, por lo que es responsabilidad del usuario entregar un archivo bien formateado, de otro modo, el servidor se caerá.

Redireccionamiento

El usuario puede otorgar al programa utilizando la flag `-F` un string conteniendo el path a algun archivo JSON conteniendo los dominios a redireccionar por el servidor, por cada dominio se debe otorgar una IP a la que se redireccionará. A continuación se presenta un ejemlo de JSON que redirecciona las consultas de facebook.com a la IP de google.com y las consultas de google.com a facebook.com:

forward.js

```
{
  "facebook.com": "64.68.90.1",
  "google.com": "69.63.176.13"
}
```

El servidor no verifica que el formato del JSON sea correcto, por lo que es responsabilidad del usuario entregar un archivo bien formateado, de otro modo, el servidor se caerá.

Otras Asunciones

- El resolver devuelve en el caso de los sitios redireccionados solo un sitio a modo de respuesta
- El programa tiene permisos de escritura en la carpeta en la que corre
- Se está conectado a internet (no probamos el caso en que no lo estuviera)