



PROGRAMA COMANDOS PERSONALIZADOS PARA SISTEMA OPERATIVO

AA3. Automatización básica de la administración de un Sistema Operativo

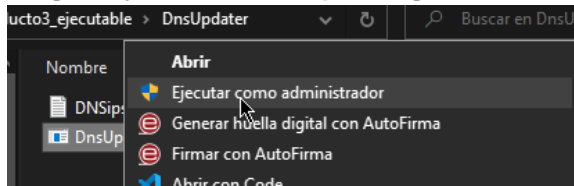
DESCRIPCIÓN DEL PROGRAMA	2
Análisis de los requisitos del programa	2
Modularización del programa	4
Lista de módulos	4
Solución de los requisitos de cada módulo	5
1. Módulo data.h	6
2. Módulo api	6
3. Módulo dns	6
4. Módulo menu	8
5. Módulo principal	8
COMPILACIÓN Y EJECUCIÓN	9
Compilación sin errores	9
Ejecución	9
1. Datos de sistema previos a la ejecución:	9
2. Carga de datos y menú principal	10
3. Imposibilidad de realizar la actualización sin haber escogido un adaptador previamente.	10
4. Selección de adaptador.	11
5. Actualización de DNS.	11
ARCHIVOS ADJUNTOS	13
ANEXOS	13
I. Soluciones para la ejecución de un comando con privilegios elevados.	13
RECURSOS	14

AA3. Automatización básica de la administración de un Sistema Operativo.

Descripción del programa

¡MUY IMPORTANTE!

El programa debe ejecutarse como administrador, ya que los comandos *set* de *netsh* exigen ejecutarse con privilegios elevados.



Análisis de los requisitos del programa

La identificación de los requisitos va a ser de gran ayuda en el proceso de análisis descendente del programa, la clasificación de los diferentes módulos necesarios y el diseño de estos. Para ello, la ordenación en forma de árbol y ramas permite identificar rápidamente los módulos principales y los requisitos internos de cada módulo para encontrar qué funciones debe implementar.

Tras la lectura del enunciado de la actividad, podemos identificar los siguientes requisitos para el programa:

1. **Carga de datos iniciales:** El programa debe cargar los datos de un archivo externo .txt con las direcciones DNS, la ruta y nombre del cual se preguntan al usuario.
 - a. El programa debe obtener los datos mediante un menú contextual y generar la ruta al archivo.
 - b. Abrirá el archivo, leerá los datos y cerrará el archivo.
 - c. El contenido del archivo se mostrará por pantalla.
 - d. Los datos quedarán cargados en el programa.
2. **Selección de un adaptador de red:**
 - a. El programa debe disponer de los datos necesarios:
 - i. En este caso, he creído que lo mejor es que solo muestre los adaptadores activos para no realizar la prueba sobre adaptadores no conectados o sobre el *Loopback*.
 - ii. Para facilitar el proceso, en lugar de capturar el nombre del adaptador, trabajaremos a través del índice del adaptador.
 - b. La selección del adaptador se realizará mediante un menú contextual con el usuario, deberá simplificarse para evitar posibles errores.
 - c. Una vez escogido, debemos ser capaces de poder recuperar el índice del adaptador en cualquier momento.

3. Actualización al mejor servidor DNS:

- a. El programa no debe iniciar este paso a no ser que haya sido escogido el adaptador de red.
- b. Crearemos **un archivo temporal** con direcciones DNS:
 - i. Escribiremos las direcciones DNS, si las hay, configuradas para el adaptador escogido por el usuario.
 - ii. Solo se escribirán aquellas direcciones que pasen una prueba de accesibilidad.
- c. Obtenemos una lista con las direcciones DNS del archivo inicial y las direcciones accesibles del adaptador. Esto simplifica el paso de la comprobación.
- d. Realizamos un test de velocidad sobre todas las direcciones DNS obtenidas.
 - i. Registramos el tiempo de respuesta medio y el número total de saltos para cada dirección.
- e. Discriminamos cuál es la dirección más rápida según el siguiente orden de criterios:
 - i. Menor tiempo de respuesta total medio.
 - ii. Si este es coincidente, menor número de saltos.
- f. Cambiamos la configuración del SO y actualizamos al mejor servidor DNS.
 - i. Si ya está escogido, no se realiza ninguna operación.
 - ii. Si no, se cambia la configuración.

Estos son los requisitos principales que se extraen del enunciado de la actividad, además, he decidido incluir los siguientes aspectos para intentar optimizar mi programa dentro de lo que se he sido capaz de discernir:

4. La interfaz del programa se trata de un menú contextual por línea de comandos.

5. El programa usará estructuras de datos TAD y tablas: De este modo simplifica el manejo de datos posterior a su obtención y toda la información se encuentra ordenada. Además, todas estas estructuras deberán poderse limpiar o liberar una vez ya no necesiten usarse.

- a. Creación de las estructuras:
 - i. Estructura TAD para almacenar los datos de un adaptador de red.
 - ii. Tabla de estructuras TAD de adaptador.
 - iii. Tabla de direcciones DNS.
 - iv. Estructura TAD para almacenar los resultados del test de velocidad a un servidor.
 - v. Tabla de estructuras TAD de test de velocidad.
- b. Inicialización de las estructuras.
- c. Funciones necesarias para:
 - i. Añadir elementos a las estructuras.
 - ii. Búsqueda de un adaptador de red en concreto.
 - iii. Mostrar información de las estructuras.
- d. Eliminación de las estructuras.

6. Todos los archivos intermedios que genere el programa serán de carácter temporal.

Durante el desarrollo del producto, encontrado dos formas de gestionar esto:

- a. Mediante la gestión manual de creación y eliminación de archivos.
 - b. Mediante el uso de la función *tempfile()* que genera un archivo temporal que se elimina al liberar el buffer con *fclose()* o al cerrar el programa.
7. **El programa usa una API interna:** Que gestionará la comunicación interna entre el módulo del menú y el módulo de operaciones, la carga de los datos iniciales y la inicialización de estructuras. **Este punto absorbe al requisito 1.**
8. **Uso de un algoritmo Quicksort para ordenar los resultados del test de velocidad:** De esta forma, la posición 0 de la tabla de resultados siempre será la del mejor adaptador. **Esto complementa el requisito 3.e.**
9. **Gestión del retorno en el test de velocidad:** En la traza a una dirección DNS podemos obtener como resultado de un salto *tiempo de espera agotado*, que marca con el carácter '*' los tiempos de respuesta. Esto no significa que el salto no sea accesible, sino que ese servidor en específico bloquea el paquete de datos, normalmente por el *firewall*. Ignorar esta respuesta no es correcto porque enturbiaría el cálculo del tiempo de respuesta medio. **Esto complementa el requisito 3.d.**
10. **Gestión de la memoria dinámica:** Al usar apuntadores, deberemos asegurarnos de gestionar el acceso a memoria de forma eficiente, asignado y liberando memoria según sea necesario.
- a. Este requisito genera necesidades sobre el resto de los puntos.

A través del análisis de los requisitos, podemos definir el problema en bruto. Durante el proceso de análisis descendente, iremos subdividiendo cada subproblema en subproblemas menores para encontrar que funciones debemos implementar en nuestro programa:

- **Problema general:** Actualizar la dirección DNS del adaptador de red de un equipo a la más rápida de una lista de posibles candidatas.
- **Subproblema:** Obtener la lista de DNS candidatas.
- **Subproblema:** Obtener la información sobre los adaptadores de red del equipo.
- **Subproblema:** Obtener las DNS de un adaptador de red activo escogido por el usuario.
- **Subproblema:** Decidir cuál es el mejor servidor DNS.
- **Subproblema:** Cambiar la dirección DNS del adaptador si fuera necesario.

Modularización del programa

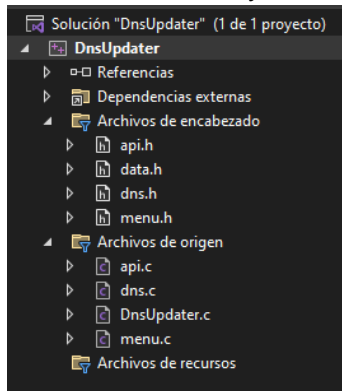
Para poder usar bien los módulos, hay que crear un módulo con los archivos de cabecera, *nombreModulo.h* que contendrá las constantes, estructuras y los prototipos de todas las funciones, y un archivo *nombreModulo.c* donde se implementen todas las funciones. Además, hay que enlazar las librerías de cabecera siempre que se vayan a usar en cualquier otro módulo, para ello usamos la sintaxis *#include "modulo.h"* que indica al compilador donde ir a buscar las cabeceras.

Lista de módulos

Mediante el análisis de requisitos hallamos las funciones principales que debemos implementar y, de esta forma, los módulos del programa, el primer paso en el análisis descendente del

problema. A través de la profundización en los subrequisitos y esta primera asociación, podremos ir encontrando todas las funciones que deberá implementar cada módulo que deberán ir solucionando los problemas particulares. La unión de los módulos deberá solucionar el problema principal.

1. **Módulo api** → Soluciona: Realizar la carga de datos iniciales, inicializar las estructuras y manejar la comunicación entre los módulos y la obtención de la ruta del archivo inicial.
2. **Módulo dns** → Soluciona: Seleccionar un adaptador de red y actualizar su dirección DNS a la óptima de una lista de direcciones candidatas.
3. **Módulo menú** → Soluciona: La interfaz del programa se trata de un menú contextual por línea de comandos.
4. **Módulos data.h** → Extra: Archivo de cabeceras que aísla la declaración de las constantes y de los tipos de estructuras. No es necesario crear un .c.
5. **Módulo principal con el *main*: DnsUpdater** → Módulo para inicializar y controlar el flujo del programa y el valor de retorno. En este producto he optado por intentar mejorar el control y manejo de errores llevando una gestión del posible valor de retorno.



Solución de los requisitos de cada módulo

En general, en todos los módulos he intentado trabajar siguiendo las medidas de seguridad necesarias para **gestionar la integridad de los datos**; así, aunque no se describa en cada punto en particular, se han usado bloques condicionales para controlar que no haya punteros NULL, se han usado funciones de librerías estándar más seguras, por ejemplo, `strncpy()` en lugar de `strcpy()`, o, en la declaración de constantes, se ha usado el bloque `#ifndef` para evitar repeticiones en la definición de estas y, en la creación de cabeceras, la cláusula `#pragma once` para asegurarnos de que una librería solo se carga una vez en la compilación. También, como he comentado, he intentado mejorar la gestión de errores y, no solo mostrar por pantalla un mensaje de error, sino **actualizar el posible valor de retorno final del programa**.

En cuanto a las constantes, su uso intenta facilitar el cambio de cualquier valor literal en el programa sin tener que revisar todo el código línea a línea. Entre otras se indican: Rutas de los archivos log, tamaños máximos de las cadenas de caracteres y opciones de apertura de los archivos.

1. Módulo data.h

Para facilitar el trabajo con los datos que vamos obteniendo durante la ejecución del programa, tras su lectura son almacenados en diferentes tipos de estructuras.

Lista de estructuras: `typedef struct tAdapter {} adapter; typedef struct tAdptsTable {} adptsTable; typedef struct tDnsTable {} dnsTable; typedef struct tDnsTest {} dnsTest; typedef struct tDnsTestTable {} dnsTestTable;`

2. Módulo api

Gestiona la inicialización de las diferentes estructuras asignando la memoria dinámica en caso de ser necesario y valores por defecto a los campos de las estructuras para poder gestionar errores y, cuando sea necesario, realizar de forma correcta los cálculos para la asignación de los valores reales. Por otro lado, va realizando la delegación necesaria a funciones del módulo dns para la carga de datos y la comunicación con el módulo menú, es decir, obtiene los datos necesarios mediante las funcionalidades de dns y los envía a donde sean requeridos

Lista de funciones: `void initAdptr(adapter* adptr); void initAdptrTable(adptsTable* table); void initDnsTest(dnsTest* src); void initDnsTable(dnsTable* table); void initDnsTestTable(dnsTestTable* table); void loadData(adptsTable* adptsTab, dnsTable* dnsTab, int* retVal); char* askPath(); void showDnsIps(const adptsTable table, const int adptrNum, dnsTable* adptrDnsSrvs, int* adptrInd, int* retVal); void getDnsTests(const adptsTable adptsTbl, const dnsTable dnsDirs, const dnsTable adptrDnsSrvs, const int adptrInd, int* retVal); void eraseLogs();`

3. Módulo dns

Este módulo se encarga del trabajo y la gestión de datos. Es el módulo más grande y complejo, por lo tanto, he intentado guiarme a través de la creación de funciones principales y, a continuación, de todas las funciones auxiliares necesarias para implementar las principales. Como en todos los casos, gestiona la memoria dinámica, el control y manejo de errores y la eliminación de estructuras que ya no son necesarias.

Me gustaría comentar algunos aspectos importantes. En cuanto a la gestión de archivos temporales, al principio se usa una **gestión manual**, los dos archivos log para los adaptadores y los DNS del adaptador seleccionado se crean mediante una función que lanza un `system()` con una redirección al archivo log que lo genera, la eliminación se realiza al finalizar las operaciones, antes de cerrar el programa, mediante la función `remove()`. Pero informándome, encontré que existe una **gestión automatizada**; así, para el **requisito obligatorio de generar un archivo temporal con las direcciones DNS accesibles del adaptador de red seleccionado** he usado la función `tempfile()`, esta función genera un archivo temporal binario, con lo que es mucho más ligero, y lo abre en modo de escritura. El archivo se elimina al liberar el buffer con un `fclose()` o al cerrar el programa.

En lo que corresponde a la lectura del retorno de un comando CMD, también he usado dos aproximaciones. La primera, mediante la lectura de los logs temporales comentados; la segunda, tras ampliar mi búsqueda de información sobre posibles soluciones, mediante el uso de un *pipe* a un puntero a FILE. De este modo, puedo trabajar como si estuviera leyendo el archivo log, pero sin necesidad de generar el archivo, así, ahorro uso de memoria por parte del programa. Para ello, se hace uso de las funciones `_popen()` y `_pclose()`.

También, me gustaría destacar que realizo **el manejo de palabras clave en español** en la obtención de las direcciones DNS de un adaptador y las pruebas de accesibilidad. Esto conlleva que debería buscar la forma de adaptar el programa para que funcione en otros idiomas. El resto de caracteres clave funcionarían en cualquier idioma y, para otros casos, he buscado soluciones mediante el uso de `strlen()` o contadores de líneas leídas.

En cuanto a la gestión del retorno tipo *tiempo de espera agotado* en los tests de velocidad, no se pueden ignorar ya que sería un problema en el cálculo del tiempo de respuesta medio. La solución que he ideado es, por un lado, lanzar el comando *tracert* con la opción `-w 1000`, así me aseguro de que el tiempo máximo de espera a la respuesta sea de 1000 ms; por otro, capturar las líneas que devuelven el carácter "*" y en este caso, asignar 1000 al valor de tiempo de respuesta. De este modo puedo realizar un cálculo relativo del tiempo medio de respuesta.

Hablando de los comandos, me gustaría especificar algunas particularidades. El programa usa los comandos *tracert*, para comprobar la accesibilidad de una DNS y para calcular el tiempo de respuesta y el número de saltos de las DNS candidatas, y *netsh* para obtener información de las interfaces y para asignar una nueva dirección DNS al adaptador. Puntualizar que, por un lado, el comando *tracert* que se usa para probar la accesibilidad de una dirección DNS se lanza con la opción `-h` porque no es necesario realizar los 30 saltos para comprobar si una dirección es accesible. Por otro lado, el comando *netsh* que cambia la dirección DNS solo puede ejecutarse con privilegios elevados, por lo que debe el programa debe ejecutarse con privilegios de administrador.

Así mismo, como trabajamos con estructuras de datos que además contienen campos tipo *char**, hay funciones específicas para copiar estructuras y añadir datos a las estructuras, que hacen uso interno de la función `strncpy()` y asignan uno a uno los campos de la estructura. Igualmente, se genera la función `mergeToDnsTestTable()` que une las direcciones obtenidas del archivo inicial de DNS y de las DNS accesibles del adaptador. Comentar también, antes de finalizar, que la función de cambio de DNS comprueba si la DNS actual coincide con la DNS más rápida de forma sencilla gracias al uso de estas estructuras.

Por último, he decidido usar el algoritmo Quicksort para solucionar el problema de escoger el mejor adaptador, primero, porque ordenar la tupla me parece una manera sencilla de obtener el resultado, y segundo, porque es un algoritmo muy útil y usado y quería aprender a crear, personalizarlo e implementarlo. Además, me permite seguir probando funcione srecursivas.

Funciones principales: `void adptsLoad(adptsTable* table, int* retVal); void dnsLoad(dnsTable* table, const char* dnsPath, int* retVal); void getDnsIps(const adptsTable table, const int adptrNum, dnsTable* adptrDnsSrvs, int* adptrInd, int* retVal); void testDnsServers(const adptsTable adptsTbl, const dnsTable dnsDirs, const dnsTable adprDnsSrvs, const int adptrInd, int* retVal);`

Funciones auxiliares: `void createAdptsLog(); void createDnsLog(const int index, int* retVal); void addAdapter(adptsTable* table, const adapter adptr, int* retVal); void getAdptr(const char* str, adapter* adptr); int searchAdptrInd(const adptsTable table, const int adptrNum); void addDns(dnsTable* table, const char* str, int* retVal); void addDnsToTestTable(dnsTestTable* table, const char* str, int* retVal); void readDnsLog(dnsTable* table, int* retVal); void findDnsAccessibility(const dnsTable adptrDnsSrvs, FILE* fTempFile, int* retVal); void mergeToDnsTestTable(const dnsTable src, FILE* fTempFile, dnsTestTable* testTable, int* retVal); void speedTest(dnsTestTable* testTable, int* retVal); void changeDns(const adptsTable adptsTbl, const dnsTestTable testTable, const dnsTable adptsDns, const int adptsInd, int* retVal); void qSortDnsTestTable(dnsTestTable* table, int low, int high); int partitionDnsTestTable(dnsTestTable* table, int low, int high); void swapDnsTest(dnsTest* src1, dnsTest* src2); void cpyDnsTest(dnsTest src, dnsTest* dest); void adptrCopy(adapter* dst, const adapter src); void printAdptsList(const adptsTable table); void printDnsTable(const dnsTable table); int printAdptsNames(const adptsTable table); void printBestDns(const dnsTestTable testTable); void clearDnsTable(dnsTable* table); void clearAdptsTable(adptsTable* adptsTbl); void clearDnsTestTable(dnsTestTable* testTable); void freeDnsTest(dnsTest* test); void freeAdapter(adapter* adptr); bool isFullAdpts(const adptsTable table); bool isFullDns(const dnsTable table); bool isFullDnsTest(const dnsTestTable table);`

4. Módulo menu

Para poder implementar los menús, necesitamos comenzar por ser capaces de obtener la lectura de un entero introducido por un usuario. Esta función también incluye el manejo de errores si el usuario entra un entero fuera de las opciones permitidas.

A continuación, implementamos la llamada al menú principal, para ello usaremos un bloque Switch Case que lanzará un *driver* en concreto según la opción escogida por el usuario. Las siguientes funciones implementan los diferentes drivers que delegan en los distintos módulos. Todos tienen nombres que facilitan la lectura y comprensión del flujo del programa. Además, se usa un *driver* de menú y otro de submenú que implementa los bucles de los menús, así simplificamos aún más el código

Lista de funciones: `int readInt(int limit); void callMenu(const adptsTable adpts, const dnsTable dnsDirs, dnsTable* adptrDnsSrvs, bool* isSalir, int* adptrInd, int* retVal); void driverEscogerAdaptador(const adptsTable adpts, dnsTable* adptrDnsSrvs, int* adptrInd, int* retVal); void driverTestServidores(const adptsTable adptsTbl, const dnsTable dnsDirs, const dnsTable adptrDnsSrvr, const int adptrInd, int* retVal);`

5. Módulo principal

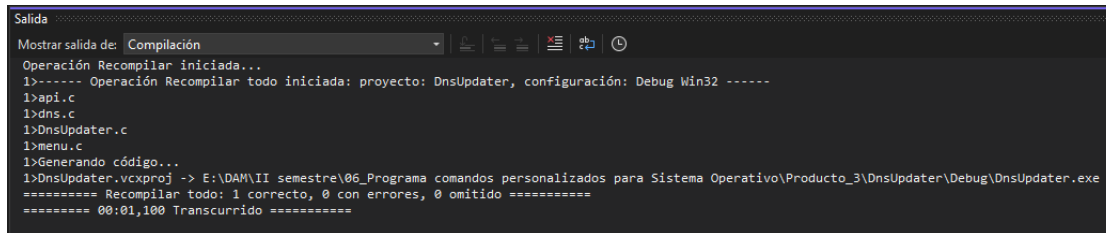
Como en el anterior producto, he intentado mantener el módulo con la función *main* lo más limpia posible, con la lectura más sencilla para ver cómo funciona el flujo del programa. Este módulo,

implementa una función *driver* que se encargará de realizar las llamadas secuenciales y que devuelve el valor de retorno.

Lista de funciones: `int mainDriver();`

Compilación y ejecución

Compilación sin errores

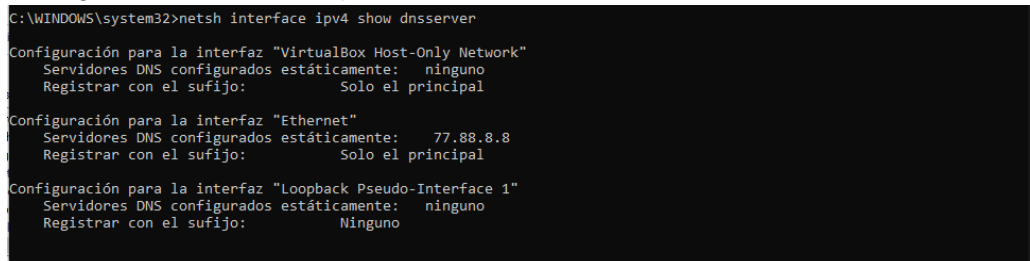


```
Salida
Mostrar salida de: Compilación
Operación Recompilar iniciada...
1>----- Operación Recompilar todo iniciada: proyecto: DnsUpdater, configuración: Debug Win32 -----
1>api.c
1>dns.c
1>DnsUpdater.c
1>menu.c
1>Generando código...
1>DnsUpdater.vcxproj -> E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_3\DnsUpdater\Debug\DnsUpdater.exe
===== Recompilar todo: 1 correcto, 0 con errores, 0 omitido =====
===== 00:01,100 Transcurrido =====
```

Ejecución

1. Datos de sistema previos a la ejecución:

Configuro la DNS del adaptador como:



```
C:\WINDOWS\system32>netsh interface ipv4 show dnsserver

Configuración para la interfaz "VirtualBox Host-Only Network"
  Servidores DNS configurados estáticamente: ninguno
  Registrar con el sufijo: Solo el principal

Configuración para la interfaz "Ethernet"
  Servidores DNS configurados estáticamente: 77.88.8.8
  Registrar con el sufijo: Solo el principal

Configuración para la interfaz "Loopback Pseudo-Interface 1"
  Servidores DNS configurados estáticamente: ninguno
  Registrar con el sufijo: Ninguno
```

2. Carga de datos y menú principal

```
E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_3\DnsUpdater\Debug\DnsUpdater.exe
-----
Bienvenido al programa de Actualizacion de direccion DNS.
-----
Carga de los datos iniciales.
-----
Por favor, indique la ruta de acceso al archivo, debe usar / para separar los directorios, no use \.
Por ejemplo, C:/dev/
./
Por favor, indique el nombre del archivo con su tipo.
Por ejemplo, mifichero.txt
DNSips.txt
#1. Direcci n DNS: 192.168.17.127
#2. Direcci n DNS: 8.8.8.8
#3. Direcci n DNS: 6.6.6.6
#4. Direcci n DNS: 192.168.17.125
-----
1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----
```

3. Imposibilidad de realizar la actualización sin haber escogido un adaptador previamente.

```
-----
1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----
2
Por favor, escoja primero un adaptador de red mediante el paso 1.
-----
1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----
```

4. Selección de adaptador.

```

E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_3\DnsUpdater\Debug\DnsUpdater.exe
-----
1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----
1
-----
Lista de adaptadores de red de este equipo:
#1 Name: Ethernet
#2 Name: VirtualBox
-----
Por favor, indique el numero de adaptador del que desea obtener informacion.
Escoja un entero entre 1 y 2:
1
-----
Las direcciones DNS para este adaptador son:
-----
Direcciones DNS para el adaptador Ethernet:
#1. Dirección DNS: 77.88.8.8
-----
1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----

```

5. Actualización de DNS.

Test de accesibilidad:

```

-----
1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----
2
-----
Iniciando la actualizacion del servidor DNS:
-----
Realizando el test de accesibilidad a los servidores del adaptador:
(Las acciones pueden demorarse un poco)
-----
#1. Test sobre el servidor 77.88.8.8
Accesible
-----
Realizando el test de velocidad a los servidores DNS:
(Las acciones pueden demorarse un poco)
-----

```

Test de velocidad:

```

E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_3\DnsUpdater\Debug\DnsUpdater.exe
Realizando el test de velocidad a los servidores DNS:
(Las acciones pueden demorarse un poco)
-----
#1. Test sobre el servidor 192.168.17.127
*****
Test de velocidad finalizado.
-----
Resultados del test sobre el servidor: 192.168.17.127
Saltos totales: 30, Tiempo medio de respuesta: 801.87
-----
#2. Test sobre el servidor 8.8.8.8
*****
Test de velocidad finalizado.
-----
Resultados del test sobre el servidor: 8.8.8.8
Saltos totales: 12, Tiempo medio de respuesta: 95.42
-----
#3. Test sobre el servidor 6.6.6.6
*****
Test de velocidad finalizado.
-----
Resultados del test sobre el servidor: 6.6.6.6
Saltos totales: 30, Tiempo medio de respuesta: 801.67
-----
#4. Test sobre el servidor 192.168.17.125
*****
Test de velocidad finalizado.
-----
Resultados del test sobre el servidor: 192.168.17.125
Saltos totales: 30, Tiempo medio de respuesta: 801.73
-----
#5. Test sobre el servidor 77.88.8.8
*****
Test de velocidad finalizado.
-----
Resultados del test sobre el servidor: 77.88.8.8
Saltos totales: 11, Tiempo medio de respuesta: 119.36
-----
Datos del servidor mas rapido:
Direccion: 8.8.8.8      Numero de salto: 12      Tiempo de respuesta medio: 95.42

Se ha cambiado el servidor DNS del adaptador de red con exito!

-----
1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----

```

Configuración actualizada en el equipo:

```

C:\WINDOWS\system32>netsh interface ipv4 show dnsserver

Configuración para la interfaz "VirtualBox Host-Only Network"
  Servidores DNS configurados estáticamente:  ninguno
  Registrar con el sufijo:                      Solo el principal

Configuración para la interfaz "Ethernet"
  Servidores DNS configurados estáticamente:    8.8.8.8
  Registrar con el sufijo:                      Solo el principal

Configuración para la interfaz "Loopback Pseudo-Interface 1"
  Servidores DNS configurados estáticamente:  ninguno
  Registrar con el sufijo:                    Ninguno

```

Prueba con el mejor adaptador de red ya seleccionado:

```

-----
Lista de adaptadores de red de este equipo:
#1 Name: Ethernet
#2 Name: VirtualBox
-----

Por favor, indique el numero de adaptador del que desea obtener informacion.
Escoja un entero entre 1 y 2:
1
-----

Las direcciones DNS para este adaptador son:
-----
Direcciones DNS para el adaptador Ethernet:
#1. Dirección DNS: 8.8.8.8
-----

#4. Test sobre el servidor 192.168.17.125
*****
Test de velocidad finalizado.
-----
Resultados del test sobre el servidor: 192.168.17.125
Saltos totales: 30, Tiempo medio de respuesta: 801.63
-----

#5. Test sobre el servidor 8.8.8.8
*****
Test de velocidad finalizado.
-----
Resultados del test sobre el servidor: 8.8.8.8
Saltos totales: 12, Tiempo medio de respuesta: 99.75
-----

Datos del servidor mas rapido:
Dirección: 8.8.8.8      Numero de salto: 12      Tiempo de respuesta medio: 95.92
Ya esta seleccionado el mejor adaptador.
-----

1. Escoger adaptador de red.
2. Realizar test de servidores DNS sobre el adaptador.
0. Salir
Escoja 1, 2, o 0:
-----
  
```

Archivos adjuntos

Junto con los archivos requeridos por la actividad, he incluido dos archivos más que pienso pueden resultar útiles:

1. **Directorio con el archivo ejecutable por separado**, está en una carpeta a parte para que sea más sencilla su ejecución y comprobar que no deja archivos residuales.
2. **Documentación de todas las estructuras y funciones del programa**: Además de incluirla en los comentarios, he pensado que resultaría útil extraerla a un archivo externo.

Anexos

I. Soluciones para la ejecución de un comando con privilegios elevados.

Uno de los puntos en los que he tenido que realizar más búsquedas era en las posibles soluciones que tenía para poder ejecutar el comando *set* de *netsh*. Para ello, necesitaba poder lanzar un comando en un CMD con privilegios elevados. He estado informándome para intentar dar con una solución que no requiriera la ejecución del programa como administrador, pero tras revisar las opciones, he creído que esta era la más sencilla para el usuario.

Tras las primeras búsquedas, comprobé que no se puede lanzar un CMD con privilegios elevados directamente desde C, así que seguí buscando y di con 3 posibles vías:

1. **Lanzar un el comando por PowerShell:** Estuve investigando para ver si podía lanzar un comando por PowerShell en lugar de por CMD, pero toda la información que encontré dice que no es posible hacerlo con funciones de C o C++. De este modo, la manera de poderlo hacer es crear un Bash Script y ejecutarlo leyendo el archivo desde C. El problema es que creo que esto implicaría complicar mucho más el código y aún no domino muy bien la creación de Bash.
2. **Aumentar los privilegios del usuario con un *runas*:** Pensando en como en Linux usa el comando *sudo*, la otra solución más habitual que hallé fue la de lanzar el comando mediante un *runas*. Para ello, tenía que anidar la sentencia del comando *netsh* dentro del comando *runas*. Mediante *runas /user:<administrador> "comando"* podemos ejecutar el comando con privilegios de administrador; sin embargo, esta solución presenta dos problemas. El primero, que para poderlo ejecutar es necesario conocer el nombre y la contraseña de una cuenta de administrador y, el segundo, que el equipo tiene que tener la cuenta de administrador habilitada. Al final, esta es la solución que usé para ir realizando los tests de funcionamiento de la aplicación ejecutando el run desde Visual Code, ya que era la única forma de comprobar que todo el proceso se ejecutaba correctamente sin errores. Una función temporal preguntaba el nombre de cuenta de usuario administrador y lo añadía mediante *sprintf()* al string con la sentencia del comando: *"runas /user:? \"netsh interface ipv4 set dnsserver ? static ?\"*. Donde los '?' se correspondían a los parámetros "adminName", "adapterIndex" y "dnsDir". Dejo constancia porque pienso que puede resultar una solución muy útil según el escenario.
3. **Ejecutar el programa como administrador:** Al final, por la simpleza de uso, decidí mantener esta opción en la versión final.

Recursos

MSFT, T. (2022, 2 diciembre). *_popen, _wopen*. Microsoft Learn. <https://learn.microsoft.com/es-es/cpp/c-runtime-library/reference/popen-wopen?view=msvc-170>

MSFT, T. (2022a, octubre 26). *fprintf_s, _fprintf_s_l, fwprintf_s, _fwprintf_s_l*. Microsoft Learn. <https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/fprintf-s-fprintf-s-l-fwprintf-s-fwprintf-s-l?view=msvc-170>

C Library -. (s. f.). https://www.tutorialspoint.com/c_standard_library/string_h.htm

Gerend, J. (2021, 27 octubre). *Network Shell (Netsh)*. Microsoft Learn. <https://learn.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh>

sscanf() - c. (s. f.). <https://sites.google.com/site/sencillamentec/home/entrada-de-datos/sscanf>

C library function - *sscanf()*. (s. f.). https://www.tutorialspoint.com/c_standard_library/c_function_sscanf.htm

C library function - *strtoul()*. (s. f.). https://www.tutorialspoint.com/c_standard_library/c_function_strtoul.htm

C library function - strtoul(). (s. f.-b). https://www.tutorialspoint.com/c_standard_library/c_function_strtoul.htm

www.techonthenet.com. (s. f.). *C Language: strtod function (Convert String to Double)*.

https://www.techonthenet.com/c_language/standard_library_functions/stdlib_h/strtod.php

GeeksforGeeks. (2017, 4 septiembre). *tmpfile() function in C*. <https://www.geeksforgeeks.org/tmpfile-function-c/>

GeeksforGeeks. (2022, 27 septiembre). *QuickSort*. <https://www.geeksforgeeks.org/quick-sort/>

C library function - strtol(). (s. f.). https://www.tutorialspoint.com/c_standard_library/c_function_strtol.htm

tmpfile, tmpfile_s - cppreference.com. (s. f.). <https://en.cppreference.com/w/c/io/tmpfile>

GeeksforGeeks. (2017b, septiembre 4). *tmpfile() function in C*. <https://www.geeksforgeeks.org/tmpfile-function-c/>

Albert, M. (2014, 8 enero). *Windows: Show and configure network settings using netsh*. Michls Tech Blog.

<https://michlstechblog.info/blog/windows-show-and-configure-network-settings-using-netsh/>