



PROGRAMA COMANDOS PERSONALIZADOS PARA SISTEMA OPERATIVO

AA2. ADQUIRIENDO LAS DESTREZAS BÁSICAS

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 Ping	5
2. Módulo Adaptador	6
3. Módulo Menú	7
4. Módulo LinkedList	8
5. Módulo <i>main</i> PingProgram	8
COMPILACIÓN Y EJECUCIÓN	9
Compilación sin errores	9
Ejecución	9
1. Menú principal	9
2. Realizar ping	9
3. Información del adaptador de red	10
4. Funcionalidades extra	11
5. Finalización del programa sin errores	12
ARCHIVOS ADJUNTOS	13
RECURSOS	13

AA2. ADQUIRIENDO LAS DESTREZAS BÁSICAS

Descripción del programa

Análisis de los requisitos del programa

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

1. **Función ping** → El programa debe implementar una función para realizar ping a una lista de direcciones IPv4 almacenadas en un documento de texto exterto.
 - a. El programa tiene que acceder a un archivo externo ya creado con las direcciones IP, para la facilidad de lectura, cada IP se grabará en una nueva línea. Guardamos el documento en la carpeta raíz para facilitar el acceso.
 - b. La ruta del archivo externo se obtendrá a través de la interacción con el usuario.
 - c. Abriremos el archivo y realizaremos una lectura de los datos que contiene.
 - d. Lanzamos un comando de *Shell* desde el código C que ejecute el Ping a cada una de las direcciones.
 - e. Obtenemos el resultado de cada Ping y discriminamos si ha sido realizado con éxito o ha fallado, mostrando los resultados por pantalla.
2. **Función información de los adaptadores de red** → El programa debe implementar una función para obtener la dirección IPv4, la máscara de red y la puerta de enlace de un adaptador de red de los disponibles por el sistema que escoja el usuario. El programa mostrará solo la información del adaptador de red escogido por el usuario y discriminará entre la información requerida y la no requerida, mostrando por pantalla solo la primera.
 - a. El programa debe conseguir una lista de los adaptadores de red instalados en el equipo.
 - b. Mostrará los adaptadores de red al usuario y este debe de escoger sobre cuál desea obtener la información.
 - c. Obtendrá la información del adaptador de red solicitado lanzando un comando de *Shell ipconfig*.
 - d. Discriminará entre toda la información obtenida y mostrará solo los datos requeridos del adaptador solicitado por el usuario.
3. Toda la **selección de funcionalidades se realizará desde un menú interactivo** donde el usuario podrá escoger qué desea hacer.
 - a. El programa debe de tener un menú principal para escoger entre las distintas funcionalidades,
 - b. Se la funcionalidad lo requiere, el programa debe tener submenús para que el usuario escoja entre las distintas opciones.

Estos serían los requisitos principales según el enunciado de la actividad; además, con objetivo de completar el programa con funcionalidades extra y de aprender elementos importantes para el desarrollo de programas como pueden ser; el uso de estructuras de datos o mejorar la interacción con los archivos externos, el uso de punteros o el manejo de la memoria dinámica, he intentado implementar una serie de requisitos añadidos:

4. **El programa usará una lista enlazada** para almacenar la información de los adaptadores. La lista enlazada es una de las estructuras de datos más usadas junto al array, ya que permite una gestión dinámica del uso de la memoria.
 - a. Creación de las estructuras necesarias para implementar una lista enlazada.
 - b. Implementación del inicializador para facilitar el trabajo con la lista.
 - c. Creación de las funciones necesarias para trabajar con la lista:
 - i. Adjuntar un nuevo elemento.
 - ii. Buscar un elemento de la lista.
 - iii. Imprimir los elementos de la lista.
 - iv. Limpiar la lista.
 - d. Implementación de la lista en el módulo para obtener la información de los adaptadores → Genera nuevo código en el punto 2.
5. **El programa generará archivos de log** con la información obtenida del comando *ipconfig* para facilitar el tratamiento de los datos → Este requisito añade requisitos al punto 2:
 - a. Generar un archivo de log para obtener la lista de adaptadores de red.
 - b. Generar un archivo de log para obtener la información de los adaptadores de red.
6. **Funcionalidad nueva: Borrar los archivos de log** → El programa deberá permitir al usuario ejecutar una eliminación de los archivos de log generados para liberar memoria.
 - a. Borrado del log con la lista de adaptadores de red del sistema.
 - b. Borrar el log con la información de los adaptadores de red del sistema.
7. **Funcionalidad nueva: Añadir una nueva dirección IP** → El programa permitirá al usuario añadir una nueva dirección a la lista de IPs a las que realizar el Ping.
 - a. Solicitar los datos necesarios al usuario: ruta, nombre de archivo y dirección IP a añadir.
 - b. Acceder al archivo externo en modo adjuntar.
 - c. Escribir la nueva línea en el archivo.
8. **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.

La identificación de los requisitos es muy importante ya que nos ayudará a realizar un análisis descendente del programa y diseñar las funciones necesarias para su correcta implementación. Ordenarlo de forma arborescente, creando subestructuras de requisitos, será útil para diseñar la modularización del programa, ya que nos informa de qué módulos generales vamos a necesitar y nos guía sobre las funciones que agrupará cada módulo.

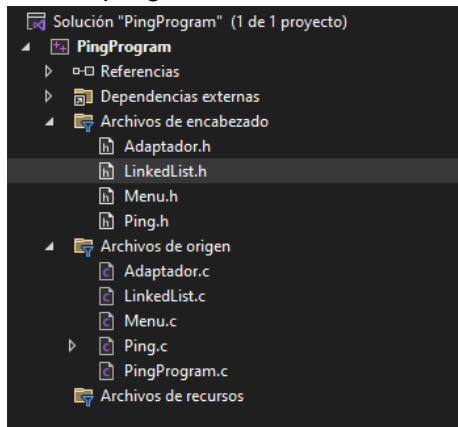
Modularización del programa

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 linkar 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

A través del análisis de requisitos podemos empezar a realizar el diseño descendente del programa. Mediante los requisitos principales, conoceremos las funciones principales que debemos crear; cada función principal la asociamos a un módulo, donde crearemos todas las funciones necesarias que surjan de la división de la principal en sus requisitos individuales. Así, cada módulo representa un problema general a solucionar y, dentro de cada módulo, desarrollamos la solución de los problemas particulares que surjan del análisis descendente del problema general.

1. **Módulo Ping** → Soluciona: Realizar un ping a una lista de direcciones IP obtenidas de un archivo y mostrar los resultados por pantalla.
 - a. Extra: Implementar una función para añadir una nueva dirección IP a la lista.
2. **Módulo Adaptador** → Soluciona: Mostrar la lista de adaptadores al usuario y, tras este indicar uno de ellos, imprimir por pantalla la siguiente información relacionada con ese adaptador: Nombre del adaptador, dirección IPv4, máscara de red y puerta de enlace.
 - a. Extra: Implementar la funcionalidad que permita borrar los archivos de log generados por el uso de este módulo.
3. **Módulo Menú** → Soluciona: Toda la selección de funcionalidades se realizará desde un menú interactivo donde el usuario podrá escoger las opciones que desea ejecutar e introducir la información necesaria.
4. **Módulo LinkedList** → Soluciona: La información de los adaptadores se almacenará en una estructura de datos tipo lista enlazada.
5. **Módulo con el *main*: PingProgram** → Módulo para inicializar y controlar el flujo del programa.



Solución de los requisitos de cada módulo

A lo largo del código he intentado implementar 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()` e incluso he probado `fopen_f()` en lugar de `fopen()` para aprender cómo funciona, ya que en *Microsoft Learn* explican que es un método más seguro. También, 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.

Del mismo modo, en los archivos de cabecera `.h`, se han incluido constantes para usar como parámetros de funciones de forma más cómoda y con mayor control. 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 Ping

Para trabajar más cómodamente dentro del código, guardaremos los valores de las direcciones IP que leamos del documento en una tabla que inicializamos con un tamaño de 100. Tras esto, obtendremos los datos de la ruta de localización del archivo, le preguntamos al usuario por pantalla los datos y los asignamos a dos variables tipo string (`char []`) que concatenaremos mediante la función `strncat()` y asignamos la ruta completa un parámetro `en/sal` mediante `strncpy()`. Como son funciones muy semejantes, solo se incluye un bloque extra de código, seguidamente se codifica la función para obtener los datos de la nueva IP, igualmente preguntamos la ruta del archivo y, además, la IP, que también se pasa mediante un parámetro de `en/sal`. Todas las lecturas de strings del usuario se realizan mediante el método `fgets()` ya que es más seguro y lee todo la cadena introducida, incluidos los espacios en blanco, pero hay que recordar eliminar el carácter `'\n'` de la lectura obtenida.

A continuación, debemos abrir el archivo, lo he codificado en una función porque realizado el manejo de errores dentro de la función y así no tiene que repetirse en cada uso de `fopen()`. Seguidamente, con el archivo ya abierto, debemos leer cada línea del archivo con `fgets()` y la almacenamos en la tabla, además, como en la estructura de la tabla hemos usado punteros (`char *`) en lugar de un array de tamaño fijo, hemos de asignar la memoria antes de asignar el valor. Como trabajamos dentro del bucle `While` para realizar la lectura, lo aprovechamos para actualizar un índice de posición en el array. Hemos de recordar cerrar el archivo tras finalizar la operación de lectura.

La siguiente función, implementa el lanzamiento del comando ping a las diferentes direcciones IP que hemos cargado en la tabla. Para ello usaremos el método `system()` y le pasaremos una variable que contendrá un String con el comando, concatenando a una parte fija, las diferentes IP, para no ver el comando por pantalla, usamos otra parte fija que redirija el

comando (> nul). Seguimos, separando en una función a parte el código para lanzar la llamada ping a cada una de las direcciones de la tabla, para ello recorremos una tabla con una **función recursiva**.

Finalizamos esta parte de los requisitos creando una función para mostrar los resultados por pantalla que además discrimine si el resultado ha sido un éxito o un fallo. Para ello, cuando hemos generado la estructura de la tabla, hemos creado dos arrays, uno para guardar la dirección y otro para guardar un entero que indica el resultado del ping. Como ambos campos están relacionados con la misma posición en la tabla, podemos gestionar fácilmente los resultados del ping. Por último, una vez finalizada esta funcionalidad, podemos liberar la memoria usando un método para liberar las posiciones de la tabla y reiniciarla.

El módulo finaliza codificando la función extra para grabar una nueva dirección en el archivo de externo de direcciones y dos funciones que sirven como drivers para que el módulo Menú pueda delegar en ellas y ser más sencillo de escribir y leer. La función de escritura, abre el archivo en modo adjuntar, para acceder a la última posición, y escribe en el archivo mediante `fprintf()`, para añadir tanto la nueva dirección como un salto de línea.

Lista de funciones: `void initTablaIpAddr(tablaIpAddr* tabla), void getDatos(char rutaAbsoluta[]), void getDatosIp(char ruta[], char ip[]), FILE* openFile(const char* nombre, const char* modo), tablaIpAddr listIp(char rutaAbsoluta[], char modo[]), int pingIp(const char* ipAddr), int getPingResponse(tablaIpAddr* tabla, const int end, const int ind), void printPingResult(const tablaIpAddr tabla), int clearTable(tablaIpAddr* table, const int ind), void addIp(const char ruta[], const char ipDir[]), int ipPingDriver(), void addIpDriver().`

2. Módulo Adaptador

Para implementar esta solución, dividiremos la estructura en dos partes, la obtención de los nombres de los adaptadores y la obtención de la información de un adaptador en concreto.

Comenzamos creando, como en el módulo anterior, una función de apertura de archivos que maneje los errores. En esta ocasión he usado `open_f()` porque en Microsoft Learn explican que es más segura. A continuación, apoyándonos en la función de apertura, generamos la función para la creación de los logs. Con estas funciones de apoyo, implementamos la función que obtiene los nombres de los adaptadores y los graba en el log que hemos generado. Para ello, lanzaremos un comando de CMD que incluye un *pipe* y permite obtener únicamente el nombre del adaptador al lanzar *ipconfig*. Así, cada adaptador se quedará grabado como una línea en el archivo. El comando se genera uniendo una constante con la ruta del archivo que queremos crear unida a la parte fija con el *pipe* (*findstr*) y el operador de redirección.

A continuación, he decidido usar una lista enlazada, para aprender como funcionan estas estructuras tan comunes, y almacenar los nombres de los adaptadores para que sea más sencillo trabajar con ellos, igualmente, la estructura del nodo guarda una llave que funciona

como índice para localizarlos y facilitar la selección por parte del usuario. Además, grabamos el nombre purgando el carácter ':' del String original.

La siguiente parte del módulo, implementa las funciones para mostrar la información del adaptador seleccionado por el usuario. La primera función crea un log para almacenar la información arrojada por un comando *ipconfig()* y poder trabajar posteriormente con ella. Esta función lanza el comando de *Shell* mediante el método *system()* y redirecciona la salida al archivo log. Por último, creamos la función para la discriminación de los datos y la impresión de estos por pantalla. Esta función accede a la lista, escoge el nombre del adaptador seleccionado por el usuario y lo usa para encontrar los datos correctos en el log de información de los adaptadores. Dentro de esta función se usan punteros, con lo que hay que asignar y liberar memoria y asignación de variables temporales para la localización de la información. Accedemos a los datos del archivo línea a línea, usamos el método de la librería *string.h* para localizar el adaptador que buscamos y, a partir de ahí, con *ftell()* y *fseek()* mantenemos la posición de lectura en el punto que nos interesa para seguir buscando los datos relacionados con el adaptador. Para localizarlos, usamos la función *strstr()* que permite localizar substrings dentro de un String, lo hacemos buscando palabras o Strings clave que identifican la información y devolviendo el substring que hemos hallado. Como C trabaja con ASCII, manejo el problema con algunos caracteres tanto en la impresión de los datos como en las *keywords*.

Por último, el módulo contiene los drivers para que Menú delegue sus funciones en este módulo y sea más simple, y las dos funciones para poder borrar los logs que usan el método *remove(archivo)*.

Lista de funciones: FILE* createArchivoLog(char* nombre, char* modo), void createFileNombresAdptr(char* nombreArchivo, char* modo), listaAdptr fillListaAdaptadores(), void cargarInformacionAdaptadores(), void mostrarInformacionAdptr(const listaAdptr lista, const int numAdptr), void deleteLogAdptrs(), void deleteLogInfoAdptrs(), listaAdptr driverAdptrs(), void driverMostrarInformacionAdptrs(listaAdptr* adptrs, int key).

3. Módulo Menú

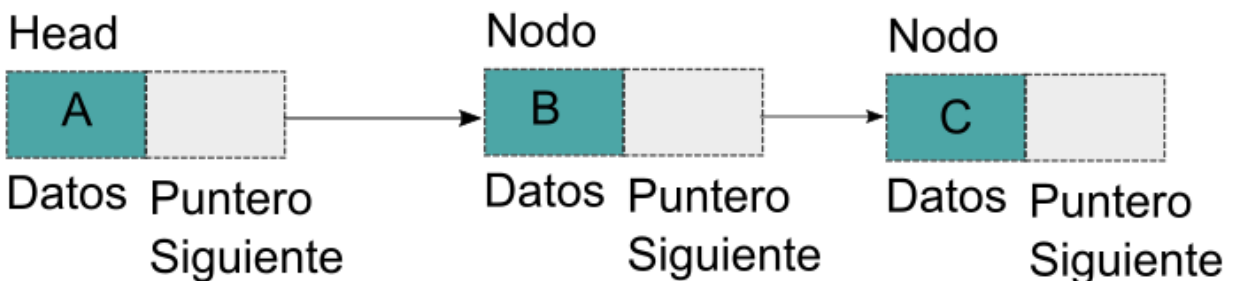
Para poder implementar los menús, necesitamos comenzar por ser capaces de obtener la lectura de un entero introducido por un usuario. Esta función también incluye el manejo de errores si el usuario entra un entero fuera de las opciones permitidas.

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 leerEntero(int limit), bool callMenu(), void menuDriver(), void driverRealizarPing(), void driverInformacionAdaptadores(), void borrarLogListaAdptrs(), void borrarLogInfoAdptrs(), bool callSubmenuBorrado(), void submenuDriver(), void driverAniadirIp()`.

4. Módulo LinkedList

Este módulo define las estructuras necesarias y las funciones para implementar una lista enlazada con la que trabajaremos en el módulo Adaptador. Una lista enlazada trabaja de forma dinámica, creando nodos que contienen los datos cuando sea necesario y enlazando los nodos a través de punteros al siguiente nodo. Para la inicialización, debemos crear el puntero un primer nodo, que se conoce como head, con valores nulos para los datos y para el puntero al siguiente nodo.



Además, implementamos las funciones necesarias para trabajar con la lista: Un iniciador de la lista, una función para añadir un elemento nuevo a la última posición de la lista, una para la búsqueda de un adaptador a través de un entero que funciona como clave de acceso, una para limpiar la lista y liberar la memoria y otra para imprimir por pantalla los datos de cada nodo; además, he intentado usar funciones recursivas para mejorar mi conocimiento sobre el uso de estas.

Lista de funciones: `void initListaAdptr(listaAdptr* lista), void appendAdptr(listaAdptr* lista, const char* nombreAdptr), adptr* buscarAdptrPorKey(adptr* head, const int key), void clearListaAdptr(listaAdptr* lista, adptr* head), void printListaAdptrs(const adptr* head)`.

5. Módulo *main* PingProgram

He intentado mantener el módulo que contiene el método *main* lo más sencillo y limpio posible, encargándose solo de manejar el flujo del programa. Así, únicamente implementa un *driver* que realiza la llamada al menú y la llamada a este *driver* y el retorno de salida sin errores.

Lista de funciones: `void driverMain()`.

Compilación y ejecución

Compilación sin errores

```

Salida
Mostrar salida de: Compilación
Operación Compilar iniciada..
1>----- Operación Compilar iniciada: Proyecto: PingProgram, configuración: Debug Win32 -----
1>Adaptador.c
1>LinkedList.c
1>Menu.c
1>Generando código...
1>PingProgram.vcxproj -> \\DESKTOP-NPC70GD\fp_uoc\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_2\Pi
===== Compilación: 1 correcto, 0 erróneo, 0 actualizado, 0 omitido =====
===== 00:09,783 Transcurrido =====
  
```

Ejecución

1. Menú principal

```

E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_2\PingProgram\Debug\PingProgram.exe
Bienvenido al programa de llamadas ping!
Por favor, escoja una opcion del menu:
-----
1. Realizar ping a direccion IP -> Debera escoger un archivo donde esten almacenadas las direcciones.
2. Mostrar parametros de su adaptador de red.
3. Borrar los archivos de log generados.
4. Aniadir una nueva direccion IP
0. Salir
Escoja 1, 2, 3, 4 o 0:
-----
  
```

2. Realizar ping

Para facilitar el acceso, he colocado el archivo en la carpeta raíz del programa, con lo que se puede acceder mediante una ruta “./”. Hay 5 direcciones en la lista: 4 direcciones de red Lan de equipos conectados con IP fija, 2 a equipos reales que devolverán respuesta y 2 a direcciones que no están adjudicadas para que devuelvan error. La última dirección es al servidor de DNS público de Google, para comprobar la respuesta a una IP externa.

```

*ipdirs: Bloc de notas
Archivo Edición Formato Ver Ayuda
192.168.1.22
192.168.1.41
192.168.1.40
192.168.1.28
8.8.8.8
  
```

```

E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_2\PingProgram\Debug\PingProgram.exe
-----
1
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
ipdirs.txt

Lista de IP:
-----
#1: 192.168.1.22
#2: 192.168.1.41
#3: 192.168.1.40
#4: 192.168.1.28
#5: 8.8.8.8

Resultados de los echo:
-----
El ping a la IP 192.168.1.22 ha respondido con exito.
El ping a la IP 192.168.1.41 ha respondido con exito.
El ping a la IP 192.168.1.40 no ha devuelto respuesta.
El ping a la IP 192.168.1.28 no ha devuelto respuesta.
El ping a la IP 8.8.8.8 ha respondido con exito.
-----

Bienvenido al programa de llamadas ping!
Por favor, escoja una opcion del menu:

```

3. Información del adaptador de red

Primero obtenemos la lista y la mostramos por pantalla, luego realizamos una selección interactiva donde el usuario, para simplificar la usabilidad, escoge el adaptador por su número. Para finalizar, el programa muestra la IPv4, máscara de red y puerta de enlace del adaptador escogido. Además, el programa habrá creado dos archivos log que usa para gestionar los datos.

```

E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_2\PingProgram\Debug\PingProgram.exe
-----
2
Lista de adaptadores de red de este equipo:
#1: Adaptador de Ethernet VirtualBox Host-Only Network
#2: Adaptador de Ethernet Ethernet
-----

Por favor, indique el numero de adaptador del que desea obtener informacixn.
Escoja un entero entre 1 y 2:
2
2
Informacion del Adaptador de Ethernet Ethernet:
Direccion IPv4. . . . . : 192.168.1.21
Mascara de de subred . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . . . : 192.168.1.1
-----

Bienvenido al programa de llamadas ping!
Por favor, escoja una opcion del menu:
-----
1. Realizar ping a direccion IP -> Debera escoger un archivo donde esten almacenadas las direcciones.
2. Mostrar parametros de su adaptador de red.
3. Borrar los archivos de log generados.
4. Aniadir una nueva direccion IP
0. Salir
Escoja 1, 2, 3, 4 o 0:
-----

```

Mostramos el retorno de *ipconfig* para comprobar la validez de los datos.

```

C:\Users\UrHek>ipconfig

Configuración IP de Windows

Adaptador de Ethernet VirtualBox Host-Only Network:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::f181:1714:6a6:2a75%6
    Dirección IPv4. . . . . : 192.168.56.1
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . . :

Adaptador de Ethernet Ethernet:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::c4fc:5883:4257:282f%3
    Dirección IPv4. . . . . : 192.168.1.21
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . . : 192.168.1.1
  
```

Y los dos archivos log generados:

```

log_adaptadores: Bloc de notas
Archivo Edición Formato Ver Ayuda
Adaptador de Ethernet VirtualBox Host-Only Network:
Adaptador de Ethernet Ethernet:

log_ipconfig: Bloc de notas
Archivo Edición Formato Ver Ayuda

Configuración IP de Windows

Adaptador de Ethernet VirtualBox Host-Only Network:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::f181:1714:6a6:2a75%6
    Dirección IPv4. . . . . : 192.168.56.1
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . . :

Adaptador de Ethernet Ethernet:

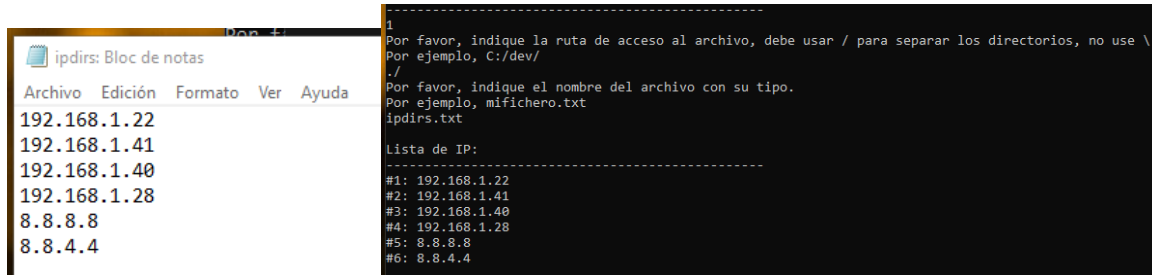
    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::c4fc:5883:4257:282f%3
    Dirección IPv4. . . . . : 192.168.1.21
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . . : 192.168.1.1
  
```

4. Funcionalidades extra

A. Añadir dirección IP nueva:

```

E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_2\PingProgram\Debug\PingProgram.exe
4. Añadir una nueva dirección IP
0. Salir
Escoja 1, 2, 3, 4 o 0:
4
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
ipdirs.txt
Por favor, indique la nueva dirección IP.
Por ejemplo, 192.168.1.2
8.8.4.4
Se ha añadido la IP con éxito!
  
```



ipdirs: Bloc de notas

Archivo Edición Formato Ver Ayuda

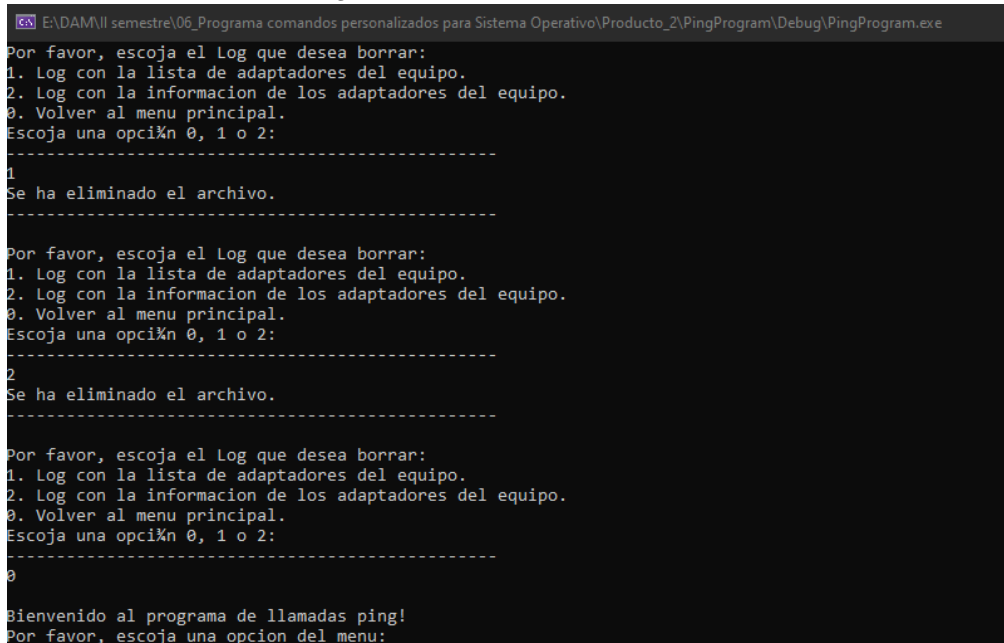
192.168.1.22
192.168.1.41
192.168.1.40
192.168.1.28
8.8.8.8
8.8.4.4

```

1
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
ipdirs.txt

Lista de IP:
-----
#1: 192.168.1.22
#2: 192.168.1.41
#3: 192.168.1.40
#4: 192.168.1.28
#5: 8.8.8.8
#6: 8.8.4.4
  
```

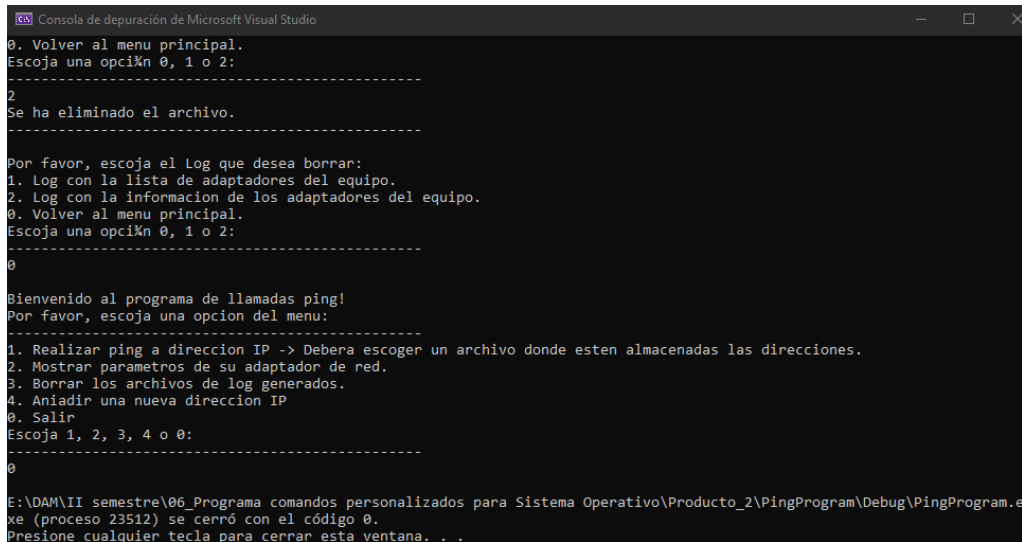
B. Borrar archivos log



```

E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_2\PingProgram\Debug\PingProgram.exe
Por favor, escoja el Log que desea borrar:
1. Log con la lista de adaptadores del equipo.
2. Log con la informacion de los adaptadores del equipo.
0. Volver al menu principal.
Escoja una opción 0, 1 o 2:
-----
1
Se ha eliminado el archivo.
-----
Por favor, escoja el Log que desea borrar:
1. Log con la lista de adaptadores del equipo.
2. Log con la informacion de los adaptadores del equipo.
0. Volver al menu principal.
Escoja una opción 0, 1 o 2:
-----
2
Se ha eliminado el archivo.
-----
Por favor, escoja el Log que desea borrar:
1. Log con la lista de adaptadores del equipo.
2. Log con la informacion de los adaptadores del equipo.
0. Volver al menu principal.
Escoja una opción 0, 1 o 2:
-----
0
Bienvenido al programa de llamadas ping!
Por favor, escoja una opción del menu:
  
```

5. Finalización del programa sin errores



```

Consola de depuración de Microsoft Visual Studio
0. Volver al menu principal.
Escoja una opción 0, 1 o 2:
-----
2
Se ha eliminado el archivo.
-----
Por favor, escoja el Log que desea borrar:
1. Log con la lista de adaptadores del equipo.
2. Log con la informacion de los adaptadores del equipo.
0. Volver al menu principal.
Escoja una opción 0, 1 o 2:
-----
0
Bienvenido al programa de llamadas ping!
Por favor, escoja una opción del menu:
-----
1. Realizar ping a direccion IP -> Debera escoger un archivo donde esten almacenadas las direcciones.
2. Mostrar parametros de su adaptador de red.
3. Borrar los archivos de log generados.
4. Anadir una nueva direccion IP
0. Salir
Escoja 1, 2, 3, 4 o 0:
-----
0
E:\DAM\II semestre\06_Programa comandos personalizados para Sistema Operativo\Producto_2\PingProgram\Debug\PingProgram.exe (proceso 23512) se cerró con el código 0.
Presione cualquier tecla para cerrar esta ventana. . .
  
```

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 se pueda comprobar como genera los logs en la carpeta raíz.
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.

Recursos

Uso de comandos cmd para redes. (s. f.). <https://www.tecnologia-informatica.es/uso-de-cmd-en-redes/>

ipconfig. (2021, 3 marzo). Microsoft Learn. <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/ipconfig>

System() Function in C/C++. (s. f.). <https://www.tutorialspoint.com/system-function-in-c-cplusplus>

[SOLVED]Calling ping from a C program. / Programming & Scripting / Arch Linux Forums. (s. f.). <https://bbs.archlinux.org/viewtopic.php?id=213878>

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

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

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

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

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

GeeksforGeeks. (2017, 2 junio). *fseek() in C/C++ with example.* <https://www.geeksforgeeks.org/fseek-in-c-with-example/>

GeeksforGeeks. (2022, 10 marzo). *ftell() in C with example.* <https://www.geeksforgeeks.org/ftell-c-example/>

fopen_s, _wfopen_s. (2022, 25 octubre). Microsoft Learn. <https://learn.microsoft.com/es-es/cpp/c-runtime-library/reference/fopen-s-wfopen-s?view=msvc-170>

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

IP Helper - Win32 apps. (2022, 19 agosto). Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/_iphlp/