

(P) ICX0_P4 Programa comandos personalizados para el sistema operativo

Código del proyecto: ICX0_P4

Producto 3 - Automatización básica de la administración de un sistema operativo

| | |
|-------------------|--------------------|
| Alumna: | Ana Real Tovar |
| Consultor/a: | Óscar Baltà Fabró |
| Fecha de entrega: | 31 de mayo de 2019 |

1. Mostrar inicialmente el contenido del archivo hosts proporcionado por pantalla.

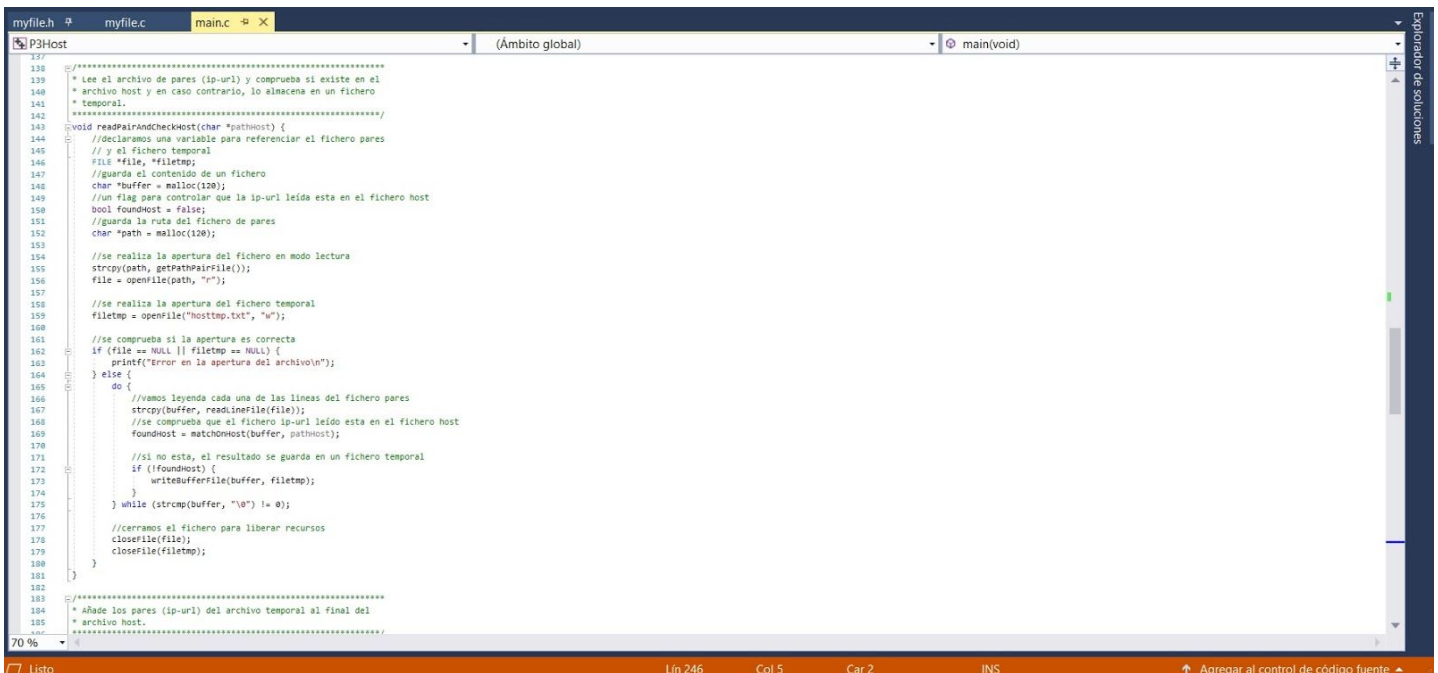
```
C:\Users\jesus\Desktop\P3Host\P3Host\Debug\P3Host.exe
-- FICHERO HOSTS --
Introduzca el nombre del fichero: hosts.txt
Quieres especificar una ruta absoluta (s/n)? : n
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97      rhino.acme.com      # source server
#      38.25.63.10      x.acme.com         # x client host
#
# localhost name resolution is handled within DNS itself.
#      127.0.0.1        localhost
#      ::1              localhost

127.0.0.1 localhost
::1 localhost

-- FICHERO IP-URLS --
Introduzca el nombre del fichero:
```

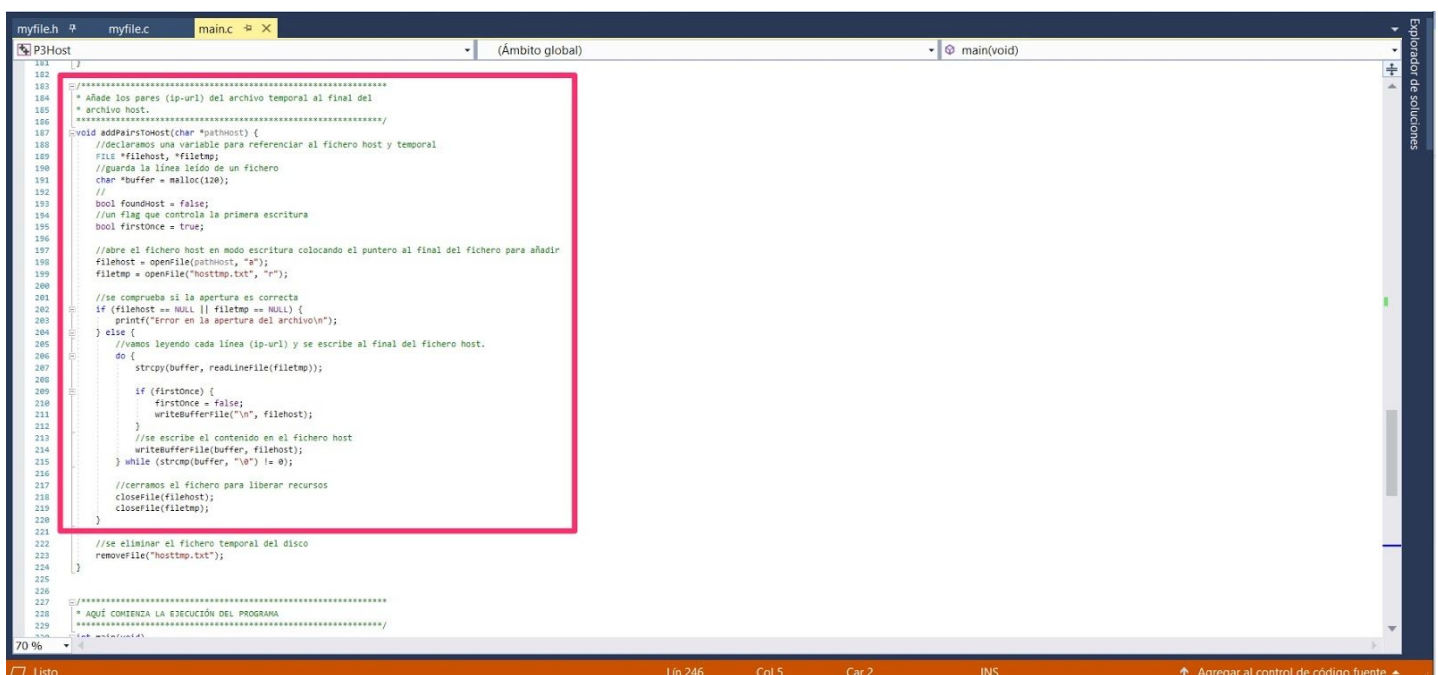
```
myfile.h  myfile.c  main.c  (Ámbito global)  main(void)
42
43
44  * Muestra por pantalla el contenido de un fichero ubicado en
45  * la variable path.
46  */
47  void showFile(char *path) {
48      //declaramos una variable para referenciar el fichero
49      FILE *file;
50      //aquí se guarda cada una de las líneas leídas del fichero
51      char *buffer = malloc(120);
52
53      //se realiza la apertura del fichero en modo lectura
54      file = fopen(path, "r");
55
56      //se comprueba si la apertura es correcta
57      if (file == NULL) {
58          printf("Error en la apertura del archivo\n");
59      } else {
60          //vamos leyendo línea a línea hasta que nos indica que es una cadena vacía
61          do {
62              strcpy(buffer, readline(file));
63              //se muestra el contenido de la línea por pantalla
64              printf("%s", buffer);
65          } while (strcmp(buffer, "\0") != 0);
66
67          //cerramos el fichero para liberar recursos
68          fclose(file);
69      }
70  }
71
72
73  * Comprueba si la cadena introducida por el usuario existe
74  * dentro del fichero hosts.
75  */
```

2. Leer el archivo proporcionado inicialmente con los pared ip url y chequear.



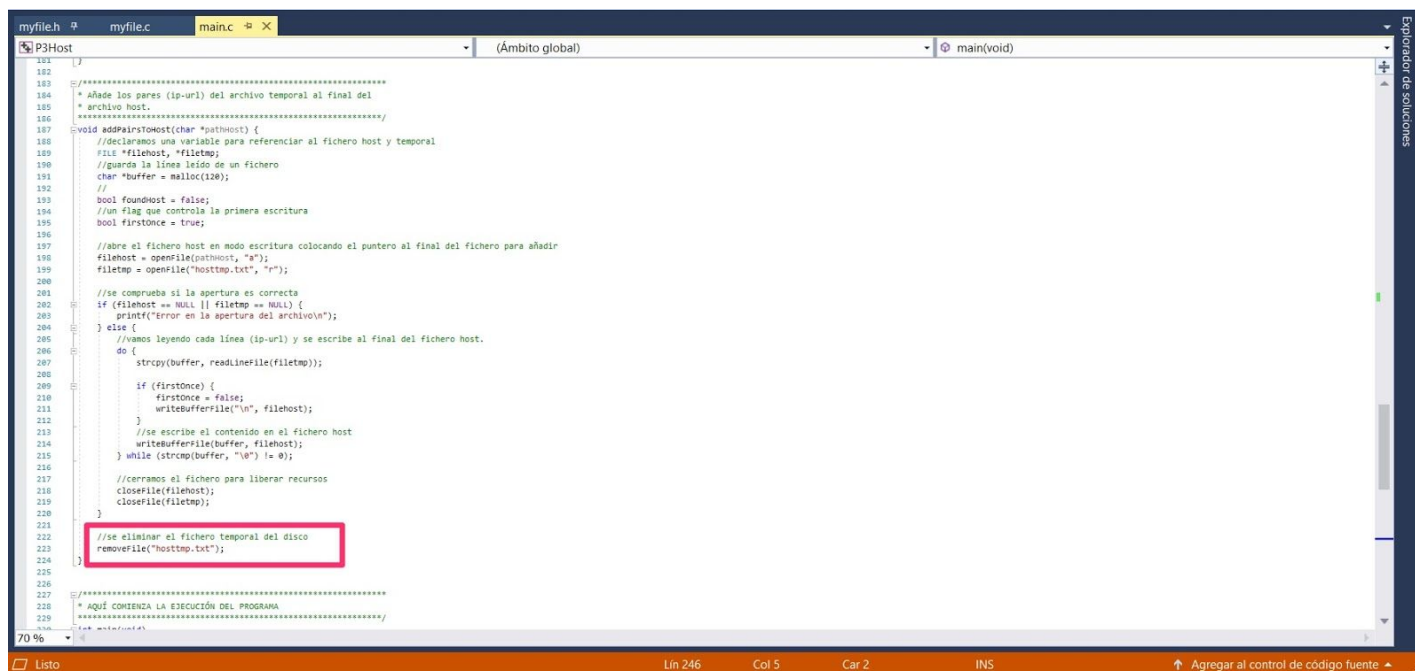
```
139 //*****  
140 // Lee el archivo de pares (ip-url) y comprueba si existe en el  
141 // archivo host y en caso contrario, lo almacena en un fichero  
142 // temporal.  
143 //*****  
144 void readPairAndCheckHost(char *pathhost) {  
145     //Declaramos una variable para referenciar el fichero pares  
146     // y el fichero temporal  
147     FILE *file, *filetmp;  
148     //guarda el contenido de un fichero  
149     char *buffer = malloc(128);  
150     //un flag para controlar que la ip-url leída esta en el fichero host  
151     bool foundHost = false;  
152     //guarda la ruta del fichero de pares  
153     char *path = malloc(128);  
154     //se realiza la apertura del fichero en modo lectura  
155     strcpy(path, getPathPairFile());  
156     file = fopen(path, "r");  
157     //se realiza la apertura del fichero temporal  
158     filetmp = fopen("hosttmp.txt", "w");  
159     //se comprueba si la apertura es correcta  
160     if (file == NULL || filetmp == NULL) {  
161         printf("Error en la apertura del archivo\n");  
162     } else {  
163         do {  
164             //vamos leyendo cada una de las líneas del fichero pares  
165             strcpy(buffer, readLineFile(file));  
166             //se comprueba que el fichero ip-url leído esta en el fichero host  
167             foundHost = matchHost(buffer, pathhost);  
168             //si no está, el resultado se guarda en un fichero temporal  
169             if (!foundHost) {  
170                 writeBufferFile(buffer, filetmp);  
171             }  
172             while (strcmp(buffer, "\n") != 0);  
173         } while (strcmp(buffer, "\n") != 0);  
174     }  
175     //cerramos el fichero para liberar recursos  
176     fclose(file);  
177     fclose(filetmp);  
178 }  
179  
180 //*****  
181 // Añade los pares (ip-url) del archivo temporal al final del  
182 // archivo host.  
183 //*****  
184 void addPairToHost(char *pathhost) {  
185     //Declaramos una variable para referenciar al fichero host y temporal  
186     FILE *filehost, *filetmp;  
187     //guarda la línea leída de un fichero  
188     char *buffer = malloc(128);  
189     //un flag que controla la primera escritura  
190     bool firstOnce = true;  
191     //Abre el fichero host en modo escritura colocando el puntero al final del fichero para añadir  
192     filehost = fopen(pathhost, "a");  
193     filetmp = fopen("hosttmp.txt", "r");  
194     //se comprueba si la apertura es correcta  
195     if (filehost == NULL || filetmp == NULL) {  
196         printf("Error en la apertura del archivo\n");  
197     } else {  
198         //vamos leyendo cada línea (ip-url) y se escribe al final del fichero host.  
199         do {  
200             strcpy(buffer, readLineFile(filetmp));  
201             if (firstOnce) {  
202                 firstOnce = false;  
203                 writeBufferFile("\n", filehost);  
204             }  
205             //se escribe el contenido en el fichero host  
206             writeBufferFile(buffer, filehost);  
207             while (strcmp(buffer, "\n") != 0);  
208         } while (strcmp(buffer, "\n") != 0);  
209     }  
210     //cerramos el fichero para liberar recursos  
211     fclose(filehost);  
212     fclose(filetmp);  
213 }  
214  
215 //se elimina el fichero temporal del disco  
216 removeFile("hosttmp.txt");  
217 }  
218  
219 //*****  
220 // AQUÍ COMIENZA LA EJECUCIÓN DEL PROGRAMA  
221 //*****  
222 int main(void) {  
223     //se llama a la función readPairAndCheckHost  
224     readPairAndCheckHost(pathhost);  
225     //se llama a la función addPairToHost  
226     addPairToHost(pathhost);  
227     return 0;  
228 }
```

3. Abrir el archivo temporal creado en el punto anterior una vez esté lleno, para introducir su contenido en el archivo host proporcionado a modificar.



```
184 void addPairToHost(char *pathhost) {  
185     //Declaramos una variable para referenciar al fichero host y temporal  
186     FILE *filehost, *filetmp;  
187     //guarda la línea leída de un fichero  
188     char *buffer = malloc(128);  
189     //un flag que controla la primera escritura  
190     bool firstOnce = true;  
191     //Abre el fichero host en modo escritura colocando el puntero al final del fichero para añadir  
192     filehost = fopen(pathhost, "a");  
193     filetmp = fopen("hosttmp.txt", "r");  
194     //se comprueba si la apertura es correcta  
195     if (filehost == NULL || filetmp == NULL) {  
196         printf("Error en la apertura del archivo\n");  
197     } else {  
198         //vamos leyendo cada línea (ip-url) y se escribe al final del fichero host.  
199         do {  
200             strcpy(buffer, readLineFile(filetmp));  
201             if (firstOnce) {  
202                 firstOnce = false;  
203                 writeBufferFile("\n", filehost);  
204             }  
205             //se escribe el contenido en el fichero host  
206             writeBufferFile(buffer, filehost);  
207             while (strcmp(buffer, "\n") != 0);  
208         } while (strcmp(buffer, "\n") != 0);  
209     }  
210     //cerramos el fichero para liberar recursos  
211     fclose(filehost);  
212     fclose(filetmp);  
213 }  
214  
215 //se elimina el fichero temporal del disco  
216 removeFile("hosttmp.txt");  
217 }  
218  
219 //*****  
220 // AQUÍ COMIENZA LA EJECUCIÓN DEL PROGRAMA  
221 //*****  
222 int main(void) {  
223     //se llama a la función readPairAndCheckHost  
224     readPairAndCheckHost(pathhost);  
225     //se llama a la función addPairToHost  
226     addPairToHost(pathhost);  
227     return 0;  
228 }
```

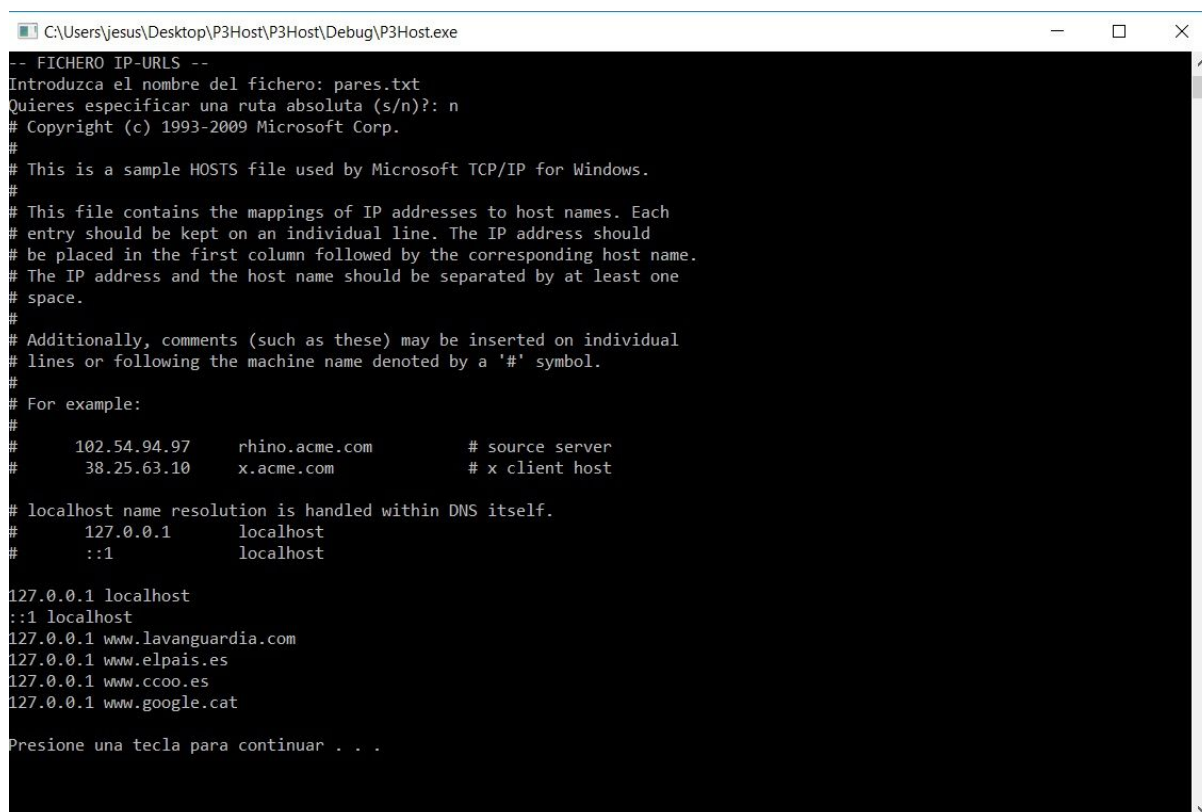
4. Eliminar el archivo temporal una vez ha sido usado.



The screenshot shows a C code editor with a file named `myfile.c`. The code is in the `main` function. It defines a `FILE` pointer `filehost` and a `FILE` pointer `filetmp`. It opens `filehost` in append mode and `filetmp` in write mode. It then reads lines from `filehost` and writes them to `filetmp`. After the loop, it closes both files. A red box highlights the line `removefile("hosttmp.txt");`, which is used to delete the temporary file.

```
101 }
102
103 //*****
104 * Añade los pares (ip-url) del archivo temporal al final del
105 * archivo host.
106 //*****
107 void addpairsToHost(char *pathHost) {
108     //declaramos una variable para referenciar al fichero host y temporal
109     FILE *filehost, *filetmp;
110     //guarda la línea leído de un fichero
111     char *buffer = malloc(128);
112     //
113     bool foundHost = false;
114     //un flag que controla la primera escritura
115     bool firstOnce = true;
116
117     //abre el fichero host en modo escritura colocando el puntero al final del fichero para añadir
118     filehost = fopen(pathHost, "a");
119     filetmp = fopen("hosttmp.txt", "w");
120
121     //se comprueba si la apertura es correcta
122     if (filehost == NULL || filetmp == NULL) {
123         printf("Error en la apertura del archivo\n");
124     } else {
125         //vamos leyendo cada línea (ip-url) y se escribe al final del fichero host.
126         do {
127             strcpy(buffer, readline(filetmp));
128
129             if (firstOnce) {
130                 firstOnce = false;
131                 writebuffer(filetmp, filehost);
132             }
133             //se escribe el contenido en el fichero host
134             writebuffer(filetmp, filehost);
135             while (strcmp(buffer, "\0") != 0);
136
137             //cerramos el fichero para liberar recursos
138             fclose(filehost);
139             fclose(filetmp);
140
141         } while (1);
142
143     }
144
145     //se eliminar el fichero temporal del disco
146     removefile("hosttmp.txt");
147
148     //*****
149     * AQUÍ COMIENZA LA EJECUCIÓN DEL PROGRAMA
150     //*****
151 }
```

5. Mostrar el contenido del archivo hosts modificado por pantalla.



The screenshot shows a command prompt window titled `C:\Users\jesus\Desktop\P3Host\P3Host\Debug\P3Host.exe`. It displays the contents of a `hosts` file, which is a sample file used by Microsoft TCP/IP for Windows. The file contains mappings of IP addresses to host names. The output is as follows:

```
-- FICHERO IP-URLS --
Introduzca el nombre del fichero: pares.txt
Quieres especificar una ruta absoluta (s/n)? n
# Copyright (c) 1993-2009 Microsoft Corp.

# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.

# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.

# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.

# For example:
#
#      102.54.94.97      rhino.acme.com      # source server
#      38.25.63.10      x.acme.com         # x client host

# localhost name resolution is handled within DNS itself.
#
#      127.0.0.1        localhost
#      ::1              localhost

127.0.0.1 localhost
::1 localhost
127.0.0.1 www.lavanguardia.com
127.0.0.1 www.elpais.es
127.0.0.1 www.ccoo.es
127.0.0.1 www.google.cat

Presione una tecla para continuar . . .
```