(P) ICX0_P4 Programa comandos personalizados para el sistema operativo

Código del proyecto: ICX0_P4

Producto 2 - Adquiriendo las destrezas básicas

Alumna:	Ana Real Tovar
Consultor/a:	Óscar Baltà Fabró
Fecha de entrega:	23 de mayo de 2019

1. Mostrar inicialmente un menú

```
7
8  /*************************************************************
9   * Muestra el menú al usuario y solicita una opción
10  *************************************************************/
11 int menu() {
12      //guarda la opción seleccionada por el usuario
13      int option = NULL;
14
15      //se muestra el menú principal con todas las opciones del programa
16      printf("**********************************************************************\n");
17      printf("**                                                                  **\n");
18      printf("**                      MENU PRINCIPAL                              **\n");
19      printf("**                                                                  **\n");
20      printf("** 1. Mostrar el contenido de un archivo                           **\n");
21      printf("** 2. Guardar un archivo                                           **\n");
22      printf("** 3. Chequear archivo                                             **\n");
23      printf("** 4. Salir                                                        **\n");
24      printf("**                                                                  **\n");
25      printf("**                                                                  **\n");
26      printf("**********************************************************************\n");
27      printf("Qu\202 opci\242n desea realizar?: ");
28
29      //se guarda la opción introducida por teclado
30      fflush(stdin);
31      scanf("%d", &option);
32
33      //se comprueba que la opción es correcta, en caso contrario, se solicita de nueva una opción
34      while (option <= 0 || option > 4) {
35          printf("Opci\242n incorrecta, introduzca de nuevo una opci\242n: ");
36          fflush(stdin);
37          scanf("%d", &option);
38      }
39
40      return option;
41 }
```

2. Diseñar una función que muestre el contenido de un archivo

```
/*************************************************************
* Muestra por pantalla el contenido de un fichero
*************************************************************/
void showContent() {
    //declaramos una variable para referenciar el descriptor del fichero
    int fd;
    //guarda la ruta del fichero y el caracter leido del fichero
    char pathfile[120], c = NULL;

    //se guarda en la variable 'pathfile' la ruta del fichero
    setPathFile(pathfile);

    //se realiza la apertura del fichero en modo lectura
    fd = openFile(pathfile, 'r');

    //se comprueba si la apertura es correcta
    if (fd == -1) {
        printf("Error en la apertura del archivo\n");
    } else {
        //vamos leyendo cada uno de los caracterés del fichero cuyo contenido lo mostramos por pantalla.
        while (readCharFile(fd, &c) > 0) {
            printf("%c", c);
        }

        //cerramos el fichero para liberar recursos
        if (closeFile(fd) == -1) {
            printf("Error en el cierre del fichero\n");
        }
    }
}
```

3. Diseñar una función que implemente el guardado como de un archivo

```
/*************************************************************
* Realiza una copia de un fichero y se guarda en la ruta
* indicada por el usuario.
*************************************************************/
void saveAs() {
    //declaramos dos variables que hacen referencia al fichero origen y destino
    int fdFrom, fdTo;
    //declaramos dos varaibles para guardar las ruta de los fichero asi como el carácter leido del fichero origen
    char pathfileFrom[120], pathfileTo[120], c = NULL;

    //se guarda la ruta del fichero origen
    printf("-- FICHERO ORIGEN --\n");
    setPathFile(pathfileFrom);

    //se guarda la ruta del fichero destino
    printf("\n-- FICHERO DESTINO --\n");
    setPathFile(pathfileTo);

    //se realiza la apertura del fichero origen en modo lectura
    fdFrom = openFile(pathfileFrom, 'r');

    //se realiza la apertura del fichero destino en modo escritura
    fdTo = openFile(pathfileTo, 'w');

    //se comprueba si la apertura de los ficheros es correcta
    if (fdFrom == -1 || fdTo == -1) {
        if (fdFrom == -1) {
            printf("Error en la apertura del archivo origen\n");
        } else {
            printf("Error en la apertura del archivo destino\n");
        }
    } else {
        //se van leyendo caracteres y en cada interación se guarda el carácter en el fichero
        while (readCharFile(fdFrom, &c) > 0) {
            writeCharFile(fdTo, &c);
        }

        //cerramos el fichero para liberar recursos
        if (closeFile(fdFrom) == -1 || closeFile(fdTo) == -1) {
            printf("Error en el cierre del fichero\n");
        } else {
            printf("\nEl archivo se ha guardado correctamente \n");
        }
    }
}
```

4. Diseñar una función que chequee el contenido de un archivo

```
/*************************************************************
 * Comprueba si la cadena introducida por el usuario existe
 * dentro del fichero.
 *************************************************************/
void checkContent() {
    //con esta variable se controla que la cadena existe dentro del fichero
    bool fullmatch;
    //declaramos una variable que hace referencia al descriptor del fichero
    //y otras dos variables para controlar el matcheo y la longitud máximo de
    //la cadena introducida por el usuario
    int fd, posMatch, max;
    //guarda la ruta del fichero y el caracter leído del fichero
    char pathfile[120], c = NULL, inputData[120];

    //se guarda en la variable 'pathfile' la ruta del fichero
    setPathFile(pathfile);

    //se realiza la apertura del fichero en modo lectura
    fd = openFile(pathfile, 'r');

    //se guarda en la variable inputData el texto solicitado al usuario por teclado
    printf("Introduza una cadena para comprobar si esta en el fichero: ");
    scanf("%s", inputData);

    //se comprueba si la apertura es correcta
    if (fd == -1) {
        printf("Error en la apertura del archivo\n");
    }
    else {
        posMatch = 0;
        fullmatch = false;
        max = strlen(inputData);

        while (readCharFile(fd, &c) > 0 && !fullmatch) {
            //comprueba si el caracter leído coincide con el siguiente carácter del texto introducido por el usuario
            if (c == inputData[posMatch]) {
                //avanzamos la posición que hace referencia al matching entre el texto del fichero y el del usuario
                posMatch++;
            }
            else {
                //reiniciamos la comparación
                posMatch = 0;
                if (c == inputData[posMatch]) {
                    posMatch++;
                }
            }

            //si la posición del matching tiene el tamaño del texto, hemos llegado al final.
            if (posMatch == max) {
                fullmatch = true;
            }
        }

        if (fullmatch) {
            printf("El texto introducido se encuentra en el fichero\n");
        }
        else {
            printf("No se ha encontrado el texto introducido en el fichero\n");
        }

        //cerramos el fichero para liberar recursos
        if (closeFile(fd) == -1) {
            printf("Error en el cierre del fichero\n");
        }
    }
```

]

5. Realizar la aplicación modularizada

Para llevar a cabo la modularización de la aplicación, se encapsulado la lógica relacionada con los ficheros en una librería llamada myfile donde podremos abrir, leer, escribir, cerrar, buscar y solicitar la ruta de un fichero. Esta librería nos servirá de apoyo para el desarrollo de nuestra aplicación cuyas lógica está dividida en funciones.

```c
//librerías necesarias para trabajar con las funciones a bajo nivel
#include "myfile.h"

#include <sys/stat.h>
#include <fcntl.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

void setPathFile(char *path) {
    //declaramos las variables para guardar la ruta y el nombre del fichero
    char namefile[20], abspath[120];
    //en esta variable se guarda la preferencia del usuario sobre la ubicación del fichero (s: ruta absoluta, otro valor: ruta relativa)
    char isAbs;

    //se pregunta al usuario por el nombre y ubicación del fichero
    printf("Introduzca el nombre del fichero: ");
    fflush(stdin);
    scanf(" %s", namefile);
    printf("Quieres especificar una ruta absoluta (s/n)?: ");
    fflush(stdin);
    scanf(" %c", &isAbs);

    if (isAbs == 's') {
        //si el usuario indica una ruta absoluta, se solicita la ubicación completa concatenándolo con el nombre del fichero
        printf("Introduzca la ruta completa del fichero: ");
        fflush(stdin);
        scanf("%s", abspath);

        strcat(abspath, "\\");
        strcat(abspath, namefile);
        strcpy(path, abspath);
    } else {
        //si la ruta es relativa, se devuelve directamente el nombre del fichero
        strcpy(path, namefile);
    }
}


int openFile(char *path, char mode) {
    if (mode == 'r') {
        //se abre el archivo en modo lectura
        return _open(path, O_RDONLY);
    } else if (mode == 'w') {
        //se abre el archivo en modo lectura/escritura y se crea en caso de no existir
        return _open(path, O_CREAT | O_RDWR);
    } else {
        return -1;
    }
}

int closeFile(int fd) {
    //se usa la función de bajo nivel para cerrar el fichero
    return _close(fd);
}


int readCharFile(int fd, char *c) {
    //se usa la función de bajo nivel para leer un fichero
    return _read(fd, c, 1);
}


void writeCharFile(int fd, char *c) {
    //se usa la función de bajo nivel para escribir un carácter
    return _write(fd, c, 1);
}

```

```
1
2   /*****************************************************************
3   * Se solicita al usuario la ubicación del fichero.
4   *
5   * Recibe un parámetro de entrada:
6   * - path: se guarda la ruta absoluta/completa junto al nombre
7   * del fichero.
8   *****************************************************************/
9   void setPathFile(char *path);
10
11  /*****************************************************************
12  * Realiza la apertura del fichero.
13  *
14  * Recibe dos parámetros de entrada:
15  * - path: contiene la ubicación abosoluta/relativa del fichero
16  * - mode: modo apertura del fichero (r: sólo lectura, w: escritura).
17  *
18  * Devuelve un entero que hace referencia al descriptor del fichero,
19  * si el valor es -1 significa que hay un error en la apertura.
20  *****************************************************************/
21  int openFile(char *path, char mode);
22
23  /*****************************************************************
24  * Realiza el cierre del fichero.
25  *
26  * Recibe un parámetro de entrada:
27  * - fd: es la referencia al descriptor del fichero
28
29  * Devuelve un entero con el resultado de la operación,
30  * en caso de error, su valor será -1.
31  *****************************************************************/
32  int closeFile(int fd);
33
34
35  /*****************************************************************
36  * Leer el siguiente carácter del fichero
37  *
38  * Recibe dos parámetros de entrada:
39  * - fd: es la referencia al descriptor del fichero
40  * - c: es una variable donde se almacena el carácter leído
41  *
42  * Devuelve un entero que indica el estado de la operación (1: ok,
43  * 0: fin de fichero)
44  *****************************************************************/
45  int readCharFile(int fd, char *c);
46
47
48  /*****************************************************************
49  * Escribe un carácter en la siguiente posición del curso del
50  * fichero
51  *
52  * Recibe dos parámetros de entrada:
53  * - fd: es la referencia al descriptor del fichero
54  * - c: contiene el caracter a copiar en el fichero
55  *
56  *****************************************************************/
57  void writeCharFile(int fd, char *c);
58
```