Fundamentos de Software de Comunicaciones
E.T.S.I. Telecomunicación

Ficheros y directorios

1. Implementa un programa que imprima el contenido de un fichero por pantalla. Obviamente, lo lógico es que se trate de un archivo de texto.

2. Una de las principales misiones de un fichero es la de guardar contenidos y configuraciones de un programa para recuperarlos la siguiente vez que se ejecute, y así no se pierden. Se pide implementar dos programas:
   - **`GuardaDatos.c`**:
     i. Toma como argumento el nombre de un fichero nuevo donde almacenar una serie de datos.
     ii. Guarde en el fichero el contenido de las siguientes variables:

```
struct Estructura{
    int a;
    float b;
    char c;
};

int x = 7;
int array_enteros[4] = {0x00,0x01,0x02,0x03};
struct Estructura est;
est.a = 1; est.b = 2.0; est.c = '3';
```

     Nota 1: Los datos se deben guardar evitando posibles bytes de relleno que aparecen en la estructura.
     Nota 2: Por ejemplo, guardar 'x' supone llevar a disco los 4 bytes que ocupa dicha variable en memoria.
     Nota 2: Pruebe a visualizar el contenido del fichero abriéndolo con un editor de texto (o el propio VSCode).
   - **`LeeDatos.c`**:
     i. Toma como argumento el nombre de un fichero nuevo donde están almacenados los datos anteriores.
     ii. Lea del fichero el valor de las variables y las muestre por pantalla.

3. Si no lo ha hecho así, modifique lo programas anteriores para que, evitando también los bytes de relleno de la estructura, escriba/lea en/desde el fichero toda la información con una única llamada a **write/read**.

4. Implementa una función

   ```
   int copiaFichero(char *forigen, char *fdestino)
   ```

   que copie un fichero a otro, suponiendo que los nombres de ambos se pasan como relativos al directorio en el que se ejecuta el programa. Utiliza esta

función en un programa, llamado cp_fsc, similar al comando "cp" del sistema operativo, que se ejecute como:

`cp_fsc <nombre_forigen> <nombre_fdestino>`

5. Codifica un programa **`ls_fsc`** que acepte la opción -l y que muestre el directorio en curso empleando el ejecutable original ls del sistema y la función **`system()`**. (Consulta el manual para el uso de la función **`system(...)`**).

6. Implementa una función **`int estadof(char* f)`** que muestre por pantalla la información disponible de un fichero a partir de su nombre. La información debe incluir si el fichero es un fichero regular o es un directorio, así como su tamaño en bytes. La función devuelve 0 en caso de éxito, y -1 en caso de que haya algún error.
   - Notas
     i. Utiliza la llamada al sistema **`stat(...)`**.
     ii. Las macros para consultar si un fichero es regular o es un directorio, a partir del **`struct stat {...}`** son **`S_ISDIR()`** y **`S_ISREG()`**, y tienen como argumento el campo **`st_mode`** del **`struct stat`**.

7. Implementa la función **`int listdir(char *d)`** que muestre un directorio empleando la función *estadof* del ejercicio anterior.
   - Implementa la función usando funciones de la biblioteca de C (es decir, usando **`opendir`** y **`readdir`**).
   - Un esqueleto de la función sería:

```c
int listdir(char* directorio){

    DIR* dir = opendir(directorio);
    if (!dir){
        printf("error al intentar abrir directorio\n");
        return -1;
    }

    struct dirent* entry;
    printf("Contenidos del directorio:\n");
    while ((entry = readdir(dir)) != NULL) {//va leyendo entradas a medida que se llama
        printf("Nombre de fichero: %s\n", entry->d_name);
        //utiliza esta funcion de otro ejercicio:
        //estadof(entry->d_name);
        printf("\n\n");
    }

    if (closedir(dir) == -1){
        printf("problema al cerrar directorio\n");
        return -1;
    }

    return 0;
}
```