Tema 1

Estructuras de Datos y Algoritmos

Ficheros de texto
Algunas funciones para manipular cadenas de caracteres

Prof. Mary Luz Mouronte López
Prof. Javier Sánchez Soriano

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Universidad Francisco de Vitoria
UFV Madrid

Ficheros de texto (en C)

Grado en Ingeniería Informática
Escuela Politécnica Superior

Ficheros

- Los ficheros o archivos informáticos consisten en una ristra de bits almacenados en un soporte informático.

- Se identifican por su **nombre**, **extensión** (define el tipo de datos que contiene) así como por la carpeta o directorio que lo contiene (la **ruta**).

- Son los equivalentes digitales de los archivos escritos en expedientes, tarjetas, libretas, papel…

Ejemplos de extensiones habituales

- **ISO**: Imagen de disco. Este archivo contiene una copia de un CD/DVD/… para su grabación en otro CD/DVD/….
- **JPG / GIF /PNG**: formatos de imágenes.
- **AVI / MPEG / MP4**: formatos de vídeo.
- **MP3, WAV**: formatos de audio.
- **PDF**: formato utilizado para la difusión de archivos de texto, ya que una vez creado no puede ser modificado.
- **DOCX / PPT / XLSX** / …: Ficheros con documentos, presentaciones, hojas de cálculo, … de MS Office.
- **TXT**: creación y edición de archivos de texto plano.
- **EXE**: Fichero ejecutable.
- **ZIP / RAR / TAR** …: Archivos comprimidos que se utilizan para almacenar la información en poco espacio o difundirla a través de Internet.
- …

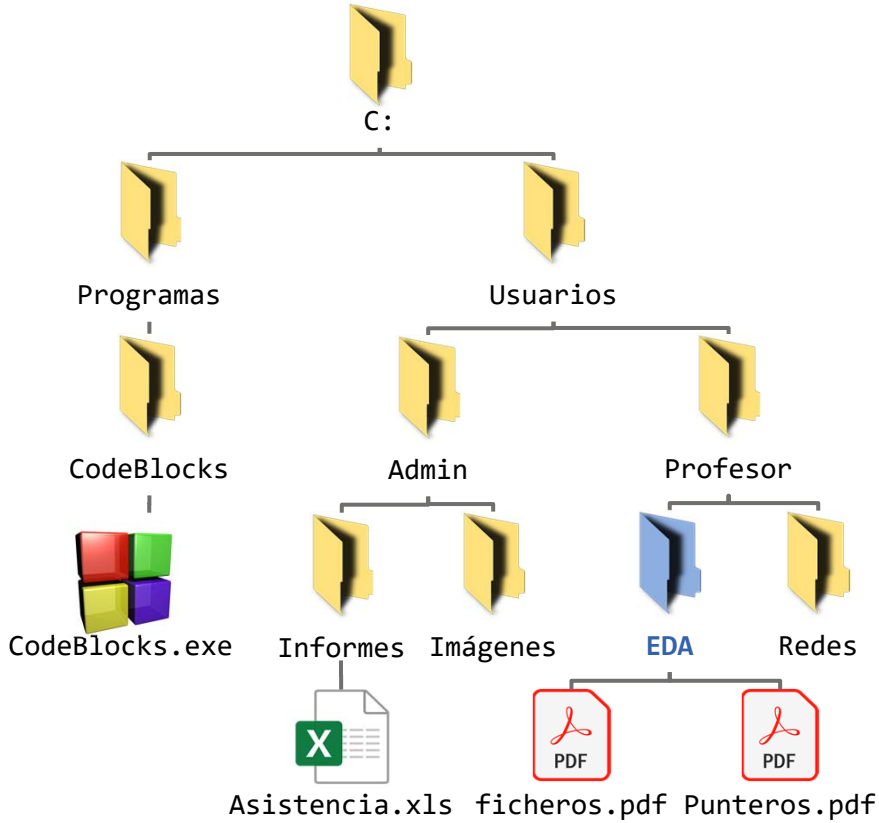
Tipos de rutas

- Señalan la ubicación de un archivo/directorio. Hay 2 formatos, con diferente nivel de detalle:
  1. **Rutas absolutas:** Desde directorio raíz del sistema de archivos. Describe todo el detalle de la ubicación. Ejemplos:
     - `/home/datos/notas.xls` => `/` es la raíz en sistemas Unix/Linux/MacOS…
     - `C:/Usuarios/Profesor/EDA/Notas.xlsx`
  2. **Rutas relativas:** A partir de la posición actual del sistema operativo en el sistema de archivos. Ejemplos:
     - `EDA/notas.xlsx` => `notas.xlsx` está dentro de la carpeta `EDA` desde ubicación actual
     - `Notas/FundProgramacion.xls`

- **IMPORTANTE**: En programación y como norma general, la mejor opción son las **rutas relativas** al ejecutable, lo que facilita al redistribución de los programas y el correcto acceso a los ficheros de datos por los programas.

Ejemplos de rutas

- RUTAS ABSOLUTAS
  - `C:\Usuarios\Profesor\Redes`
  - `C:\Programas\CodeBlocks\Codeblocks.exe`



- RUTAS RELATIVAS (EDA)

Ruta relativa	Ruta absoluta equivalente
`..`	`C:\Usuarios\Profesor`
`..\..`	`C:\Usuarios`
`..\..\Admin`	`C:\Usuarios\Admin`
`.`	`C:\Usuarios\Profesor\EDA`
`.\..\Redes`	`C:\Usuarios\Profesor\Redes`

- `..\..\.\Admin\..\Profesor \.\Redes  =>  C:\Usuarios\Profesor\Redes`

¿Qué es un fichero?

- Dispositivo de entrada-salida cuyo objetivo es almacenar o suministrar información. Los archivos* empleados serán secuenciales.

- Operaciones con ficheros:
  - Apertura de ficheros: **`fopen()`**
  - Cierre de ficheros: **`fclose()`**
  - Lectura de ficheros
  - Escritura en ficheros.

- Se requiere determinar un área de buffer. Elemento que almacena la información mientras se está transfiriendo hacia o desde el archivo.

```
FILE *punt_fichero;
```

** Se utilizan indistintamente los sinónimos archivo / fichero*

¿Qué es un fichero?

- Apertura de un fichero: **`fopen()`**. Prototipo:

```
FILE *fopen (char * nombre_completo_archivo, char *modo);
```

  Modos:
  - **`"r"`** : Abre en modo de solo lectura.
  - **`"w"`** : Abre para escritura (si no existe lo crea, si existe lo destruye).
  - **`"a"`** : Abre para agregar información (si no existe lo crea).
  - **`"r+"`** : Abre para lectura/escritura (comienza al principio del archivo).
  - **`"w+"`** : Abre para lectura/escritura (lo sobrescribe si este ya existe o lo crea si no).
  - **`"a+"`** : Abre para lectura/escritura (se sitúa al final del archivo).
  - **`"b"`** : Para usar ficheros binarios, añadiéndose a los anteriores modos entre la letra y el símbolo +, dependiendo del modo deseado.

- Cierre de un fichero: **`fclose()`**. Prototipo:

```
int fclose(FILE *puntero_al_archivo);
```

Ficheros: apertura y cierre

```
#include <stdio.h>
int main(void)
{
  FILE *pfich;

  pfich = fopen("info.txt","w");

  if (pfich == NULL) {
      printf("\n Error al abrir el fichero");
      exit(1);
  }
  /* Se emplea el fichero para escritura */
  ...
  /* Al acabar de usar el archivo */
  if (fclose(pfich) != 0) {
      printf("\n Error al cerrar el fichero ");
      exit(1);
  }
}
```

Se declara un puntero a fichero

Se abre en modo escritura (w) el fichero “info.txt”

Se verifica la correcta apertura del fichero

Cierre del fichero y verificación de acción correctamente efectuada.

Archivos: lectura y escritura

- Clases de archivos: de texto y binarios
  - **Archivo de texto**: También conocidos como texto plano dado que sólo contienen caracteres y carecen de formato. Existen diferentes opciones de codificación en función del idioma (juego de caracteres, usual UTF-8). **Legibles por las personas.**
  - **Archivo binario**. Son ficheros que contienen datos de todo tipo, codificados byte a byte. Usados para ser procesados por máquinas, aunque en algunos binarios existen secciones que pueden interpretarse como texto. Binario que solo contiene información de tipo textual sin información sobre el formato del mismo, se dice que es un archivo de texto. De forma general **no son legibles por las personas.**

- Las funciones para efectuar la lectura y escritura de archivos son diferentes según la clase de ficheros:
  - Texto: `fprintf`, `fscanf`, `getc`, `putc`, `fgets`, `fputs`
  - Binarios: `fwrite`, `fread`, `fseek` (No se verán en la asignatura)

Archivos: lectura y escritura

- **FICHEROS de TEXTO**
  - **`fprintf()`:** Como `printf()` pero indicando en que fichero se escribe
    - Valor de retorno:
      - **Éxito**: devuelve el número de bytes que ha escrito
      - **Fracaso**: devuelve `EOF`
  - **`fscanf()`:** Como `scanf()` pero indicando de que fichero se lee
    - Valor de retorno (cuando sólo se lee un valor):
      - **Éxito**: devuelve el valor 1 si ha leído de modo correcto
      - **Fracaso**: devuelve 0 si ha ocurrido un error en la lectura por una mala especificación de formato.
      - `EOF`: si no se ha leído nada porque se ha llegado al final del archivo

fscanf() , fprintf()

```
#include <stdio.h>
int main(void) {
    int ctrl;        /* Valor devuelto por fscanf */
    FILE *pfich;     /* Puntero al archivo */
    double dato;     /* dato leído del fichero */
    double sum_tot; /* Suma de los datos */

    pfich = fopen("info.txt", "r+");
    if (pfich == NULL) {
        printf("Error: No se puede abrir el fichero \"info.txt\"\n");
    }
    else {
        sum_tot = 0; /* Inic. de suma antes de entrar en el bucle */
        do {
            ctrl = fscanf(pfich, " %lf ", &dato);
            if (ctrl==1) {
                sum_tot+=dato;
            }
        }
        while(ctrl==1);

        printf("El valor de la suma es: %f\n", sum_tot);
        fprintf(pfich,"\nEl valor de la suma es: %f", sum_tot);
        if( fclose(pfich) != 0) {
            printf("Error al cerrar el fichero\n");
        }
    }
    return 0;
}
```

Programa que **lee el archivo** “info.txt" que contiene **números en punto flotante** y calcula su **suma**

Carácter: getc()/fgetc(), putc()/fputc()

- **Lectura de un carácter**:
  `int` **`fgetc`**`(FILE *`*`puntero_a_archivo`*`)` (o igual con getc)
  - Lee el siguiente carácter del archivo.
  - Valor de retorno:
    - **Éxito**: El carácter leído
    - **Error** : Si llega al final de fichero u ocurre un error, el valor de retorno es `EOF`.

- **Escritura de un carácter**:
  `int` **`fputc`**`(int` *`car`*`, FILE *` *`punt_a_arch`*`)` (o igual con putc)
  - Escribe el carácter *car* en el archivo al que apunta *`punt_a_arch`*
  - Valor de retorno:
    - **Éxito**: El propio carácter escrito
    - **Error**: `EOF`

Cadena de caracteres: fgets(), fputs()

- **Lectura de una cadena de caracteres:**
  `char * fgets(char *cadena, int tam_cad, FILE *punt_a_arch)`
  - Almacena en *cadena* añadiendo `'\0'` al final. La lectura termina cuando:
    - Encuentra el `'\n'` (que sí se copia)
    - Encuentra el fin de fichero `EOF` (no se escribe \n)
    - Se han leído (`tam_cad-1`) caracteres
  - Valor de retorno:
    - **Éxito**: Es un puntero a la cadena leída.
    - **Fracaso**: `NULL`, si se llega al final de fichero u ocurre un error

- **Escritura de una cadena de caracteres:**
  `int fputs(const char *cadena, FILE *punt_a_arch)`
  - Escribe *cadena* en archivo apuntado por *`punt_a_arch`* SIN EL `'\0'` DEL FINAL
  - Valor de retorno:
    - **Éxito**: número positivo si se ha escrito la cadena correctamente
    - **Error**: `EOF`

Archivos binarios

NO SE VEN EN LA ASIGNATURA

Universidad Francisco de Vitoria
UFV Madrid

Algunas funciones para manipular cadenas caracteres (en C)

Grado en Ingeniería Informática
Escuela Politécnica Superior

Cadena de caracteres

Universidad Francisco de Vitoria UFV Madrid

– Las funciones printf(), scanf(), gets(), puts() y fgets() permiten trabajar con cadenas **sin necesidad de utilizar bucles**

```
#include<stdio.h>
int main(void)
{
   char  cad[20];
   puts("Introduce una frase: ");
   gets(cad);
   puts(cad);
   printf("Otra vez: %s", cad);

   printf("\n\nUna palabra: ");
   scanf("%s", cad);
   printf("La palabra es: %s",cad);


   printf("\n\nOtra frase: ");
   scanf(" %[^\n]", cad);
   printf("La frase es: %s",cad);

   printf("\nLectura segura: ");
   fgets(cad, 20, stdin);
   return 0;
}
```

Lectura de una frase

Muestra una frase

Muestra una frase (%s)

¡ ATENCIÓN !
& NO precede al nombre de la cadena

¡ ATENCIÓN !
`scanf("%s", cad);`
sólo lee una **palabra**

PELIGRO:
En la lectura se podría sobrepasar el tamaño

¡ ATENCIÓN !
espacio

Lectura de una frase %[^\n]

LECTURA SEGURA

Funciones para el manejo de cadenas de caracteres: strcpy(), stcat(), strcmp(), strlen()

Hay que incluir el archivo de cabecera **<string.h>**

strcpy(), "*string copy*"

```
strcpy(cad1, cad2);
```

Copia el contenido de cad2 en cad1

Cuidado con sobrepasar el tamaño

strcat(), "*string cat*"

```
strcat( cad1, cad2);
```

Añade la cad2 al final de la cad1, y lo guarda en cad1

strcmp(), "*string compare*"

```
strcmp(cad1, cad2);
```

- Si son iguales, devuelve un cero.
- Si cad1 es mayor que cad2 devuelve un número positivo.
- Si cad1 es menor que cad2 devuelve un número negativo (Valores ASCII)

strlen(), "*string length*"

```
strlen(cad);
```

Devuelve la longitud de cad, sin contar el '\0' final

Funciones para el manejo de cadenas de caracteres: strtok()

- Permite romper una cadena en subcadenas basándose en un conjunto especificado de caracteres de separación. Su prototipo es:

```
char *strtok(char *s1, const char *s2);
```

  - Lee la cadena `s1` como una serie de cero o más símbolos y la cadena `s2` como el conjunto de caracteres que se utilizan como separadores de los símbolos de la cadena `s1`.
  - Los símbolos en la cadena `s1` pueden encontrarse separados por un carácter o más del conjunto de caracteres separadores de la cadena `s2`.
  - La segunda y posteriores llamadas a `strtok()` han de hacerse con el primer argumento a `NULL`.

Funciones para el manejo de cadenas de caracteres: strtok()

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

#define SEPARADOR " primero"

void main()

{

char cadena[] = "Esta cadena se separa usando espacios como delimitador";

char *ptrtoken;

system ("cls");

ptrtoken = strtok( cadena, SEPARADOR); /* primera llamada*/

if (ptrtoken != NULL)
  printf( "%s\n", ptrtoken );

// continua la división en tokens hasta que ptrtoken sea NULL
while((ptrtoken = strtok( NULL, SEPARADOR )) != NULL ) /* sucesivas llamadas */

printf( "%s\n", ptrtoken );
}
```

Salida:

```
Esta
cadena
se
separa
usando
espacios
como
delimitador
```

Funciones para el manejo de cadenas de caracteres: strtok()

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

#define SEPARADOR " primero"

void main()

{

char cadena[] = "Esta cadena se separa usando espacios como delimitador";

char *ptrtoken;

system ("cls");

ptrtoken = strtok( cadena, SEPARADOR); /* primera llamada*/

if (ptrtoken != NULL)
  printf( "%s\n", ptrtoken );

// continua la división en tokens hasta que ptrtoken sea NULL
while((ptrtoken = strtok( NULL, SEPARADOR )) != NULL ) /* sucesivas llamadas */

printf( "%s\n", ptrtoken );
}
```

¿Salida?

Universidad Francisco de Vitoria
UFV Madrid

Anexo: Ficheros de texto (en C#)

Grado en Ingeniería Informática
Escuela Politécnica Superior

Escritura fichero de texto

```
using System;
using System.IO;
namespace EscribirFichTexto_V1 {
  public class CEscribirCars {
    public static void Main(string[] args) {
      StreamWriter sw = null;
      String str;
      try {
        sw = new StreamWriter("docw.txt");  // Crear un flujo hacia fichero doc.txt
        Console.WriteLine("Escriba líneas de texto a almacenar en el fichero.\n" +
                          "Finalice cada línea pulsando la tecla <Entrar>.\n" +
                          "Para finalizar pulse sólo la tecla <Entrar>.\n" );
        // Leer una línea de la entrada estándar
        str = Console.ReadLine();
        while (str.Length != 0) { // Escribir la línea leída en el fichero
           sw.WriteLine(str);
           // Leer la línea siguiente
           str = Console.ReadLine();
        }
      } catch (IOException e) {
           Console.WriteLine("Error: " + e.Message);
      } finally {
           if (sw != null) sw.Close();
      }
    }
  }
}
```

Lectura fichero de texto

```
using System;
using System.IO;
namespace LeerFichTexto_V1 {
  public class CLeerCars {
    public static void Main(string[] args) {
        StreamReader sr = null;
        String str;
        try {
            // Crear un flujo desde el fichero doc.txt
            sr = new StreamReader("doc.txt");
            // Leer del fichero una línea de texto
            str = sr.ReadLine();
            while (str != null) {
                Console.WriteLine(str); // Mostrar la línea leída
                str = sr.ReadLine();    // Leer la línea siguiente
            }
        } catch (IOException e) {
             Console.WriteLine("Error: " + e.Message);
        }
        finally {  // Cerrar el fichero
           if (sr != null) sr.Close();
        }
    }
  }
}
```

Algunos métodos y propiedades de la clase String

Métodos:

- `String[] Split(char[] seps )`

  Ejemplo:
```
string str1 = "hola, ¿que tal?. Bien";
char[] seps = {',', '.'};
string[] cad = str1.Split(seps);
```

- `int CompareTo(String otroString )`

  Devuelve:

  `< 0` si el String que recibe el mensaje es menor que el `otroString`

  `= 0` si el String que recibe el mensaje es igual que el `otroString`

  `> 0` si el String que recibe el mensaje es mayor que el `otroString`

- `static String Concat(String str1 , String str2 )`

  Ejemplo:
```
string str1 = "Ayer ";
string str2 = "llovió";
string str3 = System.String.Concat(str1, str2);
```

Propiedades

- `int Length`

  Ejemplo:
```
string str1 = "La provincia de Santander es muy bonita";
System.Console.WriteLine("Longitud: " + str1.Length);
```

Derechos de Autor

Universidad Francisco de Vitoria
UFV Madrid