

Seminario sobre Ficheros con MATLAB y Octave

Asignatura: Informática

Departamento de Matemáticas e Informática aplicadas a la Ingeniería
Civil y Naval

Escuela de Ingenieros de Caminos, Canales y Puertos

Santiago Higuera

Universidad Politécnica de Madrid

Diciembre 2021



```

010010100100001001010100010100100001001010000100101000010010100001010010000100101
1000101001010101010101000010010101010101010001000101010101010100010001010101010
010011000100010001010001000110001000100010001000100010001000100010001000100010001
0101000010100101001101010000101001010011010100001010010100110101000010100101001
10011010100000010101001101001000000101010011010010000001010011010010000001010
01010010010010010100010100100100100101000101001001001001001001010001010010010010100
1001001010101010100100100101010101011001001001010101010100100100101010101010100
10101010000100010011010101000010001001101010100001000100110101010000100010011
00101000100101001001001010001001001001001001001001001001001001001000100101001001
00010100100001001010000101001000010010100001010010000100101000010100100001001010
1000101001010101010101000010100101010101010001010010101010101000101001010101010
01001100010001010001010011000100010100010100110001000101000101001100010001010001
01010000101001010011010100001010010100110101000010100101001101010000101001010011
100110100100000010101001101001000000101010011010010000001010011010010000001010
010100100100100101000101000100100100100100100100100100100100100100100100100100100
10010010101010101001001001010101010110010010010101010101001001001010101010100

```

La única realidad dentro del ordenador son ceros y unos

Sistemas de almacenamiento

Cintas magnéticas (I)



Sistemas de almacenamiento

Cintas magnéticas (II)

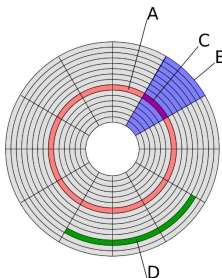


0	1	1	0	1	1	1	0	1	1	1	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Acceso secuencial: para alcanzar una posición de la cinta, hay que recorrer todas las anteriores

Sistemas de almacenamiento

Discos duros



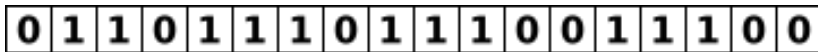
A: Pista

B: Sector geométrico

C: Sector de pista

D: Cluster

XATAKA BASICS

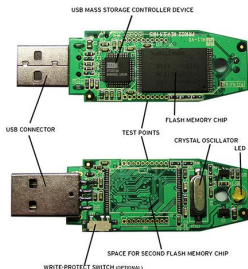


Cada **pista** del disco es una tira de ceros y unos

Acceso aleatorio: el disco gira y el brazo de lectura se mueve => se puede alcanzar una posición sin tener que pasar por todas las anteriores

Sistemas de almacenamiento

Dispositivos de Estado Sólido (SSD)

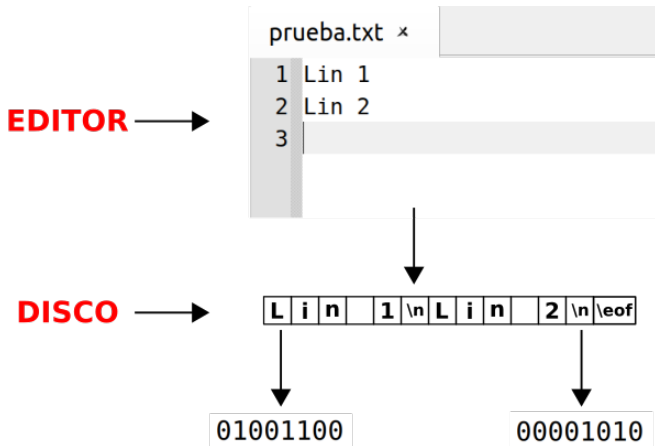


1	1	0	0	1	1	1	0	0
0	1	1	0	1	1	1	0	1
1	1	0	0	1	1	1	0	0
0	1	1	0	1	1	1	0	1

Acceso aleatorio

Sistema de archivos

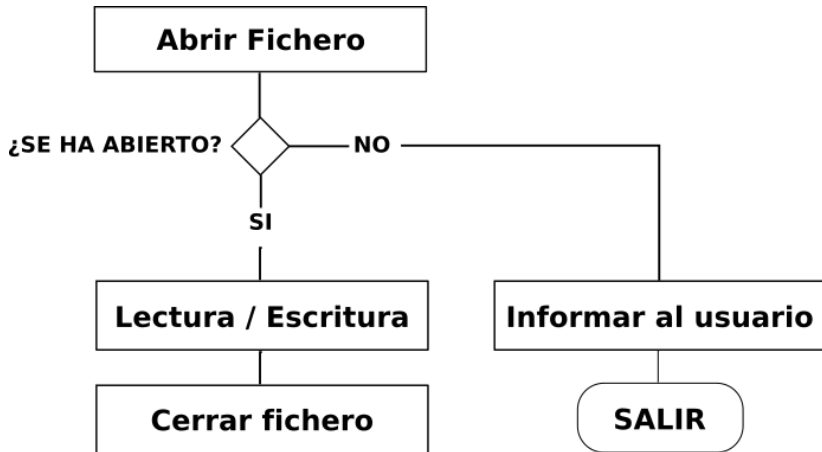
Fichero en editor Vs Fichero en disco



Realmente, en el disco no hay caracteres,
lo que hay son los códigos binarios de cada carácter

Operativa con ficheros

Esquema para programas de lectura o escritura de ficheros



Operativa con ficheros

Apertura del fichero

`fid = fopen(nombre_del_fichero, modo_de_apertura)`

- **nombre_del_fichero:** Nombre del fichero entre comillas o nombre de la variable que guarda el nombre del fichero
- **modo_de_apertura:** Los ficheros se pueden abrir para leer datos o para escribir en ellos
- **fid:** La función *fopen()* devuelve un identificador (*file identifier*) que se usará en todas las operaciones de lectura o escritura para identificar el fichero. Si el identificador devuelto por *fopen()* es **-1**, quiere decir que se ha producido un **error** al intentar abrir el fichero

Operativa con ficheros

Modo de apertura

Los ficheros se abren para leer los datos contenidos en el mismo o para escribir información en ellos. Los modos de apertura recomendados son:

- **'w'**: (*write*) Se crea un fichero nuevo para escribir en él. Si ya existe un fichero con ese nombre, se elimina y se crea uno nuevo vacío
- **'a'**: (*append*) Se abre un fichero existente para añadir nuevos datos, manteniendo lo que ya tenga escrito. Los datos nuevos se añaden después de la última línea. Si el fichero no existe, se crea uno nuevo
- **'r'**: (*read*) El fichero se abre para leer los datos contenidos en él. La lectura comienza al principio de la primera línea del fichero. Si el fichero no existe, se genera un error, y la función *fopen()* devuelve **-1**

Operativa con ficheros

Comprobación de apertura correcta del fichero

- Al intentar abrir un fichero, pueden producirse errores
- Si se produce algún error de apertura, la función *fopen()* devuelve **-1**
- Los errores son más frecuentes cuando se abre el fichero para lectura, en general, porque no se encuentra el fichero (nombre mal escrito, fichero que no existe o que no está en el directorio de trabajo)

Siempre, después de la sentencia *fopen()*, conviene hacer el *checking* de la apertura de la siguiente manera:


```
fid = fopen(nombre, modo)

if fid == -1
    fprintf('Error al abrir fichero\n')
    return
end
```

Operativa con ficheros

Cierre del fichero

Cuando se termina de usar un fichero, hay que cerrarlo con la instrucción *fclose(fid)*



```
fid = fopen(nombre, modo)
if fid == -1
    fprintf('Error al abrir fichero\n')
    return
end

% Operaciones de lectura o escritura

fclose(fid)
```

`fclose()` devuelve cero si se cierra bien el fichero

NOTA: `fclose()` es lo que hace un ordenador cuando, antes de quitar un pincho USB le decimos que lo 'expulse', o 'desenchufar de forma segura'. Muchas veces no pasa nada si desenchufamos el USB sin elegir esa opción pero a veces, si hay algún fichero sin cerrar, nos podemos cargar todo el contenido del USB

Modos de apertura:

- 'w': Escribir
- 'a': Añadir

fprintf(): como en pantalla, pero con identificador del fichero

```
fprintf (fid, 'cadena_formatos', lista_de_variables)
```

Especificadores de formato:

- | | |
|----------------|---|
| • '%s': Texto | • '%.2f': double, 2 decimales |
| • '%d': Entero | • '%10.2f': double, ancho 10, 2 decimales |
| • '%f': Double | |

Escribir ficheros

Programas de ejemplo

Se adjuntan tres programas de ejemplo:

- `ej_fprintf_1` : Ejemplos de escribir enteros y vectores
- `ej_fprintf_2` : Ejemplos de escribir números en columnas
- `ej_fprintf_3` : Ejemplo escribir una matriz

Inspeccione detenidamente el código, leyendo atentamente los comentarios.

Tras ejecutarlos, inspeccione el fichero que se genera, para poder relacionar las instrucciones del programa con la salida que se obtiene.

Leer datos de ficheros

El cabezal de lectura

Vuelva a la diapositiva de las cintas magnéticas y observe el **cabezal de lectura** (allí se llama *de reproducción*). Era el dispositivo que iba leyendo de la cinta. La cinta iba enrollándose y avanzando, ofreciendo distinta parte de la cinta al cabezal. Al leer de ficheros, se utiliza esa analogía, y hay que tener siempre en mente dónde está el cabezal antes de ejecutar cada instrucción.

FICHERO

L	i	n		1	\n	L	i	n		2	\n	eof
---	---	---	--	---	----	---	---	---	--	---	----	-----



Posición del cabezal de lectura
nada más abrir el fichero

Al abrir un fichero para lectura, el *cabezal de lectura* se posiciona al principio de la primera línea.

Cada vez que ejecutemos una instrucción de lectura, el *cabezal de lectura* avanzará a lo largo de la tira de datos del fichero.

Leer datos de ficheros

Get Line : `fgetl()`

`fgetl()`: devuelve una cadena de caracteres con el contenido que hay en el fichero desde la posición actual del cabezal de lectura hasta el final de la línea.

```
cad = fgetl(fid)
```

- En la cadena que devuelve no está el carácter fin de línea
- Si se encuentra el final del fichero, devuelve **-1**

Ejemplos:

- **`contalin()`**: función que cuenta las líneas de un fichero
- **`test_contalin.m`**

frewind(): devuelve el cabezal de lectura al principio del fichero

frewind(fid)

FICHERO

L	i	n		1	\n	L	i	n		2	\n	\eof
---	---	---	--	---	----	---	---	---	--	---	----	------



**Posición del cabezal de lectura
nada más abrir el fichero o
después de hacer frewind()**

Leer datos de ficheros

Get String : `fgets()`

`fgets()`: admite varios modos de uso, según los parámetros que le pasemos

```
cad = fgets(fid, num_caracteres)
```

- **fid**: Identificador de fichero
- **num_caracteres**: Número de caracteres a leer. Si se encuentra el fin de línea, lo incluye en la cadena y para la lectura. Si se omite este parámetro, lee hasta el fin de línea, incluyendo el carácter fin de línea. Si encuentra el final del fichero devuelve -1

Ejemplos:

- **test_fgets()**:

Final del fichero

File End Of File : feof()

feof(): devuelve 1 (verdadero) si el cabezal de lectura está en el final del fichero. Devuelve 0 (falso) en caso contrario

```
result = feof(fid)
```

Se suele usar en bucles *while()* de la siguiente forma:

```
while ~feof(fid)

    % Esto se ejecuta si el cabezal
    % de lectura no está en el
    % final del fichero

end
```

El bucle se ejecutará mientras el cabezal de lectura no esté en el fin del fichero

fscanf(): permite leer datos con especificación de formato. En cierta forma, es la instrucción opuesta de *fprintf()*

```
[A, contador] = fscanf(fid, 'formato', size)
```

- **A**: variable con los datos leídos. Según la especificación del formato, pueden ser datos numéricos o de caracteres
- **contador**: número de datos leídos. Se puede omitir y leer solo los datos, sin contador, de la siguiente forma:

```
A = fscanf(fid, 'formato', size)
```

- **fid**: Identificador de fichero
- **'formato'**: Especificadores del formato de los datos a leer
- **size**: Puede ser un número, en cuyo caso se leerán ese número de datos, o un vector con dos valores, en cuyo caso se leerán en forma de matriz. Se explicará en detalle más adelante.

Lectura con formato

Especificadores de formato y tamaño en fscanf()

Especificadores de formato:

- `%d`: dato entero
- `%s`: cadena de caracteres
- `.*s`: saltar cadena de caracteres
- `%f`: dato double
- `%c`: un carácter
- `%nc`: n caracteres

Especificadores de tamaño:

- `N`: N datos
- `[n,m]`: lo que lee, lo guarda en una matriz de n filas y m columnas. La matriz la va rellenando por columnas, por lo que, en general, quedará traspuesta
- `[n,inf]`: lo que lee, lo almacena en una matriz de n filas y el número necesario de columnas para que quepan todos los datos que haya en el fichero

Ejemplos fscanf()

Lineas con datos de distintos tipos

datos_3.txt *

```
1 Madrid 3245 17.5
2 Sevilla 4672 13.2
3 Murcia 234 2.3
.
```

```
>> % Abrir el fichero
>> fid = fopen('datos_3.txt', 'r');
>> % Leer la primera palabra de la primera línea
>> cad = fscanf(fid, '%s', 1)
cad = Madrid
>> % Rebobinar
>> frewind(fid)
>> % Leer el primer número de la primera línea,
>> % saltándose la palabra que hay delante
>> n = fscanf(fid, '%*s %d', 1)
n = 3245
>> % Rebobinar
>> frewind(fid)
>> % Leer el segundo número de la primera línea,
>> % saltandose la palabra y el número que hay
>> % delante. Para saltarse un dato usar siempre
>> % el formato %*s, aunque se quiera saltar un
>> % número
>> x = fscanf(fid, '%*s %*s %f', 1)
x = 17.500
```

Ejemplos fscanf()

Ficheros con columnas de números

No especificando el número de datos a leer

datos_4.txt ×

1	x	y	z	t
2	1	2	3	4
3	5	6	7	8
4				

```
>> % Abrir fichero
>> fid = fopen('datos_4.txt', 'r');
>> % Saltarse la línea de las cabeceras
>> fgets(fid);
>> % Si no se especifica el número de datos
>> % a leer, se lee en el formato indicado
>> % hasta el final del fichero y se
>> % guarda en un vector columna
>> v = fscanf(fid, '%d')
v =
     1
     2
     3
     4
     5
     6
     7
     8
```

Ejemplos fscanf()

Ficheros con columnas de números

Especificar número de datos a leer

datos_4.txt ×

```
1 x y z t
2 1 2 3 4
3 5 6 7 8
4
```

```
>> % Abrir fichero
>> fid = fopen('datos_4.txt', 'r');
>> % Saltarse la línea de las cabeceras
>> fgets(fid);
>> % Especificando cuántos números se leen,
>> % se almacenarán en forma de columna
>> v = fscanf(fid, '%d', 3)

v =

     1
     2
     3
```


Ejemplos fscanf()

Ficheros con columnas de números

Guardando los datos en una matriz

datos_4.txt x

1	x	y	z	t
2	1	2	3	4
3	5	6	7	8
4				

```
>> % Abrir fichero
>> fid = fopen('datos_4.txt', 'r');
>> % Saltarse la línea de las cabeceras
>> fgets(fid);
>> % Especificando un formato de matriz,
>> % se guardan en ese formato, pero
>> % la matriz se rellena por columnas...
>> A = fscanf(fid, '%d', [2,4])
A =

     1     3     5     7
     2     4     6     8
```

Ejemplos fscanf()

Ficheros con columnas de números

Guardando los datos en una matriz

datos_4.txt x

1	x	y	z	t
2	1	2	3	4
3	5	6	7	8
4				

```
>> % Abrir fichero
>> fid = fopen('datos_4.txt', 'r');
>> % Saltarse la línea de las cabeceras
>> fgets(fid);
>> % Para leer la matriz en el mismo formato
>> % que está, hay que leer columnas x filas
>> % y luego trasponer
>> A = fscanf(fid, '%d', [4,2])
```

```
A =
     1     5
     2     6
     3     7
     4     8
```

```
>> A = A'
A =
```

```
     1     2     3     4
     5     6     7     8
```

Ejemplos fscanf()

Ficheros con columnas de números

Número de filas indeterminado. Parámetro `inf`

```
>> % Abrir fichero
>> fid = fopen('datos_4.txt', 'r');
>> % Saltarse la línea de las cabeceras
>> fgets(fid);
>> % Si el número de filas del fichero
>> % es indeterminado, se leen infinitas
>> % columnas, y luego se traspone
>> A = fscanf(fid, '%d', [4, inf])
```

A =

```
1 5
2 6
3 7
4 8
```

```
>> A = A'
```

A =

```
1 2 3 4
5 6 7 8
```

datos_4.txt x

```
1 x y z t
2 1 2 3 4
3 5 6 7 8
4
```

Ejemplos fscanf()

Columnas de texto de ancho conocido

Hay ficheros que tienen alguna columna de texto cuyo ancho es conocido. Aunque tengan espacios, podemos leerlas por su ancho y llegar al dato numérico que pueda haber detrás «*contando caracteres*».

datos_5.txt x

```
1 Pedro Sanz      1.72
2 Roberto Flack   1.68
3 Andrés Calamar  1.90
4
```

```
>> % Abrir fichero
>> fid = fopen('datos_5.txt', 'r');
>> % Leer el nombre
>> nombre = fscanf(fid, '%14c', 1)
nombre = Pedro Sanz
>> % Leer la altura
>> altura = fscanf(fid, '%f', 1)
altura = 1.7200
>> % Saltarse el caracter fin de linea
>> fgetc(fid);
>> nombre = fscanf(fid, '%14c', 1)
nombre = Roberto Flack
>> altura = fscanf(fid, '%f', 1)
altura = 1.6800
>>
```



¡Gracias por su atención!

Si desea más información:

Santiago Higuera de Frutos
santiago.higuera@upm.es