Tema 3: Entrada/Salida de Ficheros

- Objetivo: Almacenar datos en un dispositivo de almacenamiento secundario (p.e. disco duro).
- Pasos a seguir:
  - 1 Abrir fichero
  - 2 Escribir/leer del fichero
  - 3 Cerrar fichero

Tema 3: Entrada/Salida de Ficheros

- Abrir el fichero:
  - FILE *fopen(const char *nombre, const char *modo);
    - Modo de apertura:
      - "r" lectura en un fichero existente (si no existe devuelve un error)
      - "w" escritura en un fichero vacío (si existe se destruye)
      - "a" escritura al final del fichero (si no existe se crea un nuevo fichero)
      - "r+" escritura y lectura (debe existir el archivo)
      - "w+" escritura y lectura de un fichero vacío (si existe se destruye)
      - "a+" lectura y escritura al final del fichero (si no existe se crea un nuevo fichero)
      - "b" modo binario
      - "t" modo texto (modo por defecto)
    - Un puntero nulo (NULL) indica un error.
- Cerrar el fichero:
  - int fclose(FILE *stream);

Entrada/Salida de Ficheros

- Los datos necesarios para abrir un fichero dependen del sistema operativo.
- El tipo de dato FILE* se encarga de almacenar toda esa información.
- La función fopen devuelve un tipo FILE*, NULL si existe algún error.
- Si no se especifica lo contrario trabajaremos con ficheros en modo texto. Los datos que se desean escribir son formateados como texto, de forma similar a la función printf.

```
int main(void)
{
   FILE *fp;

   fp = fopen("datos.txt", "w");

   if( fp == NULL )
   {
      printf(”Error abriendo fichero\n");
      return -1;
    }
  else
  {
   … //Escribimos en el fichero
  }
   fclose(fp);
}
```

3

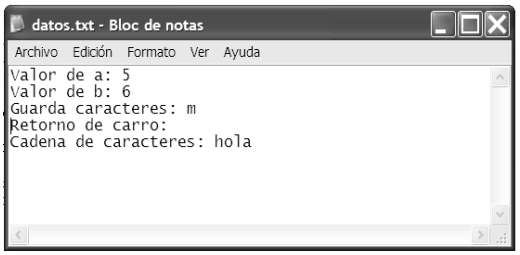
Entrada/Salida de Ficheros

- Entrada/Salida formateada:
  - El funcionamiento es similar a las funciones printf y scanf. Se debe especificar el fichero mediante un tipo FILE*.
  - int fprintf(FILE *fp, const char *format [, argument]...);
  - int fscanf(FILE *fp, const char *format [,argument]...);
- Entrada/salida de cadenas de caracteres:
  - char * fgets (char * str, int num, FILE * stream);
  - int fputs (const char * str, FILE * stream);
- Entrada/Salida de caracteres:
  - Si queremos escribir/leer un carácter cada vez.
  - int fputc(int c, FILE *fp);
  - int fgetc(FILE *fp);

4

Entrada/Salida de Ficheros: Ejemplo

```
int main(void)
{
   FILE *fp;
    int a=5, b=6;
    char caracter1='m';
    char caracter2='\n';
   char cadena[]="hola";

   fp = fopen("datos.txt", "w");

   if(fp == NULL)
   {
      printf("Error abriendo fichero\n");
       return -1;
   }
  else
  {
    fprintf(fp,"Valor de a: %d\n", a);
    fprintf(fp,"Valor de b: %d\n", b);
    fprintf(fp,"Guarda caracteres: %c\n", caracter1);
    fprintf(fp,"Retorno de carro: %c", caracter2);
    fprintf(fp,"Cadena de caracteres: %s", cadena);
  }
  fclose(fp);
}
```



5

Entrada/Salida de Ficheros: Ejemplo

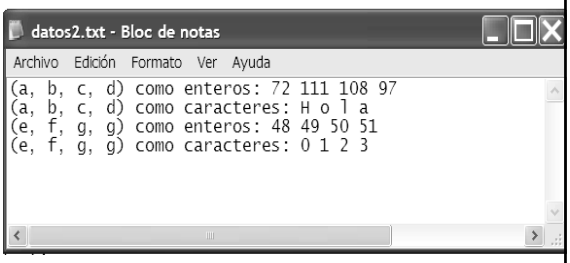
- La función fprintf convierte el dato pasado según se le especifique (%c, %d, %f)
- La tabla ASCII estándar define 128 códigos de caracteres (0-127). Los 32 primeros son códigos de control (no imprimibles), el resto representa caracteres.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

6

Entrada/Salida de Ficheros: Ejemplo

```
int main(void)
{
     FILE *fp;
     int a, b, c, d, e, f, g, h;

   a = 72;   b = 111;   c = 108;   d = 97;
   e = 48;   f = 49;   g = 50;     h = 51;

   fp = fopen("datos2.txt", "w");

   if(fp == NULL){
      printf("Error abriendo fichero\n");
           return -1;
    }
   else{
        fprintf(fp,"(a, b, c, d) como enteros: %d %d %d %d\n", a, b, c, d);
        fprintf(fp,"(a, b, c, d) como caracteres: %c %c %c %c\n", a, b, c, d);
        fprintf(fp,"(e, f, g, g) como enteros: %d %d %d %d\n", e, f, g, h);
        fprintf(fp,"(e, f, g, g) como caracteres: %c %c %c %c", e, f, g, h);
   }
   fclose(fp);
}
```

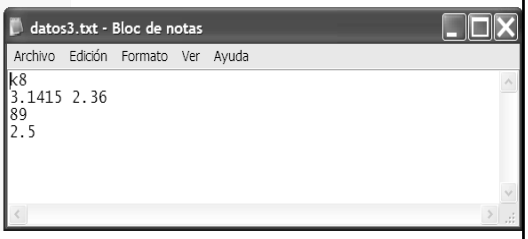


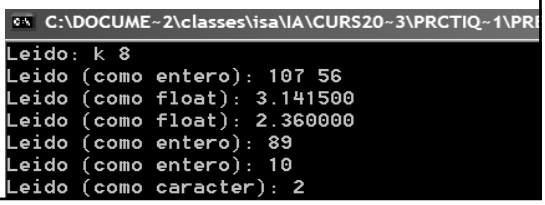
Entrada/Salida de Ficheros

- Debemos recordar que el tipo de dato se convierte a texto según el especificador que se le indique (%c, carácter, %d, entero, %f, float).
- Debemos pensar que tenemos unos bytes en memoria y que pueden representar diferente información, dependiendo de cómo se interpreten.
- Nótese que estamos visualizando el fichero con un editor de textos (Notepad).
- Por ejemplo, el entero a es convertido a dos caracteres ‘7’ y ‘2’.
- Si se escribe como un carácter, entonces se imprime el caracter ‘H’ (véase la tabla ASCII)
- Si escribimos:
- float h=5.63;
- fprintf(fp, “%f”, h);
- Se escribe ‘5’ ‘.’ ‘6’ ‘3’
- El mismo concepto se aplica a la lectura de un fichero
- Nótese que con un especificador %f y %d se convierte a un número en coma flotante o entero. Los espacios se desprecian al hacer la conversion.

Entrada/Salida de Ficheros: Ejemplo

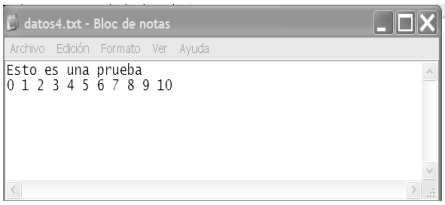
```
int main(void)
{
  char carac1, carac2;
  float comaflot;
  int a;
  fp = fopen("datos3.txt", "r"); //Estamos leyendo del f

  if(fp == NULL)   {
      printf("Error abriendo fichero\n");
       return -1;
    }
  else  {
     fscanf(fp, "%c", &carac1); //Leemos dos caracteres
     fscanf(fp, "%c", &carac2);
     printf("Leido: %c %c\n", carac1, carac2); //Sacamos por pantalla
     printf("Leido (como entero): %d %d\n", carac1, carac2);
     fscanf(fp, "%f", &comaflot); //Leemos un coma flotante
     printf("Leido (como float): %f\n", comaflot);
     fscanf(fp, "%f", &comaflot); //Leemos otro coma flotante
     printf("Leido (como float): %f\n", comaflot);
     fscanf(fp, "%d", &a);
     printf("Leido (como entero): %d\n", a);
     fscanf(fp, "%c", &a);
     printf("Leido (como entero): %d\n", a);
     fscanf(fp, "%c", &a);
     printf("Leido (como caracter): %c\n", a);
   }
   fclose(fp);
}
```





Entrada/Salida de Ficheros

- Normalmente no conocemos el número de datos existentes en el fichero. Debemos dejar de leer cuando se ha llegado al final del fichero.
- La función feof devuelve 1 si se ha llegado al final del fichero, 0 en caso contrario.
  - int feof(FILE *stream);




```
pFich = fopen("datos4.txt", "r");
if(pFich==NULL)
     return -1;
while(!feof(pFich))
{
          fscanf(pFich, "%c",
  &car);
          printf("%c", car);
}
fclose(pFich);
```

10

Entrada/Salida de Ficheros

- Cada vez que se escribe o se lee de un fichero se hace un una posición determinada (que se denomina cursor).
- Al abrir un fichero, el cursor se sitúa al comienzo, con lo que podemos comenzar a leer por el primer byte.
- La función fseek nos permite mover la posición de ese cursor (por ejemplo, situarnos al comienzo del fichero)
  - int fseek(FILE *stream, long offset, int origin);
    - Posiciona el punto de escritura/lectura en un fichero *offset* bytes a contar a partir del origen.
    - Origen:
      - SEEK_CUR Posición actual
      - SEEK_END final del fichero
      - SEEK_SET comienzo del fichero
  - Para situarnos al comienzo del fichero: `fseek(pFich, 0, SEEK_SET );`

11

Entrada/Salida de Ficheros

```
//***********************
// Uso de fopen, fclose, fprintf, fscanf,
//***********************
#include <stdio.h>

void main( void )
{
   FILE *stream;

   int ent;
   float fp;
   char s[81];
   char c;

   stream = fopen("datos.dat", "w+");
   if( stream == NULL )
      printf(”Error abriendo fichero\n");
  else
  {
fprintf(stream, "%s %ld %f %c", ”Datos:",
   65000, 3.14159, 'x' );
// Pone el puntero de posición al
   //principio
fseek(stream, 0, SEEK_SET);
// Lee los datos desde el fichero
fscanf(stream, "%s", s);
fscanf(stream, "%d", &ent);
fscanf(stream, "%f", &fp);
fscanf(stream, "%c", &c);
// Imprime los datos del fichero
printf("%s\n", s);
printf("%d\n", ent);
printf("%f\n", fp);
printf("%c\n", c);
fclose(stream);
  }
}
```

12

Errores frecuentes

- No comprobar si el fichero se ha abierto correctamente

```
FILE *fp;
int dato;
fp = fopen(“datos.txt”, “r”);
fscanf(fp, “%d”, &dato); //Error en ejecución si no se ha podido abrir correctamente
```

- Si el fichero no se ha abierto correctamente, no lo debemos cerrar.

```
FILE *fp;
int dato;
fp = fopen(“datos.txt”, “r”);
if(fp==NULL)
{
   fclose(fp); //Error en ejecución
   return;
}
fscanf(fp, “%d”, &dato);
```

13