UNIVERSIDAD DE MURCIA

Fundamentos de Informática
7. Introducción al manejo de archivos

Fundamentos de Informática
Grado en Ingeniería Química

2

Contenidos

- ***Introducción***
- ***Apertura y cierre de archivos***
- ***Lectura/escritura de datos en ASCII***
- ***Lectura/escritura de bytes***

3

Archivos

Desde C se puede:
- Abrir archivos para leer su contenido
- Crear archivos nuevos con contenido
- Abrir archivos para modificar su contenido

Dos tipos de archivo
- Texto ASCII
- Binarios

Nosotros veremos solamente:
- Archivos de texto ASCII
  - Lectura/escritura de datos ASCII
  - Lectura/escritura de bytes ASCII

4

Archivos

Introducción

Usaremos un tipo de datos especial `FILE` Ej: `FILE` *archivoDatos;

Y las funciones predefinidas en `stdio.h`:

`fopen(…);`	Abre un archivo
`fclose(…);`	Cierra un archivo
`feof(…);`	indica fin de archivo
`fscanf(…);`	Lee un texto con formato de un archivo
`fprintf(…);`	Escribe un texto con formato en un archivo
`fputc(…);`	Escribe un byte en un archivo
`fgetc(…);`	Lee un byte de un archivo

Además veremos la función ***`exit`*`()`**, que está en `stdlib.h`

5

Apertura y cierre de archivos

6

Apertura de un archivo

Devuelve el enlace al archivo,
o `NULL` si se produce un error.

```
FILE fopen(nombre,modo)
```

El nombre del archivo a abrir

Modos de apertura

r : lectura ASCII

w: escritura ASCII (sobreescribe)

a: añade al final del archivo ASCII

rb: lectura de datos binarios

wb: escritura de datos binarios

ab: añade al final del archivo binario

7

Archivos

Apertura de un archivo

El nombre del archivo a abrir (se podría introducir por teclado, claro)

f es una variable del tipo FILE

Abre archivo modo lectura ASCII

Si se produce un error…

El programa acaba irregularmente

```
#include <stdio.h>
#include <stdlib.h>
#define NARCH “datos.txt”
main()
{

 FILE *f;

 f = fopen ( NARCH, "r“ );

 if (f==NULL)
 {
    printf("Error al abrir archivo %s\n",NARCH);
    printf(“Pulsa <enter> …”); getchar();
    exit(1);
 }

. . .
```

8

Archivos

Apertura de un archivo

Se pueden abrir varios archivos a la vez, y leer y/o escribir de unos y otros.

```
#include <stdio.h>
#include <stdlib>
#define NARCH1 “entrada1.txt”
#define NARCH2 “entrada2.txt”
#define NARCH3 “salida.txt”

main()
{
 FILE *fe1,*fe2,*fs;

 fe1 = fopen ( NARCH1, "r“ );
 fe2 = fopen ( NARCH2, "r“ );
 fs = fopen ( NARCH3, “w“ );

if (fe1==NULL || fe2==NULL || fs=NULL)
 {printf("Error”); getchar(); exit(1); }

. . .
```

Cierre de un archivo

Devuelve `EOF` si se produce un error al cerrar

```
int fclose(FILE)
```

La variable asociada al archivo

```
main()
{
 int err;
 FILE *fe1,*fe2,*fs;
 fe1 = fopen ( NARCH1, "r" );
 fe2 = fopen ( NARCH2, "r" );
 fs = fopen ( NARCH3, "w" );
 if (fe1==NULL || fe2==NULL || fs=NULL)
 {printf("Error"); getchar(); exit(1); }

 . . .

err=fclose(fe1);
if (err==EOF) printf("Error al cerrar");
fclose(fe2);
fclose(fe3);
. . .
```

Controlar el error no es obligatorio

Lectura/escritura de datos en archivos ASCII

fscanf y fprintf

Se usan cuando:

- Se trabaja con archivos ASCII
- Conocemos la estructura de su contenido

datos.txt

```
Datos: 5
2.3 15 7.0 2.22 13.7
Datos: 3
3 5 7
```


Lectura de texto con `fscanf`

Devuelve el número de datos leídos, o "0" si se produce un error. Es opcional recoger o no este valor.

La cadena de formato habitual, seguida de las variables a leer, con el símbolo &.

```
int fscanf(FILE,formato, …)
```

La variable del archivo

```
e = fscanf(f,"%d %d %d\n",&a,&b,&c);
```

Lee una línea formada por tres enteros separados por espacios. Debe acabar con <intro>
Si la lectura es correcta **e** vale 3

13

Archivos

Lectura de texto con `fscanf`

```
#include <stdio.h>
#define NA “datos.txt”
main()
{
 int err,a,b,c;
 FILE *f;

 f=fopen(NA,"r");
 if (f==NULL )  {printf("Err”); getchar(); exit(1); }

 err = fscanf(f,"%d %d %d\n",&a,&b,&c);
 if (err!=3)
  {
   printf("Error al leer datos en %s\n",NA);
   printf(“Pulsa…”); getchar();
   exit(1);
  }

 printf(“Datos: %d %d %d\n", a,b,c);
 fclose(f);
}
```

Aquí se controla que la lectura sea incorrecta, pero no es obligatorio ponerlo

14

Archivos

Lectura de texto con `fscanf`

Este ***fscanf*** no solo lee los números, también la cadena de caracteres “Dato x:” …

datos.txt

```
Dato 1: 0
Dato 2: 10
Dato 3: 20
F
```

```
#define NA “datos.txt”
main()
{
 int a,b,c,err; FILE *f;  char fin=‘a’;

 f=fopen(NA,"r");
 if (f==NULL) {printf("Error\n");getchar();exit(1);}

 fscanf(f,"Dato 1: %d\n Dato 2: %d \n Dato 3: %d\n",&a,&b,&c);
 fscanf(f,"%c\n",&fin);

 if (fin==‘F’) printf ("Lectura completada\n");
 else printf("Error en lectura\n");

 fclose(f);

 printf(“Datos: %d %d %d\n", a,b,c);
}
```

15

Archivos

Lectura de texto con `fscanf`

```
fscanf (f,"Dato 1: %d\n", &a);
fscanf (f,"Dato 2: %d\n“, &b);
fscanf (f,"Dato 3: %d\n", &c);
```

También puedo leer así

16

Archivos

Escritura de texto con `fprintf`

Devuelve el número de caracteres escritos, o “0” si se produce un error. Es opcional recoger o no este valor.

La cadena de formato habitual, seguida de las variables a escribir.

```
int fprintf(FILE,formato, …)
```

La variable del archivo

```
fprintf(f,"1:%d 2:%d 3:%d\n",a,b,c);
```

17

Archivos

Escritura de texto con `fprintf`

Crear un archivo y escribir en él

Crea archivo para escribir ASCII

Escritura

```
#define NA “datos.txt”
…
int a,b,c,err;
FILE *f;

a=10;b=20;c=30;

f=fopen(NA,"w");
if (f==NULL) {printf("Error\n");getchar();exit(1);}

fprintf(f,"1: %d\n2: %d \n3: %d\n",a,b,c);

fclose(f);

…
```

18

Archivos

Escritura de texto con `fprintf`

Si quiero controlar el posible error, sería así

```
e=fprintf(f,"1: %d\n2: %d \n3: %d\n",a,b,c);
if (e<1)
{
 printf("Error escribiendo\nPulsa…");
 getchar();
 exit(1);
}
```

19

Lectura/escritura de bytes ASCII

20

fgetc y fputc

- Leen/escriben carácter a carácter (byte a byte) por lo que no son buenos para leer datos numéricos.

datos.txt

```
Todas las islas, incluso las
conocidas, son desconocidas
mientras no desembarquemos en
ellas.
```

21

Archivos

Lectura con `fgetc`

`fgetc` lee de un archivo un caracter ASCII (un byte)

Devuelve el byte leido. Es un unsigned char convertido a int, un valor entre 0 y 255. Devuelve EOF si es el fin del archivo.

La variable del archivo

```
int fgetc(FILE)
```

```
c = fgetc(f);
```

22

Archivos

Lectura con `fgetc`

Abre un archivo ascii y lee 10 caracteres

Puedo hacer esto solo si se exactamente cuantos caracteres quiero leer

```
#define NA "datos.txt"
…
int i,c; FILE *f;

f=fopen(NA,"r");
if (f==NULL) {printf("Error\n");getchar();exit(1);}

for (i=0; i<10; i++)
{
  c=fgetc(f);
  …
}

fclose(f);
…
```

23

Archivos

Lectura con `fgetc`

De este modo puedo leer hasta fin de archivo

```
c=fgetc(f);
while  (c != EOF)
{
  …
  c=fgetc(f);
}
```

Así también, más condensado

```
while  ( (c=fgetc(f)) != EOF)
{
  …

}
```

24

Archivos

Lectura con `fgetc`

También usando la función **`feof`**

¡ojo! No confundir la función **`feof`** con el define **`EOF`**

```
while (feof(f) == 0)
{
 c = fgetc(f);
 …
}
```

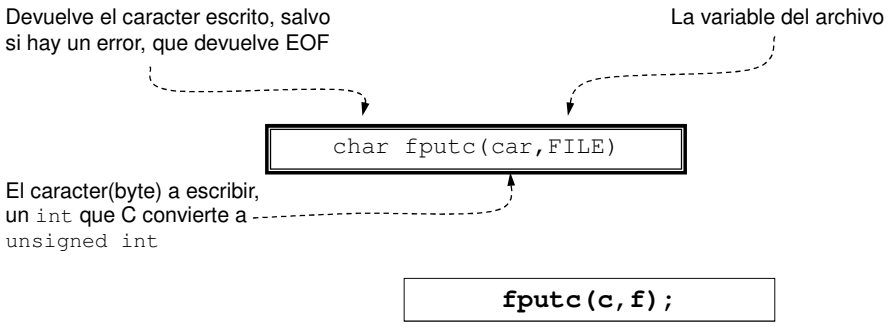
Devuelve un 0 si no es fin de archivo, y un !=0 si es fin de archivo

La variable del archivo

```
int feof(FILE);
```

Escritura con `fputc`

`fputc` escribe en un archivo un caracter ASCII (un byte)




```
fputc(c,f);
```


Escritura con `fputc`

Este programa lee un archivo ASCII y y genera otro al que le quitado las 'a'

datos.txt

```
caravaca es una
ciudad alegre
```

```
crvc es un
ciudd legre
```

salida.txt

```
#include<stdio.h>
#define NA1 "datos.txt"
#define NA2 "salida.txt"
main()
{
 int c;   FILE *fe,*fs;
 fe=fopen(NA1,"r");
 fs=fopen(NA2,"w");

 c=fgetc(fe);
 while  (c != EOF)
 {
   if (c!='a') fputc(c,fs);
   c=fgetc(fe);
 }
 fclose(fs);
 fclose(fe);
}
```

27

Archivos

Hay más funciones para manejo de archivos.

Si tienes curiosidad, consulta las funciones :

```
fread          frwite
fgets          fputs
rewind         fflush
fseek          ftell
...
```
