Conceptos

- **Los nombres de los ficheros: externos o físico (nombre único del archivo o fichero en el sistema de archivos) e interno o lógico (identificador del fichero dentro de un programa).**

- **Las operaciones básicas en lenguaje algorítmico:**
  - **Abrir_f (f: fichero de T, nombre: cadena, modo: cadena),**
  - **Cerrar_f (f: fichero de T),**
  - **Leer_f (f: fichero de T, V:T),**
  - **Escribir_f (f: fichero de T, V:T),**
  - **Fin_fichero (f : fichero de T, eof : lógico)**

- **¿Cómo implementa C estos conceptos?**

Manejo básico de Ficheros en C

- Para el lenguaje C un fichero no es más que una porción de almacenamiento de memoria (que representará ***como un conjunto de caracteres*** o como un conjunto de bytes).
- C representa un fichero como una estructura: la definición de la biblioteca ***stdio.h*** hacer referencia a una estructura FILE:

```
struct _iobuf {
    char *_ptr; /* Puntero al buffer actual */
    int  _cnt; /* Contador del byte actual */
    char *_base; /* Dirección base del buffer de E/S */
    char _flag; /* Flags de control */
    char _file; /* Número de fichero */
};
typedef FILE struct _iobuf; /* Notación abreviada */
```

- ***Pero no es necesario utilizar los campos de esta estructura para acceder al contenido de un fichero. De hecho, sólo usaremos la notación abreviada FILE y no accederemos a los campos***
- FILE corresponde al patrón del fichero. Cuando se manejen ficheros, se utilizará este tipo de dato estructurado.

Manejo básico de Ficheros en C

- Definición de un fichero (nombre interno) en C:

```
#include <stdio.h>
main() {
FILE *f;  /* Descriptor de fichero, su nombre interno; es la única variable que utilizaremos para
     referirnos al fichero*/
}
```

Las funciones básicas de manejo de ficheros en C son:

- **FILE * fopen (const char *nombre, const char * modo)**
- **int fclose (FILE * fich)**
- **int fprintf (FILE *fich, const char * formato, …)**
- **int fscanf (FILE *fich, const char * formato, …)**
- **int getc (FILE *fich)**
- **int putc (int car, FILE * fich)**
- **char * fgets (char *texto, int n, FILE * fich)**
- **int fputs (const char *texto, FILE *fich)**
- **int fflush (FILE * fich)**

Los ficheros secuenciales en C como ficheros de texto

- Las funciones ***fopen()*** y ***fclose()*** trabajan con ficheros de texto con *buffer*: la entrada y salida se almacenan temporalmente en un área de memoria llamada *buffer* (tampón o depósito).
- Cuando el *buffer* se llena, el contenido se traspasa a un bloque de memoria (en lectura) o a la salida (en escritura), y se recomienza el proceso.
- Una de las tareas principales de ***fclose()*** es el vaciado del *buffer*, que podría haber quedado parcialmente lleno al cerrar el fichero, y que puede ocasionar que realicemos una lectura errónea en el futuro.
- También se puede utilizar la función ***fflush()*** para, literalmente, vaciar el contenido del *buffer,* sin cerrar el fichero.

Nota: un fichero de texto es aquel en el que la información se almacena como caracteres, utilizando su código ASCII (o equivalente).

- Un fichero binario es aquel en el que se almacena información binaria (como puede ser un código en lenguaje máquina).
- Las funciones de Entrada/Salida que aparecen en la transparencia anterior, y que se van a describir a continuación, están diseñadas únicamente para trabajar con ficheros de texto. Existen versiones equivalentes para ficheros binarios (***open(), close(), read(), write()***).

Funciones de apertura y cierre

- **Abrir_f (f: fichero de T, nombre: cadena, modo: cadena),**
- **Cerrar_f (f: fichero de T),**

¿Cómo implementa C estos conceptos?
Están definidos como funciones en stdio.h

- **FILE * fopen (const char *nombre, const char * modo)**
  - Nombre: Nombre externo de fichero: cadena de caracteres.
  - Modo: cadena de caracteres: Uso del fichero: "r", "w" o "a": equivale a "l", "e" ó "a".
  - Devuelve um puntero a fichero: FILE * fich → nombre interno del fichero (a partir de ese momento f está vinculado al fichero externo)

Si ***fopen()*** no consigue abrir el fichero, devuelve NULL.

Es importante realizar la comprobación de que se ha abierto correctamente el fichero antes de proceder a su utilización.

```
FILE *Fich;
Fich = fopen("datos.txt", "r");
If (Fich != NULL) {
    /* puedo usar el fichero */ fclose(Fich);
}
```

Funciones de apertura y cierre (II)

- **int fclose (FILE * fich)**
  - Se utiliza el nombre interno para cerrarlo.
  - Cierra el fichero, y desvincula ***fich*** del fichero externo.
  - Cuando ha ocurrido algún problema al cerrar un fichero devuelve el valor -1, mientras que si el cierre del fichero ha sido satisfactorio devuelve 0.

```
#include <stdio.h>
main() {
FILE *entrada; /* Declara un puntero a un fichero */
int car;
/* Abre el fichero test para lectura, comprueba si existe */
if ((entrada = fopen("test", "r")) != NULL) {
    /* entrada: el puntero a FILE apunta ahora a "test", que es el nombre externo del fichero; a
    partir de ahora entrada es su nombre interno */
    fclose(entrada);  /* Cierra el fichero sin haberlo modificado*/
}
}
```

Funciones de lectura y escritura

- **Leer_f (f: fichero de T, V:T),**
- **Escribir_f (f: fichero de T, V:T),**

¿Cómo implementa C estos conceptos?

– Igual que la E/S estándar (teclado/pantalla) vistos en el tema anterior, pero ahora indicando sobre qué fichero se están haciendo.

1. Lectura / Escritura en fichero con formato

   int fprintf (FILE *fich, const char * formato, …)

   int fscanf (FILE *fich, const char * formato, …)

2. Lectura / Escritura en fichero línea a línea

   char * fgets (char *texto, int n, FILE * fich)

   int fputs (const char *texto, FILE *fich)

3. Lectura / Escritura en fichero carácter a carácter

   int getc (FILE *fich)

   int putc (int car, FILE * fich)

Fin_fichero(f: fichero de T, eof: lógico) → No existe implementación directa. Las funciones de lectura devolverán EOF ó -1 para indicar si en la lectura se alcanzó el fin de fichero.

- Lectura / Escritura en fichero con formato

  int fprintf (FILE *fich, const char * formato, …)

  int fscanf (FILE *fich, const char * formato, …)

  – Funcionan como printf() y scanf(), salvo que llevan un argumento adicional. Ese argumento indica el descriptor de fichero, que se convierte en el primer argumento de la lista.

```
#include <stdio.h>
main(){
FILE *fi;
int edad;
fi = fopen("entrada.txt", "r");     /* Apertura en modo lectura */          /* fi apunta a "entrada.txt" */
If (fi != NULL) {
     fscanf (fi, "%d", &edad);
     fclose (fi);
}
If ((fi = fopen ("datos.txt", "a"))  != NULL) {    /* Modo añadir */          /* fi apunta a "datos.txt" */
     fprintf (fi,"pedro tiene %d\n", edad);
     fclose(fi);
}
}
```

- Lectura / Escritura en fichero línea a línea

char * fgets (char *texto, int n, FILE * fich)

int fputs (const char *texto, FILE *fich)

fgets(char *texto, int longitud, FILE *fich)

- Utiliza tres argumentos:
  - Puntero al lugar de destino de la línea que se va a leer.
  - Longitud de la tira que se está leyendo. La función se detiene cuando se lee el carácter de nueva línea o cuando se han leido MAXLIN -1 caracteres.
  - Puntero al fichero en el que se está leyendo.
- Una diferencia entre gets() y fgets(), es que la primera convierte el carácter retorno de carro en '\0'*, mientras* que fgets lo mantiene.
- Además devuelve el símbolo NULL cuando ha leído un fin de fichero (EOF).

fputs(char *texto, FILE* fich)

- Simplemente añade un argumento adicional al final de fputs() que es un puntero a fichero:

  fputs("Esto es un ejemplo", fich);

  donde *fich* debe ser un puntero a fichero previamente abierto mediante una sentencia fopen().

- La forma general de uso es:

  control = fputs(cadena, puntero-a-chero)

  donde control es un entero que toma el valor EOF si se encuentra una marca de fin de fichero o un error.
- Al igual que puts() esta función no copia el '\0'del final de la tira. Pero tampoco añade un carácter de nueva línea a la salida.

- Lectura / Escritura en fichero carácter a carácter

int getc (FILE *fich)

int putc (int car, FILE * fich)

Funcionan de forma similar a getchar() y putchar().

Además ha de especificarse el fichero sobre el que han de leer o escribir un carácter.

```
car = getchar();
car = getc(entrada);
putchar(car);
putc(car, salida);
```

En ambos casos, tanto ***entrada*** como ***salida*** han de ser:

FILE *entrada, *salida;

Existen dos descriptores (nombres internos) de ficheros predefinidos:

stdin, está definido como la entrada estándar predefinida (teclado, en general);

stdout, que no es más que la salida estándar predefinida (pantalla, en general).

Ejemplo 1: El siguiente programa lee el contenido de un fichero llamado ***"prueba.txt"*** y lo imprime por pantalla.

```
/* Nos dice qué hay en el fichero "prueba.txt" */
#include <stdio.h>
main() {
FILE *entrada; /* Declara un puntero a un fichero */
int car;
/* Abre el fichero test para lectura, comprueba si existe */
if ((entrada = fopen("test", "r")) != NULL) {
    /* entrada: el puntero a FILE apunta ahora a "prueba.txt"; "prueba.txt" es el
    nombre externo del fichero, ahora entrada es el nombre interno */
while ((car=getc(entrada)) != EOF ) putc (car, stdout);
/* Toma caracter de entrada y lo coloca en la pantalla */
fclose(entrada); /* Cierra el fichero */
}
else printf ("No puedo abrir el fichero \"prueba.txt\".\n");
} /* main */
```

Ejemplo 2: /* Lee de un fichero una linea cada vez */

```
#include <stdio.h>
#define MAXLIN 80
main() {
    FILE *entrada;
    char  tira[MAXLIN];
    if ((entrada = fopen("cuanto","r")) != NULL) {
    while (fgets(tira, MAXLIN, entrada) != NULL)
       puts(tira);
    fclose(entrada);
    }
}
```

Ficheros de acceso aleatorio en C

Acceso aleatorio con fseek()

- La funcion fseek() permite tratar los ficheros como vectores, moviéndose directamente a un byte determinado del fichero abierto por fopen().
- Obviamente, esta función sólo se puede utilizar sobre ficheros que admitan acceso aleatorio.
- Los argumentos de fseek() son los siguientes:
  - Puntero a fichero.
  - Offset (desplazamiento): indica la distancia a que debemos movernos desde el punto de comienzo. Este parámetro debe declararse de tipo long y puede ser positivo o negativo (movimiento hacia delante o hacia atras): Indicador del punto de referencia para el offset.
  - Modo: El offset se mide desde:
    - 0, el comienzo del fichero,
    - 1, posicion actual,
    - 2, fin del fichero
- La funcion fseek() devuelve el valor 0 si todo ha ido bien, y devuelve el valor -1 si ha habido algún error.

Argumentos en línea

```
#include <stdio.h>
main(argc, argv)
int argc;   /*int argc: indica número de argumentos, incluído el nombre del programa ejecutable */
char *argv[];   /* char *argv[]: argv[i] es el (i+1)-ésimo argumento. El primero es el nombre del programa. */
/* fseek() para imprimir el contenido de un fichero */

#include <stdio.h>
main(numero, nombres) /* No es obligatorio usar argc y argv como nombres de las variables*/
int numero;
char *nombres[];
{
      FILE *fp;
      long offset = 0L; /* 0 de tipo long */
      if (numero < 2) puts ("Necesito un fichero como argumento\n");
      else {
            if ((fp =fopen(nombres[1], "r")) == 0)
                  printf ("No puedo abrir %s", nombres[1]);
            else {
                  while (fseek(fp, offset++, 0) == 0)
                    putchar(getc(fp));
                  fclose(fp);
            }
      }
}
```

Ejemplo

```
/* Reduce un fichero a su tercera parte */
#include <stdio.h>
main(int argc, char *argv[]) {
FILE *in, *out; /* Declara dos punteros a fichero */
int ch;
static char nombre[20]; /* Fichero de salida */
int con = 0;
if (argc < 2) /* Existe fichero de entrada? */
  printf ("No hay nombre de fichero como argumento.");
else {
   if ((in = fopen(argv[1], "r" )) != NULL){  /* Copia el nombre del fichero en un array*/
        strcpy (nombre, argv[1]);
        strcat (nombre, ".red"); /* Crea fichero de extensión reducida: nombre_fichero.red*/
        out = fopen (nombre, "w");          /* Abre fichero como salida de escritura */
        /* Envía un carácter de cada tres */
        while ((ch = getc(in)) != EOF)
                  if (cont++ % 3 == 0) putc (ch, out) ;
        fclose(in);
        fclose(out);
  }
else printf ("No puedo abrir el fichero %s\n", argv[1]);
}
}
```