Llamadas al sistema para manejo de ficheros

open()

Sintaxis: int open(char *nombre, int operaciones[, int permisos]);

Archivo de cabecera: #include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

La llamada *open()* crea un fichero o abre un fichero ya creado para lectura y/o escritura.

El parámetro operaciones tomará los siguientes valores:

Valor	Significado
O_RDONLY	Abre para lectura
O_WRONLY	Abre para escritura
O_RDWR	Abre para lectura y escritura (equivalente a O_RDONLY|O_WRONLY)
O_CREAT	Crea el fichero si no existe e ignora el parámetro si ya existe.
O_APPEND	Sitúa la posición al final antes de escribir

El parámetro opcional *permisos* sólo tiene sentido al utilizar *O_CREAT*, y sirve para asignar los permisos del fichero recién creado, en caso de que no existiera.

La llamada *open()* devuelve -1 en caso de error, o un entero positivo como descriptor o manejador de fichero, en caso de éxito, Este entero será la posición de la *Tabla de Canales* donde se asigna el fichero. Este entero debe ser almacenado para acceder posteriormente a los datos del fichero, ya que el resto de operaciones sobre el mismo se realizarán a través del descriptor de fichero.

close()

Sintaxis: int close(int fichero);

Archivo de cabecera: #include <unistd.h>

La llamada *close()* cierra un fichero abierto, cuyo descriptor es *fichero* y libera la entrada correspondiente en la *tabla de canales*. Devuelve –1 en caso de error y 0 en caso de éxito.

creat()

Sintaxis: int creat(char *nombre, int permisos);

Archivo de cabecera: #include <unistd.h>

La llamada *creat()* crea un nuevo fichero con el nombre indicado en el parámetro *nombre* y con los permisos indicados en el parámetro *permisos*, y además lo abre para escritura aunque los permisos no lo permitan. Si el fichero ya existe, los trunca a cero y lo abre para escritura, ignorando el parámetro de permisos.

creat() devuelve –1 en caso de error o el descriptor del fichero abierto en caso de éxito.

read()

Sintaxis: int read(int fichero, char *buffer, int nbytes);

Archivo de cabecera: #include <unistd.h>

La llamada *read()* lee una secuencia de *nbytes* octetos (bytes) del fichero o dispositivo abierto (para lectura) cuyo descriptor es *fichero* y los escribe a partir de una posición de memoria indicada por el parámetro *buffer*. Devuelve –1 en caso de error o el número de bytes realmente leídos en caso de éxito.

Es posible que el número de bytes que realmente se leen no coincida con el pedido a través del parámetro *nbytes*. Esto puede suceder cuando, por ejemplo, se intentan leer más bytes de los que realmente hay en un fichero. El hecho de que la llamada *read()* devuelva 0 indicará que se ha alcanzado el final del fichero

buffer es un puntero a carácter (array de caracteres) y por tanto debe apuntar a una zona de memoria suficientemente grande como para contener todos los caracteres solicitados en el parámetro *nbytes*. De lo contrario, la llamada *read* podría sobrescribir otras variables del proceso, producir un error o incluso producir un error de protección y abortar.

Después de una llamada *read()* la posición del fichero (punto del fichero desde el cual empieza a leerse o escribirse) avanza tantos octetos como caracteres (bytes) hayan sido leídos.

write()

Sintaxis: int write(int fichero, char *buffer, int nbytes);

Archivo de cabecera: #include <unistd.h>

La llamada *write()* es semejante a *read()*, pero a la inversa. Escribe una secuencia de *nbytes* octetos (bytes) en el fichero o dispositivo abierto (para lectura) cuyo descriptor es

fichero. La secuencia de datos a escribir se obtiene a partir de una posición de memoria indicada por el parámetro *buffer*. Devuelve –1 en caso de error o el número de bytes realmente escritos en caso de éxito.

Después de una llamada `write()` la posición del fichero avanza tantos bytes como caracteres hayan sido escritos.

EJEMPLO

Copia el contenido del fichero f1.in (debiendo haber sido creado previamente) en el fichero f2.out

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

void main(){
  int         file1,file2;
  int         total,nbytes;
  char buf[512];
  char *s;
  if ((file1 = open("f1.in",O_RDONLY)) == -1){
            s = "error al abrir el fichero de entrada. \n";
            write(2,s,strlen(s));
            exit(1);
  }
  if ((file2 = creat("f2.out",0777)) == -1){
            s = "error al crear el fichero de salida.\n";
            write(2,s,strlen(s));
            exit(1);
  }
  nbytes = 512;
  while ((total = read(file1,buf,nbytes)) > 0)
            if (write(file2,buf,total) == -1){
               s = "error al escribir en el fichero.\n";
               write(2,s,strlen(s));
               exit(1);
            }
  if (total == -1){
            s = "error al leer del fichero.\n";
            write(2,s,strlen(s));
            exit(1);
  }
  if (close(file1) == -1){
            s = "error al cerrar el fichero de entrada.\n";
            write(2,s,strlen(s));
            exit(1);
  }
  if (close(file2) == -1){
            s = "error al cerrar el fichero de salida.\n";
            write(2,s,strlen(s));
            exit(1);
  }
  exit(0);
}
```