Control de errores de las llamadas al sistema

Los programas deben siempre comprobar después de una **llamada al sistema**[1] si todo es correcto. Cuando se produce un error al realizar una llamada al sistema, ésta devuelve el valor -1. Todo proceso contiene una variable global llamada ***errno***, de manera que cuando una llamada al sistema termina de forma satisfactoria, el valor de esta variable permanece inalterado, pero cuando la llamada falla, en *errno* se almacena un código de error.
errno no da información detallada sobre el error que se ha producido pero el lenguaje de programación C proporciona las funciones **perror ()** y **strerror ()** que se pueden utilizar para mostrar el mensaje de texto asociado a la variable *errno* .

- perror() muestra la cadena que se pasa a la función, seguida de dos puntos, un espacio, y luego la representación textual del valor de la variable errno actual.
- strerror() devuelve un puntero a la representación textual del valor actual de la variable *errno*.

Ejemplo:

```c
// uso_perror.c - Funciones perror() y fprintf() con stderr y errno

#include <stdio.h>
#include <sys/file.h>
#include <errno.h> //errno
#include <string.h> //strerror()

int main()
{
  int fd;
  fd=open("/asdf", O_RDONLY); //No existe el fichero
  if (fd==-1) {
     fprintf(stderr, "Error %d: %s\n", errno, strerror(errno));  //o bien:
     perror("Error");
  }
  if ((fd=open("/", O_WRONLY))==-1) {
     fprintf(stderr, "Error %d: %s\n", errno, strerror(errno)); //o bien:
     perror("Error");
  }
}
```

Cuando el código se compila y ejecuta, se produce el siguiente resultado:

---

[1] Las llamadas al sistema están recopiladas en la sección 2 de Man. Ejemplos en nuestra aventura: open(), close(), chdir(), getcwd(), dup(), dup2(), execvp(), exit(), fork(), getpid(), kill(), pause(),, wait(), waitpid(). Hay algunas llamadas al sistema que siempre tienen éxito, y no retornan ningún valor para indicar si se ha producido un error; ejemplos en nuestra aventura: getpid(), getppid()

```
uib ~/Documentos/Asignaturas/21708-SOI/practicas/ejemplos$ ./uso_perror
Error 2: No such file or directory
Error: No such file or directory
Error 21: Is a directory
Error: Is a directory
```

En general para mostrar errores es preferible utilizar la salida de error estándar ***stderr*** (con **fprintf**()), en vez de utilizar la salida estándar *stdout* con printf(). De esta manera cuando utilicemos redirección a un fichero, las salidas del programa irán al fichero en vez de al *stdout,* pero los errores se visualizarán por la pantalla y no se incluirán en el fichero.
También es preferible **perror**() a printf() por dos razones, la primera porque se utiliza el *stream* correcto (la salida estandar de errores) y la segunda es la generacion de un mensaje de error describiendo el problema (representación textual del valor actual de la variable *errno*).

errno.h contiene los códigos de error de C en Linux. Se puede obtener un listado desde consola ejecutando el comando **errno -l**