MANEJO DE DIRECTORIOS

Operaciones básicas sobre directorios:

int **mkdir**(const char *pathname);	Crea un directorio, devuelve 0 si éxito o -1 si error.
int **rmdir**(const char *pathname);	Borra un directorio si está vacio.
int **chdir**(const char *pathname);	Cambia el directorio actual del programa
char ***getcwd** (char *buffer, int logmax)	Copia en la variable buffer de tamaño longmax el nombre del directorio actual de trabajo
int **remove**(const char *filename);	Borra el archivo indicado
int **rename**(const char *name_old, const char *name_new);	Cambia el nombre de archivo name_old por name_new

RECORRIDO DE UN DIRECTORIO

Usamos las funciones de la librería estándar definidas en el archivo <dirent.h>

// Se utiliza una estructura de datos similar a FILE *, para realizar el recorrido, sólo existen funciones para leer el directorio.
DIR *dirp;

DIR ***opendir**(const char *nombre);	Abrir un directorio para recorrerlo, devuelve NULL si falla.
struct dirent * **readdir** (DIR *dirp)	Lee la siguiente entrada del directorio almacenando el resultado en la estructura dirent.
void **rewindir** (DIR *dirp)	Nos sitúa al principio del directorio
int **closedir** (DIR *dirp)	Cierra el directorio

La estructura dirent varia según el sistema operativo y/o el tipo de sistema de ficheros que usemos (FAT/NFTS/EXT2) sólo el campo d_name es común en todas las implementaciones.

```
struct dirent {
                /* Resto de campos dependientes de S.O */

                 char          d_name[256];    /* nombre del fichero */
               }
```

Las dos primeras entradas de cualquier directorio corresponden los directorios . (El propio directorio) y .. (directorio padre). En el directorio raiz .. es igual al directorio .

<Consultar los ejemplos de la página web>

Consulta sobre la información de un fichero

Para obtener la información del sistema sobre un archivo podemos utilizar la función stat

#include <sys\stat.h>

int stat(const char *path, struct stat *statbuf);

La función recibe como parámetro de entrada el nombre del fichero (path), rellenado la estructura statbuf con la información del sistema. La función devuelve 0 si éxito o -1 si error.

La estructura stat varía según el S.O. o el tipo de partición FAT / NTFS, en general los campos comunes son:

```
struct stat
    {
         // Resto de campos según implementación

         dev_t         st_dev;      /* dispositivo */
         mode_t        st_mode;     /* protección */
         off_t         st_size;     /* tamaño total, en bytes */
         time_t        st_atime;    /* hora último acceso */
         time_t        st_mtime;    /* hora última modificación */
         time_t        st_ctime;    /* hora último cambio */
     };
```

Según los valores de los bits del campo st_mode podemos saber si el fichero es un archivo normal, un directorio, si se puede leer o escribir, etc., para lo cual existen unos valores predefinidos.

```
if ( statbuf.st_mode & S_IFDIR  ) puts (“Es un directorio”);
if ( statbuf.st_mode & S_IREAD  ) puts (“El archivo se puede leer”);
if ( statbuf.st_mode & S_IWRITE ) puts (“El archivo se puede escribir”);
```