FSlib

The FSlib module is a shared library providing the C runtime library routines which assume an underlying filing system, and the ROME-specific routines for handling multiple filing system and device formats.

Data Definitions

The *fslib.h* header file in the module contains the following type definitions, some of which are also passed by pointer reference in the *Files* messageset:

FSLIB_ATTR	This data structures contains all the attributes of a single file in the filing system. Not all values (for example the *inode* number) are provided for all filing system types.
FSLIB_DEVICE	This data structure contains the *name* of a device in the system which supports a filing system and the list of attributes for that system.
FSLIB_DEVICE_ATTR	This data structure contains the device-specific attributes of a filing system, Currently these comprise the maximum block number in the system, *maxblk*, and the file block size *blksize*.
FSLIB_DIRENT	This data structure contains the *name* of a file within the filing system and a list of the attributes for that file.
FSLIB_PARTITION	This data structure contains information about multiple filing systems residing on a single device. Each partition has a *name* and *type*. Depending on the system, there may be a *password* associated with the partition. All partitions are represented by *startblock* and *endblock* physical block numbers on the device.

Shared Library Macros and Routines

In general, except for *fseek* and *ftell*, the filing system routines do not use an open *FILE* structure to communicate with the filing system layer. This is mainly to avoid protection problems, for example by needing to open a file in order to change its permissions, but also for efficiency (one message exchange rather than three). Therefore, most of these routines take a full URL string as the filename argument, and use the scheme part to determine the destination process. It is up to the caller to ensure that the full string is presented to the routine, for example by adding the current device root and directory to a ‘simple’ name.

delete

int *delete*(
const char **url*)

The *delete* routine removes the specified *url* from the filing system. The endpoint must be a regular file or link, the operation will fail if the endpoint is a directory.

fslib_get_attr

int *abs*(
const char **url*,
FSLIB_ATTR **attr*)

The *fslib_get_attr* routine returns the filing system attributes for the file specified by *url* into the structure pointed to by *attr.*

fslib_get_device_attr

int **fslib_get_device_attr*(
const char **url*,
FSLIB_DEVICE_ATTR **attr*)

The *fslib_get_device_attr* routine returns the low-level device attributes for the filing system device specified by *url.* The list of valid devices for a particular filing system is obtained from the *fslib_get_device_list* call.

fslib_get_device_list

int *fslib_get_device_list*(
const char **url*,
FSLIB_DEVICE **devlist*,
int *count*,
int *offset*)

The *fslib_get_device_list* routine returns a list of up to *count* devices into the supplied *devlist* structure from the list of devices controlled by the filing system manager addressed by *url. offset* is the number of entries to skip before starting to populate the return array. This enables an application to extract all the entries by making repeated calls using only a small buffer.

fslib_set_attr

int *fslib_set_attr*(
const char **url*,
FSLIB_ATTR **attr*)

The *fslib_set_attr* routine sets the attributes of the file or directory specified by *url* to the values given in the *attr* area. Exactly which parameters are updated depends on the underlying filing system.

fseek

int *fseek*(
FILE **fp*,
long *offset*,
int *ptrname*)

The *fseek* routine updates the current location of the file *fp* to the point *offset*, calculated relative to the specified *ptrname*: *SEEK_SET*, from the start of the file; *SEEK_CUR*, from the curent file position; or *SEEK_END* from the end of the file. This will determine from where the next data are read, or to where the next data will be written.

ftell

long *ftell*(
FILE **fp*)

The *ftell* routine returns the current location of the file *fp* as a byte-count from the start of the file.

ftruncate

int *ftruncate*(
FILE **fp*,
int *fpos*)

The *ftruncate* routine makes the file *fp* be *fpos* bytes long. The value of *fpos* must be less that or equal to the current file size.

link

int *link*(
const char **url_from*,
const char **url_to*,
int *type*)

The *link* routine creates a link in the filing system. The *url_to* parameter point to the file which is the final destination of the link. The *url_from* field is the new file to be created. The *type* field determines if this is a hard link *FS_TYPE_FILE*, a symbolic link *FS_TYPE_LINK* or a mount point *FS_TYPE_MOUNT* (RFS only).

mkdir

int *mkdir*(
const char **url*)

The *mkdir* routine creates a new directory in the filing system as specified by the *url* parameter.

readdir

int *readdir*(
FILE **fp*,
FSLIB_DIRENT **dir*,
int *count*)

The *readdir* routine returns up to *count* entries in the *dir* array from the current point in the directory opened by *fp*. This returns an external, formatted representation of the directory entries. Depending on the underlying filing system, it may be possible to access the raw binary format of the entries through regular mblk operations, but these results will be specific to the underlying filing system organisation.

readlink

int readlink(
const char *_url_from_,
char *_contents_,
int _len_)

The *readlink* routine returns the contents of the symbolic link pointed to by the URL *url_from*, i.e. the path to the file to which the link points. The value is returned in the *contents* array, and *len* specifies the maximum number of characters which are to be placed in this array.

rename

int *rename*(
const char *_url_from_,
const char *_url_to_)

The *rename* routine changes the name of the file *url_from* to be *url_to*. This operation may not span different filing systems. Depending on the underlying filing system, the operation may or may not work between different directories.

rewind

void *rewind*(
FILE *_fp_)

The *rewind* routine sets the file pointer for *fp* to the start of the file (offset 0).

rmdir

int *rmdir*(
const char *_url_)

The *rmdir* routine removes the (empty) directory specified by the *url* parameter.