Files MessageSet

The *Files* messageset contains a set of messages used to control the operation of modules implementing a filing system. Unlike the generic dataflow operations (e.g. OPEN) these messages assume a directory and filename system. The structures referenced within the messages are defined in the *fslib.h* header file within the FSLIB module.

Message Definitions

FSLIB_ATTR_GET

```
typedef struct
{
    char                    *name       file name
    struct fslib_attr_d     *attr       returned attributes
}ROME_T_FSLIB_ATTR_GET;
```

The *ATTR_GET* message requests the set of attributes for the file specified in the *name* URL. The exact results, in terms of the fields supplied in the structure, will depend on the underlying filing system.

FSLIB_ATTR_SET

```
typedef struct
{
    char                    *name       file name
    struct fslib_attr_d     *attr       attributes to change
}ROME_T_FSLIB_ATTR_SET;
```

The *ATTR_SET* message is used to change the attributes of the file specified in the *name* URL. The exact results, in terms of which fields in the structure are used, will depend on the underlying filing system.

FSLIB_DELETE

```
typedef struct
{
    char                    *name       file name
}ROME_T_FSLIB_DELETE;
```

The *DELETE* message requests that the destination process remove the file, and contents, specified by the *name* URL from the filing system.

FSLIB_DEV_ATTR

typedef struct
{

char	**devspec*	device name
struct fslib_device_attr_d	**attr*	returned device attributes

}**ROME_T_FSLIB_DEV_ATTR;**

The *DEV_ATTR* message requets the set of attributes associated with the device specified by the *devspec* URL. The exact results, in terms of the fields set in the returned structure, will depend on the underlying physical device.

FSLIB_DEV_GETROOT

typedef struct
{

char	**devspec*	device name
uint	*root1*	root block value 1
uint	*root2*	root block value 2
uint	*root3*	root block value 3
uint	*root4*	root block value 4

}**ROME_T_FSLIB_DEV_GETROOT;**

The *GETROOT* message requests that the four values stored in the root block of the device specified by the *devspec* URL be returned in the message. Currently, only devices running RFS support the retrieval of root block data.

FSLIB_DEV_LIST

typedef struct
{

char	**devspec*	sub-device name
int	*offset*	start position in list
int	*count*	number of devices to return
struct fslib_device_d	**devlist*	array of device structures

}**ROME_T_FSLIB_DEV_LIST;**

The *DEV_LIST* message requests the destination process to supply a list of its known filing system devices. To cope with potentially long lists, at most *count* values are returned in the reply after *offset* previous values in the list has been skipped. The *devspec* parameter may be used to specify which devices are to be returned (for example to restrict the NFS code to a particular machine).

FSLIB_DEV_PARTITION

typedef struct
{

char	**devspec*	device name
int	*count*	number of partitions returned
struct fslib_device_part_d	**part*	array of partition structures

}**ROME_T_FSLIB_DEV_PARTITION;**

The *DEV_PARTITION* message requests that the partition information for the device specified by the *devspec* URL be returned in the reply. The number of partition entries is set in the *count* variable. It is assumed that the array is large enough to for the highest number of partitions on a device.

FSLIB_DEV_SETROOT

typedef struct
{

char	**devspec*	device name
uint	*root1*	root block value 1
uint	*root2*	root block value 2
uint	*root3*	root block value 3
uint	*root4*	root block value 4

}**ROME_T_FSLIB_DEV_SETROOT;**

The *SETROOT* message requests that the four values specified in the message be stored in the root block of the device specified by the *devspec* URL . Currently, only devices running RFS support the setting of root block data. The meaning of the four values is transparent to the lower layer. It is expected that the same values will be used by a later call to *GETROOT.*

FSLIB_LINK

typedef struct
{

char	**name*	new file name
char	**dest*	destination of link
int	*type*	link type

}**ROME_T_FSLIB_LINK;**

The *LINK* message requests that a new entry be made in the filing system identified by the *name* URL. The contents of the file is set to the *dest* string, and the type of the file is set to the *type* field. As its name implies, the usual application of this message is to create a symbolic link in the a filing system.

FSLIB_MKDIR

typedef struct
{

char	**name*	directory name

}**ROME_T_FSLIB_MKDIR;**

The *MKDIR* message requests that a new directory be created in the filing system identified by the *name* URL.

FSLIB_READDIR

typedef struct
{

int	*count*	count of entries to return
struct fslib_dirent_d	**dir*	array of directory entries

}**ROME_T_FSLIB_READDIR;**

The *READDIR* message is used to retrieve entries from a filing system directory as an array of structures. This message provides a filing-system independent interface to scanning directories, but the exact results, in terms of the fields supplied in the data structures, will vary between filing systems. *count* specifies the maiumum number of entries to return. It is expected that this message is sent over a open *FILE* interface, as the usual operating mode is to move sequentially through the directory by sending multiple messages, and the *FILE* structure preserves the local context at the destination process.

FSLIB_READLINK

```
typedef struct
{
    char        *name       file (link) name
    char        *contents   link value
    int         len         length of contents
}ROME_T_FSLIB_READLINK;
```

The *READLINK* message requests that the contents of the symbolic link identified by the *name* URL be returned in the *contents* string. *len* is the maximum length of string that can be placed in the buffer.

FSLIB_RENAME

```
typedef struct
{
    char        *from       old file name
    char        *to         new file name
}ROME_T_FSLIB_RENAME;
```

The *RENAME* message requests that the name of the file identified by the *from* URL be changed to the *to* URL. Depending on the relative position of the two names in the filing system namespace this may move the file between directories. Renames across different filing systems are not supported by this message.

FSLIB_RMDIR

```
typedef struct
{
    char        *name       directory name
}ROME_T_FSLIB_RMDIR;
```

The *RMDIR* message requests that the directory identified by the *name* URL be removed from the filing system. Attempting to remove non-empty directories may or may succeed, depending on the underlying filing system.

FSLIB_SEEK

```
typedef struct
{
    int         offset      relative file position
    int         ptrname     position type
}ROME_T_FSLIB_SEEK;
```

The *SEEK* message is sent over an open *FILE* interface to move the file pointer to the position specified as *offset* from *ptrname.* This will determine the point to which subsequent data are written or from which subsequent data are read.

FSLIB_TELL

```
typedef struct
{
    int         fpos        file position
}ROME_T_FSLIB_TELL;
```

The *TELL* message is sent over an open *FILE* interface to request that the current file pointer position be returned. The returned value is the byte count from the start of the file.

FSLIB_TRUNCATE

typedef struct
{
    int *fpos* file position
}**ROME_T_FSLIB_TRUNCATE;**

The *TRUNCATE* message is sent over an open *FILE* interface driver to set the end-of-file marker for the file at *fpos* bytes from theis start of the file.