DOS

The DOS module implements the filing system layer for FAT12 and FAT16 DOS disks. It handles block allocation, directory traversal and operations, and operations on files themselves.

Caveat

The DOS module was written primarily as a way of getting data into and out of a ROME machine using its floppy drive. It has also been used for fixed ATA-type disks which were formatted and written on other systems. As such, a number of operations that would be found in a full MSDOS system are not present, in particular the ability to partition and format blank drives.

Although the code attempts to be 'safe' in its handling of removable media, it is possible to corrupt a disk by replacing media when the floppy drive motor is running. In contrast, floppy drives are re-scanned whenever the motor is spun up, which means that floppy disks operate very slowly in ROME systems.

Process Information

Prototype Name	dos
Link Order	does not matter
Process Name	usually "dos"

Configuration Command

Drive letters are associated with actual devices through a configuration command.

drive

drive *device* id:*letter*

The *drive* command associates a drive letter, *letter*, with a device, specified by its *device*, URL. The process specified by the device is opened at this point, but no attempt is made to access the physical medium. The root directory for the device can then be accessed as dos:/*letter*/

Example: "drive floppy:0 id:a"

Process Operation

The module has a queue handler and a main process. It accepts the *Standard* messageset for dataflow communications and the *Files* messageset for filing system operations.

CLOSE messages caused any buffered output to be written to the device, and the data structures associated with the file are freed.

EVENT replies are received for *MEDIA_EJECT* events from lower layer which support removable media (i.e. the floppy drive). Upon receiving an *EJECT* event the local *FAT* cache for the device is cleared and any files opened by *DOS* clients are marked as invalid.

FETMBLK/GETMBLK messages cause internally-buffered data to be transferred to the application's buffer or a new cluster to be read from the disk into local stoarage.

FLUSH messages force any buffered output to be written to the disk file.

FSLIB_ATTR_GET messages return the filing system attributes for the URL passed in the *name* parameter of the message.

FSLIB_ATTR_SET messages set the attributes for the URL passed in the *name* parameter. Currently only the *mode,* parameter is updated.

FSLIB_DELETE messages mark the specified file as deleted (by changing the first character of the filename to 0xe5 and freeing the FAT entries).

FSLIB_DEV_ATTR messages return the device attributes for the file system specified in the *devspec* parameter.

FSLIB_DEV_LIST messages return a list devices previously configured through *drive* commands, as above.

FSLIB_LINK messages are returned with an 'unsupported' error indicarion.

FSLIB_MKDIR messages create a directort in the filing system at the specified point.

FSLIB_READDIR messages retrieve directory entries from the open directory.

FSLIB_RENAME messages use change the name of the final component of the file. Renaming a file across different directories is *not* supported.

FSLIB_RMDIR messages delete the specified (empty) directory.

FSLIB_SEEK messages set the file position which will be used to read or write data on subsequent calls.

FSLIB_TELL messages return the current file position.

FSLIB_TRUNCATE messages return an 'unsupported' error.

OPEN messages cause the filing system to be initialised if this is the first *OPEN* on that device (or since the last *MEDIA_EJECT* event). The filign system type and FAT format are determined. Then the filing system is traversed to locate the directory point, and the terminal component is created, if necessary.

OUTMBLK/PUTMBLK messages transfer data to the disk file.

NEWMBLK messages return buffers for applications to use

RETMBLK messages just free the associated buffers.