GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	storage/filesystem/source/FileSystem.cpp	**Lines:**	3	74	4.1 %
Date:	2021-05-06 12:39:05	**Branches:**	0	18	0.0 %

Line	Branch	Exec	Source
1			/* mbed Microcontroller Library
2			* Copyright (c) 2006-2013 ARM Limited
3			* SPDX-License-Identifier: Apache-2.0
4			*
5			* Licensed under the Apache License, Version 2.0 (the "License");
6			* you may not use this file except in compliance with the License.
7			* You may obtain a copy of the License at
8			*
9			* http://www.apache.org/licenses/LICENSE-2.0
10			*
11			* Unless required by applicable law or agreed to in writing, software
12			* distributed under the License is distributed on an "AS IS" BASIS,
13			* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14			* See the License for the specific language governing permissions and
15			* limitations under the License.
16			*/
17			
18			#include "filesystem/Dir.h"
19			#include "filesystem/File.h"
20			#include "filesystem/FileSystem.h"
21			#include <errno.h>
22			
23			namespace mbed {
24			
25		4	FileSystem::FileSystem(const char *name)
26		4	: FileSystemLike(name)
27			{
28		4	}
29			
30			int FileSystem::reformat(BlockDevice *bd)

```
{
    return -ENOSYS;
}

int FileSystem::remove(const char *path)
{
    return -ENOSYS;
}

int FileSystem::rename(const char *path, const char *newpath)
{
    return -ENOSYS;
}

int FileSystem::stat(const char *path, struct stat *st)
{
    return -ENOSYS;
}

int FileSystem::mkdir(const char *path, mode_t mode)
{
    return -ENOSYS;
}

int FileSystem::statvfs(const char *path, struct statvfs *buf)
{
    return -ENOSYS;
}

int FileSystem::file_sync(fs_file_t file)
{
    return 0;
}

int FileSystem::file_isatty(fs_file_t file)
{
    return false;
}

off_t FileSystem::file_tell(fs_file_t file)
```

```
71  {
72      return file_seek(file, 0, SEEK_CUR);
73  }
74
75  void FileSystem::file_rewind(fs_file_t file)
76  {
77      file_seek(file, 0, SEEK_SET);
78  }
79
80  off_t FileSystem::file_size(fs_file_t file)
81  {
82      off_t off = file_tell(file);
83      off_t size = file_seek(file, 0, SEEK_END);
84      file_seek(file, off, SEEK_SET);
85      return size;
86  }
87
88  int FileSystem::file_truncate(fs_file_t file, off_t length)
89  {
90      return -ENOSYS;
91  }
92
93  int FileSystem::dir_open(fs_dir_t *dir, const char *path)
94  {
95      return -ENOSYS;
96  }
97
98  int FileSystem::dir_close(fs_dir_t dir)
99  {
100     return -ENOSYS;
101 }
102
103 ssize_t FileSystem::dir_read(fs_dir_t dir, struct dirent *ent)
104 {
105     return -ENOSYS;
106 }
107
108 void FileSystem::dir_seek(fs_dir_t dir, off_t offset)
109 {
110 }
```

```
111
112 off_t FileSystem::dir_tell(fs_dir_t dir)
113 {
114     return 0;
115 }
116
117 void FileSystem::dir_rewind(fs_dir_t dir)
118 {
119     // Note, the may not satisfy rewind on all filesystems
120     dir_seek(dir, 0);
121 }
122
123 size_t FileSystem::dir_size(fs_dir_t dir)
124 {
125     off_t off = dir_tell(dir);
126     size_t size = 0;
127     struct dirent *ent = new struct dirent;
128
129     dir_rewind(dir);
130     while (true) {
131         int res = dir_read(dir, ent);
132         if (res <= 0) {
133             break;
134         }
135
136         size += 1;
137     }
138     dir_seek(dir, off);
139
140     delete ent;
141     return size;
142 }
143
144 // Internally used file wrapper that manages memory on close
145 template <typename F>
146 class Managed : public F {
147 public:
148     virtual int close()
149     {
150         int err = F::close();
```

```
151                 delete this;
152                 return err;
153             }
154         };
155
156         int FileSystem::open(FileHandle **file, const char *path, int flags)
157         {
158             File *f = new Managed<File>;
159             int err = f->open(this, path, flags);
160             if (err) {
161                 delete f;
162                 return err;
163             }
164
165             *file = f;
166             return 0;
167         }
168
169         int FileSystem::open(DirHandle **dir, const char *path)
170         {
171             Dir *d = new Managed<Dir>;
172             int err = d->open(this, path);
173             if (err) {
174                 delete d;
175                 return err;
176             }
177
178             *dir = d;
179             return 0;
180         }
181
182         } // namespace mbed
```