```
.\" Modified from man(1) of FreeBSD, the NetBSD mdoc.template, and
mdoc.samples.
.\" See Also:
.\" man mdoc.samples for a complete listing of options
.\" man mdoc for the short list of editing options
.\" /usr/share/misc/mdoc.template
.Dd 12/2/2023
.Dt Project 3
.Sh Project 3: File Systems
.Sh SYNOPSIS
.Nm
Develop library (libWad) to read/write from WAD files, create
directory/file structure.

Implement daemon via FUSE to test/access mounted directory structure
.Sh DESCRIPTION
All the steps are as follows:
.Sh Library
.Bl

Complied using Makefile:
.Pp
.Op libWad: Wad.cpp Wad.h
.Pp
.Op g++ -c -g Wad.cpp -o Wad.o
.Pp
.Op ar cr libWad.a Wad.o
.Pp


Filesystem implemented w/unordered_map (Key=path, Value=*object) allowing
accessibility to information w/filepath.

Descriptor object contains name, offset, length, isDirectory, filePath and
vector of children *objects that form an n-ary tree.
.Pp

-
.Pp
.Op string getMagic();
.Pp
Returns magic, read from loadWad()
.Pp

-
.Pp
.Pp
.Op bool isContent(const string &path);
.Pp
Returns true if object in map at path is file, else false
.Pp

-
.Pp
```

.Pp
.Op bool isDirectory(const string &path);
.Pp
Returns true if object in map at path is directory, else false
.Pp

-
.Pp
.Pp
.Op int getSize(const string &path);
.Pp
Returns read-in file size if object in map at path is file, else -1
.Pp

-
.Pp
.Pp
.Op string pathMaker(deque<Descriptor *> paths);
.Pp
Implements deque for .pop/.push/.front functionalities to create filepaths (not ending in /)
Given deque has all paths in order ex: [/, Gl, ad, os, cake.txt], and it's called on file cake.txt, returns "/Gl/ad/os/cake.txt"
.Pp

-
.Pp
.Op void dfsDirectory(Descriptor *root, fstream &file);
.Pp
Implements recursive dfs through n-ary tree to write all files. Uses regex to check name, and if directory and not E#M# adds _start/_end when complete.
Recursively calls function for the number of children in vector!
.Pp

-
.Pp
.Op static Wad *loadWad(const string &path);
.Pp
Given .wad, reads 12 header bytes (magic, offset, buffer)

Initializes rootDescriptor(/), looping through wad file descriptors.

Uses regex to check descriptor name cases (E#M#, _START, _END), creates paths using .pathMaker
.Pp
E#M#: push descriptor to parent vector & deque, add next 10 descriptors to vector
.Pp
_START: push descriptor to parent vector and onto deque (for file path)
.Pp
_END: pops back of deque (for file path)
.Pp

-
.Pp
.Op int getDirectory(const string &path, vector<string> *directory);
.Pp
Checks if path exists in map & .isDirectory.

Success: loops through children vector, pushing names into the directory vector, returns #children

Failure=-1
.Pp

-
.Pp
.Op int Wad::getContents(const string &path, char *buffer, int length, int offset)
.Pp
Check if path exists & .isContent

Parameters describe how to read lump data.
.Pp
If offset >= lump size, return 0
.Pp
Else, move to new offset location and read into buffer by:
.Pp
If lumpSize > length + offset, can read all length
.Pp
If lumpSize = length + offset, can read all lumpData
.Pp
Else, length goes over lumpSize, reading (lumpSize - offset)
.Pp
Adds /0 to buffer and returns #bytes read
.Pp

-
.Pp
.Op void createDirectory(const string &path); void createFile(const string &path);
.Pp
Parses path into existing path, newDirectory/File, parentDirectory using REGEX
.Pp
EDGE CASES:
.Pp
BOTH: new and parent can't be E#M#, existingPath must be in map, parent must be a directory, new path can't already exist
.Pp
Directory: name = 1-2 characters
.Pp
File: name <= 8 characters
.Pp
Fail=return
.Pp
Success:

Creates newDescriptor node, pushes it into parentDirectory children vector, new path into the map, calls on dfsDirectory to write all descriptors based on the new map at the descriptor offset.

Increases descriptor count in the header
.Pp

-
.Pp
.Op int Wad::writeToFile(const string &path, const char *buffer, int length, int offset)
.Pp
If path points to empty file in map:

Updates files data (offset/length)

Writes data to the end of the lump, calls dfsDirectory to print entire updated map into wad in new space (offset + length)

Updates descriptor offset in the header to reflect shift.

Success=#bytesRead

Fail=-1

.Sh Daemon

Complied using Makefile:
.Pp

-
.Pp
wadfs: FuseDaemon.cpp
.Pp
g++ -D_FILE_OFFSET_BITS=64 -DFUSE_USE_VERSION=26 FuseDaemon.cpp -o wadfs -lfuse -L ../libWad -lWad -w
.Pp
From discussion
.Pp
Permissions:
.Pp
sudo chmod 666 /dev/fuse
.Pp
.Pp
Mounting:
.Pp
./wadfs/wadfs -s sample1.wad ./mountdir

.Pp

-
.Pp
.Op int main(int argc, char *argv[])

.Pp
Set wadPath to passed in .wad, converts relative paths to absolute paths w/get_current_dit_name, loads .wad
.Pp

-
.Pp
.Op argv[argc - 2] = argv[argc - 1];

.Pp
.Op argc--;
.Pp
Augments cmd arguments by moving the mountingDir at position 2 to match fuse requirements
.Pp

-
.Pp
.Op return fuse_main(argc, argv, &operations, myWad);
.Pp
Callback functions in operations struct, myWad passed in to access wad object in operation functions

Wad functions accessible via:

(Wad *)fuse_get_context()->private_data
.Pp

-
.Pp
.Op getattr_callback(const char *path, struct stat *stbuf)
.Pp
Check 1: path=directory (.isDirectory), declares directory.

Check 2: path=file (.isContent), declare file of explicit size (.getSize())

Success=1, failure=-ENOENT
.Pp

-
.Pp
.Op static int read_callback(const char *path, char *buf, size_t size, off_t offset, struct fuse_file_info *fi)
.Pp
Check 1: path=file (.isContent)

Check 2: reads file details (.getContents)

Success=(.getContents return_val), failure=-ENOENT
.Pp

-
.Pp

```
.Op static int readdir_callback(const char *path, void *buf,
fuse_fill_dir_t filler, off_t offset, struct fuse_file_info *fi)
.Pp
Calls on (.getDirectory) filling buf with returned files (success=0,
failure=-ENOENT)
.Pp

-
.Pp
.Op static int do_mkdir(const char *path, mode_t mode)
.Pp
If directory/path doesn't already exist, call .createDirectory to create
directory/return 0
.Pp

-
.Pp
.Op static int do_mknod(const char *path, mode_t mode, dev_t rdev)
.Pp
If directory/path doesn't already exist, call .createFile to create
file/return 0
.Pp

-
.Pp
.Op static int do_write(const char *path, const char *buffer, size_t size,
off_t offset, struct fuse_file_info *info)
.Pp
Calls .writeToFile writing content in file, success=.writeToFile return,
failure=-errno

.Sh TESTING
Ran sudo ./run_libtest to test the library functionality

Ran mounting as described above to further assess functionality through
the command line (also tests DAEMON)

.Sh BUGS
n/a

.Sh LINK
https://youtu.be/hKBD_acDPDk
.Sh CITATIONS
DAEMON:

https://www.cs.nmsu.edu/~pfeiffer/fuse-tutorial/html/
https://engineering.facile.it/blog/eng/write-filesystem-fuse/
https://maastaar.net/fuse/linux/filesystem/c/2019/09/28/writing-less-
simple-yet-stupid-filesystem-using-FUSE-in-C/

.Pp
REGEX:

https://regexr.com/
```

```
.Pp
DEQUE:

https://www.geeksforgeeks.org/deque-cpp-stl/

.Pp
GENERAL:

https://chat.openai.com/

.Sh AUTHOR
Ivan Saldarriaga
.\" .Sh BUGS
.\" .Sh HISTORY
```