.TH COP4600: P2
.SH Name
P3 - File Systems
.SH Synopsis
Implement a userspace filesystem daemon using the FUSE (Filesystem in UserSpacE) API to access data in WAD format, the standard used in a number of classic PC game titles (including DOOM and Hexen)

.SH Files changed and added
/home/reptilian/P3/libWad/Wad.cpp
/home/reptilian/P3/libWad/Wad,h
/home/reptilian/P3/wadfs/wadfs.cpp

.SH Steps taken to accomplish assignment:

.B Step(1) - Wad.h
Start by making Wad.h and Wad.cpp. define a structure for the project to follow. I used an n-ary tree to store my information in the Node struct (name, length, subsequent children, and information). Then made a public Node called root where the tree would be held.
.PP

.B Step(2) - Wad.cpp: loadwad
use ifstream infile to read the path location. The header contains the file magic (4B), descriptor count(4B), and location (descriptor offset). we read magic and store that data into thw wad object variable 'magic' because it is a string we can just read it in as is. we then read in the descriptor count and location which are ints entered as chars so we bit shift to convert to int. then use seekg to start at the offset and at the beginning with std::ios::beg. A Node is created called root and the name of root is set to '/' and we create a stack to candle our n-ary tree and push the root on the stack. We then enter the for loop (for the duration of the number of descriptors). we read in the element offset (4B), length(4B), and name (8)and increment by 16 (4+4+8=16).if the length is 0 and we find "_start" then we are at a namespaceâ□□s beginning and create a subsequent node, set root to top, and add the new node to the stack. if we identify "E#M#" we do the same, but set root to the new not and push at the start, we also increment by 10 because markers are followed by ten (10) map element descriptors. if we find "_end" we pop our stack and set root to top. if our iterarion is less than count, a file is in E#M# and we read that, move forward with seekg, create a node, then seekg again to push past the element (set root to top if i is the count). we then close the file and return wad
.PP

.B Step(3) - retrieve/construct
get the node at a path. use root and path to iterate the tree till the location is found and returns it. Construct makes a new node and sets the relative name, data, and length.
.PP

.B Step(4) - magic
returns magic

.PP

.B Step(5) - isContent/isDirectory
retrive the node at the given path, if the node exists and has no children it is a directory. if the node is Nullptr and has children it is content
.PP

.B Step(6) - getContent/getDirectory
if Content/Directory exists get the entries
.PP

.B Step(7) - wadfs
getattr callback reads the metadata of a given path. â□□st_gidâ□□ represents the "owner group of the files/directories", its the same group as the user who mounted the filesystem. If the wad is a directory we set permissions to 555 (everyone can read, and execute but can't write. If the wad is a content we set permissions to 444 (everyone can, can read, can't write and can't execute.) If the entry is directory, st_mode set to S_IFDIR and st_nlink to 2, while if itâ□□s a file, st_mode set to S_IFREG (regular file) and st_nlink to 1. files muat have the st_size specified. read_callback: if iscontent, get the contents or return -ENOENT(nothing exists at the given path). readdir_callback tells FUSE the structure of the accessed directory. "return its representation, we are doing it by filling buf with the two links for the upper directory .. and current directory . and with the only file we have: file". then we configure arguments with the implemented FUSE fuse_operations object.
.PP

.SH Testing
I stuck to the main tests with a few alterations to adjust for certain specifications; for instance, used sample1 2 and doom.
.PP

.SH Problems
had issues getting the directory to build correctly. it was an issue in my n-arry tree where I looked at my element instead of the element length in my loadwad for loop.
.PP

.SH Sources
https://www.maastaar.net/fuse/linux/filesystem/c/2016/05/21/writing-a-simple-filesystem-using-fuse/
https://engineering.facile.it/blog/eng/write-filesystem-fuse/
https://www.cyotek.com/blog/reading-doom-wad-files
.PP

.SH Video link
https://youtu.be/sOangFEdvaY
.PP