```
1
2   class File {
3   public:
4     //inode metadata;                    // for use by all files
5     struct stat metadata;                // for use by all files
6     // one or the other of the following (data and frame) should be empty.
7     string data;                         // for use by regular files
8     vector<dirent_frame> dentries;       // for use by directories
9     //list<dirent_frame> dentries;       // it'd be cleaner to make this a list
10  };
11
12
13  class Ilist {
14  public:
15    int count;
16    map<ino_t,File> entry;
17    int next() {
18      static int count = 2;  // ino counter stats at 2.
19      return count++;
20    }
21  } ilist;
22
23
24  Implemeted and seem okay
25  void initialize()
26  int my_lstat( const char* path, struct stat *statbuf )
27  int my_fstat( ino_t fh, struct stat* statbuf )
28  int my_mkdir( const char *path, mode_t mode )
29  MY_DIR* fopendir( ino_t fh )
30  dirent* my_readdir( MY_DIR* dirp )
31  int my_closedir( MY_DIR* dirp )
32  MY_DIR * my_opendir( char path[PATH_MAX] )
33
34  Implemented and being debugged.
35  int my_mknod( const char *path, mode_t mode, dev_t dev ) // being debugged
36
37  Implemented but untested
38  int my_link(const char *path, const char *newpath)
39
40  Helpers (implemented and seem fine)
41  int ls(string s)
42  ino_t lookup( string name, ino_t fh )
43  ino_t find_ino( string path )
44  File* find_file( ino_t ino )
45  File* find_file( string s )
46  void show_stat( struct stat& root ) // displays a stat as Pfeiffer does.
47  split(const string s, const string pat )
48  join( const vector<string> v, const string pat, int start=0, int end=-1 )
49
50
51  ============================================================
52
53  High priority (Get these working first)
54  int my_rename( const char *path, const char *newpath )
55  // changes name on a dentry and/or moves it to another directory
56  int my_link(const char *path, const char *newpath)  // needs testing/debugging
57  // my_link creates hard links, i.e., puts dentries into directories.
58  int my_unlink( const char *path ) // must destruct file when nlink == 0.
59  int my_creat( const char *fpath, mode_t mode )  // see my_mkdir for ideas.
60  int my_open( const char *path, int flags ) // with certain flags,
61  // open() must create a file, say via my_creat()
62  int my_close( int fh )
63  int my_pread( int fh, char *buf, size_t size, off_t offset )
64  int my_pwrite( int fh, const char *buf, size_t size, off_t offset )
65
66  Medium priority
67  int my_access( const char *fpath, int mask ) // check bits of mode field
68  int my_rmdir( const char *path ) // erase a dirent_frame from dentries
69  int my_chown(const char *path, uid_t uid, gid_t gid)
70  int my_chmod(const char *path, mode_t mode)
71  // changes permission bits in the metadata's mode field
72
73  Low priority (note that we won't implement symbolic links for now.)
```



```
74  int my_readlink( const char *path, char *link, size_t size )
75  int my_symlink(const char *path, const char *link)
76  int my_utime(const char *path, struct utimbuf *ubuf)
77  int my_fdatasync( ino_t fh )
78  int my_fsync( ino_t fh )
79  int my_ftruncate( ino_t fh, off_t offset )
80  int my_truncate(const char *path, off_t newsize)
81  int my_statvfs(const char *fpath, struct statvfs *statv)
82
83  Very low priority
84  int my_lsetxattr( const char *fpath, const char *name, const char *value, size_t
     size, int flags )
85  int my_lgetxattr( const char *fpath, const char *name, char *value, size_t size,
     int flags )
86  int my_llistxattr( const char *path, char *list, size_t size )
87  int my_lremovexattr( const char *path, const char *name )
88
89
90
91
```