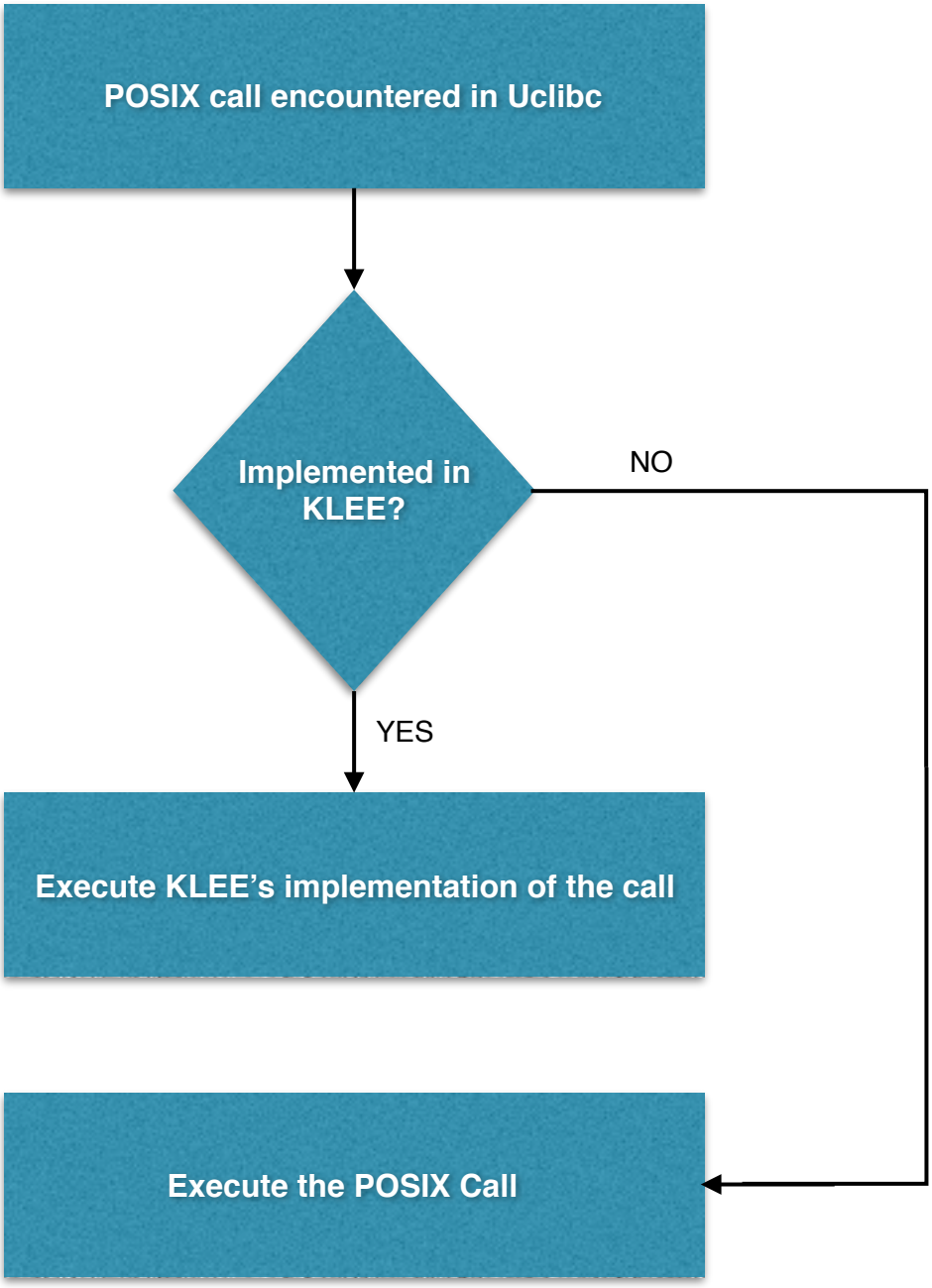
How does KLEE Works

I. KLEE defines a number of system calls inside its own files (can be found under KLEE/runtime/POSIX
II. A copy of Uclibc is made (called Klee-uclibc).
III. The basic idea is to bypass those system calls that are defined inside KLEE (and calling KLEE's version instead). For this to happen, we make changes to KLEE-UCLIBC so that the system call is bypassed.



Example

POSIX Call: fopen
Files: fopen.c, _fopen.c, __syscall_fcntl, KLEE/runtime/POSIX (fd.c, fd_32.c, fd_64.c)
Notation: green color is the uclibc call while black is klee-uclibc's

1.
fcntl(filedes, F_SETFL, O_APPEND))
__libc_fcntl(filedes, F_SETFL, O_APPEND))

2.
stream->__filedes = open(((const char *) fname_or_mode),open_mode, 0666)
stream->__filedes = __libc_open(((const char *) fname_or_mode), open_mode, 0666)

1. **__libc_fcntl:** calls KLEE's fcntl (defined inside fd.c). fcntl inside KLEE performs the following operations:

A. Gets the file using __get_file
B. In case of no symbolic file returned, make the system calls (__NR_fcntl) which is as per my understanding processor dependent)

2. **__libc_open**: calls KLEE's open() function which in turn calls__fd_open(const char *pathname, int flags, mode_t mode). Below is what __fd_open basically does:

A. If the max number of files (MAX_FDS) have already been opened, then it returns error
B. It tries to access the file using fd and the file is returned if it exists (and the file is truncated i.e. contents of the file are removed)
C. Then a pointer is requested for the symbolic file structure
D. If file is created successfully, tIt checks for the following conditions:
   A. Flags
      A. If O_CREAT and O_EXCL are set, the operation fails (Reason: If O_CREAT and O_EXCL are set, open() shall fail if the file exists)
      B. if O_TRUNC and O_RDONLY are set, the operation fails (Reason: The result of using O_TRUNC with O_RDONLY is undefined, so we return error)
      C. if O_EXCL is set and O_CREATE is not set, the operation fails (Reason: The result of using O_EXCL without O_CREAT is undefined, so we return error)
   B. Permissions
   C. st_mode
E. If file cannot be created, then we try to make a system call and create the file using there (concrete one I guess). If it is successful, we keep reference to the file else we return error
F. Finally, the flags are set on the file (Readable or Writable or both)