1. Explain how your open file table design supports the fork semantics.

We implemented the fork semantics in thread\_fork() and copies the current file table into the forked thread. In more detail, we retrieve the current and new thread’s file table, and check if current file table exists. If so, we malloc a new file table for new thread if it doesn’t already exist, and start copying every page entry into the new thread file table, if not, the new file table does not exist as well.

1. Explain your implementation of sfs\_getdirentry.

We are trying mimic the vop implementation.

1. Explain what changes you needed to make to use wasted inode space to store file data.

/\* not implemented\*/

1. Discuss the single biggest challenge you had to address for the assignment

Searching through and reading the code for important predefined structs and methods was challenging and also grasping the design of the system before we could implement any functions. The codebase is large and identifying what is already predefined that could be useful is hard to do, since many were separated into numerous different files. Nearly all functions were unable to be started until the entire context was identified and learned.

We did not build off of A2 and have no waitpid implemented. As a result, the badcall for lseek does not completely work. All badcall tests succeed except for the stdin test and all other syscalls succeed with badcall.