open_close_lseek.c

int open_file (char *filename, int mode)
// ask for mode in main
First, I get the inonumber for the pathname and load the contents into a MINODE. (If the file doesn't exist, I create it first, then load into MINODE.) I create an OpenFileTable entry and point it at the MINODE. Then, I check to make sure the MINODE is a regular file, and not already opened for any writing modes. (I mark the MINODE as dirty if opened already for an incompatible mode to help with checking.) I then set file size based on the mode and point the lowest open file descriptor (FD) in the running process to the OpenFileTable entry I set earlier. I touch time and if open for any writing mode, I mark the MINODE as dirty and return the open file descriptor number.

int truncate (MINODE *mip)
// called when file is opened for writing mode
First, releases all data blocks in the MINODE, then changes size, updates the time fields and marks MINODE as dirty.

int close_file (int* fd)

First, I verify that the given file descriptor is within the range and pointing to a valid OpenFileTable entry in the running->fd array. Next I allocate a local OpenFileTable entry and point it to the file descriptor in the running->fd array. Then, I set running->fd[fd] to 0 (deallocate) and decrement refcount and index. I check if this process is the last user of the file, and if so I call iput to write the MINODE back to the disk.

int my_lseek(int fd, int position)

First, I record the original position as the running processes open file descriptors current offset position. I check to make sure the user supplied position is within the bounds of the open file. I set the open file's new offset to the user given position (assuming it's in range), and return the original position.

int pfd()

In this function, I loop through all possible file descriptors. If the running file descriptor is opened, I print out the file descriptor number, mode opened for, offset position, and INODE device + ino number. This function is used as more of a helper function to allow the user to know which file is currently opened, and in which mode.

int dup(int fd)

I begin by verifying the supplied file descriptor is opened. Next, I run through all file descriptors and copy the open fd into the first empty fd in the running process. Lastly, increment OFT's refcount by 1.

int dup2(int fd, int gd)

First, I check if the supplied gd coorespondes to an opened file in the running->fd array. If so, I close the file. Lastly, I copy the running->fd[fd] into the running->fd[gd] in the open file descriptors array of the current running process.