Link_Unlink.C

int link(char *old_file, char *new_file)

First, I get the ino number of the old file w/ getino to check != 0 making sure old_file actually exists. If the ol_file exists, I load it into an INODE (Omip) and make sure old_file is not a dir. I also call getino(newfile) to make sure that new_file doesn't yet exist. I use dirname function to get new_file's parent and call getino(parent) to help load an INODE (Pmip) using iget. I call enter_name (Pmip, Oino, child) to enter child entry into the data block of Pmip. I increment parent links count to show new file and mark as dirty before calling iput (Pmip) //Parent minode and iput (Omip) //child minode

int unlink(char * filename)

First, I get file ino number with getino and make sure the file exists. Then, I use iget to load the file's minode and make sure the mode is not type DIR. I break the filename into Parent and child so I can get the parents ino number and minode as well as the basename as the child. I set isunlink to true and call rm_child (Pmip, child) to remove child name from Parent directory with correct print statements. Then change back to false. I mark Parent minode as dirty and put it back with iput(Pmip); Decrement inodes link count by 1. then deallocate all data blocks with bdalloc and the inode (mip) with idalloc. Then, call iput (mip) to release the Mip for the file we just unlinked.