```c
#include "type.h"

/*
get ino of source
get parent minode of dest
make new entry in parent inode named the child of dest
set that dest dir entry's ino to source ino
*/

void my_link(char *path)
{
        int ino;
        int p_ino;
        char file1[64], file2[64], temp[64];
        char link_parent[64], link_child[64];

        MINODE *mip;
        MINODE *p_mip;
        INODE *ip;
        INODE *p_ip;

        if(!strcmp(path, ""))//checks to see if user entered a file to be linked
        {
                printf("error, no first file entered.\n");
                return;
        }

        if(!strcmp(third, ""))//checks to see if user entered a link name
        {
                printf("Error, no second file entered.\n");
                return;
        }

        strcpy(file1, path);//copies file that will be linked into file1
        strcpy(file2, third);//copies file2 link into file2

        ino = get_Inode(running->cwd, file1);//gets the inode of the file we want to
link to
        mip = iget(dev, ino);//gets the memory inode of the link file

        if(!mip)//checks to see if the file even exists
        {
                printf("Error, %s does not exist!\n", file1);
                return;
        }

        if(S_ISDIR(mip->INODE.i_mode))//checks to see whether or not we tried to link
to a directory
        {
                printf("ERROR: Can't link a directory!\n");
                return;
        }

        if(!strcmp(file2, "/"))//if the link name is root then copy that into link
parent
                strcpy(link_parent, "/");

        else//copy the dirname of file2 into link parent
        {
                strcpy(temp, file2);
                strcpy(link_parent, dirname(temp));
        }
        strcpy(temp, file2);
        strcpy(link_child, basename(temp));//copies the basename of the link name
into link child

        p_ino = get_Inode(running->cwd, link_parent);//get the parent inode
        p_mip = iget(dev, p_ino);//get the parent memory inode

        if(!p_mip)//checks to see whether or not the parent inode exists
        {
                printf("Error, no parent exists.\n");
                return;
        }


        if(!S_ISDIR(p_mip->INODE.i_mode))//checks to see whether or not the parent is
a directory
        {
                printf("Error, not a directory.\n");
                return;
        }


        if(get_Inode(running->cwd, file2))//checks to see whether or not the child
already exists
        {
                printf("Error, %s already exists.\n", file2);
                return;
        }

        enter_name(p_mip, ino, link_child);//enters the name of the link into the
parent memory inode

        ip = &mip->INODE;//points the inode pointer to the node that the memory inode
points at


        ip->i_links_count++;//updates the link count of the inode pointer
        mip->dirty = 1;//sets the memory inode dirty to 1
        p_ip = &p_mip->INODE;//points the parent inode pointer the parent memory
pointer
        p_ip->i_atime = time(0L);//updates the access time on the parent inode
        p_mip->dirty = 1;//sets the parent inoe dirty to 1

        iput(p_mip);//disuper_poses of parent memory pointer
        iput(mip);//disuper_poses of child memory inode pointer
        return;
}
```