```
#include "type.h"

/*
Frees the fd in the argument from running->OFT[fd], decrements the refcount
*/

void close_file(int fd)
{
        int i;
        OFT *file_p;


        if(fd < 0 || fd >= NFD)//Checks if the file descriptor is valid
        {
                printf("Error, the file descriptor out of range.\n");
                return;
        }


        for(i = 0; i < NOFT; i++)//Checks if the file descriptor is in the open file
table
        {
                file_p = &OpenFileTable[i];//accesses the open file table and sets it
to the file pointer

                if(file_p->inodeptr == running->fd[fd]->inodeptr)//checks to see if
the file pointer points at the same inode as the running process's file descriptor
                        break;

                if(i == NOFT - 1)//reached the end of the open file table meaning
that the file does not exist in the table
                {
                        printf("Error, the file is not in the OpenFileTable.\n");
                        return;
                }
        }


        file_p = running->fd[fd];//sets file_p to the correct fd from the running
proccess
        running->fd[fd] = NULL;//sets the running PROC's file descriptor at index fd
to null


        file_p->refCount--;//Ensures that the minodes are running as should be

        if(file_p->refCount == 0)//checks to see if the file pointer's ref count is
ero, if it is send it to iput to be disuper_posed
                iput(file_p->inodeptr);//calls iput on the file pointer's inode
pointer

        return;
}

//Calls close file for the given fd
void my_close(char *path)
{
        int fd;

        if(!path)//checks to see if a file was give
        {
                printf("Error, there was no file name given.\n");
                return;
        }

        fd = atoi(path);//converts the pathname to int
        close_file(fd);//sends the file descriptor to close file
        return;
}
```