    /\*

    Author: Utsav Krishnatra

    Student Number: 110095341

    Subject: Advanced Systems Programming

     submitted to Dr. Prashanth Ranga

\*/

#define \_XOPEN\_SOURCE 500

#define Max 4096

#include <sys/stat.h>

#include <unistd.h>

#include <ftw.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char \*srcPath;

char \*destPath;

int will\_it\_move;

int num\_of\_exts;

char \*\*exts;

void createFileDir(const char \*src, const char \*dest) {

      //variable to get status of the file such as name,type, size.

  struct stat stat\_f\_src;

    //gettting the status of the file

  stat(src, &stat\_f\_src);

   //checks if this is a regular file or not

  if (S\_ISREG(stat\_f\_src.st\_mode)) {

    //create a file at the destination if source is a regular file

    //rb to read binary data

    FILE \*filePointerSrc = fopen(src, "rb");

      //wb to write as binary data

    FILE \*filePointerDest = fopen(dest, "wb");

    //buffer for read and write operations

    char buffer[4096];

    //stores the number of bytes read

    size\_t readNumberOfBytes;

    while ((readNumberOfBytes = fread(buffer, 1, sizeof(buffer), filePointerSrc)) > 0) {

      fwrite(buffer, 1, readNumberOfBytes, filePointerDest);

    }

    fclose(filePointerSrc);

    fclose(filePointerDest);

  } else if (S\_ISDIR(stat\_f\_src.st\_mode)) {

       // If WE FIND DIRECTORY AT SOURCE, THEN CREATE A DIRECTORY AT DESTINATION in read write and execute mode of user group and others

    mkdir(dest, 0777);

  }

}

void recursiveDirectoryCreation(const char \*path) {

     //creates a duplicate of the path

  char \*cpy = strdup(path);

  for (char \*ptr = cpy + 1; \*ptr != '\0'; ptr++) {

    if (\*ptr == '/') {

          //after first slash, looking for the next slash(or the sub file or directory). Creating corresponding file or directory in the destination folder

      \*ptr = '\0';

        // printf("%s\n", copy);

      //creating the file in read, write and execute mode for user,group and others.

      mkdir(cpy, 0777);

           //replacing null character back with the  slash character to detect further slashes in the oath

      \*ptr = '/';

    }

  }

   //freeing up the space occupied during the dynamic allocation while creating copy of the pointer to character array path

  free(cpy);

  mkdir(path, 0777);

}

void mkFileDirR(const char \*src, const char \*dest) {

  char \*cpy = strdup(dest);

  for (char \*p = cpy + 1; \*p != '\0'; p++) {

    if (\*p == '/') {

      \*p = '\0';

      recursiveDirectoryCreation(cpy);

      \*p = '/';

    }

  }

  free(cpy);

   //callling code to create file/directory

  createFileDir(src, dest);

}

//this function will get extensions for all the extensions given in the command line to be excluded

char\* getExtn(char\* path) {

    char \*fileName = strrchr(path, '/');

    char \*dots = strrchr(fileName, '.');

    char \*slash = strrchr(fileName, '/');

    if (dots == NULL || strlen(dots) > strlen(slash)) {

        return "";

    } else {

        return dots;

    }

}

int cpRmv(const char \*fpath, const struct stat \*sb, int typeflag, struct FTW \*ftwbuf) {

       //this bool value indicates whether it is a file or a directory

    int mustCreate = 0;

    if (num\_of\_exts == 0) {

        mustCreate = 1;

    } else if (S\_ISREG(sb->st\_mode)) {

        for (int i = 0; i < num\_of\_exts; i++) {

              //walk through the list of exts

            int LengthOfExtn = strlen(exts[i]);

            int path\_len = strlen(fpath);

            //find whether the extension is there in the extention list or not

            if (strcmp(fpath + path\_len - LengthOfExtn, exts[i]) == 0 && \*(fpath + path\_len - LengthOfExtn - 1) == '.') {

                mustCreate = 1;

            }

        }

    } else {

            //this is to indicate that we have to create a directory

        mustCreate = 1;

    }

    if (mustCreate) {

        char \*tmp\_path = strdup(fpath);

        int src\_len = strlen(srcPath);

        tmp\_path = tmp\_path + src\_len;

        char path[strlen(tmp\_path) + strlen(destPath)];

        strcpy(path, "");

        strcat(path, destPath);

        strcat(path, tmp\_path);

        mkFileDirR(fpath, path);

         //remove only if move op is selected

        if (will\_it\_move) {

            int result = remove(fpath);

        }

    }

    return 0;

}

int ifDirectory(const char\* fileName) {

    struct stat path;

    stat(fileName, &path);

    return S\_ISDIR(path.st\_mode) ? 1 : 0;

}

int main(int argc, char\* argv[]) {

     //if arguments are less than it is an invalid command, prompt the user about this.

    if (argc < 4) {

        printf("Program Name src dest [ops] [extension list]\n");

        exit(1);

    }

     //creating duplicates of source path

    srcPath = strdup(argv[1]);

    //creating duplicate of destination path

    destPath = strdup(argv[2]);

     //if the source path end character is slash, replace it will null character to mark the end of the source path destination.

    if (srcPath[strlen(srcPath) - 1] == '/') {

        \*(srcPath + strlen(srcPath) - 1) = '\0';

    }

    if (!ifDirectory(srcPath)) {

        printf("directory is already existing!!\n");

        exit(1);

    }

   //add the source folder to the destination

    if (destPath[strlen(destPath) - 1] == '/') {

        \*(destPath + strlen(destPath) - 1) = '\0';

    }

    char\* folder = strrchr(srcPath, '/');

    char new\_destPath[strlen(destPath) + strlen(folder)];

    strcat(new\_destPath, destPath);

    strcat(new\_destPath, folder);

    destPath = new\_destPath;

    char \*op = argv[3];

    int dest\_len = strlen(destPath);

    if (strcmp(op, "-cp") == 0) {

        will\_it\_move = 0;

    } else if (strcmp(op, "-mv") == 0) {

        will\_it\_move = 1;

    } else {

        printf("options: -cp -mv");

        exit(1);

    }

    num\_of\_exts = argc - 4;

    //extension starts from 4th index

    exts = (argv + 4);

    return nftw(srcPath, cpRmv, 64, FTW\_PHYS | FTW\_DEPTH);

}