OS Assignment 5

Q1. Write a C program to change the permissions of files in a directory created after a certain date. Inputs to the program: directory, date and new permission to be set as run time arguments

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

#include <sys/stat.h>

#include <time.h>

#include <pwd.h>

#include <grp.h>

//time\_t strtotime(const char \*date\_str);

// Helper function to convert a string date to a timestamp

time\_t strtotime(char \*date) {

    struct tm tm = {0};

    if (strptime(date, "%Y-%m-%d", &tm) == NULL) {

        return -1;

    }

    return mktime(&tm);

}

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

    DIR \*dir;

    struct dirent \*ent;

    struct stat st;

    struct passwd \*pwd;

    struct group \*grp;

    time\_t cutoff\_time;

    char \*dir\_path, \*new\_owner, \*new\_group;

    mode\_t new\_perm;

    int len;

    // Check if the correct number of arguments are provided

    if (argc != 4) {

        fprintf(stderr, "Usage: %s <directory\_path> <cutoff\_date> <new\_owner> <new\_group> <new\_permission>\n", argv[0]);

        return EXIT\_FAILURE;

    }

    // Get the cutoff time from the argument

    cutoff\_time = strtotime(argv[2]);

    if (cutoff\_time == -1) {

        fprintf(stderr, "Invalid cutoff date format. Use YYYY-MM-DD.\n");

        return EXIT\_FAILURE;

    }

    // Get the new owner and group from the argument

    new\_owner = argv[3];

    new\_group = argv[4];

    // Get the new permission from the argument

    new\_perm = strtol(argv[5], NULL, 8);

    if (new\_perm == 0) {

        fprintf(stderr, "Invalid new permission format. Use octal.\n");

        return EXIT\_FAILURE;

    }

    // Open the directory

    dir\_path = argv[1];

    dir = opendir(dir\_path);

    if (dir == NULL) {

        fprintf(stderr, "Cannot open directory: %s\n", dir\_path);

        return EXIT\_FAILURE;

    }

    // Loop through the directory entries

    while ((ent = readdir(dir)) != NULL) {

        // Ignore the "." and ".." entries

        if (strcmp(ent->d\_name, ".") == 0 || strcmp(ent->d\_name, "..") == 0) {

            continue;

        }

        // Get the file path

        len = strlen(dir\_path) + strlen(ent->d\_name) + 2;

        char file\_path[len];

        sprintf(file\_path, "%s/%s", dir\_path, ent->d\_name);

        // Get the file status

        if (stat(file\_path, &st) != 0) {

            fprintf(stderr, "Cannot get file status: %s\n", file\_path);

            continue;

        }

        // Check if the file was created after the cutoff date

        if (st.st\_mtime > cutoff\_time) {

            // Change the file ownership and permission

            pwd = getpwnam(new\_owner);

            if (pwd == NULL) {

                fprintf(stderr, "Invalid new owner: %s\n", new\_owner);

                continue;

            }

            grp = getgrnam(new\_group);

            if (grp == NULL) {

                fprintf(stderr, "Invalid new group: %s\n", new\_group);

                continue;

            }

            if (chown(file\_path, pwd->pw\_uid, grp->gr\_gid) != 0) {

                fprintf(stderr, "Cannot change file ownership: %s\n", file\_path);

                continue;

            }

            if (chmod(file\_path, new\_perm) != 0) {

                fprintf(stderr, "Cannot change file permission: %s\n", file\_path);

                continue;

            }

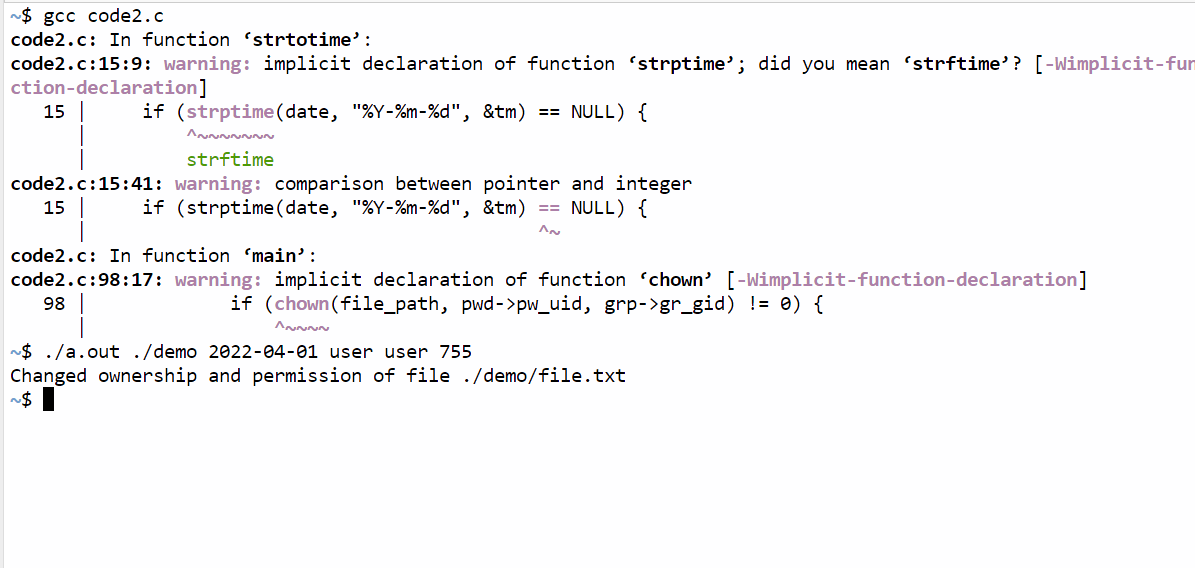
            printf("Changed ownership and permission of file %s\n", file\_path);

        }

    }

}

Screenshots



Q2. Write a C program to truncate the files in a directory created after a certain Date to half its original size. Inputs to the program: directory and date as run time arguments.

code2.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

#include <sys/stat.h>

#include <time.h>

// Helper function to convert a string date to a timestamp

time\_t strtotime(char \*date) {

    struct tm tm = {0};

    if (strptime(date, "%Y-%m-%d", &tm) == NULL) {

        return -1;

    }

    return mktime(&tm);

}

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

    DIR \*dir;

    struct dirent \*ent;

    struct stat st;

    time\_t cutoff\_time;

    char \*dir\_path;

    int len;

    long half\_size;

    // Check if the correct number of arguments are provided

    if (argc != 3) {

        fprintf(stderr, "Usage: %s <directory\_path> <cutoff\_date>\n", argv[0]);

        return EXIT\_FAILURE;

    }

    // Get the cutoff time from the argument

    cutoff\_time = strtotime(argv[2]);

    if (cutoff\_time == -1) {

        fprintf(stderr, "Invalid cutoff date format. Use YYYY-MM-DD.\n");

        return EXIT\_FAILURE;

    }

    // Open the directory

    dir\_path = argv[1];

    dir = opendir(dir\_path);

    if (dir == NULL) {

        fprintf(stderr, "Cannot open directory: %s\n", dir\_path);

        return EXIT\_FAILURE;

    }

    // Loop through the directory entries

    while ((ent = readdir(dir)) != NULL) {

        // Ignore the "." and ".." entries

        if (strcmp(ent->d\_name, ".") == 0 || strcmp(ent->d\_name, "..") == 0) {

            continue;

        }

        // Get the file path

        len = strlen(dir\_path) + strlen(ent->d\_name) + 2;

        char file\_path[len];

        sprintf(file\_path, "%s/%s", dir\_path, ent->d\_name);

        // Get the file status

        if (stat(file\_path, &st) != 0) {

            fprintf(stderr, "Cannot get file status: %s\n", file\_path);

            continue;

        }

        // Check if the file was created after the cutoff date

        if (st.st\_mtime > cutoff\_time) {

            // Truncate the file to half its original size

            half\_size = st.st\_size / 2;

            if (truncate(file\_path, half\_size) != 0) {

                fprintf(stderr, "Cannot truncate file: %s\n", file\_path);

                continue;

            }

            printf("Truncated %s to %ld bytes.\n", file\_path, half\_size);

        }

    }

    closedir(dir);

    return EXIT\_SUCCESS;

}

Screenshot

