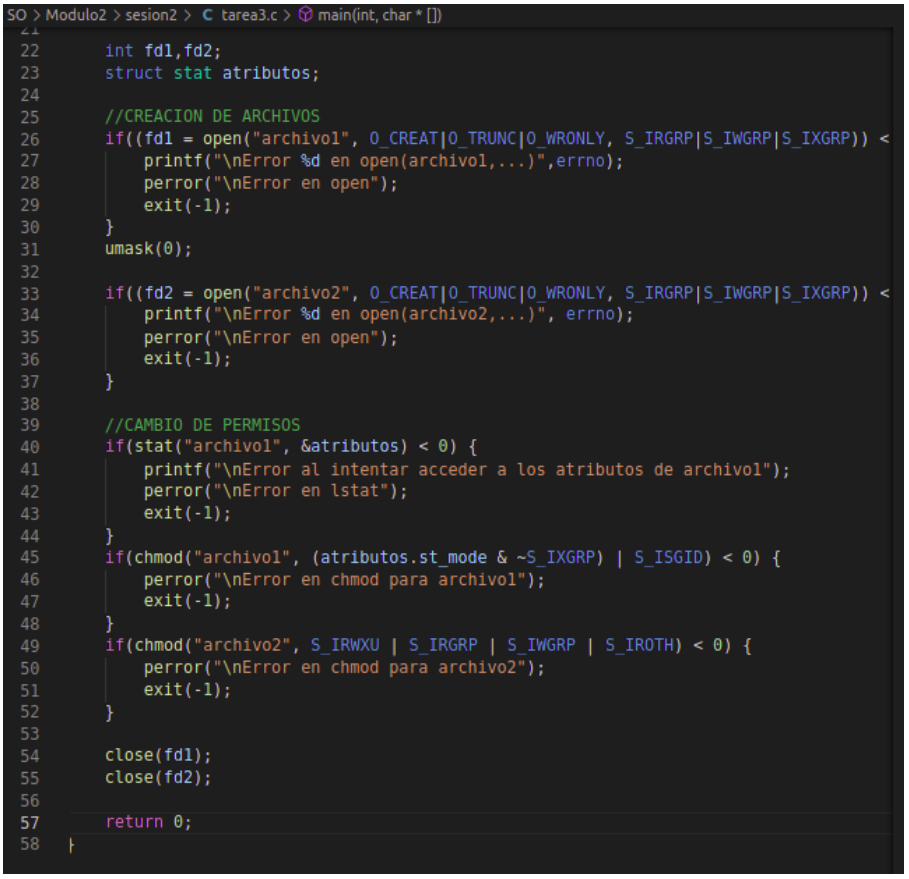
SESIÓN 2 MÓDULO II SO

Ejercicio 3. ¿Qué hace el siguiente programa?

tarea3.c




```c
int fd1,fd2;
struct stat atributos;

//CREACION DE ARCHIVOS
if((fd1 = open("archivo1", O_CREAT|O_TRUNC|O_WRONLY, S_IRGRP|S_IWGRP|S_IXGRP)) <
    printf("\nError %d en open(archivo1,...)",errno);
    perror("\nError en open");
    exit(-1);
}
umask(0);

if((fd2 = open("archivo2", O_CREAT|O_TRUNC|O_WRONLY, S_IRGRP|S_IWGRP|S_IXGRP)) <
    printf("\nError %d en open(archivo2,...)", errno);
    perror("\nError en open");
    exit(-1);
}

//CAMBIO DE PERMISOS
if(stat("archivo1", &atributos) < 0) {
    printf("\nError al intentar acceder a los atributos de archivo1");
    perror("\nError en lstat");
    exit(-1);
}
if(chmod("archivo1", (atributos.st_mode & ~S_IXGRP) | S_ISGID) < 0) {
    perror("\nError en chmod para archivo1");
    exit(-1);
}
if(chmod("archivo2", S_IRWXU | S_IRGRP | S_IWGRP | S_IROTH) < 0) {
    perror("\nError en chmod para archivo2");
    exit(-1);
}

close(fd1);
close(fd2);

return 0;
}
```

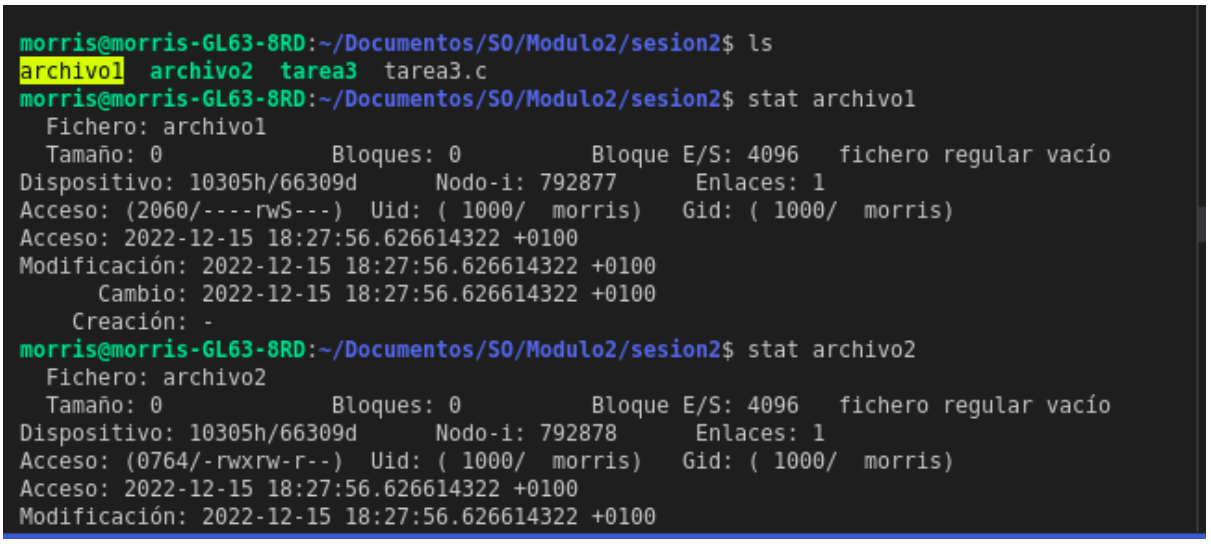
En este programa, se crean 2 archivos, archivo1 y archivo2 y se especifican sus correspondientes errores si no cumplen determinadas condiciones para crearse.

Antes de crearse el segundo archivo se pone la máscara a 0 con umask(0), lo que significa que el archivo2 se creará sin ningún tipo de permisos, ya que si no pusiéramos eso, por defecto tendrán permisos que se dan al crear archivos por defecto.

A continuación, con stat, copiamos el contenido del archivo1 a la estructura stat atributos, para poder acceder a su campo st_mode y poder modificar los permisos.

En el chmod del archivo1, hacemos un and lógico para cambiarle los permisos, negando el permiso de ejecución para el grupo (S_IXGRP), por lo que le quitamos ese permiso, y también le activamos la asignación del GID del propietario al GID efectivo del proceso que ejecute el archivo.

En el chmod del archivo2, cambiamos sus permisos tal que el usuario tenga permisos de lectura, escritura y ejecución; el grupo tenga permiso de lectura; el grupo tenga permiso de escritura; y otros tenga permiso de lectura.



Ejercicio 2. Realiza un programa en C utilizando las llamadas al sistema necesarias que acepte como entrada:
• Un argumento que representa el 'pathname' de un directorio.
• Otro argumento que es un número octal de 4 dígitos (similar al que se puede utilizar para cambiar los permisos en la llamada al sistema chmod). Para convertir este argumento tipo cadena a un tipo numérico puedes utilizar la función strtol. Consulta el manual en línea para conocer sus argumentos.

El programa tiene que usar el número octal indicado en el segundo argumento para cambiar los permisos de todos los archivos que se encuentren en el directorio indicado en el primer argumento.

El programa debe proporcionar en la salida estándar una línea para cada archivo del directorio que esté formada por:

nombre archivo permisos antiguos permisos nuevos

Si no se pueden cambiar los permisos de un determinado archivo se debe especificar la siguiente información en la línea de salida:

nombre archivo errno permisosAntiguos

SO > Modulo2 > sesion2 > C ejercicio2.c > main(int, char * [])

```c
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>

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

    DIR *directorio;
    struct dirent *entrada;
    struct stat atributos;
    int permisosNuevos, permisosAntiguos;
    char pathname[257]; //256 era muy pequeño para sprintf debido a que también cont
                        //carácter nulo /0

    directorio = opendir(argv[1]); //abrimos el directorio y lo guardamos en el punt

    //strol convierte el número en un long int de forma octal, que es lo que
    //necesitamos nosotros para dar permisos, pues están de esa forma.
    permisosNuevos = strtol(argv[2], NULL, 8);

    //Lee donde está el puntero directorio, adelanta una posicion y lo devuelve
    //a la estructura dirent (entrada).
    while ((entrada == readdir(directorio)) != 0){

        //guarda en pathname el nombre del archivo
        sprintf(pathname, "%s/%s", argv[1], entrada->d_name);

        //lstat examina el archivo al que apunta pathname, y llena el buffer atribut
        lstat(pathname, &atributos);

        //con el stat atributos accede a los permisos del archivo y se queda solo
        //con los de la derecha de &.
        permisosAntiguos = atributos.st_mode & (S_IRWXU | S_IRWXG | S_IRWXO);

        //Con strcmp se comprueba si el nombre del directorio es . o .. y ya
        //en función de eso le damos los nuevos permisos.
        //Lo que hace strcmp es comprobar 2 strings, en este caso compara el nombre
        //archivo correspondiente con los string . y ..
        if ((strcmp(entrada->d_name, ".") && strcmp(entrada->d_name, "..")) != 0){
            //Si el archivo al que apunta pathname se le puede cambiar su modo con l
            //nuevos permisos, será correcto e imprimirá los permisos antiguos y nue
            if (chmod(pathname, permisosNuevos) == 0)
                printf("%s: %o %o\n", entrada->d_name, permisosAntiguos, permisosNue

            //Si no se le puede cambiar su modo, tan solo imprimirá el error y los a
            else
                printf("%s: %o %o\n", entrada->d_name, errno, permisosAntiguos);

        }

    }
    //cerramos el directorio que hemos pasado como argumento.
    closedir(directorio);
}
```