



Instituto Politécnico Nacional

ESCOM

Practica 2

Introducción al sistema operativo Linux y Windows (2)

Sistemas Operativos – 2CM17

Integrantes:

Mora Ayala José Antonio

Ramírez Cotonieto Luis Fernando

Torres Carrillo Josehf Miguel Angel

Tovar Jacuinde Rodrigo

Profesor:

Cortes Galicia Jorge

INSTITUTO POLITÉCNICO NACIONAL



INTRODUCCIÓN

A continuación, se presenta una introducción a los temas que serán profundizados dentro de esta practica mediante la implementación de código funcional relacionado a cada uno de los temas que competen este documento, así como a cada uno de los integrantes del equipo

Como todos sabemos, conforme ha pasado el tiempo las generaciones van cambiando la manera en que experimentan la evolución tecnológica que se ha dado en el mundo desde hace varios años y la cual, hoy continua, pues la tecnología cada día avanza y se van realizando nuevos descubrimientos a diario. Situación que resulta benéfica la mayoría del tiempo, pero también puede llegar a ser contraproducente, pues conforme mas acceso y facilidad se tiene a estas tecnologías y lo que conlleva interactuar con las mismas las personas en general han perdido el interés en conocer como es que funcionan o que partes son las que componen los dispositivos con los cuales interactúan día a día.

Es por eso que comenzaremos dando un antecedente y conceptos generales y básicos para comprender de una forma más completa esta practica

¿QUÉ ES UN SISTEMA OPERATIVO?

El sistema operativo es el software que coordina y dirige todos los servicios y aplicaciones que utiliza el usuario en una computadora, por eso es el más importante y fundamental. Se trata de programas que permiten y regulan los aspectos más básicos del sistema. Los sistemas operativos más utilizados son Windows, Linux, OS/2 y DOS.

Los sistemas operativos, también llamados núcleos o kernels, suelen ejecutarse de manera privilegiada respecto al resto del software, sin permitir que un programa cualquiera realice cambios de importancia sobre él que puedan comprometer su funcionamiento.

El sistema operativo es el protocolo básico de operatividad del computador, que coordina todas sus demás funciones de comunicaciones, de procesamiento, de interfaz con el usuario.

Los sistemas operativos consisten en interfaces gráficas, entornos de escritorio o gestores de ventanas que brindan al usuario una representación gráfica de los procesos en marcha. También puede ser una línea de comandos, es decir, un conjunto de instrucciones ordenado según su prioridad y que funciona en base a órdenes introducidas por el usuario.

Las primeras versiones de las computadoras no tenían sistemas operativos. En la década de los sesenta los ordenadores usaban procesamientos por lotes y fue durante estos años cuando comenzaron a desarrollarse los sistemas operativos.

COMPONENTES

El sistema operativo posee tres componentes esenciales o paquetes de software que permiten la interacción con el hardware:

Sistema de archivos. Es el registro de archivos donde adquieren una estructura arbórea.

Interpretación de comandos. Se logra con aquellos componentes que permiten la interpretación de los comandos, que tienen como función comunicar las órdenes dadas por el usuario en un lenguaje que el hardware pueda interpretar (sin que aquel que dé las órdenes conozca dicho lenguaje).

Núcleo. Permite el funcionamiento en cuestiones básicas como la comunicación, entrada y salida de datos, gestión de procesos y la memoria, entre otros.

PARA QUE SIRVE

- Gestionar **la memoria** de acceso aleatorio y ejecutar las aplicaciones, designando los recursos necesarios.
- Administrar al **CPU** gracias a un algoritmo de programación.
- Direccionar **las entradas y salidas de datos** (a través de *drivers*) por medio de los periféricos de entrada o salida.
- Administrar **la información** para el buen funcionamiento de la PC.
- Dirigir **las autorizaciones** de uso para los usuarios.
- Administrar **los archivos**

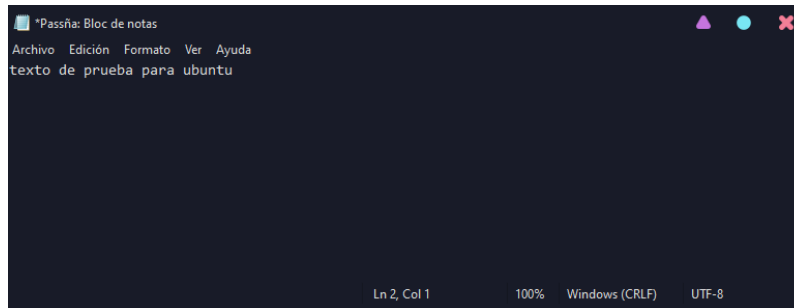
TIPOS DE SISTEMA OPERATIVO

- **Según el usuario pueden ser:** multiusuario, sistema operativo que permite que varios usuarios ejecuten simultáneamente sus programas; o monousuario, sistema operativo que solamente permite ejecutar los programas de un usuario a la vez.
- **Según la gestión de tareas pueden ser:** monotarea, sistema operativo que solamente permite ejecutar un proceso a la vez; o multitarea, sistema operativo que puede ejecutar varios procesos al mismo tiempo.
- **Según la gestión de recursos pueden ser:** centralizado, sistema operativo que solo permite utilizar los recursos de un solo ordenador; o distribuido, sistema operativo que permite ejecutar los procesos de más de un ordenador al mismo tiempo.

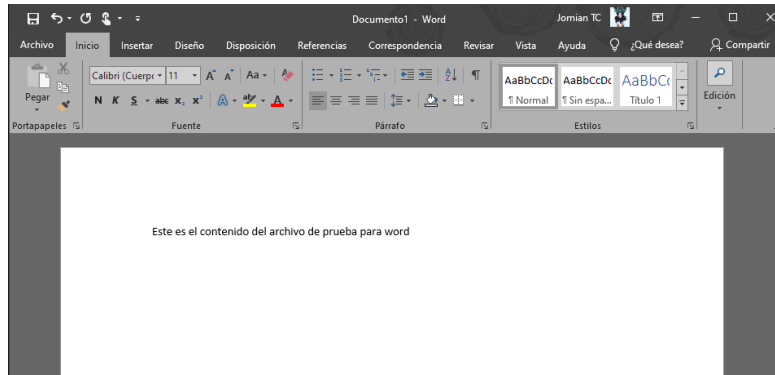
DESARROLLO EXPERIMENTAL

1. Cree un archivo de texto (con cualquier contenido) y un archivo en Word (con cualquier contenido) en el sistema operativo Windows y guárdelo en una memoria usb.

Archivo .txt

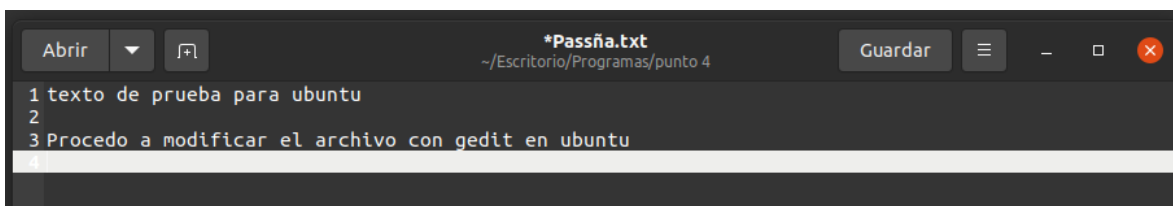


Archivo .docx



2. Inicie sesión en Linux.
3. Verifique si está montada la unidad de memoria usb en su sistema, para ello introduzca una memoria usb y observe si es reconocida en el escritorio.
4. Edite tanto el contenido del archivo de texto como de Word modificándolo mediante el uso de gedit. Guarde sus archivos.

Edición de archivo .txt



el texto para

00\00\00\00'

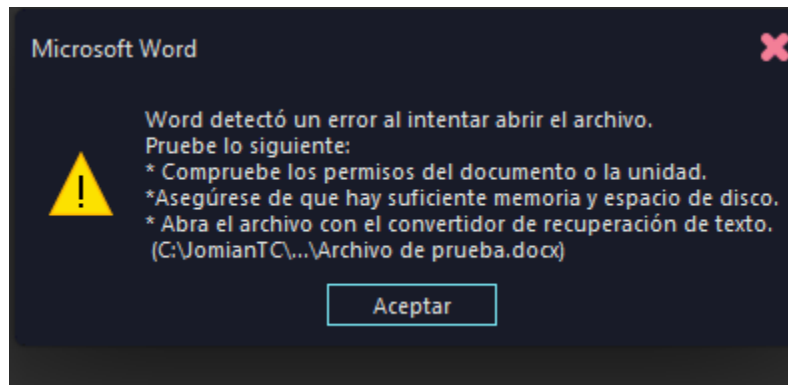
[illegible]

5. Inicie sesión en Windows y observe el contenido de sus archivos en su memoria usb. ¿Se observan las modificaciones realizadas en Linux?, explique el por qué si o no se observan.

```
Passña: Bloc de notas
Archivo Edición Formato Ver Ayuda
texto de prueba para ubuntu

Procedo a modificar el archivo con gedit en ubuntu
```

En el
en e



Mientras que el archivo de .docx sufrió fallas al intentar abrirlo de nuevo en Windows mostrando el cuadro de advertencia anterior, no se pudo recuperar el archivo abriendo una versión anterior, o regresando la parte borrada el Ubuntu

Esto se debe al formato propietario de Microsoft office que usa únicamente su paquetería y al ser un formato restringido su compilación y modificación solo puede ser hecha por el mismo software

Para el caso del txt es distinto ya que el txt es un formato de texto plano que cualquier máquina puede abrir y ejecutar sin problemas

6. A través de su manual en línea de Linux (man) en la segunda o tercera sección (man 2 o man 3), investigue y reporte para qué sirven las siguientes llamadas al sistema de Linux: open, close, read, write, creat, lseek, access, stat, chmod, chown, fcntl, opendir, y readdir.

LLAMADA A SISTEMA LINUX

1. open

El comando open nos permite abrir un archivo en un directorio específico, si el archivo no está creado también podemos poner como parámetro en el comando el comando creat que crea el archivo en la misma ruta

al parecer solo es un comando que podemos de manera que esté integrado en un programa ya que si intentamos poner open en nuestra terminal esta arroja que la función no existe

La función open se usa de la siguiente manera

```
open(const char *pathname, int flags, mode_t mode)
```

Donde pathname es un char que contiene la ruta del archivo, flags es la forma en la que vamos a abrir nuestro archivo ya sea lectura o escritura y el mode son los permisos que tendrá el archivo si es que este no se ha creado en nuestra ruta de pathname

2. close

La función close es más sencilla nos permite cerrar un archivo que se encuentre en nuestro código, de igual manera es inutilizable en la terminal solo mediante algún programa o ejecutable

La función close se usa como

```
close(int fd)
```

donde el entero fd es el archivo que tengamos abierto en ese momento

3. read

Esta función permite leer lo que hay dentro de un archivo mediante la cuenta de bytes del archivo que intentemos leer alojado en el buffer

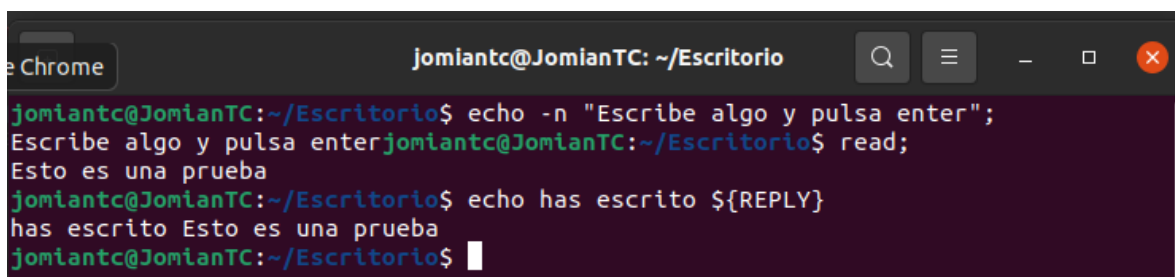
una vez que la función retorne 0 quiere decir que ha llegado al final del archivo y podemos imprimirlo para ver lo que contiene el archivo

Podemos declarar read de la siguiente manera

```
read(int fd, void *buf, size_t count)
```

Donde fd es el archivo que vamos a leer, el buf es una estructura que esta alojada en nuestra biblioteca, y count es la variable que almacenara el número de bytes encontrados

También lo podemos usar de la siguiente manera en un terminal

A terminal window titled 'Jomiantc@JomianTC: ~/Escritorio' with standard window controls. The terminal shows a sequence of commands and their outputs: 1. 'echo -n "Escribe algo y pulsa enter";' followed by 'Escribe algo y pulsa enter'. 2. 'read;' followed by 'Esto es una prueba'. 3. 'echo has escrito \${REPLY}' followed by 'has escrito Esto es una prueba'. 4. The prompt 'Jomiantc@JomianTC:~/Escritorio\$' is shown again with a cursor. The terminal has a dark background with light-colored text.

```
Jomiantc@JomianTC:~/Escritorio$ echo -n "Escribe algo y pulsa enter";
Escribe algo y pulsa enterJomiantc@JomianTC:~/Escritorio$ read;
Esto es una prueba
Jomiantc@JomianTC:~/Escritorio$ echo has escrito ${REPLY}
has escrito Esto es una prueba
Jomiantc@JomianTC:~/Escritorio$
```

4. write

Con esta función podemos escribir una cadena dentro de un archivo de texto

la forma de declarar write es la siguiente

```
write(int fd, const void *buf, size_t count);
```

donde fd es nuestro archivo, *buf es la cadena que nosotros queremos ingresar en el archivo y count es el número total de bytes de la cadena

con la consola de comandos podemos mandar un mensaje a otro archivo o desplegar 2 funciones de ejemplo

5. creat

Con la función create podemos crear archivos en una ubicación que nosotros decidamos

La función create se escribe de la siguiente manera

```
create(const char *pathname, mode_t mode)
```

Donde pathname es la ruta con el nombre del archivo a crear y mode son los permisos que tendrá dicho archivo para ser abierto

No se puede usar el comando en una terminal

6. lseek

Esta función nos permite reposicionar el puntero de lectura en nuestro archivo para empezar a leer o escribir desde ahí

podemos escribir de esta manera lseek

```
lseek(int fd, off_t offset, int whence);
```

Donde fd es el archivo, offset es a cuantos bytes queremos que se haga el recorrido y whence es desde donde partirá el offset

7. access

Con la función access podemos observar la accesibilidad de un archivo o directorio ya sea lectura, escritura o ejecutable en los casos de archivos

```
int access(const char *pathname, int mode);
```

Pathname es el directorio del archivo y mode es el modo ya sea lectura o escritura o ejecutable

8. stat

Con stat podemos observar la información de un archivo como su nombre, tamaño, permisos etc.

```
int stat(const char *pathname, struct stat *statbuf);
```

Donde pathname es la ruta de acceso, y statbuf es una estructura donde podremos utilizar las diversas funciones de stat

9. chmod

Permite cambiar los permisos de un archivo o directorio mediante números o letras además de poner hacerlo a distintos usuarios

```
int chmod(const char *pathname, mode_t mode);
```

pathname ruta y mode el modo al que queremos cambiarlo

10. chown

El comando chown permite cambiar el propietario de un archivo o directorio en sistemas tipo UNIX. Puede especificarse tanto el nombre de un usuario, así como el identificador de usuario (UID) y el identificador de grupo (GID).

11. fcntl

Es una función multipropósito dependiendo de los parámetros que le demos para modificar nuestro archivo de manera eficiente

```
int fcntl(int fd, int cmd);
```

tenemos la función `fcntl` donde `fd` es el archivo y `cmd` es la operación que realizara el comando dependiendo del mismo

12. `opendir`

Permite abrir un directorio dependiendo de la ruta que nosotros mismo asignemos dentro del parámetro de la función

```
DIR *opendir(const char *name);
```

```
DIR *fdopendir(int fd);
```

Podemos usarla tanto si hemos abierto nuestro archivo como si no lo hemos hecho anteriormente

13. `readdir`

Lee un directorio devolviendo un puntero a una estructura siguiente entrada de directorio en el flujo de directorio

```
struct dirent *readdir(DIR *dirp);
```

14. A través del sitio MSDN de Microsoft, investigue y reporte para qué sirven las siguientes llamadas al sistema de Windows: `OpenFile`, `CloseFile`, `ReadFile`, `WriteFile`, `CreateFile`, `SetFilePointer`. A través del sitio OpenGroup, investigue para que sirven las siguientes llamadas al sistema de Windows: `stat`, `opendir`, y `readdir`. Investigue y reporte si existe una llamada idéntica a `chmod` en Windows, en caso de no existir indique el motivo por el cual no existe.

LLAMADA A SISTEMA WINDOWS

1. `OpenFile`

Crea, abre, elimina o vuelve a abrir un archivo

Los parámetros que recibe son: “nombre del archivo”, “tamaño del Buffer”, y “acción a tomar (a elegir de las listadas a continuación)”

Value	Meaning
OF_CANCEL 0x00000800	Ignored. To produce a dialog box containing a Cancel button, use OF_PROMPT .
OF_CREATE 0x00001000	Creates a new file. If the file exists, it is truncated to zero (0) length.
OF_DELETE 0x00000200	Deletes a file.
OF_EXIST 0x00004000	Opens a file and then closes it. Use this to test for the existence of a file.
OF_PARSE 0x00000100	Fills the OFSTRUCT structure, but does not do anything else.
OF_PROMPT 0x00002000	Displays a dialog box if a requested file does not exist. A dialog box informs a user that the system cannot find a file, and it contains Retry and Cancel buttons. The Cancel button directs OpenFile to return a file-not-found error message.
OF_READ 0x00000000	Opens a file for reading only.
OF_READWRITE 0x00000002	Opens a file with read/write permissions.
OF_REOPEN 0x00008000	Opens a file by using information in the reopen buffer.
OF_SHARE_COMPAT 0x00000000	For MS-DOS–based file systems, opens a file with compatibility mode, allows any process on a specified computer to open the file any number of times. Other efforts to open a file with other sharing modes fail. This flag is mapped to the FILE_SHARE_READ FILE_SHARE_WRITE flags of the CreateFile function.
OF_SHARE_DENY_NONE 0x00000040	Opens a file without denying read or write access to other processes. On MS-DOS-based file systems, if the file has been opened in compatibility mode by any other process, the function fails. This flag is mapped to the FILE_SHARE_READ FILE_SHARE_WRITE flags of the CreateFile function.

OF_SHARE_DENY_READ 0x00000030	<p>Opens a file and denies read access to other processes.</p> <p>On MS-DOS-based file systems, if the file has been opened in compatibility mode, or for read access by any other process, the function fails.</p> <p>This flag is mapped to the FILE_SHARE_WRITE flag of the CreateFile function.</p>
OF_SHARE_DENY_WRITE 0x00000020	<p>Opens a file and denies write access to other processes.</p> <p>On MS-DOS-based file systems, if a file has been opened in compatibility mode, or for write access by any other process, the function fails.</p> <p>This flag is mapped to the FILE_SHARE_READ flag of the CreateFile function.</p>
OF_SHARE_EXCLUSIVE 0x00000010	<p>Opens a file with exclusive mode, and denies both read/write access to other processes. If a file has been opened in any other mode for read/write access, even by the current process, the function fails.</p>
OF_VERIFY	<p>Verifies that the date and time of a file are the same as when it was opened previously.</p> <p>This is useful as an extra check for read-only files.</p>
OF_WRITE 0x00000001	<p>Opens a file for write access only.</p>

2. CloseFile

Como tal no encontramos la función denominada como CloseFile, si no que en Windows manejamos CloseHandle, el cual cierra un identificador de objeto abierto

```
BOOL CloseHandle(
    HANDLE hObject
);
```

- Access token
- Communications device
- Console input
- Console screen buffer
- Event
- File
- File mapping
- I/O completion port
- Job
- Mailslot
- Memory resource notification
- Mutex
- Named pipe
- Pipe
- Process
- Semaphore
- Thread
- Transaction
- Waitable timer

3. ReadFile

Lee datos de un archivo de entrada específico o bien de un dispositivo de entrada/salida. La lectura sucede en la posición especificada por el puntero del archivo.

Esta función está diseñada para síncrona y asíncronas operaciones.

```
BOOL ReadFile(  
    HANDLE    hFile,  
    LPVOID    lpBuffer,  
    DWORD     nNumberOfBytesToRead,  
    LPDWORD   lpNumberOfBytesRead,  
    LPOVERLAPPED lpOverlapped  
);
```

4. WriteFile

Escribe datos en un archivo o dispositivo de entrada/salida específico

```
BOOL WriteFile(  
    HANDLE    hFile,  
    LPCVOID   lpBuffer,  
    DWORD     nNumberOfBytesToWrite,  
    LPDWORD   lpNumberOfBytesWritten,  
    LPOVERLAPPED lpOverlapped  
);
```

5. CreateFile

Crea o abre un archivo o dispositivo de E/S. Los dispositivos de E/S más utilizados son los siguientes: archivo, secuencia de archivos, directorio, disco físico, volumen, búfer de consola, unidad de cinta, recurso de comunicaciones, mailslot y canalización. La función devuelve un identificador que se puede utilizar para acceder al archivo o dispositivo para varios tipos de E/S en función del archivo o dispositivo y los indicadores y atributos especificados.

Para realizar esta operación como una operación transaccionada, que da como resultado un identificador que se puede usar para la E/S transaccionada, utilice la función CreateFileTransacted.

```
HANDLE CreateFileA(  
    LPCSTR      lpFileName,  
    DWORD       dwDesiredAccess,  
    DWORD       dwShareMode,  
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,  
    DWORD       dwCreationDisposition,
```

```
DWORD      dwFlagsAndAttributes,  
HANDLE      hTemplateFile  
);
```

6. SetFilePointer

Mueve el puntero de archivo del archivo especificado.

Esta función almacena el puntero de archivo en dos valores LONG. Para trabajar con punteros de archivo que son mayores que un único valor LONG, es más fácil usar la función SetFilePointerEx.

```
DWORD SetFilePointer(  
    HANDLE hFile,  
    LONG   lDistanceToMove,  
    PLONG   lpDistanceToMoveHigh,  
    DWORD  dwMoveMethod  
);
```

7. Stat

La función `_stat` obtiene información sobre el archivo o directorio especificado por la ruta de acceso y lo almacena en la estructura señalada por el búfer. `_stat` controla automáticamente los argumentos de cadena de caracteres multibyte según corresponda, reconociendo secuencias de caracteres multibyte según la página de códigos multibyte actualmente en uso.

8. opendir

Nuevamente nos encontramos con la misma situación, como tal no existe una llamada al sistema denominada como `opendir` para este sistema operativo, por lo que optamos por usar `dirent`, el cual si las contiene aparte de que es compatible con UNIX también

9. readdir

Nuevamente nos encontramos con la misma situación, como tal no existe una llamada al sistema denominada como `opendir` para este sistema operativo, por lo que optamos por usar `dirent`, el cual si las contiene aparte de que es compatible con UNIX también

¿CHMOD EN WINDOWS?

No hay nada llamado chmod en Windows porque el modelo de seguridad de Windows es diferente al de Linux. Puede usar el attrib comando para cambiar las propiedades de los objetos. (Pero son más hacia propiedades globales).

El attrib comando es la coincidencia más cercana para cosas muy básicas (solo lectura, marcas de archivo). Luego está el comando ACL (lista de control de acceso) cacls.

10. Utilizando únicamente las llamadas al sistema revisadas para Linux que sean necesarias, desarrolle un programa en C que cree una serie aleatoria de archivos (en una ruta especificada a través de la línea de comando), el contenido de los archivos serán cadenas que estén almacenadas en un arreglo. Restricción: Únicamente utilizar las llamadas al sistema para manipulación de archivos revisadas en el punto 6 de esta práctica.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <string.h>
6
7  char ruta[53] = {"/home/jomiantc/Escritorio/Programas/Punto_8/arc0.txt"};
8  char save[11];
9  char cadena[46] = {"Este es el archivo numero 0 de los txt creados"};
10
11 int variable, i;
12 mode_t mode = S_IRUSR | S_IWUSR;
13
14 int main (void){
15
16     for (i = 0; i < 10; ++i){
17
18         sprintf(save, "%d", i);
19
20         ruta[47] = save[0];
21         cadena[26] = save[0];
22
23         variable = creat(ruta, mode);
24         variable = open(ruta, O_WRONLY);
25
26         write(variable, &cadena, sizeof(cadena));
27
28         close(variable);
29     }
30 }
```

11. Una vez creados los archivos con sus contenidos por el programa del punto 8 y utilizando las llamadas al sistema revisadas para Linux que sean necesarias, desarrolle un programa en C para cambiar los permisos de un archivo seleccionado por el usuario.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <sys/stat.h>
8
9  char ruta[53] = {"/home/jomiantc/Escritorio/Programas/Punto_8/arc0.txt"};
10 char mode[4] = "";
11 char save[1];
12
13 int variable, i, j, k, pausa;
14
15 void modificar();
16 void comando();
17
18 int main(void){
19
20     do{
21
22         printf("A que archivo deseas cambiarle los permisos de usuario?\n");
23         printf("1 - arc0.txt\n");
24         printf("2 - arc1.txt\n");
25         printf("3 - arc2.txt\n");
26         printf("4 - arc3.txt\n");
27         printf("5 - arc4.txt\n");
28         printf("6 - arc5.txt\n");
29         printf("7 - arc6.txt\n");
30         printf("8 - arc7.txt\n");
31         printf("9 - arc8.txt\n");
32         printf("10 - arc9.txt\n\n");
33
34         printf("0 Salir del programa\n\n");
35
36         printf("Digite el numero del archivo: \t");
37
38         scanf("%d", &i);
39
40         system("clear");
41
42         if(i == 0){
43
44             i = -1;
45         }
46
47         else{
48
49             modificar();
50         }
51
52         system("clear");
53
54     }while(i != -1);
55
56 }
```



```

60     i--;
61
62     sprintf(save, "%d", i);
63
64     ruta[47] = save[0];
65
66     mode[0] = '0';
67
68     do{
69
70         printf("Como quieres administrar los permisos del archivo?\n");
71         printf("1 - solo lectura\n");
72         printf("2 - solo escritura y lectura\n");
73         printf("3 - solo lectura y ejecutable\n");
74         printf("4 - sin ningun permiso\n");
75         printf("5 - con todos los permisos\n\n");
76
77         printf("0 Regresar al menu anterior\n\n");
78
79         printf("Digite el numero del permiso: \t");
80
81         scanf("%d", &j);
82
83         system("clear");
84
85         switch (j){
86
87             case 0:
88
89                 j = -1;
90                 pausa = 0;
91
92                 break;
93
94             case 1:
95
96                 mode[1] = '4'; mode[2] = '4'; mode[3] = '4';
97                 comando();
98
99                 break;
100
101             case 2:
102
103                 mode[1] = '6'; mode[2] = '6'; mode[3] = '6';
104                 comando();
105
106                 break;
107
108             case 3:
109
110                 mode[1] = '5'; mode[2] = '5'; mode[3] = '5';
111                 comando();
112
113                 break;
114
115             case 4:
116
117                 mode[1] = '0'; mode[2] = '0'; mode[3] = '0';
118                 comando();
119
120                 break;

```

```

121
122     case 5:
123
124         mode[1] = '7'; mode[2] = '7'; mode[3] = '7';
125         comando();
126
127         break;
128
129     default:
130
131         printf("No has ingresado un numero valido\n");
132
133     }
134
135     system("clear");
136
137     if(pausa == 10){
138
139         printf("Modificacion exitosa \n");
140         printf("Digite 0 para continuar: \t");
141
142         scanf("%d", &i);
143
144         system("clear");
145
146     }
147
148 }while(j != -1);
149 }
150
151 void comando(){
152
153     k = strtol(mode, 0, 8);
154
155     chmod(ruta, k);
156
157     pausa = 10;
158
159     j = -1;
160 }

```

```

jomiantc@JomianTC: ~/Escritorio/Programas/Punto_8
Programa9.c:127:19: warning: 'sprintf' writing a terminating nul past the end of
the destination [-Wformat-overflow=]
127 |     sprintf(save,"%d",i);
    |             ^
Programa9.c:127:3: note: 'sprintf' output between 2 and 12 bytes into a destinat
ion of size 1
127 |     sprintf(save,"%d",i);
    |     ^^^^^^^^^^^^^^^^^
jomiantc@JomianTC:~/Escritorio/Programas/Punto_8$ ./p9.exe
A que archivo deseas cambiarle los permisos de usuario?
1 - arc0.txt
2 - arc1.txt
3 - arc2.txt
4 - arc3.txt
5 - arc4.txt
6 - arc5.txt
7 - arc6.txt
8 - arc7.txt
9 - arc8.txt
10 - arc9.txt

0 Salir del programa
Digite el numero del archivo:

```

```
jomiantc@JomianTC: ~/Escritorio/Programas/Punto_8
Como quieres administrar los permisos del archivo?
1 - solo lectura
2 - solo escritura y lectura
3 - solo lectura y ejecutable
4 - sin ningun permiso
5 - con todos los permisos

0 Regresar al menu anterior
Digite el numero del permiso: 
```

Propiedades de arc0.txt

Básico **Permisos** Abrir con

Propietario Yo

Acceso Lectura y escritura

Grupo jomiantc

Acceso Ninguno

Otros

Acceso Ninguno

Ejecución Permitir_ejecutar el archivo como un programa

Contexto de seguridad desconocido

Propiedades de arc0.txt

Básico **Permisos** Abrir con

Propietario Yo

Acceso Lectura y escritura

Grupo jomiantc

Acceso Lectura y escritura

Otros

Acceso Ninguno

Ejecución Permitir_ejecutar el archivo como un programa

Contexto de seguridad desconocido

12. Una vez creados los archivos con sus contenidos por el programa del punto 8 y utilizando únicamente las llamadas al sistema revisadas para Linux que sean necesarias, desarrolle un programa en C que liste los archivos creados, mostrando su tamaño, fecha y hora de acceso.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <string.h>
6  #include <sys/stat.h>
7  #include <sys/types.h>
8  #include <unistd.h>
9  #include <time.h>
10
11 struct stat statbuf;
12
13 char archivo[8] = {"arc0.txt"};
14 char save[1];
15 char times[25];
16
17 int i;
18
19 int main (void){
20
21     for (i = 0; i < 10; ++i){
22
23         sprintf(save, "%d", i);
24
25         archivo[3] = save[0];
26
27         stat(archivo, &statbuf);
28
29         printf("Archivo: %s  ", archivo);
30
31         printf("tamaño del archivo: %ld bytes  ", statbuf.st_size);
32
33         strcpy(times, ctime(&statbuf.st_mtime));
34
35         printf("últ. modificación: %s", times);
36
37     }
38
39 }
```

```
Jomiantc@JomianTC: ~/Escritorio/Programas/Punto_8
Jomiantc@JomianTC:~/Escritorio/Programas/Punto_8$ gcc Programa10.c -o p10.exe
Programa10.c: In function 'main':
Programa10.c:23:19: warning: 'sprintf' writing a terminating nul past the end of the destination [-Wformat-overflow=]
   23 |     sprintf(save, "%d", i);
      |           ^
Programa10.c:23:19: note: 'sprintf' output between 2 and 12 bytes into a destination of size 1
   23 |     sprintf(save, "%d", i);
      |           ^
Jomiantc@JomianTC:~/Escritorio/Programas/Punto_8$ ./p10.exe
Archivo: arc0.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc1.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc2.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc3.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc4.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc5.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc6.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc7.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc8.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Archivo: arc9.txt tamaño del archivo: 46 bytes últ. modificación: Tue Mar 9 21:47:08 2021
Jomiantc@JomianTC:~/Escritorio/Programas/Punto_8$
```

13. Una vez creados los archivos con sus contenidos por el programa del punto 8 y utilizando únicamente las llamadas al sistema revisadas para Linux que sean necesarias, desarrolle un programa en C para mostrar el contenido de un archivo seleccionado por el usuario, y que copie uno o más de los archivos creados a un directorio previamente establecido.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <sys/stat.h>
8
9  char rutacopiado[34] = {"/home/jomiantc/Escritorio/arc0.txt"};
10 char ruta[53] = {"/home/jomiantc/Escritorio/Programas/Punto_8/arc0.txt"};
11 char save[1];
12 char cadena[46];
13
14 int variable, i, j, pausa;
15 char buf[46];
16 mode_t mode = S_IRUSR | S_IWUSR;
17
18 void leer();
19 void impricontent();
20
21 int main(void){
22
23     do{
24
25         printf("Que deseas hacer?\n");
26         printf("1 - Leer contenido de archivo\n");
27         printf("2 - Copiar archivos\n\n");
28
29         printf("0 - Salir del programa\n\n");
30
31         printf("Digite el numero de la accion: \t");
32
33         scanf("%d", &i);
34
35         system("clear");
36
37         if(i == 0){
38
39             i = -1;
40
41         }
42
43         else{
44
45             leer();
46
47         }
48
49         system("clear");
50     }while(i != -1);
51 }
52
53 void leer(){
54
55     do{
56
57         if(i == 1) printf("De cual archivo desear ver el contenido?\n");
58         if(i == 2) printf("Que archivo desea copiar?\n");
```

```

60     printf("1 - arc0.txt\n");
61     printf("2 - arc1.txt\n");
62     printf("3 - arc2.txt\n");
63     printf("4 - arc3.txt\n");
64     printf("5 - arc4.txt\n");
65     printf("6 - arc5.txt\n");
66     printf("7 - arc6.txt\n");
67     printf("8 - arc7.txt\n");
68     printf("9 - arc8.txt\n");
69     printf("10 - arc9.txt\n\n");
70
71     printf("0 Regresar al menu anterior\n\n");
72
73     printf("Digite el numero del archivo: \t");
74
75     scanf("%d", &j);
76
77     system("clear");
78
79     if(j == 0){
80
81         j = -1;
82         pausa = 0;
83     }
84
85     else{
86
87         impricontent();
88     }
89
90     if(pausa == 10){
91
92         printf("Digite 0 para continuar: \t");
93
94         scanf("%d", &i);
95
96         system("clear");
97
98     }
99
100
101 }while(j != -1);
102 }
103
104 void impricontent(){
105
106     j--;
107
108     sprintf(save, "%d", j);
109
110     ruta[47] = save[0];
111
112     variable = open(ruta, O_RDONLY);
113
114     read(variable, buf, 46);
115
116     close(variable);
117
118
119     if(i == 1){
120

```

```

121     printf("Contenido: %s\n", buf);
122
123 }
124
125 if(i == 2){
126     strcpy(cadena, buf);
127     rutacopiado[29] = save[0];
128
129     variable = creat(rutacopiado, mode);
130
131     variable = open(rutacopiado, O_WRONLY);
132
133     write(variable, &cadena, sizeof(cadena));
134
135     close(variable);
136
137     printf("El contenido se ah copiado en el Escritorio con exito \n");
138 }
139
140 pausa = 10;
141
142 j = -1;
143
144 }
145

```

```

jomiantc@JomianTC: ~/Escritorio/Programas/Punto_8
jomiantc@JomianTC:~/Escritorio/Programas/Punto_8$ ./p11.exe
Que deseas hacer?
1 - Leer contenido de archivo
2 - Copiar archivos

0 - Salir del programa
Digite el numero de la accion:

```

```

jomiantc@JomianTC: ~/Escritorio/Programas/Punto_8
De cual archivo desear ver el contenido?
1 - arc0.txt
2 - arc1.txt
3 - arc2.txt
4 - arc3.txt
5 - arc4.txt
6 - arc5.txt
7 - arc6.txt
8 - arc7.txt
9 - arc8.txt
10 - arc9.txt

0 Regresar al menu anterior
Digite el numero del archivo:

```

```
jomiantc@JomianTC: ~/Escritorio/Programas/Punto_8
Que archivo desea copiar?
1 - arc0.txt
2 - arc1.txt
3 - arc2.txt
4 - arc3.txt
5 - arc4.txt
6 - arc5.txt
7 - arc6.txt
8 - arc7.txt
9 - arc8.txt
10 - arc9.txt

0 Regresar al menu anterior
Digite el numero del archivo: 
```


14. Desarrolle las versiones para Windows de los programas descritos en los puntos 8, 10 y utilizando únicamente las llamadas al sistema revisadas para Windows que sean necesarias.

Programa 1

```
1 #include <Windows.h>
2 #include <string.h>
3 int main()
4 {
5     char cad[50]="Hola a todo el mundo";
6     char cad2[50]="Hola a todo el otro mundo";
7     char cad3[50]="Hola a todo el ultimo mundo";
8
9     // Open a handle to the file
10    HANDLE hFile = CreateFile(
11        "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text1.txt", // Filename
12        GENERIC_WRITE, // Desired access
13        FILE_SHARE_READ, // Share mode
14        NULL, // Security attributes
15        CREATE_NEW, // Creates a new file, only if it doesn't already exist
16        FILE_ATTRIBUTE_NORMAL, // Flags and attributes
17        NULL); // Template file handle
18    if (hFile == INVALID_HANDLE_VALUE)
19    {
20        // Failed to open/create file
21        return 2;
22    }
23    // Write data to the file
24    LPSTR(strText); // For C use LPSTR (char*) or LPWSTR (wchar_t*)
25    DWORD bytesWritten;
26    WriteFile(
27        hFile, // Handle to the file
28        cad, // Buffer to write
29        strlen(cad), // Buffer size
30        &bytesWritten, // Bytes written
31        NULL); // Overlapped
32    // Close the handle once we don't need it.
33    CloseHandle(hFile);
34
35    HANDLE hFilee = CreateFile(
36        "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text2.txt", // Filename
37        GENERIC_WRITE, // Desired access
38        FILE_SHARE_READ, // Share mode
39        NULL, // Security attributes
40        CREATE_NEW, // Creates a new file, only if it doesn't already exist
41        FILE_ATTRIBUTE_NORMAL, // Flags and attributes
42        NULL); // Template file handle
43    if (hFilee == INVALID_HANDLE_VALUE)
44    {
45        // Failed to open/create file
46        return 2;
47    }
48    // Write data to the file
49    WriteFile(
50        hFilee, // Handle to the file
51        cad2, // Buffer to write
52        strlen(cad2), // Buffer size
53        &bytesWritten, // Bytes written
54        NULL); // Overlapped
55    // Close the handle once we don't need it.
56    CloseHandle(hFilee);
57
58    HANDLE hFileee = CreateFile(
59        "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text3.txt", // Filename
```

```

60 GENERIC_WRITE, // Desired access
61 FILE_SHARE_READ, // Share mode
62 NULL, // Security attributes
63 CREATE_NEW, // Creates a new file, only if it doesn't already exist
64 FILE_ATTRIBUTE_NORMAL, // Flags and attributes
65 NULL); // Template file handle
66 if (hFileeee == INVALID_HANDLE_VALUE)
67 {
68     // Failed to open/create file
69     return 2;
70 }
71 // Write data to the file
72 WriteFile(
73     hFileeee, // Handle to the file
74     cad3, // Buffer to write
75     strlen(cad3), // Buffer size
76     &bytesWritten, // Bytes written
77     NULL); // Overlapped
78 // Close the handle once we don't need it.
79 CloseHandle(hFileeee);
80 }

```



text1: Bloc de notas

Archivo Edición Formato Ver Ayuda

Hola a todo el mundo



text2: Bloc de notas

Archivo Edición Formato Ver Ayuda

Hola a todo el otro mundo



text3: Bloc de notas

Archivo Edición Formato Ver Ayuda

Hola a todo el ultimo mundo




Programa 2

```
1 // crt_stat.c
2 // This program uses the _stat function to
3 // report information about the file named crt_stat.c.
4
5 #include<stdio.h>
6 #include<stdlib.h>
7 #include <time.h>
8 #include <sys/types.h>
9 #include <sys/stat.h>
10 #include <stdio.h>
11 #include <errno.h>
12
13 int main( void )
14 {
15     struct _stat buf;
16     struct stat attrib;
17     int result;
18     char timebuf[26];
19     char* filename = "text1.txt";
20     char* filenameee = "text2.txt";
21     char* filenameeee = "text2.txt";
22
23     // Get data associated with "crt_stat.c":
24     result = _stat( filename, &buf );
25
26     // Check if statistics are valid:
27     if( result != 0 )
28     {
29         perror( "Problem getting information" );
30         switch (errno)
31         {
32             case ENOENT:
33                 printf("File %s not found.\n", filename);
34                 break;
35             case EINVAL:
36                 printf("Invalid parameter to _stat.\n");
37                 break;
38             default:
39                 /* Should never be reached. */
40                 printf("Unexpected error in _stat.\n");
41         }
42     }
43     else
44     {
45         // Output some of the statistics:
46         printf("archivo 1: \n");
47         printf( "File size      : %ld\n", buf.st_size );
48         printf( "Drive        : %c:\n", buf.st_dev + 'A' );
49         stat("text1.txt", &attrib);
50         char time[50];
51         strftime(time, 50, "%Y-%m-%d %H:%M:%S", localtime(&attrib.st_mtime));
52         printf ("%s\n", time);
53     }
54
55     result = _stat( filenameee, &buf );
56
57     // Check if statistics are valid:
```

```

58 if( result != 0 )
59 {
60     perror( "Problem getting information" );
61     switch (errno)
62     {
63         case ENOENT:
64             printf("File %s not found.\n", filenameee);
65             break;
66         case EINVAL:
67             printf("Invalid parameter to _stat.\n");
68             break;
69         default:
70             /* Should never be reached. */
71             printf("Unexpected error in _stat.\n");
72     }
73 }
74 else
75 {
76     // Output some of the statistics:
77     printf("archivo 2: \n");
78     printf( "File size      : %ld\n", buf.st_size );
79     printf( "Drive        : %c:\n", buf.st_dev + 'A' );
80     stat("text2.txt", &attrib);
81     char time[50];
82     strftime(time, 50, "%Y-%m-%d %H:%M:%S", localtime(&attrib.st_mtime));
83     printf ("%s\n", time);
84 }
85
86 result = _stat( filenameeee, &buf );
87
88 // Check if statistics are valid:
89 if( result != 0 )
90 {
91     perror( "Problem getting information" );
92     switch (errno)
93     {
94         case ENOENT:
95             printf("File %s not found.\n", filenameeee);
96             break;
97         case EINVAL:
98             printf("Invalid parameter to _stat.\n");
99             break;
100        default:
101            /* Should never be reached. */
102            printf("Unexpected error in _stat.\n");
103    }
104 }
105 else
106 {
107     // Output some of the statistics:
108     printf("archivo 3: \n");
109     printf( "File size      : %ld\n", buf.st_size );
110     printf( "Drive        : %c:\n", buf.st_dev + 'A' );
111     stat("text3.txt", &attrib);
112     char time[50];
113     strftime(time, 50, "%Y-%m-%d %H:%M:%S", localtime(&attrib.st_mtime));
114     printf ("%s\n", time);
115 }
116
117 }

```

 text1	11/03/2021 09:29 p. m.	Documento de te...	0 KB
 text2	11/03/2021 09:27 p. m.	Documento de te...	0 KB
 text3	11/03/2021 09:28 p. m.	Documento de te...	0 KB

```








archivo 1:
File size      : 0
Drive          : C:
2021-03-11 21:29:09
archivo 2:
File size      : 0
Drive          : C:
2021-03-11 21:27:39

```

```

archivo 2:
File size      : 0
Drive          : C:
2021-03-11 21:27:39
archivo 3:
File size      : 0
Drive          : C:
2021-03-11 21:28:44

```

 p4	09/03/2021 05:13 p. m.	C++ Source File	3 KB
 p4	14/03/2021 12:28 a. m.	Aplicación	43 KB
 t1	10/03/2021 08:11 p. m.	C Source File	4 KB
 t1	14/03/2021 12:28 a. m.	Aplicación	42 KB
 text1	14/03/2021 12:28 a. m.	Documento de te...	1 KB
 text2	14/03/2021 12:28 a. m.	Documento de te...	1 KB
 text3	14/03/2021 12:28 a. m.	Documento de te...	1 KB

Programa 3

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <Windows.h>
4  #include <string.h>
5
6  int main()
7  {
8      int n;
9      HANDLE hFile;
10     BOOL bFile;
11     char chBuffer [50];
12     DWORD dwNoByteToWrite = strlen(chBuffer);
13     DWORD dwNoByteWritten = 0;
14     DWORD dwNoByteToRead = strlen(chBuffer);
15     DWORD dwNoByteRead = 0;
16
17     printf("escoja que archivo quiere saber info y copiarlo a otro
        directorio: \n");
18     printf("1 para el primero\n");
19     printf("2 para el segundo\n");
20     printf("3 para el tercero\n");
21     scanf("%d", &n);
22
23     if(n==1){
24         char chBuffer [] = "Hola a todo el mundo";
25         hFile = CreateFile(
26             "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text1.txt",
27             GENERIC_READ|GENERIC_WRITE,
28             FILE_SHARE_READ,
29             NULL,
30             CREATE_NEW,
31             FILE_ATTRIBUTE_NORMAL,
32             NULL
33         );
34         //read file
35         bFile = ReadFile(
36             hFile,
37             chBuffer,
38             dwNoByteToRead,
39             &dwNoByteRead,
40             NULL
41         );
42         printf("datos del archivo: %s", chBuffer);
43         CloseHandle(hFile);
44
45         system("copy C:\\Users\\Rodrigo\\Desktop\\pruebas\\text1.txt C
            :\\Users\\Rodrigo\\Desktop\\pruebas\\destino");
46     }
47
48     if(n==2){
49         char chBuffer [] = "Hola a todo el otro mundo";
50         hFile = CreateFile(
51             "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text2.txt",
52             GENERIC_READ|GENERIC_WRITE,
53             FILE_SHARE_READ,
54             NULL,
55             CREATE_NEW,
56             FILE_ATTRIBUTE_NORMAL,
57             NULL
```



```

58     );
59     //read file
60     bFile = ReadFile(
61         hFile,
62         chBuffer,
63         dwNoByteToRead,
64         &dwNoByteRead,
65         NULL
66     );
67     printf("datos del archivo: %s", chBuffer);
68     CloseHandle(hFile);
69
70     system("copy C:\\Users\\Rodrigo\\Desktop\\pruebas\\text2.txt C
71         :\\Users\\Rodrigo\\Desktop\\pruebas\\destino");
72
73     }
74
75     if(n==3){
76         char chBuffer [] = "Hola a todo el ultimo mundo";
77         hFile = CreateFile(
78             "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text3.txt",
79             GENERIC_READ|GENERIC_WRITE,
80             FILE_SHARE_READ,
81             NULL,
82             CREATE_NEW,
83             FILE_ATTRIBUTE_NORMAL,
84             NULL
85         );
86         //read file
87         bFile = ReadFile(
88             hFile,
89             chBuffer,
90             dwNoByteToRead,
91             &dwNoByteRead,
92             NULL
93         );
94         printf("datos del archivo: %s", chBuffer);
95         CloseHandle(hFile);
96
97         system("copy C:\\Users\\Rodrigo\\Desktop\\pruebas\\text3.txt C
98             :\\Users\\Rodrigo\\Desktop\\pruebas\\destino");
99
100     }
101
102     return 0;
103 }

```

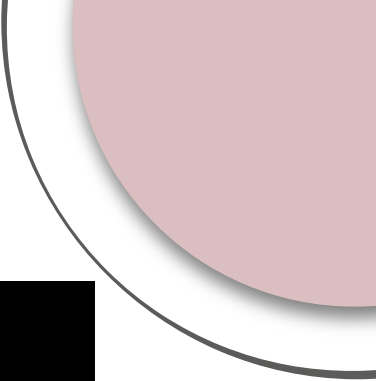
```

C:\Users\Rodrigo\Desktop\pruebas>ya
escoja que archivo quiere saber info y copiarlo a otro directorio:
1 para el primero
2 para el segundo
3 para el tercero

1
datos del archivo: Hola a todo el mundo          1 archivo(s) copiado(s).

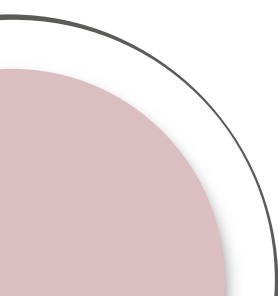
C:\Users\Rodrigo\Desktop\pruebas>_

```

```
C:\Users\Rodrigo\Desktop\pruebas>ya
escoja que archivo quiere saber info y copiarlo a otro directorio:
1 para el primero
2 para el segundo
3 para el tercero
2
datos del archivo: Hola a todo el otro mundo      1 archivo(s) copiado(s).
```

```
C:\Users\Rodrigo\Desktop\pruebas>ya
escoja que archivo quiere saber info y copiarlo a otro directorio:
1 para el primero
2 para el segundo
3 para el tercero
3
datos del archivo: Hola a todo el ultimo mundo      1 archivo(s) copiado(s).
```



CONCLUSIONES

Me parece que la practica asignada presento un buen reto para el equipo y a nivel personal, pues me hizo volver a temas que había dejado un poco abandonados por cuestiones académicas, de igual forma me permitió meterme mucho más en el ámbito de la lectura, pues uno va conociendo mucho mas acerca de los conceptos necesarios mediante libros y pdf's, tanto los proporcionados por el profesor como aquellos a los que el equipo tuvo que recurrir de internet para poder conocer mucho mas acerca del manejo de archivos tanto de Linux como de Windows.

Me parece muy bien la forma en que realizamos la implementación de código para ambos sistemas operativos, pues nos permite relacionarnos mas con ambos entornos y poder implementar cada uno de los comandos que estos poseen (identifique de una mejor manera algunas de las diferencias, así como similitudes en ciertos comandos que tuvieron que ser usados)

-Mora Ayala José Antonio

En esta segunda práctica nos pudimos adentrar a visualizar de una manera más formal el proceso que lleva el funcionamiento de una computadora, apegado al del sistema operativo. Fue interesante poder conocer y leer sobre los procesos paso a paso, pues no es algo de lo que habría ya razonado y me intrigó. Hablar de su estructura interna, de su kernel, de su arquitectura y poderlos vincular a los programas tanto para Linux como para Windows, despertó en mi aun más el querer estar en Linux e investigar más de él, pues ofrece muchas más herramientas al desarrollador y simplifica de enorme manera los proyectos desde mi perspectiva.

Fue una práctica muy interesante y llena de conocimiento.

-Ramírez Cotonieto Luis Fernando

Esta practica me ayudo mucho a entender lo más fácil e interactivo que es Linux o en este caso Ubuntu hacia los desarrolladores en general, los programas me resultaron sencillos en interesante aun que a diferencia de Windows que tuvimos muchos problemas a la hora de la implementación de las funciones y muchos errores de compatibilidades con los IDE, Linux es un entorno que cada vez me gusta más y más, su fluidez y sencillez me agradan y al ver lo fácil que es crear código en este plataforma en el lenguaje C me hace cada ves mas pensar si puedo mudar de sistema operativo definitivamente, por otro lado la parte de la teoría también fue interesante ver como algunos comandos funcionan en la terminal muestras que otros simplemente funcionan como métodos para programas, es algo que no me imaginaba que existiera, aun que los comandos que encontramos son para cuestiones básicas de escritura, lectura, directorios y copiado de archivos también hay comandos con funciones mas especificas que nos facilitan en la obtención de información y/o manipulación de los archivos

-Torres Carrillo Josehf Miguel Angel

Después de realizar esta práctica me di cuenta de la diversidad de funciones que hay entre sistemas operativos, en este caso Linux y Windows, también pude notar como es más fácil el manejo de ficheros en Linux puesto que es un sistema operativo libre, en Windows me tope con problemas como que no es de muy fácil uso estas funciones de llamadas al sistema y que no todas están actualizadas porque hasta en la misma página de Microsoft no estaba actualizada su información, pero a final de cuentas también pude lograr darme cuenta de que estas mismas funciones pueden ser muy útiles para el manejo de ficheros y de directorios

-Rodrigo Tovar Jacuinde