



# **Instituto Politecnico Nacional**

## **Escuela Superior de Cómputo**



### **Práctica 1**

### **Introducción al sistema operativo Linux y Windows (2)**

### **Sistemas Operativos**

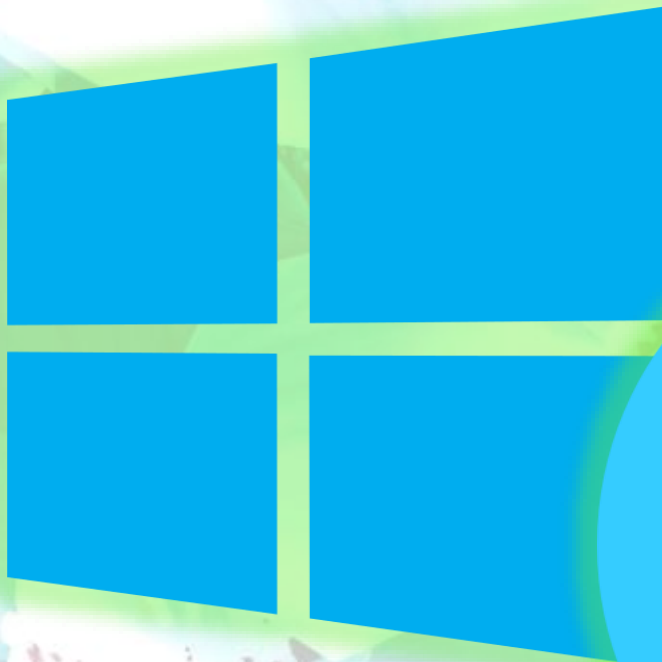
Grupo: 2CM12

Integrantes:

- ⇒ Baldovinos Gutiérrez Kevin
- ⇒ Bocanegra Heziquio Yestlanezi
- ⇒ Castañares Torres Jorge David
- ⇒ Hernández Hernández Rut Esther

Profesor

Jorge Cortes Galicia



## Contenido

Introducción .....	3
Sistema operativo .....	3
Windows .....	4
Linux .....	4
Objetivo .....	5
Desarrollo .....	5
Llamadas al sistema .....	8
open .....	8
Close .....	8
read .....	8
Write .....	9
Creat .....	9
Iseek .....	9
Access .....	9
Stat .....	9
Chmod .....	10
Chown .....	10
Fcnl .....	10
Opendir .....	10
Readdir .....	10
Llamadas a sistema Windows .....	11
OpenFile .....	11
WriteFile .....	14
CreateFile .....	14
SetFilePointer .....	14
Stat .....	15
Opendir .....	15
Readdir .....	15
.....	24
.....	24
Programa 1 .....	25
.....	25
.....	26
Programa 2 .....	27
.....	28
Programa 3 .....	29
.....	29
.....	30
Conclusiones .....	31

## Introducción

### Sistema operativo

Los sistemas operativos son, hoy más que nunca, activos estratégicos de primer orden que pueden encumbrar a una firma (Apple con iOS, Samsung con Android) o condenarla al ostracismo o la venta (Nokia con su falta de reflejos con Symbian, o Blackberry y los errores estratégicos con BB OS).

En la practica 1 nos enfocaremos en hacer la comparación de dos sistemas operativos los cuales serán Windows y Linux.

Es el software que se sitúa entre la máquina y los programas. Básicamente su función es administrar los recursos del sistema.



El sistema operativo es el software (programa o conjunto de programas) que en un sistema informático gestiona los recursos de la máquina y provee servicios básicos a los programas de aplicación. El sistema operativo siempre se ejecuta en modo privilegiado.



## Windows



Windows es un sistema operativo desarrollado por la compañía Microsoft. Consiste en un software conformado por un conjunto de programas que permiten gestionar y controlar el funcionamiento de las partes de un ordenador, como la memoria, el disco de almacenamiento y los dispositivos periféricos, y la ejecución de otros programas y aplicaciones.

El software se acciona con el encendido del hardware, es decir, de un ordenador o dispositivo. Una vez iniciada la sesión, el usuario puede realizar múltiples tareas y acciones en el ordenador a través del sistema operativo, como el manejo de diferentes programas o la instalación de nuevos dispositivos.

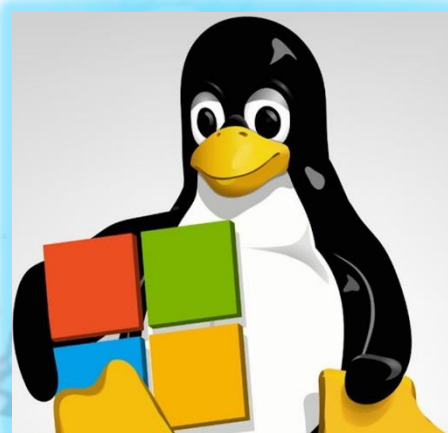
La palabra Windows, que proviene del inglés y significa *ventanas*, alude a la estructura del software que permite visualizar múltiples contenidos (como programas y archivos) organizados en compartimentos o ventanas diferentes.

El hecho de organizar los contenidos en diferentes ventanas permite que el usuario pueda visualizarlas en simultáneo, ya sea en formato de mosaico, minimizadas o superpuestas. Es decir, la interfaz de las ventanas permite ver y ejecutar varias acciones a la vez, como un procesador de datos (archivo de Word), un reproductor de video o un editor de gráficos, entre muchos otras.

## Linux

Linux es un sistema operativo semejante a Unix, de código abierto y desarrollado por una comunidad, para computadoras, servidores, mainframes, dispositivos móviles y dispositivos embebidos. Es compatible con casi todas las principales plataformas informáticas, incluyendo x86, ARM y SPARC, por lo que es uno de los sistemas operativos más soportados.

Cada versión del sistema operativo Linux gestiona los recursos de hardware, lanza y gestiona las aplicaciones, y proporciona alguna forma de interfaz de usuario. La enorme comunidad de desarrollo y la amplia gama de distribuciones significa que una versión de Linux está disponible para casi cualquier tarea, y Linux ha penetrado en muchas áreas de la informática



## Objetivo

El alumno aprende a familiarizarse con los sistemas operativos Linux y Windows (en su funcionalidad básica), mediante el desarrollo de programas bajo el lenguaje C para la invocación de llamadas al sistema propias de cada sistema operativo.

## Desarrollo

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.

Imagen 1 Archivo .txt

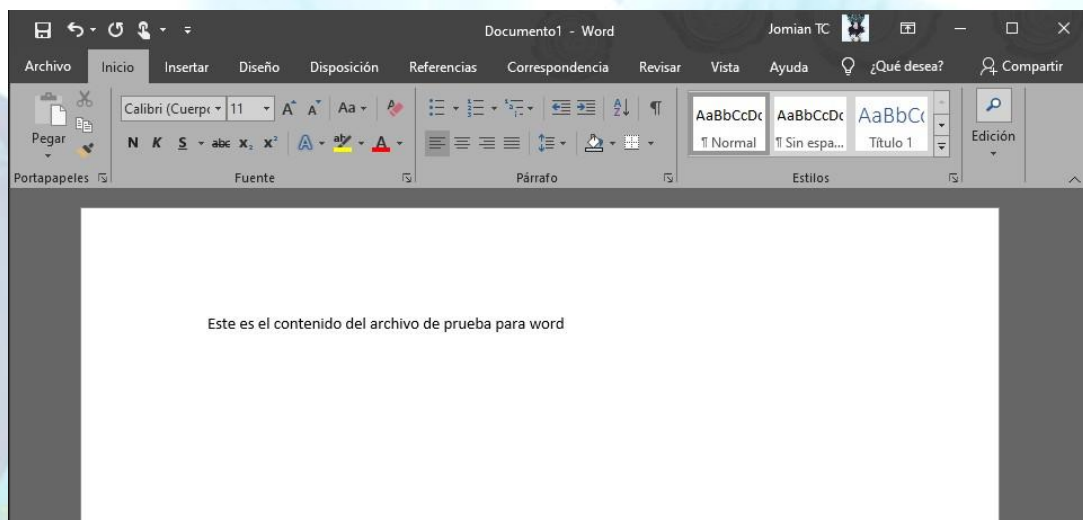
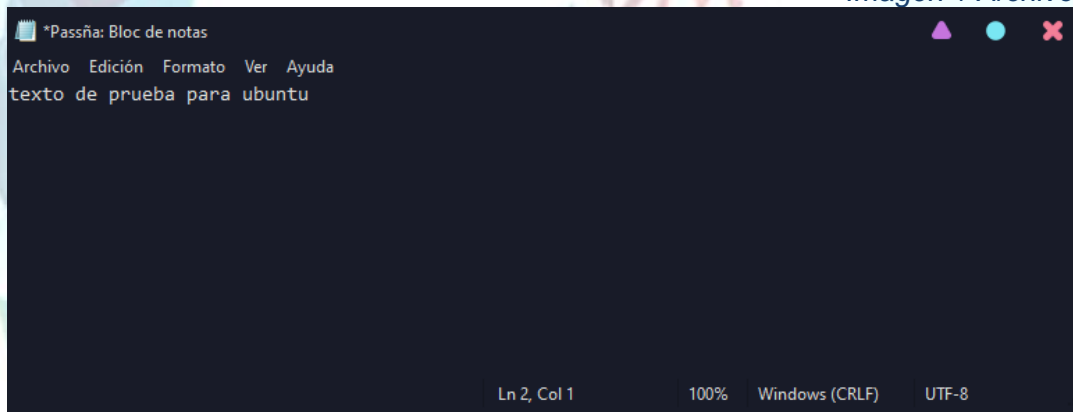


Imagen 2 Archivo .docx

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.



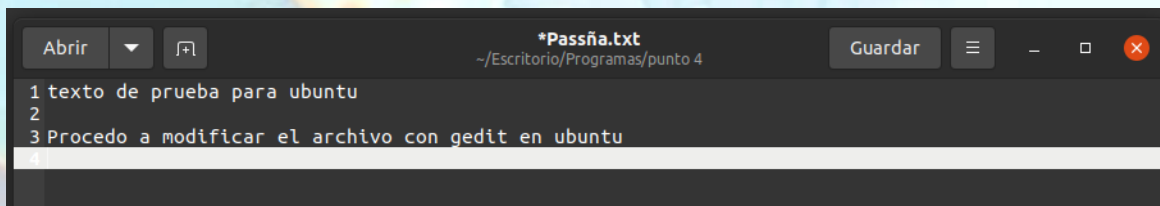


Imagen 3 Edición de archivo .txt

Al abrir gedit tenemos el siguiente texto al abrir el .docx y procedí a quitar parte del texto paraver su modificación en Windows

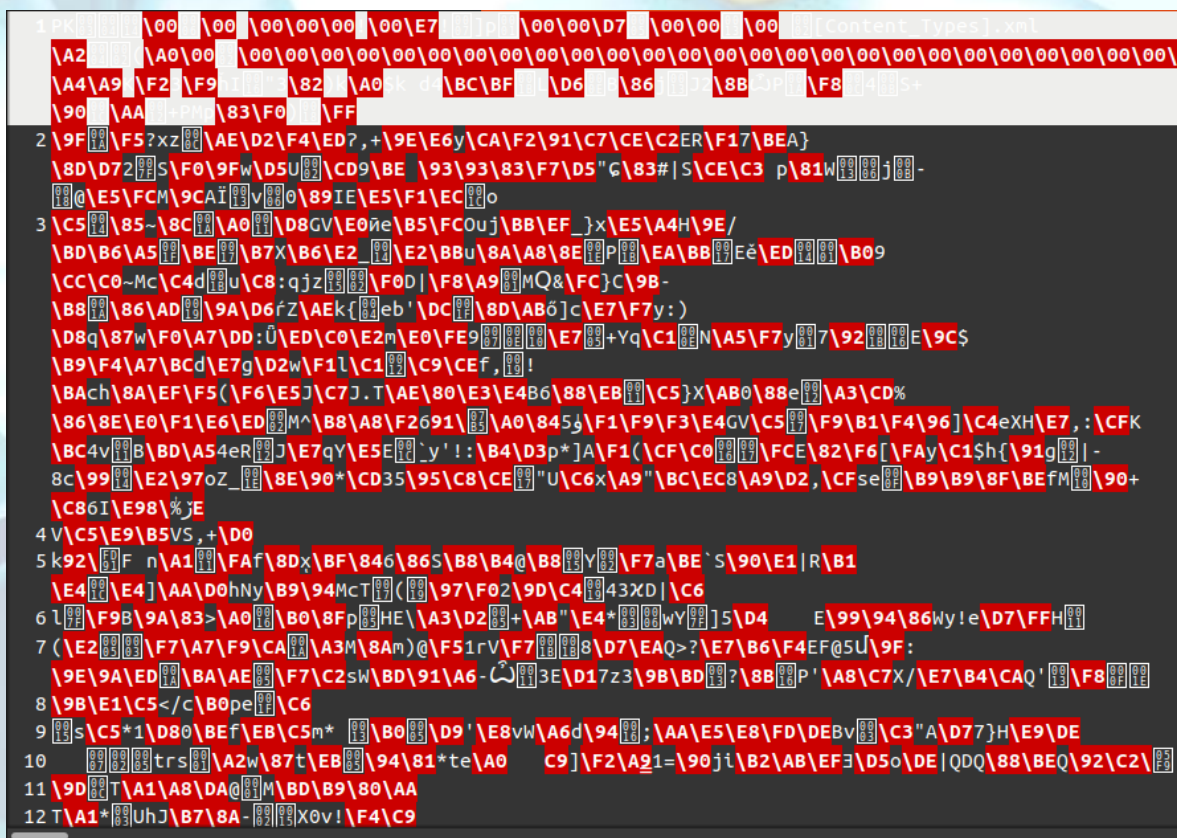
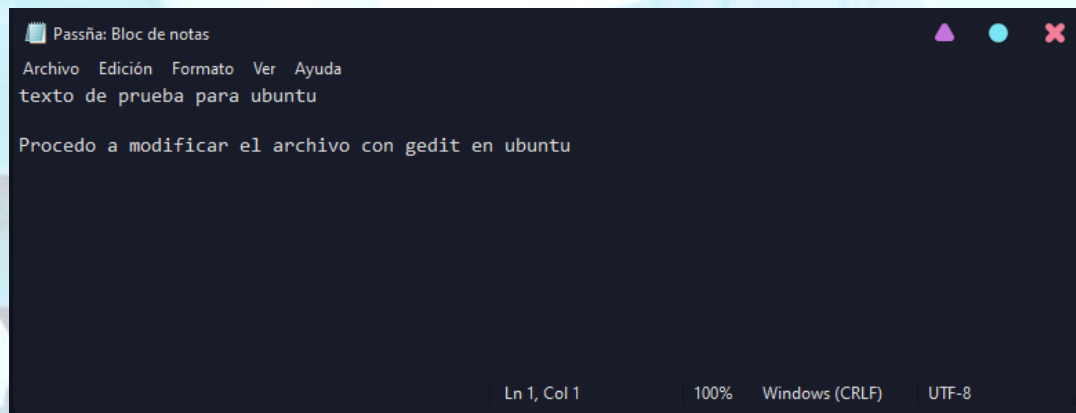


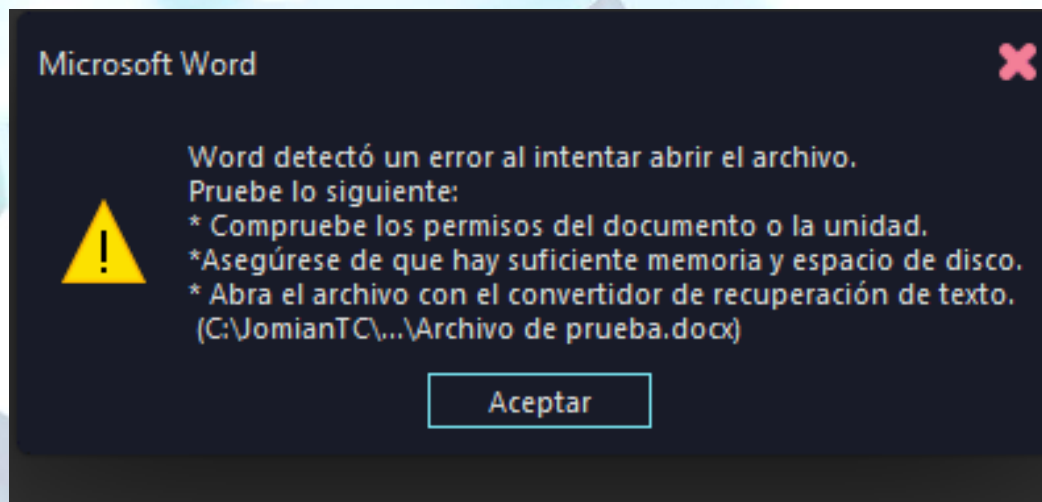
Imagen 4 gedit.docx

5. Inicie sesión en Windows y observe el contenido de sus archivos en su memoria usb.



*Imagen 5 Archivo de texto .txt*

El archivo de texto .txt fue modificado y abierto en ambos sistemas sin problemas por lo que en este caso la prueba fue un éxito



*Imagen 6 Archivo de texto .txt*

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. ¿Se observan las modificaciones realizadas en Linux?, explique el por qué si o no se observan.

### Llamadas al sistema

#### 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.

#### 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.

#### 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 está alojada en nuestra biblioteca, y count es la variable que almacenará el número de bytes encontrados.



## 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

## 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

## Iseek

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.

## 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

## 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

## 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.

## 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).

## 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.

## 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

## Readdir

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

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



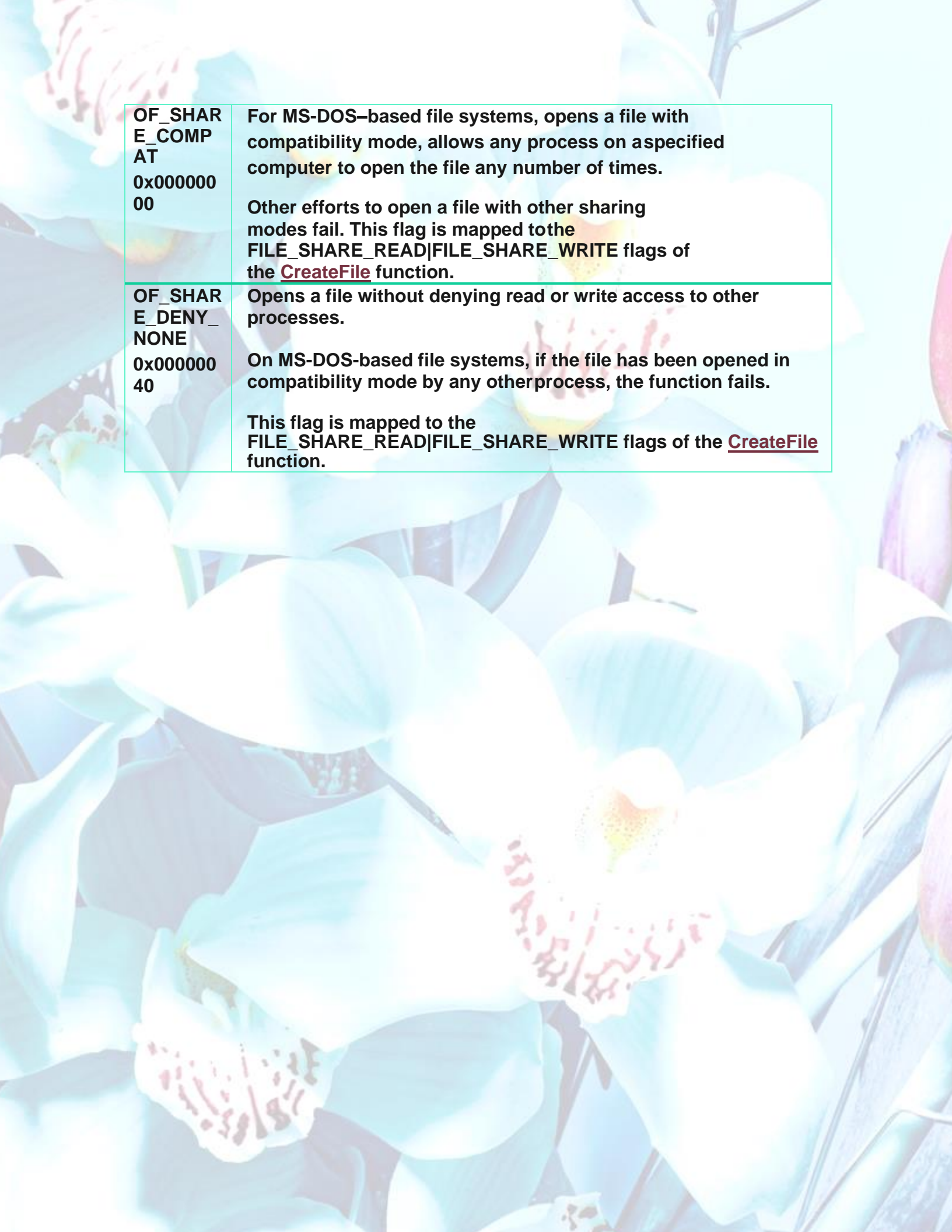
## Llamadas a sistema Windows

### 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 <u>OFSTRUCT</u> 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.



<b>OF_SHARE_COMPAT</b> <b>0x00000000</b>	<p>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.</p> <p>Other efforts to open a file with other sharing modes fail. This flag is mapped to the <b>FILE_SHARE_READ FILE_SHARE_WRITE</b> flags of the <b>CreateFile</b> function.</p>
<b>OF_SHARE_DENY_NONE</b> <b>0x00000040</b>	<p>Opens a file without denying read or write access to other processes.</p> <p>On MS-DOS-based file systems, if the file has been opened in compatibility mode by any other process, the function fails.</p> <p>This flag is mapped to the <b>FILE_SHARE_READ FILE_SHARE_WRITE</b> flags of the <b>CreateFile</b> function.</p>



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

Memory resource notification

Mutex

Named pipe

Pipe

Process

Semaphore

Thread

Transaction

Waitable timer

Access token

Communications device

Console input

Console screen buffer

Event

File

File mapping

I/O completion port

Job

Mailslot

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  
                );
```

## WriteFile

Escribe datos en un archivo o dispositivo de entrada/salida específico. **BOOL WriteFile(**

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

## 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,  
DWORDHANDLE  
);
```

## 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  
);
```





## 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.

## 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

## 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

8. 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.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

char ruta[53] = {"/home/jomiantc/Escritorio/Programas/Punto_8/arc0.txt"};
char save[1];
char cadena[46] = {"Este es el archivo numero 0 de los txt creados"};

int variable, i;
mode_t mode = S_IRUSR | S_IWUSR;

int main (void){

    for (i = 0; i < 10; ++i){

        sprintf(save,"%d",i);

        ruta[47] = save[0];
        cadena[26] = save[0];

        variable = creat(ruta, mode);
        variable = open(ruta, O_WRONLY);

        write(variable, &cadena, sizeof(cadena));

        close(variable);
    }
}
```

*Imagen 7 Código 0*



9. 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     do{
20
21         printf("A que archivo deseas cambiarle los permisos de usuario?\n");
22         printf("1 - arc0.txt\n");
23         printf("2 - arc1.txt\n");
24         printf("3 - arc2.txt\n");
25         printf("4 - arc3.txt\n");
26         printf("5 - arc4.txt\n");
27         printf("6 - arc5.txt\n");
28         printf("7 - arc6.txt\n");
29         printf("8 - arc7.txt\n");
30         printf("9 - arc8.txt\n");
31         printf("10 - arc9.txt\n\n");
32
33         printf("0 Salir del programa\n\n");
34
35         printf("Digite el numero del archivo: \t");
36
37         scanf("%d", &i);
38
39         system("clear");
40
41         if(i == 0){
42             i = -1;
43         }
44
45         else{
46             modificar();
47         }
48
49         system("clear");
50
51     }while(i != -1);
52 }
53
54
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 }
```

*Imagen 11 Código 3*

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:

*Imagen 12 Permisos de usuario desde linux*

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:

*Imagen 13 Permisos de usuario desde linux 1*



10. 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 }
```

Imagen 14 Código 4

```
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
```

Imagen 15 Tamaño del archivo

11. 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         system("clear");
49     }while(i != -1);
50 }
51
52 void leer(){
53
54     do{
55
56         if(i == 1) printf("De cual archivo desear ver el contenido?\n");
57         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         impricotent();
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 impricotent(){
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

```

Imagen 17 Código 6

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

*Imagen 18 Código 7*

```
Que deseas hacer?
1 - Leer contenido de archivo
2 - Copiar archivos

0 - Salir del programa

Digite el numero de la accion: █
```

*Imagen 19 Leer archivo*

```
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: █
```

*Imagen 20 Ver contenido*

12. 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 hFileee = 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 (hFileee == INVALID_HANDLE_VALUE)
44    {
45        // Failed to open/create file
46        return 2;
47    }
48    // Write data to the file
49    WriteFile(
50        hFileee, // 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(hFileee);
57
58    HANDLE hFileeee = CreateFile(
59        "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text3.txt", // Filename
```

Imagen 21 Código programa 1



```
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 }
```

*Imagen 22 Código programa 1.1*

## 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:
```

Imagen 23 Código programa 2

```

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 }

```

Imagen 24 Código programa 2.1



## 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
18    directorio: \n");
19    printf("1 para el primero\n");
20    printf("2 para el segundo\n");
21    printf("3 para el tercero\n");
22    scanf("%d", &n);
23
24    if(n==1){
25        char chBuffer [] = "Hola a todo el mundo";
26        hFile = CreateFile(
27            "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text1.txt",
28            GENERIC_READ|GENERIC_WRITE,
29            FILE_SHARE_READ,
30            NULL,
31            CREATE_NEW,
32            FILE_ATTRIBUTE_NORMAL,
33            NULL
34        );
35        //read file
36        bFile = ReadFile(
37            hFile,
38            chBuffer,
39            dwNoByteToRead,
40            &dwNoByteRead,
41            NULL
42        );
43        printf("datos del archivo: %s", chBuffer);
44        CloseHandle(hFile);
45
46        system("copy C:\\Users\\Rodrigo\\Desktop\\pruebas\\text1.txt C
47        :\\Users\\Rodrigo\\Desktop\\pruebas\\destino");
48    }
49
50    if(n==2){
51        char chBuffer [] = "Hola a todo el otro mundo";
52        hFile = CreateFile(
53            "C:\\Users\\Rodrigo\\Desktop\\pruebas\\text2.txt",
54            GENERIC_READ|GENERIC_WRITE,
55            FILE_SHARE_READ,
56            NULL,
57            CREATE_NEW,
58            FILE_ATTRIBUTE_NORMAL,
59            NULL
60        );
```

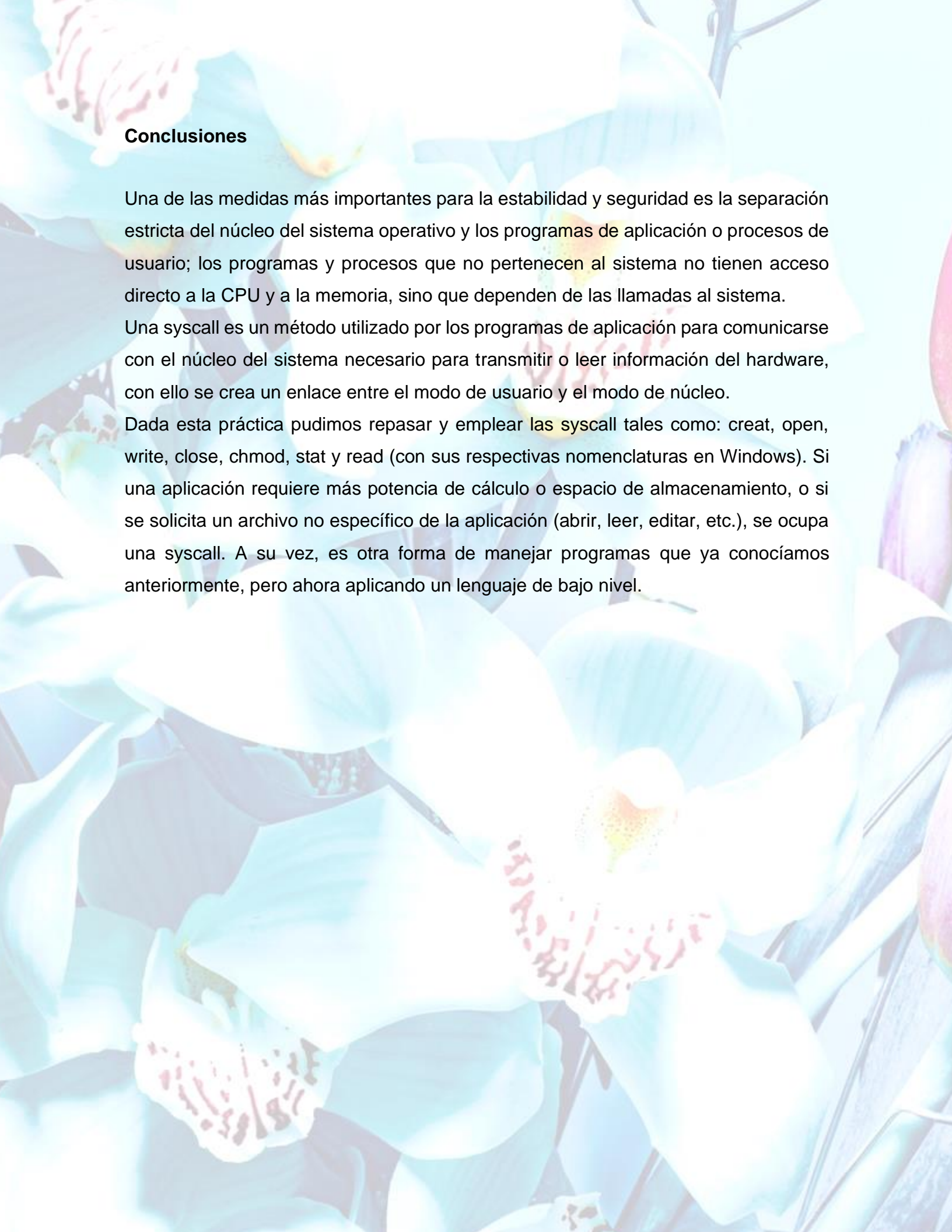
Imagen 25 Código programa 3

```

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

```

*Imagen 26 Código programa 3.1*



## Conclusiones

Una de las medidas más importantes para la estabilidad y seguridad es la separación estricta del núcleo del sistema operativo y los programas de aplicación o procesos de usuario; los programas y procesos que no pertenecen al sistema no tienen acceso directo a la CPU y a la memoria, sino que dependen de las llamadas al sistema.

Una syscall es un método utilizado por los programas de aplicación para comunicarse con el núcleo del sistema necesario para transmitir o leer información del hardware, con ello se crea un enlace entre el modo de usuario y el modo de núcleo.

Dada esta práctica pudimos repasar y emplear las syscall tales como: creat, open, write, close, chmod, stat y read (con sus respectivas nomenclaturas en Windows). Si una aplicación requiere más potencia de cálculo o espacio de almacenamiento, o si se solicita un archivo no específico de la aplicación (abrir, leer, editar, etc.), se ocupa una syscall. A su vez, es otra forma de manejar programas que ya conocíamos anteriormente, pero ahora aplicando un lenguaje de bajo nivel.






Imagen 1 Archivo .txt .....	5
Imagen 2 Archivo .docx .....	5
Imagen 3 Edición de archivo .txt .....	6
Imagen 4 gedit.docx.....	6
Imagen 5 Archivo de texto .txt .....	7
Imagen 6 Archivo de texto .txt .....	7
Imagen 7 Código 0.....	16
Imagen 8 Código 1.....	17
Imagen 9 Código 2.....	18
Imagen 10 Código programa 1.1 .....	19
Imagen 11 Código 3.....	19
Imagen 12 Permisos de usuario desde linux .....	20
Imagen 13 Permisos de usuario desde linux 1 .....	20
Imagen 14 Código 4.....	21
Imagen 15 Tamaño del archivo .....	21
Imagen 16 Código 5.....	22
Imagen 17 Código 6.....	23
Imagen 18 Código 7.....	24
Imagen 19 Leer archivo .....	24
Imagen 20 Ver contenido.....	24
Imagen 21 Código programa 1 .....	25
Imagen 22 Código programa 1.1 .....	26
Imagen 23 Código programa 2 .....	27
Imagen 24 Código programa 2.1 .....	28
Imagen 25 Código programa 3 .....	29
Imagen 26 Código programa 3.1 .....	30