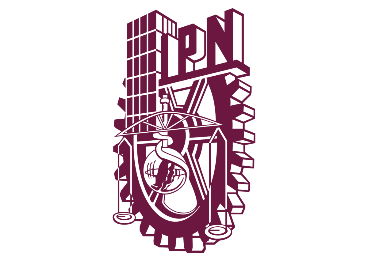
Logotipo

Descripción generada automáticamente

***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

Pantalla de computadora

Descripción generada automáticamente con confianza baja

Contenido

[Introducción 3](#_Toc97838537)

[Sistema operativo 3](#_Toc97838538)

[Windows 4](#_Toc97838539)

[Linux 4](#_Toc97838540)

[Objetivo 5](#_Toc97838542)

[Desarrollo 5](#_Toc97838543)

[Llamadas al sistema 8](#_Toc97838544)

[open 8](#_Toc97838545)

[Close 8](#_Toc97838546)

[read 8](#_Toc97838547)

[Write 9](#_Toc97838548)

[Creat 9](#_Toc97838549)

[Iseek 9](#_Toc97838550)

[Access 9](#_Toc97838551)

[Stat 9](#_Toc97838552)

[Chmod 10](#_Toc97838553)

[Chown 10](#_Toc97838554)

[Fcnlt 10](#_Toc97838555)

[Opendir 10](#_Toc97838556)

[Readdir 10](#_Toc97838557)

[Llamadas a sistema Windows 11](#_Toc97838558)

[OpenFile 11](#_Toc97838559)

[WriteFile 14](#_Toc97838560)

[CreateFile 14](#_Toc97838561)

[SetFilePointer 14](#_Toc97838562)

[Stat 15](#_Toc97838563)

[Opendir 15](#_Toc97838564)

[Readdir 15](#_Toc97838565)

[24](#_Toc97838566)

[24](#_Toc97838567)

[Programa 1 25](#_Toc97838568)

[25](#_Toc97838569)

[26](#_Toc97838570)

[Programa 2 27](#_Toc97838571)

[28](#_Toc97838572)

[Programa 3 29](#_Toc97838573)

[29](#_Toc97838574)

[30](#_Toc97838575)

[Conclusiones 31](#_Toc97838576)

# 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.

Usuario

Aplicaciones

Sistemas Operativos

Sistemas Operativos

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

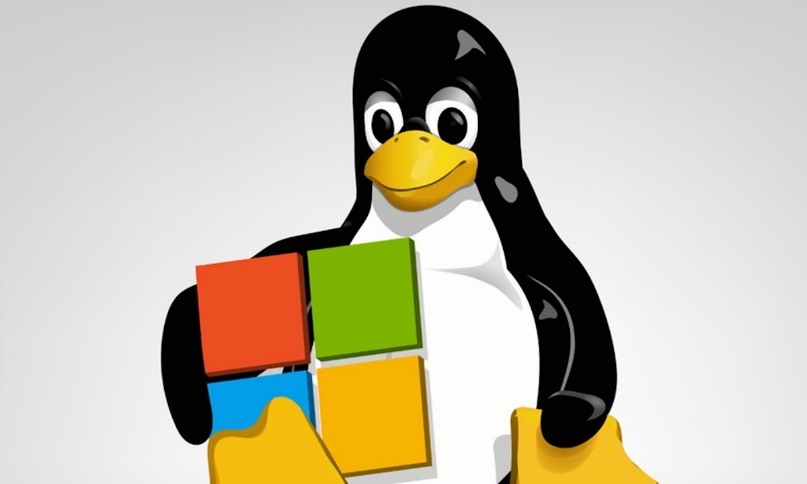
Windows es un sistema operativo desarrollado por la compañía Microsoft. Consiste en un [software](https://www.caracteristicas.co/software/) conformado por un conjunto de programas que permiten gestionar y controlar el funcionamiento de las partes de un ordenador, como la memoria, el disco de almacenamiento y los dispositivos periféricos, y la ejecución de otros programas y aplicaciones.

El software se acciona con el encendido del [hardware](https://www.caracteristicas.co/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.

Captura de pantalla con la imagen de una pantalla

Descripción generada automáticamente

Imagen 1 Archivo .txt

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Imagen 2 Archivo .docx

1. Inicie sesión en Linux.
2. 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.
3. Edite tanto el contenido del archivo de texto como de Word modificándolo mediante el uso de gedit. Guarde sus archivos.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Imagen 3 Ediciòn de archivo .txt

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

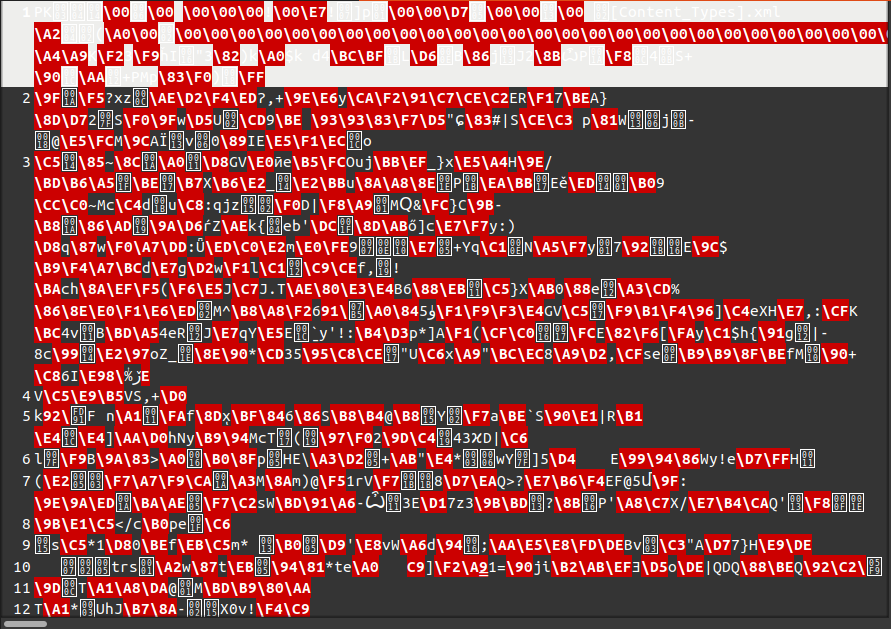


Imagen 4 gedit.docx

1. Captura de pantalla con la imagen de una pantalla

   Descripción generada automáticamenteInicie 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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

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 paqueteria 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 maquina puede abrir y ejecutar sin problemas

1. ¿Se observan las modificaciones realizas 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 especifico, si el archivo no está creado también podemos poner como parámetro en el comando el comando creat que crear 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 siga manera

Read (int fd, void \*buf, size\_y 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.

### 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 funcion nos permite reposicionar el puntero de lectura en nuestro archivo para empezar a leer o escribir desde ahi

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).

### Fcnlt

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 [OFSTRUCT](https://docs.microsoft.com/en-us/windows/desktop/api/winbase/ns-winbase-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](https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-createfilea) 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](https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-createfilea) function. |

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 especifico 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 p dispositivo de entrada/salida especifico 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,

DWORD HANDLE

);

## 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.

Texto

Descripción generada automáticamente con confianza media

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



Imagen 8 Código 1

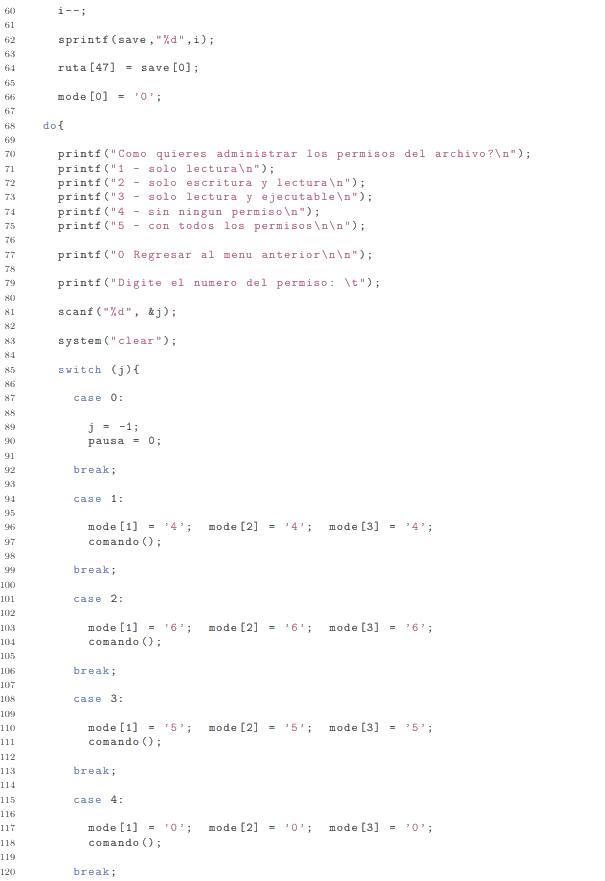


Imagen 9 Código 2

Imagen 10 Código programa 1.1

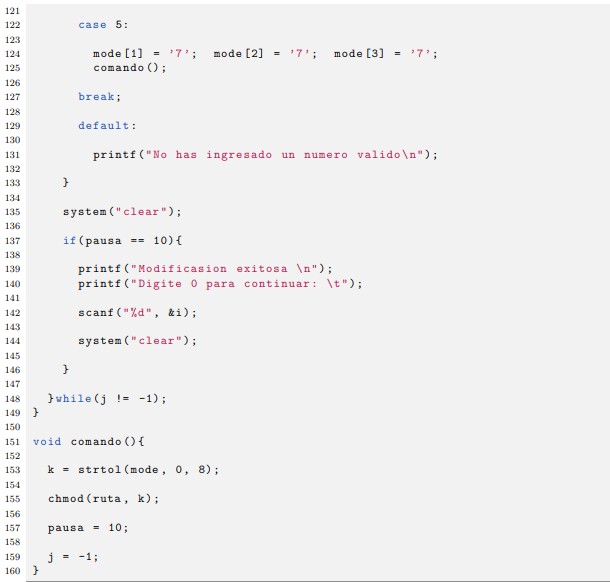


Imagen 11 Código 3

Texto

Descripción generada automáticamente

Imagen 12 Permisos de usuario desde linux

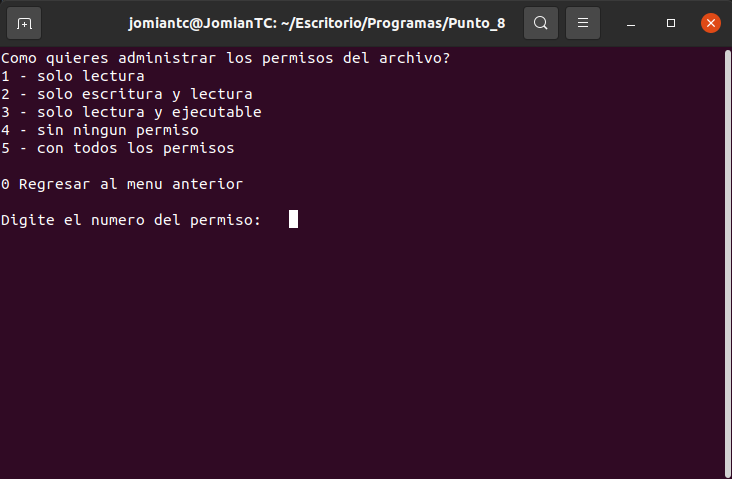


Imagen 13 Permisos de usuario desde linux 1

Texto

Descripción generada automáticamente10. 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.

Imagen 14 Código 4

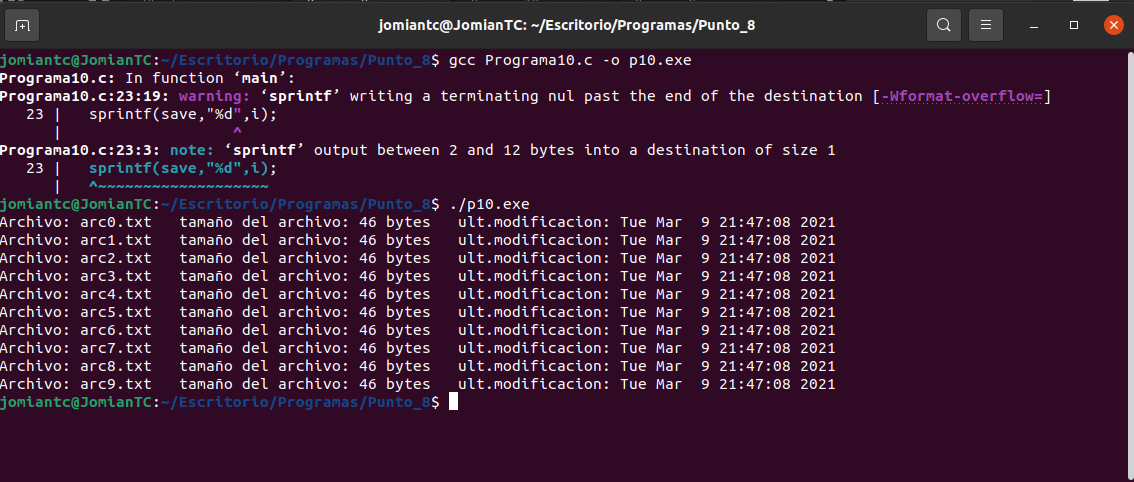


Imagen 15 Tamaño del archivo

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.



Imagen 16 Código 5

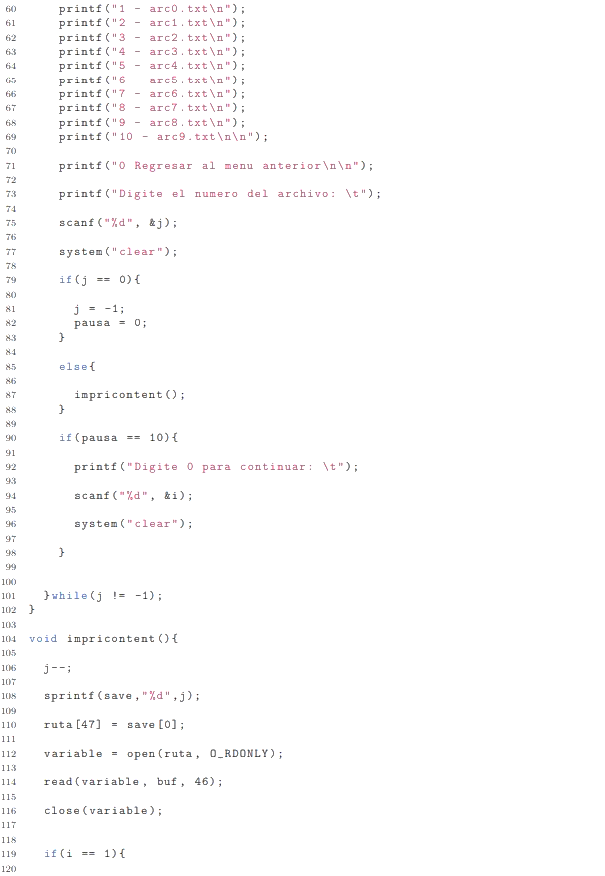


Imagen 17 Código 6

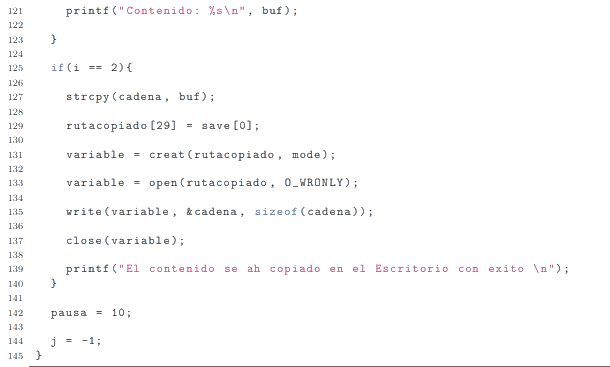


Imagen 18 Código 7

# 

Imagen 19 Leer archivo

# Texto Descripción generada automáticamente

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

# 

Imagen 21 Código programa 1

# 

Imagen 22 Código programa 1.1

## Programa 2



Imagen 23 Código programa 2

# 

Imagen 24 Código programa 2.1

## Programa 3

# 

Imagen 25 Código programa 3

# 

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](#_Toc97838450)

[Imagen 2 Archivo .docx 5](#_Toc97838451)

[Imagen 3 Ediciòn de archivo .txt 6](#_Toc97838452)

[Imagen 4 gedit.docx 6](#_Toc97838453)

[Imagen 5 Archivo de texto .txt 7](#_Toc97838454)

[Imagen 6 Archivo de texto .txt 7](#_Toc97838455)

[Imagen 7 Código 0 16](#_Toc97838456)

[Imagen 8 Código 1 17](#_Toc97838457)

[Imagen 9 Código 2 18](#_Toc97838458)

[Imagen 10 Código programa 1.1 19](#_Toc97838459)

[Imagen 11 Código 3 19](#_Toc97838460)

[Imagen 12 Permisos de usuario desde linux 20](#_Toc97838461)

[Imagen 13 Permisos de usuario desde linux 1 20](#_Toc97838462)

[Imagen 14 Código 4 21](#_Toc97838463)

[Imagen 15 Tamaño del archivo 21](#_Toc97838464)

[Imagen 16 Código 5 22](#_Toc97838465)

[Imagen 17 Código 6 23](#_Toc97838466)

[Imagen 18 Código 7 24](#_Toc97838467)

[Imagen 19 Leer archivo 24](#_Toc97838468)

[Imagen 20 Ver contenido 24](#_Toc97838469)

[Imagen 21 Código programa 1 25](#_Toc97838470)

[Imagen 22 Código programa 1.1 26](#_Toc97838471)

[Imagen 23 Código programa 2 27](#_Toc97838472)

[Imagen 24 Código programa 2.1 28](#_Toc97838473)

[Imagen 25 Código programa 3 29](#_Toc97838474)

[Imagen 26 Código programa 3.1 30](#_Toc97838475)