

INSTITUTO POLITÉCNICO NACIONAL

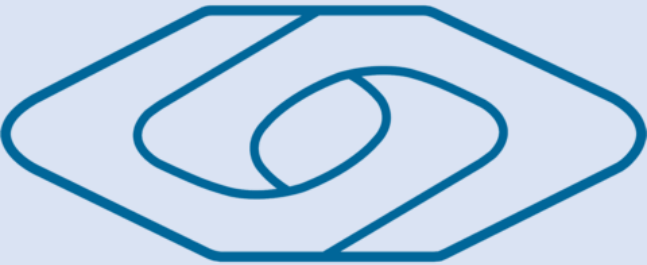
ESCUELA SUPERIOR DE COMPUTO



“Practica 2”

Sistemas Operativos

INSTITUTO POLITÉCNICO NACIONAL



ESCOM

Equipo 7

- Héctor Chávez Rodríguez
 - María Fernanda Delgado Mendoza
 - Jonathan Said Gómez Marbán
 - Luis Antonio Ramírez Fárias
-
- **Grupo:** 4CM1

I. Introducción Teórica

¿Qué es un Sistema Operativo?

El sistema operativo es el conjunto de programas informáticos, que permite la administración eficaz de los recursos de un ordenador.

El sistema operativo también es conocido como sistema o software y puede definirse como el conjunto de programas que están hechos, específicamente, para ejecutar varias tareas en las que actúa como intermediario entre el usuario y el ordenador.

El sistema operativo representa el programa más importante de la computadora, ya que comienza a trabajar nada más encender el equipo, ya que se encarga de gestionar el hardware y permite la interacción con el usuario.

Algunas de las funciones básicas de este software son:

- Gestionar procesos o recursos para que los programas puedan ejecutarse de manera correcta.
- Administrar los puertos de entrada y salida, por ejemplo: micrófonos, altavoces, impresoras o el monitor.
- Garantizar la seguridad del ordenador, impidiendo el acceso a ciertos archivos

Principales características del Sistema Operativo Linux:

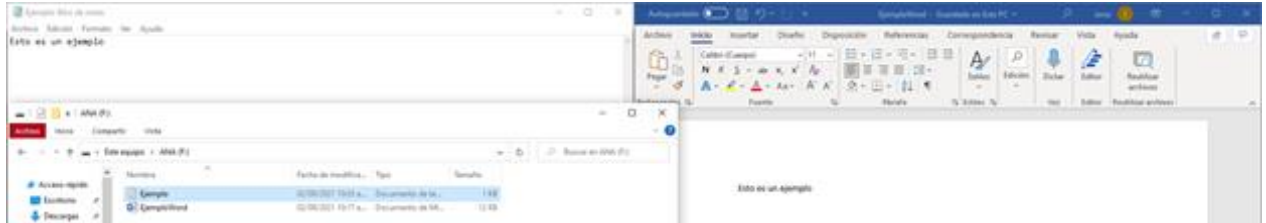
- Libre: Se puede descargar de internet, se puede copiar y distribuir sin que por ello se incurra en ningún tipo de delito.
- Eficiente: Linux aprovecha bien los recursos hardware. Incluso los viejos Pentium pueden funcionar bien con Linux y servir para alguna tarea.
- Hecho por voluntarios: Cuando alguien necesita un determinado programa, simplemente lo crea y lo pone al servicio de la comunidad para que lo use y para que cada cual lo mejore y lo adapte a sus propias necesidades.
- Multitarea: Pueden funcionar varios programas al mismo tiempo en la misma máquina.
- Multiplataforma: Hay versiones de Linux para gran cantidad de plataformas.
- Multiusuario: Varios usuarios pueden conectarse y usar el mismo ordenador a la vez.
- Estable: Linux es un sistema operativo muy maduro, probado durante mucho tiempo.
- Hay miles de programas libres: Hay una gran cantidad de programas, desde procesadores de texto hasta programas de dibujo pasando por todo tipo de servidores, totalmente libres y gratuitos que se pueden descargar e instalar desde el propio entorno de Linux.

Principales características del Sistema Operativo Windows:

- **Es privada:** su sistema operativo pertenece a la compañía Microsoft y su código no es libre, por lo que no puede ser utilizado por los usuarios. Muchos consideran que Windows es en realidad un subsistema operativo, ya que para su funcionamiento precisa de otro sistema operativo llamado MS-DOS.
- **Larga evolución:** con el paso del tiempo, Microsoft ha ido presentando diferentes versiones de Windows con distintos cambios evolutivos. Como el recorrido es muy largo, ha podido ofrecer diferentes interfaces y funciones.
- **Ventanas:** en Windows la interfaz gráfica está representada por ventanas, que son diferentes cuadros que presentan la información al usuario. De ahí deriva su nombre, pues es radicalmente diferente a su antecesor.
 - **Escritorio:** su escritorio abarca la pantalla del monitor. Todas las versiones disponibles poseen una plataforma visualmente amena sobre la que presentan todas sus características.

II. Desarrollo Experimental

Paso 1



Paso 2 y 3

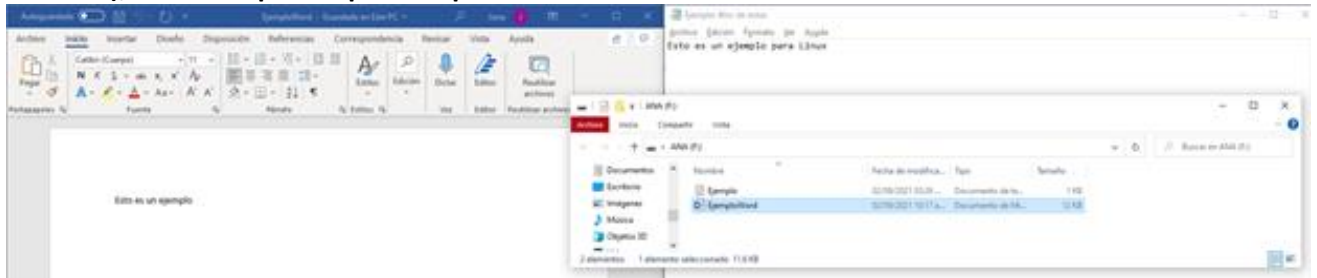


Paso 4



Paso 5

El documento tipo docx no puede ser modificado con el comando gedit (ejecutado desde la terminal), mientras que el tipo txt sí pudo ser modificado



Paso 6

Manual de Linux:

Open: Te permite abrir un archivo o directorio

```
open <filename>
```

```
open <filename>
```

Close: Cierra archivos / documentos.

Read: interrumpe la ejecución del shell hasta que el usuario introduzca una cadena de caracteres (aunque sea vacía) en su entrada estándar.

Las palabras que componen la cadena de caracteres escrita por el usuario se asignan a las variables cuyos nombres se pasan como argumentos al comando **read**:

```
read [-opciones] [Cadena] variable
```

```
read miVariable
```

Write: La llamada al sistema write hace que los primeros bytes del buffer sean escritos en el archivo asociado con el descriptor de archivos fichero. Envía el número de bytes escritos realmente. Puede ser menor que bytes si ha habido un error en el descriptor de archivos o si el controlador del dispositivo subyacente es sensible al tamaño del bloque.

Create: Crea archivos

lseek: Permite colocar el puntero de fichero después del final de fichero

lseek - reposition read/write file offset

Access: Comprueba los permisos de usuario para un fichero

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

Stat: Se utiliza para mostrar por pantalla los atributos de un archivo.

```
stat aprender-linux.txt
  File: 'aprender-linux.txt'
  Size: 57          Blocks: 8          IO Block: 4096   regular file
Device: fc00h/64512d Inode: 3674171   Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  vagrant)   Gid: ( 1000/  vagrant)
Access: 2020-01-28 18:53:37.946459999 +0000
Modify: 2020-01-28 18:52:29.168088002 +0000
Change: 2020-01-28 18:52:29.172090002 +0000
Birth: -
```

Chmod: Puedes cambiar los permisos dados a un archivo usando el comando chmod. chmod puede ser usado de dos maneras. La primera es usando argumentos simbólicos, la segunda es usando argumentos numéricos. Argumentos simbólicos:
Se escribe chmod seguido de un espacio y una letra:

- a significa *todos*
- u significar *usuario*
- g significa grupo
- o significa *otros*

Luego se escribe + o - para agregar permiso, o para eliminarlo. Después se ingresa uno o más símbolos de permiso (r, w, x). Todo seguido por el nombre del archivo o la carpeta.

```
chmod a+r filename # todos pueden leerlo
chmod a+rw filename # todos puede leerlo y escribirlo
chmod o-rwx filename # otros (no el propietario, no en el mismo grupo del archivo) no pueden
```

Chown: El propietario (y el usuario root) puede cambiar el propietario a otro usuario

```
chown <owner> <file>
```

Fcntl: Este módulo realiza control de archivos y control de E/S en descriptores de ficheros.

Opendir: Esta función abre una secuencia de directorio correspondiente al nombre del directorio.

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

Readdir: devuelve la estructura que apunta a dirent, esta estructura representa la siguiente entrada de directorio en el flujo de directorio al que apunta dir, si se lee el fin de archivo o se produce un error, se devuelve NULL

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

Paso 7

Manual de Windows:

Openfile: Use este componente dentro de Windows aplicación basada en archivos como una solución sencilla para la selección de archivos

```
HFILE OpenFile(
    LPCSTR    lpFileName,
    LPOFSTRUCT lpReOpenBuff,
    UINT      uStyle
);
```

Closefile: Cierra el archivo que está en uso

Readfile: Un subproceso usa las funciones ReadFile o ReadFileEx para leer desde un recurso de comunicaciones

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

Writefile: La función se utiliza WriteFile o WriteFileEx para escribir en un recurso de comunicaciones

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

Createfile: Crea, abre o trunca un fichero, pipe, recurso de comunicación, dispositivo de disco o consola.

```
HANDLE CreateFile2(
    LPCWSTR    lpFileName,
    DWORD      dwDesiredAccess,
    DWORD      dwShareMode,
    DWORD      dwCreationDisposition,
    LPCREATEFILE2_EXTENDED_PARAMETERS pCreateExParams
);
```

SetFilePointer: Mueve el puntero de archivo del archivo especificado.

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

Stat: Obtiene información del estado de un archivo

```
int _stat(  
    const char *path,  
    struct _stat *buffer  
);
```

Opendir: Abre un gestor de directorio

```
opendir(string $path, resource $context = ?): resource
```

Readdir: Lee una entrada desde un gestor de directorio

```
readdir(resource $dir_handle = ?): string
```

No hay nada llamado `chmod` en Windows porque el modelo de seguridad de Windows es diferente a Linux.

“`attrib`” puede establecer atributos de solo lectura / ocultos de un solo archivo; no proporciona controles específicos como lo `icacls` hace.

`icacls` establece / restablece las listas de control de acceso, por lo que puede otorgar / denegar derechos para SID y grupos individuales. Sin embargo, es bastante complicado.

Programas Windows

Copiar archivos

Programa que ejecutara el comando copy, recibirá la cadena que en este caso será la ruta del archivo a copiar, verificara si existe el archivo, si esto es correcto copiará el archivo, de lo contrario no existe el archivo.

Mediante otra función el archivo encontrado lo copiará en la ruta de destino.

En la función main mostrara la función init que realizara el trabajo antes creado.

Programa ejecutado.

```

C:\Users\Hector\OneDrive\Desktop\Escuela\SO\Practicas\P2\W1>copy.exe
Ruta del archivo: C:\Users\Hector\OneDrive\Desktop\Escuela\SO\Practicas\P2\W1
Destino: C:\Users\Hector\OneDrive\Desktop\Escuela
1 archivo(s) copiado(s).
Ruta del archivo:
  
```

Programa List_dir

Se necesitara un puntero que apunte la dirección, necesitaremos una variable tipo struct para entrar a los

```

void copy_file(string ruta, string destino)
{
    FILE * archivo;
    string cadena;
    if (archivo = fopen(ruta.c_str(), "r"))
    {
        cadena = "copy " + ruta + " " + destino;
        system(cadena.c_str());
        fclose(archivo);
    }
    else
    {
        cout << "El archivo no existe" << endl;
    }
}

void init()
{
    cout << "Ruta del archivo: ";
    string ruta;
    getline(cin, ruta);
}

void list_dir(string dir){

    DIR * directorio;
    struct dirent * elemento;
    string elem;
    if (directorio = opendir(dir.c_str()))
    {
        while(elemento = readdir(directorio))
        {
            elem= elemento->d_name;
            cout<<elem<<endl;
        }
    }
    closedir(directorio);
}

void init(){

    cout<<"Ingrese la ruta del directorio: ";
    string dir;
    getline(cin,dir);
    list_dir(dir);
    init();
}

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

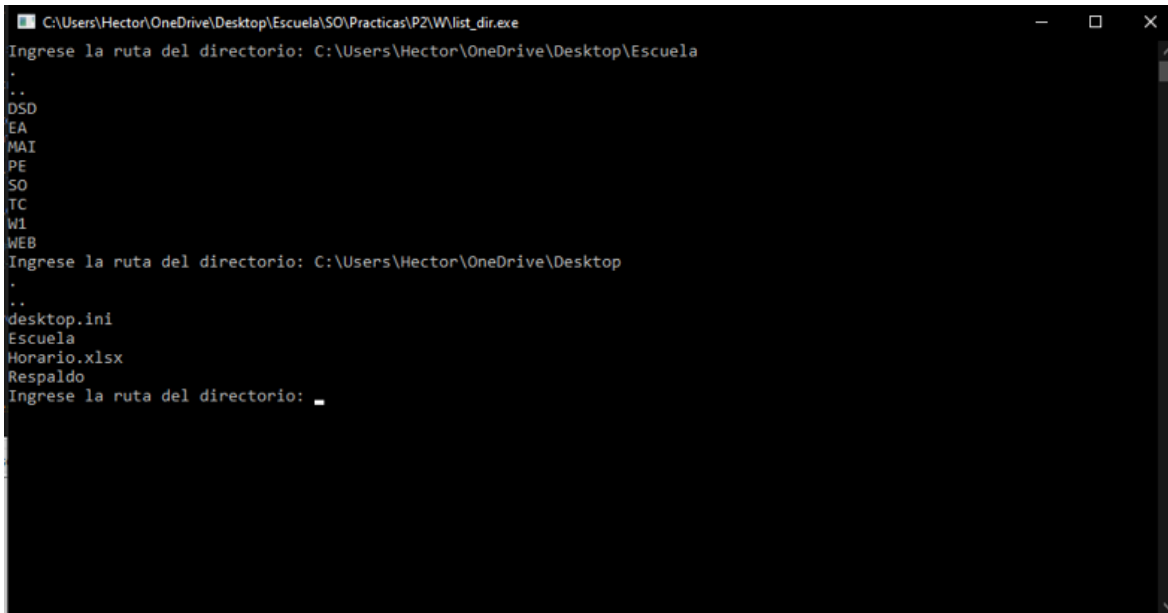
    init();
    system("pause");
    return 0;
}
  
```


datos, una variable string para guardar los elementos, si nuestro apuntador está abierto, con ayuda de un ciclo ira leyendo cada elemento del directorio abierto y a su vez mostrara cada elemento de él.

Una función init que ejecutara la función anterior, se ingresara los datos (ruta), y mostrara.

Función main, se invocará la función principal.

Programa ejecutado.



```
C:\Users\Hector\OneDrive\Desktop\Escuela\SO\Practicas\P2\W\list_dir.exe
Ingrese la ruta del directorio: C:\Users\Hector\OneDrive\Desktop\Escuela
.
..
DSD
EA
MAI
PE
SO
TC
W1
WEB
Ingrese la ruta del directorio: C:\Users\Hector\OneDrive\Desktop
.
..
desktop.ini
Escuela
Horario.xlsx
Respaldo
Ingrese la ruta del directorio: _
```

Programa Permisos.

```
int main(){
    char mode[4]="";
    char buf[100]="C:\\Users\\Hector\\OneDrive\\Desktop\\Escuela\\SO\\Practicas\\P2";
    scanf("%s", mode);
    int i = atoi(mode);
    chmod(buf,i);
}
```

Función Que modificara los permisos, de un programa, en este caso es específico ya que a la variable buf se le declara la ruta del archivo a modificar, esto será posible con ayuda del comando chmod();

Programa ejecutado

```
C:\Users\jona-\Documents\Jona\ESCOM\semestre 4\Sistemas Operativos\Practica y tarea 2\W\W>gcc permisos.cpp
permisos.cpp: In function 'int main()':
permisos.cpp:6:16: warning: unknown escape sequence: '\C'
    char buf[100]="C:\\Users\\jona-\Documents\Prueba";
                  ^
permisos.cpp:6:16: error: incomplete universal character name \U
permisos.cpp:6:16: warning: unknown escape sequence: '\j'
permisos.cpp:6:16: warning: unknown escape sequence: '\D'
permisos.cpp:6:16: warning: unknown escape sequence: '\P'
permisos.cpp:9:13: error: 'chmod' was not declared in this scope
    chmod(buf,i);
              ^

C:\Users\jona-\Documents\Jona\ESCOM\semestre 4\Sistemas Operativos\Practica y tarea 2\W\W>gcc permisos.cpp
permisos.cpp: In function 'int main()':
permisos.cpp:6:16: error: incomplete universal character name \U
    char buf[100]="C:\\Users\\jona-\Documents\Prueba";
                  ^
permisos.cpp:6:16: warning: unknown escape sequence: '\j'
permisos.cpp:6:16: warning: unknown escape sequence: '\D'
permisos.cpp:6:16: warning: unknown escape sequence: '\P'
permisos.cpp:9:13: error: 'chmod' was not declared in this scope
    chmod(buf,i);
              ^

C:\Users\jona-\Documents\Jona\ESCOM\semestre 4\Sistemas Operativos\Practica y tarea 2\W\W>
```

Programa archivos aleatorios

Crea archivos aleatorios, mediante un arreglo dará el nombre al archivo creado, como primer paso, pedirá el directorio a donde se crearan los archivos, con un ciclo asignara al arreglo la ruta, al ser aleatorio ocuparemos la función random, y este valor random lo guardaremos en una variable numero_arch, mostrara cuantos archivos se crearan, con ayuda de otro ciclo ira creando los archivos con la condición de ser menor al numero random obtenido, se concatenara el nombre del archivo con el numero de este para que no tengan el mismo nombre, con una variable entera será la bandera para ver si se pudo abrir el fichero, si esta bandera es igual a -1, marcara error y terminara su ejecución, de lo contrario escribirá en el archivo y lo cerrara.

```
int i;
char* nombres[10];

nombres[0] = calloc(100,sizeof(char));
printf("Escriba el directorio donde se crearan los archivos:\n");
scanf("%s",nombres[0]);

for(i=1; i<=10; i++) nombres[i]=nombres[0];

char* a[10]={ "1","2","3","4","5",
             "6","7","8","9","10"}; //nombre de archivo

char* cadena[15]={ "1","2","3","4","5","6",
                  "7","8","9","10","11","12","13","14","15"}; //contenido del archivo

srand(time(NULL));
int numero_arch = rand()%10;

printf("Se crearan %d archivos:\n", numero_arch);

for ( i = 0; i < numero_arch; i++)
{
    int contenido = rand()%15;

    strcat(nombres[i],a[i]);
    //printf("\n%s\n",nombres[i]);

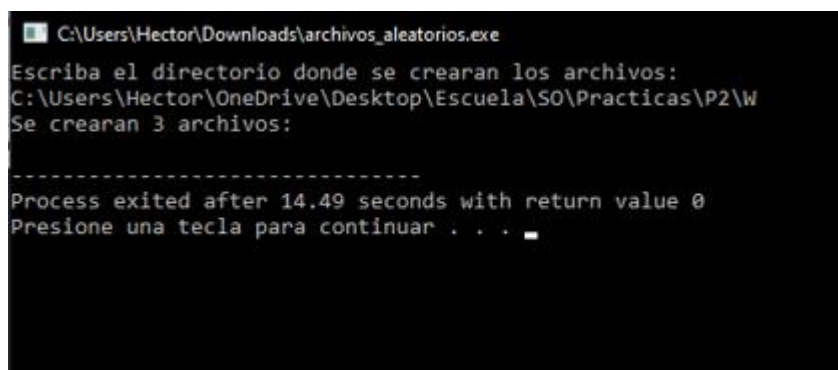
    int fichero = open (nombres[i], O_CREAT|O_WRONLY,0644);

    if (fichero==-1){

        printf("Error al abrir fichero:");
        exit(1);
    }
    write(fichero, cadena[contenido], strlen(cadena[contenido]));
    close(fichero);
}

return 0;
```

Programa ejecutado



```
C:\Users\Hector\Downloads\archivos_aleatorios.exe
Escriba el directorio donde se crearan los archivos:
C:\Users\Hector\OneDrive\Desktop\Escuela\SO\Practicas\P2\W
Se crearan 3 archivos:

-----
Process exited after 14.49 seconds with return value 0
Presione una tecla para continuar . . .
```

Linux

Se crea todos los programas en uno solo, mediante un menú.

Caso 1.-

```

case 1:
    printf("Da la direccion del directorio donde quieras que se creen los archivos \n");
    printf("Ejemplo: Documents/ \n");
    direccion = recibirDireccion();
    srand(time(NULL));
    int num = numAleatorio();

    for(int i = 0; i < num; i++){
        crearArchivos(direccion, i);
    }
    break;

```

En este caso se crearán los archivos, guardara la dirección ingresada,, creara un numero aleatorio mediante un ciclo ejecutara el código.

```

void crearArchivos(char* direccion, int iterador){
    char** cadenas = (char **){ "HOLA", "ESTO", "Es", "UN", "EJEMPLO", "DE", "LLAMADAS", "AL", "SISTEMA", "." };
    char* reset = (char*) malloc(200 * sizeof(char));
    char* numero = (char*) malloc(6 * sizeof(char));

    strcpy(reset, direccion);
    sprintf(numero, "%d.txt", iterador);
    strcat(reset, numero);

    int fd1 = open(reset, O_CREAT | O_WRONLY, 0666);

    write(fd1, cadenas[iterador], strlen(cadenas[iterador]));

    close(fd1);
}

```

crearArchivos no regresa nada, lo que hace es crear un numero aleatorio de archivos .txt con contenido, recibe la dirección donde se crean los archivos (esta dirección ya tiene que estar creada) y un iterador que ayudará a dar nombre a los archivos y colocar el contenido a los archivos.

Caso 2.-

```

case 2:
    printf("Da la direccion del archivo al cual quieras cambiar los permisos\n");
    printf("Ejemplo: Documents/0.txt \n");
    direccion = recibirDireccion();
    cambiarPermisos(direccion);
    break;

```

En este caso, cambiaremos los permisos de un archivo, dada la dirección ingresada por el usuario.

```

void cambiarPermisos (char* direccion){
    int* permisos = (int*) malloc(4 * sizeof(int));
    printf("Escribe los numeros necesarios para cambiar los permisos del archivo, por ejemplo 777: ");
    scanf("%o", permisos);
    chmod (direccion, *permisos);
}

```

Caso 3.-

Mostrara los detalles de un archivo.

```
case 3:
    printf("Da la direccion del directorio del cual quieras saber los detalles \n");
    printf("Ejemplo: Documents/ \n");
    direccion = recibirDireccion();
    mostrarDetalles(direccion);
    printf("Estos fueron los detalles \n");
    break;
```

```
void mostrarDetalles(char* direccion){
    DIR *dir;
    struct dirent *sd;
    struct stat fileinfo;

    dir = opendir(direccion);

    if(dir == NULL){
        printf("No se puede abrir el directorio");
        exit(1);
    }

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

        printf("Nombre del archivo: %s, ",sd->d_name);
        printf("Tamaño del archivo: %ld bytes, ",fileinfo.st_size);
        printf("Hora y fecha de acceso: %s.\n",ctime(&fileinfo.st_atime));

    }
    closedir(dir);
}
```

mostrarDetalles no regresa nada, recibe la dirección de la cual se quieren mostrar los detalles, nombre de archivos, tamaño y fecha, contendrá lo siguiente:

- Tipo de datos de los objetos de flujo de directorio.
- Estructura que es necesaria para acceder al nombre del archivo, viene dentro de la descripción de readdir
- Guardamos el flujo del directorio
- Si no tiene nada, salir de la función de inmediato
- Mientras el flujo sea distinto de nulo se mostrarán los detalles
- Muestra el nombre
- Muestra el tamaño
- Muestra la fecha y hora
- Cierra el flujo de archivos

Caso 4.-

Copiar archivos

```
case 4:
    printf("Da la direccion del directorio del cual estan los archivos que quieras copiar \n");
    direccion = recibirDireccion();
    copiarArchivos(direccion);
    break;
```

copiarArchivos no regresa nada, el usuario indica los archivos que quiere copiar (colocando la dirección de estos), a otra dirección (igualmente indicada por el usuario). Esta función recibe como parámetro una cadena que es la dirección donde están los archivos que se van a copiar.

NOTA: Cuando se piden los archivos se colocan todos en una fila, el programa sabrá cuantos archivos diferentes son por los espacios colocados, ejemplo: "9.txt 0.txt 5.txt 2.txt"

```
void copiarArchivos(char* direccion){
    char* nuevaDireccion = (char*)malloc(200*sizeof(char));
    char* lsArchivos = (char*)malloc(200*sizeof(char));
    char** archivos = (char**)malloc(sizeof(char*)*10);

    printf("Que archivos quieres copiar? \n");
    printf("Coloca los archivos separandolo solo por un espacio. Ejemplo: 9.txt 0.txt 5.txt 2.txt \n");
    scanf("%s", lsArchivos);

    int contadorLetra=0;
    int contadorArch=0;
    archivos[0] = (char*)malloc(6*sizeof(char));

    for(int i=0; i<strlen(lsArchivos); i++){
        archivos[contadorArch][contadorLetra] = lsArchivos[i];
        contadorLetra++;
        if(lsArchivos[i+1] == ' '){
            i++;
            contadorArch++;
            archivos[contadorArch] = (char*)malloc(6*sizeof(char));
            contadorLetra = 0;
        }
    }
}
```

Esta parte tendrá las realizará lo siguiente:

- Dirección de donde se copiarán los archivos
- Lista de archivos que se van a copiar
- una matriz que va a guardar los archivos por separado
- Este formato de especificación me inca que al leer la cadena no la corte al leer un espacio
- Guardando los caracteres que hacen parte del nombre del archivo
- Si se encuentra un espacio
- Se cambia la fila de la matriz porque se va a guardar un nuevo archivo

```
printf("Donde deseas copiar estos archivos \n");
scanf("%s", nuevaDireccion);

for(int i=0; i<contadorArch+1; i++){

    char* reset = (char*)malloc(100*sizeof(char));

    strcpy(reset, direccion);
    strcat(reset, archivos[i]);
    char* contenido = (char*)malloc(50*sizeof(char));
    int fd1 = open(reset, O_RDONLY);
    if(fd1 < 0){
        printf("El archivo no existe");
    }else{
        read(fd1, contenido, 50);
        printf("%s \n", contenido);

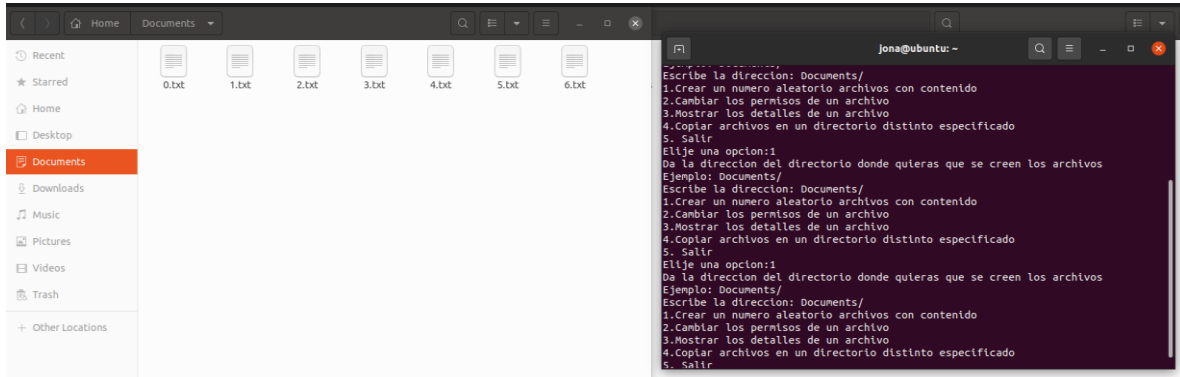
        strcpy(reset, nuevaDireccion);
        strcat(reset, archivos[i]);

        //Se copia el archivo en la nueva direccion
        int fd2 = open(reset, O_CREAT | O_WRONLY, 0666);
        write(fd2, contenido, 50);

        //Se cierran los archivos
        close(fd1);
        close(fd2);
    }
}
```

- La dirección donde se van a copiar
- La variable auxiliar
- se guarda la dirección en la variable auxiliar
- se concatena la dirección con el archivo
- Esta variable guardara el contenido que tiene un archivo
- Abrimos el archivo
- Si la función regresa un -1 significa que no encontró nada en la dirección
- Se toma el contenido que tenga el archivo abierto y se guarda
- Lo imprimimos en pantalla
- En la variable auxiliar se guarda la nueva dirección
- Se concatena la nueva dirección y el archivo que se va a copiar
- Se copia el archivo en la nueva dirección
- Se cierran los archivos

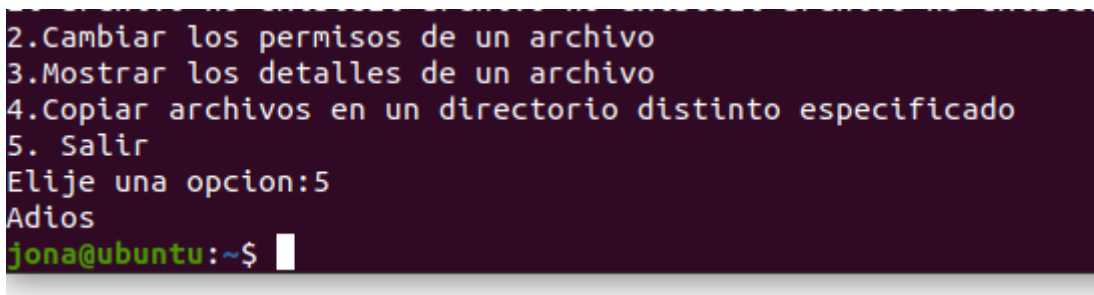
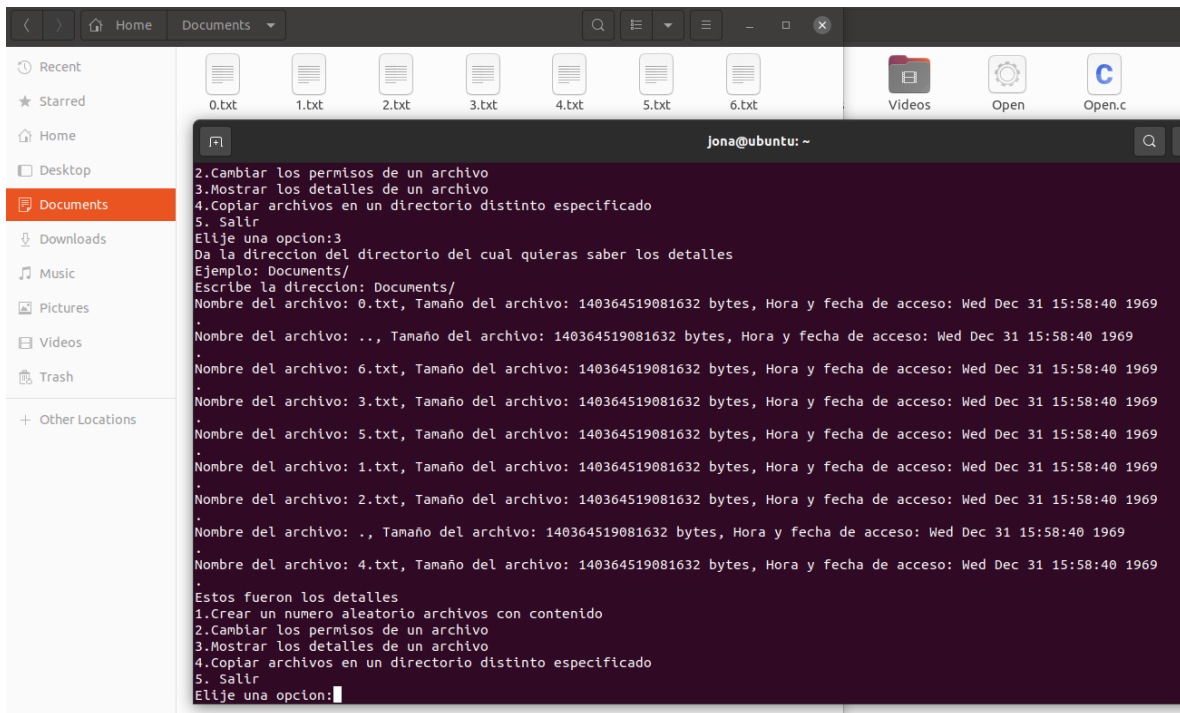
Capturas de pantalla



```
Estos fueron los detalles
1.Crear un numero aleatorio archivos con contenido
2.Cambiar los permisos de un archivo
3.Mostrar los detalles de un archivo
4.Copiar archivos en un directorio distinto especificado
5. Salir
Elije una opcion:1
Da la direccion del directorio donde quieras que se creen los archivos
Ejemplo: Documents/
Escribe la direccion: Documents/
1.Crear un numero aleatorio archivos con contenido
2.Cambiar los permisos de un archivo
3.Mostrar los detalles de un archivo
4.Copiar archivos en un directorio distinto especificado
5. Salir
Elije una opcion:1
Da la direccion del directorio donde quieras que se creen los archivos
Ejemplo: Documents/
Escribe la direccion: Documents/
1.Crear un numero aleatorio archivos con contenido
2.Cambiar los permisos de un archivo
3.Mostrar los detalles de un archivo
4.Copiar archivos en un directorio distinto especificado
5. Salir
Elije una opcion:2
Da la direccion del archivo al cual quieras cambiar los permisos
Ejemplo: Documents/0.txt
Escribe la direccion: Documents/0.txt
Escribe los numeros necesarios para cambiar los permisos del archivo, por ejemplo 777: 774
```

Comprobamos que si se cambian los permisos

```
Elije una opcion:5
Adios
jona@ubuntu:~$ cd Documents
jona@ubuntu:~/Documents$ ls
0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt
jona@ubuntu:~/Documents$ ls -l
total 28
-rwxrwxr-- 1 jona jona 4 Sep  7 20:32 0.txt
-rw-rw-r-- 1 jona jona 4 Sep  7 20:32 1.txt
-rw-rw-r-- 1 jona jona 2 Sep  7 20:32 2.txt
-rw-rw-r-- 1 jona jona 2 Sep  7 20:32 3.txt
-rw-rw-r-- 1 jona jona 7 Sep  7 20:32 4.txt
-rw-rw-r-- 1 jona jona 2 Sep  7 20:32 5.txt
-rw-rw-r-- 1 jona jona 8 Sep  7 20:32 6.txt
jona@ubuntu:~/Documents$
```

Conclusiones

La línea de comandos es una herramienta sencilla y potente que se puede aprender a utilizar en pocos minutos. Ya que es un programa que te permite comunicarte con tu sistema operativo mediante una interfaz textual, que utiliza un lenguaje de alto nivel, cercano al lenguaje natural. Con esta segunda practica reforzamos los comandos tanto de Windows como Linux, así mismo programar los comandos que a nuestro respecto son los importantes.

Algo que tenemos que remarcar en esta practica es que por fallos de la máquina virtual nos suscitaron varios problemas, desde que no reconocía un puerto externo (usb) hasta errores en el sistema operativo (se borro), pero se pudo dar solución lo antes posible.

Y para acabar cabe mencionar que el sistema operativo es de suma importancia para un equipo (computadora), ya que, sin él, una computadora no enciende.

Existe mucha variedad de sistemas operativos pero los más conocidos son el Windows 7, Unix, Linux y MacOS. Estos sistemas operativos, aunque tienen nombre diferente, tienen un mismo objetivo al ser instalado en una computadora.