# Proyecto #1

Angel Orlando Racancoj Garcia

Carnet 201631547

Septiembre 2019.

Universidad de San Carlos de Guatemala - Centro Universitario de Occidente.

División de Ciencias de la Ingeniería - Ingeniería en Ciencias y Sistemas.

Sistemas Operativos 2

# Tabla de Contenidos

# Capítulo 1

## Introducción e información general

Para el diseño de este programa fue necesario realizar una vasta investigación, realizar una gran cantidad de busqueda en internet en busca qué información que permitiera realizar cada una de las tareas que el proyecto requería.

Para el proyecto se ha planteado de manera preliminar, una diseño de diagrama de flujo para identificar el proceso que era necesario para el funcionamiento del programa, el cual ha quedado de la siguiente manera (Figura 1):

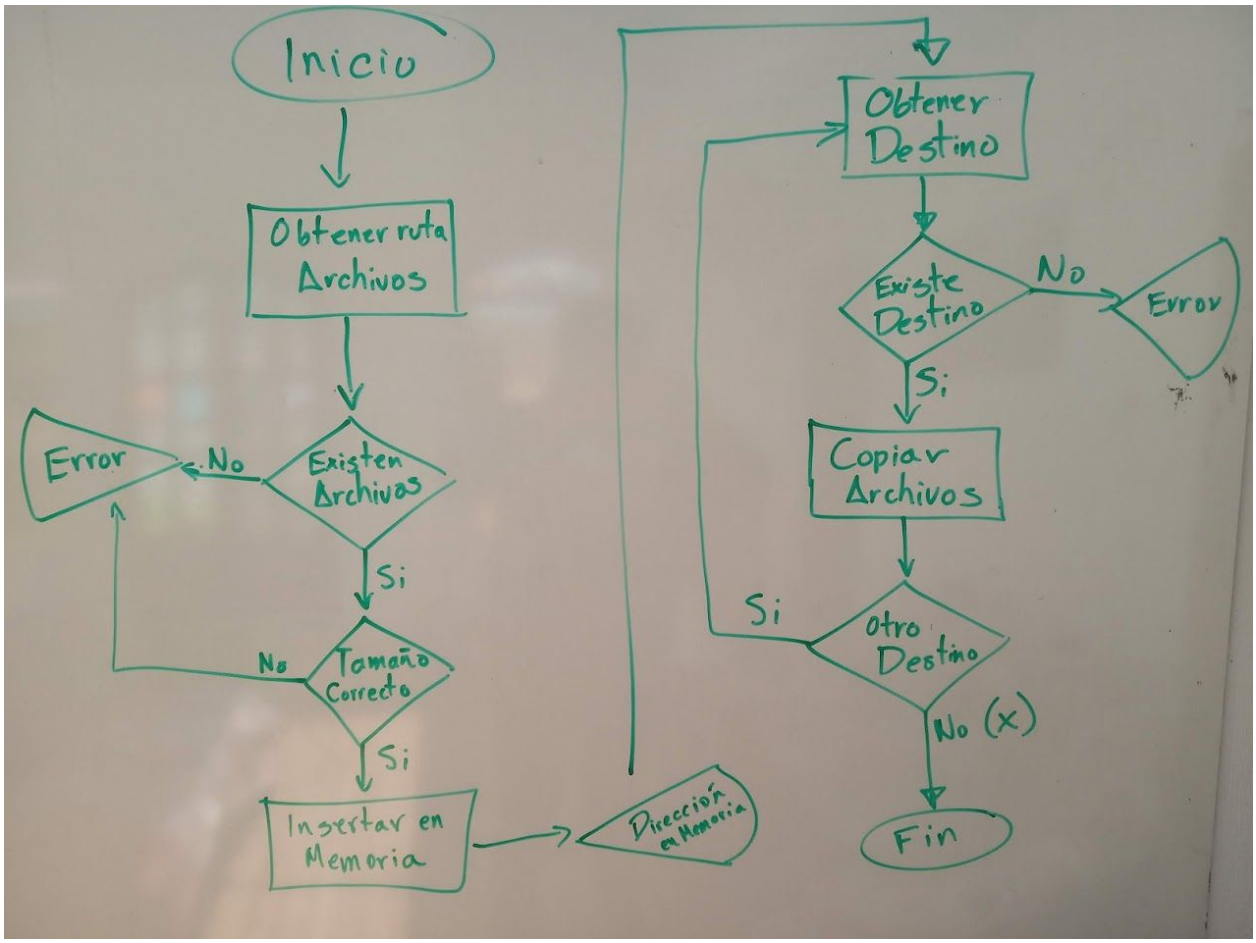



**Figura 1.**

El diagrama busca ilustra el funcionamiento de manera abstracta del programa.

Para el desarrollo del programa se han diseñado 4 clases, separando en bloques de prueba los diferentes programas, asi que a continuación se explicara el funcionamiento del codigo de estas 4 clases.

# Leer, Almacenar y Guardar

Esta parte de codigo esta dedicado a obtener la direccion en disco (Path) de un archivo (al ejecutarlo desde consola), al ingresarse como argumento el path del archivo a guardar. Luego el archivo binario es almacenado dentro de la memoria RAM  de la computadora, se pide el ingreso del Path destino y se almacena.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main (int argc, char *argv[]) {
  streampos size;
  char * memblock;
  string pathOut;
  string pathConcat;
  string inputFile(argv[1]);


  ifstream file (inputFile, ios::in|ios::binary|ios::ate);
  if (file.is_open())
  {
    size = file.tellg();
    memblock = new char [size];
    file.seekg (0, ios::beg);
    file.read (memblock, size);
    file.close();

    cout << "the entire file content is in memory"<<endl;
    cout << "Ingrese la ruta destino: ";
    cin >> pathOut;

    pathConcat = pathOut + inputFile;

    cout << "Path destino: "<< pathConcat <<endl;

    ofstream fileout(pathConcat, ios::out | ios::binary);
    fileout.write(memblock, size);
    fileout.close();

    delete[] memblock;
  }
  else cout << "Unable to open file"<<endl;
  return 0;
}
```

Se explica por bloques el codigo anteriormente expuesto.

## Apertura del archivo

```
  ifstream file (inputFile, ios::in|ios::binary|ios::ate);
  if (file.is_open())
  {
```

En esta parte se toma el path de archivo (**inputFile**) y se le indica a **ifstream** con lo siguientes atributos **ios::in|ios::binary|ios::ate** que se está aperturando el archivo y que es un archivo binario. Por último dentro del if con **file.is_open()** se verifica que el archivo esté abierto para trabajarlo.

## Almacenar en Memoria

```
size = file.tellg();
    memblock = new char [size];
    file.seekg (0, ios::beg);
    file.read (memblock, size);
    file.close();
```

En esta parte obtiene el tamaño del archivo con: **memblock = new char [size];,** con las siguientes dos líneas de instrucciones se inserta en memoria el archivo. Por último debe cerrarse el archivo, ya que no nos es útil, ya que está almacenado en memoria.

## Obtener Path destino

```
    cout << "the entire file content is in memory"<<endl;
    cout << "Ingrese la ruta destino: ";
    cin >> pathOut;

    pathConcat = pathOut + inputFile;

    cout << "Path destino: "<< pathConcat <<endl;
```

Con esta parte se pide el path destino con la instrucción **cin >> pathOut;** y se une el path ingresado con el nombre del archivo que ha sido seleccionado, para crear la nueva direccion en memoria para nuestro archivo.

```
    ofstream fileout(pathConcat, ios::out | ios::binary);
    fileout.write(memblock, size);
    fileout.close();

    delete[] memblock;
```

Con **ofstream fileout(pathConcat, ios::out | ios::binary);** se inicializa el nuevo archivo a crear y se escribe el su nueva direccion con **fileout.write(memblock, size);,** se cierra el archivo creado, y por último con la instrucción **delete[] memblock;** se elimina de la memoria RAM el archivo almacenado.

# Manejo de Directorios

La intención de esta parte del codigo, es permitir abrir un directorio o carpeta de archivo, y obtener el nombre de cada uno de los archivo que encontramos dentro de la carpeta, esta parte no será útil al momento de querer copiar los archivo que encontramos dentro de un directorio, sin necesitar indicar cada uno de los archivos contenidos.

También tenemos una parte del codigo, que tiene como función la creacion de directorios, esto para al momento de copiar un directorio, previo a copiar el archivo debe existir ya el nuevo directorio destino.

```
#include <dirent.h>
#include <cstring>
#include <iostream>
#include <vector>
#include <memory>
#include <sys/stat.h>
#include <sys/types.h>
using namespace std;

vector<string> GetDirectoryFiles(string dir) {
  vector<string> files;
  shared_ptr<DIR> directory_ptr(opendir(dir.c_str()), [](DIR* dir){ dir &&
closedir(dir); });
  struct dirent *dirent_ptr;
  if (!directory_ptr) {
    cout << "Error opening : " << strerror(errno) << dir << endl;
    return files;
  }

  while ((dirent_ptr = readdir(directory_ptr.get())) != nullptr) {
    files.push_back(string(dirent_ptr->d_name));
  }
  return files;
}

int main() {
  string directory_path ="Screenshots/";
  vector<string> files = GetDirectoryFiles(directory_path);

  for (int i = 0; i < files.size(); ++i){
    cout<<"File: "<<files[i]<<endl;
  }

  string directoryPathAux = "holaMundo";
  if (mkdir(directoryPathAux.c_str(),0777) == 0){
    cout<<"Directorio creado"<<endl;
```

```
  }

  return 0;
}
```

## Abrir directorio y verificar

```
shared_ptr<DIR> directory_ptr(opendir(dir.c_str()), [](DIR* dir){ dir &&
closedir(dir); });
  struct dirent *dirent_ptr;
  if (!directory_ptr) {
    cout << "Error opening : " << strerror(errno) << dir << endl;
    return files;
  }
```

Con la instruccion **shared_ptr<DIR> directory_ptr(opendir(dir.c_str()), [](DIR* dir){ dir && closedir(dir); });,** y luego se verifica que el directorio se abriera de manera adecuada con **!directory_ptr**, de no existir el directorio o tener algún problema nos lanzará un mensaje de error y devolverá un listado vacío de resultados.

## Imprimir resultados

```
for (int i = 0; i < files.size(); ++i){
    cout<<"File: "<<files[i]<<endl;
  }
```

Con un ciclo **For** se leer el resultado de la lectura de carpeta de archivos del directorio indicado

## Crear Directorio

```
string directoryPathAux = "holaMundo";
  if (mkdir(directoryPathAux.c_str(),0777) == 0){
    cout<<"Directorio creado"<<endl;
  }
```

El comando **mkdir** es el encargado de crear el directorio, y de poder crear el directorio nos lanzará un mensaje indicando que el directorio fue creado exitosamente.

# Tamaño de un archivo

Con este método se obtiene el tamaño de un archivo en bytes, es bastante simple, y utiliza un función anteriormente utilizada para la lectura de un archivo

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

long getFile(string inputFile){
    long size;

    ifstream file (inputFile, ios::in|ios::binary|ios::ate);
    if (file.is_open()) {
        size = file.tellg();
        file.close();
    }
    return size;
}

int main(){
    long ans = getFile("th.jpeg");
    if (ans > 1)
       cout<<"size of file is "<<(ans)<< "bytes \n"<< endl;
    return 0;
}
```

## Obtener tamaño del archivo

```
ifstream file (inputFile, ios::in|ios::binary|ios::ate);
    if (file.is_open()) {
        size = file.tellg();
        file.close();
    }
    return size;
```

Mediante el uso de **ifstream** se abre el archivo, y con **file.tellg();** se obtiene el tamaño del archivo en bytes, se cierra el archivo y por último se devuelve el peso.

# Obtener información de la memoria RAM

```
#include <unistd.h>
#include <iostream>
 #include <sys/sysinfo.h>
using namespace std;

struct sysinfo info;

int main(){
    if (sysinfo(&info) != 0){
        return false;
    }

    long result = info.freeram / 1048576;
    long ram = info.totalram / 1048576;
    cout << "Free ram: "<< (ram - result) <<"Mb, Total ram: "<< ram <<"Mb"<< endl;
    return 0;
}
```

En esta parte no es necesario caer en muchos detalles, ya que básicamente nos devuelve el espacio que tenemos en la RAM.

Con **info.freeram** obtenemos la RAM ocupada en bytes.

Con **info.totalram** obtenemos el tamaño total de la RAM en bytes.

# Capítulo 2

## Codigo fuente para el Proyecto #1

Cabe destacar que este proyecto parte de los ejemplos vistos en el capítulo anterior, cabe resaltar que aquí se agregaran métodos nuevos muchos de ellos no son mencionados ya que su nivel de complejidad es bastante bajo y un programador conocimientos básicos puede entenderlo, varios métodos están basados en archivos anteriores dándole un nivel de complejidad acorde a lo que se requiere para complir con las diferentes funciones diseñadas.

De siguiente enunciado, no se ha podido cumplir con una petición, que es mostrar las direcciones en memoria donde se han almacenado los archivos, pero el resto ha sido cubierto de manera adecuada.

*"Deberá desarrollar un programa que permita copiar el contenido de uno o varios archivos (también puede ser un directorio y su contenido interno) a la memoria de la computadora, una vez copiados los datos en memoria se deberá solicitar el medio de almacenamiento (que puede ser un medio extraible), para realizar la copia de estos datos (similar a un copy), la diferencia es que al terminar de copiar los datos puede repetirse la operación de copiado tantas veces como se requiera en varios medios (insertar un pendrive origen y luego intercambiarlo por varios pendrive en el mismo puerto). Si no se desea hacer una copia más, al salir de la aplicación ésta deberá liberar la memoria utilizada. En caso de copiar un archivo que exceda la capacidad de la memoria utilizada se deberá indicar que no es posible hacer la copia, la utilización de la memoria se deberá corroborar y reflejar con la utilización de las herramientas usadas en el laboratorio, tanto si el programa utiliza poca memoria por copiar un archivo pequeño o más en el caso de un archivo/s muy grande/s."*

### Codigo:

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include<vector>
#include <dirent.h>
#include <cstring>
#include <memory>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/sysinfo.h>

using namespace std;

struct sysinfo info;

struct fileMemory{
	string filePath;
```

```
	streampos sizeF;
	char * memblockF;
};

string delimiterFile = ".archivo";
string delimiterDir = ".directorio";

vector <string> getFilesPath(string textoEntrada);
string getPath(string textoEntrada, string delimiter);
string getFileName(string textoEntrada);
string getDirectoryName(string textoEntrada);
fileMemory getFile(string inputFile);
void writeFile(fileMemory fOut, string destinyPath);
bool existFile(string filePath);
bool existDirectory(string filePath);
vector<string> GetDirectoryFiles(string dir);
long getFileSize(string inputFile);
long getFilesSize(vector <string> paths);
long freeRam();

int main (int argc, char *argv[]) {
	string pathOut = "/home/";
	string pathDirectory;
	string inputFile(argv[1]);
	vector <fileMemory> fileIn;
	vector <fileMemory> directoryFiles;
	vector <string> paths = getFilesPath(inputFile);;
	vector <string> directoryPaths;

	cout<<"List size: "<< paths.size() <<endl;

	if (freeRam() > getFilesSize(paths)){
		for (int i = 0; i < paths.size(); ++i){

			if (paths[i].find(delimiterFile) != string::npos) {
				string pathAux = getPath(paths[i],delimiterFile);
				if (existFile(pathAux)){
					fileIn.push_back(getFile(pathAux));
				} else {
					cout << "No existe el archivo: "<<pathAux<<", este
archivo no se copiara"<<endl;
				}

			} else if (paths[i].find(delimiterDir) != string::npos) {
				string pathAux = getPath(paths[i],delimiterDir);

				if (existDirectory(pathAux)){
```

```
                                    pathDirectory = getDirectoryName(pathAux);
                                    directoryPaths = GetDirectoryFiles(pathAux);
                                    for (int j = 0; j < directoryPaths.size(); ++j){
                                          cout<<"Path File:
"<<directoryPaths[j]<<endl;
                                          directoryFiles.push_back(getFile((pathAux +
directoryPaths[j])));
                                    }
                              } else {
                                    cout << "No existe el directorio: "<<pathAux<<",
no se copiara"<<endl;
                              }
                        } else {
                              cout<<"Error en: "<<paths[i]<<endl;
                              return 0;
                        }
                  }
                  cout << "Carga en memoria completada ;)\n\n";
                  //ask for final destiny
                  while((pathOut.compare("x") != 0) || (pathOut.compare("X") != 0)){
                        cout << "Ingrese la ruta destino(\"X\" para salir): ";
                      cin >> pathOut;
                      string directoryPathAux = pathOut+pathDirectory;
                      if (existDirectory(pathOut)){
                         for (int i = 0; i < fileIn.size(); ++i){
                               writeFile(fileIn[i],pathOut);
                            }
                            if (directoryFiles.size() > 0){
                               if (mkdir(directoryPathAux.c_str(),0777) == 0){
                                     for (int i = 0; i < directoryFiles.size(); ++i){

writeFile(directoryFiles[i],(directoryPathAux+"/"));
                                    }
                               } else {
                                     cout<<"Ya existe la carpeta: "<<
(pathOut+pathDirectory)<<endl;
                               }
                            }
                            cout << "Copia completada :)"<<endl;
                      } else {
                            if ((pathOut.compare("x") == 0) || (pathOut.compare("X") ==
0)){
                               cout << "Tarea completada :)"<<endl;
                               return 0;
                            } else {
                               cout<<"La direccion destino: "<< pathOut<<" no
existe"<<endl;
```

```
                        }
                    }

                }
        } else {
                cout << "Memoria insuficiente, intente de nuevo :'(" << endl;
        }
        return 0;
}

vector <string> getFilesPath(string textoEntrada){
        vector <string> paths;

        string delimiter=",";

        size_t pos = 0;
        string token;
        while ((pos = textoEntrada.find(delimiter)) != string::npos) {
            token = textoEntrada.substr(0, pos);
            paths.push_back(token);

            textoEntrada.erase(0, pos + delimiter.length());
        }
        paths.push_back(textoEntrada);

        return paths;
}

string getPath(string textoEntrada, string delimiter){

        string pathFile;
        size_t pos = 0;
        string token;
        while ((pos = textoEntrada.find(delimiter)) != string::npos) {
            token = textoEntrada.substr(0, pos);
            pathFile = token;

            textoEntrada.erase(0, pos + delimiter.length());
        }
        cout << "Path origin: "<<pathFile << endl;
        return pathFile;
}

string getFileName(string textoEntrada){

        string delimiter="/";
```

```
        size_t pos = 0;
        string token;
        while ((pos = textoEntrada.find(delimiter)) != string::npos) {
            token = textoEntrada.substr(0, pos);
            textoEntrada.erase(0, pos + delimiter.length());
        }
        return textoEntrada;
}

string getDirectoryName(string textoEntrada){

        string delimiter="/";

        size_t pos = 0;
        string token;
        while ((pos = textoEntrada.find(delimiter)) != string::npos) {
            token = textoEntrada.substr(0, pos);
            textoEntrada.erase(0, pos + delimiter.length());
        }
        return (token);
}

fileMemory getFile(string inputFile){
        streampos size;
        char * memblock;
    fileMemory fileInMemory;

        ifstream file (inputFile, ios::in|ios::binary|ios::ate);
        if (file.is_open()) {
            size = file.tellg();
            memblock = new char [size];
            file.seekg (0, ios::beg);
            file.read (memblock, size);
            file.close();
            fileInMemory.filePath = getFileName(inputFile);
            fileInMemory.sizeF = size;
            fileInMemory.memblockF = memblock;
        }
        return fileInMemory;
}

void writeFile(fileMemory fOut, string destinyPath){
        string pathOut = destinyPath + fOut.filePath;
        cout << "Path destino: "<< pathOut <<endl;

    ofstream fileout(pathOut, ios::out | ios::binary);
    fileout.write(fOut.memblockF, fOut.sizeF);
```

```
    fileout.close();

    delete[] fOut.memblockF;
}

bool existFile(string filePath){
	ifstream file (filePath, ios::in|ios::binary|ios::ate);
	if (file.is_open()){
		file.close();
		return true;
	} else {
		return false;
	}
}

bool existDirectory(string filePath){
	vector<string> files = GetDirectoryFiles(filePath);
	return (files.size() > 0);
}

vector<string> GetDirectoryFiles(string dir) {
	vector<string> files;
	shared_ptr<DIR> directory_ptr(opendir(dir.c_str()), [](DIR* dir){ dir &&
closedir(dir); });
	struct dirent *dirent_ptr;
	if (!directory_ptr) {
		cout << "Error opening : " << strerror(errno) << dir << endl;
		return files;
	}

	while ((dirent_ptr = readdir(directory_ptr.get())) != nullptr) {
		string nameAux = string(dirent_ptr->d_name);
		if ((nameAux.compare(".") != 0) && (nameAux.compare("..") != 0)){
			files.push_back(nameAux);
		}
	}
	return files;
}

//Return the size en bytes
long getFileSize(string inputFile){
    long size;

    ifstream file (inputFile, ios::in|ios::binary|ios::ate);
    if (file.is_open()) {
        size = file.tellg();
        file.close();
```

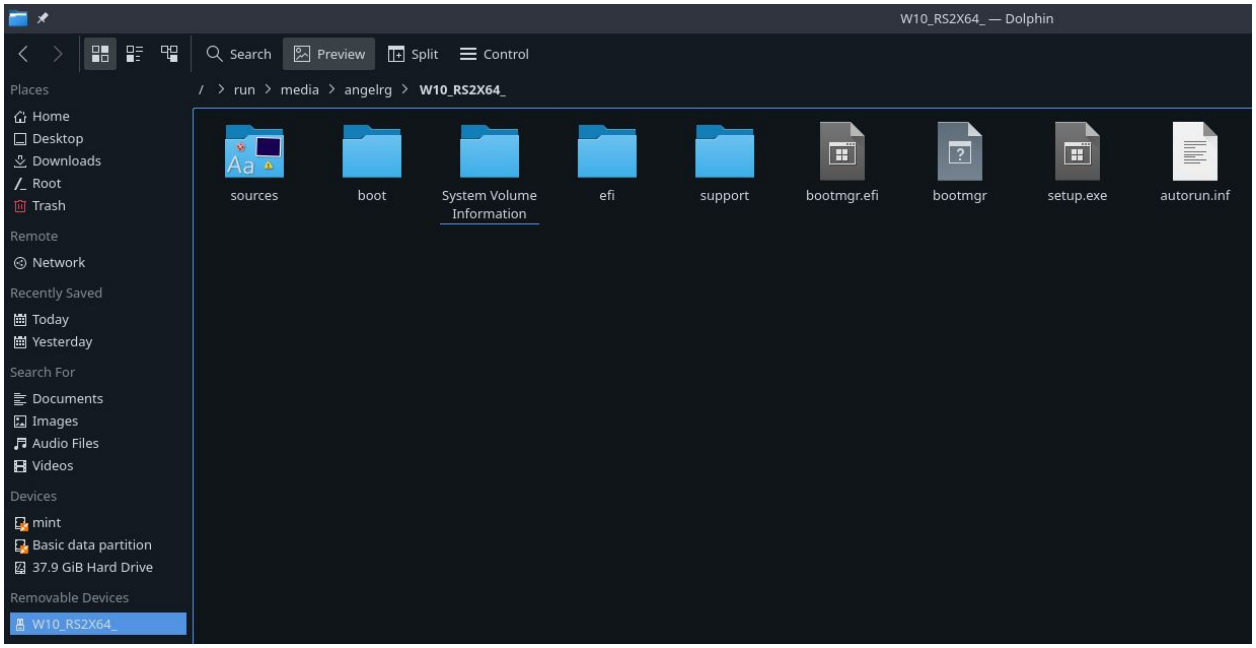
```
    }
    return size;
}

long getFilesSize(vector <string> paths){
       long filesSize = 0;
       vector <string> directoryPaths;
       string pathDirectory;
       for (int i = 0; i < paths.size(); ++i){

              if (paths[i].find(delimiterFile) != string::npos) {
                     string pathAux = getPath(paths[i],delimiterFile);
                     if (existFile(pathAux)){
                            filesSize += getFileSize(pathAux);
                     } else {
                            cout << "No existe el archivo: "<<pathAux<<", no se ha
calculado su peso"<<endl;
                     }

              } else if (paths[i].find(delimiterDir) != string::npos) {
                     string pathAux = getPath(paths[i],delimiterDir);

                     if (existDirectory(pathAux)){
                            pathDirectory = getDirectoryName(pathAux);
                            directoryPaths = GetDirectoryFiles(pathAux);
                            for (int j = 0; j < directoryPaths.size(); ++j){
                                   filesSize += getFileSize(pathAux +
directoryPaths[j]);
                            }
                     } else {
                            cout << "No existe el directorio: "<<pathAux<<", no se ha
calculado su peso"<<endl;
                     }
              } else {
                     cout<<"Error en: "<<paths[i]<<endl;
                     return 0;
              }
       }
       cout << "Total files Size: "<< filesSize << " bytes"<< endl;
       return filesSize;
}

long freeRam(){
       if (sysinfo(&info) != 0){
         return false;
    }
    long result =info.totalram - info.freeram;
```

```
    cout << "Free ram: "<< result <<" bytes"<< endl;
    return result;
}
```

# Como utilizarlo:

Insertar la memoria

La memoria destino no contiene los archivos que agregaremos




## Compilacion:

Debe compilarse de la siguiente manera:

```
[angelrg@angelrg-asus Copy-Paste-sopes2]$ c++ readTextIn.cpp -o copyos2
```

## Uso:

### Ejecución:

```
user$ ./copyos2
/home/angelrg/debian-live-10.0.0-amd64-standard.iso.archivo,/home/angelrg/payara-5.192.zip.archivo,th.jpeg.archivo,/home/angelrg/Pictures/Screenshots/.directorio
```

Para agregar un archivo es: **/home/angelrg/debian-live-10.0.0-amd64-standard.iso.archivo**

Para agregar un directorio es: **/home/angelrg/Pictures/Screenshots/.directorio**

## Luego de dar Enter:

```
[angelrg@angelrg-asus Copy-Paste-sopes2]$ ./copyos2 /home/angelrg/d
List size: 4
Free ram: 5901246464 bytes
Path origin: /home/angelrg/debian-live-10.0.0-amd64-standard.iso
Path origin: /home/angelrg/payara-5.192.zip
Path origin: th.jpeg
Path origin: /home/angelrg/Pictures/Fondos/
Total files Size: 1017188279 bytes
Path origin: /home/angelrg/debian-live-10.0.0-amd64-standard.iso
Path origin: /home/angelrg/payara-5.192.zip
Path origin: th.jpeg
Path origin: /home/angelrg/Pictures/Fondos/
Path File: bing1.jpg
Path File: 3ae6c7d73671de4f89a21be5a41d0093.jpg
Path File: H6dclqX.jpg
Path File: 6c16779abd231579358276d72965540a.jpg
Path File: YNPFirefall_ROW1422035150_1920x1080.jpg
Path File: th.jpg
Path File: WinterGrand_ROW7075820680_1920x1080.jpg
Carga en memoria completada ;)

Ingrese la ruta destino("X" para salir):
```

## Luego ingresamos el path destino

El programa pedirá ingresar una ruta destino con: **Ingresar la ruta destino(“X” para salir):**

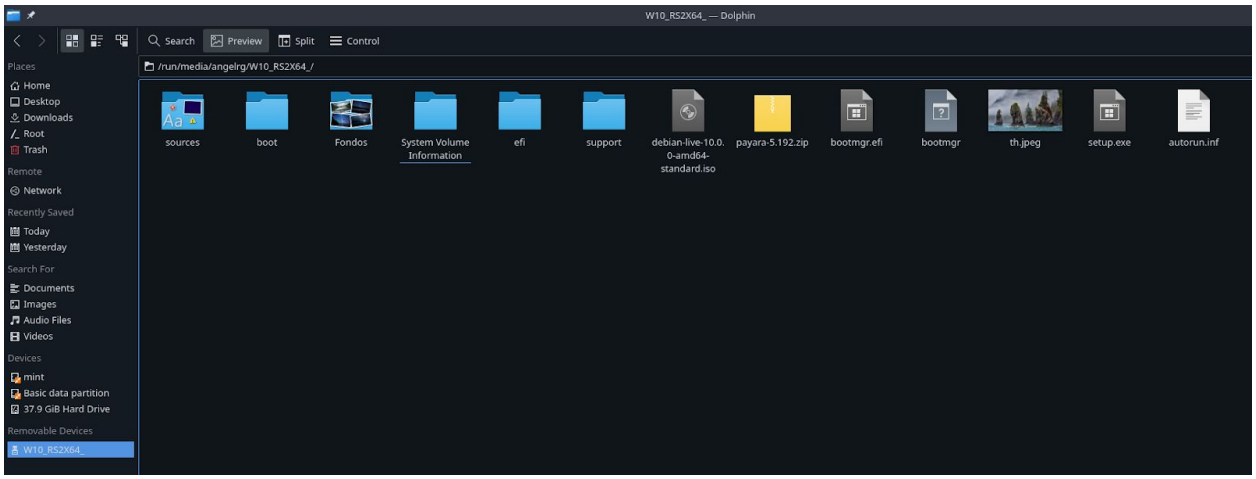
**Nota:** Si se escribe **X** o **x** se cerrará el programa.

Dependiendo de la velocidad de escritura destino y el tamaño de los archivos le llevará su tiempo al programa copiar los archivos.

Resultado:

```
[angelrg@angelrg-asus Copy-Paste-sopes2]$ ./copyos2 /home/angelrg/debian-live-10.0.0-amd64-s
List size: 4
Free ram: 5901246464 bytes
Path origin: /home/angelrg/debian-live-10.0.0-amd64-standard.iso
Path origin: /home/angelrg/payara-5.192.zip
Path origin: th.jpeg
Path origin: /home/angelrg/Pictures/Fondos/
Total files Size: 1017188279 bytes
Path origin: /home/angelrg/debian-live-10.0.0-amd64-standard.iso
Path origin: /home/angelrg/payara-5.192.zip
Path origin: th.jpeg
Path origin: /home/angelrg/Pictures/Fondos/
Path File: bing1.jpg
Path File: 3ae6c7d73671de4f89a21be5a41d0093.jpg
Path File: H6dclqX.jpg
Path File: 6c16779abd231579358276d72965540a.jpg
Path File: YNPFirefall_ROW1422035150_1920x1080.jpg
Path File: th.jpg
Path File: WinterGrand_ROW7075820680_1920x1080.jpg
Carga en memoria completada ;)

Ingrese la ruta destino("X" para salir): /run/media/angelrg/W10_RS2X64_/
Path destino: /run/media/angelrg/W10_RS2X64_/debian-live-10.0.0-amd64-standard.iso
Path destino: /run/media/angelrg/W10_RS2X64_/payara-5.192.zip
Path destino: /run/media/angelrg/W10_RS2X64_/th.jpeg
Path destino: /run/media/angelrg/W10_RS2X64_/Fondos/bing1.jpg
Path destino: /run/media/angelrg/W10_RS2X64_/Fondos/3ae6c7d73671de4f89a21be5a41d0093.jpg
Path destino: /run/media/angelrg/W10_RS2X64_/Fondos/H6dclqX.jpg
Path destino: /run/media/angelrg/W10_RS2X64_/Fondos/6c16779abd231579358276d72965540a.jpg
Path destino: /run/media/angelrg/W10_RS2X64_/Fondos/YNPFirefall_ROW1422035150_1920x1080.jpg
Path destino: /run/media/angelrg/W10_RS2X64_/Fondos/th.jpg
Path destino: /run/media/angelrg/W10_RS2X64_/Fondos/WinterGrand_ROW7075820680_1920x1080.jpg
Copia completada :)
Ingrese la ruta destino("X" para salir): x
Error opening : No such file or directoryx
Tarea completada :)
[angelrg@angelrg-asus Copy-Paste-sopes2]$
```

## Salida grafica:




Podemos observar que se han agregado los archivos indicados en el Path, incluida la carpeta **“Fondos”** con imagenes.

# Lista de referencias


(n.d.). Retrieved from http://man7.org/linux/man-pages/man2/sysinfo.2.html

C Program to list all files in the Directory on Windows / Linux. (2018, February 11). Retrieved from http://www.codebind.com/cpp-tutorial/cpp-program-list-files-directory-windows-linux/

Input/output with files. (n.d.). Retrieved from http://www.cplusplus.com/doc/tutorial/files/

Input/output with files. (n.d.). Retrieved from http://www.cplusplus.com/doc/tutorial/files/

Lizard, B. T. (1959, April 01). How do you determine the amount of Linux system RAM in C ? Retrieved from https://stackoverflow.com/questions/349889/how-do-you-determine-the-amount-of-linux-system-ram-in-c

Related:. (n.d.). Retrieved from https://databasefaq.com/index.php/answer/39431/c-11-ram-file-format-recognizing-file-formats-from-binary-c-

TheCrazyProgrammerTheCrazyProgrammer 2, Vincenzo PiiVincenzo Pii 11.5k66 gold badges3131 silver badges4747 bronze badges, Moswaldmoswald 8, SviatoslavSviatoslav 40144 silver badges33 bronze badges, Arafat HasanArafat Hasan 48044 silver badges1919 bronze badges, Ryanbworkryanbwork 1, . . . YileiYilei 15511 silver badge44 bronze badges. (1963, May 01). Parse (split) a string in C using string delimiter (standard C ). Retrieved from https://stackoverflow.com/questions/14265581/parse-split-a-string-in-c-using-string-delimiter-standard-c#14266139

VincentVincent 22.1k4242 gold badges145145 silver badges302302 bronze badges, PherricOxidePherricOxide 11.4k11 gold badge2020 silver badges3939 bronze badges, Harrynghharryngh 1, Jim BalterJim Balter 13.8k11 gold badge3131 silver badges5757 bronze badges, Anhoppeanhoppe 2, Viktor LiehrViktor Liehr 35322 silver badges1313 bronze badges, . . . Miksiiimiksiii 2. (1963, February 01). Fastest way to check if a file exist using standard C /C 11/C? Retrieved from https://stackoverflow.com/questions/12774207/fastest-way-to-check-if-a-file-exist-using-standard-c-c11-c

Std::string::find. (n.d.). Retrieved from http://www.cplusplus.com/reference/string/string/find/

Std::vector::data. (n.d.). Retrieved from http://www.cplusplus.com/reference/vector/vector/data/

User1543915user1543915 43511 gold badge44 silver badges1515 bronze badges, Hmjdhmjd 105k1313 gold badges170170 silver badges225225 bronze badges, SamanSaman 50944 silver badges1111 bronze badges, Alkalk 61.1k88 gold badges7171 silver badges184184 bronze badges, Unwindunwind 332k5454 gold badges410410 silver badges544544 bronze badges, TheFriendlyDragonTheFriendlyDragon 6122 bronze badges, . . . Ferbuntuferbuntu 3055 bronze badges. (1963, February 01). How can I check if a directory exists? Retrieved from https://stackoverflow.com/questions/12510874/how-can-i-check-if-a-directory-exists

Vector of strings - C Forum. (n.d.). Retrieved from http://www.cplusplus.com/forum/beginner/14792/