

République du Cameroun

Paix – Travail – Patrie

------------------------------------------------------------

Université de Yaoundé I

Sapienta – Collativia – Cognitio

------------------------------------------------------------

Ecole Nationale Supérieure Polytechnique

------------------------------------------------------------

Département du Génie informatique

------------------------------------------------------------

Republic of Cameroon

Peace – Work – Fatherland

------------------------------------------------------------

University of Yaoundé I

Sapienta – Collativia – Cognitio

------------------------------------------------------------

National Advanced School of Engineering

------------------------------------------------------------

Department of computer Engineering

------------------------------------------------------------

  -

**Réalisée par:**

**ENYEGUE OTTOU Junnie Love Doria**

**Matricule 19p177**

Étudiant au Niveau 3 de l’ENSPI, génie informatique,

Sous la Direction de :

Pr. DJOTIO Thomas

Année académique 2021-2022

**UE : SYSTEME D’EXPLOITATION**

**PROJET** : **REALISATION D’UN GESTIONNAIRE DE FICHIER**

Table of Contents

[*A.* *INTRODUCTION* 3](#_Toc93414197)

[*I.* *CHOIX DU LANGAGE* 3](#_Toc93414198)

[*II.* *PRESENTATION DE My JLD~MANAGER* 4](#_Toc93414199)

[*B.* *IMPLEMENTATION* 11](#_Toc93414200)

[*C.* *CONCLUSION* 24](#_Toc93414201)

[*D.* *BIBLIOGRAPHIE* 24](#_Toc93414202)

# 

# *INTRODUCTION*

Durant ce premier semestre , il nous a été demande de mettre sur pied un gestionnaire de fichier dans un langage de programmation de notre choix. Ainsi donc , l’établissement du plan de travail nous envoie d’abord a l’analyse du sujet étant : Qu’est-ce qu’un gestionnaire de fichier ? Un gestionnaire de fichier est premièrement une des fonctionnalités du système d’exploitation se définissant comme un logiciel travaillant le plus souvent dans une interface graphique en permettant l’ouverture , la visualisation,l’impression,le renommage, le déplacement ,la copie, la suppression, l’affichage et la modification de propretés et recherche de fichiers. Ayant ainsi cette définition, nous avons mis sur pied un gestionnaire de fichier copie sur le modèle de Windows pouvant remplir les fonctionnalités d’un gestionnaire de fichier lambda. Nous vous présentons donc notre « My JLD~MANAGER »

## *CHOIX DU LANGAGE*

Pour la réalisation de notre projet , nous avons choisit comme langage de programmation le python plus précisément python version 3.8 64bits. Ce choix n’est pas anodin en effet, le python est un langage programmation que l’on peut qualifier de facile a apprendre de ce fait, n’ayant pas eu de bonnes bases d’apprentissage du langage C durant notre passage au MSP et vue le peu de temps que nous disposions, il nous fallait un langage dont l’apprentissage est facile à acquérir . Mais encore , le python est aussi «  populaire » c’est-à-dire qu’il dispose d’une pléthore de documentation abordant le sujet ainsi qu’une immense communauté . Tous ces critères réunis justifie notre choix du python pour la réalisation de My JLD~MANAGER

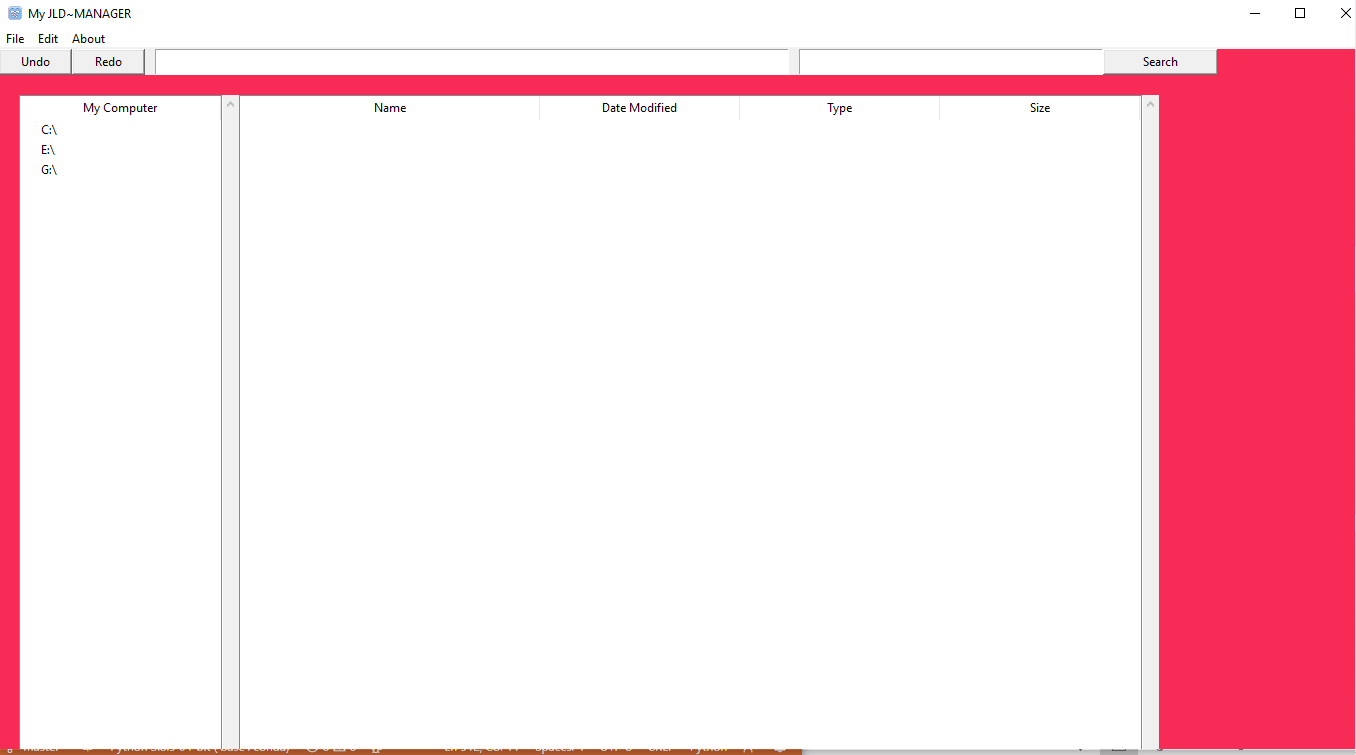
## *PRESENTATION DE My JLD~MANAGER*

1. *Présentation de l’interface*

L’interface de notre projet se présente comme suit :

Elle comprend :

* Une barre d’accès : Elle représente le chemin d’accès de l’élément sélectionne
* Une barre de recherche pour la recherche d’un élément
* Un « TreeView » : ici on représente les différents répertoires que nous avions
* Un « ListView » : ici on représente les fichiers et dossiers avec tous les informations sur ces derniers c’est-a-dire le nom de l’élément, sa taille, la nature de son extension, et sa date de modification
* La barre de menu ayant les fonctionnalités File, Edit, Undo et Redo



1. *Presentation des fonctions implementees*

Notre projet My JLD~MANAGER est un gestionnaire de fichier copier sur le modèle de Windows , il comprend les fonctionnalités suivantes :

* Copy : Pour la copie des fichiers, nous avons la fonction Copy se trouvant dans la barre de menu « Edit ». Cette fonction procède comme suit :
* On récupère le chemin du fichier ou dossier a copier que l’on insère dans un tableau , tabcopy s’il s’agit d’un fichier et tabcopydir s’il s’agit d’un dossier, ce chemin sera recuperer plus tard pour coller l’élément choisit vers sa destination finale
* Cut : pour couper les fichiers , nous avons la fonction Cut , ici on copie toujours les fichiers mais en les supprimant dans le répertoire d’origine ceci se fait en récupérant le chemin de l’élément sélectione que nous insérons dans le tableau tabcut s’il s’agit d’un fichier, tabcutdir s’il s’agit d’un dossier ; cet élément sera utilise plus tard par la fonction Paste pour coller l’élément sélectionné dans sa destination finale

* Paste : il s’agit de la fonction coller, elle récupère le chemin d’accès se trouvant dans un tableau , puis effectue une copie si Copy a été sélectionné ou colle si c’était Cut. En récupérant ce chemin,on utilise les méthodes

copy\_tree pour les dossiers et copy2 pour les fichiers

* Compress : Pour  la compression des fichiers nous avons la fonction Compress utilisant la bibliothèque gzip
* Decompress : Afin de décompresser les fichiers a extension .zip,

nous avons utilise la méthode shutil.unpack\_archive de la

bibliotheque shutlib

* Delete : On récupère le chemin du fichier a supprimer, on teste la nature de l’élément c’est-a-dire s’il s’agit d’un fichier , on utilise la méthode os.remove, s’il s’agit d’un dossier , on effectue une suppression récursive grâce a la methode os.rmtree
* Rename : Pour renommer un fichier ou un dossier, nous récupérons le chemin du fichier sélectionné , demande le nouveau nom a attribuer, et , grâce à la fonction os.rename, on effectue l’echange
* New folder : On crée un nouveau dossier grâce à la fonction os.mkdir dans le dossier sélectionné
* Undo : pour retourner sur la page précédente, le problème est que cela n’est possible qu’une seule fois c’est-a-dire pour le passage d’un répertoire vers un autre, nous n’avions pas réussi a le faire pour plusieurs sous-répertoires
* Redo : pour aller vers la page suivante, nous avons la fonction redo mais elle a le même problème que la fonction undo
* Sort\_by : Il s’agit du tri par ordre alphabetique,on utilise la mathode reversed() sur le chemin selectionne et en retour on a une liste des fichiers trie par ordre croissante ou decroissante de l’alphabet

1. *Difficultés rencontrées*

Les différents problèmes rencontres lors de la réalisation de ce projet sont multiples :

Comme difficultés rencontrées ,nous pouvons citer:

* l’apprentissage d’un nouveau langage
* la difficulté d’inclure les notions apprises en cours dans notre projet
* La recherche des bibliothèques propres au système Windows
* La non implémentation de certaines fonctions comme la fonctions « Open With » permettant d’ouvrir avec d’autres logiciels que celui reconnu par défaut par le système
* La sélection d’un élément sans pour autant l’ouvrir : Nous avons essayé d’implémenter cette fonctionnalité mais en vain , c’est le pourquoi nous avons fait un menu contextuel afin de pouvoir effectuer une opération quelconque sur un fichier ou un dossier

1. *Implementation des notions de cours*

Nous avons reussi a implemente la notion de semaphores dans notre projet sur la fonction coller. En effet, lorsqu’on choisit l’action copier et l’action couper puis on decide de mener les deux actions, le système va donner la priotite a l’action copier, pendant ce temps, l’action couper va etre en veille durant 8 s,tant que l’action copier ne sera pas achever, l’action couper viendra toute les 8s verifier si elle peut déjà se derouler, si l’action copier a fint de s’excecuter, alors couper prendra le relais.

# *IMPLEMENTATION*

from ast import Pass

from cgitb import text

from distutils.dir\_util import copy\_tree

from importlib.resources import path

import os

from re import search

import threading

from time import sleep

import time

import shutil

from tkinter import \*

from tkinter import ttk

from pathlib import \*

from tkinter.ttk import Treeview

from typing import Any, Literal

import subprocess

from tkinter import messagebox as mb

from tkinter.messagebox import \*

from tkinter.simpledialog import\*

import gzip

from zipfile import ZipFile

from zipfile import ZIP\_STORED

window = Tk()

sem = threading.Semaphore()

enc = "utf-8"

try:

        import locale

        locale.setlocale(locale.LC\_ALL,'')

        enc = locale.nl\_langinfo(locale.CODESET)

except (ImportError, AttributeError):

        pass

window.title("My JLD~MANAGER")

window.geometry("2048x720")

window.minsize(1080, 360)

window.iconbitmap('icons8.ico')

window.config(background="#F72B56")

FileOptions =["New", "Open","Close"]

NewOptions=["New folder"]

EditOptions=["Copy","Cut","Compress","Delete","Move","Rename","Decompress"]

EditOptionsAfter=["Undo","Redo"]

AboutOptions=["About "]

menuBar = Menu(window)

File = Menu(menuBar, tearoff=0)

File.add\_command(label="Close",command=window.quit)

New=Menu(File,tearoff=1)

File.add\_cascade(label="New",menu=New)

menuBar.add\_cascade(label="File",menu=File)

Edit = Menu(menuBar, tearoff=0)

menuBar.add\_cascade(label="Edit", menu=Edit)

Sort\_by=Menu(menuBar,tearoff=0)

menuBar.add\_cascade(label="Sort\_by",menu=Sort\_by)

About = Menu(menuBar, tearoff=0)

menuBar.add\_cascade(label="About",menu=About)

window.config(menu=menuBar)

top\_frame = Frame(window)

v = StringVar()

undobutton = Button(top\_frame, text="Undo", width=9)

undobutton.grid(row=0, column=0, sticky='nsew')

redobutton = Button(top\_frame, text="Redo", width=9)

redobutton.grid(row=0, column=1, sticky='nsew')

statusbar = Entry(top\_frame, textvariable=v, width=105)

statusbar.grid(row=0, column=2, sticky='nsew', padx=10)

searchbar=Entry(top\_frame, textvariable='',width=50)

searchbar.grid(row=0,column=3,sticky='nswe')

searchbut=Button(top\_frame,text="Search",width=15)

searchbut.grid(row=0,column=4,sticky='nswe')

top\_frame.grid(row=0, column=0, columnspan=4, sticky='nsew')

left\_frame = Frame(window)

listview = ttk.Treeview(left\_frame, height=34)

listview["columns"] = ("1", "2", "3")

ybar2 = Scrollbar(left\_frame, orient=VERTICAL, command=listview.yview)

listview.configure(yscroll=ybar2.set)

listview.column("#0", width=300, stretch=False)

listview.column("#1", width=200, stretch=False)

listview.column("#2", width=200, stretch=False)

listview.column("#3", width=200, stretch=False)

listview.heading("#0", text="Name", anchor="center")

listview.heading("#1", text="Date Modified", anchor="center")

listview.heading("#2", text="Type", anchor="center")

listview.heading("#3", text="Size", anchor="center")

tv1= ttk.Treeview(left\_frame, height=34)

ybar1 = Scrollbar(left\_frame, orient=VERTICAL, command=tv1.yview)

tv1.configure(yscroll=ybar1.set)

tv1.heading("#0",text="My Computer")

disk=[chr(x)+":\\" for x in range(10,90) if os.path.exists(chr(x)+":\\")]

for d in disk:

    parent\_iid = tv1.insert('', 'end', text=d, open=False)

def actu\_disque():

    disk=[chr(x)+":\\" for x in range(10,90) if os.path.exists(chr(x)+":\\")]

    for d in disk:

        if d not in tv1.get\_children()['text']:

            tv1.insert('', 'end', text=d, open=False)

        else:

            pass

    for i in tv1.get\_children():

        if i['text'] not in disk:

            tv1.delete(i)

        else:

            pass

    window.after(1,actu\_disque)

window.after(1,actu\_disque)

def getpathfromtree():

    item\_iid = tv1.focus()

    path = tv1.item(item\_iid)['text']

    parent = tv1.parent(item\_iid)

    while parent != "":

        node = tv1.item(parent)['text']

        path = os.path.join(node, path)

        parent = tv1.parent(parent)

    return path

def new\_folder(event):

    node = tv1.focus()

    v.set(getpathfromtree())

    directory\_entries = os.listdir(getpathfromtree())

    for name in listview.get\_children():

        listview.delete(name)

    for name in tv1.get\_children(node):

        tv1.delete(name)

    for name in directory\_entries:

        item\_path = getpathfromtree() + os.sep + name

        root = os.path.splitext(name)

        ext = name.split(".")[-1]

        if os.path.isdir(item\_path):

            try:

                tv1.insert(parent=node,

                           index='end',

                           text=name)

                values1 = [time.ctime(os.path.getmtime(item\_path)), 'Folder', '', item\_path]

                listview.insert(parent='', text=name, index="end", values=values1)

            except PermissionError:

                pass

        else:

            tv1.insert(parent=parent\_iid,

                       index='end',

                       text=name)

            values2 = [time.ctime(os.path.getctime(item\_path)), 'File '+ext, str(os.path.getsize(item\_path)) + ' Octets', item\_path]

            listview.insert(parent='', text=root, index="end", values=values2)

            #print(item\_path)

def peuplerlistview(event):

    node = listview.focus()

    chem = (listview.item(node)['values'])[3]

    try:

      if os.path.isdir(chem):

          directory\_entries = os.listdir(chem)

          for name in listview.get\_children():

            listview.delete(name)

          for name in directory\_entries:

            item\_path = os.path.join(chem,name)

            ext = name.split(".")[-1]

            if os.path.isdir(item\_path):

                try:

                    values1 = [time.ctime(os.path.getmtime(item\_path)), 'Folder', '', item\_path]

                    listview.insert(parent='', text=name, index="end", values=values1)

                except PermissionError:

                    pass

            else:

                values2 = [time.ctime(os.path.getctime(item\_path)), 'File '+ext, str(os.path.getsize(item\_path)) + ' Octets', item\_path]

                listview.insert(parent='', text=name, index="end", values=values2)

      else:

             os.startfile(chem)

    except PermissionError as error:

            mb.showerror("Open Failed",message=str(error))

    v.set(chem)

def open():

    node=listview.focus()

    chem=Path(os.path.join(statusbar.get(),listview.item(node)['text']))

    if os.path.isfile(chem):

        try:

          os.startfile(chem)

        except OSError as error:

            mb.showerror("Open Failed",message=str(error))

File.add\_command(label="Open",command=open)

def Openwith():

    node=listview.selection()

    root=os.path.splitext(listview.item(node)['text'])

    chemin=Path(os.path.join(statusbar.get(),listview.item(node)['text']))

    if os.path.isfile(chemin):

        try:

            newpath=Path(os.path.join(statusbar.get(),root))

            os.rename(chemin,newpath)

            pth=Path(os.path.join(statusbar.get(),root))

            os.startfile(pth)

            os.rename(newpath,chemin)

        except OSError as error:

            mb.showerror("Open failed",message=str(error))

File.add\_command(label="Open With",command=Openwith)

def repopulate():

    for i in listview.get\_children():

            listview.delete(i)

    try:

            for name in os.listdir(statusbar.get()) :

               item\_path = os.path.join(statusbar.get(),name)

               ext = name.split(".")[-1]

               if os.path.isdir(item\_path):

                    try:

                        values1 = [time.ctime(os.path.getmtime(item\_path)), 'Folder', '', item\_path]

                        listview.insert(parent='', text=name, index="end", values=values1)

                    except PermissionError:

                        pass

               else:

                    values2 = [time.ctime(os.path.getctime(item\_path)), 'File '+ext, str(os.path.getsize(item\_path)) + ' Octets', item\_path]

                    listview.insert(parent='', text=name, index="end", values=values2)

    except OSError as error:

            mb.showerror('Error',message=str(error))

def inverse1() :

    for i in listview.get\_children():

        listview.delete(i)

    try:

            for name in reversed(os.listdir(statusbar.get())) :

               item\_path = os.path.join(statusbar.get(),name)

               ext = name.split(".")[-1]

               if os.path.isdir(item\_path):

                    try:

                        values1 = [time.ctime(os.path.getmtime(item\_path)), 'Folder', '', item\_path]

                        listview.insert(parent='', text=name, index="end", values=values1)

                    except PermissionError:

                        pass

               else:

                    values2 = [time.ctime(os.path.getctime(item\_path)), 'File '+ext, str(os.path.getsize(item\_path)) + ' Octets', item\_path]

                    listview.insert(parent='', text=name, index="end", values=values2)

    except OSError as error:

            mb.showerror('Error',message=str(error))

Sort\_by.add\_command(label="Alphabet",command=inverse1)

def folder():

    try:

        name=askstring("Create a new folder","Enter the new name ")

        try:

            path=os.path.join(statusbar.get(),name)

            os.mkdir(path)

        except TypeError:

            pass

        for i in listview.get\_children():

            listview.delete(i)

        try:

            for name in os.listdir(statusbar.get()) :

               item\_path = os.path.join(statusbar.get(),name)

               ext = name.split(".")[-1]

               if os.path.isdir(item\_path):

                    try:

                        values1 = [time.ctime(os.path.getmtime(item\_path)), 'Folder', '', item\_path]

                        listview.insert(parent='', text=name, index="end", values=values1)

                    except PermissionError:

                        pass

               else:

                    values2 = [time.ctime(os.path.getctime(item\_path)), 'File '+ext, str(os.path.getsize(item\_path)) + ' Octets', item\_path]

                    listview.insert(parent='', text=name, index="end", values=values2)

            repopulate()

        except OSError as error:

            mb.showerror('Error',message=str(error))

    except OSError as error:

        mb.showerror("You can create a folder",message=str(error))

New.add\_command(label="New Folder",command=folder)

def delete():

    try:

        selected=listview.selection()

        if mb.askyesno("Warning",message="Do you really want to delete this file\folder ?"):

            for i in selected:

                path=(listview.item(i)['values'])[3]

                if os.path.isdir(path):

                    try:

                        shutil.rmtree(path)

                    except OSError as error:

                        mb.showerror("You can delete this folder",message=str(error))

                else:

                    try:

                        os.remove(path)

                    except:

                        mb.showerror("You can delete this file")

        else:

            pass

    except IndexError:

        pass

    repopulate()

Edit.add\_command(label="Delete",command=delete)

def rename():

 node=listview.focus()

 parent=listview.parent(node)

 try:

     oldname=os.path.join(statusbar.get(),listview.item(parent)['text'])

     newname=askstring("Rename","Enter the new name")

     if os.path.isfile(oldname):

         extension=os.path.splitext(oldname)[1]

         newname=newname +extension

     newname=Path(os.path.join(statusbar.get(),newname))

     os.rename(oldname,newname)

     repopulate()

 except OSError as error:

      mb.showerror("Rename failed",message=str(error))

Edit.add\_command(label="Rename",command=rename)

def compress():

    node=listview.focus()

    path=(listview.item(node)['values'])[3]

    try:

            if os.path.isdir(path):

                shutil.make\_archive(path,"zip",path)

                repopulate()

            else:

                with open(path,'rb') as f\_input:

                  with gzip.open(str(path)+'.zip','wb') as f\_output:

                       shutil.copyfileobj(f\_input,f\_output)

                repopulate()

    except TypeError as error:

                pass

Edit.add\_command(label="Compress",command=compress)

def decompress():

    node=listview.selection()

    path=(listview.item(node)['values'])[3]

    shutil.unpack\_archive(str(path))

    repopulate()

Edit.add\_command(label="Decompress",command=decompress)

tabcopy=[]

tabcopydir=[]

tabcut=[]

tabcutdir=[]

def copy():

    global tabcopy

    for i in listview.selection():

       try:

        try:

            path=(listview.item(i)['values'])[3]

            tabcopy.append(path)

            if os.path.isdir(path):

                tabcopydir.append(listview.item(i)['text'])

        except IndexError:

            pass

       except OSError as error:

           mb.showerror("Cut Failed",message=str(error))

Edit.add\_command(label="Copy",command=copy)

def copy1(event):

    global tabcopy

    for i in listview.selection():

       try:

        try:

            path=(listview.item(i)['values'])[3]

            tabcopy.append(path)

            if os.path.isdir(path):

                tabcopydir.append(listview.item(i)['text'])

        except IndexError:

            pass

       except OSError as error:

           mb.showerror("Cut Failed",message=str(error))

    tabcut.clear()

listview.bind('<Control-KeyPress-C>',copy1)

def cut():

    global tabcut

    for i in listview.selection():

       try:

        try :

            path=(listview.item(i)['values'])[3]

            tabcut.append(path)

            if os.path.isdir(path):

                tabcutdir.append(listview.item(i)['text'])

        except IndexError:

            pass

       except OSError as error:

           mb.showerror("Cut Failed",message=str(error))

Edit.add\_command(label="Cut",command=cut)

def cut1(event):

    global tabcut

    for i in listview.selection():

       try:

        try :

            path=(listview.item(i)['values'])[3]

            tabcut.append(path)

            if os.path.isdir(path):

                tabcutdir.append(listview.item(i)['text'])

        except IndexError:

            pass

       except OSError as error:

           mb.showerror("Cut Failed",message=str(error))

    tabcopy.clear()

listview.bind('<Control-KeyPress-X>',cut1)

def paste():

    try:

        try:

            if len(tabcopy)!=0:

             j=0

             sem.acquire()

             for i in tabcopy:

                if os.path.isdir(i):

                    print(tabcopydir[j])

                    copy\_tree(j,os.path.join(statusbar.get(),tabcopydir[j]))

                    sleep(5)

                    j=j+1

                    for k in listview.get\_children():

                        listview.delete(k)

                    for entry in os.listdir(statusbar.get()):

                        try:

                            path1=Path(statusbar.get()+"\\"+entry)

                            listview.insert(parent='',text=entry,values=path1)

                        except OSError as error:

                            mb.showerror('Error',message=str(error))

                    mb.showinfo("Copy Status","Your folder was copied")

                    repopulate()

                else:

                 shutil.copy2(i,statusbar.get())

                 sleep(5)

                 for k in listview.get\_children():

                        listview.delete(k)

                 for entry in os.listdir(statusbar.get()):

                        try:

                            path1=Path(statusbar.get()+"\\"+entry)

                            listview.insert(parent='',text=entry,values=path1)

                        except OSError as error:

                            mb.showerror('Error',message=str(error))

             sem.release()

             mb.showinfo("Copy Status","Your file was copied")

             tabcopydir.clear()

             tabcopy.clear()

        except IndexError:

            pass

    except PermissionError as error:

        mb.showwarning("Error",message="You can not paste")

def paste2():

    try:

        try:

            if len(tabcut)!=0:

             j=0

             for i in tabcut:

                 while not sem.acquire(blocking=False):

                     mb.showinfo("Cut info","The cutting is not ready")

                     sleep(5)

                 else:

                    if os.path.isdir(i):

                        shutil.rmtree(i)

                        copy\_tree(i,os.path.join(statusbar.get(),tabcutdir[j]))

                        j=j+1

                        sleep(5)

                        for x in listview.get\_children():

                            listview.delete(x)

                        for entry in os.listdir(statusbar.get()):

                           try:

                             path1=Path(statusbar.get()+"\\"+entry)

                             listview.insert(parent='',text=entry,values=path1)

                           except OSError as error:

                            mb.showerror('Error',message=str(error))

                        mb.showinfo("Cut status","Cutting finished")

                        repopulate()

                    else :

                          os.remove(i)

                          shutil.copy2(i,statusbar.get())

                          sleep(5)

                          for x in listview.get\_children():

                              listview.delete(x)

                          for entry in os.listdir(statusbar.get()):

                              try:

                                path1=Path(statusbar.get()+"\\"+entry)

                                listview.insert(parent='',text=entry,values=path1)

                              except OSError as error:

                                mb.showerror('Error',message=str(error))

                    mb.showinfo("Cut status","Cutting finished")

             sleep(5)

             repopulate()

             tabcut.clear()

             tabcutdir.clear()

            sem.acquire()

        except IndexError:

            pass

    except PermissionError as error:

        mb.showwarning("Error",message="You can not paste")

p1=threading.Thread(target=paste)

p2=threading.Thread(target=paste2)

p1.start()

p2.start()

p1.join()

p2.join()

Edit.add\_command(label="Paste",command=paste2)

Edit.add\_command(label="Paste",command=paste)

def paste1(event):

    try:

        try:

            if len(tabcopy)!=0:

             j=0

             mb.showwarning("Warnig","Your file is still copy")

             for i in tabcopy:

                if os.path.isdir(i):

                    print(tabcopydir[j])

                    copy\_tree(j,os.path.join(statusbar.get(),tabcopydir[j]))

                    j=j+1

                    for k in listview.get\_children():

                        listview.delete(k)

                    for entry in os.listdir(statusbar.get()):

                        try:

                            path1=Path(statusbar.get()+"\\"+entry)

                            listview.insert(parent='',text=entry,values=path1)

                        except OSError as error:

                            mb.showerror('Error',message=str(error))

                    mb.showinfo("Copy Status","Your folder was copied")

                    repopulate()

                else:

                 shutil.copy2(i,statusbar.get())

                 for k in listview.get\_children():

                        listview.delete(k)

                 for entry in os.listdir(statusbar.get()):

                        try:

                            path1=Path(statusbar.get()+"\\"+entry)

                            listview.insert(parent='',text=entry,values=path1)

                        except OSError as error:

                            mb.showerror('Error',message=str(error))

             mb.showinfo("Copy Status","Your file was copied")

             tabcopydir.clear()

             tabcopy.clear()

            else:

             j=0

             mb.showinfo("Cut info","The cutting is ready")

             for i in tabcut:

                if os.path.isdir(i):

                     shutil.rmtree(i)

                     copy\_tree(i,os.path.join(statusbar.get(),tabcutdir[j]))

                     j=j+1

                     for x in listview.get\_children():

                         listview.delete(x)

                     for entry in os.listdir(statusbar.get()):

                        try:

                            path1=Path(statusbar.get()+"\\"+entry)

                            listview.insert(parent='',text=entry,values=path1)

                        except OSError as error:

                            mb.showerror('Error',message=str(error))

                     mb.showinfo("Cut status","Cutting finished")

                     repopulate()

                else :

                    os.remove(i)

                    shutil.copy2(i,statusbar.get())

                    for x in listview.get\_children():

                        listview.delete(x)

                    for entry in os.listdir(statusbar.get()):

                        try:

                            path1=Path(statusbar.get()+"\\"+entry)

                            listview.insert(parent='',text=entry,values=path1)

                        except OSError as error:

                            mb.showerror('Error',message=str(error))

                    mb.showinfo("Cut status","Cutting finished")

                    repopulate()

             tabcut.clear()

             tabcutdir.clear()

        except IndexError:

            pass

    except PermissionError as error:

        mb.showwarning("Error",message="You can not paste")

    repopulate()

listview.bind('<Control-KeyPress-V>',paste1)

tabpath=[]

def undo():

    global tabpath

    x=tabpath.index(statusbar.get())

    undobutton.config(state=NORMAL)

    if x-1 >=0:

        v.set(tabpath[x-1])

        for i in listview.get\_children():

            listview.delete(i)

        try:

                for name in os.listdir(statusbar.get()):

                    item\_path=os.path.join(statusbar.get(),name)

                    ext=name.split(".")[-1]

                    if os.path.isdir(item\_path):

                     try:

                        values1 = [time.ctime(os.path.getmtime(item\_path)), 'Folder', '', item\_path]

                        listview.insert(parent='', text=name, index="end", values=values1)

                     except PermissionError:

                        pass

                    else:

                     values2 = [time.ctime(os.path.getctime(item\_path)), 'File '+ext, str(os.path.getsize(item\_path)) + ' Octets', item\_path]

                     listview.insert(parent='', text=name, index="end", values=values2)

                repopulate()

        except OSError as error:

              mb.showerror('Error',message=str(error))

    else:

        undobutton.config(state=DISABLED)

Edit.add\_command(label="Undo",command=undo)

undobutton.config(command=undo)

def redo():

      global tabpath

      x=tabpath.index(statusbar.get())

      redobutton.config(command=NORMAL)

      if x+1 < len(tabpath):

          try:

              v.set(tabpath[x+1])

              for i in listview.get\_children():

                  listview.delete(i)

              try:

                    for name in os.listdir(statusbar.get()) :

                      item\_path = os.path.join(statusbar.get(),name)

                      ext = name.split(".")[-1]

                      if os.path.isdir(item\_path):

                        try:

                          values1 = [time.ctime(os.path.getmtime(item\_path)), 'Folder', '', item\_path]

                          listview.insert(parent='', text=name, index="end", values=values1)

                        except PermissionError:

                         pass

                      else:

                         values2 = [time.ctime(os.path.getctime(item\_path)), 'File '+ext, str(os.path.getsize(item\_path)) + ' Octets', item\_path]

                         listview.insert(parent='', text=name, index="end", values=values2)

                    repopulate()

              except OSError as error:

                   mb.showerror('Error',message=str(error))

          except IndexError:

               pass

      else:

            redobutton.config(command=DISABLED)

Edit.add\_command(label="Redo",command=redo)

undobutton.config(command=redo)

def about():

    mb.showinfo("About the application",message="This application is an example of a file manager configured as Window file manager\n It was written in Python 3.8.5 ")

About.add\_command(label="About",command=about)

def searcfunc():

    s=searchbar.get()

    d=statusbar.get()

    o=[]

    if s :

        if d:

            for root,dirs, files in os.walk(d):

                for i in dirs:

                    if s in i:

                        o.append(os.path.join(root,i))

                for f in files:

                    if s in f:

                        o.append(os.path.join(root,f))

            for x in listview.get\_children():

                listview.delete(x)

            for path in o:

                values=[]

                if os.path.isdir(path):

                    valtype="Folder"

                else:

                    valtype="File"

                valname=os.path.basename(path)

                valsize=os.path.getsize(path)

                valtime=time.asctime(time.localtime(os.path.getmtime(path)))

                valpath=path

                values.append(valname)

                values.append(valtime)

                values.append(valtype)

                values.append(valsize)

                values.append(valpath)

                listview.insert("",'end',text=valname,values=values,open=False)

searchbut.config(command=searcfunc)

contextual=Menu(window,tearoff=0)

contextual.add\_command(label="Cut",command=cut)

contextual.add\_separator()

contextual.add\_command(label="Copy",command=copy)

contextual.add\_separator()

contextual.add\_command(label="Paste",command=paste)

contextual.add\_separator()

contextual.add\_command(label="Rename",command=rename)

contextual.add\_separator()

contextual.add\_command(label="Delete",command=delete)

contextual.add\_separator()

contextual.add\_command(label="New Folder",command=folder)

contextual.add\_separator()

contextual.add\_command(label="Compress",command=compress)

contextual.add\_separator()

contextual.add\_command(label="Decompress",command=decompress)

def context(r):

 contextual.tk\_popup(window.winfo\_pointerx(),window.winfo\_pointery())

listview.bind('<Button-3>',context)

tv1.bind('<<TreeviewOpen>>', new\_folder)

tv1.grid(row=1, column=0, sticky='nsew')

ybar1.grid(row=1, column=1, sticky='nsew')

listview.grid(row=1, column=2, columnspan=5, sticky='nsew')

listview.bind('<<TreeviewOpen>>', peuplerlistview)

listview.bind('<<Double-button>>')

listview.bind('<<TreeviewSelect>>')

ybar2.grid(row=1, column=7, sticky='nsew')

left\_frame.grid(row=1, column=0, padx=20, pady=20, sticky='nw')

window.mainloop()

# *CONCLUSION*

En somme il était question pour nous de vous presentez notre projet « My JLD~MANAGER » qui est un gestionnaire de fichier ecrit en python. Ce gestionnaire etant copie sur le model Windows presnte les memes fonctionnalites que cette derniere a l’exeption de la fonctionnalite « ouvrir avec »

# *BIBLIOGRAPHIE*

1. *Think in Python O’REILLY by Marc Hammond*
2. *www.python.org*
3. *Apprendre Python pour les nuls*
4. *Apprenez Tkinter*
5. *Apprenez a programmer avec Python 3*
6. *Pyton Programming on Win32Api O’REILLY by Marc Hammond*