#!/usr/bin/python3

from itertools import count

from time import sleep

import tkinter as TkC

from tkinter import NW, TOP, Tk, Frame, Button, Label, PhotoImage, filedialog

import subprocess

from subprocess import Popen, PIPE, STDOUT

import os

import sys

from math import sqrt, floor, ceil

import tkinter

from xmlrpc.client import boolean

from cv2 import log

from matplotlib.pyplot import fill, text

import yaml

from threading import Thread

import json

import logging

logging.basicConfig(format='Date-Time : %(asctime)s : Line No. : %(lineno)d - %(message)s', level = logging.DEBUG)

class disk\_states:

    disk\_name   :str

    label1      :Label

    end\_task    :bool

    type        :int

    counter     :int

    def \_\_init\_\_(self, counter:int, disk:str, label: Label, type=0):

        self.disk\_name = disk

        self.label1 = label

        self.end\_task = False

        self.type = type

        self.counter = counter

    def task\_kill (self, end: bool):

        self.end\_task = end

    def task\_kill (self):

        self.end\_task = True

    def read\_process(self):

        if self.type == 1:

            p = Popen('lxterminal -e "sudo wipe -Q {0} -kqfZ {1} -R /dev/zero"'.format(self.counter, self.disk\_name), stdout = PIPE, stderr = STDOUT, shell = True)

        elif self.type == 0:

            p = Popen('lxterminal -e "sudo wipe -Q {0} -kqfZ {1} -R /dev/random"'.format(self.counter, self.disk\_name), stdout = PIPE, stderr = STDOUT, shell = True)

        elif self.type == 2:

            p = Popen('lxterminal -e "sudo shred -v -n1 -z {}"'.format(self.disk\_name), stdout = PIPE, stderr = STDOUT, shell = True)

        elif self.type == 3:

            print('lxterminal -e "sudo shred -v -n1 -z \'{0}\' && sudo rm -rf \'{0}\'"'.format(self.disk\_name))

            p = Popen('lxterminal -e "sudo shred -v -n1 -z \'{0}\' && sudo rm -rf \'{0}\'"'.format(self.disk\_name), stdout = PIPE, stderr = STDOUT, shell = True)

        else:

            return

class FlatButton(Button):

    def \_\_init\_\_(self, master=None, cnf=None, \*\*kw):

        Button.\_\_init\_\_(self, master, cnf, \*\*kw)

        self.config(

            compound=TkC.TOP,

            relief=TkC.FLAT,

            bd=0,

            bg="#b91d47",

            fg="white",

            activebackground="#b91d47",

            activeforeground="white",

            highlightthickness=0

        )

    def set\_color(self, color):

        self.configure(

            bg=color,

            fg="white",

            activebackground=color,

            activeforeground="white"

        )

    def set\_font (self, font):

        self.configure(

            font=font

        )

class PiMenu(Frame):

    framestack = []

    icons = {}

    path = ''

    lastinit = 0

    disks = []

    thread\_breaking = False

    items\_doc = []

    drive\_stat = []

    thread\_tasks = []

    disk\_names = []

    disk\_stats = []

    status\_disk = []

    disk\_menu : Frame

    def \_\_init\_\_(self, parent):

        Frame.\_\_init\_\_(self, parent, background="white")

        self.parent = parent

        self.pack(fill=TkC.BOTH, expand=1)

        self.path = os.path.dirname(os.path.realpath(sys.argv[0]))

        self.disk\_menu = Frame(self, bg="#000000")

        disk\_det = Thread(target=self.disk\_info)

        disk\_det.start()

        self.initialize()

    def initialize(self):

        with open(self.path + '/pimenu.yaml', 'r') as f:

            doc = yaml.load(f, Loader=yaml.SafeLoader)

        self.lastinit = os.path.getmtime(self.path + '/pimenu.yaml')

        #while not len(self.disks):

        #    sleep(1)

        doc[0]['items'] = self.disks

        doc[2]['items'] = self.status\_disk

        self.items\_doc = doc

        if len(self.framestack):

            self.destroy\_all()

            self.destroy\_top()

        self.show\_items(items=doc)

    def disk\_info (self):

        with open(self.path + '/disk\_utils.yaml', 'r') as f:

            doc = yaml.load(f, Loader=yaml.SafeLoader)

        while self.thread\_breaking == False:

            self.disks = []

            self.disk\_names = []

            p = Popen('lsblk --json', stdout = PIPE, stderr = STDOUT, shell = True)

            lines = p.stdout.readlines()

            json\_data = ''

            for line in lines:

                json\_data += str(line, 'utf-8')

            objs = json.loads(json\_data)

            for obj in objs['blockdevices']:

                if obj['type'] == 'disk' and obj['name'][0:2] == 'sd':

                    self.disk\_names.append('/dev/' + obj['name'])

                    str\_obj = {

                        'name': '/dev/' + obj['name'],

                        'label': str(obj['name'][-1]).upper() + ':/ disk\n' + obj['size'],

                        'icon': 'database',

                        'color': '#3335c4',

                        'items': doc

                    }

                    if str\_obj['name'] in self.disk\_names:

                        if str\_obj['name'] in self.disk\_stats:

                            continue

                    self.disks.append(str\_obj)

            self.items\_doc[0]['items'] = self.disks

            self.items\_doc[2]['items'] = self.status\_disk

            sleep(5)

    def has\_config\_changed(self):

        return self.lastinit != os.path.getmtime(self.path + '/pimenu.yaml')

    def show\_items(self, items, upper=None):

        if upper is None:

            upper = []

        num = 0

        wrap = Frame(self, bg="black")

        if len(self.framestack):

            self.hide\_top()

            back = FlatButton(

                wrap,

                text='orqaga…',

                image=self.get\_icon("arrow.left"),

                command=self.go\_back,

            )

            back.set\_color("#2b5797")

            back.set\_font("Sans 20")

            back.grid(row=0, column=0, padx=1, pady=1, sticky=TkC.W + TkC.E + TkC.N + TkC.S)

            num += 1

        self.framestack.append(wrap)

        self.show\_top()

        allitems = len(items) + num

        rows = floor(sqrt(allitems))

        cols = ceil(allitems / rows)

        for x in range(int(cols)):

            wrap.columnconfigure(x, weight=1)

        for y in range(int(rows)):

            wrap.rowconfigure(y, weight=1)

        for item in items:

            #print (item['name'])

            if item['name'] == 'empty':

                break

            #if item['name'] in self.disk\_names:

            #    if item['name'] in self.disk\_stats:

            #        break

            act = upper + [item['name']]

            #print (act)

            if 'icon' in item:

                image = self.get\_icon(item['icon'])

            else:

                image = self.get\_icon('scrabble.' + item['label'][0:1].lower())

            btn = FlatButton(

                wrap,

                text=item['label'],

                image=image

            )

            if 'items' in item:

                btn.configure(command=lambda act=act, item=item: self.show\_items(item['items'], act),

                                text=item['label'])

                btn.set\_color("#2b5797")

            else:

                btn.configure(command=lambda act=act: self.go\_action(act), )

            if 'color' in item:

                btn.set\_color(item['color'])

            btn.grid(

                row=int(floor(num / cols)),

                column=int(num % cols),

                padx=1,

                pady=1,

                sticky=TkC.W + TkC.E + TkC.N + TkC.S

            )

            btn.set\_font("Sans 20")

            num += 1

    def get\_icon(self, name):

        if name in self.icons:

            return self.icons[name]

        ico = self.path + '/ico/' + name + '.png'

        if not os.path.isfile(ico):

            ico = self.path + '/ico/' + name + '.gif'

            if not os.path.isfile(ico):

                ico = self.path + '/ico/cancel.gif'

        self.icons[name] = PhotoImage(file=ico)

        return self.icons[name]

    def hide\_top(self):

        self.framestack[len(self.framestack) - 1].pack\_forget()

    def show\_top(self):

        self.framestack[len(self.framestack) - 1].pack(fill=TkC.BOTH, expand=1)

    def destroy\_top(self):

        self.framestack[len(self.framestack) - 1].destroy()

        self.framestack.pop()

    def destroy\_all(self):

        while len(self.framestack) > 1:

            self.destroy\_top()

    def go\_action(self, actions):

        self.hide\_top()

        delay = Frame(self, bg="#2d89ef")

        delay.pack(fill=TkC.BOTH, expand=1)

        print(actions)

        if actions[len(actions) - 1] == 'quit':

            label = Label(delay, text='Dasturdan chiqish...', fg="white", bg="#2d89ef", font="Sans 30")

            label.pack(fill=TkC.BOTH, expand=1)

            self.parent.update()

            self.thread\_breaking = True

            for task\_disk in self.thread\_tasks:

                task\_disk.task\_kill()

            self.quit()

        elif actions[len(actions) - 1] == 'reboot':

            label = Label(delay, text='Tizimini qayta yuklash...', fg="white", bg="#2d89ef", font="Sans 30")

            label.pack(fill=TkC.BOTH, expand=1)

            self.parent.update()

            sleep(3)

            p = Popen('sudo reboot', stdout = PIPE, stderr = STDOUT, shell = True)

        elif actions[len(actions) - 1] == 'shutdown':

            label = Label(delay, text="Tizimini o'chirish...", fg="white", bg="#2d89ef", font="Sans 25")

            label.pack(fill=TkC.BOTH, expand=1)

            self.parent.update()

            sleep(3)

            p = Popen('shutdown -h now', stdout = PIPE, stderr = STDOUT, shell = True)

        elif actions[len(actions) - 1] == 'shred':

            for dds in self.items\_doc[0]['items']:

                if dds['name'] == actions[1]:

                    dds['items'] = [

                        {

                            'name': 'proc',

                            'label': dds['label'] + '\nburdalash jarayonida...',

                            'color': '#2b5797',

                        }

                    ]

                    self.status\_disk.append(dds)

            label = Label(delay, text='{}:/ diskni burdalash'.format(str(actions[1][-1]).upper()), fg="white", bg="#2d89ef", font="Sans 30")

            label.pack(fill=TkC.BOTH, expand=1)

            self.parent.update()

            lbl = Label(self.disk\_menu, text='', fg="white", bg="#2d89ef", font="Sans 15")

            lbl.pack(side=TOP, anchor=NW)

            dsks = disk\_states(disk=actions[1], label=lbl, type=2, counter=1)

            thr = Thread(target=dsks.read\_process)

            self.thread\_tasks.append(dsks)

            thr.start()

            self.disk\_stats.append(actions[1])

            sleep(3)

        elif actions[len(actions) - 1] == 'shredFile':

            label = Label(delay, text='{} burdalash!'.format(str(actions[1][-1]).upper()), fg="white", bg="#2d89ef", font="Sans 30")

            label.pack(fill=TkC.BOTH, expand=1)

            self.parent.update()

            lbl = Label(self.disk\_menu, text='', fg="white", bg="#2d89ef", font="Sans 15")

            lbl.pack(side=TOP, anchor=NW)

            file\_name = filedialog.askopenfilename(initialdir='/media/pi')

            dsks = disk\_states(disk=file\_name, label=lbl, type=3, counter=1)

            thr = Thread(target=dsks.read\_process)

            self.thread\_tasks.append(dsks)

            thr.start()

            self.disk\_stats.append(actions[1])

            sleep(3)

        elif actions[len(actions) - 1] == 'zero':

            for dds in self.items\_doc[0]['items']:

                if dds['name'] == actions[1]:

                    dds['items'] = [

                        {

                            'name': 'proc',

                            'label': dds['label'] + '\nnollar bilan\nto\'ldirish jarayonida...',

                            'color': '#2b5797',

                        }

                    ]

                    self.status\_disk.append(dds)

            label = Label(self.disk\_menu, text='{}:/ diskni nollash'.format(str(actions[1][-1]).upper()), fg="white", bg="#2d89ef", font="Sans 30")

            label.pack(fill=TkC.BOTH, expand=1)

            self.parent.update()

            lbl = Label(self.disk\_menu, text='', fg="white", bg="#2d89ef", font="Sans 15")

            lbl.pack(side=TOP, anchor=NW)

            dsks = disk\_states(disk=actions[1], label=lbl, type=1, counter=int(actions[len(actions)-2]))

            thr = Thread(target=dsks.read\_process)

            self.thread\_tasks.append(dsks)

            thr.start()

            self.disk\_stats.append(actions[1])

            sleep(3)

        elif actions[len(actions) - 1] == 'random':

            for dds in self.items\_doc[0]['items']:

                if dds['name'] == actions[1]:

                    dds['items'] = [

                        {

                            'name': 'proc',

                            'label': dds['label'] + "\ntasodifiy belgilar\nbilan to'ldirish\n jarayonida...",

                            'color': '#2b5797',

                        }

                    ]

                    self.status\_disk.append(dds)

            label = Label(delay, text="{}:/ diskni tasodifiy\nqiymatga to'ldirish".format(str(actions[1][-1]).upper()), fg="white", bg="#2d89ef", font="Sans 30")

            label.pack(fill=TkC.BOTH, expand=1)

            self.parent.update()

            lbl = Label(self.disk\_menu, text='', fg="white", bg="#2d89ef", font="Sans 15")

            lbl.pack(side=TOP, anchor=NW)

            dsks = disk\_states(disk=actions[1], label=lbl, type=0, counter=int(actions[len(actions)-2]))

            thr = Thread(target=dsks.read\_process)

            self.thread\_tasks.append(dsks)

            thr.start()

            self.disk\_stats.append(actions[1])

            sleep(3)

        elif actions[len(actions) - 1] == 'info':

            self.show\_disk\_state()

        delay.destroy()

        self.destroy\_all()

        self.show\_top()

    def show\_disk\_state (self):

        self.hide\_top()

        back = FlatButton(

            self.disk\_menu,

            text='orqaga…',

            image=self.get\_icon("arrow.left"),

            command=self.go\_back,

        )

        back.set\_color("#2b5797")

        back.set\_font("Sans 20")

        back.grid(row=0, column=0, padx=1, pady=1, sticky=TkC.W + TkC.E + TkC.N + TkC.S)

        self.disk\_menu.pack(fill=TkC.BOTH, expand=1)

        self.parent.update()

    def go\_back(self):

        if self.has\_config\_changed():

            self.initialize()

        else:

            self.destroy\_top()

            self.show\_top()

def main():

    root = Tk()

    root.geometry("640x480")

    root.wm\_title('PiMenu')

    if len(sys.argv) > 1 and sys.argv[1] == 'fs':

        root.wm\_attributes('-fullscreen', True)

    PiMenu(root)

    root.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

    main()