**Code and Screenshot of working**

**ext\_disk\_tool.py:**

DISK\_FILE   = "disk.img"

BLOCK\_SIZE  = 512

curdir = ["/"]

curblock = [66]

my\_dict = {}

blockindex = 0

TYPE\_DIR = 0x1111

TYPE\_FILE = 0x2222

DIR\_FREE = 0xffff

class my\_dictionary(dict):

    def \_\_init\_\_(self):

        self = dict()

    def add(self, key, value):

        self[key] = value

class Disk:

    def \_\_init\_\_(self, diskfile, blocksize):

        try:

            self.diskfile = open(diskfile, "r+b")

        except IOError:

            raise Exception("Failed to open disk file")

        self.blocksize = blocksize

        self.blockreads = 0

        self.blockwrites = 0

    def \_\_del\_\_(self):

        self.diskfile.close()

    def readBlock(self, n):

        self.diskfile.seek(self.blocksize \* n)

        data = self.diskfile.read(self.blocksize)

        self.blockreads += 1

        return data

    def writeBlock(self, n, data):

        self.diskfile.seek(self.blocksize \* n)

        self.diskfile.write(data)

        self.blockwrites += 1

    def printStats(self):

        print("")

        print("Disk usage :")

        print(" Blocks Reads Total:  {}".format(self.blockreads))

        print(" Block Writes Total: {}".format(self.blockwrites))

        print("")

def reverseBytes(size):

    updatedSize = []

    total = '0'

    updatedSize = size[::-1]

    for ele in updatedSize:

        total += (ele)

    total = total.replace('0x', '')

    return (int(total,16))

def dataBlock(block):

    metadata = my\_dictionary()

    imgdata = my\_dictionary()

    imgdata\_namelocation = my\_dictionary()

    locations = []

    d = Disk(DISK\_FILE, BLOCK\_SIZE)

    metadata = inodesList(d.readBlock(2))

    for x in range(0, 512, 32):

        inode = block[x]

        if(inode == 255):

            continue

        name = []

        name = block[x:x+24].decode('unicode\_escape').strip()

        if inode in metadata:

            name = name.replace("\x00", "").replace("\x07", "").replace("\x02", "").replace("\x01", "")

            name = name.replace("\x03", "").replace("\x05", "").replace("\x04", "").replace("\x06", "").replace("\x08", "")

            imgdata.key = name

            imgdata.value = metadata[inode]

            imgdata.add(imgdata.key, imgdata.value)

            imgdata[name].append(inode)

            imgdata\_namelocation.key = name

            locations = [imgdata[name][3], imgdata[name][4], imgdata[name][5], imgdata[name][6]]

            imgdata\_namelocation.value = locations

            imgdata\_namelocation.add(imgdata\_namelocation.key, imgdata\_namelocation.value)

            print(metadata[inode][0], end='   ')

            print(metadata[inode][1], end='   ')

            if(int(metadata[inode][2]) !=0 ):

                print(metadata[inode][2], end='  ')

            else:

                print(end='       ')

            print(name)

    return imgdata\_namelocation, imgdata

def indirectData(block):

    for x in range(0, 512):

        while block[x]!=0:

            print(d.readBlock(66+ (block[x])).decode('unicode\_escape').strip())

            x= x+1

def inodesList(block):

    my\_dict = my\_dictionary()

    k = 0

    type = ''

    for x in range(0, 512, 16):

        name = ''

        if(block[x] == 0x11):

            type = 'DIR'

        elif block[x] == 0x22:

            type = 'File'

        else:

            continue

        linkamt = 0

        for x in range(x+2, x+4):

            linkamt += (block[x])

        size = []

        sizeTotal = 0

        for x in range(x+1, x+5):

            size.append(hex(block[x]))

        sizeTotal = reverseBytes(size)

        nodes = []

        direct1 = 0

        for x in range(x+1, x+3):

            if(block[x] != 00):

                nodes.append(hex(block[x]))

        direct1 = reverseBytes(nodes)

        node2 =  []

        direct2 = 0

        for x in range(x+1, x+3):

            if(block[x] != 00):

                node2.append(hex(block[x]))

        direct2 = reverseBytes(node2)

        node3 = []

        direct3 = 0

        for x in range(x+1, x+3):

            if(block[x] != 00):

                node3.append(hex(block[x]))

        direct3 = reverseBytes(node3)

        indirects = []

        indirectsSum = 0

        for x in range(x+1, x+3):

            if(block[x] != 00):

                indirects.append(hex(block[x]))

        indirectsSum = reverseBytes(indirects)

        my\_dict.key = k

        my\_dict.value = [type, linkamt, sizeTotal,direct1, direct2, direct3, indirectsSum]

        my\_dict.add(my\_dict.key, my\_dict.value)

        k =k+1

    return my\_dict

if \_\_name\_\_ == "\_\_main\_\_":

    d = Disk(DISK\_FILE, BLOCK\_SIZE)

    my\_dict = my\_dictionary()

    my\_dict2 = my\_dictionary()

    x=0

    print("Browsing:  {}".format(DISK\_FILE))

    print("Disk label: %s" % d.readBlock(0).decode('unicode\_escape').strip())

    print("\n")

    while True:

        print("Disk File:", \*curdir)

        commands = input().split()

        if commands[0] =="dir":

                my\_dict, my\_dict2 = dataBlock(d.readBlock(curblock[blockindex]))

        if commands[0] =="pwd":

                ##update curdir values

                print('home', \*curdir)

        if commands[0] =="cd":

            ## .. goes to parent  directory

            if commands[1] == '..':

                if len(curdir) != 0:

                    del curdir[-1]

                    del curblock[-1]

                    blockindex = blockindex-1

                else:

                        continue

            elif str(commands[1]) in my\_dict:

                curdir.append(str(commands[1]+"/"))

                curblock.append(66+ int(my\_dict[commands[1]][0]))

                blockindex = blockindex+1

            else:

                print("Not a directory:", str(commands[1]))

        if commands[0] =="stat":

            ## .. goes to parent  directory

            if str(commands[1]) in my\_dict2:

                    print("Name  :", str(commands[1]), "\n")

                    print("Inode :",int(my\_dict2[commands[1]][7]), "\n")

                    print("Type  :", str(my\_dict2[commands[1]][0]), "\n")

                    print("Links :", int(my\_dict2[commands[1]][1]), "\n")

                    print("Size  :",int(my\_dict2[commands[1]][2]), "\n")

                    print("Directs :", int(my\_dict2[commands[1]][3]),int(my\_dict2[commands[1]][4]),int(my\_dict2[commands[1]][5]), "\n")

                    print("Indirects :", int(my\_dict2[commands[1]][6]), "\n")

            else:

                print("No stat:", commands[1])

        if commands[0] =="read":

            if str(commands[1]) in my\_dict:

                print(d.readBlock(66+ int(my\_dict[str(commands[1])][0])).decode('unicode\_escape'))

                while int(my\_dict[str(commands[1])][x]) != 0 and x<3:

                        print(d.readBlock(66+ int(my\_dict[str(commands[1])][x])).decode('unicode\_escape'))

                        x=x+1

                if int(my\_dict[str(commands[1])][3]) != 0:

                    indirectData(d.readBlock(66+ int(my\_dict[str(commands[1])][3])))

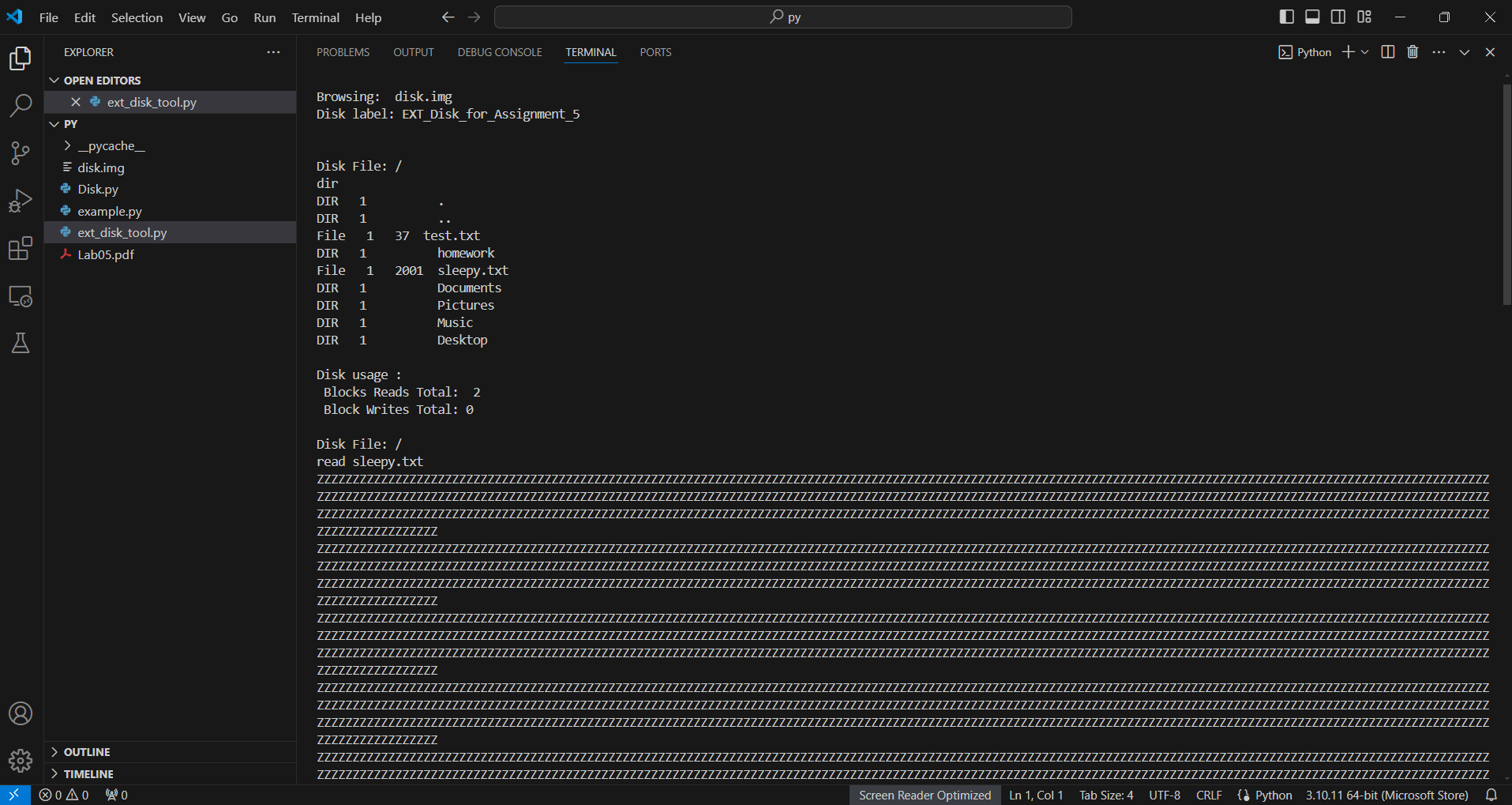
            else:

                    print("Wrong filename:", commands[1])

        if commands[0] =="x":

                break

        d.printStats()



A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated