Національний технічний університет України
“Київський політехнічний інститут
імені Ігоря Сікорського”
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №2

з дисципліни “ Системне програмне забезпечення”
на тему

Виконав:
Студент 4 курсу
групи ІО-62
Лавріненко Н.Т.

Перевірив: Сімоненко А.В.

Київ 2019

Лістинг:

lab2.py

```
import os
import pickle

class Command:
    def __init__(self, func, num_of_args):
        self.func = func
        self.num_of_args = num_of_args

class Image:
    def __init__(self):
        self.files = []

class Descriptor:
    def __init__(self):
        self.size = 0
        self.type = "f"
        self.count_links = 0

class File:
    def __init__(self, name):
        self.name = name
        self.links = []
        self.blocks = []
        self.linked = None
        self.descriptor = Descriptor()

class FileSystem:

    def __init__(self):
        self.is_mount = False
        self.mount_image = None
        self.descriptors = {}
        self.mount_name = ''
        self.commands = {
        "ls"       :    Command(self.ls, 0),
        "mount"    :    Command(self.mount, 1),
        "unmount"  :    Command(self.unmount, 0),
        "filestat" :    Command(self.filestat_id, 1),
        "create"   :    Command(self.create_name, 1),
        "open"     :    Command(self.open_name, 1),
        "read"     :    Command(self.read_name, 2),
        "close"    :    Command(self.close_fd, 1),
```

```
    "write"    :   Command(self.write_fd, 3),
    "link"     :   Command(self.link_name1_name2, 2),
    "unlink"   :   Command(self.unlink_name, 1),
    "truncate" :   Command(self.truncate_name, 2)
    }

def ls(self):
    for i in self.mount_image.files:
        print(i.name)

def mount(self, name):
    if not self.is_mount:
        try:
            with open(name, "rb") as f:
              self.mount_image = pickle.load(f)
            self.is_mount = True
            self.mount_name = name
            self.mount_image.descriptors = {}
            print("Mounted")
        except FileNotFoundError:
            print("Can`t find your file")
    else:
        print("You are already mounted")

def unmount(self):
    self.is_mount = False
    self.mount_image = None
    print("Unmounted")

def filestat_id(self, id):
    if int(id) in self.mount_image.descriptors:
        doc = self.mount_image.descriptors[int(id)]
        descript = doc.descriptor
        link = "->" + doc.linked.name if doc.linked else ''
        print("fd  type  countlinks  size  name")
        print("{} {:>7} {:>10} {:>15} {:>15} {}".format(str(id), descript.type,
        descript.count_links, descript.size, doc.name, link))
    else:
        print("There no file with that fd")

def create_name(self, name):
    self.mount_image.files.append(File(name))
    self.save_iso()
    print("Create file with name ", name)

def open_name(self, name):
    fd = 0 if not self.mount_image.descriptors else\
```

```
            max(self.mount_image.descriptors.keys()) + 1
        doc = self.find_file(name)
        if doc:
            self.mount_image.descriptors[fd] = doc
            print("Open file with fd: ", fd)
        else:
            print("File not exist")


    def read_name(self, fd, shift):
        fd = int(fd)
        shift = int(shift)
        if fd in self.mount_image.descriptors:
            info = ''
            read_file = self.mount_image.descriptors[fd]
            if shift < read_file.descriptor.size:
                info = info.join([str(i) for i in read_file.blocks[shift : read_file.descriptor.size]])
                print(info)
            else:
                print("Size of file {} you try to read from {}\
                    ".format(read_file.descriptor.size, shift))
        else:
            print("File not exist")

    def close_fd(self, fd):
        if int(fd) in self.mount_image.descriptors:
            del self.mount_image.descriptors[int(fd)]
            print("Delete")
        else:
            print("There no file with that fd")

    def write_fd(self, fd, shift, new_info):
        fd = int(fd)
        shift = int(shift)
        if fd in self.mount_image.descriptors:
            write_file = self.mount_image.descriptors[fd]
            need_size = shift + len(new_info)
            if need_size > write_file.descriptor.size:
                write_file.blocks += ['' for _ in range(need_size - write_file.descriptor.size)]
                write_file.descriptor.size = need_size
            for i in range(len(new_info)):
                write_file.blocks[shift + i] = new_info[i]
            self.save_iso()
            print("Writed")

        else:
            print("There no file with that fd")
```

```
def link_name1_name2(self, name1, name2):
    file1 = self.find_file(name1)
    if file1:
        file2 = File(name2)
        file2.descriptor = file1.descriptor
        file2.descriptor.count_links = 0
        file2.linked = file1
        file2.blocks = file1.blocks
        file1.links.append(file2)
        file1.descriptor.count_links += 1
        self.mount_image.files.append(file2)
        self.save_iso()
        print("Linked")
    else:
        print("File not exist")

def unlink_name(self, name):
    doc = self.find_file(name)
    if doc:
        parent_file = doc.linked
        parent_file.links.remove(doc)
        parent_file.descriptor.count_links -= 1
        self.mount_image.files.remove(doc)
        self.save_iso()
        print("Delete link ", name)
    else:
        print("File not exist")

def truncate_name(self, name, size):
    doc = self.find_file(name)
    size = int(size)
    if doc:
        if doc.descriptor.size < size:
            doc.blocks = doc.blocks + ['' for _ in range(size - doc.descriptor.size)]
        elif doc.descriptor.size > size:
            doc.blocks = doc.blocks[:size]
        doc.descriptor.size = size
        self.save_iso()
        print("Trunate to ", size)
    else:
        print("File not exist")

def find_file(self, filename):
    for i in self.mount_image.files:
        if i.name == filename:
            return i
```

```
    def save_iso(self):
        with open(self.mount_name, "wb") as f:
            pickle.dump(self.mount_image, f)



def parse_stdin(file_sys):

    args = input('>').split(' ')
    if (not args[0]):
        return
    try:
        if (file_sys.mount_image or args[0] == "mount"):
            func = file_sys.commands[args[0]].func
            num_of_args = file_sys.commands[args[0]].num_of_args
            if (len(args) - 1 == num_of_args):
                if (num_of_args == 0):
                    func()
                elif (num_of_args == 1):
                    func(args[1])
                elif (num_of_args == 2):
                    func(args[1], args[2])
                elif (num_of_args == 3):
                    func(args[1], args[2], args[3])
            else:
                print("Bad number of args: ", len(args) - 1)
        else:
            print("Please mount filesystem")
    except KeyError:
        print("Unknow command: ", args[0])
        return

file_sys = FileSystem()
while (True):
    parse_stdin(file_sys)
```

Тестування

```
>mount filesystem
Mounted
>ls
file1
linked_file
>open file1
Open file with fd:  0
>read file1
```

```
Bad number of args:  1
>read 0 0
HELLO
>read 0 3
LO
>write 0 2 BY
Writed
>read 0 0
HEBYO
>filestat 0
fd    type    countlinks    size        name
0      f         1              5          file1
>link file1 linked_file1
Linked
>ls
file1
linked_file
linked_file1
>open linked_file1
Open file with fd:  1
>read 1 0
HEBYO
>write 1 0 Write_from_linked
Writed
>read 0 0
Write_from_linked
>filestat 0
fd    type    countlinks    size        name
0      f         1              17          file1
>filestat 1
fd    type    countlinks    size        name
1      f         1              17    linked_file1 ->file1
>truncate file1 100
Trunate to  100
>filestat 0
fd    type    countlinks    size        name
0      f         1            100          file1
>unmount
Unmounted
>ls
Please mount filesystem
>mount filesystem
Mounted
>ls
file1
linked_file
linked_file1
```