**Фам Тхе Винь ИКБО-30-21**

Домашние задания

ДЗ №1. Эмулятор командной строки

|  |
| --- |
| Разработать эмулятор командной строки vshell. В качестве аргумента vshell принимает образ файловой системы известного формата (tar, zip).  Обратите внимание: программа должна запускаться прямо из командной строки, а файл с виртуальной файловой системой не нужно распаковывать у пользователя. В vshell должны поддерживаться команды pwd, ls, cd и cat. Ваша задача сделать работу vshell как можно более похожей на сеанс bash в Linux. Реализовать vshell можно на Python или других ЯП, но кроссплатформенным образом. |

Для тестирования необходимо разместить main.py, archive.py в одном каталоге (для удобства тестируемый архив можно разместить в этом же каталоге, тогда вторым аргументом при запуске main.py передать только имя архива и тип) и запустить main.py.

Эмулятор поддерживает команды cd, ls, ls -l, cat, pwd

**Структура программы:**

main.py — содержит функцию main, которая создает объект класса.

Archive и в бесконечном цикле передает ему аргументы командной строки, обрабатывая их и вызывая соответствующие методы объекта

archive.py — содержит описание класса Archive и несколько вспомогательных функций для работы

import zipfile

import archive

import sys

def main():

    name = sys.argv[1]

    with zipfile.ZipFile(name, 'r') as zip:

        arch = archive.Archive(zip)

        while True:

            print(f"[User]:{arch.currentDir}", end=": ")

            commands = input().split()

            if len(commands) < 1:

                continue

            elif commands[0] == "pwd":

                print(arch.currentDir)

            elif commands[0] == "ls":

                try:

                    arch.getData(commands)

                except Exception as e:

                    print(e)

            elif commands[0] == "cd":

                try:

                    arch.comeDirectory(commands[1])

                except Exception as e:

                    arch.comeDirectory("/"+arch.rootDir)

            elif commands[0] == "cat":

                try:

                    arch.catenate(commands[1])

                except Exception as e:

                    print()

            elif commands[0] == "exit":

                break

            else:

                print("Command is not recognized")

main()

Код main.py

import os

import calendar as c

class Archive:

    def \_\_init\_\_(self, zip):

        self.zip = zip

        self.namelist = zip.namelist()

        self.currentDir = zip.namelist()[0]

        self.rootDir = zip.namelist()[0]

        datalist = zip.infolist()

        self.infodict = {}

        for data in datalist:

            if (data.is\_dir()):

                self.infodict[data.filename[:-1]] = data

            else:

                self.infodict[data.filename] = data

        self.setNamePath = {}

        for name in self.namelist:

            if name == self.currentDir:

                 continue

            modules = cutPath(name)

            self.setNamePath[name] = modules

    def getData(self, commands):

        if len(commands) > 1:

            dir = self.currentDir

            if commands[1] != '-l':

                dir = self.clearPatn(commands[1])

                # path = self.currentDir+commands[1]

                catalog = self.chooseItemInDir(dir)

                print(\*catalog, sep='    ', end='\n')

                # print("ZipFile: {}".format(self.zip.getinfo(path)))

            elif commands[1] == '-l':

                if len(commands) > 2:

                    dir = self.clearPatn(commands[2])

                catalog = self.chooseItemInDir(dir)

                for item in catalog:

                    path = os.path.join(dir, item)

                    try:

                        zipdata = self.infodict[path]

                        d = zipdata.date\_time

                        clock = convertClock(d)

                        print(c.month\_abbr[d[1]]," ",d[2]," ",clock,"\t", zipdata.file\_size, "\t", zipdata.external\_attr, "\t", item)

                    except Exception as e:

                        print(e)

        else:

            catalog = self.chooseItemInDir(self.currentDir)

            print(\*catalog, sep='    ', end='\n')

    def catenate(self, path):

        try:

            dir = self.clearPatn(path)

        except Exception as e:

            print(e)

            return

        try:

            with self.zip.open(dir) as myfile:

                print(myfile.read())

        except Exception as e:

            print(f"{dir} is not file")

    def chooseItemInDir(self, directory):

        partsPath = cutPath(directory)

        border = len(partsPath)

        lastDir = partsPath[border-1]

        items = set()

        for key in self.setNamePath:

            choosenPath = self.setNamePath[key]

            if len(choosenPath) > border:

                if choosenPath[border-1] == lastDir:

                    modules = self.setNamePath[key]

                    items.add(modules[border])

        return items

    def allPath(self):

        print(self.namelist)

    def getAll(self):

        for data in self.infolist:

            print(data)

    def comeDirectory(self, path):

        if path == "/":

            self.currentDir = self.rootDir

        elif path == "..":

            path = self.currentDir

            parts = path.split("/")

            newPath = ''

            if len(parts) == 2:

                newPath = self.rootDir

            for i in range (len(parts)-2):

                newPath += parts[i] + '/'

            self.currentDir = newPath

        else:

            try:

                dir = self.clearPatn(path)

                if (dir in self.namelist) and not "." in dir:

                    self.currentDir = dir

                else:

                    raise ValueError(f"Cannot access '{dir}': No such file or directory")

            except Exception as e:

                print(e)

    def clearPatn(self, path):

        dir = ''

        path = self.normalizePath(path)

        if path[0] == "/" and path[1:] in self.namelist:

            dir = path[1:]

        elif path[0] != "/":

            dir =  os.path.join(self.currentDir, path)

        else:

            raise ValueError(f"Cannot access '{path}': No such file or directory")

        return dir

    def normalizePath(self, path):

        if path[len(path)-1] != '/' and not "." in path:

            path += '/'

        return path

def chooseItemInDir(arch):

    currentDirectory = cutPath(arch.currentDir)

    index = len(currentDirectory)

    items = set()

    for key in arch.setNamePath:

        modules = arch.setNamePath[key]

        items.add(modules[index])

    print(items)

def cutPath(path):

    modules = []

    modules = path.split("/")

    if modules[len(modules)-1] == '':

        modules.pop()

    return modules

def convertClock(d):

    str\_hour = f"0{d[3]}" if d[3] < 10 else str(d[3])

    str\_min = f"0{d[4]}" if d[4] < 10 else str(d[4])

    str\_sec = f"0{d[5]}" if d[5] < 10 else str(d[5])

    return f"{str\_hour}:{str\_min}:{str\_sec}"

Код archive.py

**Демонстрация:**

Изображение выглядит как текст

Автоматически созданное описание