



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Корпоративных информационных систем

ОТЧЕТ ПО ДОМАШНЕЙ РАБОТЕ № 1

по дисциплине

«Конфигурационное управление»

Тема: «Эмулятор командной строки»

Выполнил студент группы ИКБО-02-22

Кузнецов Я.А.

Принял преподаватель

Емельянов А.М.

Работа выполнена

«__»_____2023 г.

(подпись студента)

«Зачтено»

«__»_____2023 г.

(подпись руководителя)

Москва 2023

1 ПОСТАНОВКА ЗАДАЧИ

Разработать эмулятор командной строки vshell. В качестве аргумента vshell принимает образ файловой системы известного формата (tar, zip).

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.

Необходимо поддержать ключ командной строки --script имя_файла для загрузки списка выполняемых команд из файла. Кроме того, в коде должна присутствовать функция тестирования всех реализованных команд.

Задача сделать работу vshell как можно более похожей на сеанс bash в Linux. Реализовать vshell можно на Python или других ЯП, но кроссплатформенным образом.

2 ОПИСАНИЕ ФУНКЦИОНАЛА

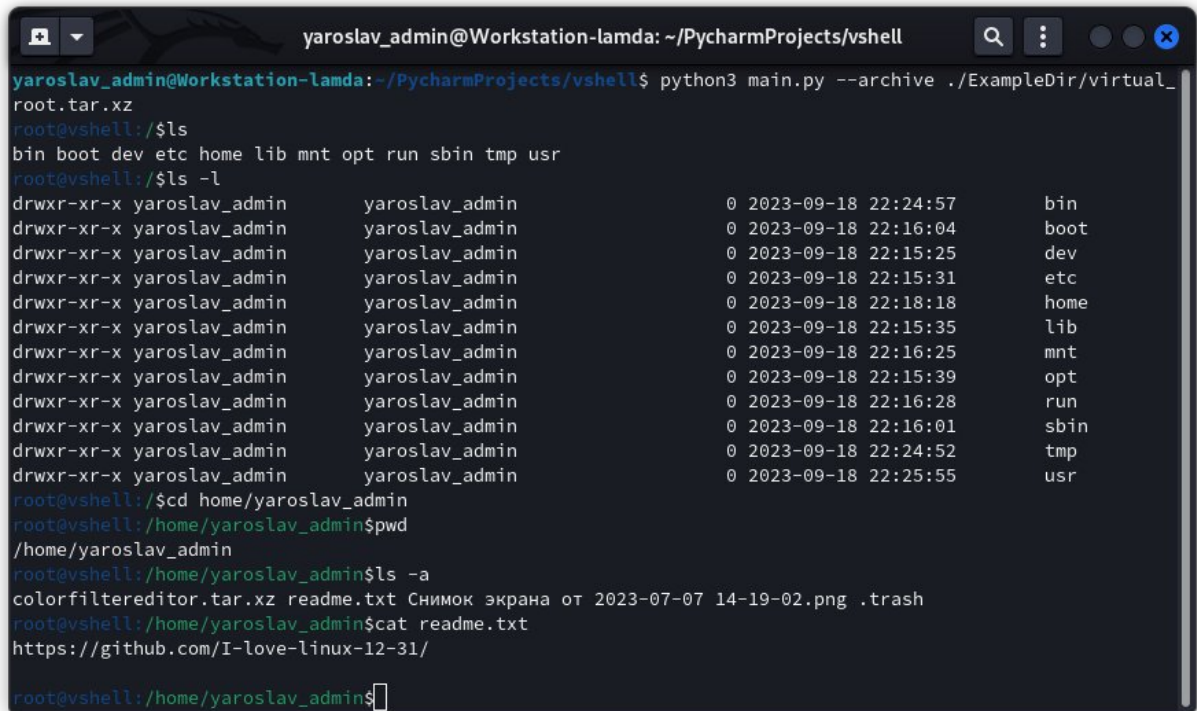
Программа vshell это крос-платформенный эмулятор командной строки в стиле unix работающий внутри архива. Программа поддерживает архивы zip и tar. Vshell написанна на языке Python (3.11+) с использованием библиотек: zipfile, tarfile, stat, datetime, os, sys, io, argparse

Программма поддерживает запуск скриптов из архива. А также работу в интерактивном режиме. Vshell корректно обрабатывает некорректный ввод команд и ситуации, когда команду выполнить нельзя (Перейти по несуществующему пути.).

Поддерживаются команды: cd, ls, echo, pwd, cat, exit.

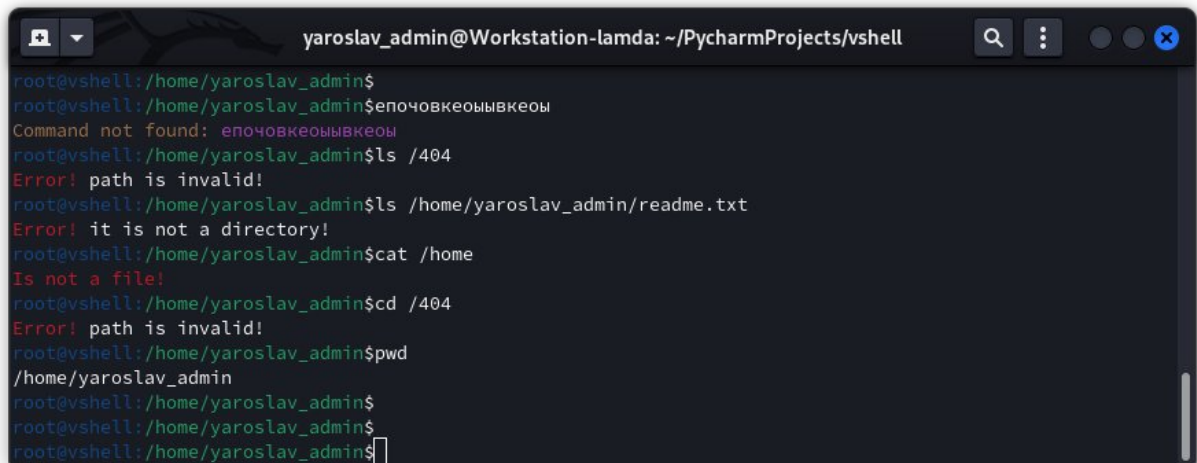
3 ТЕСТИРОВАНИЕ

- 1) Тестирование работы в интерактивном режиме на корректных командах (tar архив)



```
yaroslav_admin@Workstation-lamda: ~/PycharmProjects/vshell
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ python3 main.py --archive ./ExampleDir/virtual_
root.tar.xz
root@vshell:/$ls
bin boot dev etc home lib mnt opt run sbin tmp usr
root@vshell:/$ls -l
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:24:57 bin
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:16:04 boot
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:15:25 dev
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:15:31 etc
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:18:18 home
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:15:35 lib
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:16:25 mnt
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:15:39 opt
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:16:28 run
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:16:01 sbin
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:24:52 tmp
drwxr-xr-x yaroslav_admin yaroslav_admin 0 2023-09-18 22:25:55 usr
root@vshell:/$cd home/yaroslav_admin
root@vshell:/home/yaroslav_admin$pwd
/home/yaroslav_admin
root@vshell:/home/yaroslav_admin$ls -a
colorfiltereditor.tar.xz readme.txt Снимок экрана от 2023-07-07 14-19-02.png .trash
root@vshell:/home/yaroslav_admin$cat readme.txt
https://github.com/I-love-linux-12-31/
root@vshell:/home/yaroslav_admin$
```

- 2) Тестирование работы в интерактивном режиме на не корректных командах



```
yaroslav_admin@Workstation-lamda: ~/PycharmProjects/vshell
root@vshell:/home/yaroslav_admin$
root@vshell:/home/yaroslav_admin$епочовкеоыывкеоы
Command not found: епочовкеоыывкеоы
root@vshell:/home/yaroslav_admin$ls /404
Error! path is invalid!
root@vshell:/home/yaroslav_admin$ls /home/yaroslav_admin/readme.txt
Error! it is not a directory!
root@vshell:/home/yaroslav_admin$cat /home
Is not a file!
root@vshell:/home/yaroslav_admin$cd /404
Error! path is invalid!
root@vshell:/home/yaroslav_admin$pwd
/home/yaroslav_admin
root@vshell:/home/yaroslav_admin$
root@vshell:/home/yaroslav_admin$
root@vshell:/home/yaroslav_admin$
```

3) Тестирование работы с путями

```
yaroslav_admin@Workstation-lamda: ~/PycharmProjects/vshell
root@vshell:/home/yaroslav_admin$
root@vshell:/home/yaroslav_admin$cd .trash
root@vshell:/home/yaroslav_admin/.trash$cat ../s1.sh
Invalid path!
root@vshell:/home/yaroslav_admin/.trash$ls ../
colorfiltereditor.tar.xz readme.txt Снимок экрана от 2023-07-07 14-19-02.png
root@vshell:/home/yaroslav_admin/.trash$cat ../readme.txt
https://github.com/I-love-linux-12-31/

root@vshell:/home/yaroslav_admin/.trash$ls
archive.tar archive.zip dir f2 f2-link make_tar.sh s1.sh
root@vshell:/home/yaroslav_admin/.trash$cat s1.sh
#!/bin/bash

echo "Hello world!"
root@vshell:/home/yaroslav_admin/.trash$cd ../../yaroslav_admin
root@vshell:/home/yaroslav_admin$pwd
/home/yaroslav_admin
root@vshell:/home/yaroslav_admin$cd ../../../.././
root@vshell:/$ls ../../../.././
bin boot dev etc home lib mnt opt run sbin tmp usr
root@vshell:/$
```

4) Тестирование работы с zip архивом

```
yaroslav_admin@Workstation-lamda: ~/PycharmProjects/vshell
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ python3 main.py --archive ./ExampleDir/virtual_
root.zip
Failed to open archive!
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ python3 main.py --archive ./ExampleDir/default_
root.zip
root@vshell:/$ls
bin boot dev etc home lib mnt opt run sbin tmp usr
root@vshell:/$cat /home/yaroslav_admin/readme.txt
https://github.com/I-love-linux-12-31/

root@vshell:/$cd /usr/local/bin
root@vshell:/usr/local/bin$ls
colorfiltereditor HardMyPassword script_1.vsh script_2.vsh
root@vshell:/usr/local/bin$exit
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$
```

5) Тестирование запуска скрипта

```
yaroslav_admin@Workstation-lamda: ~/PycharmProjects/vshell
root.zip
root@vshell:/$ls
bin boot dev etc home lib mnt opt run sbin tmp usr
root@vshell:/$cat /home/yaroslav_admin/readme.txt
https://github.com/I-love-linux-12-31/

root@vshell:/$cd /usr/local/bin
root@vshell:/usr/local/bin$ls
colorfiltereditor HardMyPassword script_1.vsh script_2.vsh
root@vshell:/usr/local/bin$exit
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ python3 main.py --archive ./ExampleDir/default_
root.zip --script /usr/local/bin/script_2.vsh
"This_is_script_2!!!"
bin boot dev etc home lib mnt opt run sbin tmp usr
"End!"
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ python3 main.py --archive ./ExampleDir/default_
```

Исходный код скрипта:

```
yaroslav_admin@Workstation-lamda: ~/PycharmProjects/vshell
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ python3 main.py --archive ./ExampleDir/default_root.zip
root@vshell:/$cat /usr/local/bin/script_2.vsh
echo "This_is_script_2!!!"

ls

echo "End!"
root@vshell:/$
```

6) Тестирование кросс-платформенности программы

```
Вс, 24 сентября 15:39:29
yaroslav_admin@Workstation-lamda: ~/PycharmProjects/vshell
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ cat /proc/version
Linux version 6.4.15-200.fc38.x86_64 (mockbuild@2e6cd98d8465441c8330a02794035256) (gcc (GCC) 13.2.1 20230728 (Red Hat 13.2.1-1), GNU ld version 2.39-9.fc38) #1 SMP PREEMPT_DYNAMIC Thu Sep  7 00:25:01 UTC 2023
yaroslav_admin@Workstation-lamda:~/PycharmProjects/vshell$ python3 main.py --archive ./ExampleDir/default_root.zip
root@vshell:/$ls
bin boot dev etc home lib mnt opt run sbin tmp usr
root@vshell:/$cat /home/yaroslav_admin/readme.txt
https://github.com/I-love-linux-12-31/
root@vshell:/$
```

win10 на QEMU/KVM

туальная машина Вид Отправить комбинацию клавиш

```
Выбрать Администратора: Командная строка - python main.py --archive ./ExampleDir/virtual_root.tar.xz
C:\Users\User\Downloads\vshell>
C:\Users\User\Downloads\vshell>
C:\Users\User\Downloads\vshell>neofetch
##### User@DESKTOP-F1UPT10 #####
##### OS: Майкрософт Windows 10 Pro #####
##### Uptime: 20 minutes and 1 second #####
##### Local IP: 192.168.124.127 #####
##### Motherboard: Not found...?? #####
##### CPU: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz #####
##### GPU: NVIDIA GeForce GT 610 #####
##### GPU: Red Hat QXL controller #####
##### Memory: 2.5GiB / 17.6GiB (13.95%) #####
##### Disk: C: 27.6GiB / 199.9GiB (13.79%) #####
##### D: 3.0GiB / 3.0GiB (100.0%) #####
##### E: 510.0MiB / 510.0MiB (100.0%) #####
#####
C:\Users\User\Downloads\vshell>python main.py --archive ./ExampleDir/virtual_root.tar.xz
root@vshell:/$ls
bin boot dev etc home lib mnt opt run sbin tmp usr
root@vshell:/$cd /home/yaroslav_admin/
root@vshell:/home/yaroslav_admin$cat readme.txt
https://github.com/I-love-linux-12-31/
root@vshell:/home/yaroslav_admin$
```



```
yaroslav_admin@Workstation-lamda:~$ neofetch
      ,,:;::;,,
      ,,:;cccccccccc;,,
      ,,:cccccccccccccccccccc;
      ,,:cccccccccccccccccccccccccccccccccccc;
      ,,:cccccccccccccc;OwMK00XmWd;cccccccc;
      ,,:cccccccccccccc;KMMc;cc;xMMc;cccccccc;
      ,,:cccccccccccccc;MMM;cc;WW;:cccccccc;
      ,,:cccccccccccccc;MMM;cccccccccccccccccc;
      ,,:cccccc;ox00o;MMM00k;cccccccccccccc;
      ,,:cccc;0MMKxdd;MMMkddc;cccccccccccccc;
      ,,:cccc;XM0';cccc;MMM;cccccccccccccccccc'
      ,,:cccc;MMo;cccc;MMW;cccccccccccccccccc;
      ,,:cccc;0Mnc.ccc.xMMd;cccccccccccccccccc;
      ,,:cccc;dnMWXXWM0;cccccccccccccccccc;
      ,,:cccccc;.:odl;.:cccccccccccccccccc;,,
      ,,:cccccccccccccccccccccccccccccccccc;'.
      ,,:cccccccccccccccccccccccccc;,,
      ,,:cccccccccccccccccc;,,
      ,,:cccccccccccccccccc;,,

      OS: Fedora Linux 38 (Workstation Edition) x86_64
      Kernel: 6.5.5-200.fc38.x86_64
      Uptime: 5 hours, 4 mins
      Packages: 4974 (rpm), 59 (flatpak)
      Shell: bash 5.2.15
      Resolution: 1920x1200, 1600x900, 1600x1200
      DE: GNOME 44.5
      WM: Mutter
      WM Theme: Adwaita
      Theme: Kali-Dark [GTK2/3]
      Icons: Flat-Remix-Cyan-Dark [GTK2/3]
      Terminal: gnome-terminal
      CPU: Intel Xeon E5-2650 v4 (48) @ 2.900GHz
      GPU: NVIDIA GeForce GT 610
      Memory: 20131MiB / 64272MiB

      [Color calibration bar with 11 color patches]

yaroslav_admin@Workstation-lamda:~$ python3 /run/media/yaroslav_admin/Projects/PythonProjects/vshell/main.py
Failed to open archive!
yaroslav_admin@Workstation-lamda:~$ python3 /run/media/yaroslav_admin/Projects/PythonProjects/vshell/main.py --archive /run/media/yaroslav_admin/Projects/PythonProjects/vshell/ExampleDir/virtual_root.tar.xz
root@vshell:/#ls
bin boot dev etc home lib mnt opt run sbin tmp usr
root@vshell:/#ls /home/yaroslav_admin/.
colorfiltereditor.tar.xz readme.txt Снимок экрана от 2023-07-07 14-19-02.png
root@vshell:/#cat /home/yaroslav_admin/readme.txt
https://github.com/I-love-linux-12-31/

root@vshell:/#
```

4 ЛИСТИНГ

Файл main.py:

```
import argparse
import os.path
import sys
from io import StringIO

import Lib.vfs
from Lib.argparse import Command
from Lib.utils import *

if sys.platform == "win32":
    import ctypes
    kernel32 = ctypes.windll.kernel32
    kernel32.SetConsoleMode(kernel32.GetStdHandle(-11), 7)

def main(path: str, mode: str = "auto", script: str = None,
external_script: str = None):
    try:
        Lib.vfs.init(path, mode)
    except FileNotFoundError:
        print("\033[31mFailed to open archive!\033[0m")
        exit(1)

    if external_script is not None and
os.path.exists(external_script):
        scriptmode = True
        f = open(external_script, "rt", encoding="utf-8")
        sys.stdin = f
    elif script is not None:
        scriptmode = True
        try:
            target_obj = Lib.vfs.get_object_by_path(script)
        except:
            print("\033[33mBad scrit path!\033[0m")
            exit(1)
        # print("Script:", script)

    if target_obj is not None:
        sys_out = sys.stdout
        sys.stdout = StringIO()

        # cat(args=[script, ])
        Command(f"cat {script}")(cat)

        script_code = sys.stdout.getvalue()
```



```

        sys.stdout.close()
        sys.stdout = sys_out
        sys.stdin = StringIO(script_code)
        # print("Script code:")
        # print(script_code)
        # print("Script code ended")
    else:
        print("Path error!")
        exit(0)

    else:
        scriptmode = False

    cmd = ""
    while cmd.upper() != "EXIT":
        if not scriptmode:
            try:
                cmd =
input(f"\033[34mroot@vshell:\033[32m{Lib.vfs.current_dir}\033[0m$")
            except KeyboardInterrupt:
                exit(0)
        else:
            try:
                cmd = input()
            except UnicodeDecodeError:
                print("\033[31mBad script file encoding! UTF-8
required.\033[0m")
                exit(1)
            except KeyboardInterrupt:
                exit(0)
            except EOFError:
                exit(0)

        if not cmd.strip():
            continue

        parsed_command = Command(cmd)

        try:

            match parsed_command.cmd:
                case "ls":
                    parsed_command(ls)
                case "pwd":
                    parsed_command(pwd)
                case "cd":
                    parsed_command(cd)
                case "cat":
                    parsed_command(cat)
                case "echo":
                    parsed_command(echo)

```

```

        case "exit":
            pass
        case "EXIT":
            pass
        case _:
            print(f"\033[33mCommand not found:
\033[35m{parsed_command.cmd}\033[0m")
    except EOFError:
        return
    except KeyboardInterrupt:
        return
    except Exception as e:
        print(f"\033[31mError:\033[0m {e.__class__.__name__} :
{e}")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='VShell - cross
platform linux shell emulator.'
                                     ' App works in
virtual filesystem inside a archives (tar, zip)')
    parser.add_argument('--archive', type=str, help='Path to
archive.', default="ExampleDir/archive.tar")
    parser.add_argument('--script', type=str, help='Path to script
INSIDE a archive')
    parser.add_argument('--external_script', type=str, help='Path to
script', default=None)
    args = parser.parse_args()
    main(path=args.archive, script=args.script,
external_script=args.external_script)

```

Файл `argparse.py`:

```

class Command:
    cmd: str = ""
    args_raw: list = None
    kwargs: dict = None
    args: list = None

    def __init__(self, line_in: str):
        self.args_raw = line_in.split()
        self.cmd = self.args_raw[0]
        self.kwargs = dict()
        self.args = []

        stack = self.args_raw[1:]
        while stack:
            item = stack.pop(0)
            item: str

```

```

        if item.startswith("--"):
            if len(stack) > 0 and not stack[0].startswith("-"):
                item = item.lstrip("--")
                self.kwargs[item] = stack.pop(0)
            else:
                self.args.append(item)
        else:
            self.args.append(item)

        # for i in self.args:
        #     print(f"arg: {i}")
        #
        # for i in self.kwargs:
        #     print(f"kwarg: {i} : {self.kwargs[i]}")

def __getitem__(self, item):
    if isinstance(item, int):
        return self.args[item]
    return self.kwargs[item]

def __call__(self, function):
    function(*self.args, **self.kwargs)

```

Файл `__init__.py` модуля `vfs`:

```

import tarfile
import zipfile

from .vfs_classes import *
from .vfs_loaders import build_root_from_tar_info,
build_root_from_zip_info

vfs_root = None
current_dir = "/"
vfs_current_dir_obj = None

mode_on_init = ""
archive_path_on_init = ""

def init(path: str, mode="auto"):
    global vfs_root
    global vfs_current_dir_obj

    global mode_on_init, archive_path_on_init

    if mode == "auto":
        if ".zip" in path:
            mode = "zip"

```

```

        elif ".tar" in path:
            mode = "tar"
        else:
            raise IOError("Not supported archive format! Use
only .zip or .tar")

        archive_path_on_init = path
        mode_on_init = mode
        if mode == "zip":
            with zipfile.ZipFile(path, 'r') as zip_archive:
                vfs_root =
build_root_from_zip_info(zip_archive.infolist())
        elif mode == "tar":
            with tarfile.TarFile.open(path) as tar:
                vfs_root = build_root_from_tar_info(tar.getmembers())
        else:
            raise IOError("Not supported mode! Use only 'zip' or 'tar' or
'auto' modes!")

        vfs_current_dir_obj = vfs_root

def simplify_path(path: str) -> str:
    stack = []
    for s in path.split('/'):
        if s in ('', '.'):
            continue
        if s == '..' and len(stack) > 0 and stack[-1] != "..":
            if stack:
                stack.pop()
        else:
            stack.append(s)

    if path.startswith('/'):
        return '/' + '/'.join(stack)
    return "./" + '/'.join(stack)

def get_object_by_path(path: str) -> VFSFile or VFSDirectory or
VFSRoot:
    path = simplify_path(path)
    # if path == "..":
    #     return '/' +
"/".join(vfs_current_dir_obj.path.split("/")[:-1])

    # print("\033[35mDBG:\033[0m Get object by path : ", path)
    if path == '/':
        # print("\033[35mDBG:\033[0m Get object by path : returned
/")
        return vfs_root
    if path == './':

```

```

        # print("\033[35mDBG:\033[0m Get object by path :
returned ./")
        return vfs_current_dir_obj

    if path.startswith("./"):
        path = current_dir + '/' + path[2:]
    if path.startswith("/"):
        cur = vfs_root
    else:
        cur = vfs_current_dir_obj

    # print(cur)

    # print(path)
    stack = simplify_path(path).split("/")
    # print(stack)
    if len(stack) > 1:
        stack = stack[1:]
    if len(stack) == 1 and stack[0] == '':
        return vfs_root
    # print(path)
    # print(stack)
    path_before = []
    while stack:

        cur: VFSObject
        s = stack.pop(0)
        path_before.append(s)
        # print(f"\033[35mDBG:\033[0m path before {path_before} : now
{s} || cur = {cur.name}")
        if s != ".." and s != "./..":
            # print(s, [i.name for i in cur.items])
            cur = cur.find_item_by_name(s)
        else:
            req = '/' + join(path_before[:-2])
            # print("REQ: ", req)
            if req:
                cur = get_object_by_path(req)
            else:
                return vfs_root
        # print(f"\033[35mDBG: Get object by path : returned
{cur}\033[0m")
        return cur

def is_path_exists(path: str) -> bool:
    try:
        obj = get_object_by_path(path)
    except Exception:
        return False

```

Файл `vfs_classes.py` модуля `vfs`:

```
import datetime

class VFSObject:
    name = ""
    access = "-----"
    #          d123456789
    # d - is a directory
    file_size = 0
    items = None
    date = (1980, 1, 1, 0, 0, 0)
    user = "root"
    group = "root"
    path = "/"
    path_in_archive = ""
    owner_name = "Unknown user"
    owner_group_name = "Unknown group"

    def __str__(self) -> str:
        return self.name

    def _get_date_string(self) -> str:
        if isinstance(self.date, tuple):
            return f"{self.date[1]}{self.date[2]} {self.date[0]}"
        return str(datetime.datetime.fromtimestamp(self.date))

    def get_ls_string(self) -> str:
        return f"{self.access} {self.owner_name.ljust(18, ' ')}\t{self.owner_group_name.ljust(18, ' ')}\t" \
            f"{self.file_size:10} {self._get_date_string()[:19]}\t{self.name}"

    def find_item_by_name(self, name: str):
        for item in self.items:
            if item.name == name:
                return item
        return None

    def print_recursion(self, t=0):
        print("\t" * t + '|-', self.name)
        for item in self.items:
            item.print_recursion(t=t + 1)

class VFSRoot(VFSObject):
    def __init__(self):
        self.name = "/"
```



```

        self.access = "drwxrwxr-x"
        self.items = []
        self.path = "/"
        self.owner_name = "root"
        self.owner_group_name = "root"

class VFSDirectory(VFSObject):
    def __init__(self, name, date, path, arh_path,
                  access=None, items=None, owner_name: str = None,
owner_group_name: str = None):
        if access is None:
            access = "d-----"

        if owner_group_name:
            self.owner_group_name = owner_group_name
        if owner_name:
            self.owner_name = owner_name

        self.path_in_archive = arh_path
        self.name = name.strip("/")
        self.path = path.strip("./")
        if not self.path.startswith('/'):
            self.path = '/' + self.path
        self.access = access
        if items is None:
            self.items = []
        else:
            self.items = items
        self.date = date

class VFSFile(VFSObject):
    def __init__(self, name, size, date, path, arh_path,
                  access=None, owner_name: str = None,
owner_group_name: str = None):
        if access is None:
            access = "- - - - -"

        if owner_group_name:
            self.owner_group_name = owner_group_name
        if owner_name:
            self.owner_name = owner_name

        self.name = name.strip("/")
        self.path = path.strip("./")
        self.path_in_archive = arh_path
        if not self.path.startswith('/'):
            self.path = '/' + self.path
        self.file_size = size
        self.access = access

```

```
self.date = date
self.items = tuple()
```

Файл **vfs_loaders.py** модуля **vfs**:

```
import stat
import tarfile
import zipfile
import Lib.vfs.vfs_classes as vfs_classes

def get_access_string_from_tar_info(info: tarfile.TarInfo) -> str:
    if info.isdir():
        res = "d"
    else:
        res = "-"
    flags = oct(info.mode)[2:]
    # res: DRWXRWXRWX
    for i in flags:
        # i: RWX
        match i:
            case '0':
                res += "---"
            case '1':
                res += "--x"
            case '2':
                res += "-w-"
            case '3':
                res += "-wx"
            case '4':
                res += "r--"
            case '5':
                res += "r-x"
            case '6':
                res += "rw-"
            case '7':
                res += "rwx"

    return res

def build_root_from_zip_info(infolist: [...]):
    root = vfs_classes.VFSRoot()
    for node in infolist:
        # print(node)
        node: zipfile.ZipInfo
        path = node.filename.strip("./").strip("/").split("/")
        # print("Loading item ith path: ", node.filename, path)
        name = path[-1]
```

```

current_dir = root

for item in path[:-1]:
    current_dir = current_dir.find_item_by_name(item)

access_flags = node.external_attr >> 16
if access_flags:
    access = stat.filemode(access_flags)
else:
    access = None

if node.is_dir():
    obj = vfs_classes.VFSDirectory(
        name=name, date=node.date_time,
        access=access, path='/' + '/'.join(path),
        arh_path=node.filename,
    )
else:
    obj = vfs_classes.VFSFile(
        name=name, date=node.date_time,
        access=access, size=node.file_size,
        path='/' + '/'.join(path), arh_path=node.filename,
    )

current_dir.items.append(obj)

return root

def build_root_from_tar_info(infolist: [...]):
    root = vfs_classes.VFSRoot()
    passed_paths = set()
    for node in infolist:
        node: tarfile.TarInfo
        if node.name in passed_paths:
            # print("\033[33mScipp wrong item!\033[0m path: ",
node.name)
            continue
        else:
            passed_paths.add(node.name)
            # print(node)
            path = node.name.strip("./").strip("/").split("/")
            # print("Loading item ith path: ", node.name, path)
            name = path[-1]
            current_dir = root

            for item in path[:-1]:
                t = current_dir.find_item_by_name(item)
                if t is None:
                    print("\033[31mError in loading data\033[0m:",
current_dir, item, path)

```

```

        else:
            current_dir = t

    access_flags = get_access_string_from_tar_info(node)

    if node.isdir():
        obj = vfs_classes.VFSDirectory(
            name=name, date=node.mtime,
            access=access_flags, path=node.path,
            arh_path=node.name,
            owner_name=node.uname,
            owner_group_name=node.gname
        )
    else:
        obj = vfs_classes.VFSFile(
            name=name, date=node.mtime,
            access=access_flags, size=node.size,
            path=node.path, arh_path=node.name,
            owner_name=node.uname,
            owner_group_name=node.gname
        )

    current_dir.items.append(obj)

return root

# with zipfile.ZipFile("ExampleDir/archive.zip", 'r') as zip_archive:
#     data = zip_archive.infolist()
# build_root_from_zip_info(data).print_recursion()

# for i in data:
#     print(i)
#     print(i.comment)
#     print(i.CRC)
#     print(i.extra)
#     print(i.internal_attr)
#     print(i.reserved)
#     print(i.external_attr)
#     print(i.internal_attr)
#
#     hi = i.external_attr >> 16
#     if hi:
#         print("Access", stat.filemode(hi))
#     print("=====")

# with tarfile.TarFile.open('ExampleDir/archive.tar') as tar:
#     infos = tar.getmembers()
#     added_paths = set()

```

```

#     data = []
#     for inf in infos:
#         if inf.path not in added_paths:
#             data.append(inf)
#             added_paths.add(inf.path)
#             inf: tarfile.TarInfo
#             print(inf.name)
#             print(inf.path)
#             print(inf.size)
#             print(inf.type)
#             print(inf.gid)
#             print(inf.mode)
#             print("MODE: ", oct(inf.mode))
#             print(inf.get_info())
#
#
# build_root_from_tar_info(data).print_recursion()

```

Файл `__init__.py` модуля `utils`:

```

from .cd import cd
from .ls import ls
from .pwd import pwd
from .cat import cat
from .echo import echo

```

Файл `cat.py` модуля `utils`:

```

import tarfile
import zipfile

import Lib.vfs

def app(target):
    mode = Lib.vfs.mode_on_init
    path = Lib.vfs.archive_path_on_init
    if mode == "auto":
        if ".zip" in path:
            mode = "zip"
        elif ".tar" in path:

```

```

        mode = "tar"
    else:
        raise IOError("Not supported archive format! Use
only .zip or .tar")

    archive_path_on_init = path
    if mode == "zip":
        with zipfile.ZipFile(archive_path_on_init, 'r') as
zip_archive:
            with zip_archive.open(target) as f:
                for line in f:
                    print(line.decode("utf-8"))
    elif mode == "tar":
        # target = '.' + target
        with tarfile.TarFile.open(archive_path_on_init) as tar:
            f = tar.extractfile(target)
            content = f.read()
            print(content.decode("utf-8"))

def cat(*args, **kwargs):
    path = kwargs.setdefault("path", './')
    if len(args) > 0 and not args[0].startswith("-"):
        path = args[0]

    # print(path, "###")
    obj = Lib.vfs.get_object_by_path(path)
    if obj is None:
        print(f"\033[31mInvalid path!\033[0m")
        return False
    if obj.__class__.__name__ != "VFSFile":
        print(f"\033[31mIs not a file!\033[0m")
        return False
    path = obj.path_in_archive
    # print("\033[34mIn app()\033[0m", path)
    app(path)
    return True

```

Файл **cd.py** модуля **utils**:

```

import Lib.vfs

def cd(path: str = './', *args, **kwargs):
    new_path = Lib.vfs.simplify_path(path)
    try:
        if new_path.startswith('/'):
            candidate = Lib.vfs.get_object_by_path(new_path)
            if candidate.__class__.__name__ in ("VFSDirectory",
"VFSRoot"):

```



```

        Lib.vfs.current_dir = new_path
        Lib.vfs.vfs_current_dir_obj =
Lib.vfs.get_object_by_path(new_path)
        elif candidate is None:
            print("\033[31mError!\033[0m path is invalid!")
        else:
            print("\033[31mError!\033[0mIt is not a directory!")
    else:
        # if Lib.vfs.vfs_current_dir_obj.path == "/":
        #     if path.startswith("./"):
        #         path = '/' + path[2:]

        candidate = Lib.vfs.get_object_by_path(path)
        if candidate.__class__.__name__ in ("VFSDirectory",
"VFSRoot"):
            Lib.vfs.vfs_current_dir_obj = candidate
            Lib.vfs.current_dir =
Lib.vfs.vfs_current_dir_obj.path
            elif candidate is None:
                print("\033[31mError!\033[0m path is invalid!")
            else:
                print("\033[31mError!\033[0mIt is not a directory!")
except Exception as e:
    print("\033[31mError!\033[0m[E] path is invalid!")

```

Файл **echo.py** модуля **utils**:

```

import Lib.vfs

def echo(*args, **modes):
    res = ""
    if len(args) > 0:
        res = args[0]

    print(res)

```

Файл **ls.py** модуля **utils**:

```

import Lib.vfs

def ls(*args, **modes):
    path = modes.setdefault("path", './')
    if len(args) > 0 and not args[0].startswith("-"):
        path = args[0]
    default = dict()
    default["color"] = True

```

```

default["list"] = False
default["all"] = False
default["human_readable"] = False

if "--list" in args or "-l" in args:
    modes["list"] = True

if "--all" in args or "-a" in args:
    modes["all"] = True

path = Lib.vfs.simplify_path(path)
target = Lib.vfs.get_object_by_path(path)
if isinstance(target, Lib.vfs.VFSDirectory) or isinstance(target,
Lib.vfs.VFSRoot):
    if not modes.setdefault("list", default["list"]):
        if not modes.setdefault("all", default["all"]):
            print(*[i.name for i in target.items if not
i.name.startswith('.')])
        else:
            print(*[i.name for i in target.items])
    else:
        if not modes.setdefault("all", default["all"]):
            print(*[i.get_ls_string() for i in target.items if
not i.name.startswith('.')], sep="\n")
        else:
            print(*[i.get_ls_string() for i in target.items],
sep="\n")
elif target is None:
    print("\033[31mError!\033[0m path is invalid!")
else:
    print("\033[31mError!\033[0m it is not a directory!")

```

Файл **pwd.py** модуля **utils**:

```

import Lib.vfs

def pwd(*args, **kwargs):
    print(Lib.vfs.current_dir)

```

5 ВЫВОДЫ

В соответствии с поставленной задачей разработана программа vshell на языке программирования python (3.11 и новее). Программа поддерживает архивы zip, tar и его вариации. Реализованна поддержка команд ls (флаги -l

-a), cd, pwd, cat, echo. Программа поддерживает выполнение скриптов. Vshell – кросс-платформенная программа.

Поставленная задача была выполнена. В ходе выполнения задания я повторил работу с командной строкой unix и работу таких утилит как cat, ls и т.д.

6 ПРИМЕЧАНИЕ

Скриншоты в высоком разрешении можно скачать здесь:

https://disk.yandex.ru/d/cjAA_fsPLeb-cg

Исходный код: <https://github.com/I-love-linux-12-31/vshell/>

Список информационных источников

1. Курс: Конфигурационное управление [Электронный ресурс]. URL: <https://online-edu.mirea.ru/course/view.php?id=3943> (дата обращения 24.09.2023).