

Козорез Е.И. Вариант 9 группа ИКБО-20-24 Этап 1. REPL 1 требование: Приложение должно быть реализовано в форме графического интерфейса(GUI). Реализация: Это требование было выполнено с помощью следующих библиотек: `import tkinter as tk from tkinter import scrolledtext`

В интерфейсе предусмотрены поле вывода и поле ввода:

```
self.OutputArea = scrolledtext.ScrolledText(RootWindow, state='disabled', height=20, width=80)
self.OutputArea.pack(padx=10, pady=10)
```

```
self.InputEntry = tk.Entry(RootWindow, width=80) self.InputEntry.pack(padx=10, pady=(0, 10))
```

Пользователь вводит команды в это поле и нажимает Enter, после чего вызывается метод `ProcessCommand`:

```
self.InputEntry.bind("", self.ProcessCommand)
```

2 требование: Заголовок окна должен содержать имя VFS. Реализация: Создаётся главное окно приложения, инициализируемое в классе `VFSemulator`. В конструкторе задаётся заголовок окна, содержащий имя эмулятора:

```
self.VFSname = "VFS Emulator" self.RootWindow.title(self.VFSname)
```

3 требование: Реализовать парсер, который корректно обрабатывает аргументы в кавычках. Реализация: Он разбирает строку с учётом кавычек и экранирования символов:

```
def ParseArguments(self, CommandString): Args = [] CurrentArg = [] InsideQuotes = False EscapeNext = False
```

```
    for char in CommandString:
        if EscapeNext:
            CurrentArg.append(char)
            EscapeNext = False
        elif char == '\\':
            EscapeNext = True
        elif char == '"':
            InsideQuotes = not InsideQuotes
        elif char == ' ' and not InsideQuotes:
            if CurrentArg:
                Args.append(''.join(CurrentArg))
                CurrentArg = []
        else:
            CurrentArg.append(char)
```

4 требование: Реализовать команды-заглушки, которые выводят своё имя и аргументы: `ls`, `cd`. Реализация: Основные команды программы зарегистрированы в словаре `self.Commands`:

```
self.Commands = { "ls": self.CMDls, "cd": self.CMDcd, "exit": self.CMDexit }
```

Команды `ls` и `cd` являются заглушками, которые просто выводят своё имя и переданные аргументы:

```
def CMDls(self, Args): return f"Команда ls вызвана с аргументами: {Args}"
```

```
def CMDcd(self, Args): return f"Команда cd вызвана с аргументами: {Args}"
```

5 требование: Реализовать команду exit. Реализация: `def CMDexit(self, Args): self.Print(f"Команда exit вызвана. Приложение закрывается...") self.RootWindow.destroy()`

6 требование: Продемонстрировать работу прототипа в интерактивном режиме. Необходимо показать примеры работы всей реализованной функциональности, включая обработку ошибок. Реализация: Основная логика взаимодействия с пользователем реализована в методе `ProcessCommand`. Этот метод получает введенную строку, разбирает её на части с помощью парсера, затем выполняет соответствующую команду:

```
def ProcessCommand(self, event): CommandString = self.InputEntry.get() self.InputEntry.delete(0, tk.END)
self.Print(f"${CommandString}\n")
```

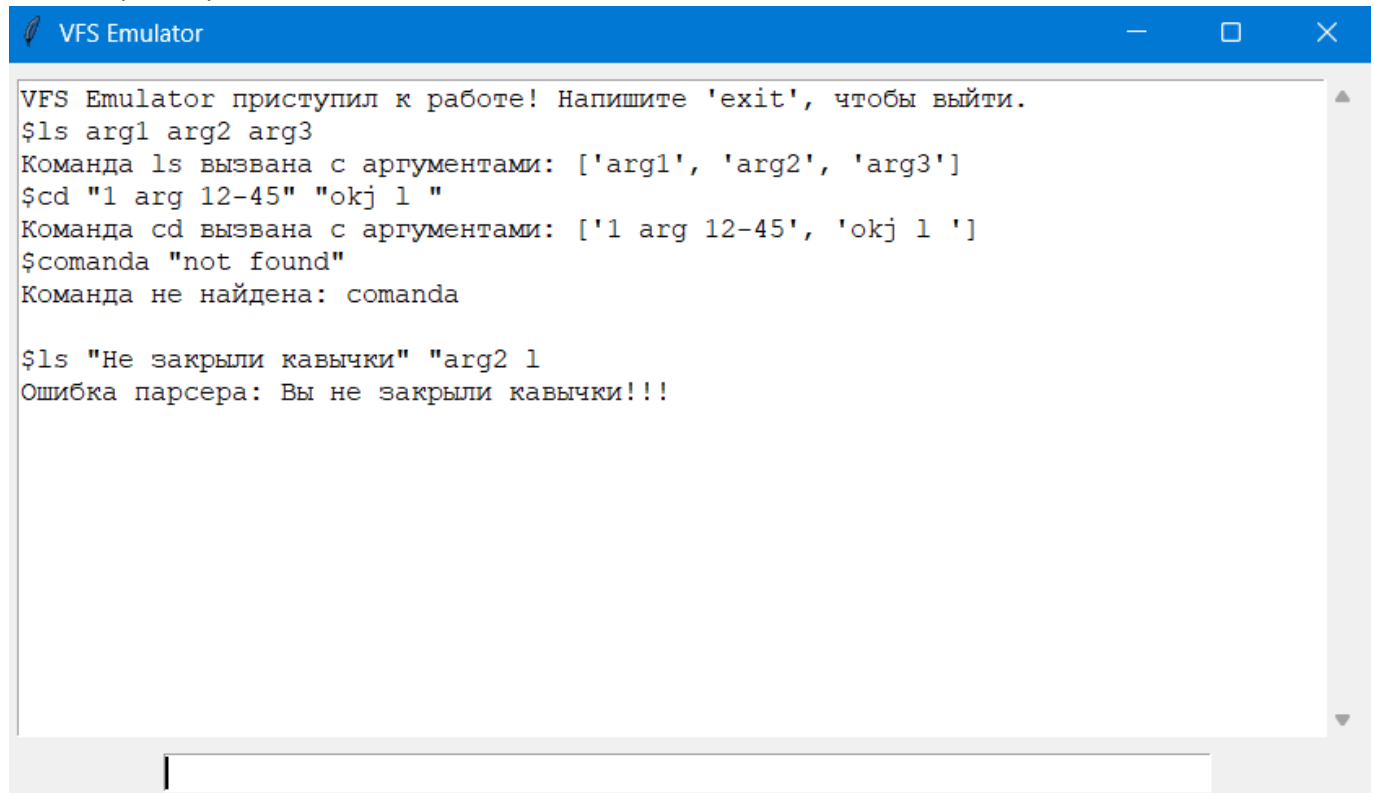
```
try:
    Parts = self.ParseArguments(CommandString)
except ValueError as e:
    self.Print(f"Ошибка парсера: {e}\n\n")
    return

if not Parts:
    self.Print("\n")
    return

CMDname = Parts[0]
CMDargs = Parts[1:]

if CMDname in self.Commands:
    try:
        Result = self.Commands[CMDname](CMDargs)
        if Result:
            self.Print(Result + "\n")
    except Exception as e:
        self.Print(f"Произошла ошибка при выполнении команды '{CMDname}':
{e}\n\n")
    else:
        self.Print(f"Команда не найдена: {CMDname}\n\n")
```

Демонстрация работы:



```

VFS Emulator
VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$ls arg1 arg2 arg3
Команда ls вызвана с аргументами: ['arg1', 'arg2', 'arg3']
$cd "1 arg 12-45" "okj 1 "
Команда cd вызвана с аргументами: ['1 arg 12-45', 'okj 1 ']
$comanda "not found"
Команда не найдена: comanda

$ls "Не закрыли кавычки" "arg2 1
Ошибка парсера: Вы не закрыли кавычки!!!

```

Этап 2:

Цель: сделать эмулятор настраиваемым, то есть поддержать ввод параметров пользователя в приложение. Организовать для этого этапа отладочный вывод всех заданных параметров при запуске эмулятора.

1 требование: Параметры командной строки: – Путь к физическому расположению VFS. – Путь к стартовому скрипту.

Реализация: `self.VFSpath = VFSpath self.StartupScript = StartupScript`

2 требование: Стартовый скрипт для выполнения команд эмулятора: выполняет команды последовательно, ошибочные строки пропускает. При выполнении скрипта на экране отображается как ввод, так и вывод, имитируя диалог с пользователем.

Реализация: Парсинг аргументов командной строки выполнен вручную в функции `ManualParseArgs`, которая извлекает `--vfs` и `--script` из `sys.argv` и передаёт их в конструктор `VFSEmulator`:
`def ManualParseArgs(): VFSpath = None StartupScript = None argv = sys.argv[1:] i = 0 while i < len(argv): if argv[i] == "--vfs" and i+1 < len(argv): VFSpath = argv[i+1] i += 2 elif argv[i] == "--script" and i+1 < len(argv): StartupScript = argv[i+1] i += 2 else: i += 1 return VFSpath, StartupScript`

if **name** == "**main**": VFSpath, StartupScript = ManualParseArgs() RootWindow = tk.Tk() App = VFSEmulator(RootWindow, VFSpath=VFSpath, StartupScript=StartupScript) RootWindow.mainloop()

Метод `RunStartupScript` читает файл скрипта, формирует очередь строк `StartupQueue`, отключает ввод, если `DisableInput=True`, и последовательно вызывает `ExecuteCommandString` для каждой непустой и некомментируемой строки.

```
def RunStartupScript(self, ScriptPath, DelayMs=50, UseComments=True, DisableInput=True): if not
os.path.exists(ScriptPath): self.Print(f"Стартовый скрипт не найден: {ScriptPath}\n\n") self._startup_had_errors
= True return with open(ScriptPath, 'r', encoding='utf-8') as f: RawLines = list(enumerate(f, start=1))
```

```
self.StartupQueue = []
for linenumber, rawline in RawLines:
    line = rawline.rstrip('\n').rstrip('\r')
    self.StartupQueue.append((linenumber, line))

if DisableInput:
    try:
        self.InputEntry.configure(state='disabled')
    except Exception:
        pass

self._running_startup = True
self._startup_had_errors = False

def ProcessNext():
    if not self.StartupQueue:
        if self._startup_had_errors:
            self.Print("Стартовый скрипт завершён с ошибками.\n\n")
        else:
            self.Print("Стартовый скрипт завершён.\n\n")
        self._running_startup = False
        if DisableInput:
            try:
                self.InputEntry.configure(state='normal')
            except Exception:
                pass
        return

    linenumber, line = self.StartupQueue.pop(0)
    stripped = line.strip()

    if not stripped:
        self.RootWindow.after(DelayMs, ProcessNext)
        return

    if UseComments and stripped.startswith('#'):
        self.RootWindow.after(DelayMs, ProcessNext)
        return

    try:
        self.ExecuteCommandString(stripped, EchoInput=True)
    except Exception as e:
        self.Print(f"Необработанная ошибка на строке {linenumber}: {e}\n")
        if getattr(self, "_running_startup", False):
            self._startup_had_errors = True
```

```
self.RootWindow.after(DelayMs, ProcessNext)
```

```
self.RootWindow.after(0, ProcessNext)
```

3 требование: Сообщить об ошибке во время исполнения стартового скрипта. Реализация: Во время выполнения стартового скрипта используется флаг `self._running_startup`. Если произошла ошибка парсера или была введена неизвестная команда, устанавливается `self._startup_had_errors = True`. В конце выводится: "Стартовый скрипт завершён." или "Стартовый скрипт завершён с ошибками." В `ExecuteCommandString` при ошибках:

```
except ValueError as e: self.Print(f"Ошибка парсера: {e}\n\n") if getattr(self, "_running_startup", False):
self._startup_had_errors = True return
```

```
При неизвестной команде: else: self.Print(f"Команда не найдена: {cmd}\n\n") if getattr(self,
"_running_startup", False): self._startup_had_errors = True
```

```
По окончании RunStartupScript: if self._startup_had_errors: self.Print("Стартовый скрипт завершён с
ошибками.\n\n") else: self.Print("Стартовый скрипт завершён.\n\n")
```

```
В CMDexit: def CMDexit(self, Args): if getattr(self, "_startup_had_errors", False): self.Print("Внимание: при
выполнении стартового скрипта были ошибки.\n") self.Print("Команда exit вызвана. Приложение
закрывается...\n") self.RootWindow.destroy()
```

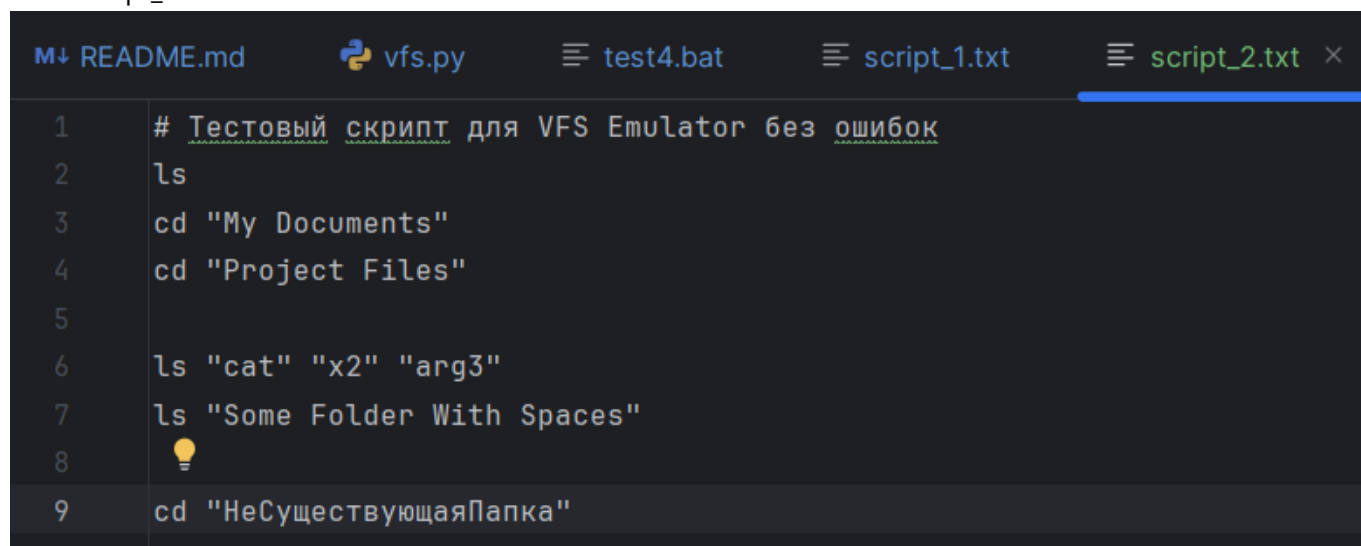
4 Требование: Создать несколько скриптов реальной ОС, в которой выполняется эмулятор. Включить в каждый скрипт вызовы эмулятора для тестирования всех поддерживаемых параметров командной строки.

Реализация: Файл `script_1.txt`:

```

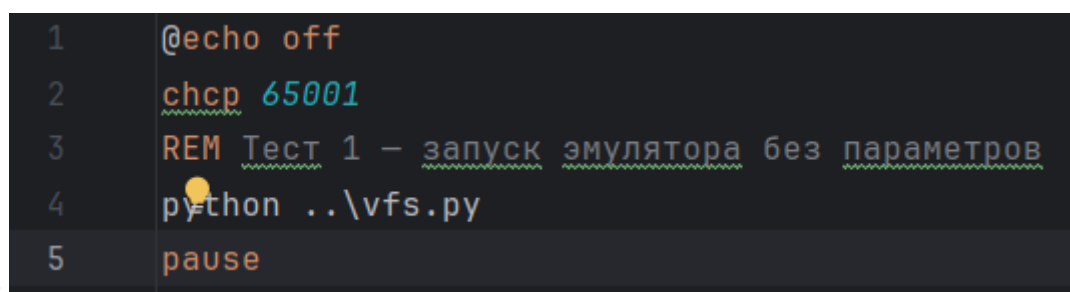
1  # Тестовый скрипт для VFS Emulator с ошибками
2  ls
3  cd "My Documents"
4  ls
5  cd "Project Files"
6  ls "cat" "x2" "arg3"
7  ls "Some Folder With Spaces"
8
9  cd "jld
10 foobar
11 cd "НеСуществующаяПапка"
```

Файл script_2.txt:



```
1 # Тестовый скрипт для VFS Emulator без ошибок
2 ls
3 cd "My Documents"
4 cd "Project Files"
5
6 ls "cat" "x2" "arg3"
7 ls "Some Folder With Spaces"
8
9 cd "НеСуществующаяПапка"
```

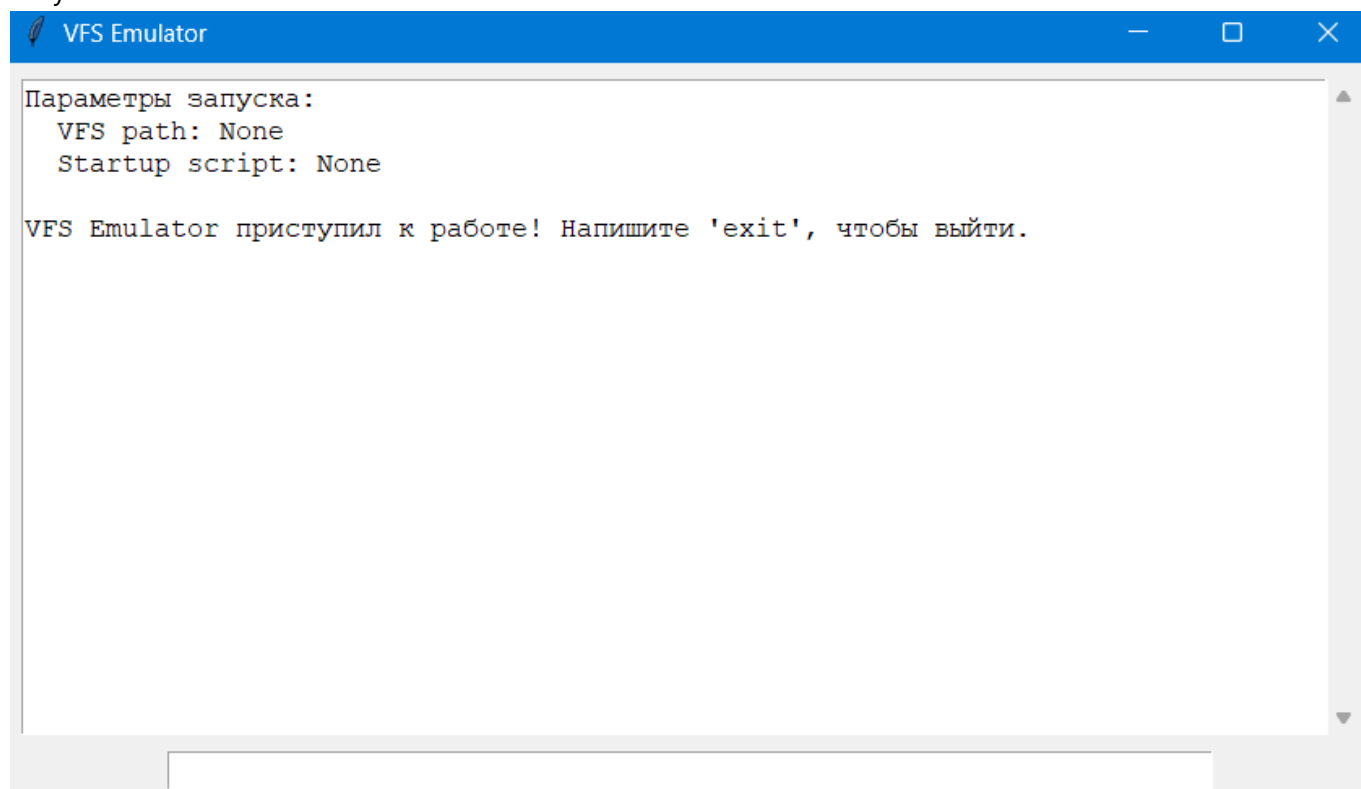
Файл VFS.csv пока что можно оставить пустым, так как он не задействован в программе.



```
1 @echo off
2 chcp 65001
3 REM Тест 1 - запуск эмулятора без параметров
4 python ..\vfs.py
5 pause
```

Файл test1.bat:

Результат выполнения:



```
VFS Emulator

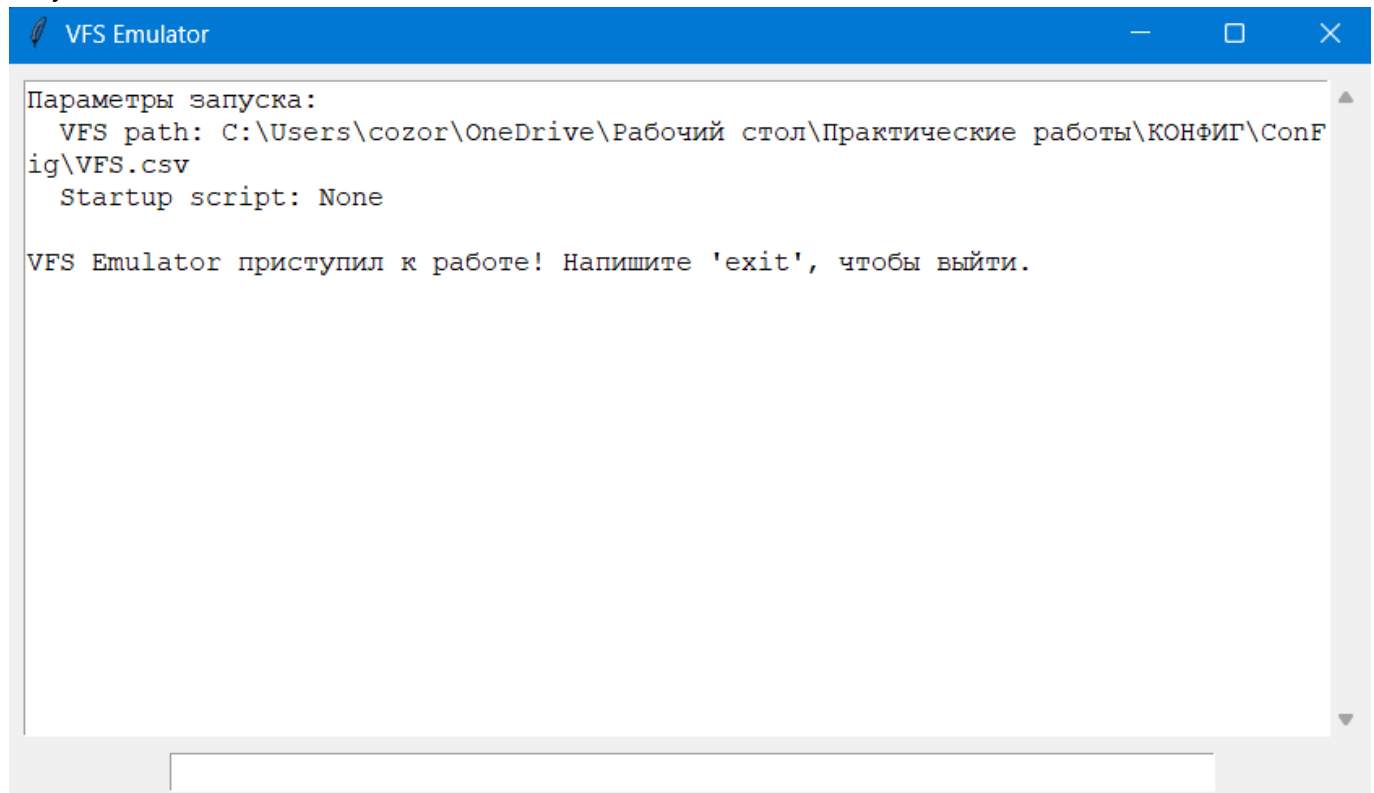
Параметры запуска:
  VFS path: None
  Startup script: None

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
```

Файл test2.bat:

```
1 @echo off
2 chcp 65001
3 REM Тест 2 - указание только VFS
4 python ..\vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\VFS.csv"
5 pause
```

Результат выполнения:

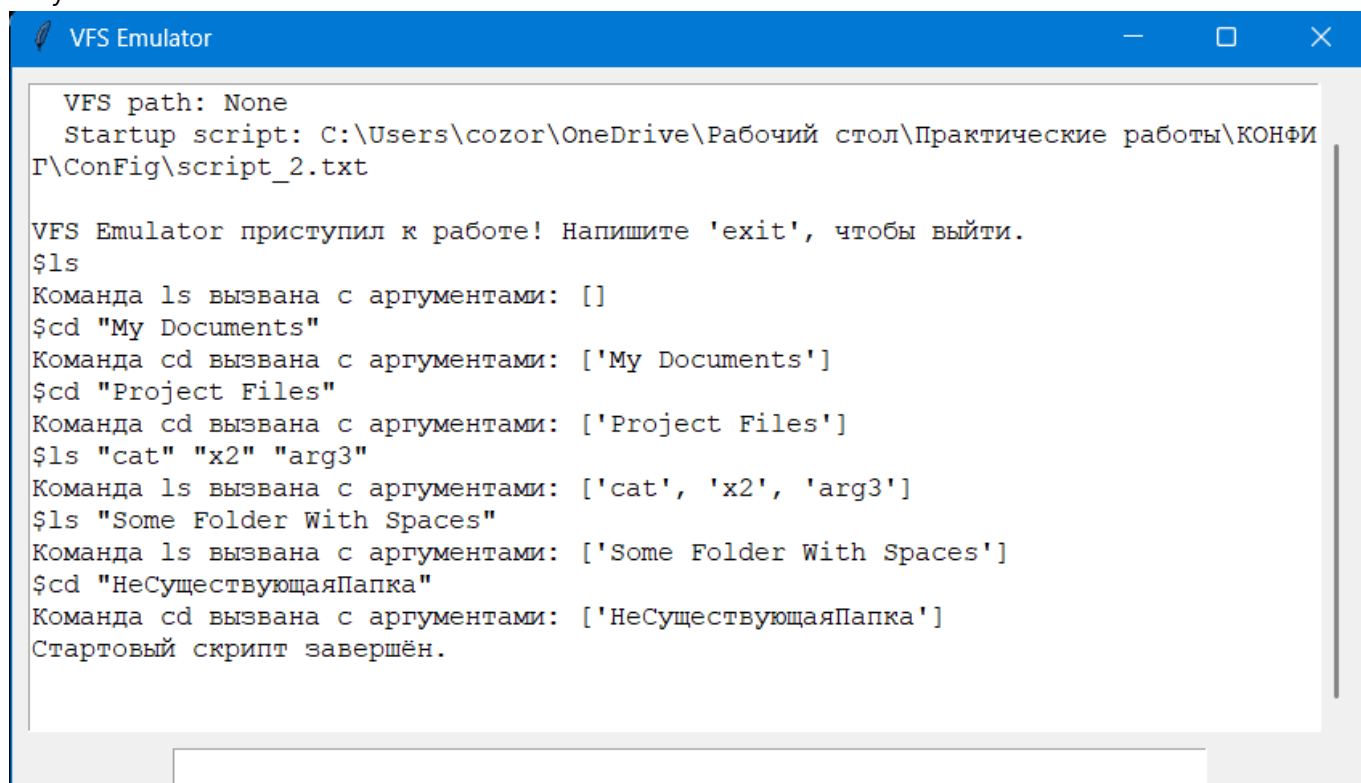


The screenshot shows a window titled "VFS Emulator". Inside, the text displays the execution parameters: "Параметры запуска:", "VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\VFS.csv", and "Startup script: None". Below this, it states "VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти." There is an empty input field at the bottom of the window.

Файл test3.bat:

```
1 @echo off
2 chcp 65001
3 REM Тест 3 - указание только стартового скрипта
4 python ..\vfs.py --script "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_2.txt"
5 pause
```

Результат выполнения:

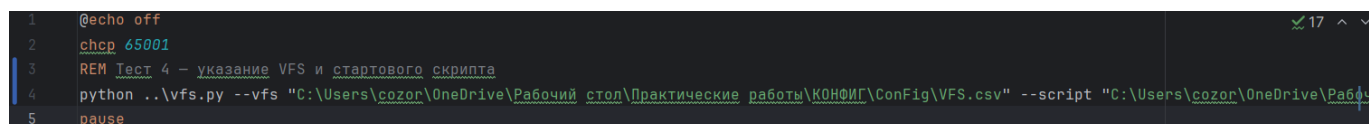


The screenshot shows a window titled "VFS Emulator" with a blue title bar. The main area contains text output from a script. The output shows the VFS path as None, the startup script path, and a series of commands being executed with their arguments. The commands include ls, cd, and ls with various arguments, and the script ends with a message indicating the startup script is complete.

```
VFS path: None
Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_2.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$ls
Команда ls вызвана с аргументами: []
$cd "My Documents"
Команда cd вызвана с аргументами: ['My Documents']
$cd "Project Files"
Команда cd вызвана с аргументами: ['Project Files']
$ls "cat" "x2" "arg3"
Команда ls вызвана с аргументами: ['cat', 'x2', 'arg3']
$ls "Some Folder With Spaces"
Команда ls вызвана с аргументами: ['Some Folder With Spaces']
$cd "НеСуществующаяПапка"
Команда cd вызвана с аргументами: ['НеСуществующаяПапка']
Стартовый скрипт завершён.
```

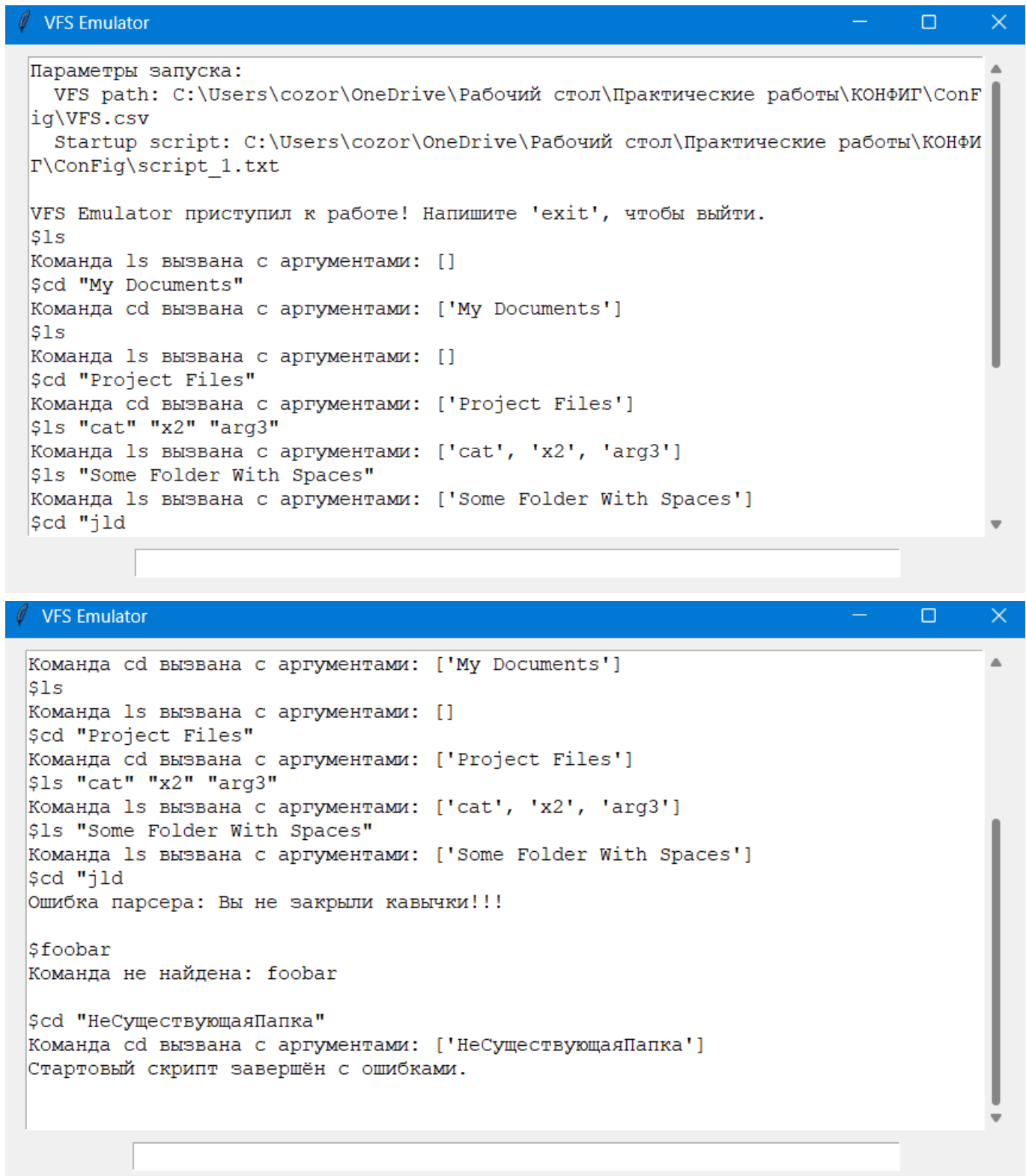
Файл test4.bat:



The screenshot shows a batch file named test4.bat with five lines of code. The first line is @echo off. The second line is chcp 65001. The third line is a comment: REM Тест 4 - указание VFS и стартового скрипта. The fourth line is a python command: python ..\vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\VFS.csv" --script "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_2.txt". The fifth line is pause.

```
1 @echo off
2 chcp 65001
3 REM Тест 4 - указание VFS и стартового скрипта
4 python ..\vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\VFS.csv" --script "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_2.txt"
5 pause
```

Результат выполнения:



The image shows two screenshots of a window titled "VFS Emulator". The first screenshot shows the startup parameters and the initial command execution. The second screenshot shows the continuation of command execution, including an error message for a missing command and a final status message.

```
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
  ig\VFS.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
  Г\ConFig\script_1.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$ls
Команда ls вызвана с аргументами: []
$cd "My Documents"
Команда cd вызвана с аргументами: ['My Documents']
$ls
Команда ls вызвана с аргументами: []
$cd "Project Files"
Команда cd вызвана с аргументами: ['Project Files']
$ls "cat" "x2" "arg3"
Команда ls вызвана с аргументами: ['cat', 'x2', 'arg3']
$ls "Some Folder With Spaces"
Команда ls вызвана с аргументами: ['Some Folder With Spaces']
$cd "jld"

Команда cd вызвана с аргументами: ['My Documents']
$ls
Команда ls вызвана с аргументами: []
$cd "Project Files"
Команда cd вызвана с аргументами: ['Project Files']
$ls "cat" "x2" "arg3"
Команда ls вызвана с аргументами: ['cat', 'x2', 'arg3']
$ls "Some Folder With Spaces"
Команда ls вызвана с аргументами: ['Some Folder With Spaces']
$cd "jld"
Ошибка парсера: Вы не закрыли кавычки!!!

$foobar
Команда не найдена: foobar

$cd "НеСуществующаяПапка"
Команда cd вызвана с аргументами: ['НеСуществующаяПапка']
Стартовый скрипт завершён с ошибками.
```

Этап 3. VFS Цель: подключить виртуальную файловую систему (VFS). 1 Требование: Все операции должны производиться в памяти. Запрещается распаковывать или иным образом модифицировать данные VFS, за исключением возможных служебных команд. Реализация: Виртуальная файловая система полностью создана в оперативной памяти с помощью класса VFSNode. С помощью него создаются узлы, каждый из которых имеет имя, тип, бинарное содержимое для файлов и словарь детей для директорий. Все методы изменяют только объекты в памяти.

```
class VFSNode: def init(self, Name, NodeType="directory", Content=None, Parent=None): self.Name = Name
self.Type = NodeType self.Content = Content if Content else b"" self.Children = {} self.Parent = Parent
```

```

def AddChildren(self, Child):
    if self.Type != "directory":
        raise ValueError("Нельзя добавлять потомка в файл")
    self.Children[Child.Name] = Child
    Child.Parent = self

def GetChild(self, Name):
    return self.Children.get(Name)

def ListChildren(self):
    return list(self.Children.values())

def Path(self):
    parts = []
    node = self
    while node is not None and node.Parent is not None:
        parts.append(node.Name)
        node = node.Parent
    return "/" + "/".join(reversed(parts))

```

2 Требование: Источником VFS является CSV-файл. Для двоичных данных используется base64 или аналогичный формат. Необходимо разобраться, как представлять вложенные элементы VFS.

Реализация: Загрузка VFS из CSV выполняется методом LoadVFSManual, который читает файл через csv.DictReader, разбивает путь на части и строит дерево в памяти. Для файлов содержимое в base64 и декодируется в байты, для директорий создаются соответствующие узлы. Родительские связи строятся по частям пути и сохраняются в словаре nodes для поиска родителя. Также в этом методе происходит изменение названия VFS, соответственно и рабочего окна.

```

def LoadVFSManual(self, CSVPath):
    if not os.path.exists(CSVPath):
        self.Print(f"Файл VFS не найден: {CSVPath}\n")
        self.VFSRoot = None
        self.CurrentDir = None
        self._startup_had_errors = True
        return

    root = VFSNode("/", "directory")
    nodes = {"/": root}

    errors = []
    processed_paths = set()

    try:
        with open(CSVPath, "r", encoding="utf-8-sig", newline='') as csvfile:
            reader = csv.DictReader(csvfile)
            if reader.fieldnames is None:
                self.Print(f"Ошибка формата VFS: CSV не содержит заголовков.\n")

```

```

        self.VFSRoot = None
        self.CurrentDir = None
        self._startup_had_errors = True
        return

    fnames = [fn.strip() for fn in reader.fieldnames]
    if "Path" not in fnames or "Type" not in fnames:
        self.Print(
            f"Ошибка формата VFS: в CSV обязателен столбец 'Path' и
'Type'. Найдены: {'', '.join(fnames)}\n")
        self.VFSRoot = None
        self.CurrentDir = None
        self._startup_had_errors = True
        return

    raw_rows = list(enumerate(reader, start=2)) # start=2 – потому что
header на 1 строке
    for lineno, row in raw_rows:
        raw_path = row.get("Path")
        raw_type = row.get("Type")
        raw_content = row.get("Content", "")

        if raw_path is None or raw_type is None:
            errors.append((lineno, "Отсутствует обязательное поле 'Path'
или 'Type'"))
            continue

        path = raw_path.replace("\\", "/").strip()
        type_ = raw_type.strip()
        content = raw_content.strip() if raw_content else ""

        if not path:
            errors.append((lineno, "Пустое значение Path"))
            continue

        if path == "/":
            continue

        stripped_path = path.strip("/")
        path_parts = [p.strip() for p in stripped_path.split("/") if
p.strip() != ""]
        if not path_parts:
            errors.append((lineno, f"Некорректный Path: '{raw_path}'"))
            continue

        node_name = path_parts[-1]
        parent_path = "/" + "/".join(path_parts[:-1]) if len(path_parts) >
1 else "/"

        parent = nodes.get(parent_path)
        if not parent:

```

```

        errors.append((lineno, f"Родительский путь не найден:
'{parent_path}' для '{path}'"))
        continue

    if type_ == "file":
        try:
            content_bytes = base64.b64decode(content) if content else
b""

        except Exception:
            content_bytes = b""
            errors.append(
                (lineno, f"Невалидный base64 в Content для файла
'{path}' – содержимое будет пустым"))
            node = VFSNode(node_name, "file", content_bytes,
Parent=parent)

        elif type_ == "directory":
            node = VFSNode(node_name, "directory", None, Parent=parent)
        else:
            errors.append((lineno, f"Неизвестный Type '{type_}' в
строке"))
            continue

        if node_name in parent.Children:
            errors.append(
                (lineno, f"Дубликат узла '{node_name}' в '{parent_path}' –
строка проигнорирована"))
            continue

        parent.AddChildren(node)
        processed_paths.add(path)

        if type_ == "directory":
            full_path = parent_path + "/" + node_name if parent_path !=
"/" else "/" + node_name
            nodes[full_path] = node

    if errors:
        self.Print(f"В процессе загрузки VFS обнаружены ошибки в файле
{CSVPath}:\n")
        for lineno, msg in errors:
            self.Print(f"    строка {lineno}: {msg}\n")
        self.Print("\nVFS не загружена из-за ошибок формата.\n")
        self.VFSRoot = None
        self.CurrentDir = None
        self._startup_had_errors = True
        return

    # Успешно загружено (и ошибок не найдено)
    self.VFSRoot = root
    self.CurrentDir = root

```

```

try:
    VFSfilename = os.path.basename(CSVPath)
    vfs_name, _ = os.path.splitext(VFSfilename)
    self.VFSname = vfs_name
    try:
        self.RootWindow.title(f"VFS Emulator - {self.VFSname}")
    except Exception:
        pass
except Exception:
    pass

self.Print(f"VFS загружена успешно: {CSVPath}\n")

except Exception as e:
    self.Print(f"Ошибка при загрузке VFS: {e}\n")
    self.VFSRoot = None
    self.CurrentDir = None
    self._startup_had_errors = True

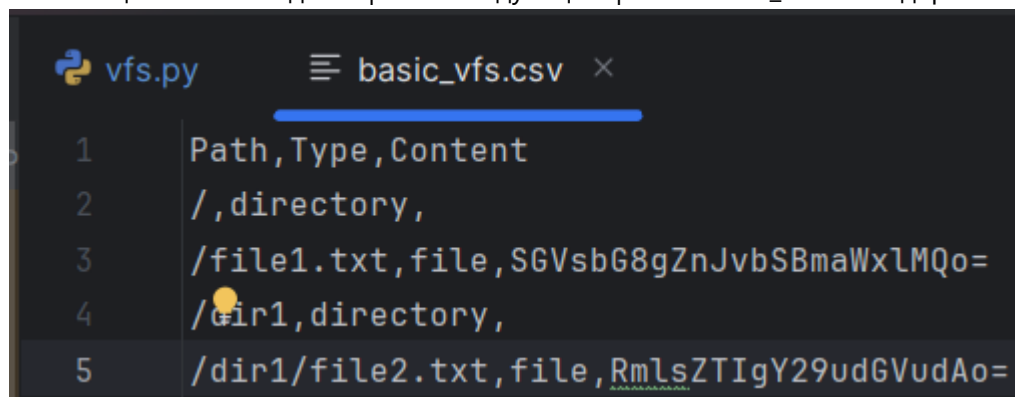
```

3 Требование: Сообщить об ошибке загрузки VFS (файл не найден, неверный формат).

Реализация: Метод LoadVFSManual проверяет наличие файла, корректность заголовков CSV, накапливает ошибки при разборе строк. При наличии ошибок выводит подробный список с номерами строк и прекращает загрузку, устанавливая флаг _startup_had_errors. Код выше.

4 требование: Создать несколько скриптов реальной ОС, в которой выполняется эмулятор. Включить в каждый скрипт вызовы эмулятора для тестирования работы с различными вариантами VFS (минимальный, несколько файлов, не менее 3 уровней файлов и папок).

Реализация: Были созданы файлы следующие файлы: basic_vfs.csv Содержимое:

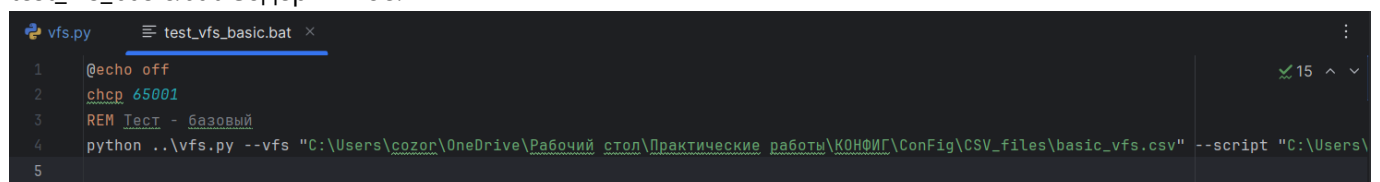


```

1 Path,Type,Content
2 /,directory,
3 /file1.txt,file,SGVsbG8gZnJvbSBmaWx1MQo=
4 /dir1,directory,
5 /dir1/file2.txt,file,RmlsZTIgY29udGVudAo=

```

test_vfs_basic.bat Содержимое:



```

1 @echo off
2 chcp 65001
3 REM Тест - базовый
4 python ../vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\CSV_files\basic_vfs.csv" --script "C:\Users\
5

```

```
test_vfs_basic.bat ×
.csv" --script "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_files\basic_test.txt"
```

```
basic_test.txt ×
1  ls
2  ls /
3  ls "dir1"
4  ls /dir1
5  cd dir1
6  ls
7  cd ..
8  ls file1.txt
9  cd nonexistent
10 ls "unclosed quote|
```

basic_test.txt Содержимое:

multiple_files_vfs.csv Содержимое:

```
basic_test.txt  multiple_files_vfs.csv ×
1  Path,Type,Content
2  /,directory,
3  /rootfile.txt,file,Um9vdCBmaWxlCg==
4  /docs,directory,
5  /docs/readme.txt,file,QW5vdGhlciBmaWxlCg==
6  /bin,directory,
7  /bin/tool.sh,file,RW1wdHkgc2NyaXB0Cg==
8  /bin/empty_dir,directory,
```

test_vsf_multiple_files.bat Содержимое:

```
test_vfs_multiple_files.bat ×
1  @echo off
2  chcp 65001
3  REM Тест с несколькими файлами
4  python ../vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\CSV_files\multiple_files_vfs.csv" --script "
5

--script "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_files\multiple_files_test.txt"
```

multiple_files_test.txt Содержимое:

```
multiple_files_test.txt ×  
1 ls "dir with spaces"  
2 ls "dir with \"escaped\" quotes"  
3 ls "unclosed  
4 # Проверка простых команд  
5 ls  
6 cd /  
7 ls file1.txt  
8 cd dir1  
9 ls  
10 cd ..  
11 cd nonexistent  
12 # Проверка относительных переходов  
13 cd /  
14 cd dir1  
15 cd ..  
16 cd .  
17 cd ..  
18 # Ошибочные команды  
19 foobar|
```

deep_structure_vfs.csv Содержимое:

```
deep_structure_vfs.csv ×
1 Path,Type,Content
2 /,directory
3 /a,directory,
4 /a/file_a.txt,file,TGV2ZWwxCkxldmVsMgpMZlZlbDMK
5 /a/b,directory,
6 /a/b/b_file.txt,file,RGVlcCBmaWxlCg==
7 /a/b/c,directory
8 /a/b/c/deep.txt,file,RGVlcCBmaWxlCg==
9 /x,directory
10 /x/y,directory
11 /x/y/z,directory
12 /x/y/z/zfile.txt,file,Um9vdCB6IGZpbGUK
```

test_vfs_deep_structure.bat Содержимое:

```
test_vfs_deep_structure.bat ×
1 @echo off
2 chcp 65001
3 REM Тест с глубокой структурой
4 python ..\vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\Config\CSV_files\deep_structure_vfs.csv" --script "
5 pause
```



```
test_vfs_deep_structure.bat x
1
2
3
4 structure_vfs.csv" --script "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_files\deep_structure_test.txt"
5
```

```
deep_structure_test.txt x
1 ls /
2 cd /a
3 ls
4 cd b
5 ls
6 cd c
7 ls
8 cd ..
9 cd ..
10 cd ..
11 ls /x/y/z
12 cd /x/y/z
13 ls
14 cat deep.txt
```

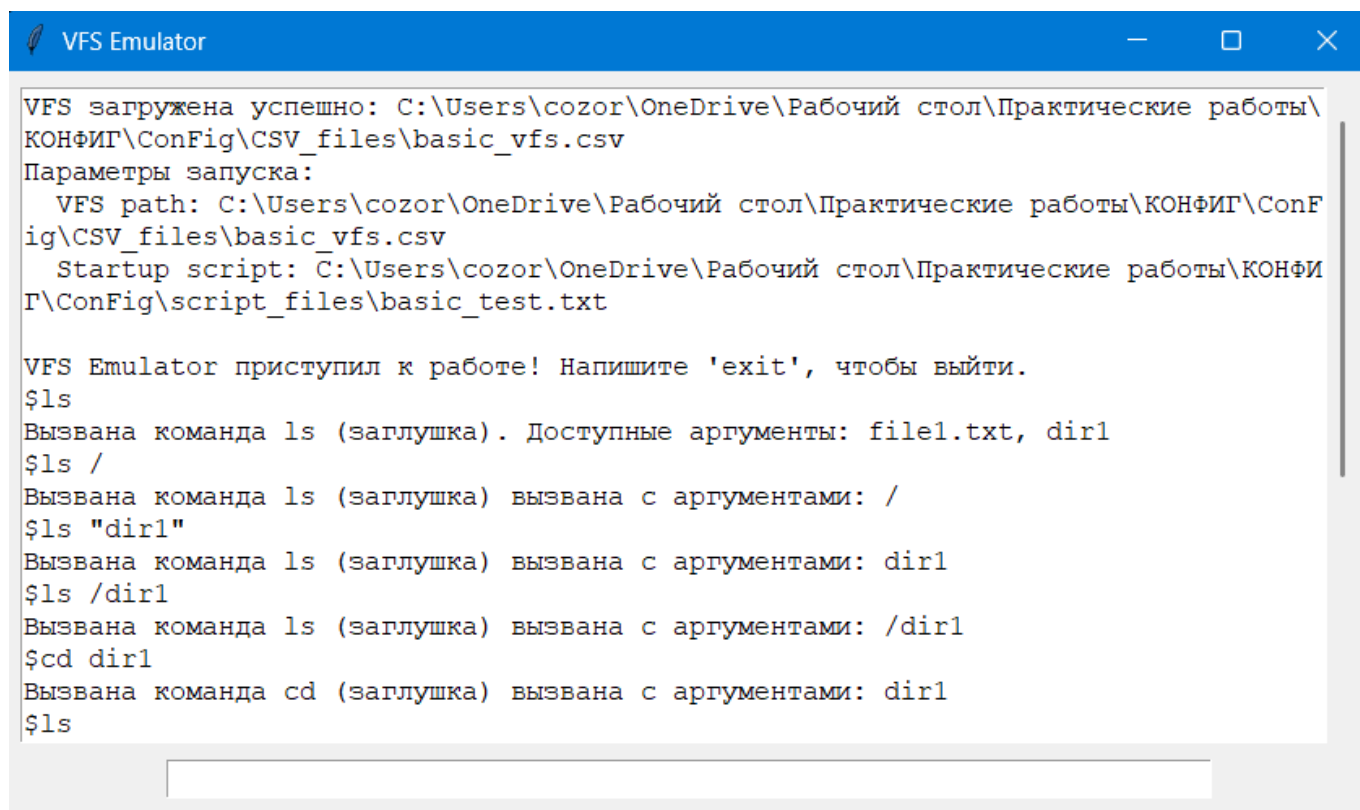
deep_structure_test.txt Содержимое:

С помощью файла test_comprehensive.bat происходит запуск трёх предыдущих тестов. Содержимое:

```
deep_structure_test.txt test_comprehensive.bat x
1 @echo off
2 chcp 65001
3
4 call test_vfs_basic.bat
5 call test_vfs_multiple_files.bat
6 call test_vfs_deep_structure.bat
7
8 pause
```

Результат запуска файла test_comprehensive.bat: Будет создано 3 окна VFS Emulator, которые будут выполнять друг за другом команды из basic_test.txt, multiple_files_test.txt, deep_structure_test.txt:

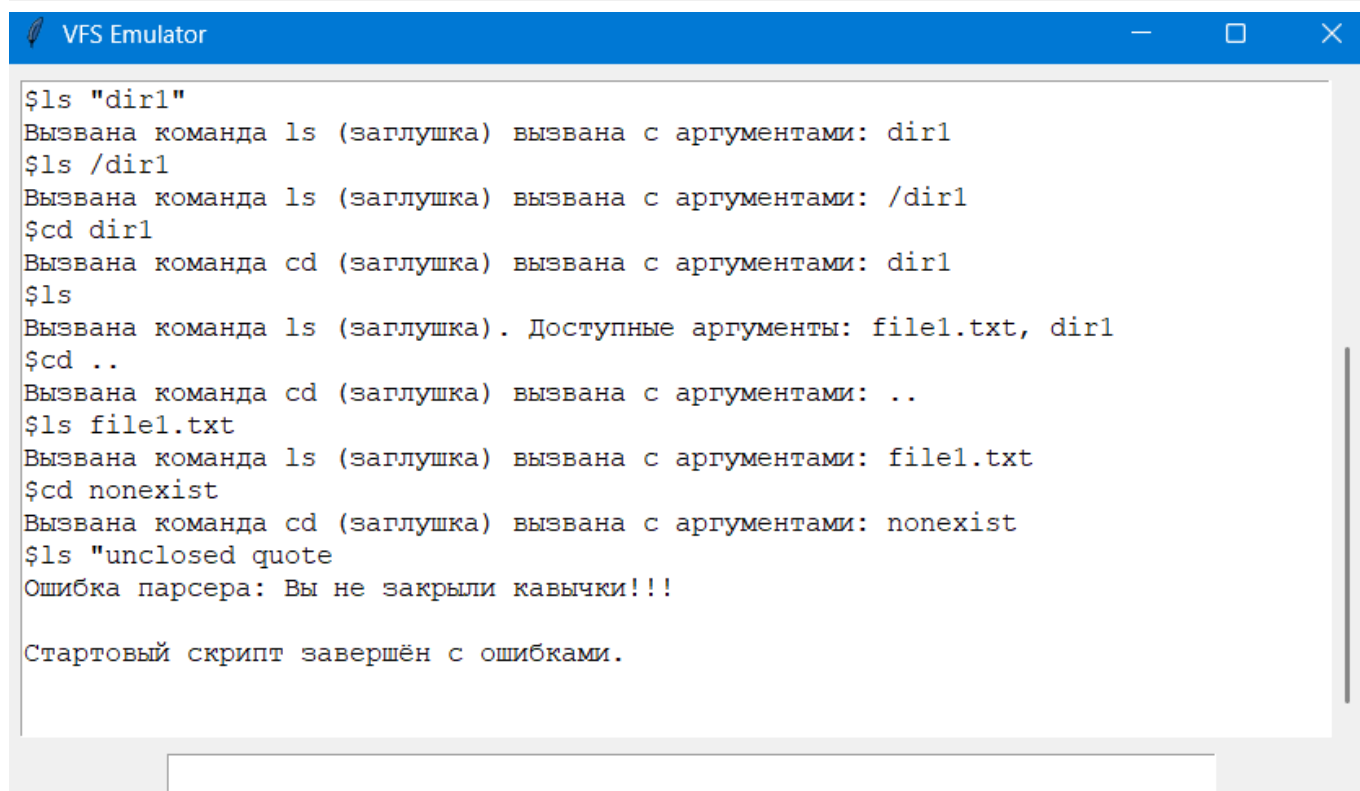
1 окно:



VFS Emulator

```
VFS загружена успешно: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\
КОНФИГ\ConFig\CSV_files\basic_vfs.csv
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
ig\CSV_files\basic_vfs.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\basic_test.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$ls
Вызвана команда ls (заглушка). Доступные аргументы: file1.txt, dir1
$ls /
Вызвана команда ls (заглушка) вызвана с аргументами: /
$ls "dir1"
Вызвана команда ls (заглушка) вызвана с аргументами: dir1
$ls /dir1
Вызвана команда ls (заглушка) вызвана с аргументами: /dir1
$cd dir1
Вызвана команда cd (заглушка) вызвана с аргументами: dir1
$ls
```

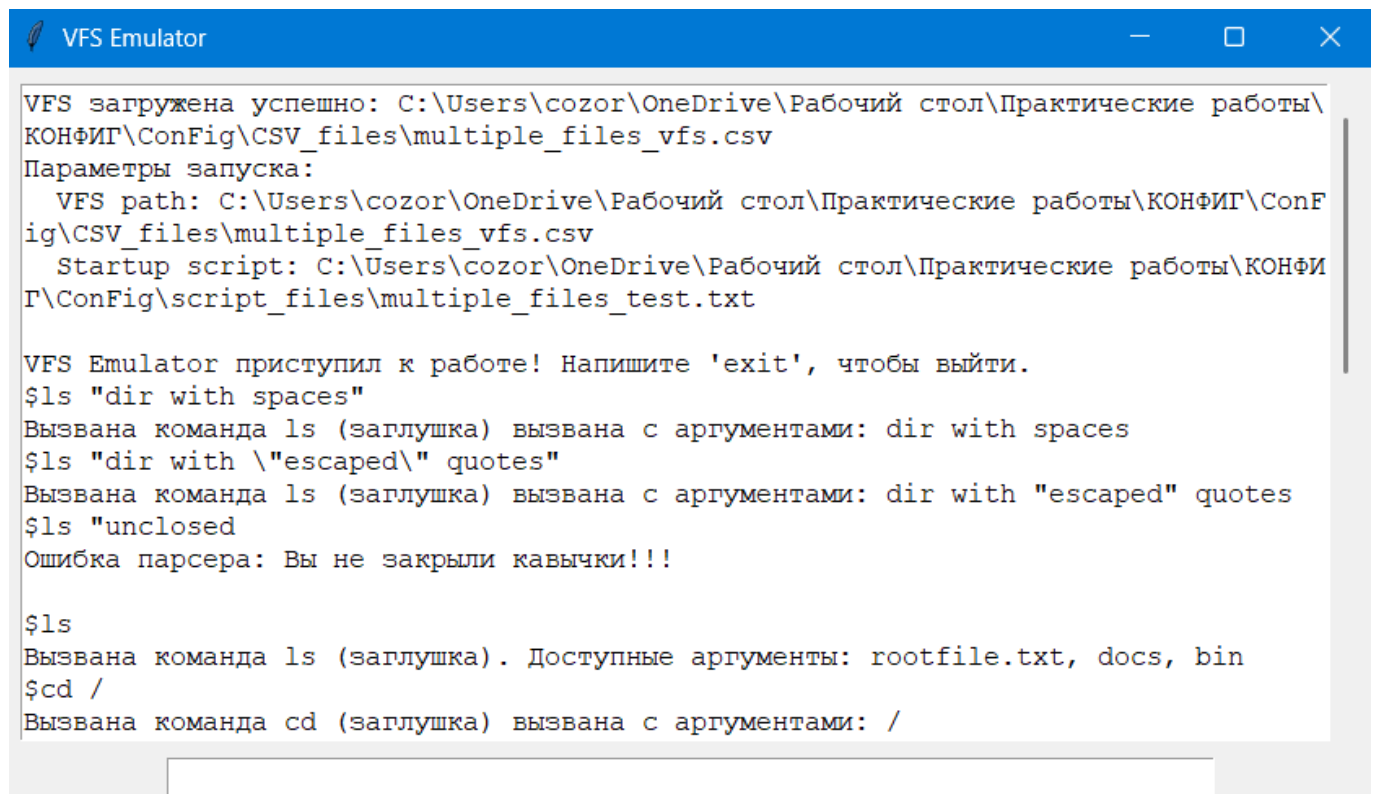


VFS Emulator

```
$ls "dir1"
Вызвана команда ls (заглушка) вызвана с аргументами: dir1
$ls /dir1
Вызвана команда ls (заглушка) вызвана с аргументами: /dir1
$cd dir1
Вызвана команда cd (заглушка) вызвана с аргументами: dir1
$ls
Вызвана команда ls (заглушка). Доступные аргументы: file1.txt, dir1
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$ls file1.txt
Вызвана команда ls (заглушка) вызвана с аргументами: file1.txt
$cd nonexist
Вызвана команда cd (заглушка) вызвана с аргументами: nonexist
$ls "unclosed quote
Ошибка парсера: Вы не закрыли кавычки!!!

Стартовый скрипт завершён с ошибками.
```

2 окно:



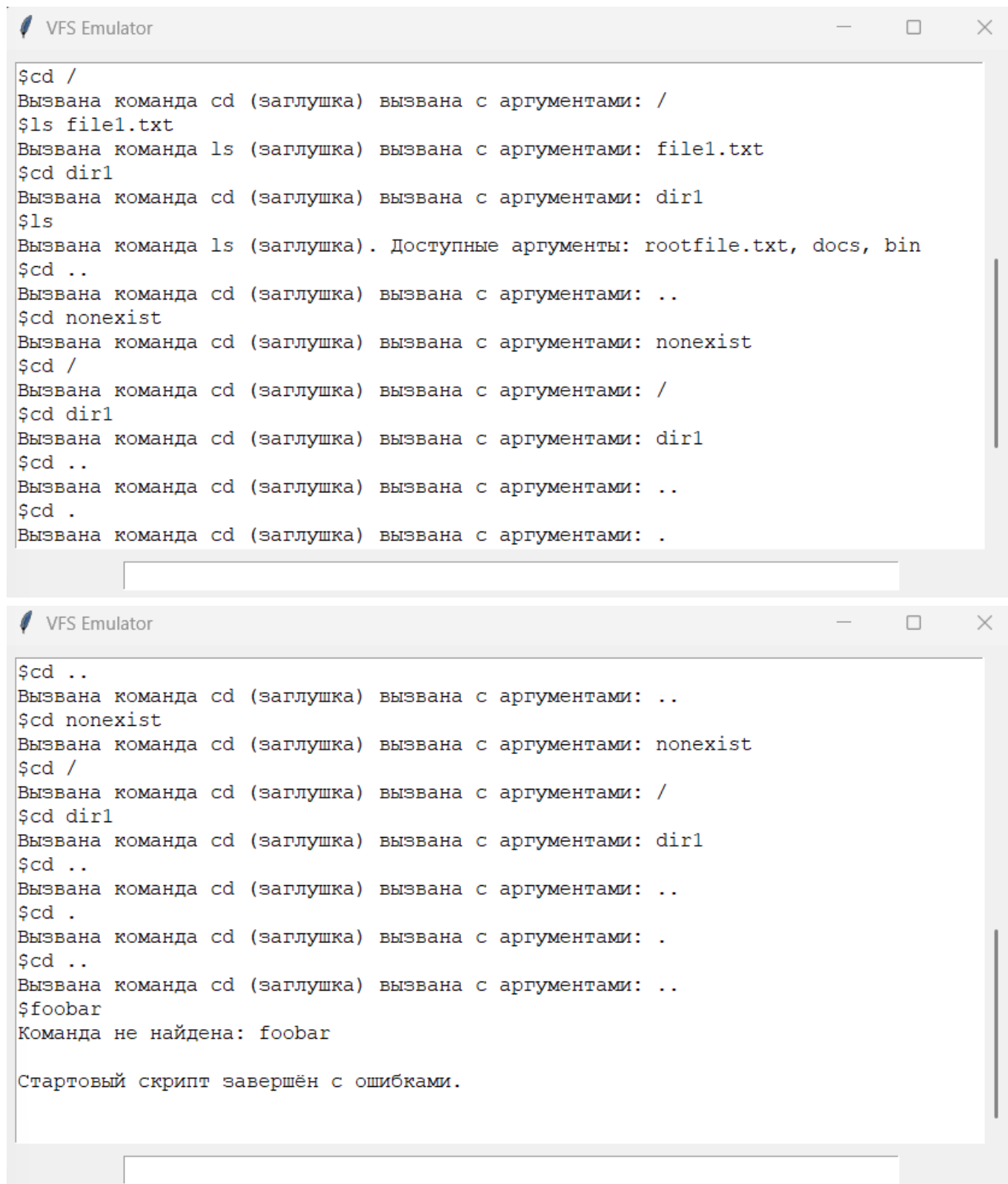
The screenshot shows a window titled "VFS Emulator" with a blue title bar. The main area contains text output from the emulator, including successful loading of a configuration file, startup parameters, and a series of test commands and their results. The text is as follows:

```
VFS загружена успешно: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\
КОНФИГ\ConFig\CSV_files\multiple_files_vfs.csv
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
ig\CSV_files\multiple_files_vfs.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\multiple_files_test.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$ls "dir with spaces"
Вызвана команда ls (заглушка) вызвана с аргументами: dir with spaces
$ls "dir with \"escaped\" quotes"
Вызвана команда ls (заглушка) вызвана с аргументами: dir with "escaped" quotes
$ls "unclosed
Ошибка парсера: Вы не закрыли кавычки!!!

$ls
Вызвана команда ls (заглушка). Доступные аргументы: rootfile.txt, docs, bin
$cd /
Вызвана команда cd (заглушка) вызвана с аргументами: /
```

At the bottom of the window, there is an empty input field for the user to type a command.



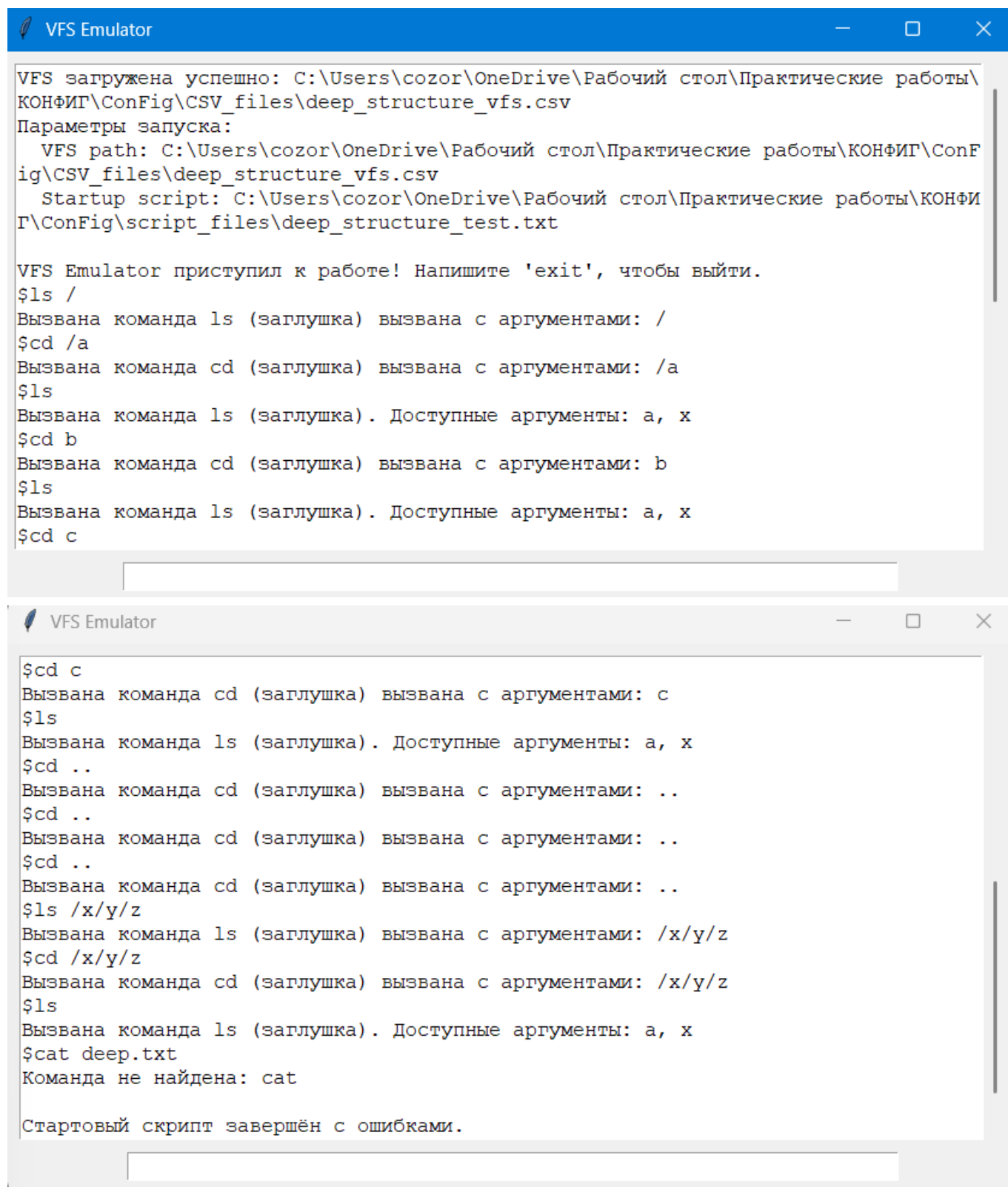
The image shows two screenshots of a terminal window titled "VFS Emulator". The terminal displays a series of commands and their corresponding system messages in Russian. The first screenshot shows commands like `$cd /`, `$ls file1.txt`, `$cd dir1`, `$ls`, `$cd ..`, `$cd nonexistent`, and `$cd /`. The second screenshot continues with `$cd dir1`, `$cd ..`, `$cd .`, and `$foobar`, followed by a message indicating the script has completed with errors.

```
VFS Emulator
$cd /
Вызвана команда cd (заглушка) вызвана с аргументами: /
$ls file1.txt
Вызвана команда ls (заглушка) вызвана с аргументами: file1.txt
$cd dir1
Вызвана команда cd (заглушка) вызвана с аргументами: dir1
$ls
Вызвана команда ls (заглушка). Доступные аргументы: rootfile.txt, docs, bin
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$cd nonexistent
Вызвана команда cd (заглушка) вызвана с аргументами: nonexistent
$cd /
Вызвана команда cd (заглушка) вызвана с аргументами: /
$cd dir1
Вызвана команда cd (заглушка) вызвана с аргументами: dir1
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$cd .
Вызвана команда cd (заглушка) вызвана с аргументами: .

VFS Emulator
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$cd nonexistent
Вызвана команда cd (заглушка) вызвана с аргументами: nonexistent
$cd /
Вызвана команда cd (заглушка) вызвана с аргументами: /
$cd dir1
Вызвана команда cd (заглушка) вызвана с аргументами: dir1
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$cd .
Вызвана команда cd (заглушка) вызвана с аргументами: .
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$foobar
Команда не найдена: foobar

Стартовый скрипт завершён с ошибками.
```

3 окно:



The image shows two screenshots of a 'VFS Emulator' window. The top screenshot shows the initial startup sequence: the VFS is loaded successfully from a specific path, parameters are displayed (VFS path and startup script), and the emulator begins operation. It then shows a series of commands being entered and executed: '\$ls /', '\$cd /a', '\$ls', '\$cd b', '\$ls', and '\$cd c'. The bottom screenshot continues the command sequence: '\$cd c', '\$ls', '\$cd ..', '\$cd ..', '\$cd ..', '\$ls /x/y/z', '\$cd /x/y/z', '\$ls', and '\$cat deep.txt'. It concludes with the message 'Стартовый скрипт завершён с ошибками.' (Startup script completed with errors).

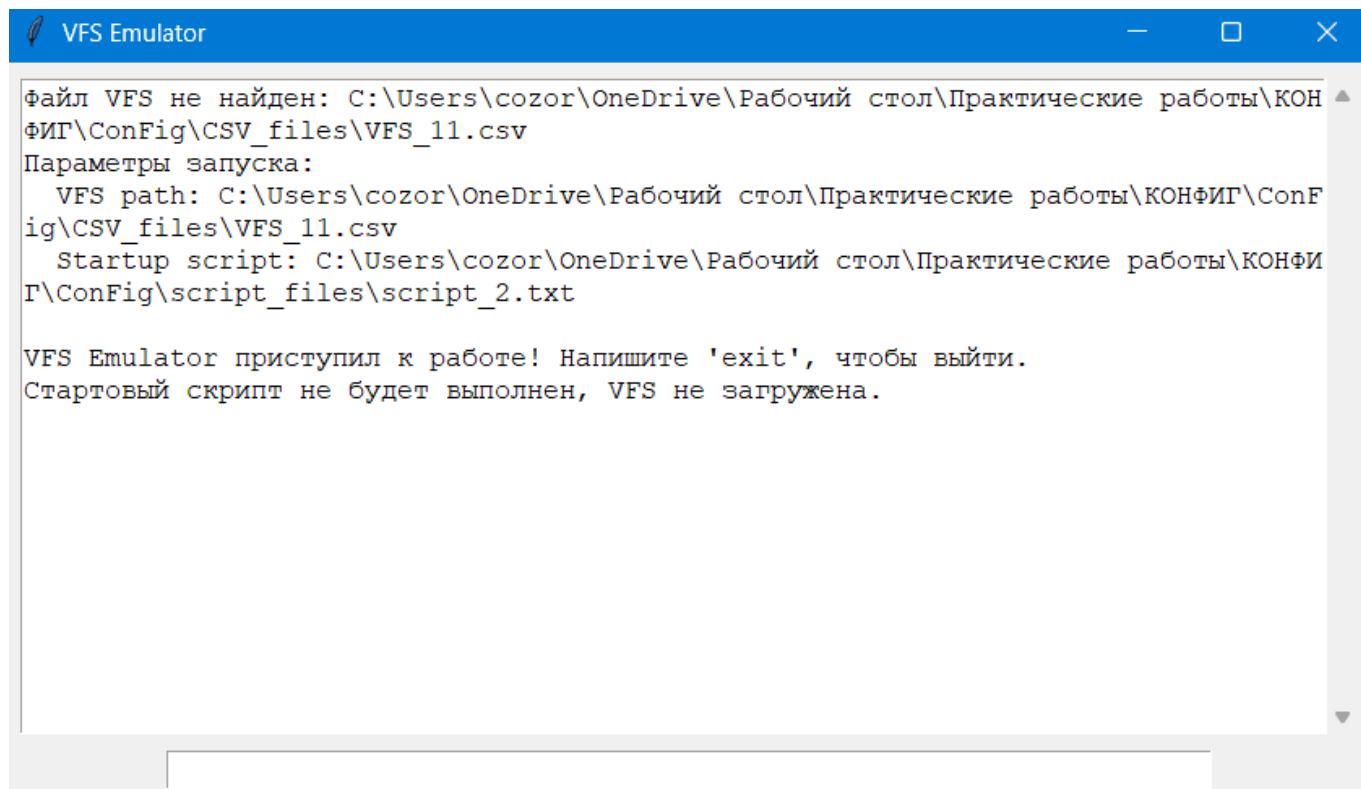
```
VFS Emulator
VFS загружена успешно: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\
КОНФИГ\ConFig\CSV_files\deep_structure_vfs.csv
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
ig\CSV_files\deep_structure_vfs.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\deep_structure_test.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$ls /
Вызвана команда ls (заглушка) вызвана с аргументами: /
$cd /a
Вызвана команда cd (заглушка) вызвана с аргументами: /a
$ls
Вызвана команда ls (заглушка). Доступные аргументы: a, x
$cd b
Вызвана команда cd (заглушка) вызвана с аргументами: b
$ls
Вызвана команда ls (заглушка). Доступные аргументы: a, x
$cd c

$cd c
Вызвана команда cd (заглушка) вызвана с аргументами: c
$ls
Вызвана команда ls (заглушка). Доступные аргументы: a, x
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$cd ..
Вызвана команда cd (заглушка) вызвана с аргументами: ..
$ls /x/y/z
Вызвана команда ls (заглушка) вызвана с аргументами: /x/y/z
$cd /x/y/z
Вызвана команда cd (заглушка) вызвана с аргументами: /x/y/z
$ls
Вызвана команда ls (заглушка). Доступные аргументы: a, x
$cat deep.txt
Команда не найдена: cat

Стартовый скрипт завершён с ошибками.
```

Также файл test3.bat показывает, что, если путь VFS указан неправильно, то выведется сообщение об ошибке и программа работать не будет.



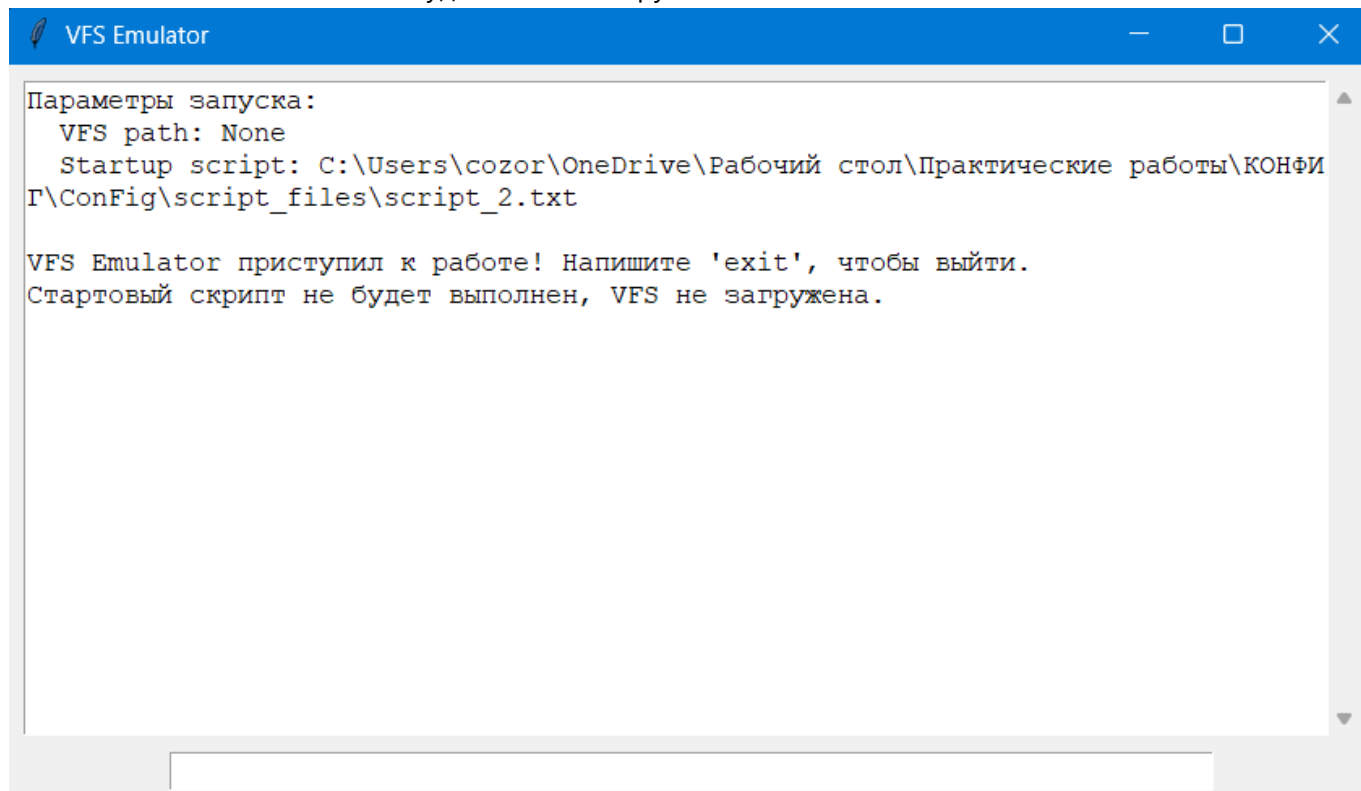
A screenshot of a Windows application window titled "VFS Emulator". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is a text box with a light gray background and a vertical scrollbar on the right. The text inside the text box is as follows:

```
файл VFS не найден: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОН
ФИГ\ConFig\CSV_files\VFS_11.csv
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
ig\CSV_files\VFS_11.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\script_2.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
Стартовый скрипт не будет выполнен, VFS не загружена.
```

At the bottom of the window, there is a small, empty rectangular input field.

Файл test2.bat показывает, что будет, если не загрузить VFS:



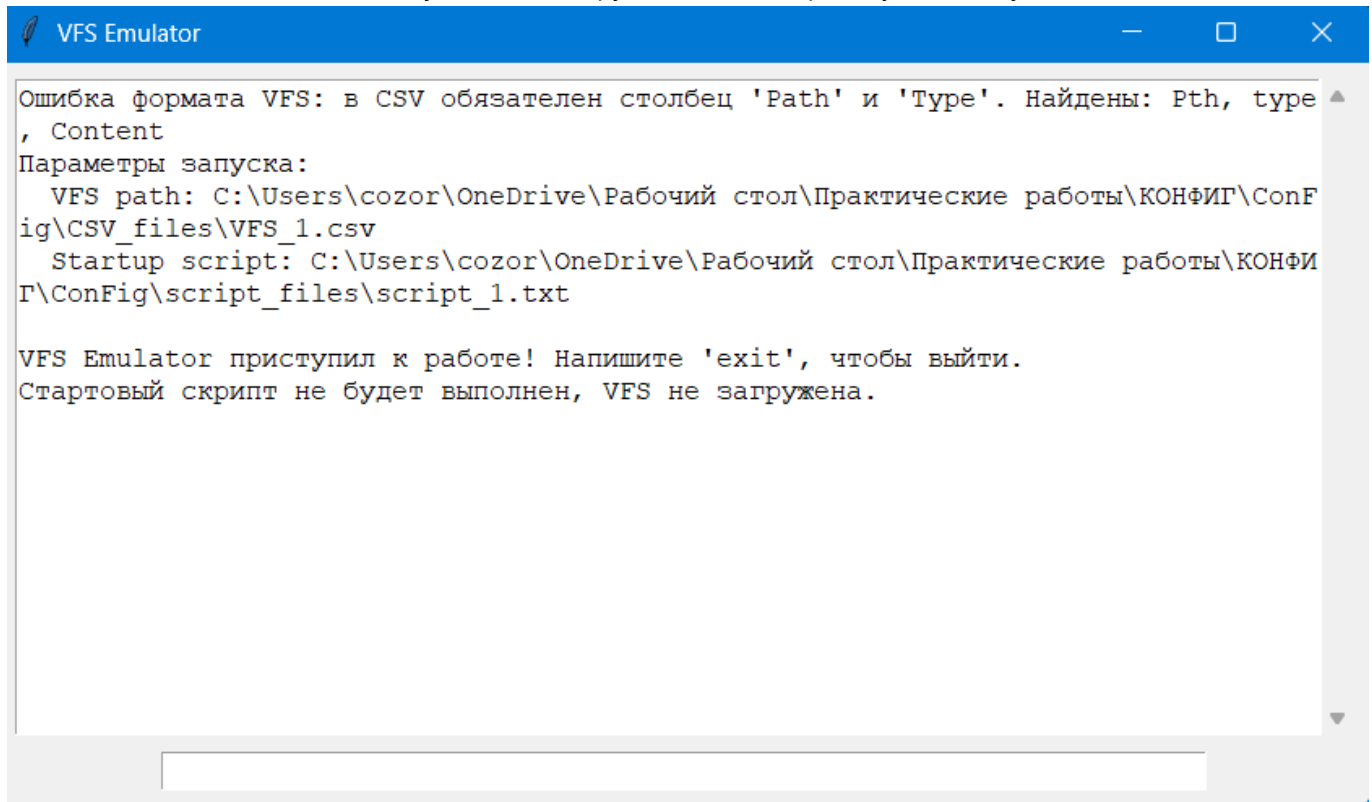
A screenshot of a Windows application window titled "VFS Emulator". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is a text box with a light gray background and a vertical scrollbar on the right. The text inside the text box is as follows:

```
Параметры запуска:
  VFS path: None
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\script_2.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
Стартовый скрипт не будет выполнен, VFS не загружена.
```

At the bottom of the window, there is a small, empty rectangular input field.

Файл test4.bat показывает, что будет, если загрузить VFS, содержащую ошибку:



The screenshot shows a window titled "VFS Emulator". The text inside the window is as follows:

```
Ошибка формата VFS: в CSV обязателен столбец 'Path' и 'Type'. Найдены: Pth, type
, Content
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
ig\CSV_files\VFS_1.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\script_1.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
Стартовый скрипт не будет выполнен, VFS не загружена.
```

Этап 4 Цель: поддержать команды, имитирующие работу в UNIX-подобной командной строке. 1 требование: Необходимо реализовать логику для ls и cd. Реализация: Был изменён функционал методов CMDls и CMDcd. Теперь это не просто заглушки, а полноценные команды, имитирующие работу в UNIX-подобной командной строке. Назначение команды ls - это вывести список содержимого текущей директории или указанного пути. Назначение команды cd - изменить текущую директорию: при вводе без аргументов возвращает пользователя в корень VFS, Если указано .., поднимается на уровень выше, Для других путей используется ResolvePath() — если найден узел-директория, текущая директория (CurrentDir) меняется на него.

```
def CMDls(self, Args):
    if not self.CurrentDir:
        return "VFS не загружена. Невозможно выполнить ls."
    DirToList = self.CurrentDir
    if Args:
        target = Args[0].strip()
        node = self.ResolvePath(self.CurrentDir, target)
        if not node:
            return f"Нет такой директории: {target}"
        if node.Type != "directory":
            return f"{target} не является директорией"
        DirToList = node
    ChildNames = []
    for ChildNode in DirToList.ListChildren():
        ChildNames.append(ChildNode.Name)
    return " ".join(ChildNames) if ChildNames else "(пусто)"
```

```
def CMDcd(self, Args):
    if not self.CurrentDir:
        return "VFS не загружена. Невозможно выполнить cd."
    if not Args:
        self.CurrentDir = self.VFSRoot
        return
    Path = Args[0].strip()
    if Path == "..":
        if self.CurrentDir.Parent:
            self.CurrentDir = self.CurrentDir.Parent
        return
    Node = self.ResolvePath(self.CurrentDir, Path)
    if not Node:
        return f"Нет такой директории: {Path}"
    if Node.Type != "directory":
        return f"{Path} не является директорией"
    self.CurrentDir = Node
```

2 требование: Реализовать новые команды: who, cat, tac. Реализация: Команда who выводит имя текущего пользователя операционной системы. Она пробует получить имя пользователя через `os.getlogin()`, если это не удаётся, например, при запуске в среде без терминала, используется переменные окружения `USER` или `USERNAME`. При отсутствии всех значений возвращается `user`.

```
def CMDwho(self, Args):
    try:
        user = os.getlogin()
    except Exception:
        user = os.environ.get("USER") or os.environ.get("USERNAME") or "user"
    return user
```

Команда `cat` выводит содержимое файла в текстовом виде. Сначала проверяется наличие аргумента, то есть путь к файлу. Путь разрешается функцией `ResolvePath()`. Если узел существует и имеет тип `file`, содержимое декодируется из `utf-8`. При ошибках декодирования символы заменяются, чтобы не прерывать работу. Полученный текст возвращается пользователю. Если путь не существует, то выводится сообщение «Нет такого файла». Если указан каталог — сообщение «... не является файлом».

```
def CMDcat(self, Args):
    if not Args:
        return "Использование: cat <путь>"
    path = Args[0]
    node = self.ResolvePath(self.CurrentDir, path)
    if not node:
        return f"Нет такого файла: {path}"
    if node.Type != "file":
        return f"{path} не является файлом"
```



```

try:
    text = node.Content.decode('utf-8')
except Exception:
    text = node.Content.decode('utf-8', errors='replace')
return text

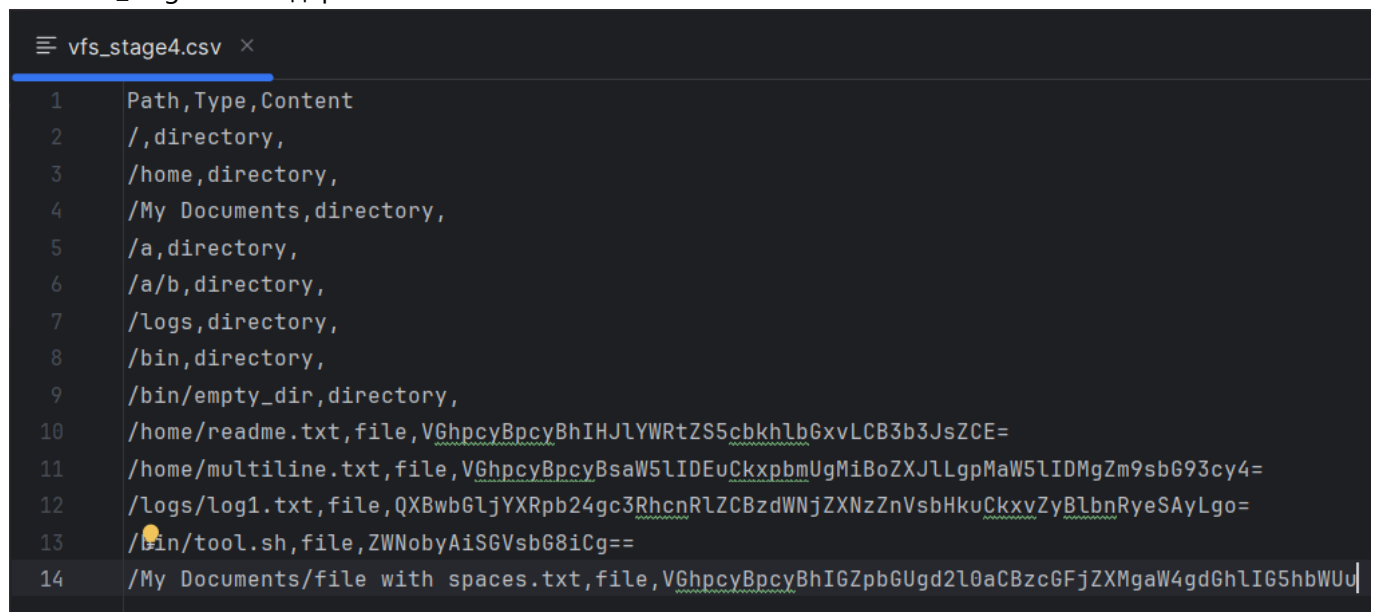
```

Команда tac отображает файл построчно в обратном порядке. Сначала загружает и декодирует содержимое файла аналогично cat. Потом разделяет текст на строки, переворачивает их и объединяет снова. Возвращает перевёрнутый текст. def CMDtac(self, Args): if not Args: return "Использование: tac <путь>" path = Args[0] node = self.ResolvePath(self.CurrentDir, path) if not node: return f"Нет такого файла: {path}" if node.Type != "file": return f"{path} не является файлом" text = node.Content.decode('utf-8', errors='replace') lines = text.splitlines() return "\n".join(reversed(lines))

3 требование: Создать стартовый скрипт для тестирования всех реализованных на этом этапе команд. Добавить туда примеры всех режимов команд, включая работу с VFS и обработку ошибок.

Реализация: были созданы новые файлы для тестирования всех существующих на этом этапе команд и их обработка ошибок.

Файл vfs_stage4.csv содержит:



```

1 Path,Type,Content
2 /,directory,
3 /home,directory,
4 /My Documents,directory,
5 /a,directory,
6 /a/b,directory,
7 /logs,directory,
8 /bin,directory,
9 /bin/empty_dir,directory,
10 /home/readme.txt,file,VGhpcyBpcyBhIHJlYWRTZSS5cbkhlbGxvLCB3b3JsZCE=
11 /home/multiline.txt,file,VGhpcyBpcyBsaW5lIDEuCkxpbmUgMiBoZXJlLgpMaW5lIDMgZm9sbG93cy4=
12 /logs/log1.txt,file,QXBwbGljYXRpb24gc3RhcnRlZCBzdWNjZXNzZnVsbHkuCkxvZyBlbnRyeSAyLgo=
13 /bin/tool.sh,file,ZWNobyAiSGVsbG8iCg==
14 /My Documents/file with spaces.txt,file,VGhpcyBpcyBhIGZpbGUgd2l0aCBzcGFjZXMGaW4gdGhlIG5hbWUu

```

Файл script_stage4.txt содержит:

```
≡ vfs_stage4.csv    ≡ script_stage4.txt ×
1  # базовые команды
2  who
3  ls
4
5  # ls: корень и подкаталоги (абсолютные и с кавычками)
6  ls /
7  ls /home
8  ls "/My Documents"
9  ls /home/readme.txt
10
11 # ls пустой директории
12 ls /bin
13 ls /bin/empty_dir
14
15 # cd переходы (абсолютный, относительный, .., переход в корень и без аргумента)
16 cd /home
17 ls
18 cd ..
19 ls
20 cd
21
22 # Перейти в глубоко вложенный каталог (абсолютный и относительный)
23 cd /a/b
24 ls
25 cd ..
26 cd /
27 ls /a/b
28
29 # Попытка cd в файл (ожидаем ошибку)
30 cd /bin/tool.sh
```

```
≡ vfs_stage4.csv    ≡ script_stage4.txt ×

31
32 # cd .. в корне (должно оставаться в /)
33 cd /
34 cd ..
35
36 # cat: просмотр файлов (правильный и ошибочные случаи)
37 cat /home/readme.txt
38 cat /home/multiline.txt
39
40 # cat на директории (ошибка)
41 cat /home
42
43 # cat несуществующего файла (ошибка)
44 cat /no/such/file.txt
45
46 # cat нескольких файлов подряд
47 cat /home/readme.txt /home/multiline.txt
48
49 # tac: перевернутый по строкам вывод
50 tac /home/multiline.txt
51 tac /logs/log1.txt
52
53 # ls и cd с относительными путями и специальными точками
54 cd /home
55 ls
56 cd .
57 ls
58 cd ..
59 ls
60
```

```
61 # Обработка кавычек и экранирования аргументов
62 ls "My Documents"
63 ls "My \"Documents\""
64 ls "My\\ Documents"
65
66 # Неизвестная команда
67 foobar
68
69 # незакрытая кавычка (парсерная ошибка)
70 ls "unclosed quote
```

В данном

скрипте нет команды exit, так как она сразу же закроет окно после выполнения. Если не нужно посмотреть, как работают команды, то можно вписать в конец exit.

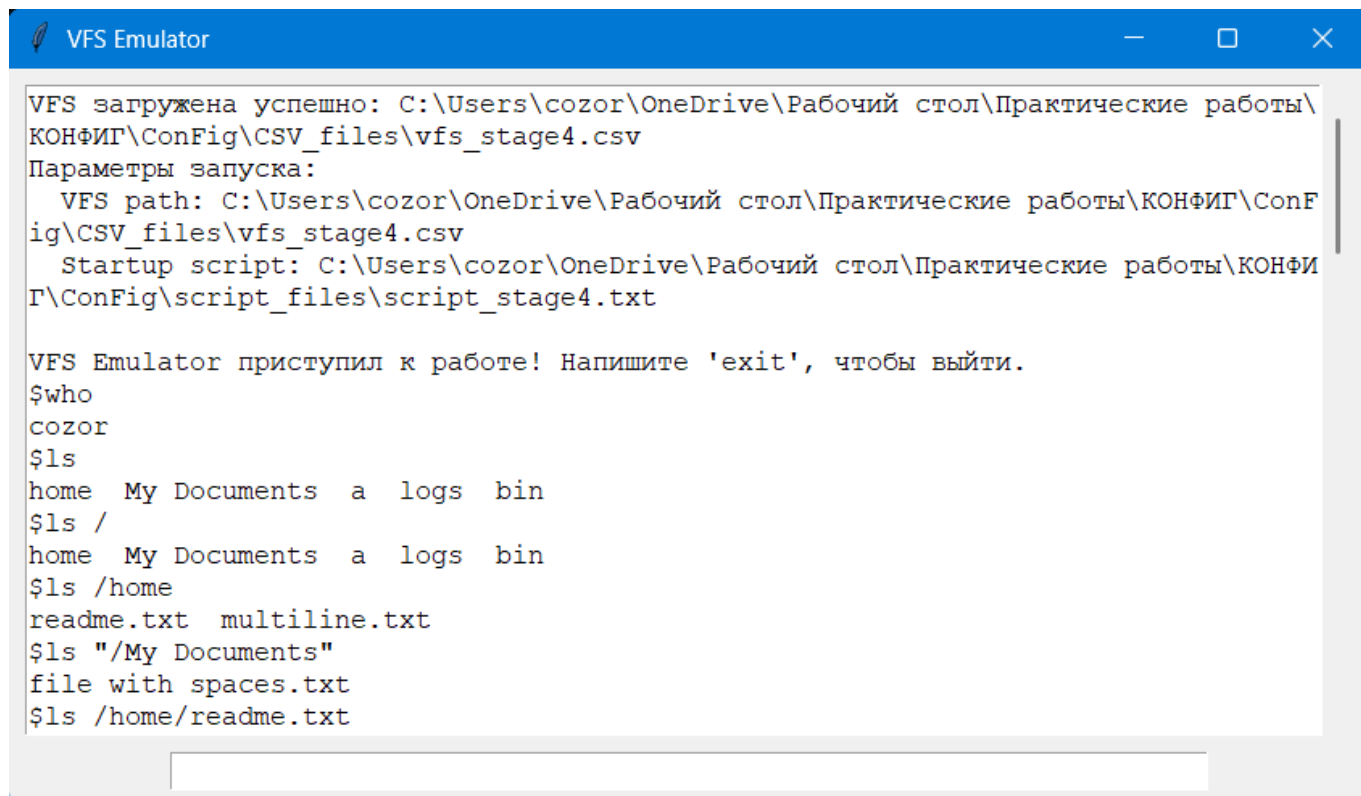
Файл test_stage4.bat содержит:

```

vfs_stage4.csv  script_stage4.txt  test_stage4.bat  x
1  @echo off
2  chcp 65001
3  REM Тест с глубокой структурой
4  python ..\vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\CSV_files\vfs_stage4.csv" --script "C:\Users
5  pause

stage4.txt  test_stage4.bat  x
--script "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConFig\script_files\script_stage4.txt"
```

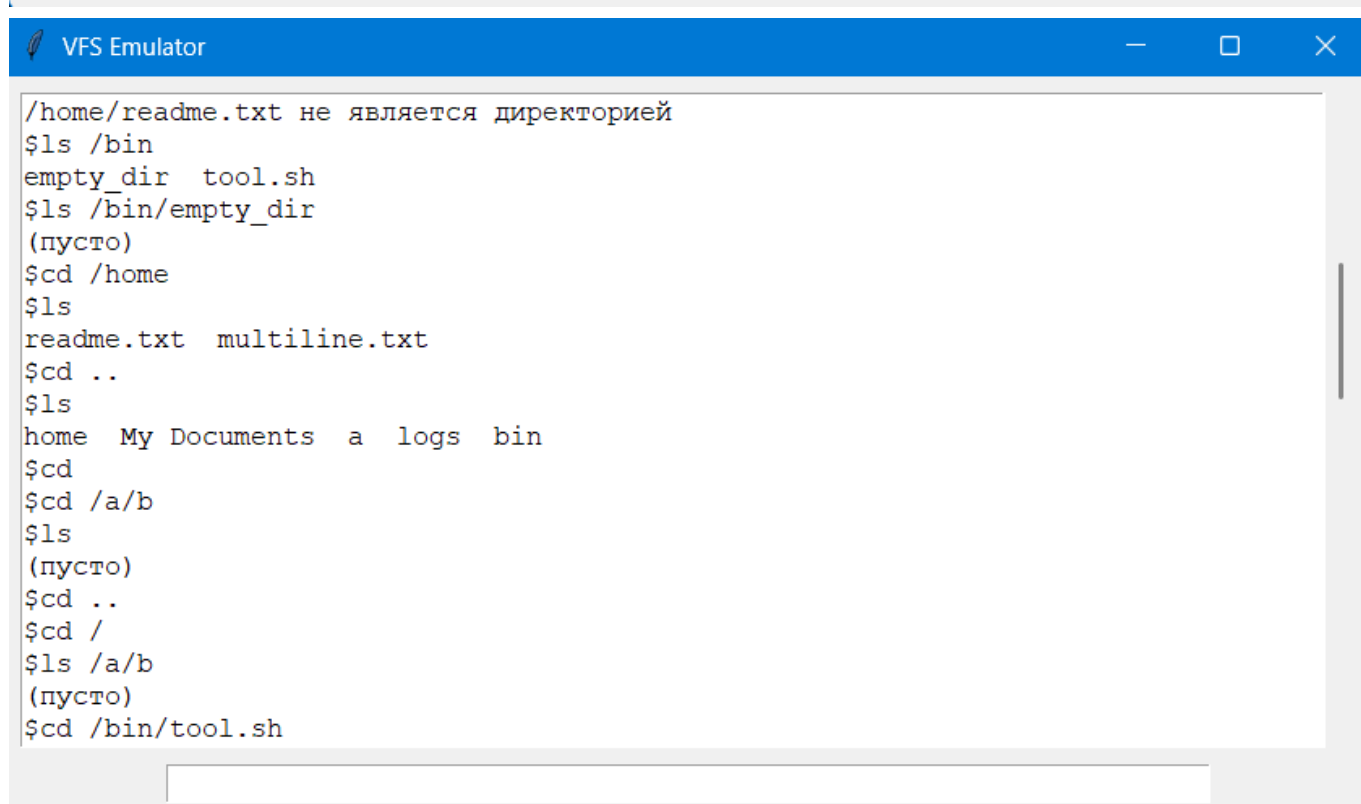
Результат работы файла:



VFS Emulator

```
VFS загружена успешно: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\
КОНФИГ\ConFig\CSV_files\vfs_stage4.csv
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
ig\CSV_files\vfs_stage4.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\script_stage4.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$who
cozor
$ls
home  My Documents  a  logs  bin
$ls /
home  My Documents  a  logs  bin
$ls /home
readme.txt  multiline.txt
$ls "/My Documents"
file with spaces.txt
$ls /home/readme.txt
```



VFS Emulator

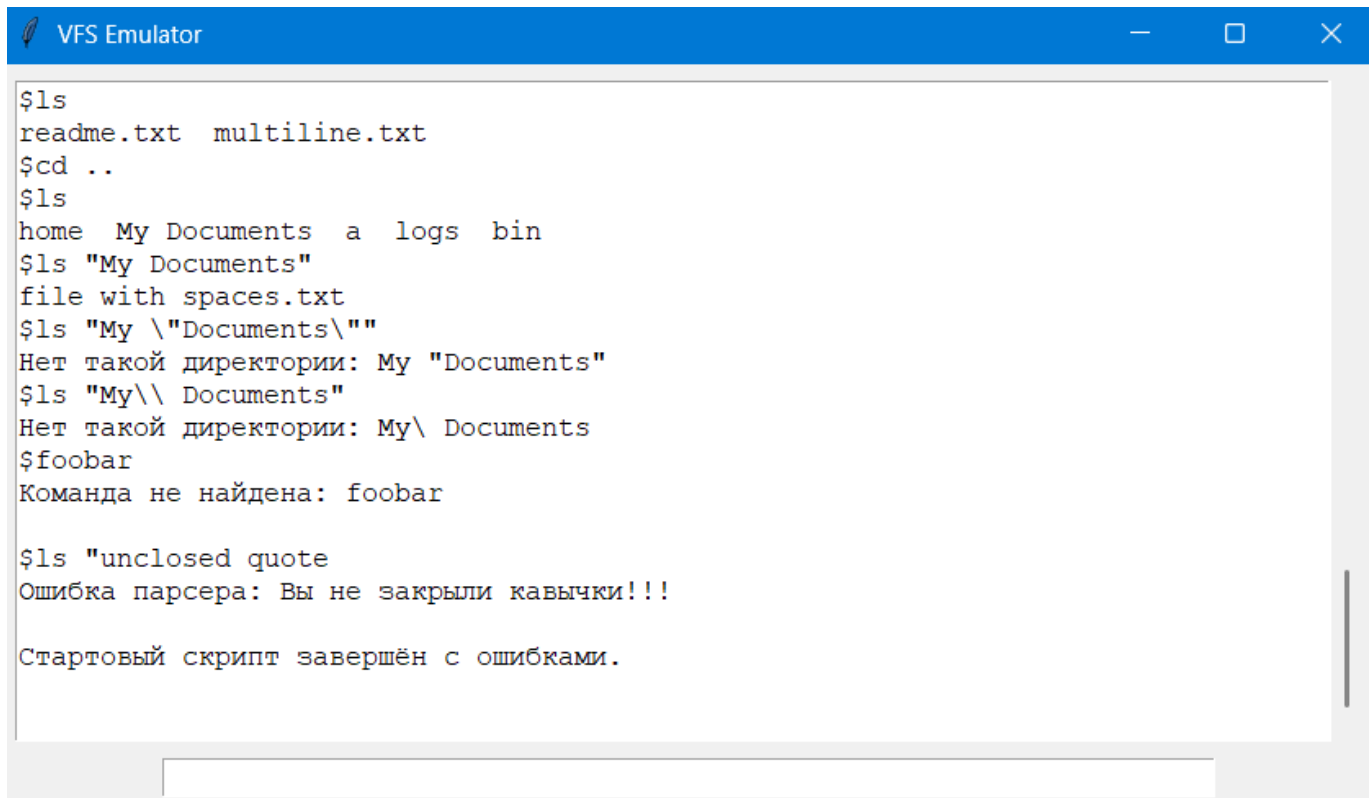
```
/home/readme.txt не является директорией
$ls /bin
empty_dir  tool.sh
$ls /bin/empty_dir
(пусто)
$cd /home
$ls
readme.txt  multiline.txt
$cd ..
$ls
home  My Documents  a  logs  bin
$cd
$cd /a/b
$ls
(пусто)
$cd ..
$cd /
$ls /a/b
(пусто)
$cd /bin/tool.sh
```

```
VFS Emulator

/bin/tool.sh не является директорией
$cd /
$cd ..
$cat /home/readme.txt
This is a readme.\nHello, world!
$cat /home/multiline.txt
This is line 1.
Line 2 here.
Line 3 follows.
$cat /home
/home не является файлом
$cat /no/such/file.txt
Нет такого файла: /no/such/file.txt
$cat /home/readme.txt /home/multiline.txt
This is a readme.\nHello, world!
$tac /home/multiline.txt
Line 3 follows.
Line 2 here.
This is line 1.
$tac /logs/log1.txt
```

```
VFS Emulator

Log entry 2.
Application started successfully.
$cd /home
$ls
readme.txt  multiline.txt
$cd .
$ls
readme.txt  multiline.txt
$cd ..
$ls
home  My Documents  a  logs  bin
$ls "My Documents"
file with spaces.txt
$ls "My \"Documents\""
Нет такой директории: My "Documents"
$ls "My\\ Documents"
Нет такой директории: My\ Documents
$foobar
Команда не найдена: foobar
```



```
VFS Emulator

$ls
readme.txt  multiline.txt
$cd ..
$ls
home  My Documents  a  logs  bin
$ls "My Documents"
file with spaces.txt
$ls "My \"Documents\""
Нет такой директории: My "Documents"
$ls "My\\ Documents"
Нет такой директории: My\ Documents
$foobar
Команда не найдена: foobar

$ls "unclosed quote
Ошибка парсера: Вы не закрыли кавычки!!!

Стартовый скрипт завершён с ошибками.
```

5 этап:

Цель: поддерживать более сложные команды, изменяющие состояние VFS, при этом модификации должны осуществляться только в памяти.

1 Требование: Реализовать команды: mv. Реализация:

В класс VFSNode был добавлен новый метод, который удаляет дочерний узел с именем Name из словаря self.Children. Если узел найден, он удаляется и у него сбрасывается ссылка на родителя child.Parent = None. Если имя не найдено, то метод ничего не делает.

```
def RemoveChild(self, Name):
    child = self.Children.pop(Name, None)
    if child is not None:
        child.Parent = None
```

В свойство Commands класса VFSEmulator была добавлена новая команда:

```
self.Commands = {"ls": self.CMDls, "cd": self.CMDcd, "who": self.CMDwho, "cat": self.CMDcat, "tac":
self.CMDtac, "mv": self.CMDmv, "exit": self.CMDexit, }
```

Добавлен новый метод CMDmv, который реализует команду mv <источник> <назначение>. Эта команда перемещает или переименовывает файл или директорию внутри виртуальной структуры. Изменения происходят только в памяти, в объектной модели VFSNode.

Как работает метод:

1. Проверяет, что передано ровно 2 аргумента, иначе возвращает подсказку по использованию.

2. Разрешает путь источника (ResolvePath), если узел не найден: ошибка. 3. Блокирует попытку переместить корень VFS.
3. Пытается разрешить путь назначения: если DestPath указывает на существующую директорию: целевой родитель = эта директория, новое имя = старое. В ином случае разбирает DestPath на родительский путь и базовое имя (новое имя), разрешает родительский путь (может быть . — текущая директория).
4. Проверяет, что TargetParent существует и является директорией, иначе: ошибка.
5. Защищает от перемещения директории внутрь её собственного потомка: идёт по родителям TargetParent вверх и, если встретит SourceNode, возвращает ошибку.
6. Если в целевой директории уже есть узел с таким именем, то удаляет его.
7. Удаляет SourceNode из старого родителя, присваивает ему новое имя и добавляет в TargetParent, корректно устанавливая ссылку Parent.
8. Формирует абсолютный путь назначения и возвращает сообщение об успешном перемещении.

```
def CMDmv(self, Args): if len(Args) != 2: return "Использование: mv <источник> <назначение> "
```

```
SourcePath = Args[0]
DestPath = Args[1]

SourceNode = self.ResolvePath(self.CurrentDir, SourcePath)
if not SourceNode:
    return f"Нет такого файла или директории: {SourcePath}"
if SourceNode == self.VFSRoot:
    return "Нельзя перемещать корень VFS"

DestNode = self.ResolvePath(self.CurrentDir, DestPath)

if DestNode and DestNode.Type == "directory":
    TargetParent = DestNode
    NewName = SourceNode.Name
else:
    DestString = DestPath.strip()
    if DestString.endswith("/") and DestString != "/":
        DestString = DestString.rstrip("/")

    if DestString.startswith("/"):
        ParentString = (
            "/" + "/".join(p for p in DestString.strip("/").split("/")
[: -1])
            if "/" in DestString.strip("/")
            else "/"
        )
    else:
        Parts = [p for p in DestString.split("/") if p != ""]
        if len(Parts) <= 1:
```



```

        ParentString = "."
    else:
        ParentString = "/" .join(Parts[:-1])

    if "/" in DestString:
        BaseName = DestString.strip("/").split("/")[-1]
    else:
        BaseName = DestString

    if BaseName != "":
        NewName = BaseName
    else:
        NewName = SourceNode.Name

    if ParentString == ".":
        TargetParent = self.CurrentDir
    else:
        TargetParent = self.ResolvePath(self.CurrentDir, ParentString)

    if not TargetParent:
        return f"Родительский путь назначения не найден: {ParentString}"
    if TargetParent.Type != "directory":
        return f"Родитель назначения '{ParentString}' не является
директорией"

    CheckNode = TargetParent
    while CheckNode is not None:
        if CheckNode == SourceNode:
            return "Нельзя переместить директорию внутрь её потомка!"
        CheckNode = CheckNode.Parent

    ExistingNode = TargetParent.GetChild(NewName)
    if ExistingNode:
        TargetParent.RemoveChild(NewName)

    if SourceNode.Parent:
        SourceNode.Parent.RemoveChild(SourceNode.Name)

    SourceNode.Name = NewName
    TargetParent.AddChildren(SourceNode)

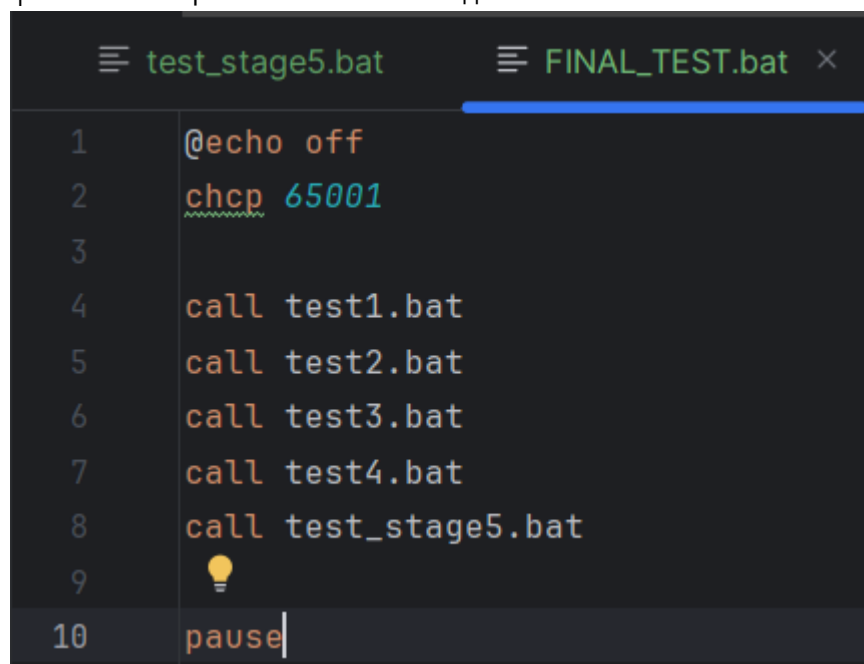
    FullDestPath = (
        TargetParent.Path()
        + ("/" if TargetParent.Path() != "/" else "")
        + NewName
    )

    return f"Перемещено: из '{SourcePath}' в '{FullDestPath}'"

```

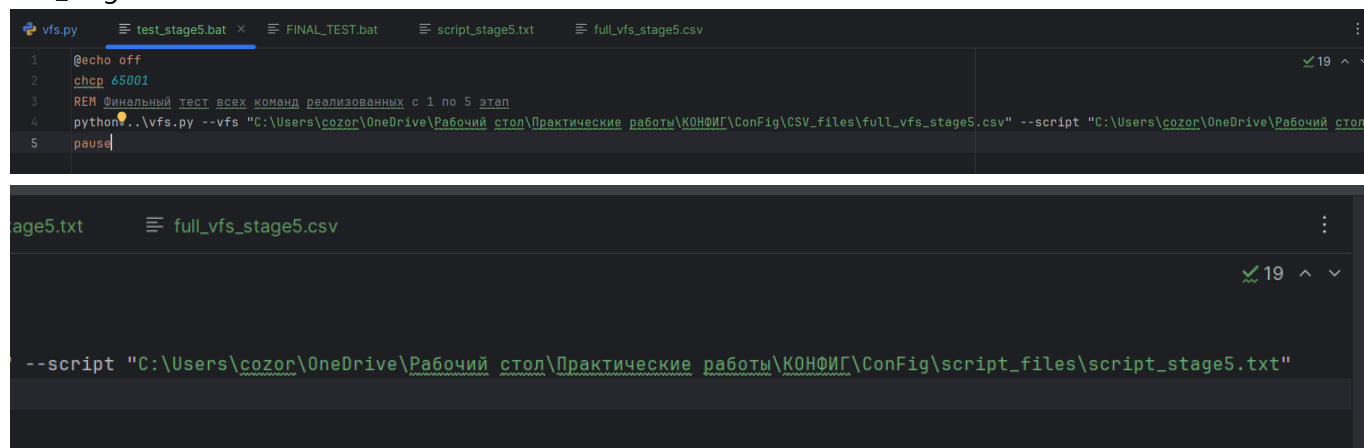
2 Требование: Создать стартовый скрипт для тестирования всех реализованных на этом этапе команд. Добавить туда примеры всех режимов команд, включая работу с VFS и обработку ошибок.

Файл FINAL_TEST.bat запускает файлы, которые покажут все реализованные команды на этом этапе. Первые 4 вызова показывают ошибки, которые могут появиться при загрузке VFS, 5 вызов запускает финальный скрипт со всеми командами.



```
test_stage5.bat
1 @echo off
2 chcp 65001
3
4 call test1.bat
5 call test2.bat
6 call test3.bat
7 call test4.bat
8 call test_stage5.bat
9
10 pause
```

test_stage5.bat:



```
test_stage5.bat
1 @echo off
2 chcp 65001
3 REM Финальный тест всех команд реализованных с 1 по 5 этап
4 python ..\vfs.py --vfs "C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\Config\CSV_files\full_vfs_stage5.csv" --script "C:\Users\cozor\OneDrive\Рабочий стол\script_files\script_stage5.txt"
5 pause
```

script_stage5.txt

```
vfs.py test_stage5.bat FINAL_TEST.bat script_stage5.txt x full_vfs_stage5.c
1 # базовые
2 who
3 ls
4
5 # ls: корень и подкаталоги (абсолютные и с кавычками)
6 ls /
7 ls /home
8 ls "/My Documents"
9 ls "/My Documents" /home
10
11 # ls на файле: ошибка "не является директорией"
12 ls /home/readme.txt
13
14 # ls пустой директории
15 ls /bin
16 ls /bin/empty_dir
17
18 # cd переходы (абсолютный, относительный, .., переход в корень и без аргумента)
19 cd /home
20 ls
21 cd ..
22 ls
23 cd
24
25 # перейти в глубоко вложенный каталог (абсолютный и относительный)
26 cd /a/b
27 ls
28 cd ..
29 cd /
30 ls /a/b
```

```
31
32 # попытка cd в файл: ошибка
33 cd /bin/tool.txt
34
35 # cat: просмотр файлов (правильный и ошибочные случаи)
36 cat /home/readme.txt
37 cat /home/multiline.txt
38
39 # cat на директории: ошибка
40 cat /home
41
42 # cat несуществующего файла: ошибка
43 cat /no/such/file.txt
44
45 # tac: перевёрнутый по строкам вывод
46 tac /home/multiline.txt
47 tac /logs/log1.txt
48
49 # ls и cd с относительными путями и специальными точками
50 cd /home
51 ls
52 cd .
53 ls
54 cd ..
55 ls
56
57 # обработка кавычек
58 ls "My Documents"
59
60 # обработка текстового файла
61 cat /bin/tool.txt
```

```
61 cat /bin/tool.txt
62
63 # неизвестная команда
64 command
65
66 # незакрытая кавычка: ошибка
67 ls "unclosed quote
68
69 # 5 этап
70 # исходное состояние
71 ls /
72 ls /home
73 ls /a
74 ls /a/b
75
76 # переместить файл в другую директорию (сохранить имя)
77 mv /home/readme.txt /a
78 ls /a
79 ls /home
80
81 # переименовать файл в той же директории
82 mv /a/readme.txt /a/readme_moved.txt
83 ls /a
84
85 # переместить вложенную директорию /a/b внутрь /home
86 mv /a/b /home
87 ls /home
88 ls /a
89
90 # попытка mv в несуществующий родитель: ошибка
91 mv /a/readme_moved.txt /no/such/place/newname.txt
92
```

```

92
93 # попытка mv корня: ошибка
94 mv / /tmp
95
96 # тест: перемещение директории внутрь её собственного потомка: ошибка
97 mv /x /x/y
98
99 # перемещение файла с пробелами в имени в /home
100 mv "/My Documents/file with spaces.txt" /home/file_with_spaces_copy.txt
101 ls /home
102
103 # перезапись существующего файла: переместим и переименуем в имя, которое уже есть в /home
104 mv /home/file_with_spaces_copy.txt /home/readme_moved.txt
105 ls /home
106
107 # переместить файл относительным путём, то есть сменим CurrentDir
108 cd /home
109 mv readme_moved.txt ../readme_back.txt
110 cd ..
111 ls /
112
113 # mv с ошибочным источником
114 mv /no/such /tmp
115
116 # Вывод финального состояния
117 ls /
118 ls /home
119 ls /a

```

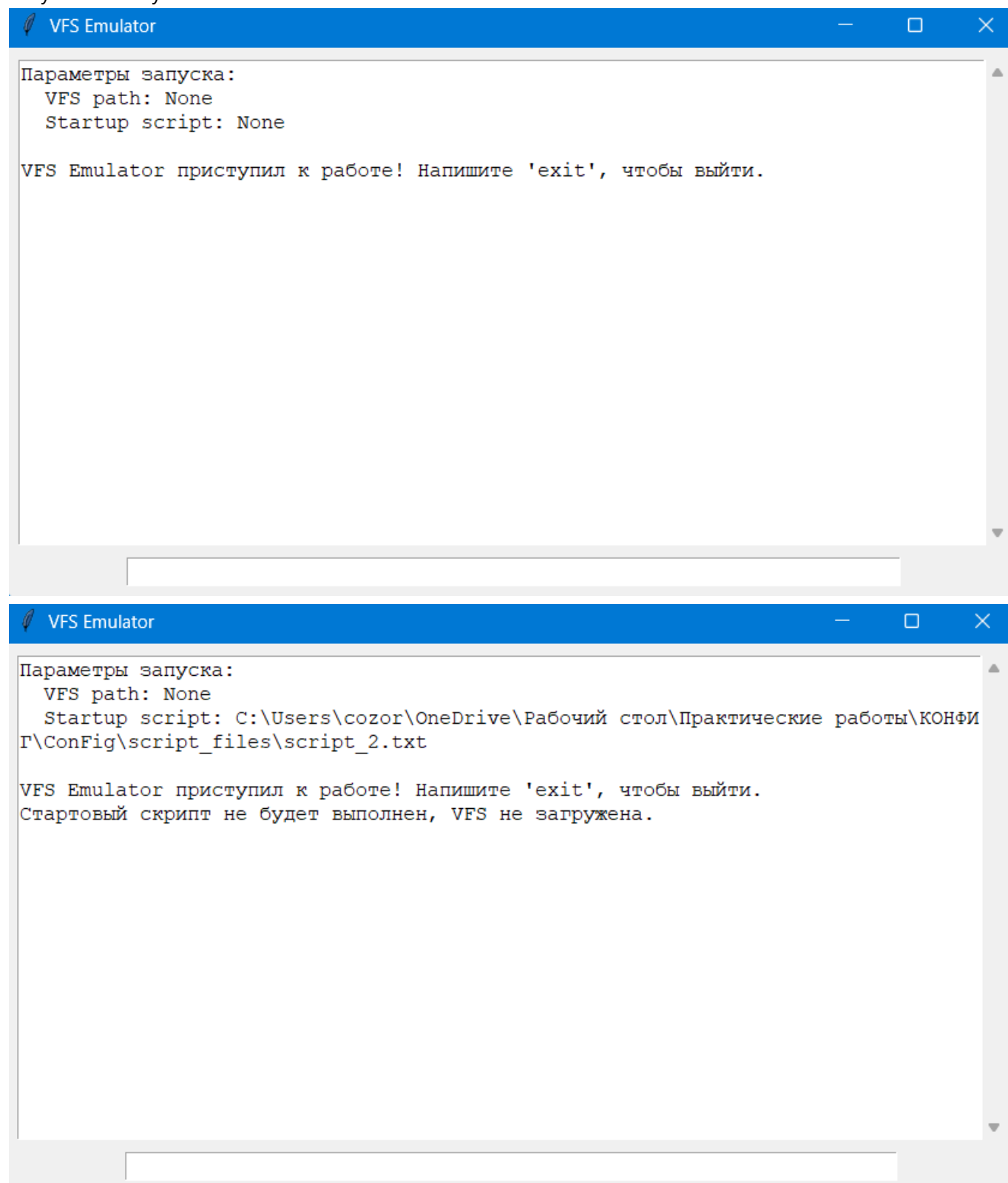
full_vfs_stage5.csv

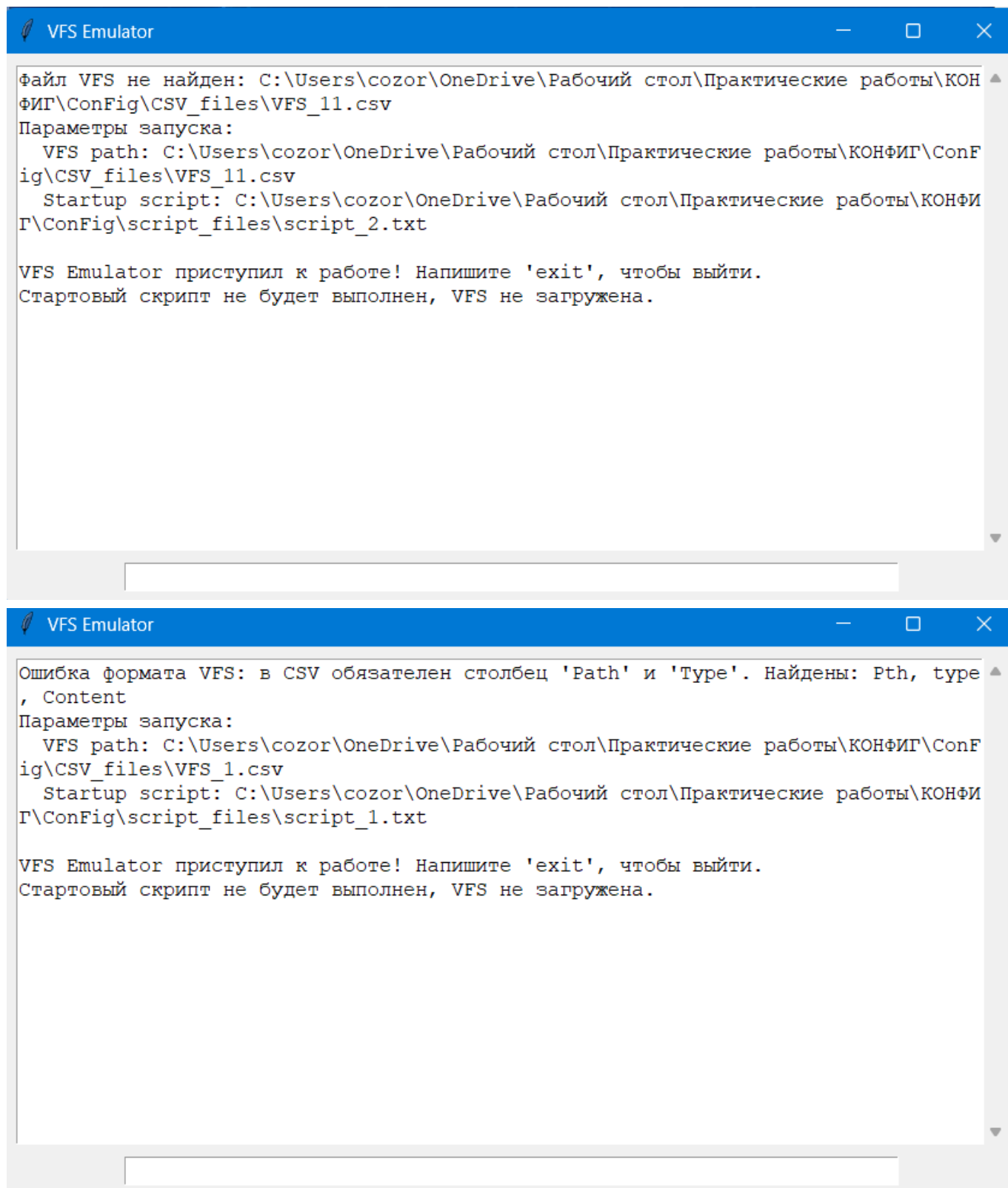
```

vfs.py  test_stage5.bat  FINAL_TEST.bat  script_stage5.txt  full_vfs_stage5.csv x
1 Path,Type,Content
2 /,directory,
3 /home,directory,
4 /My Documents,directory,
5 /a,directory,
6 /a/b,directory,
7 /x,directory,
8 /x/y,directory,
9 /logs,directory,
10 /bin,directory,
11 /bin/empty_dir,directory,
12 /home/readme.txt,file,VGhpcyBpcyBhIHJlYWRTZS5cbkh1bGxvLCB3b3JsZCE=
13 /home/readme_moved.txt,file,VGhpcyBpcyBhIHJlYWRTZS5cbkh1bGxvLCB3b3JsZCE=
14 /home/multiline.txt,file,VGhpcyBpcyBsaW5lIDEvCkxpbmUgMiBoZXJlLgpMaW5lIDMgZm9sbG93cy4=
15 /logs/log1.txt,file,QXBwbGljYXRpb24gc3RhcnRlZCBzdWNjZXNzZnVsbnRyeSAyLgo=
16 /bin/tool.txt,file,ZWNobyAiS6VsbG8iCg==
17 /My Documents/file with spaces.txt,file,VGhpcyBpcyBhIGZpbGUGd2l0aCBzcGFjZXMGaW4gdGhlIG5hbWUu

```

Результаты запуска:



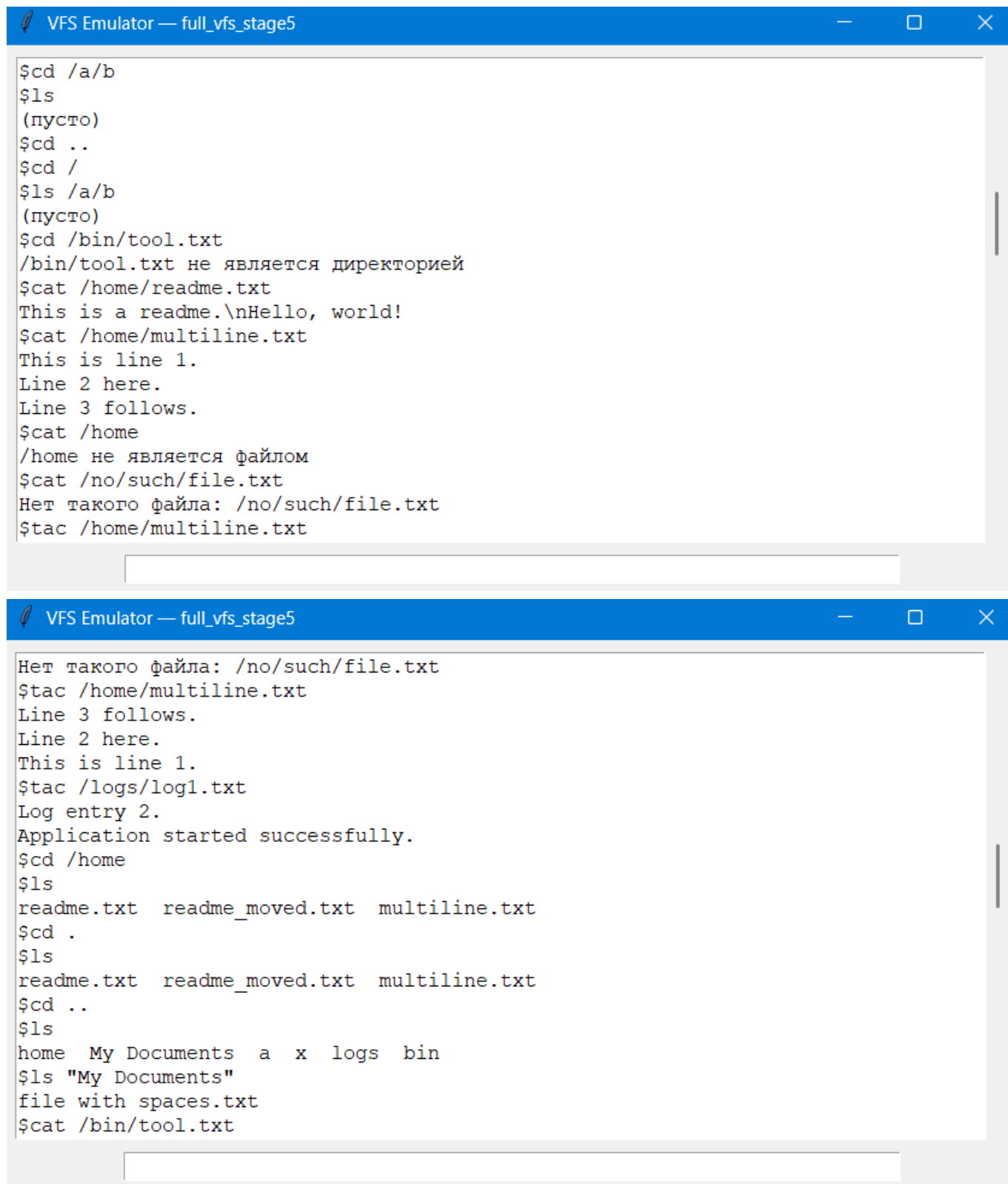



```
VFS Emulator — full_vfs_stage5

VFS загружена успешно: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\
КОНФИГ\ConFig\CSV_files\full_vfs_stage5.csv
Параметры запуска:
  VFS path: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИГ\ConF
ig\CSV_files\full_vfs_stage5.csv
  Startup script: C:\Users\cozor\OneDrive\Рабочий стол\Практические работы\КОНФИ
Г\ConFig\script_files\script_stage5.txt

VFS Emulator приступил к работе! Напишите 'exit', чтобы выйти.
$who
cozor
$ls
home  My Documents  a  x  logs  bin
$ls /
home  My Documents  a  x  logs  bin
$ls /home
readme.txt  readme_moved.txt  multiline.txt
$ls "/My Documents"
file with spaces.txt
$ls "/My Documents" /home

readme.txt  readme_moved.txt  multiline.txt
$ls "/My Documents"
file with spaces.txt
$ls "/My Documents" /home
file with spaces.txt
$ls /home/readme.txt
/home/readme.txt не является директорией
$ls /bin
empty_dir  tool.txt
$ls /bin/empty_dir
(пусто)
$cd /home
$ls
readme.txt  readme_moved.txt  multiline.txt
$cd ..
$ls
home  My Documents  a  x  logs  bin
$cd
$cd /a/b
$ls
```



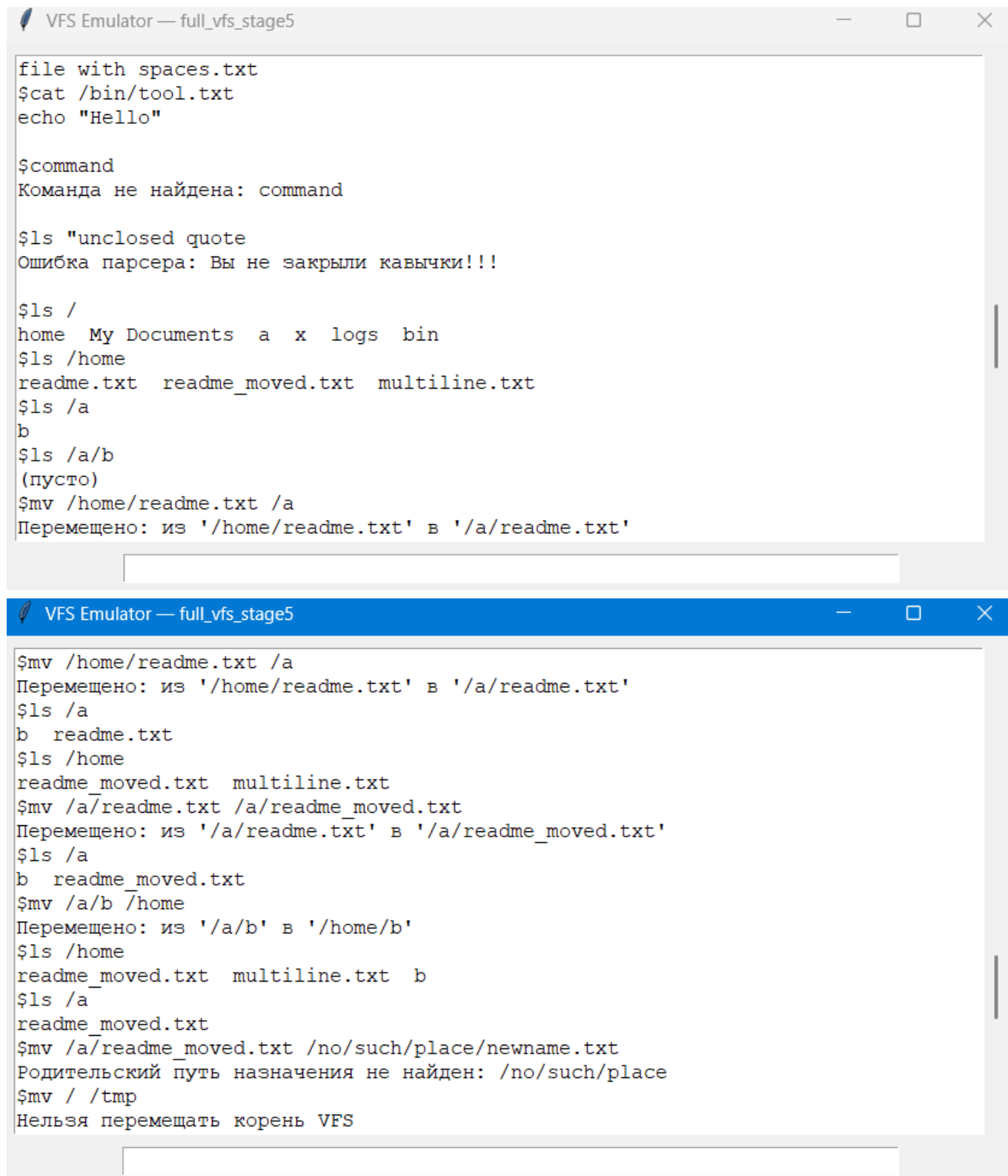
The image shows two screenshots of a terminal window titled "VFS Emulator — full_vfs_stage5". The terminal displays a sequence of commands and their outputs, simulating a file system environment.

First Screenshot:

```
$cd /a/b
$ls
(пусто)
$cd ..
$cd /
$ls /a/b
(пусто)
$cd /bin/tool.txt
/bin/tool.txt не является директорией
$cat /home/readme.txt
This is a readme.\nHello, world!
$cat /home/multiline.txt
This is line 1.
Line 2 here.
Line 3 follows.
$cat /home
/home не является файлом
$cat /no/such/file.txt
Нет такого файла: /no/such/file.txt
$tac /home/multiline.txt
```

Second Screenshot:

```
Нет такого файла: /no/such/file.txt
$tac /home/multiline.txt
Line 3 follows.
Line 2 here.
This is line 1.
$tac /logs/log1.txt
Log entry 2.
Application started successfully.
$cd /home
$ls
readme.txt  readme_moved.txt  multiline.txt
$cd .
$ls
readme.txt  readme_moved.txt  multiline.txt
$cd ..
$ls
home  My Documents  a  x  logs  bin
$ls "My Documents"
file with spaces.txt
$cat /bin/tool.txt
```



The image shows two screenshots of a VFS Emulator terminal window. The top window shows a series of commands and their outputs, including file creation, directory listing, and a move operation. The bottom window continues the sequence, showing more file operations and a final error message about moving the root directory.

```
VFS Emulator — full_vfs_stage5
file with spaces.txt
$cat /bin/tool.txt
echo "Hello"

$command
Команда не найдена: command

$ls "unclosed quote
Ошибка парсера: Вы не закрыли кавычки!!!

$ls /
home  My Documents  a  x  logs  bin
$ls /home
readme.txt  readme_moved.txt  multiline.txt
$ls /a
b
$ls /a/b
(пусто)
$mv /home/readme.txt /a
Перемещено: из '/home/readme.txt' в '/a/readme.txt'
```

```
VFS Emulator — full_vfs_stage5
$mv /home/readme.txt /a
Перемещено: из '/home/readme.txt' в '/a/readme.txt'
$ls /a
b  readme.txt
$ls /home
readme_moved.txt  multiline.txt
$mv /a/readme.txt /a/readme_moved.txt
Перемещено: из '/a/readme.txt' в '/a/readme_moved.txt'
$ls /a
b  readme_moved.txt
$mv /a/b /home
Перемещено: из '/a/b' в '/home/b'
$ls /home
readme_moved.txt  multiline.txt  b
$ls /a
readme_moved.txt
$mv /a/readme_moved.txt /no/such/place/newname.txt
Родительский путь назначения не найден: /no/such/place
$mv / /tmp
Нельзя перемещать корень VFS
```

```
VFS Emulator — full_vfs_stage5

$mv / /tmp
Нельзя перемещать корень VFS
$mv /x /x/y
Нельзя переместить директорию внутрь её потомка!
$mv "/My Documents/file with spaces.txt" /home/file_with_spaces_copy.txt
Перемещено: из '/My Documents/file with spaces.txt' в '/home/file_with_spaces_copy.txt'
$ls /home
readme_moved.txt  multiline.txt  b  file_with_spaces_copy.txt
$mv /home/file_with_spaces_copy.txt /home/readme_moved.txt
Перемещено: из '/home/file_with_spaces_copy.txt' в '/home/readme_moved.txt'
$ls /home
multiline.txt  b  readme_moved.txt
$cd /home
$mv readme_moved.txt ../readme_back.txt
Перемещено: из 'readme_moved.txt' в '/readme_back.txt'
$cd ..
$ls /
home  My Documents  a  x  logs  bin  readme_back.txt
$mv /no/such /tmp

Перемещено: из '/home/file_with_spaces_copy.txt' в '/home/readme_moved.txt'
$ls /home
multiline.txt  b  readme_moved.txt
$cd /home
$mv readme_moved.txt ../readme_back.txt
Перемещено: из 'readme_moved.txt' в '/readme_back.txt'
$cd ..
$ls /
home  My Documents  a  x  logs  bin  readme_back.txt
$mv /no/such /tmp
Нет такого файла или директории: /no/such
$ls /
home  My Documents  a  x  logs  bin  readme_back.txt
$ls /home
multiline.txt  b
$ls /a
readme_moved.txt
Стартовый скрипт завершён с ошибками.
```