Код программы

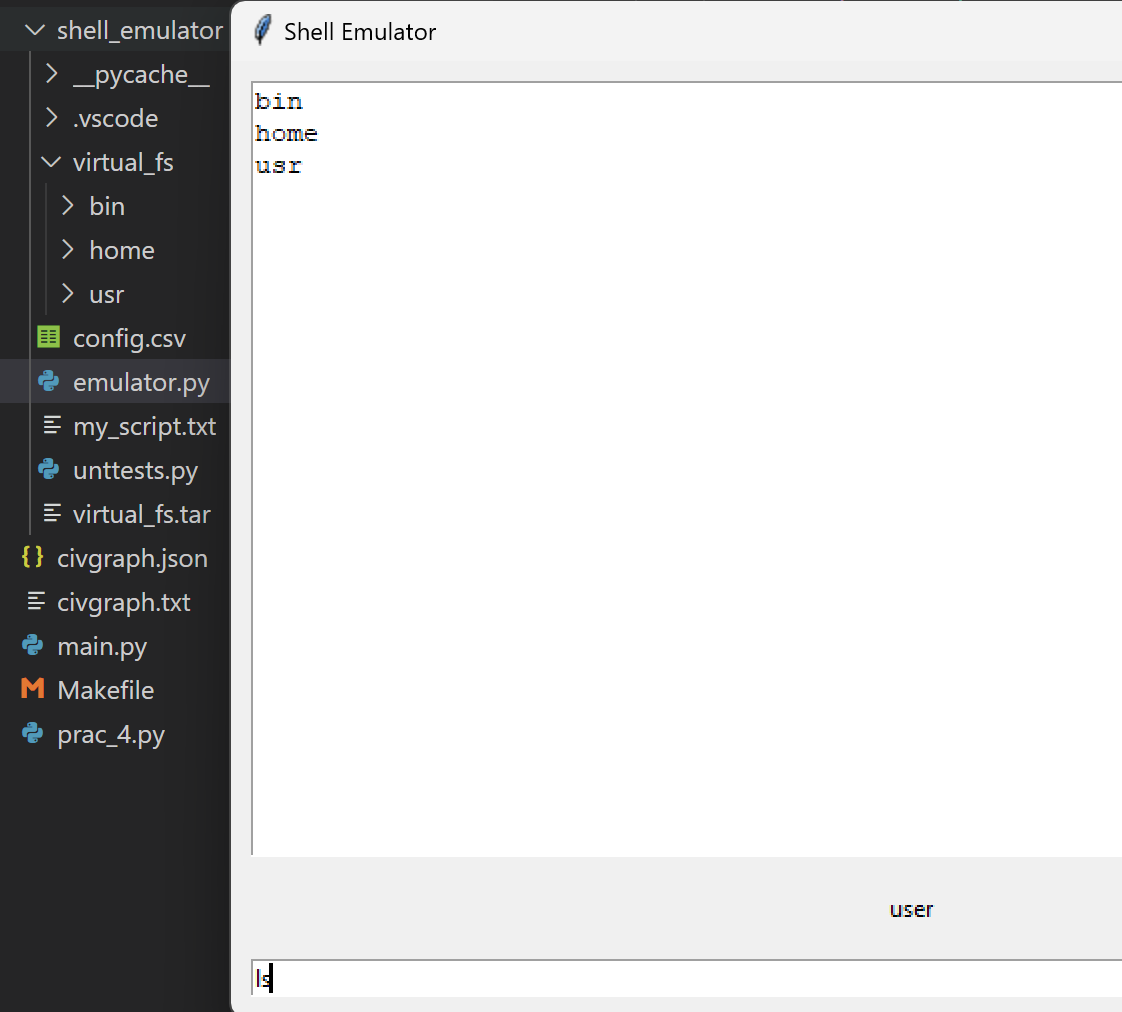
|  |
| --- |
| import tkinter as tk  from tkinter import scrolledtext, messagebox  import os  import tarfile  import sys  import argparse  class ShellEmulator:      def \_\_init\_\_(self, master, virtual\_fs\_path):          self.master = master          self.master.title("Shell Emulator")          self.current\_path = "/"          self.history = []          self.text\_area = scrolledtext.ScrolledText(master, wrap=tk.WORD)          self.text\_area.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)          self.username = os.getlogin()          self.virtual\_fs\_path = virtual\_fs\_path          self.label = tk.Label(master, text=f"{self.username}")          self.label.pack(padx=10, pady=5)          self.entry = tk.Entry(master)          self.entry.pack(padx=10, pady=10, fill=tk.X)          self.entry.bind("<Return>", self.execute\_command)          self.extract\_virtual\_fs()      @staticmethod      def parse\_arguments():          parser = argparse.ArgumentParser(description='Запуск эмулятора командной строки vshell.')          parser.add\_argument('--script', type=str, help='Имя файла со скриптом команд.')          parser.add\_argument('virtual\_fs', type=str, help='Путь к образу файловой системы (tar или zip).')            args = parser.parse\_args()            if not os.path.exists(args.virtual\_fs):              parser.error(f"Файл виртуальной файловой системы '{args.virtual\_fs}' не найден.")            return args      def extract\_virtual\_fs(self):          if not os.path.exists(self.virtual\_fs\_path):              messagebox.showerror("Ошибка", "Файл виртуальной файловой системы не найден.")              return          with tarfile.open(self.virtual\_fs\_path) as tar:              tar.extractall(path="virtual\_fs", filter=tarfile.data\_filter)      def load\_script(self, script\_file):          try:              with open(script\_file, 'r') as file:                  for line in file:                      command = line.strip()                      if command:                          self.execute\_command\_from\_script(command)          except FileNotFoundError:              messagebox.showerror("Ошибка", f"Файл скрипта {script\_file} не найден.")      def execute\_command(self, event):          command = self.entry.get().strip()          if not command:              self.entry.delete(0, tk.END)              return          self.history.append(command)          command\_dict = {              "ls": self.list\_files,              "cd": lambda: self.change\_directory(command[3:]),              "pwd": self.print\_working\_directory,              "cat": lambda: self.cat\_file(command[4:]),              "exit": self.master.quit,              "history": self.show\_history,              "touch": lambda: self.touch\_file(command[6:]),          }          cmd\_func = command\_dict.get(command.split()[0], None)          if cmd\_func:              cmd\_func()          else:              self.text\_area.insert(tk.END, f"{self.username}: команда не найдена\n")          self.entry.delete(0, tk.END)        def execute\_command\_from\_script(self, command):          command\_dict = {              "ls": self.list\_files,              "cd": lambda: self.change\_directory(command[3:]),              "pwd": self.print\_working\_directory,              "cat": lambda: self.cat\_file(command[4:]),              "exit": self.master.quit,              "history": self.show\_history,              "touch": lambda: self.touch\_file(command[6:])          }          cmd\_func = command\_dict.get(command.split()[0], None)          if cmd\_func:              cmd\_func()          else:              self.text\_area.insert(tk.END, f"{self.username}: команда не найдена\n")      def list\_files(self):          try:              files = os.listdir(f"virtual\_fs{self.current\_path}")              output = "\n".join(files) if files else "Пустая директория\n"              self.text\_area.insert(tk.END, f"{output}\n")          except FileNotFoundError:              self.text\_area.insert(tk.END, "Директория не найдена\n")      def change\_directory(self, path):          if path == "..":              if self.current\_path != "/":                  parts = self.current\_path.split("/")                  parts.pop()                  self.current\_path = "/".join(parts) or "/"                  return            new\_path = os.path.join(f"virtual\_fs{self.current\_path}", path)            if os.path.isdir(new\_path):              self.current\_path = new\_path.replace("virtual\_fs", "")              return          else:              self.text\_area.insert(tk.END, "Директория не найдена\n")      def print\_working\_directory(self):          current\_dir = f"{self.username}:{self.current\_path}\n"          self.text\_area.insert(tk.END, current\_dir)      def cat\_file(self, filename):          try:              with open(os.path.join(f"virtual\_fs{self.current\_path}", filename), 'r') as file:                  content = file.read()                  self.text\_area.insert(tk.END, f"{content}\n")          except FileNotFoundError:              self.text\_area.insert(tk.END, "Файл не найден\n")        def touch\_file(self, filename):          try:              file\_path = os.path.join(f"virtual\_fs{self.current\_path}", filename.strip())              with open(file\_path, 'a'):                  pass              self.text\_area.insert(tk.END, f"Файл '{filename}' создан.\n")          except Exception as e:              self.text\_area.insert(tk.END, f"Ошибка при создании файла '{filename}': {str(e)}\n")      def show\_history(self):          history\_output = "\n".join(self.history) or "История пуста\n"          self.text\_area.insert(tk.END, f"История команд:\n{history\_output}\n")  if \_\_name\_\_ == "\_\_main\_\_":      args = ShellEmulator.parse\_arguments()      root = tk.Tk()      app = ShellEmulator(root, args.virtual\_fs)        if args.script:          app.load\_script(args.script)      root.mainloop() |

Запуск программы

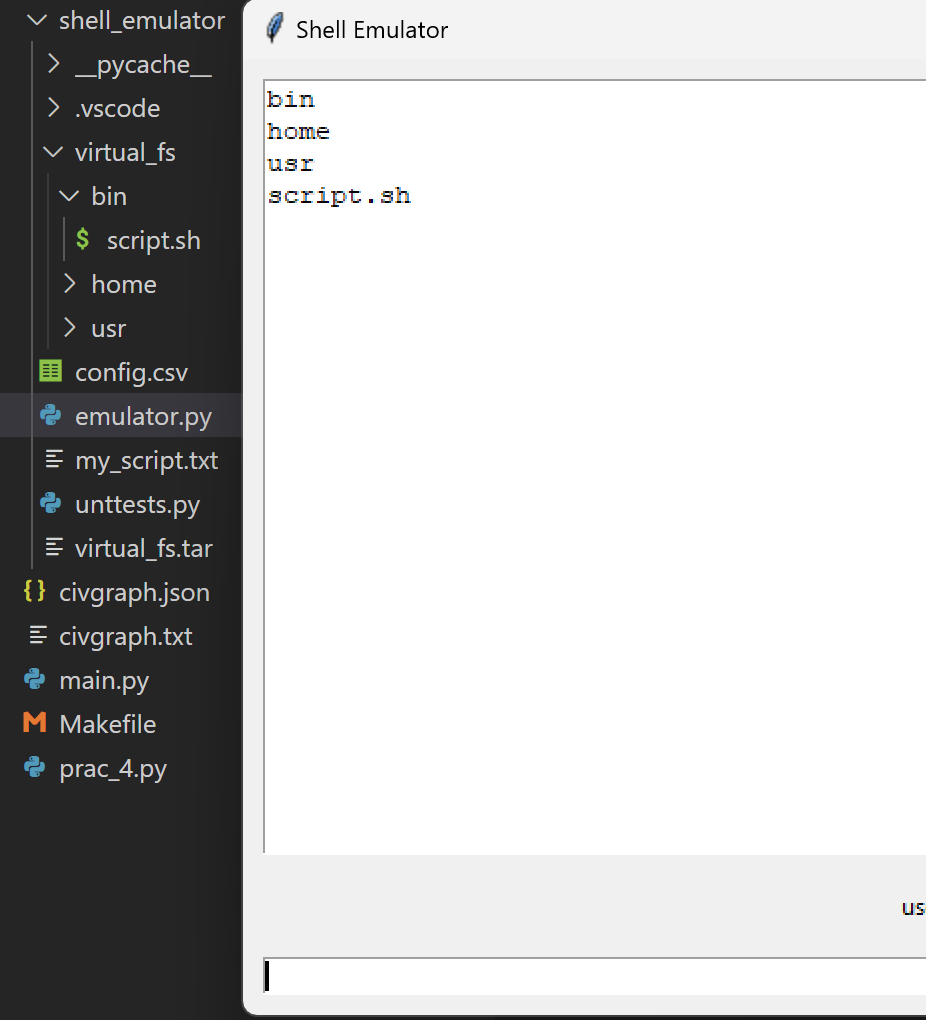


Демонстрация работы программы

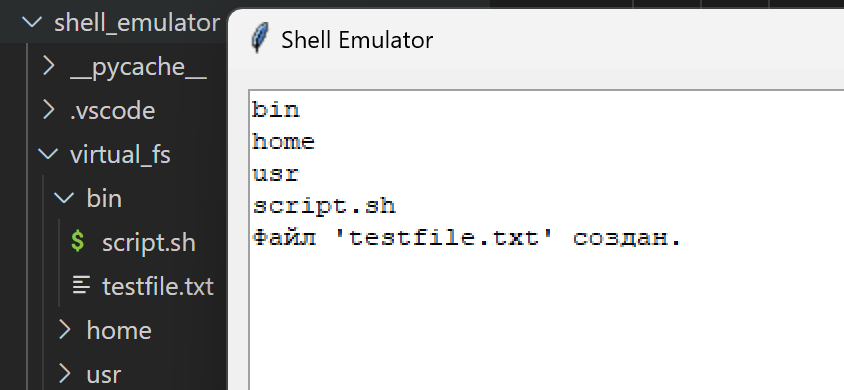
-ls показать список доступных файлов/директорий



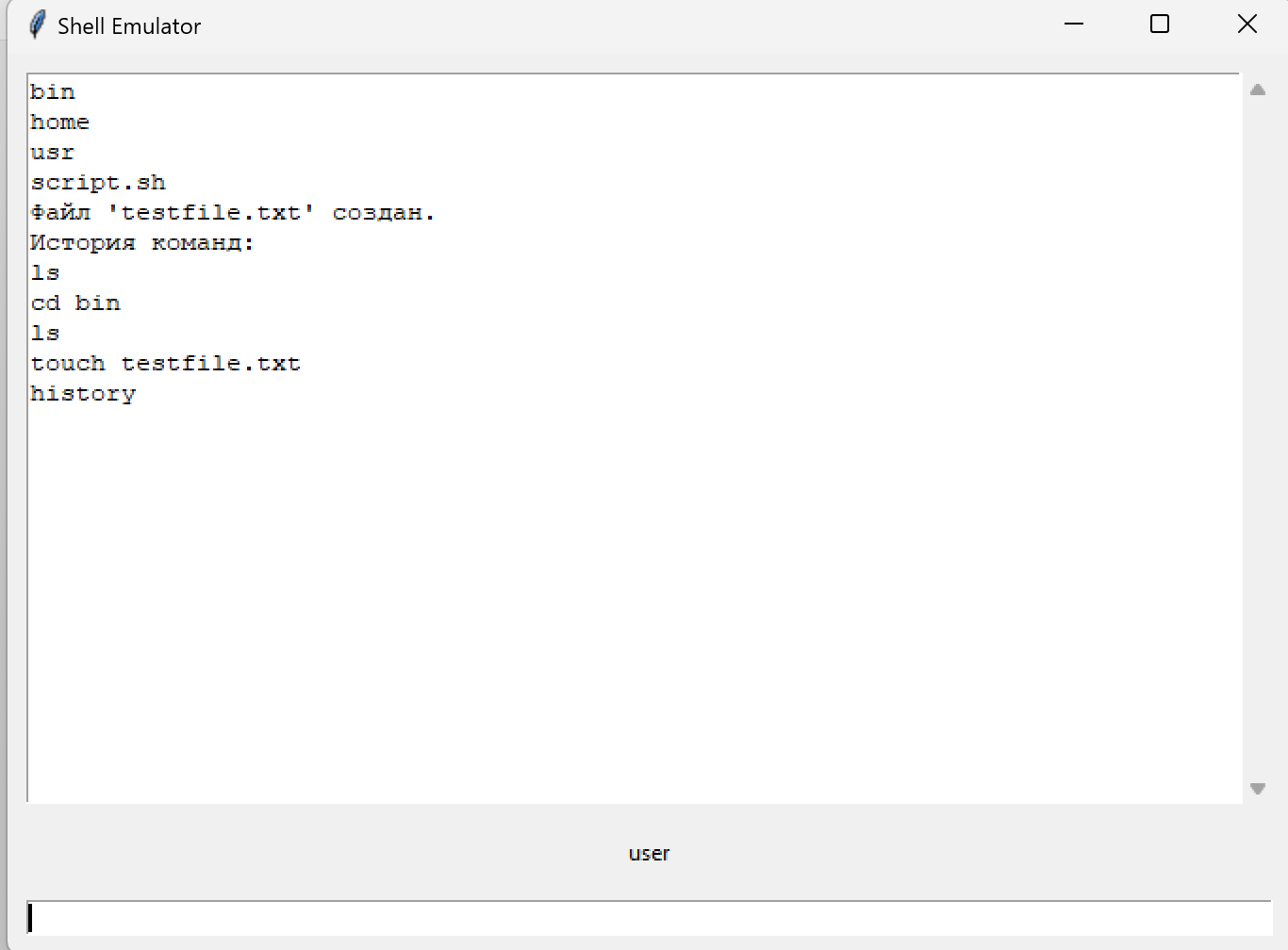
-cd bin переход в директорию



-touch testfile создание файла



-history история команд



Отработка unit тестов

