ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на создание автоматизированной системы
«Эмулятор UNIX-подобной операционной системы — GUI UNIX OS»

Код: GUI-UNIX-001

Разработчики:

Зепалов Егор Вадимович

Зернов Никита Андреевич

Коротаев Алексей Михайлович

Любишкин Савва Николаевич

Степанов Константин Кириллович

Москва, 2025

Содержание

1. Введение

1.1. Наименование программы

GUI UNIX OS — эмулятор UNIX-подобной операционной системы с графическим окном терминала (реализовано с использованием pygame), принимающий образ виртуальной файловой системы в виде zip-архива.

1.2. Краткая характеристика области применения

Программа предназначена для обучения и тестирования базовых команд UNIX, имитации работы с файловой системой и выполнения простых сценариев (start-скриптов) в контролируемой, безопасной (виртуальной) среде. Подходит для учебных лабораторий, демонстраций и самостоятельного обучения.

2. Основание для разработки, нормативные и исходные документы

2.1. Основание для проведения разработки

Необходимость создания учебного инструмента для отработки базовых операций в shell-среде без риска повредить реальную файловую систему.

2.2. Наименование и условное обозначение темы разработки

Наименование: «Эмулятор UNIX-подобной ОС — GUI UNIX OS».

Шифр темы: GUI-UNIX-001

3. Назначение разработки

3.1. Функциональное назначение

Обеспечение пользователю возможности вводить команды, аналогичные базовым UNIX-командам, просматривать и изменять структуру виртуальной файловой системы, загруженной из ZIP-архива, и получать текстовый вывод результатов.

3.2. Эксплуатационное назначение

Использование в учебных и демонстрационных целях: для студентов, преподавателей и сотрудников, желающих изучать работу shell-команд и манипуляции с файловой системой в безопасном окружении.

3.3. Задачи, решаемые в ходе разработки

Задачи, решаемые в ходе разработки

- Реализация текстового интерфейса/терминала в окне pygame.
- Обработка и монтирование структуры виртуальной файловой системы из ZIP-архива.
- Поддержка набора команд: help, ls, cd, exit, wc, mv, clear.
- Выполнение стартового скрипта (если указан) — последовательного выполнения команд из файла.
- Корректная обработка ошибок, логирование событий, простые тесты/приёмка.

4. Требования к программе

4.1. Требования к функциональным характеристикам

4.1.1 Требования к составу выполняемых функций

Запуск программы с параметрами командной строки:

--user — имя пользователя, используемое в приглашении.

--archive — путь к zip-архиву, представляющему виртуальную файловую систему.

--script — имя скрипта внутри архива, который нужно выполнить при старте.

help — вывод списка доступных команд и кратких описаний.

ls — перечисление содержимого текущей директории или указанного пути.

cd — смена текущей директории.

exit — корректное завершение работы эмулятора.

wc — подсчёт: строк, слов или символов в указанном текстовом файле.

mv — перемещение файла/папки или переименование в пределах виртуальной ФС.

clear — очистка выводимой консоли в окне эмулятора.

Логирование: ошибки и важные события записываются в файл логов.

4.1.2 Требования к организации входных данных

Входные данные программы должны быть организованы в виде:

- Архив ZIP (стандартный ZIP): корневая папка архива рассматривается как корень виртуальной файловой системы. В архиве допустимы файлы любых форматов (текстовые, бинарные), и вложенные директории.
- Файл start-скрипта (если указан) должен быть текстовым и находиться внутри архива по указанному относительному пути.

Программа должна корректно обрабатывать:

- отсутствие архива вывод корректной диагностической ошибки и завершение;
- повреждённый/некорректный архив аккуратное сообщение об ошибке и запись в лог.

4.1.3 Требования к организации выходных данных

Вывод команд — текст в окне эмулятора; при необходимости — дополнительно запись в лог.

Форматы сообщений: понятные и однозначные тексты ошибок.

При завершении работы — возвратный код процесса 0 при успешном завершении, ненулевой код при фатальных ошибках.

4.2 Требования к надежности

4.2.1 Требования к обеспечению надежного (устойчивого) функционирования программы

Корректная обработка некорректных входных данных (несуществующие пути, неверный формат файлов и т.п.) без краха хоста.

Любая ошибка в работе (исключение Python) должна быть перехвачена, краткая информация показана пользователю, подробный стек запись в лог.

4.2.2 Время восстановления после отказа

Перезапуск приложения (старт скрипта/монтажа архива) должен занимать не более 1 минуты при стандартных условиях (медленное IO не учитывается).

4.2.3 Отказы из-за некорректных действий пользователя

Некорректные команды должны возвращать сообщение об ошибке и не приводить к изменению реальной файловой системы хоста.

Скрипты и команды запускаются внутри эмулятора и не должны выполнять реальных shell-команд на хосте.

4.3 Условия эксплуатации

4.3.1 Требования к видам обслуживания

ОС: Windows 10/11, Linux (современные дистрибутивы), macOS — при установленном Python 3.10+.

Наличие интернета не обязательно для работы, требуется только при установке зависимостей.

4.3.2 Требования к численности и квалификации персонала

Для запуска: пользователь с базовыми навыками работы в командной строке и ОС.

Для сопровождения: разработчик/администратор, знакомый с Python и pygame.

4.4 Требования к составу и параметрам технических средств

- CPU: 1 ГГц или выше.
- RAM: 512 МБ или выше.
- HDD/SSD: 200 МБ свободного места (зависит от размера архивов).
- (Рекомендуется: 2 ГБ RAM, 1 ГБ свободного диска для логов и тестовых архивов.)

4.5 Требования к информационной и программной совместимости

4.5.1 Требования к информационным структурам и методам решения

Внутреннее представление структуры файлов — древовидные структуры/словарь Python; для чтения архива использовать модуль zipfile.

4.5.2 Требования к исходным кодам и языкам программирования

Язык: Python 3.10 и выше.

Структура проекта: main.py, модули fs.py (виртуальная ФС), commands.py, ui.py (pygame), logger.py, tests/.

4.5.3 Требования к программным средствам, используемым программой

Программа должна работать под управлением ОС Linux/Windows с установленным интерпретатором Python и необходимыми зависимостями.

5. Требования к интерфейсу

5.1 Общие требования к интерфейсу

Интерфейс реализован в окне pygame: область вывода текста + строка ввода (prompt).

Текст выводится моноширинным шрифтом; новая команда вводится одной строкой.

5.2 Поведение интерфейса

Полоса прокрутки или поддержка прокрутки вверх/вниз при большом объёме вывода.

Кнопки окна: стандартные (свернуть, развернуть, закрыть).

При закрытии окна — корректное завершение (сохранение логов).

6. Технико-экономические показатели

6.1 Ожидаемый экономический эффект

Ускорение усвоения студентами практических навыков работы с командной строкой в безопасной среде.

Снижение риска случайного повреждения реальной файловой системы при обучении.

6.2 Затраты на разработку

Разработка силами студентов в рамках учебного задания; коммерческие затраты минимальны (только время и усилия участников).

7. Требования к программной документации

Состав программной документации должен включать в себя:

а) техническое задание;

б) пользовательская документация;

в) техническая документация;

г) текстовая документация;

д) системная документация.

8. Порядок контроля и приемки

8.1 Организация приемки

Приемка готового продукта осуществляется заказчиком.

8.2 Методы контроля качества

Контроль качества осуществляется методом поэтапного тестирования.

8.3 Тестовые сценарии

Для приемки Заказчик предоставляет Исполнителю тестовый сценарий, включающий не менее 20 тест-кейсов, покрывающих все функциональные требования п. 4.1.

8.4 Критерии приемки

Критерием успешной приемки является успешное прохождение не менее 95% всех тест-кейсов.

8.5 Документация приемки

После успешного прохождения испытаний подписывается Акт о приемке программного обеспечения в опытную эксплуатацию.

8.6 Завершение приемки

По итогам успешной опытной эксплуатации в течение установленного срока подписывается Акт о приемке программного обеспечения в промышленную эксплуатацию.

9. Стадии и этапы разработки

9.1 План-график разработки

Разработка программного обеспечения должна быть выполнена в следующие сроки (таблица 1).

Таблица 1 - План-график разработки

№	Наименование этапа	Срок исполнения	Исполнитель
1	Анализ требований и проектирование архитектуры	3 рабочих дня	Разработчик
2	Написание кода и модульное тестирование	8 рабочих дней	Разработчик
3	Наполнение контентом	2 рабочих дня	Заказчик
4	Пилотная эксплуатация и приемочные испытания	3 рабочих дней	Совместно
5	Ввод в промышленную эксплуатацию	1 рабочий день	Разработчик

9.2 Форматы представления документации

Вся документация должна быть предоставлена в электронном виде в форматах PDF (для руководств) и .py (для исходного кода).