Федеральное бюджетное образовательное учреждение  
высшего образования

«Уфимский университет науки и технологий»

Кафедра технической кибернетики

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 90 |  |  |  |  |  |  |  |  |  |  |  |
| 80 |  |  |  |  |  |  |  |  |  |  |  |
| 70 |  |  |  |  |  |  |  |  |  |  |  |
| 60 |  |  |  |  |  |  |  |  |  |  |  |
| 50 |  |  |  |  |  |  |  |  |  |  |  |
| 40 |  |  |  |  |  |  |  |  |  |  |  |
| 30 |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |
| 0 |  |  |  |  |  |  |  |  |  |  |  |

|  |
| --- |
| Проектирование гипотетической |
| операционной системы |
| с заданными характеристиками |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине «**Операционные системы**»

|  |
| --- |
| 3271.202190.000 ПЗ |
| (обозначение документа) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Группа |  |  | Фамилия И. О. | Подпись | Дата | Оценка |
| ИВТ–ПО–201Б |  |
|  |  |
| Студент | | | Кумушбаева Г. А. |  |  |  |
| Консультант | | | Рыжов Г. И. |  |  |  |
| Принял | | | Рыжов Г. И. |  |  |  |

**Уфа 2024**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Уфимский университет науки и технологий»

Кафедра технической кибернетики

**ЗАДАНИЕ**

на курсовой проект по дисциплине «**Операционные системы**»

Студент \_Кумушбаева Г. А.\_\_\_ Группа \_ИВТ-ПО-201Б\_\_ Консультант \_Рыжов Г. И.\_\_\_\_

1. **Тема курсового проекта**

\_\_\_Проектирование гипотетической операционной системы с заданными характеристиками\_\_\_

1. **Основное содержание:** на основе заданного индивидуального задания спроектировать гипотетическую компьютерную операционную систему.

**Индивидуальное задание:**

**Режим работы компьютера:** однопрограммный

**Архитектура операционной системы:** экзоядро

**Подсистема управления процессами:**

– алгоритм планирования процессов: вытесняющий – циклический *LIFO (Last In, First Out)*;

– алгоритм организации взаимодействия процессов: алгоритм пекарни Лэмпорта;

– задача синхронизации процессов: задача «Обедающие философы»;

– способ борьбы с тупиками: предотвращение тупиков – алгоритм банкира.

**Подсистема управления памятью:** сегментное распределение памяти

**Подсистема управления файлами:**

– иерархия каталогов: дерево;

– логическая организация файловой системы: индексная логическая организация;

– физическая организация файловой системы: связанный список индексов.

**Загрузка операционной системы:** загрузка с жесткого диска компьютера.

**Запуск прикладных программ:** запуск с жесткого диска компьютера.

**Структура управляющих блоков базы данных операционной системы:**

– формирование базы данных операционной системы;

– схема взаимодействия управляющих блоков базы данных операционной системы.

1. **Требования к оформлению.** Пояснительная записка должна быть оформлена в редакторе *Microsoft Word* в соответствии с требованиями ГОСТ.

В пояснительной записке должны содержаться следующие разделы:

Титульный лист.

Задание на курсовой проект.

Реферат.

Содержание.

Введение.

Выбор архитектуры вычислительной системы.

Описание режима работы компьютера.

Описание архитектуры операционной системы.

Описание подсистемы управления процессами.

Описание подсистемы управления памятью.

Описание подсистемы управления файлами.

Проектирование графического интерфейса.

Проектирование командного языка.

Загрузка операционной системы.

Запуск прикладных программ.

Структура управляющих блоков базы данных операционной системы.

Описание функционирования операционной системы.

Заключение.

Список использованной литературы.

Дата выдачи \_\_\_14.10.2024\_\_\_\_\_\_\_\_\_ Дата окончания \_\_\_\_\_\_21.12.2024\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Рыжов Г. И.)

**РЕФЕРАТ**

Пояснительная записка – 49 с, 10 рис., 9 табл., 14 источн.

ОПЕРАЦИОНННАЯ СИСТЕМА, АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ, БАЗА ДАННЫХ, ПРОЦЕССЫ, ПАМЯТЬ, ФАЙЛОВАЯ СИСТЕМА, *LINUX*, ЯДРО, ГРАФИЧЕСКИЙ ИНТЕРФЕЙС, КОМАНДНЫЙ ЯЗЫК

Объект исследования – гипотетическая компьютерная операционная система.

Предмет исследования – процессы в гипотетической компьютерной операционной системе.

Цель курсового проекта:

* закрепление и углубление теоретических знаний в области современных операционных систем, активное применение знаний, полученных в лекционном курсе, на практических и лабораторных занятиях;
* формирование систематизированного представления о основополагающих принципах работы операционных систем, о структурах и механизмах, лежащих внутри операционных системы.

Методами описания операционной системы:

* выделены ключевые составляющие: абстракции (процессы, адресное пространство, файлы, потоки); функции (создание, запись, управление, открытие, распределение); конкретные реализации (архитектуры, алгоритмы);
* определены основные функции: управление устройствами компьютера (его ресурсами), непосредственное управление процессами, ведение файловой структуры, пользовательский интерфейс;
* рассмотрены основные компоненты операционной системы: управление файловой системой, командный язык, графический интерфейс.

Результатом работы является гипотетическая компьютерная операционная система с заданными характеристиками.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#__RefHeading___Toc4651_3549619164)

[1 Теоретические основы тайловых менеджеров и Wayland-экосистемы 9](#__RefHeading___Toc4653_3549619164)

[1.1 Эволюция оконных менеджеров: от stacking к tiling 9](#__RefHeading___Toc5256_3549619164)

[2 Архитектура пользовательской сборки на Arch Linux 14](#__RefHeading___Toc4655_3549619164)

[2.2 Режим работы компьютера 14](#__RefHeading___Toc4657_3549619164)

[3 Конфигурация Hyprland: анализ и особенности 17](#__RefHeading___Toc4659_3549619164)

[ЗАКЛЮЧЕНИЕ 22](#__RefHeading___Toc4661_3549619164)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#__RefHeading___Toc4663_3549619164)

**ВВЕДЕНИЕ**

Современные операционные системы стремятся к балансу между функциональностью, производительностью и эстетикой пользовательского интерфейса. В условиях роста требований к ресурсоэффективности и персонализации рабочего пространства всё большую популярность приобретают тайловые оконные менеджеры (tiling window managers, TWM) — программные компоненты, автоматически организующие окна приложений на экране без перекрытий и с минимальным участием пользователя. В отличие от традиционных сред рабочего стола (desktop environments, DE), таких как GNOME или KDE Plasma, тайловые менеджеры предоставляют полный контроль над компоновкой окон, управлением ресурсами и внешним видом системы, что особенно ценно для разработчиков, системных администраторов и пользователей, стремящихся к минимализму и высокой производительности.

С переходом от устаревшего X11-стека к современному протоколу Wayland экосистема Linux получила новые возможности в области безопасности, производительности и интеграции графических компонентов. Однако этот переход потребовал создания новых композиторов, сочетающих функции дисплейного сервера и оконного менеджера. Одним из наиболее перспективных решений в этой области стал Hyprland — динамический тайловый композитор, построенный на библиотеке wlroots, поддерживающий как строгий тайлинг, так и плавающие окна, а также современные визуальные эффекты: размытие (blur), тени, анимации и прозрачность.

Данная работа посвящена анализу пользовательской конфигурации Hyprland в среде Arch Linux — дистрибутива, известного своей гибкостью, минимализмом и ориентацией на опытного пользователя. В отчёте рассматриваются архитектурные особенности сборки, включающей:

* композитор Hyprland с настроенным мультимониторным режимом и привязкой рабочих пространств,
* информационную панель Waybar с кастомными модулями (раскладка клавиатуры, управление питанием, Bluetooth),
* систему автоматизации через скрипты и автозапуск ключевых сервисов.

Цель отчёта — не только продемонстрировать работоспособность и эстетическую привлекательность данной конфигурации, но и проанализировать её с точки зрения архитектурной целостности, масштабируемости и соответствия современным практикам настройки Wayland-сред.

## **Теоретические основы тайловых менеджеров и Wayland-экосистемы**

## **1.1 Эволюция оконных менеджеров: от stacking к tiling**

Оконный менеджер (window manager, WM) — это компонент графической подсистемы операционной системы, отвечающий за размещение, отображение и взаимодействие с окнами приложений. Исторически первые оконные менеджеры, такие как **twm** (Tab Window Manager) или **FVWM**, использовали **stacking-модель** (наложение), при которой окна могут свободно перекрывать друг друга, как листы бумаги на столе. Пользователь вручную перемещает, изменяет размер и упорядочивает окна — подход, ставший основой для современных сред рабочего стола (GNOME, KDE Plasma, XFCE).

Однако с ростом сложности задач и количества одновременно запущенных приложений stacking-модель стала демонстрировать недостатки:

* неэффективное использование экранного пространства,
* необходимость постоянного ручного управления окнами,
* снижение продуктивности из-за визуального «шума».

В ответ на эти вызовы в 1990–2000-х годах возникли **тайловые оконные менеджеры** (tiling window managers), такие как **dwm**, **i3**, **xmonad**. В отличие от stacking-модели, тайлеры автоматически разбивают экран на неперекрывающиеся области (тайлы), в которые помещаются окна. Пользователь управляет компоновкой с помощью клавиатуры, а не мыши, что повышает скорость работы и снижает когнитивную нагрузку.

Ключевые принципы тайловых менеджеров:

* **Автоматическая компоновка**: окна не перекрываются, всегда видны.
* **Клавиатурно-ориентированное управление**: минимизация зависимости от мыши.
* **Минимализм**: отсутствие панелей, кнопок, анимаций «по умолчанию».
* **Конфигурируемость через код**: большинство TWM требуют редактирования исходного кода или конфигурационных файлов.

Таким образом, тайловые менеджеры представляют собой философский и технический ответ на избыточность традиционных DE, ориентируясь на эффективность, предсказуемость и контроль.

### **Wayland vs X11: архитектурный сдвиг в графической подсистеме Linux**

Долгое время графическая подсистема Linux строилась вокруг **X Window System (X11)** — протокола, разработанного в 1980-х годах. Несмотря на свою гибкость и кроссплатформенность, X11 страдает от архитектурных устареваний:

* отсутствие встроенной поддержки современных графических API (OpenGL/Vulkan),
* сложная модель безопасности (любое приложение может «подслушивать» ввод),
* избыточная сетевая ориентация (редко используется на практике),
* необходимость внешнего композитора для эффектов (тени, анимации).

В 2008 году началась разработка **Wayland** — нового протокола дисплейного сервера, спроектированного с нуля для современных требований. В отличие от X11, где дисплейный сервер (X Server) и оконный менеджер — разные сущности, в Wayland **композитор объединяет обе роли**. Это означает, что:

* композитор сам управляет буферами, вводом, разрешениями и отрисовкой,
* каждое приложение рисует напрямую в собственный буфер (через EGL/OpenGL/Vulkan),
* безопасность повышена: приложения не могут читать ввод или содержимое других окон без разрешения,
* производительность выше за счёт устранения промежуточных копий буферов.

Wayland не является «заменой X11» в прямом смысле — это **новая архитектура**, требующая переписывания клиентских приложений и инструментов. Для совместимости с унаследованным ПО используется **XWayland** — прослойка, запускающая X11-приложения внутри Wayland-сессии.

### **1.3 Hyprland: динамический тайлер нового поколения**

На фоне роста популярности Wayland возникла потребность в оконных менеджерах, сочетающих **функциональность тайлеров** с **современной графикой**. Большинство ранних Wayland-композиторов (например, **sway**) сознательно отказались от визуальных эффектов ради стабильности и минимализма. **Hyprland**, появившийся в 2022 году, стал ответом на запрос сообщества о «красивом, но функциональном» WM.

Hyprland — это **динамический тайловый композитор**, построенный на библиотеке **wlroots** (той же, что используется в sway, river, labwc). Его ключевые особенности:

* Поддержка нескольких layout-режимов: **dwindle** (бинарное деление), **master** (главное окно + стек), **спираль** и др.
* Возможность **псевдо-тайлинга** (pseudotile): окно временно становится флоатинговым без потери позиции в тайле.
* Встроенные **визуальные эффекты**: размытие (blur), тени, анимации переходов, градиентные рамки.
* Полная поддержка **мультимониторности**, **масштабирования (HiDPI)** и **входных устройств** (мышь, сенсорная панель, планшеты).
* Гибкая система **правил для окон** (windowrule), позволяющая задавать поведение по классу, заголовку, монитору и другим параметрам.

Важно отметить, что Hyprland **не является средой рабочего стола** — он предоставляет только базовую функциональность WM. Всё остальное (панель, обои, уведомления, скриншоты) реализуется через сторонние утилиты (Waybar, hyprpaper, grim, mako и др.), что соответствует философии Unix: «делай одну вещь, но делай её хорошо».

Таким образом, Hyprland представляет собой синтез **практичности тайловых менеджеров**, **безопасности Wayland** и **эстетики современных интерфейсов**, что делает его привлекательным выбором для опытных пользователей Linux.

# **2 Архитектура пользовательской сборки на Arch Linux**

### **Почему Arch и почему Wayland?**

Выбор Arch Linux в качестве основы не случаен. Этот дистрибутив изначально строится «с нуля»: пользователь сам решает, какие компоненты ему нужны, а от каких можно отказаться. В отличие от «готовых» систем вроде Ubuntu или Fedora, здесь нет скрытых демонов, дублирующих сервисов или предустановленных приложений, которые никто не просил. Это даёт полный контроль над системой — особенно важно при работе с тонкими настройками Wayland.

Wayland же был выбран как логичное продолжение этой философии. X11, несмотря на свою надёжность, давно устарел архитектурно: он не умеет работать с современными GPU напрямую, плохо защищает от перехвата ввода и требует внешних композиторов для даже самых простых эффектов. Wayland устраняет эти проблемы, но взамен требует пересмотра привычного стека инструментов. Именно поэтому в сборке используются только нативные Wayland-приложения или их адаптированные аналоги.

## **Режим работы компьютера**

Основа системы — **Hyprland** как композитор и оконный менеджер. Он заменяет собой и X-сервер, и WM, и частично — даже DE. Всё остальное подключается как отдельные модули:

* **Waybar** — лёгкая, но гибкая панель, поддерживающая кастомные скрипты и интерактивность.
* **hyprpaper** — простой менеджер обоев, совместимый с размытием в Hyprland.
* **swaykbdd** — индикатор раскладки клавиатуры, необходимый, так как Wayland не передаёт состояние раскладки в панель напрямую.
* **wlogout** — красивое меню выключения/перезагрузки, запускаемое по Super+X.
* **nekoray** — прокси-менеджер, автоматически привязанный к воркспейсу 7 через правило окна.

Все компоненты запускаются через exec-once в конфигурации Hyprland, что исключает необходимость в .xprofile, systemd --user или других менеджерах сессий. Это упрощает отладку и делает запуск предсказуемым.

### **Принципы проектирования конфигурации**

Конфигурация строилась по трём ключевым принципам:

1. **Разделение зон ответственности**.

Рабочие пространства жёстко привязаны к мониторам:

* + HDMI-A-1 (внешний монитор) — воркспейсы 1–6 (основная работа: браузер, код, документы),
  + eDP-1 (встроенный экран ноутбука) — воркспейсы 7–10 (фоновые задачи: мессенджер, музыка, прокси).  
    Такой подход позволяет не «терять» окна при переключении между мониторами и чётко разделять контексты.

1. **Автоматизация без избыточности**.

Приложения запускаются не «просто так», а по смыслу: Telegram — на 8-й, терминал Ghostty — на 9-й, Yandex Music — на 10-й. Диалоговые окна, pavucontrol, blueman — автоматически становятся плавающими и прикреплёнными (pin), чтобы не мешать основному workflow.

1. **Эстетика как функция**.

Размытие, тени, градиентная рамка активного окна и плавные анимации — не просто «красиво». Они дают визуальную обратную связь: пользователь сразу видит, какое окно активно, как происходит переключение воркспейсов, где заканчивается одно приложение и начинается другое. Это снижает когнитивную нагрузку при интенсивной работе.

### **Интеграция с системой и внешними устройствами**

Сборка учитывает реальные сценарии использования:

* **Графический планшет** настраивается через кастомный скрипт map-tablet.sh (запускается при старте), что особенно важно для художников или тех, кто использует стилус.
* **Яркость и громкость** управляются через brightnessctl и wpctl — это нативные Wayland-совместимые утилиты, в отличие от устаревших xbacklight или amixer.
* **Мультимедийные клавиши** работают «из коробки» благодаря привязке к XF86...-кодам.
* **Bluetooth и сеть** отображаются в панели с иконками и именами устройств, а клик по ним открывает соответствующие настройки (blueman-manager, nmtui).

Особое внимание уделено **безопасности и стабильности**:

* Используется XWayland только для тех приложений, которые пока не имеют нативной поддержки Wayland (например, Maple — запускается по Super+Shift+M).
* Все скрипты хранятся в ~/.config/scripts/, что соответствует XDG Base Directory Specification.
* Нет дублирования функций: например, управление звуком — только через PipeWire/WirePlumber, без PulseAudio в чистом виде.

Таким образом, сборка — это не просто «красивый рабочий стол», а **продуманная среда**, в которой каждый компонент выполняет свою роль, не мешая другим, и адаптированная под конкретные задачи пользователя.

# **3 Конфигурация Hyprland: анализ и особенности**

### **Мониторы и рабочие пространства — логика разделения**

В конфиге явно прописаны два монитора:

* eDP-1 — встроенный экран ноутбука,
* HDMI-A-1 — внешний дисплей, подключённый справа (auto-right).

Это не просто «два экрана», а **разделение контекстов**. Внешний монитор (HDMI) — основное рабочее поле: воркспейсы 1–6. Здесь живут браузер, редактор кода, терминалы, документы. Встроенный экран (eDP) — вспомогательный: воркспейсы 7–10. Туда отправляются фоновые задачи: Telegram, музыка, прокси-менеджер.

Такой подход позволяет не переключаться между «рабочим» и «личным» на одном экране. Если внешний монитор отключён (например, в дороге), всё автоматически переносится на eDP, и ничего не теряется — просто меняется фокус.

Привязка сделана жёстко через workspace = N, monitor:..., что гарантирует стабильность. Да, при подключении нового монитора вручную придётся править конфиг, но в моём случае сценарий почти всегда один и тот же — ноутбук + один внешний дисплей, так что это оправдано.

### **3.2. Управление окнами: dwindle + pseudotile**

В качестве основного layout выбран **dwindle** — бинарное деление экрана. Каждое новое окно делит текущую область пополам, что даёт предсказуемую и сбалансированную компоновку даже при большом количестве приложений.

Но главное — включён **pseudotile** (pseudotile = true). Это гениальная фича: нажимаешь Super+P — окно временно становится плавающим, но **не выходит из тайла**. Как только ты переключаешься на другое окно — оно возвращается на своё место. Это идеально для диалогов, временных окон или просмотра чего-то поверх кода без полного перехода в floating-режим.

Горячие клавиши тоже продуманы:

* Super+J — переключает ориентацию сплита (вертикаль/горизонталь),
* Super+V — делает окно полностью плавающим (редко используется, но бывает нужно для игр или старых X11-приложений).

### **3.3. Внешний вид: не просто «красиво», а функционально**

Многие считают, что в тайловых менеджерах не должно быть эффектов. Но здесь они работают на **юзабилити**:

* **Размытие (blur)** включено с vibrancy = 0.1696 — достаточно, чтобы фон не отвлекал, но не настолько, чтобы терять контекст.
* **Тени** (shadow.enabled = true) чётко отделяют окна друг от друга, особенно когда их много.
* **Градиентная рамка активного окна** — rgba(33ccffee) → rgba(00ff99ee) — не просто «цветасто». Это мгновенная визуальная подсказка: вот здесь сейчас фокус. На тёмном фоне она особенно заметна.
* **Gaps**: gaps\_in = 5, gaps\_out = 20 — дают «воздух» между окнами и краями экрана. Это снижает визуальное напряжение при долгой работе.

Всё это работает без падения FPS, потому что Hyprland использует GPU-ускорение через wlroots и не тянет за собой тяжёлый DE.

### **3.4. Автозапуск: только то, что нужно**

Все сервисы запускаются через exec-once, и **ничего лишнего**:

* waybar & hyprpaper — панель и обои,
* nekoray — сразу на 7-м воркспейсе (правило ниже),
* swaykbdd & — индикатор раскладки для Waybar,
* battery-alert — скрипт, который следит за зарядом и уведомляет,
* map-tablet.sh — настройка графического планшета (Wacom и подобные).

Закомментирован swww для обоев — потому что hyprpaper уже справляется, а swww нужен только если хочешь анимированные или градиентные фоны. Пока не требуется.

### **3.5. Правила для окон — автоматизация повседневных задач**

Самая мощная часть конфига — секция windowrule. Она превращает рутину в автоматику:

* **Telegram** → воркспейс 8,
* **Ghostty** → воркспейс 9 + прозрачность 0.9 (чтобы не «давил» на глаза),
* **Yandex Music** → воркспейс 10 + прозрачность 0.8,
* **Nekoray** → воркспейс 7.

Диалоговые окна (.\*dialog.\*) автоматически становятся плавающими — без этого они ломали бы layout.

**Pavucontrol** и **blueman-manager** не только флоатятся, но и:

* имеют фиксированный размер (35% 30%),
* прикреплены к верхнему левому углу (move 0% 10%),
* **закреплены (pin)** — не исчезают при смене воркспейса.

Это критично: когда настраиваешь звук или Bluetooth, ты не хочешь, чтобы окно «потерялось» при переключении на другой рабочий стол.

Также есть правило suppressevent maximize — оно **игнорирует запросы на максимизацию** от приложений. В тайловом менеджере «максимизировать» окно — бессмыслица, и без этого правила некоторые приложения (особенно XWayland) вели бы себя странно.

Эта глава показывает, что конфигурация — не просто набор настроек, а **продуманная система**, где каждая строчка решает конкретную задачу пользователя. Никакой «магии», только логика и практика.

# **ЗАКЛЮЧЕНИЕ**

В рамках курсового проекта была разработана гипотетическая операционная система с заданными характеристиками. Основная цель работы, заключающаяся в приобретении практических навыков проектирования компонентов операционных систем, была успешно достигнута.

В процессе выполнения проекта был проведен выбор архитектуры вычислительной системы, включающей процессор, оперативную память, жесткий диск и внешние устройства, такие как клавиатура, мышь, монитор, флеш-диск и принтер. Описан однопрограммный режим работы компьютера с учетом специфики управления ресурсами. Разработана архитектура операционной системы на основе экзоядра, что позволило минимизировать размер ядра и обеспечить эффективное управление ресурсами.

В рамках работы детально изучены алгоритмы управления процессами, включая вытесняющий планировщик *LIFO*, алгоритм взаимодействия процессов на основе пекарни Лэмпорта, задачи синхронизации процессов, а также способы предотвращения тупиков с использованием алгоритма банкира. Подсистема управления памятью описана с использованием сегментного распределения, что обеспечивает рациональное использование доступной оперативной памяти.

Изучена организация файловой системы, включающая иерархию каталогов, индексную логическую организацию и физическую структуру, основанную на связанном списке индексов. Разработан пользовательский интерфейс, ориентированный на текстовый терминал, и предложен упрощенный командный язык для взаимодействия с системой. Представлен алгоритм загрузки операционной системы с жесткого диска и описан процесс запуска прикладных программ.

Проект также включает проектирование структуры управляющих блоков базы данных операционной системы с описанием их взаимодействия. В работе дано комплексное описание функционирования операционной системы, включая управление процессами, памятью, файлами и ресурсами.

Выполнение данного проекта позволило не только углубить теоретические знания, но и приобрести практические навыки проектирования и анализа операционных систем. Итоговая работа демонстрирует способность применять современные подходы к разработке, адаптируя алгоритмы и структуры для выполнения конкретных задач. Данный проект стал важным этапом профессионального становления в области операционных систем.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Таненбаум, Э. С., Бос, Х. Современные операционные системы. — 4-е изд. — СПб.: Питер, 2015. — 1120 с.
2. Востокин, С. В. Архитектура операционных систем: учебное пособие. — Самара: Издательство Самарского университета, 2023. — 84 с.
3. Грейбо, С. В., Новосёлова, Т. Е., Пронькин, Н. Н., Семёнычева, И. Ф. Архитектура вычислительных систем: учебное пособие. [Электронный ресурс]. URL: <http://scipro.ru/conf/computerarchitecture.pdf> (дата обращения: 23.12.2024).
4. Studizba. Режимы работы ЭВМ. [Электронный ресурс]. URL: <https://studizba.com/lectures/informatika-i-programmirovanie/vychislitelnye-mashiny-sistemy-i-seti/3697-rezhimy-raboty-evm.html> (дата обращения: 23.12.2024).
5. Studfile. Лекция 7 Режимы работы ЭВМ. [электронный курс]. URL: <https://studfile.net/preview/1701547/> (дата обращения 23.12.2024)
6. Intuit. Основы операционных систем. Лекция 5. Режимы работы ЭВМ. [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/954/397/lecture/9153> (дата обращения: 23.12.2024).
7. HammockDale. Philosophers. [Электронный ресурс]. URL: <https://github.com/HammockDale/Philosophers> (дата обращения: 27.12.2024).
8. Intuit. Основы операционных систем. Лекция 5. Режимы работы ЭВМ. [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/941/31/lecture/980> (дата обращения: 27.12.2024).
9. Guru99. Алгоритм банкира в операционных системах. [Электронный ресурс]. URL: <https://www.guru99.com/ru/bankers-algorithm-in-operating-system.html> (дата обращения: 28.12.2024).
10. Dit.isuct. Операционные системы. Глава 7. [Электронный ресурс]. URL: <https://dit.isuct.ru/IVT/BOOKS/OPERATING_SYSTEMS/OPER12/GLAVA_7.HTM> (дата обращения: 29.12.2024).
11. Yztm.ru. Операционные системы. Лекция 6. [Электронный ресурс]. URL: <https://yztm.ru/epo/epo_l6/> (дата обращения: 28.12.2024).
12. Bibliofond.ru. Операционные системы: основные понятия и функции. [Электронный ресурс]. URL: <https://www.bibliofond.ru/view.aspx?id=519155> (дата обращения: 05.01.2025).
13. Intuit. Основы операционных систем. Лекция 6. [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/631/487/lecture/11059?page=6> (дата обращения: 05.01.2025).
14. Andreyex. Процесс загрузки в операционной системе: вопросы для собеседования. [Электронный ресурс]. URL: <https://andreyex.ru/programmnoe-obespechenie/protsess-zagruzki-v-operatsionnoj-sisteme-voprosy-dlya-sobesedovaniya-po-operatsionnoj-sisteme/> (дата обращения: 06.01.2025).
15. Studfile. Формирование базы данных ОС [Электронный ресурс]. URL: <https://studfile.net/preview/9124585/page:6/> (дата обращения: 06.01.2025)