ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Разработка программы моделирования работы планировщиков памяти и процессов в операционных системах

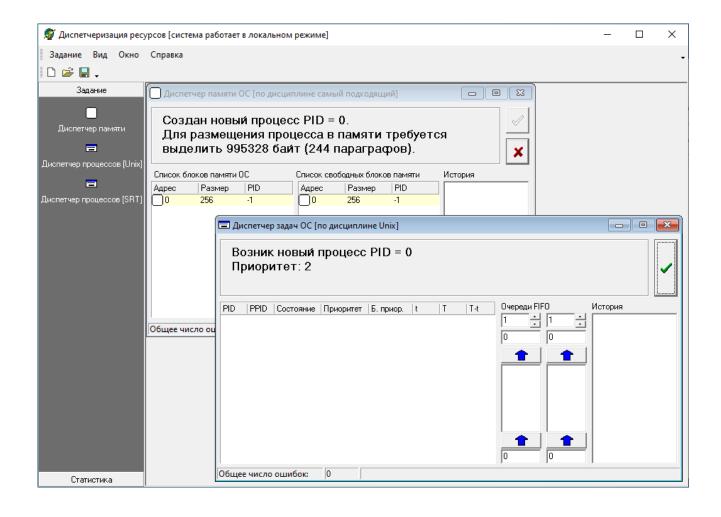
Студент: Рзаев Али Эльдар оглы, группа ИВТ-42 кафедра электронных вычислительных машин

Руководитель: д.т.н., профессор Страбыкин Д. А.

Цель и задачи

- Цель: повышение качества обучения при выполнении лабораторных работ по дисциплине «Операционные системы».
- Задача: разработать программный комплекс для изучения планировщиков памяти и процессов ОС

Текущая программная модель



Основное окно программы

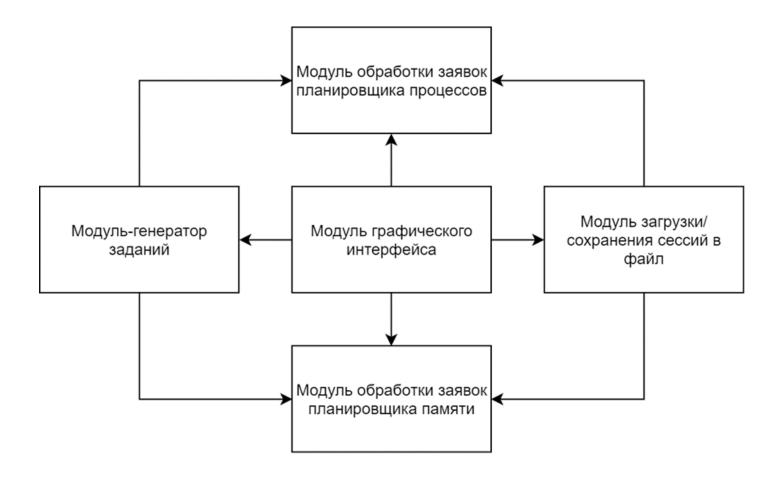
Недостатки

- Приложение доступно только для OC Windows
- Нестабильность: в ходе выполнения лабораторной работы программа несколько раз аварийно завершалась
- Ошибки при проверке пользовательских действий
- Нечеткость, «размытость» интерфейса на дисплеях со сверхвысоким разрешением

Требования к программе

- Возможность генерации заданий
- Возможность сохранения текущей сессии пользователя в файл с возможностью восстановления
- Возможность подсчета количества ошибок, сделанных в ходе выполнения задания
- Возможность просмотра и отмены действий, выполненных в ходе выполнения задания
- Доступность под различные ОС (Windows, macOS и Ubuntu)

Модульная структура



Алгоритмы функционирования

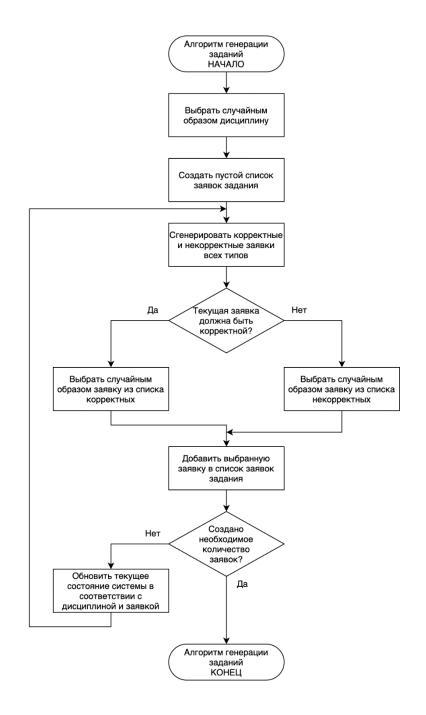
Планировщик памяти

- Алгоритм генерации заданий
- Алгоритмы обработки заявок:
 - Создание нового процесса
 - Завершение процесса
 - Выделение памяти
 - Освобождение памяти

Планировщик процессов

- Алгоритм генерации заданий
- Алгоритмы обработки заявок:
 - Создание нового процесса
 - Завершение процесса
 - Запрос на ввод/вывод
 - Завершение ввода/вывода
 - Передача управления ОС
 - Истечение кванта времени

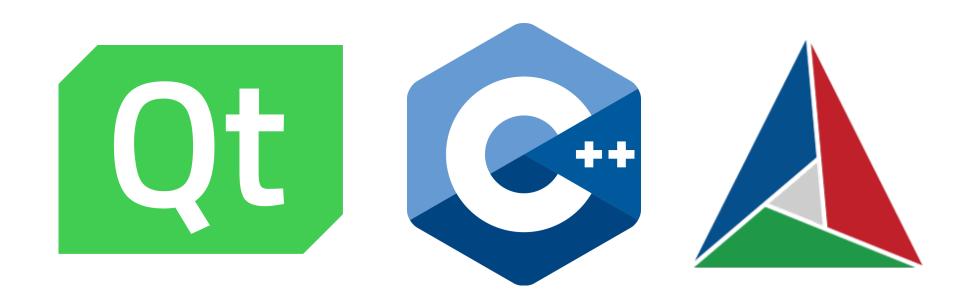
Обобщённый алгоритм генерации заданий



MemoryTask + strategy(): MemAbsStrategy* FreeMemory completed(): uint32 MemRequest MemoryBlock + state(): MemoryState + pid: int32 + address: int32 + requests(): MemRequest[] + address: int32 + create(): MemoryTask* + size: int32 pid: int32, address: int3 MemAbsStrategy + 0..* MemoryTaskWindow type: EnumType MemoryState - ui: void* + pid: int32 request: MemRequest, + blocks: MemoryBlock[] - state: MemoryState state: MemoryState): MemoryState + bytes: int32 + freeBlcoks: MemoryBlock[] - states: MemoryState[createProcess(+ pages: int32 - task: MemoryTask request: CreateProcess, state: MemoryState): MemoryState currentRequest: int32 LeastAppropriateStrategy pid: int32, bytes: int32, terminateProcess(pages: int32 request: TerminateProcess.): AllocateMemory* state: MemoryState): MemoryState processActionAllocate(index: int32): void FirstAppropriateStrategy - processActionFree(index: int32): void + allocateMemory(request: AllocateMemory, - processActionDefragment(): void state: MemoryState): MemoryState + pid: int32 MostAppropriateStrategy - processActionCompress(index: int32); void + bytes: int32 request: FreeMemory updateCurrentRequest(index; int32); void state: MemoryState): MemoryState + pages: int32 nextRequest(): void - ui: void* - refresh(): void pid: int32, bytes: int32, state: MemoryState): MemoryState pages; int32 # findFreeBlock(blocks: MemoryBlocks[], openTasks(): void - provideContextMenu(point: "QPoint): void saveTasks(): void updateMainView freeBlocks: MemoryBlocks[] state: MemoryState. createTasks(): void request: MemRequest): void TerminateProcess compressAllMemory(showHelp(): void state: MemoryState): MemoryState count: int32, total: int32, fails: int32 - loadTasks(tasks: Task[]): void request: AllocateMemory. - updateStrategyView(strategy: EnumType): void): TerminateProcess state: MemoryState, createProcess: bool): MemoryState AboutWindow - updateHistoryView(task: MemoryTask): void - ui: void* setRequest(request: MemRequest): void ProcessesTaskWindow Process ProcRequest ProcessesState + pid: int32 - ui: void* + processes: Processf1 + ppid: int32 queues: int32[][] + prority: int32 - states: ProcessesState[] basePrority: int32 - task: ProcessesTask CreateProcess timer: int32 - currentRequest: int32 + pid: int32 workTime: int32 + task(): ProcessesTask + ppid: int32 + state: EnumType processActionCreate(): void + priority: int32 SjnStrategy - processActionToActive(index: int32): void + basePriority: int32 UnixStrategy - processActionToWaiting(index: int32); void + timer: int32 - processActionToExecuting(index: int32): void ProcAbsStrategy + workTime: int32 - processActionTerminate(index: int32): void + type: EnumType FcfsStrategy TerminateProcess RRStrategy - nextRequest(): void state: ProcessesState + pid: int32 - refresh(): void # schedule(WinNtStrategy state: ProcessesState): <int32, int32> - provideContextMenu(point: "QPoint): void + pid: int32 SrtStrategy state: ProcessesState request: ProcRequest): void TerminateIO count: int32, total: int32, fails: int32 ProcessesTask + pid: int32 strategy(): ProcAbsStrategy* augment: int32 updateStrategyView(strategy; EnumType); void completed(): uint32 updateHistoryView(task: ProcessesTask): void TransferControl fails(): uint32 - setRequest(request: ProcRequest): void actions(): String[] + pid: int32 state(): ProcessesState TimeQuantumExpired + requests(): ProcRequest[] + pid: int32 + create(): ProcessesTask*

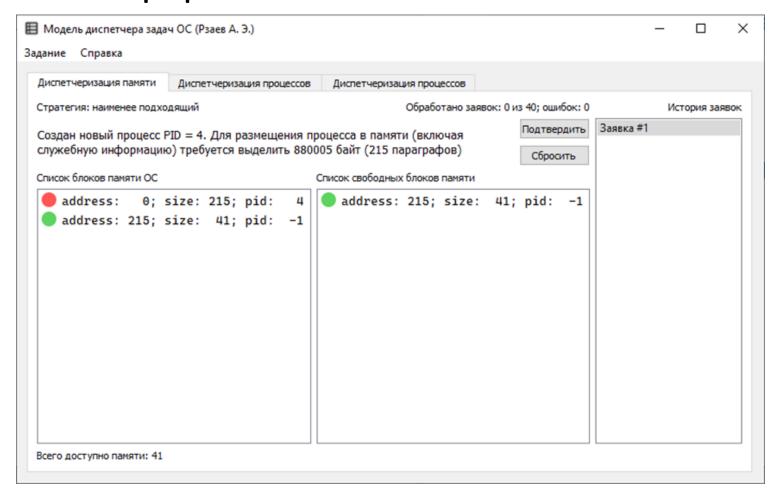
Диаграмма классов

Используемые инструменты разработки

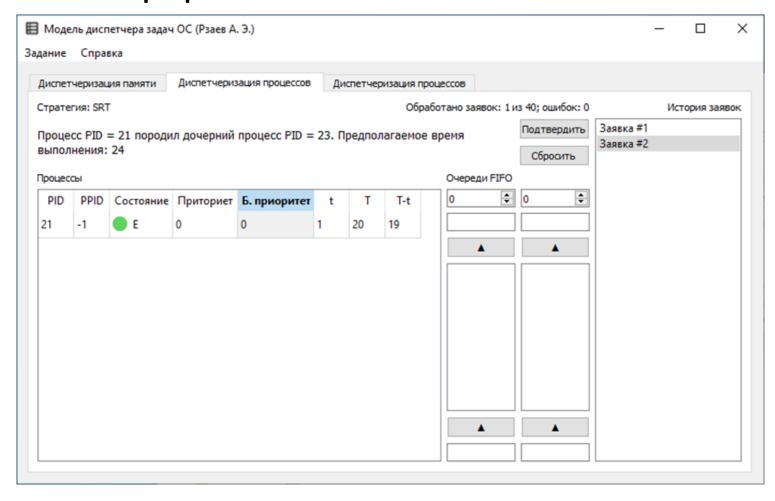


Безопасность

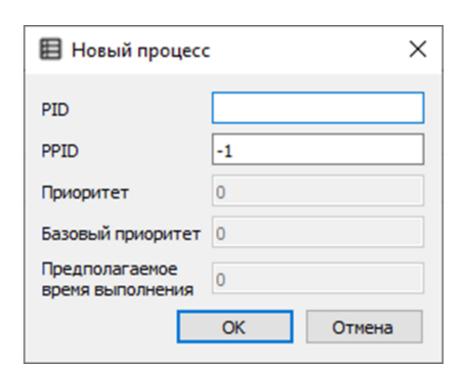
- Сохраняемые файлы сессий шифруются с помощью блочного алгоритма AES-256 в режиме CBC
- Ключ хэш-сумма SHA-256 от Ф. И. О. студента и секретной последовательности в исполняемом файле приложения
- Криптографическая библиотека Randombit Botan



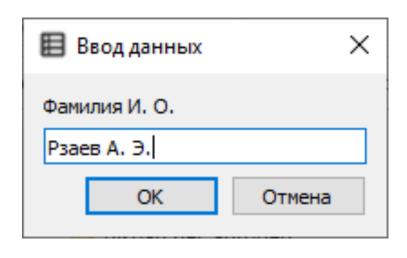
Задание по планировщику памяти



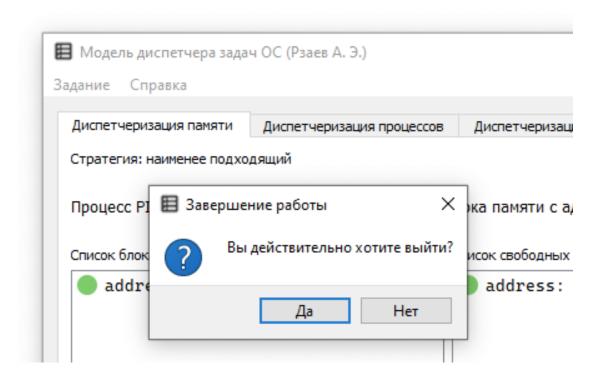
Задание по планировщику процессов



Диалог создания нового процесса



Диалог ввода Ф. И. О. пользователя



Подтверждение выхода из программы

Результаты работы

- Разработана программа «Модель диспетчера задач ОС»
- Обеспечен запуск программы на ОС Windows, macOS и наиболее популярных дистрибутивах Linux
- Проблема с недостаточным количеством заданий решена с помощью генератора заданий
- Устранена «размытость» интерфейса на HiDPI-дисплеях
- В ходе выполнения лаб. работы на новой программе студентами было отмечено:
 - Интерфейс новой программы менее перегружен
 - Время выполнения задания при том же объёме сократилось в 1.5 раза

Спасибо за внимание!