Тема ВКР

Разработка программы моделирования работы планировщиков памяти и процессов в операционных системах.

Введение

Программные модели, которые эмулируют какую-либо систему, достаточно плотно используются на лабораторных работах по дисциплине «Операционные системы». Ни одна лабораторная не проходит без лабораторной установки.
К сожалению, качество некоторых приложений оставляет желать лучшего, что затрудняет изучение нового материала.

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

Например, на первой лабораторной работе изучаются планировщики (или диспетчеры) процессов и памяти ОС
В текущей программной модели пользователю предлагается обработать поступающие в диспетчер процессов заявки.
Для заданий по планировщику памяти заявки бывают следующих видов:

- Создание нового процесса/завершение процесса
- Выделение памяти существующему процессу/освобождение памяти

Пользователь может:

- Выделить/освободить память
- Объединить соседние блоки/дефрагментировать память
- Проигнорировать заявку, ничего не изменяя

Для заданий по планировщику процессов заявки бывают следующих видов:

- Создание нового (дочернего) процесса
- Завершение процесса
- Запрос на ввод/вывод
- Завершение ввода/вывода
- Передача управления операционной системе
- Истечение кванта времени

Пользователь может:

- Создать/удалить процесс
- Перевести процесс в состояние ожидания/готовности/исполнения
- Добавлять/удалять процессы из очередей
- Проигнорировать заявку, ничего не изменяя

Недостатки

В текущей программной модели были найдены следующие недостатки:

- приложение доступно только для ОС Windows. Пользователи других операционных систем должны страдать;
- нестабильность. В ходе выполнения лабораторной работы программная модель несколько раз аварийно завершалась, из-за чего результаты выполненной работы безвозвратно терялись;

- ошибки при проверке пользовательских действий. В некоторых случаях было замечено, что программная модель принимала правильную последовательность действий как ошибочную. Это в совокупности с нестабильностью программы усложняет изучение студентами программной модели и, как следствие, увеличивает время на выполнение лабораторной работы;
- нет генератора заданий;
- нечеткость, «размытость» интерфейса на дисплеях со сверхвысоким разрешением (HiDPI) – ну это уже косметические придирки.

Актуальность

Сначала я попытался исправить часть проблем путем дизассемблирования с помощью IDA Freeware, но это оказалось крайне сложно. Т. к. без этой установки лабораторную не сделать, было принято решение о разработке новой программной модели, повторяющей функционал текущей, в которой будут исправлены вышеописанные ошибки и недостатки.

Требования к программе

Были составлены следующие 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
- Обеспечить просмотр и отмену действий, выполненных в ходе выполнения задания
- Доступность под различные ОС (Windows, macOS и Ubuntu)

Модульная структура

Я решил разбить приложение на 5 модулей:

- Модули обработки заявок планировщиков памяти и процессов. Отвечают за обработку заявок и проверку выполненных пользователем действий
- Модуль-генератор заданий. Отвечает за генерацию заданий с достаточной степенью уникальности для каждого пользователя без необходимости разработки заданий вручную
- Модуль загрузки и сохранения сессий (пользовательских) в файл. Предоставляет возможность сохранять текущее состояние выполняемого задания между запусками приложения, а также обеспечивает защиту от непредвиденного изменения структуры файла
- Модуль графического интерфейса. Является связующим звеном между приложением и пользователем; отображает данные о ходе выполнения задания в текстовом и графическом виде

Алгоритмы функционирования

Были разработаны следующие алгоритмы функционирования:
Алгоритмы генерации заданий
Алгоритмы обработки заявок для планировщика процессов:

- Создание нового процесса
- Завершение процесса
- Запрос на ввод/вывод
- Завершение ввода/вывода
- Передача управления ОС
- Истечение кванта времени

Алгоритмы обработки заявок для планировщика памяти:

- Создание нового процесса

- Завершение процесса
- Выделение памяти
- Освобождение памяти

Обобщённый алгоритм генерации заданий

Разработан обобщенный алгоритм генерации заданий. На каждом шаге генерируются как корректные, так и некорректные задания. Решение о том, какая будет выбрана заявка (корректная или нет) определяется случайным образом, при этом соотношение должно быть 1 к 7. При генерации заявок учитывается состояние системы.

Диаграмма классов

В ходе проектирования была составлена вот такая диаграмма классов.
Разработаны отдельные классы для:

- Дисциплин планировщиков
- Заявок
- Заданий
- Состояния системы (процессы и очереди)
- Справочного, основного окна и виджетов заданий

Инструменты разработки

Приложение должно запускаться на разных ОС и быть простым в запуске. Для этого лучше всего подходит связка Qt и C++. C++ - компилируемый язык, позволяет получить родные для каждой ОС исполняемые файлы. Qt – кроссплатформенный фреймворк, используется для построения GUI, предоставляет удобную IDE QtCreator со встроенным редактором форм.
В качестве системы сборки была выбрана CMake

Безопасность

Для предотвращения несанкционированного доступа (имеется в виду как открытие файла в программе пользователем, который его не сохранял, так и попытка расшифровки файла с последующей обратной шифровкой с другим ключом) содержимое файлов шифруется с помощью симметричного алгоритма AES-256 в режиме CBC. Данный алгоритм хорошо проанализирован, широко используется при разработке программного обеспечения, а также имеет аппаратную реализацию для большинства x86-совместимых процессоров.
Для шифрования файлов сессий была выбрана библиотека Botan. Данная библиотека имеет хорошую документацию, написана на C++ и предоставляет удобные способы подключения к проектам, написанных на C++

Разработка пользовательского интерфейса

Проанализировав интерфейс предыдущей установки, было принято решение внести в него несколько небольших изменений:
Для заданий по обоим планировщикам:

- Добавить кнопку «Сбросить». При нажатии на нее все изменения, сделанные в ходе обработки текущей заявки, будут сброшены;
- Вместо пиктограмм на кнопках будут содержаться соответствующие их смыслу надписи;
- Для кнопок «Подтвердить» и «Сбросить» добавлены клавиатурные сочетания «Alt+Enter» и «Ctrl+Z» соответственно;

Задание по планировщику памяти:

- Для свободных и занятых блоков памяти сделаны более заметные пиктограммы в виде кружков разного цвета.

Задание по планировщику процессов

- Для состояний процессов сделаны более заметные пиктограммы в виде кружков разного цвета.

В диалоговом окне создания нового процесса в зависимости от текущей дисциплины планирования некоторые поля становятся неактивными.
При запуске пользователю выводится диалоговое окно с предложением ввести свое имя. Ввод имени обязателен, так как с помощью него будет осуществлено шифрование сохраняемых файлов. Имя пользователя будет выведено в заголовке окна.
При попытке закрыть основное окно программы будет выводиться диалог подтверждения, чтобы исключить потерю результатов из-за случайного закрытия программы

Результаты работы

Результат работы – лабораторная установка «Модель диспетчера задач ОС». Изначально предполагалось исправление и доработка существующей установки, однако такая задача оказалось крайне сложной в сравнении с разработкой нового приложения.

- 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 (удобный и красивый)
- Время выполнения задания при том же объёме сократилось в 1.5 раза