Одесский национальный политехнический университет

Кафедра «Компьютерные системы управленния»

**КУРСОВАЯ РАБОТА**

по предмету «Современные технологии программирования»

на тему: «Обслуживание процессором ЭВМ очереди готовых заданий»

Вариант 10

Студента 3 курса, группы АТ–183

специальности «Автоматизация и компьютерно-интегрированные технологии»

ФИО: Скрипкин А. А.

Руководитель: доц. Сперанский В. О.

Национальна шкала: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Количество баллов:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка: ECTS:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (Фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (Фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (Фамилия и инициалы)

г. Одеса – 2020 год

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc58637434)

[ЗАДАНИЕ К КУРСОВОЙ РАБОТЕ 4](#_Toc58637435)

[Вариант задания 5](#_Toc58637436)

[РАЗДЕЛ 1. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ 6](#_Toc58637437)

[1.1 Описание проектирования программы 6](#_Toc58637438)

[1.2 Диаграмма работы системы 8](#_Toc58637439)

[1.3 Алгоритмы 9](#_Toc58637440)

[1.4 Объектная модель 9](#_Toc58637441)

[1.5 Библиотеки 9](#_Toc58637442)

[1.6 Описание модулей программы 10](#_Toc58637443)

[1.6.1 Реализация необходимых начальных классов системы 10](#_Toc58637444)

[1.6.2 Вспомогательные классы 11](#_Toc58637445)

[1.6.3 Интерфейсы 11](#_Toc58637446)

[РАЗДЕЛ 2. СПРАВОЧНАЯ ИНФОРМАЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ 12](#_Toc58637447)

[2.1. Краткое описание продукта 12](#_Toc58637448)

[2.2 Руководство по использованию 12](#_Toc58637449)

[ВЫВОДЫ 16](#_Toc58637450)

[ЛИТЕРАТУРА 17](#_Toc58637451)

# ВВЕДЕНИЕ

Данная курсовая работа на тему: “Обслуживание процессором ЭВМ очереди готовых заданий” является теоретической, ориентированной на изучение концепций построения операционных систем и методов управления ресурсами процессора. Изучению этой дисциплины должно предшествовать изучение таких дисциплин, как “Программирование”, “Архитектура ЭВМ”, “Объектно-ориентированное программи-рование”.

Целью курсовой работы является изучение основных методов, используемых при управлении ресурсами в различных операционных системах.

Задачей курсовой работы является получение, как теоретических знаний, так и практических навыков, достаточных для проектирования и программирования системного программного обеспечения современных компьютеров, ознакомление с проблемами моделирования и анализа эффективности функционирования реальных вычислительных систем.

# ЗАДАНИЕ К КУРСОВОЙ РАБОТЕ

Для модели вычислительной системы (ВС) с N-ядерным процессором и мультипрограммным режимом выполнения поступающих заданий требуется разработать программную систему для имитации процесса обслуживания заданий в вычислительных системах.

При построении модели функционирования вычислительной системы должны учитываться следующие основные моменты обслуживания заданий:

- генерация нового задания;

- постановка задания в очередь для ожидания момента освобождения процессора;

- выборка задания из очереди при освобождении процессора после обслуживания очередного задания.

Генерация задания:

Считается, что в распоряжении вычислительной системы имеется N ГБ оперативной памяти для размещения рабочей области процесса и M (3<=m<=5) ресурсов R1, R2,…, Rm, обращение к которым переводит процесс в состояние ожидания.

Генерация нового задания (процесса) может происходить:

- в интерактивном режиме по запросу пользователя

- автоматически системой как случайное событие

Каждый процесс характеризуется:

- именем;

- длиной рабочей области;

- интервалом непрерывного выполнения;

- причиной прекращения непрерывной работы (обращение к ресурсу или завершение работы);

- приоритетом, если он требуется используемым методом планирования процессора.

Перед постановкой задания в очередь имитируется размещения рабочей области процесса в оперативной памяти. В случае невозможности размещения процесс отвергается, в противном случае ему выделяется память и процесс помещается в очередь готовых заданий.

Размещение в ОП происходит одним из трёх методов:

1. первого подходящего;

2. наиболее подходящего;

3. наименее подходящего;

Выборка задания из очереди готовых процессов происходит в момент, когда текущий процесс исчерпал интервал непрерывной работы и освободил CPU.

В случае обращения к ресурсу процесс помещается в очередь к нему, причем время использования ресурса генерируется случайным образом.

В случае завершения процесс удаляется из очереди готовых процессов.

Все очереди к ресурсам обслуживаются алгоритмом FCFS (в порядке поступления). Считается, что в каждый момент времени процесс может обратиться только к одному ресурсу. По окончании работы с ресурсом процесс вновь помещается в очередь готовых заданий, причем генерируется новые интервал непрерывной работы и причина ее прекращения.

## Вариант задания

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n/n |  | Стратегия планирования | Наличие вытеснения | Способ организации очереди | | | | Динамическое повышение приоритета | Критерий вытеснения для SJF | |
|  | упорядоченный  список | не упорядочен.  список | список частично упорядочивается  через t тактов | каждому  приоритету своя очередь |
|  | По интервалу непрерывного выполнения | По оставшемуся времени |
| метод планирования памяти |
| 10. | 1 | HPF | + |  | + |  |  | - |  |  |

Табл.1. Вариант задания

# РАЗДЕЛ 1. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ

## Описание проектирования программы

Для построения программной модели имитации процесса обслуживания заданий в вычислительной системе необходимо построить объектную модель этой системы. Для этого необходимо продумать, какие классы будут использоваться в процессе проектирования, какие у них свойства и какие методы. Необходимо также продумать, сколько будет необходимо объектов каждого класса.

Для данного случая был выбран следующий список основных классов:

1. **Процесс:**

Поля:

• имя;

• идентификационный номер процесса;

• интервал работы в тактах;

• время поступления процесса в систему;

• время работы процесса на центральном процессоре;

• состояние процесса;

• приоритет.

Методы:

• конструктор (устанавливает начальное значение параметров процесса);

• финиширование (осуществляет проверку, завершил ли процесс работу на центральном процессоре);

• увеличение времени работы на центральном процессоре.

1. **Ресурс:**

Поля:

• имя;

• состояние ресурса;

• текущий процесс;

• очередь к ресурсу.

Методы:

• конструктор (устанавливает начальное значение для ресурса);

• визуализация (выводит в окно информацию о текущем состоянии);

• присвоение статуса ресурсу;

• загрузка процесса в центральный процессор.

1. **Очередь к ресурсу:**

Методы:

• конструктор;

• получение первого процесса;

• получение процесса с наивысшим приоритетом;

• проверка на пустоту;

• очистка очереди;

• добавление элемента в очередь;

• удаление элемента очереди;

1. **Планировщик заданий:**

Поля:

• данные о процессоре;

• данные об очереди;

• данные о планировщике памяти

• список завершенных процессов

• список отклоненных процессов

Методы:

• конструктор;

• новый такт (производит изменения в системе при увеличении такта).

• планировка нового задания;

1. **Тактовый генератор:**

Поля:

• номер такта.

Методы:

• конструктор;

• переход на новый такт;

• получение номера текущего такта.

• завершение работы (остановка при выключении системы).

1. **Планировщик памяти:**

Поля:

• список занятых блоков памяти.

Методы:

• конструктор;

• заполнение участка памяти данными;

• очистка блока памяти.

• получение информации об использовании памяти.

1. **Блок памяти:**

Поля:

• адрес начальной ячейки памяти;

• адрес конечной ячейки памяти.

Методы:

• конструктор;

1. **Центральный процессор:**

Поля:

• количество ядер;

Методы:

• конструктор;

• запуск процесса в ядре;

• остановка процесса;

• получение информации о ядрах.

1. **Ядро процессора:**

Поля:

• процессор-родитель;

• состояние ядра;

• текущий процесс;

Методы:

• конструктор;

• запуск процесса (при запуске на процессоре);

• остановка процесса;

• вытеснение процесса из ядра;

• новый такт (производит изменения в системе при увеличении такта).

Данный список является предварительным и будет дополняться в ходе дальнейшего проектирования системы.

## Диаграмма работы системы

Данная диаграмма в общих чертах демонстрирует принцип работы системы



**Рис.1.** Диаграмма работы системы

## Алгоритмы

Стратегия планирования HPF (Highest Priority First) подразумевает выбор планировщиком задания из очереди, имеющего наивысший приоритет, для последующего запуска на процессоре. Наличие в условии задачи возможности вытеснения процессов означает необходимость предусмотреть для планировщика задач функцию замещения обслуживающихся в данный момент процессов новыми. При поступлении в очередь процесса с более высоким приоритетом, чем у запущенных процессов, планировщик должен подать команду процессору на замещение одного из процессов, находящихся в ядрах процессора, только что поступившим процессом. В данном случае, если все ядра процессора заняты, для замещения будет выбираться ядро, обслуживающее процесс с наименьшим приоритетом.

Планирование памяти будет осуществляться по принципу «первого подходящего», то есть, новый процесс будет размещаться в первом свободном блоке памяти подходящего размера.

## Объектная модель



**Рис.2.** Объектная модель системы

## Библиотеки

Для написания данной программы необходимо подключить некоторые библиотеки языка Java. Их перечень представлен в коде:

**JavaFx SDK 15.0.1; // Для работы с GUI**

## Описание модулей программы

### Реализация необходимых начальных классов системы (п.1.1.)

Класс «**Process**»:

Представляет собой объект процесса, выполняемого системой. Обладает случайно сгенерированным именем, временем работы, а также необходимым объемом памяти.

Класс «**Resource**»:

Реализует абстрактный системный ресурс, используемый процессами по команде процессора. Имеет прикрепленную очередь, куда поступают процессы, ожидающие обработки.

Класс «**Queue**»:

Реализация очереди к ресурсу (CPU, R1, … Rm). Имеет необходимые методы выборки (первого процесса, процесса с наивысшим приоритетом), а также методы добавления и удаления процессов.

Класс «**TaskScheduler**»:

Один из важнейших программных модулей системы. Отвечает за планировку задач (случайную генерацию, добавление пользователем). Связывает очередь готовых процессов и центральный процессор. Отдает команды в планировщик памяти для добавления процессов в память и удаления из памяти. Ведет запись в списки отклоненных и завершенных процессов. Предоставляет различные данные о ходе выполнения программы.

Класс «**ClockGenerator**»:

Генератор тактов. При создании экземпляра принимает в качестве параметров список присоединяемых объектов. Это необходимо для подачи сигнала на центральный процессор и планировщик задач, запускающих выполнение основных процессов в системе каждый такт. Имеет возможность настройки скорости переключения на следующий такт («tps» - «ticks per second»), а также функцию паузы. Предоставляет значение системного времени (в тактах, от начала работы программы).

Класс «**MemoryScheduler**»:

Планировщик памяти. Отвечает за выделение памяти под процессы, указанные планировщиком задач. Предоставляет доступ к информации о состоянии памяти.

Класс «**MemoryBlock**»:

Представляет собой блок памяти, размещаемый планировщиком памяти.

Класс «**CPU**»:

Реализует центральный процессор. При поступлении команд от планировщика задач распределяет нагрузку на внутренние ядра. Предоставляет информацию о состоянии ядер.

Класс «**Core**»:

Ядро центрального процессора. Отвечает за симуляцию вычислений при обслуживании процесса. Имеет шанс отправить процесс в очередь к одному из системных ресурсов (R1, … Rm) для дальнейшей обработки, а также шанс генерации ошибки процесса, после чего последует его немедленное завершение.

### Вспомогательные классы

Класс «**Main**»:

Главный класс приложения. Отвечает за запуск программы и ее графического интерфейса. Связывает некоторые части программы.

Класс «**Controller**»:

Необходим для связи реализованной программной модели с графическим пользовательским интерфейсом (использован паттерн MVP – Model-View-Presenter, где данный класс является Presenter’ом).

Класс «**Configuration**»:

Хранит некоторые константы, а также параметры системы.

Класс «**NameGenerator**»:

Генератор имен процессов.

### Интерфейсы

Интерфейс «**ITickable**»:

Интерфейс, реализуемый присоединяемыми к тактовому генератору компонентами. Содержит необходимый метод «tick», вызываемый тактовым генератором. Реализация этого метода классами определяет их действия при переходе тактового генератора на следующий такт.РАЗДЕЛ 2. СПРАВОЧНАЯ ИНФОРМАЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ

## Краткое описание продукта

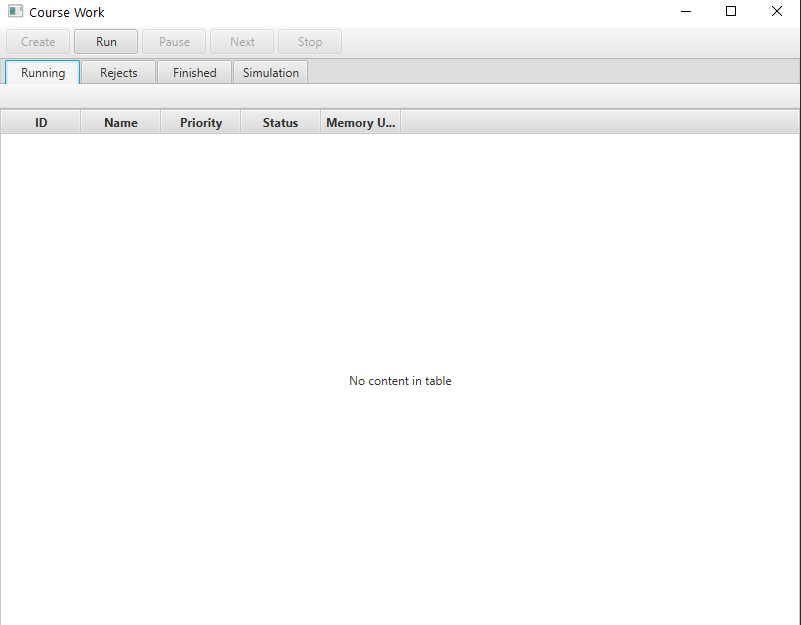
Полученное приложение представляет собой программу симуляции обслуживания процессов ЭВМ.

Функционал приложения включает:

* Автоматическую случайную генерацию задач со случайными параметрами;
* Возможность создания пользовательской задачи вручную;
* Настройку изначальных параметров системы (объем памяти, количество системных ресурсов);
* Динамическую настройку тактовой частоты;
* Динамическую регулировку частоты симуляции ошибок;
* Возможность просматривать очереди к центральному процессору и всем ресурсам;
* Возможность просматривать статистику работы программы;
* Возможность приостановить работу программы.

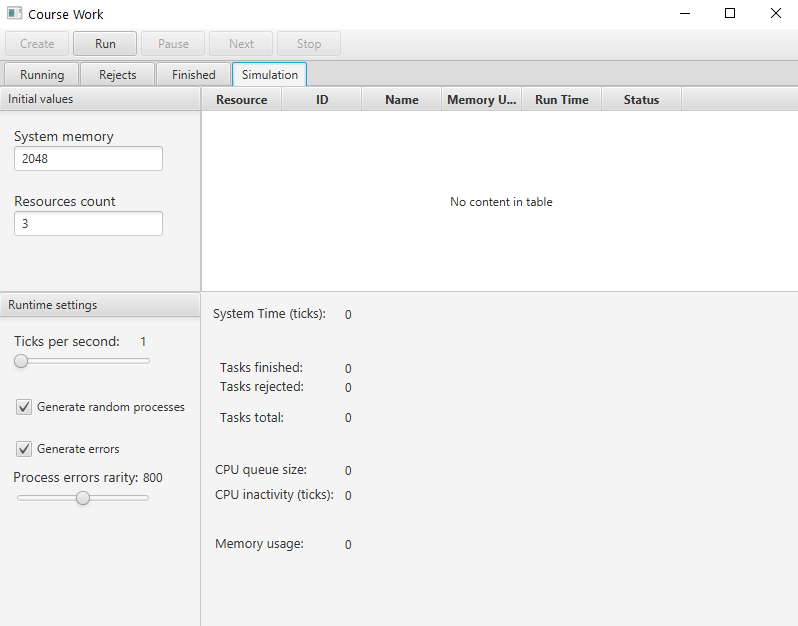
## 2.2 Руководство по использованию

Управление осуществляется с помощью интуитивно понятных кнопок, ползунков и полей для ввода.



**Рис.3.** Исходное состояние программы

Для настройки параметров симуляции, а также отслеживания статистики необходимо перейти во вкладку «Simulation».



**Рис.4.** Вкладка симуляции

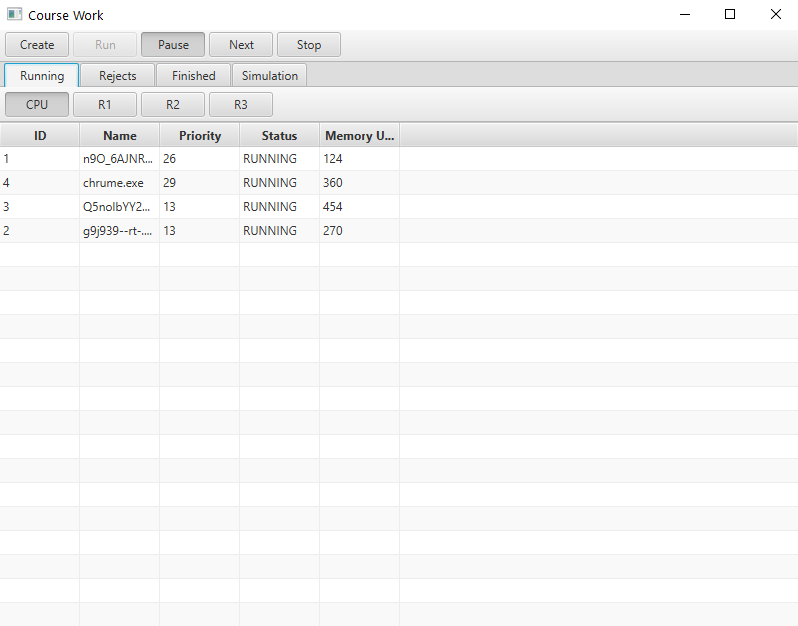
Внутреннее окно «Initial values» содержит поля настройки исходных параметров системы (памяти и количества ресурсов). Эти настройки вступают в силу только после запуска симуляции (кнопка «Run»). Их изменение во время работы симуляции не окажет влияния на параметры системы.

Внутреннее окно «Runtime settings» содержит настройки, применимые как при запущенной симуляции, так и при ее следующем запуске. Ползунок «Ticks per second» отвечает за изменение тактовой частоты тактового генератора. Опция «Generate random processes» отвечает за создание системой случайных процессов и добавление их в очередь. Опция «Generate errors» определяет возможность симуляции ошибок процессов, а ползунок «Process errors rarity» отвечает за их частоту (чем выше значение, тем частота появления ошибок меньше).

Таблица в правом верхнем углу вкладки симуляции содержит информацию о процессах, находящихся в каждом системном ресурсе (ядрах процессора и ресурсах системы).

Под таблицей ресурсов находится панель отображения статистики работы программы, отображающая системное время в тактах, количество отклоненных заданий, количество завершенных заданий (успешных и прерванных), общее количество поступивших в систему заданий, длину очереди к центральному процессору, количество тактов простоя ЦП, а также используемую процессами память.

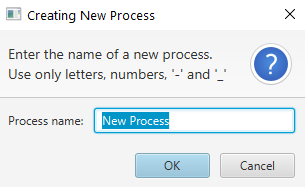
Слева от вкладки симуляции находятся вкладки, отображающие очереди к ресурсам (вкладка «Running»), список отказов (вкладка «Rejects») и список завершенных заданий (вкладка «Finished»), причем вкладка «Running» содержит кнопки для отображения очереди к конкретному ресурсу (CPU, R1, … Rm).



**Рис.5.** Вкладка «Running»

Над списком вкладок имеются кнопки управления состоянием симуляции:

* «Run» - запуск новой симуляции;
* «Pause» - остановить/возобновить симуляцию;
* «Next» - кнопка, активная при остановке симуляции. Позволяет вручную переключиться на следующий такт;
* «Stop» - кнопка прекращения текущей симуляции;
* «Create» - кнопка создания нового процесса. Вызывает диалоговое окно создания процесса.



**Рис.6.** Диалоговое окно создания процесса

Для выхода из приложения можно воспользоваться стандартным «крестиком» в правом верхнем углу окна.

# ВЫВОДЫ

На основании знаний, приобретенных на курсе «Современные технологии программирования», было смоделировано приложение, позволяющее проводить симуляции обслуживания очередей процессов электронными вычислительными машинами и обладающее эргономичным графическим интерфейсом. Благодаря данной курсовой работе были получены навыки созданий симуляций , и работа с графическим интерфейсом.

# ЛИТЕРАТУРА

1. Сетевые операционные системы / В. Г. Олифер, Н. А. Олифер. –СПб.: Питер, 2002. – 544 с.
2. Д. Цикритзис, Ф. Бернстайн. Операционные системы / пер. с англ. –М.: Мир, 1977. –336с.
3. П. Кейлингерт. Элементы операционных систем. Введение для пользователей / пер. с англ. –М.: Мир, 1985. -295с.
4. А. Шоу. Логическое проектирование операционных систем / пер. с англ. –М.: Мир, 1981. –360 с.
5. Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация (+CD). Классика CS. 3-е изд. — СПб.: Питер, 2007. — 704 с: ил.
6. Ахо А., Хопкрофт Д., Ульман Д. – Структуры данных и алгоритмы.: Пер. с англ.: Уч. пос.- М., Издательский дом «Вильямс», 2016. – 400 с.