Національний університет «Одеська політехніка»
Інститут комп’ютерних систем
Кафедра комп’ютеризованих систем управління

КУРСОВА РОБОТА

з дисципліни «Сучасні технології програмування»
за темою: «Емуляція роботи планувальника процесів в операційній системі»
Варіант №19

Студента 3 курса, групи ЛАТ–193
спеціальності «Автоматизація та
комп’ютерно-інтегровані технології»
Прізвище: Хацаюк О. О.
Керівник: доц. Сперанський В. О.

Національна шкала:_____
Кількість балів:_____
Оцінка: ECTS_____

Члени комісії _____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)

м. Одесса – 2021 рік

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
ЗАДАНИЕ
1. СТРУКТУРА ПРИЛОЖЕНИЯ 7
1.1. Модели 7
1.1.1. Process 7
1.1.2. Queue 7
1.1.3. MemoryBlock 8
1.1.4. MemoryScheduler 8
1.1.5. CPU 9
1.1.6. Core 9
1.1.7. Вспомогательные классы 9
1.1.7.1. Utils 9
1.1.7.2. TactGenerator 9
1.1.7.3. GlobalScheduler 10
1.1.8. Configuration 10
1.2. Представления 11
ВЫВОДЫ 12
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 13

ВВЕДЕНИЕ

Операционная система (ОС) – это организованная совокупность программ и данных, которая выполняет функции посредника между пользователями и компьютером. ОС служит двум целям: во-первых, сделать компьютерную систему удобной для использования, и, во-вторых, эффективно использовать аппаратные средства компьютера.

ОС является *управляющей программой*. Управляющая программа контролирует выполнение программ пользователей для предотвращения ошибок и неправильного использования компьютера.

ОС реализует множество различных функций, в том числе:

- определяет так называемый интерфейс пользователя;
- обеспечивает разделение аппаратных ресурсов между пользователями;
- дает возможность работать с общими данными в режиме коллективного пользования;
- планирует доступ пользователя к общим ресурсам;
- обеспечивает эффективное выполнение операций ввода-вывода;
- осуществляет восстановление информации и вычислительного процесса в случае ошибок.

Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами), особенностями использованных методов проектирования, типами аппаратных платформ, областями использования и многими другими свойствами.

В зависимости от особенностей использованного алгоритма управления процессором, операционные системы делят на многозадачные и однозадачные, многопользовательские и однопользовательские, на системы, поддерживающие многопоточную обработку и не поддерживающие ее, на многопроцессорные и однопроцессорные системы.

Важное влияние на облик операционной системы в целом, на возможности ее использования в той или иной области оказывают особенности и других подсистем управления локальными ресурсами — подсистем управления памятью, файлами, устройствами ввода-вывода.

Специфика ОС проявляется и в том, каким образом она реализует сетевые функции: распознавание и перенаправление в сеть запросов к удаленным ресурсам, передача сообщений по сети, выполнение удаленных запросов. Многозадачные ОС подразделяются на три типа в соответствии с использованными при их разработке критериями эффективности:

- системы пакетной обработки (например, ОС ЕС),
- системы разделения времени (UNIX, VMS),
- системы реального времени (QNX, RT/11).

Некоторые операционные системы могут совмещать в себе свойства систем разных типов, например, часть заданий может выполняться в режиме пакетной обработки, а часть — в режиме реального времени или в режиме разделения времени. В таких случаях режим пакетной обработки часто называют фоновым режимом.

Описать операционную систему можно только путем деления ее на меньшие компоненты. Не все ОС имеют одинаковую структуру. Однако во многих современных ОС ставится следующие компоненты:

- управление процессами;
- управление основной (оперативной) памятью;
- управление вторичной (внешней) памятью;
- управление вводом-выводом;
- управление файлами;
- защита системы;
- сетевое обслуживание;

Оптимизация работы ОС с процессами напрямую влияет на быстродействие обработки данных.

ЗАДАНИЕ

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 (5<=m<=10) ресурсов R1, R2,…, Rm, обращение к которым переводит процесс в состояние ожидания.

Генерация нового задания (процесса) происходит автоматически как случайное событие. Каждый процесс характеризуется:

- идентификатором;
- именем;
- приоритетом;
- длиной рабочей области;
- интервалом непрерывного выполнения;

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 методом первого 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 новый интервал непрерывной работы.

n/n	метод планирования памяти	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
19	1	SJF	-			+				

Таблица 1 – Вариант задания №19.

1. СТРУКТУРА ПРИЛОЖЕНИЯ

1.1. МОДЕЛИ

1.1.1. PROCESS

Поля:

- id – идентификатор процесса
- name – имя процесса, генерируется автоматически (“P” + ID)
- priority – приоритет процесса
- duration – время (в тактах), требуемой для выполнения процесса, генерируется автоматически
- memory – объём оперативный памяти, требуемый для выполнения процесса, генерируется автоматически
- startTact – время (такт) вхождения процесса в систему
- burstTime – количество времени (такты), которое процесс находится в обработке
- state – состояние процесса (при создании процесса всегда “READY”)
- core – ядро, загруженное процессом
- byTime – компаратор для сортировки по duration

1.1.2. QUEUE

Поля:

- actualProcesses – очередь процессов, получивших место в оперативной памяти
- rejectedProcesses – очередь процессов, для которых не удалось выделить памяти
- doneProcesses – завершенные процессы
- lastID – идентификатор последнего добавленного процесса

Методы:

- add() - генерация нового процесса и добавление его в очередь

- add(final int processQuantity) – генерация нового процесса и добавление его в очередь N раз.
- isMinDuration(Process process) – проверка, является ли переданный процесс процессом с минимальным временем выполнения

1.1.3. MEMORYBLOCK

Поля:

- start – начало размещения блока памяти
- end – конец размещения блока памяти
- memoryVolume – требуемый объём памяти
- state – состояние блока памяти (по умолчанию – EMPTY)
- id – идентификатор блока памяти
- activeProcessId – идентификатор процесса, принадлежащего блоку памяти
- byEnd – компаратор для сортировки блоков памяти в порядке размещения их в оперативной памяти

1.1.4. MEMORYSCHEDULER

Поля:

- memoryBlocks – список занятых блоков памяти

Методы:

- fillMemoryBlock(Process process) – поиск и заполнение свободного блока памяти в memoryBlocks
- release – освобождение блока памяти
- freePastBlock(int index), freeNextBlock(int index), freeBothBlocks(int index), blockBefore(int index), blockAfter(int index) – вспомогательные методы для метода release, созданные для соблюдения DRY принципа написания хорошего кода (исключения дублирования кода).

1.1.5. CPU

Поля:

- cores – массив ядер процессора (ресурсов)

Методы:

- initCores() – инициализация ядер процессора

1.1.6. CORE

Поля:

- isFree – проверка, является ли ядро занятым каким-либо процессом
- processQuantity – количество процессов на ядре

Методы:

- isFree() – проверка, является ли ядро свободным
- getProcessQuantity() – получение количества процессов на ядре

1.1.7. ВСПОМОГАТЕЛЬНЫЕ КЛАССЫ

1.1.7.1. UTILS

Представляет собой надстройку над классом java.util.Random для удобства работы с ним в рамках проекта.

Поля:

- Random RANDOM – объект класса random

Методы:

- getRandomArrayElement(int size), getRandomNumber(int size), getRandomNumber(int left, int right) – возвращают случайное число в зависимости от переданных параметров

1.1.7.2. TACTGENERATOR

Поля:

- tact – значение текущего такта

- timerTask – увеличивает на единицу значение текущего такта и освобождает память
- timer – вспомогательное поле для timerTask

Методы:

- incTact() – увеличение значения таймера на единицу
- clearTact() – обнуление таймера
- start() – приводит в действие поле timerTask с заданной задержкой

1.1.7.3. GLOBALSCHEDULER

Методы:

- programStart() – запускает основную программу, инициализируя ядра, таймер и класс MemoryScheduler.
- run() – запускает выполнение потока записи значений из очередей процессов

1.1.8. CONFIGURATION

Конфигурационные данные для приложения

Поля:

- MEMORY_VOLUME -
- MIN_PROCESS_MEMORY
- MAX_PROCESS_MEMORY
- MIN_PROCESS_TIME_REQUIRED
- MAX_PROCESS_TIME_REQUIRED
- PRIORITIES_QUANTITY
- PC_CORES
- blockId
- actualProcesses
- rejectedProcesses
- doneProcesses

- activeCores
- memoryBlocks

1.2.ПРЕДСТАВЛЕНИЯ

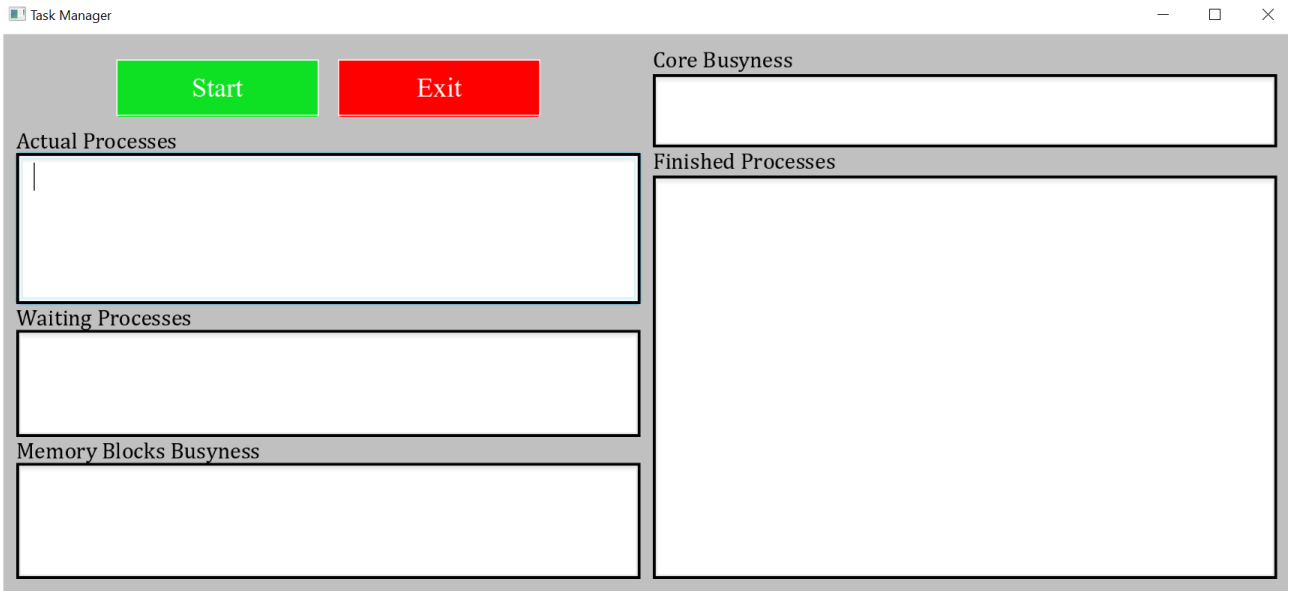

Рисунок 1 – Главное окно

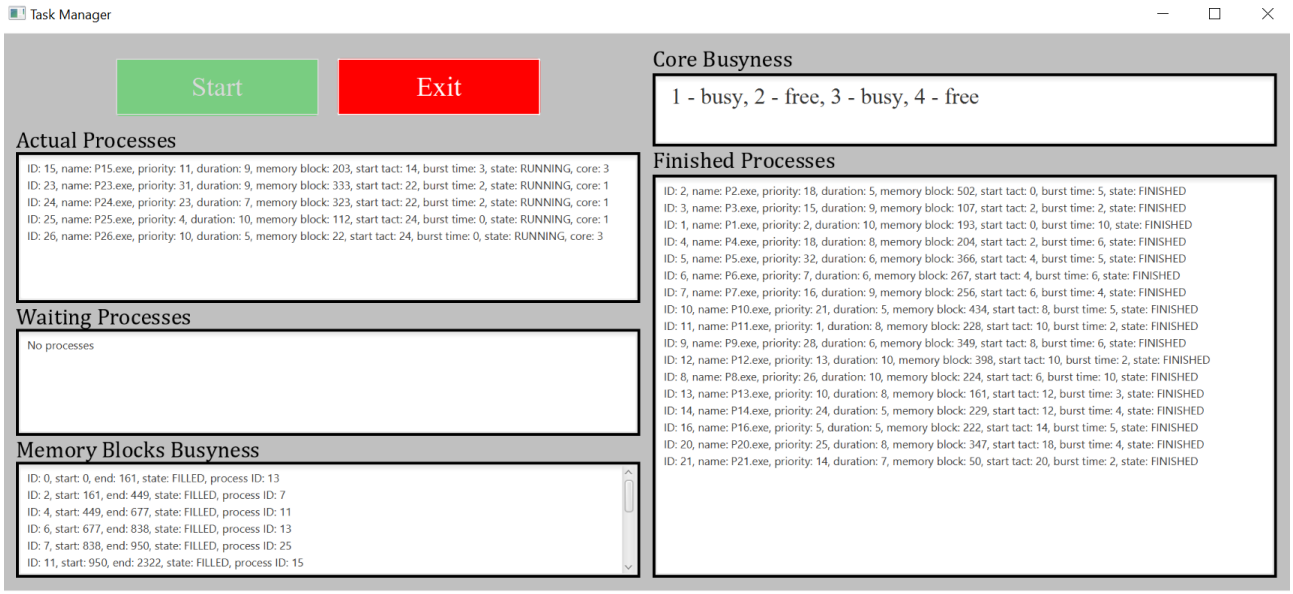

Рисунок 2 – Главное окно во время запуска приложения

ВЫВОДЫ

В результате данной работы было создано приложение, имитирующее работу операционной системы с процессами. При разработке были выполнены следующие требования:

- Метод планирования памяти – первого подходящего
- Стратегия планирования – SJF
- Способ организации очереди – список частично упорядочивается через t тактов

В процессе работы над проектом были закреплены знания, полученные за курс «Современные технологии программирования», а также улучшены навыки объектно-ориентированного программирования и проектирования приложений.

СПИСОК ЛИТЕРАТУРЫ

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.
7. Вайсфельд М. Объектно-ориентирование мышление/ пер. с англ.: Питер, 2014. – 304 с.