Національний університет «Одеська політехніка»

Кафедра комп’ютеризованих систем управління

**КУРСОВА РОБОТА**

З дісципліни «Сучасні технології програмування»

Студента групи НАТ-193

напрям підготовки *6.050201*

спеціальності *7.05020101*

Потьомкіна М. 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)

Одеса – 2021

Оглавление

[Введение 3](#_Toc90243948)

[Задание 5](#_Toc90243949)

[1. Структура приложения 7](#_Toc90243950)

[1.1. Модели 7](#_Toc90243951)

[1.1.1. Process 7](#_Toc90243952)

[1.1.2. Queue 7](#_Toc90243953)

[1.1.3. MemoryBlock 8](#_Toc90243954)

[1.1.4. MemoryScheduler 8](#_Toc90243955)

[1.1.5. CPU 8](#_Toc90243956)

[1.1.6. Core 9](#_Toc90243957)

[1.1.7. Timer 9](#_Toc90243958)

[1.1.8. Scheduler 9](#_Toc90243959)

[1.1.9. Вспомогательные классы 10](#_Toc90243960)

[1.1.9.1. Utils 10](#_Toc90243961)

[1.1.9.2. Configuration 10](#_Toc90243962)

[1.2. Представления 11](#_Toc90243963)

[2. Инструкция 13](#_Toc90243964)

[Выводы 14](#_Toc90243965)

[Список литературы 15](#_Toc90243966)

### 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, OC EC),
* 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 (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.

Таблица 1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 17 | 1 | SJF | - | + |  |  |  |  |  |  |

### Структура приложения

### Модели

### Process

Поля:

* name – имя процесса, генерируется автоматически („P” + ID)
* id – идентификатор процесса, передаётся из Queue
* memory – объём оперативный памяти, требуемый для выполнения процесса, генерируется автоматически (от Configuration.minProcessMemory до Configuration.maxProcessMemory
* time – время (в тактах), требуемой для выполнения процесса, генерируется автоматически

timeIn – время (такт) вхождения процесса в систему

* burstTime – количество времени (такты), которое процесс находится в обработке
* state – состояние процесса (При создании процесса всегда „New”)

### Queue

Поля:

* readyQueue – очередь процессов, получивших место в оперативной памяти
* rejectedQueue – очередь процессов, для которых не удалось выделить памяти
* doneProcesses – завершенные процессы
* PID – количество добавленных процессов в систему

Методы:

* Add(final int PCount) - генерация новых процессов и добавление их в очередь
* addProcess(Process p)
* Remove(Process process) – удаление процесса
* cancelProcess(Process process) – завершает процесс

### MemoryBlock

Поля:

* start – начало размещения блока памяти
* end – конец размещения блока памяти
* byEnd – компаратор для сортировки блоков памяти в порядке размещения их в оперативной памяти

### MemoryScheduler

Поля:

* memoryBlocks – список блоков занятых блоков памяти

Методы:

* findFreeBlock – поиск свободного блока памяти, наиболее подходящего для процесса
* fillMemoryBlock – добавление найденного свободного блока памяти в memoryBlocks
* releaseMemoryBlock – освобождение блока памяти
* clearMemory – освобождение оперативной памяти

### CPU

Поля:

* cores – массив ядер процессора (ресурсов)

Методы:

* setCoreJob(int coreNumb, Process process) – задает работу ядра
* getFreeCore() – получить свободное ядро
* tickEvent()

### Core

Поля:

* isFree – занято ли Core обработкой или свободно
* currentProcess – процесс который находится в обработке на данном ядре

Методы:

* getState() – получение состояния
* getCurrentProcess() – получить текущий процесс

### Timer

Поля:

* tick - такты;

Методы:

* clearTime() – обнуление таймера

### Scheduler

Поля:

* CPU cpu – процессор, находящийся под управлением OС
* MemoryScheduler memoryScheduler – планировщик памяти, управляемый ОС
* Queue queue – очередь процессов
* Timer timer

Методы:

* preLaunchInit – добавление процесса на обработку в ядро
* PDone() – проверка, выполнился ли процесс
* clearOutdated() – очистить
* setJobToCPU()– задать работу процессору
* updateTable()– обновить таблицу
* generateCoreInfo() – генерация информации о ядре

### Вспомогательные классы

### Utils

Надстройка на java.util.Random для большего удобства с ним в рамках проекта.

Поля:

* Random random – объект класса random

Методы:

* getRandomInteger – возвращает случайное число в зависимости от переданных параметров

### Configuration

Конфигурационные данные для приложения

Поля:

* coresNum
* maxMemorySize
* processesOnStart
* removeProcessMultiplier
* removeOldProcessEveryNTicks
* oldNewProcessTime
* minProcessNameL
* maxProcessNameL
* minProcessTimeRequired
* maxProcessTimeRequired
* minProcessMemoryRequire
* maxProcessMemoryRequired
* tickPerSecond

### Представления

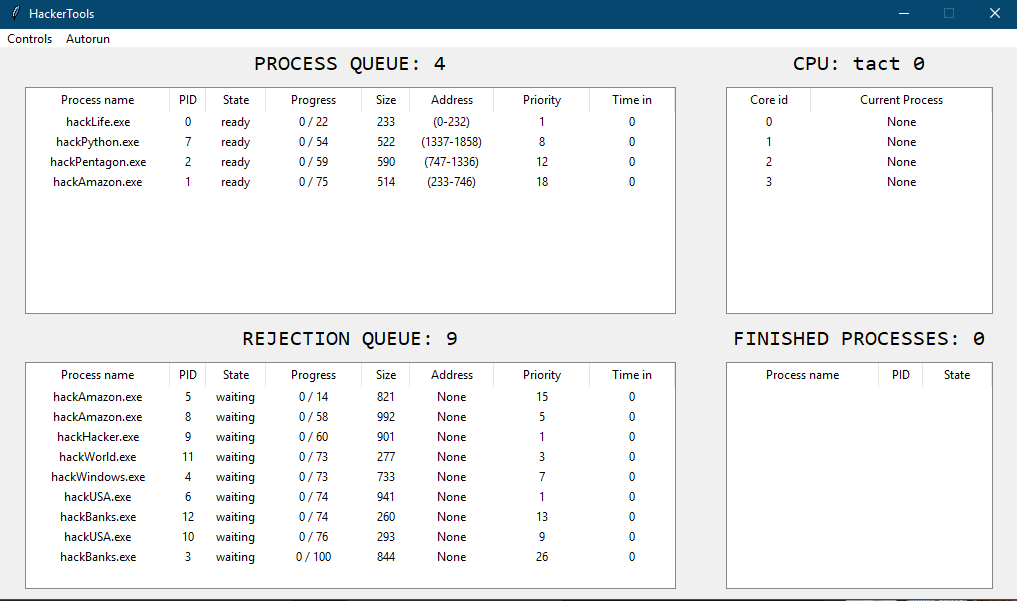
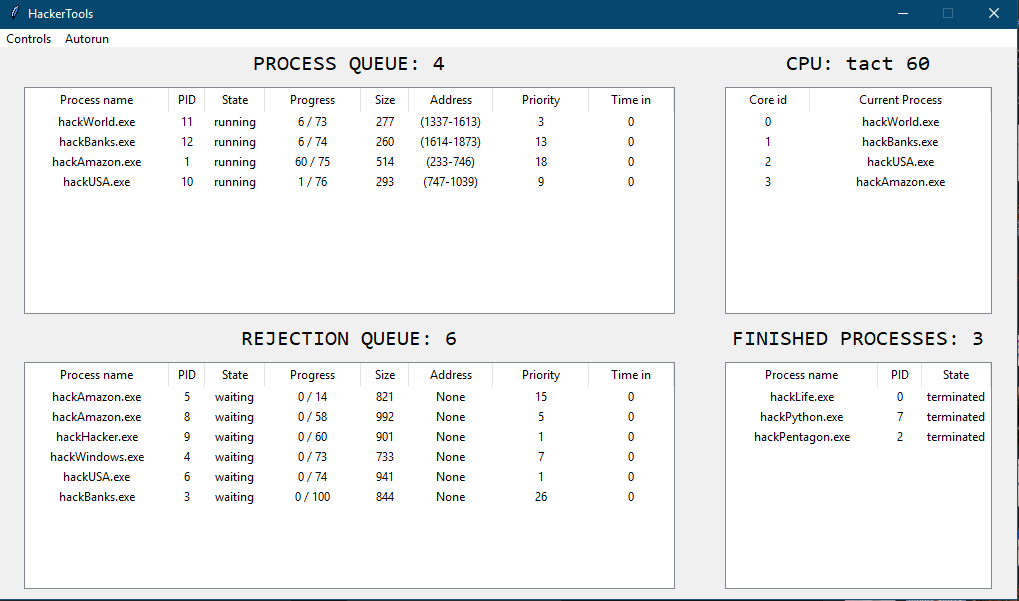


Рис.1. – Такт 0

  
Рис.2 – Такт 60

### Инструкция

Перед началом работы пользователь имеет возможность ознакомится с интерфейсом системы. Главное окно имеет 4 основных вкладки и 3 кнопки:

* Очереди – вкладка представляет собой таблицу очередей текущих процессов.
* Завершенные – таблица завершенных процессов
* Отмененные – таблица отмененных процессов
* Ядра – таблица ядер
* Кнопка «Начать» - запускает процесс генерации случайных процессов, которые поступают в систему.
* Кнопка «Пауза» - останавливает процесс генерации.
* Кнопка «Очистить» - очищает систему от процессов.

Для начала работы системы, пользователю необходимо нажать кнопку «Начать», после чего запуститься процесс генерации случайных процессов со случайными входными параметрами.

Далее пользователь может остановить процесс генерации нажатием на кнопку «Пауза». Это остановит генерации, после чего пользователь может просмотреть все процессы, затем выбрать: продолжить генерацию либо закончить её и очистить списки

### 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 так же 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.