УДК 004.414.2

Трубіна Н.Ф. (ОНУ)
Лісіцина І.М. (ОНУ)
к.т.н., доц. **Камєнєва А.В.** (НУ ОМА)

НАВЧАЛЬНИЙ СИМУЛЯТОР ПЛАНУВАННЯ РЕСУРСІВ ОПЕРАЦІЙНОЇ СИСТЕМИ

У статті представлено симулятор планування ресурсів обчислювальної системи, призначений для використання в процесі викладання в якості підтримуючого програмного забезпечення (ППО). Загальною проблемою, з якою стикаються викладачі та студенти в області комп'ютерних наук, є складність в досягненні правильного розуміння реальної динамічної природи обчислювальних процесів. У процесі підготовки навчального курсу «Системне програмне забезпечення» з'явилася потреба в системі програмного моделювання поведінки процесів, що відтворює роботу частини операційної системи (ОС). Така система, по-перше, дозволяє наочно ілюструвати алгоритми розподілу ресурсів в багатозадачній операційній системі, а, по-друге, дає можливість студентам виконувати практичні завдання в рамках великого проекту, що сприяє підвищенню інтересу студентів до предмета вивчення і поліпшенню засвоєння матеріалу.

Ідея використання симулятора ОС з візуальним інтерфейсом як інструменту для кращого викладання і засвоєння концепцій і технологій, що застосовуються в сучасних ОС, не нова. Подібні симулятори використовуються в багатьох зарубіжних університетах. Однак відомі системи мають ряд недоліків, а, головне, відсутність відкритого коду. Пропонована програмна система надає можливість моделювання роботи гіпотетичної ОС для використання в якості навчального інструменту. Система дає можливість вибору і настройки методів планування процесів і оперативної пам'яті, збору статистики та аналізу продуктивності алгоритмів. ППО є добре документований зразок проектування і програмного коду. Відкритий програмний код дає можливість використовувати симулятор в якості каркасу для виконання лабораторних завдань. Представлена робота може бути розширена за багатьма напрямками.

Ключові слова: симулятор процесів, операційні системи, планування процесора, планування ресурсів, підтримуюче програмне забезпечення, черга готових процесів.

Вступ. Загальною проблемою, з якою зіткаються викладачі та студенти в області комп'ютерних наук, є складність в досягненні правильного розуміння реальної динамічної природи обчислювальних процесів. Незалежно від того, наскільки надійні знання і здатність спілкування викладача, а також наскільки пильну увагу приділяють студенти вивченню предмета, правильне розуміння матеріалу порушується неявною статичною природою лекцій і презентацій.

Вивчення концепцій, що лежать в основі розробки операційних систем (ОС), включається в більшість програм бакалавріату в області комп'ютерних наук, як в Україні, так і за кордоном. В даному випадку лекції, як правило, обмежуються тільки презентаціями концепцій і механізмів. Реалізація цих механізмів досить складна, і не може бути виконана за мале число занять.

Курс ОС, що складається тільки з теоретичних лекцій, ніяк не гарантує того, що студенти отримають повне розуміння і засвоєння більшості розглянутих понять. Тому, частина програми, як правило, резервується для лабораторних занять і практичних вправ. Найбільш поширені

−невеличкі практичні проекти,

−модифікації на рівні коду операційних систем,

−використання тренажерів (симуляторів).

Короткі проекти, в основному, передбачають взаємодію з оболонкою ОС і програмування з використанням системних викликів. Такими проектами можуть бути програми взаємодії між процесами і програми синхронізації, системні утиліти або будь-який інший класичний алгоритм ОС. Невеликі проекти легко здійснити, і немає сумнівів, що вони

дійсно допомагають студентам краще зрозуміти концепції ОС. Однак, такі проекти, в більшій мірі, призначені для вивчення конкретної ОС, а не загальних положень теорії.

Методи навчання, засновані на внесення змін до початкового коду ОС, стикаються з рядом обмежень. По-перше, вони вимагають доброго попереднього знання комп'ютерної архітектури та програмування на C / C ++ як від викладачів, так і від студентів. По-друге, велика кількість часу, яка необхідна для установки, модифікації і налагодження системи, може порушити весь процес навчання.

У сфері інформатики є тренажери, які підтримують викладання різних дисциплін, таких, як комп'ютерні мережі, технології програмування, комп'ютерна архітектура і операційні системи. Симулятор призначений для створення динамічної і спрощеної моделі реальності. Більшість симуляторів намагається надати середовище, яке дозволяє студентам активно експериментувати з конкретними поняттями, щоб забезпечити краще їх розуміння. Наявність симулятора процесів, який допомагає наочно вивчати алгоритми розподілу ресурсів в багатозадачній ОС, безумовно, сприяє підвищенню інтересу студентів до предмета вивчення і поліпшенню засвоєння матеріалу. Імітація полягає в програмуванні моделі комп'ютерної системи, яка поводиться як реальна. Система імітації - звичайна комп'ютерна програма, яка забезпечує певний тип моделювання концепцій операційної системи.

Використання симуляторів забезпечує ряд переваг. Вони, як правило, не вимагають спеціалізованого обладнання. Чимало симуляторів також забезпечують функціональність, яка недоступна в реальних системах.

Є і проблеми з використанням подібних систем. Зроблені спрощення знижують схожість модельованої і реальної системи. Спрощення також означає, що модельовані системи є тільки наближенням до дійсності. Це може привести до ситуації, коли вони не діють як реальні системи. Однак, особливо на першому етапі навчання, такі системи незамінні.

Аналіз літературних даних і постановка проблеми. Ідея використання симулятора ОС з візуальним інтерфейсом як інструменту для кращого викладання і, відповідно, засвоєння концепцій і технологій, що застосовуються в сучасних ОС, не нова. Зокрема, такий підхід успішно застосовується у відділенні комп'ютерних наук Папського Католицького університету Ріо де Жанейро [1]. Широко використовуються подібні симулятори і в університетах США [2, 3]. На жаль, більшість таких систем недоступні для використання в Україні, а ті, до яких є доступ, мають ряд недоліків. Серед таких недоліків слід виділити незначну кількість реалізованих алгоритмів; неможливість настройки користувачем параметрів цих алгоритмів; мала кількість статистичних показників. Також одним з основних недоліків є відсутність відкритого коду, що унеможливлює розширення таких систем.

Планування є фундаментальною функцією операційної системи. Основна концепція полягає в розподілі ресурсів комп'ютера між декількома процесами. Майже кожен комп'ютерний ресурс планується перед використанням. Одним з найбільш обмежених ресурсів обчислювальної системи є процесорний час. Планування ЦП вирішує, які процеси виконуються при наявності декількох процесів, що виконуються. Планування процесора грає важливу роль в ефективному використанні ресурсів і загальної продуктивності системи.

В даний час існує досить багато симуляторів алгоритмів планування процесора. Деякі з них більш зручні, ніж інші. Деякі симулятори запускаються з командного рядка, в той час як інші мають графічний інтерфейс. Стисло розглянемо деякі з доступних тренажерів.

Симулятор *SOsim* [1] реалізує різні алгоритми планування, як з витисненням, так і без витиснення. Крім того, підтримуються статичний або динамічні пріоритети. SOsim реалізує також управління віртуальною сторінковою пам'яттю. Істотним недоліком системи є недружній інтерфейс. Вибір алгоритму планування і завдання налаштувань без пояснень викладача являє собою рішення головоломки. Процеси генеруються виключно вручну. Немає можливості порівняння різних стратегій.

PSSAV [4] є інструментом, який дозволяє користувачеві виконувати моделювання роботи ОС на системі з одним процесором для декількох процесів, і обчислити їх середній час очікування і середній час обороту. Він підтримує стратегії планування *FCFS, SJF, Round Robin*

(*RR*) і *HPF*. Модель ОС в симуляторі обмежена фіксованим набором ресурсів, що означає відсутність можливості додавання ресурсів (процесори, пам'ять, файли і пристрої) відповідно до вимоги моделювання. Симулятор може працювати на будь-якій платформі, що підтримує систему виконання *Java*.

Симулятор, розроблений у вищій школі прикладної статистики національного інституту управління розвитком Таїланду [5], здійснює імітацію різних алгоритмів планування для одного процесора. Користувач може запустити цей симулятор з зумовленими параметрами планування, також можна налаштувати параметри для набору процесів. Симулятор працює в двох режимах. Перший режим - "Режим імітації", другий називається "Режим практики". У режимі імітації користувач може взаємодіяти з моделюванням під час виконання процесу. Користувач може запускати і зупиняти моделювання в будь-який момент часу. Використання тренажера в режимі імітації дозволяє досягти кращого розуміння концепцій алгоритмів планування процесора. Режим практики призначений для перевірки того, наскільки добре засвоєні ці концепції. Користувач може передбачити, коли і як довго кожен процес буде перебувати в певному стані, і перевірити результат. Одним з недоліків даного симулятора є те, що він обмежується лише традиційними стратегіями планування. Іншим недоліком є те, що він не забезпечує можливості порівняння різних стратегій планування.

Симулятор, який використовується для навчання в інженерному університеті Вальядоліда, Іспанія [6], вигідно відрізняється від інших розширеним набором алгоритмів планування процесора і можливістю їх конфігурації. Однак, цей симулятор не дозволяє покроково відстежувати стан черг, а також отримувати інформацію про окремі процеси. Крім того, даний симулятор обмежується тільки плануванням центрального процесора, що не охоплює інших функцій ОС [7-9].

Недоліками розглянутих симуляторів на погляд авторів статті є обмеження кількості процесів, що надходять в систему, і відсутність можливості порівняння різних стратегій або варіантів стратегій на однаковому наборі подій. Крім того, у багатьох симуляторах не підтримуються алгоритми розподілу пам'яті. Також треба визначити, що використання розглянутих симуляторів не вирішує проблем практичної частини курсу.

Мета і завдання дослідження. Як показує аналіз, використання симуляторів в навчальному процесі дозволяє поліпшити процес навчання. Використання готових симуляторів, тим не менш, не вирішує всіх проблем, оскільки вони вимагають адаптації під потреби конкретного курсу, що, як правило, неможливо. Тому розробка програмної системи, яка надає можливість моделювання роботи гіпотетичної операційної системи для використання в якості навчального інструменту для студентів, є доцільною.

Симулятор ОС повинен дозволяти вибір і налаштування методів планування процесів і оперативної пам'яті, а також дозволяти порівняння ефективності цих методів на однакових наборах подій. При цьому не потрібно фактичне створення або виконання процесів, моделюються тільки системні структури, в яких фіксуються події, що відбуваються з процесами.

Крім того, до розробки програмної системи для моделювання роботи гіпотетичної операційної системи доцільно залучити студентів. Як було зазначено вище, одним з найпоширеніших методів проведення практичних занять по курсу операційних систем є розробка невеликих практичних проектів. Недоліком є те, що при такому підході, студенти не отримують достатнього досвіду з проектування великих систем. Проектування великих програмних систем досить складно вписати в навчальний процес. Виходом з положення є використання підтримуючого програмного забезпечення (ППЗ) [10]. В якості ППЗ може використовуватися модель обчислювальної системи у вигляді симулятора процесів в багатозадачній ОС.

ППЗ в даному випадку є добре документований зразок проектування і програмного коду. Система є відкритою, що дозволяє студентам виконувати завдання в рамках великої системи, відчувши себе її розробниками. ППЗ, серед іншого, надає студентам для вивчення зразки

програмного коду хорошої якості. Такий підхід, крім того, дає студентам досвід командної роботи.

Перелічимо вимоги до цієї системи, вироблені в процесі викладання курсів «Операційні системи» і «Системне програмне забезпечення», а також з урахуванням переваг і недоліків, проаналізованих раніше аналогічних програмних систем:

−симулятор процесів (тренажер для освітніх цілей), в основному, повинен використовуватися для аналізу поведінки і продуктивності алгоритмів планування процесора і оперативної пам'яті;

−в симуляторі повинна бути реалізована імітація появи процесів в системі випадковим чином;

−симулятор не повинен обмежувати число процесів, що надходять в систему;

−можливість завдання параметрів генератора процесів: інтенсивність надходження процесів, обмеження на максимальний час виконання кожного процесу¸ обмеження на максимальний розмір адресного простору процесу, при необхідності кількість допустимих пріоритетів тощо;

−можливість вибору алгоритму планування центрального процесора;

−можливість порівняння різних стратегій планування;

−можливість вибору структури для реалізації пріоритетної черги;

−можливість покроково відстежувати стан черг, а також отримувати інформацію про окремі процеси;

−можливість планування ресурсів;

−можливість вибору способу розміщення процесу в пам'яті;

−наявність відкритого коду, що робить можливим розширення системи.

Опис функціонування обчислювальної системи. Імітаційна модель обчислювальної системи фокусується на кількості і розподілі заявок на планування процесора, величині процесорного часу, необхідного для поточного процесу, і пам'яті, необхідної для чергового процесу, що надійшов.

Процеси надходять в систему з інтервалом, який визначається заданою інтенсивністю відповідно до рівномірного, експоненціального або нормального розподілу. Час обслуговування для чергового процесу також генерується випадковим чином.

Вважається, що в розпорядженні обчислювальної системи є N байт оперативної пам'яті для розміщення робочої області процесу і m ресурсів R_1, R_2, ..., R_m, звернення до яких переводить процес в стан очікування. Перед постановкою завдання в чергу імітується розміщення робочої області процесу в оперативній пам'яті. У разі неможливості розміщення процес відкидається, в іншому випадку йому виділяється пам'ять, і процес поміщається в чергу готових завдань.

Вибірка завдання з черги готових процесів відбувається в момент, коли поточний процес вичерпав інтервал безперервної роботи і звільнив центральний процесор.

На даному етапі в симуляторі не передбачена реалізація віртуальної пам'яті. Це означає, що перед розміщенням процесу в черзі готових завдань необхідно повністю розмістити в оперативній пам'яті весь адресний простір процесу. У разі неможливості такого розміщення через брак пам'яті або фрагментації відбувається відмова у виконанні процесу.

У разі звернення до ресурсу процес поміщається в чергу до нього, причому час використання ресурсу генерується випадковим чином. У разі завершення процес віддаляється з черги готових процесів.

Вважається, що в кожен момент часу процес може звернутися тільки до одного ресурсу. Після закінчення роботи з ресурсом процес знову поміщається в чергу готових завдань, причому генеруються нові інтервал безперервної роботи і причина її припинення.

Модель працює покроково, на кожному кроці збільшується значення таймера, і модифікується стан системи, відображаючи діяльність пристроїв, процесів і планувальників. Модифікація здійснюється виконанням наступних дій:

− викликається генератор процесів (приймається рішення про створення нового процесу в залежності від результату, отриманого від генератора випадкових чисел і налаштувань експерименту);

− в разі появи нового процесу генерується більшість його загальних характеристик (ім'я, ідентифікаційний номер, час обслуговування на центральному процесорі та ін.); потім процес поміщається в чергу готових процесів;

− якщо процесор або ресурс зайнятий деяким процесом, перевіряється, чи не вичерпав процес свій інтервал обслуговування, якщо так, то процес звільняє процесор або ресурс;

− для процесу, який звільнив ресурс, випадковим чином генерується час обслуговування на центральному процесорі; потім процес поміщається в чергу готових процесів;

− для процесу, який звільнив процесор, приймається рішення про завершення або запит деякого ресурсу; для незавершених процесів випадковим чином генерується подальший час обслуговування на ресурсі; потім процес поміщається в чергу до ресурсу;

− для алгоритмів з витисненням, якщо процесор зайнятий, а першим в черзі готових процесів перебуває більш пріоритетний процес, процесор звільняється, а поточний процес поміщається в чергу готових процесів;

− якщо процесор або ресурс вільний, відповідний планувальник вибирає з відповідної черги процес для виконання процесу або дій на ресурсі.

Схема життєвого циклу процесу в симуляторі представлена на рис. 1.

Рис. 1. Життєвий цикл процесу

Проектування та реалізація. Описаний вище симулятор алгоритмів планування процесів реалізований в середовищі програмування *.NET* та є *Windows Form*-додатком. Дії користувача дозволяють налаштовувати параметри симулятора та виконувати певні в ньому операції, а саме: «збереження налаштувань», «робочий такт» та «очищення». Інтерфейс програми реагує на зміни моделі та відображає їх у відповідних елементах управління.

В процесі проектування до додатка висувались такі вимоги:

− можливість зміни типу додатку без перепроектування додатку в цілому (наприклад, при переході від локального додатку до *Web*-додатку);

− можливість відображення різних аспектів симулятора ОС (наприклад, відображати як поточний стан компонентів симулятора, так і статистику роботи системи);

− можливість модифікації моделі предметної області без зміни інтерфейсу.

Іншими словами, інтерфейс програми повинен бути максимально відділений від моделі предметної області, проте повинен реагувати на зміну стану моделі (тобто, стану складових її компонентів) для адекватного відображення її поточного стану.

Опис складових обчислювальної системи. На основі проведеної деталізації було виконано проектування системи. Були специфіковані наступні основні типи:

- модель (містить компоненти системи і моделює дії обчислювальної системи);
- центральний процесор;
- зовнішній пристрій;
- пам'ять;
- процес;
- планувальник центрального процесора;
- планувальник ресурсів;
- менеджер пам'яті;
- черга до ресурсу.

Модель містить компоненти системи і моделює дії обчислювальної системи. Центральний процесор, зовнішній пристрій і пам'ять - основні компоненти обчислювальної системи. Планувальник центрального процесора реалізує алгоритм планування центрального процесора, планувальник ресурсів вибирає завдання з черги до ресурсу і, по закінченні використання ресурсу, повертає завдання в чергу готових процесів. Менеджер пам'яті реалізує метод розміщення процесів в основний пам'яті. Черга з пріоритетами використовується для моделювання черги готових процесів.

Необхідними для аналізу алгоритмів планування центрального процесора є отримані в результаті імітації показники функціонування обчислювальної системи, такі як:

- число тактів імітації;
- кількість завдань що надійшли в систему;
- кількість відмов в обслуговуванні завдань через брак пам'яті;
- кількість завершених завдань;
- середній час очікування для завершених завдань;
- середній час обороту для завершених завдань;
- поточні розміри черг до процесора і ресурсів;
- максимальні довжини черг до процесора і ресурсів;
- кількість тактів простою процесора через відсутність в черзі завдань для обслуговування.

З урахуванням цього до набору типів був доданий ще один – статистика.

Формальна специфікація типу визначає набір операцій. В ході проектування для кожної операції були визначені тип операндів, тип результату, передумови (якщо такі є) і постумови.

На наступному етапі після проведеного проектування системи були побудовані класи, відповідні специфікованим типам. Як було сказано вище, в якості мови програмування був обраний C # [11].

Особливості застосування навчального симулятора при проведенні лабораторних занять. Навчальний симулятор спроектовано таким чином, щоб його базовий варіант служив каркасом для виконання лабораторних завдань в рамках єдиного проекту. Результатом виконання цих завдань буде симулятор з обмеженим набором алгоритмів відповідно до завдання студента. Для успішного виконання завдань студентам потрібно розуміння структури проекту та ключових елементів його реалізації. Серед завдань можна виділити дві основні категорії:

- реалізація алгоритмів планування центрального процесору та інших ресурсів;
- розробка елементів інтерфейсу.

Помітимо, що студентам надається базовий варіант інтерфейсу. Оскільки в інтерфейсі не можуть бути враховані усі особливості усіх алгоритмів, необхідні доповнення є завданням студентів. Більш досвідчені студенти можуть реалізовувати повністю свій інтерфейс, наприклад, на базі *WPF*.

Елементи реалізації навчального симулятора. Зупинимося на деяких значущих з точки зору авторів моментах реалізації.

Як було сказано вище, в програмній реалізації симулятора була поставлена задача відділення інтерфейсу від моделі предметної області. Для досягнення цієї мети в інтерфейс програми (в нашому випадку це об'єкт класу *MainForm*, успадкований від класу *Form*, оскільки симулятор реалізований як *Windows Form*-додаток), додано закрите поле *model*, що реалізує модель предметної області. Об'єкт *model* створюється в конструкторі класу *MainForm*. Призначені для користувача дії, здійснювані з допомогою кнопкового меню у вікні програми, ініціюють подію натискання кнопки (*Click*). Оброблювач цієї події викликає відповідний метод класу *Model*, об'єктом якого і є поле *model*. Таким чином, локалізовано взаємодію моделі предметної області і інтерфейсу додатку. Зміни моделі ініціюють відповідну подію. Реакція інтерфейсу на цю подію за допомогою обробника, що знаходиться в класі *MainForm*, гарантує узгодженість інтерфейсу з поточним станом моделі (рис. 2).

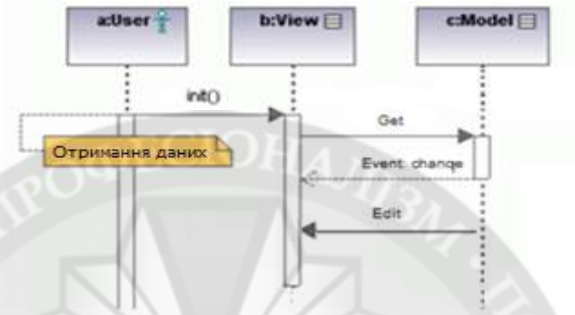

Рис. 2. Каркас додатку

Клас *Model*, діаграма якого наведена на рис. 3, містить всі компоненти симулятора і дані, необхідні для його конфігурації. Для доступу до компонентів використовуються властивості класу, а методи *SaveSettings*, *NextTime* і *Clear* класу *Model* відповідають операціям симулятора.

Метод *SaveSettings* викликається в обробці події *Click* кнопки «Пуск». Як параметр метод отримує масив об'єктів типу *object*, що містить значення параметрів системи, задані користувачем. Параметри системи студенти коректують відповідно завдання.

Метод *NextTime* викликається в обробці події *Click* кнопки «Шаг» і виконує дії, відповідні робочому такту симулятора. Якщо в цьому такті в обчислювальну систему надходить новий процес, він стає в чергу готових процесів, що означає зміну моделі. При цьому, якщо обраний алгоритм планування центрального процесора з витисненням, викликається метод *Session* класу *CPUScheduler*, що виконує перепланування. Цей же метод викликається і в разі, якщо активний процес завершив роботу на центральному процесорі, або обрано алгоритм з квантуванням часу, і процес вичерпав квант роботи. У цих випадках також відбувається зміна моделі. Зафіксовані зміни моделі «запалюють» в кінці такту подію *Changed*, на яку підписано клас *MainForm*. Обробник події *Changed* перемальовує необхідні елементи вікна, щоб привести представлення у відповідність зі зміненою моделлю.

Метод *Clear* викликається в обробці події *Click* кнопки «Стоп» і, в свою чергу, викликає однойменний метод для всіх компонентів моделі. Після очищення можна переналаштувати систему і продовжити роботу симулятора із зміненими параметрами.

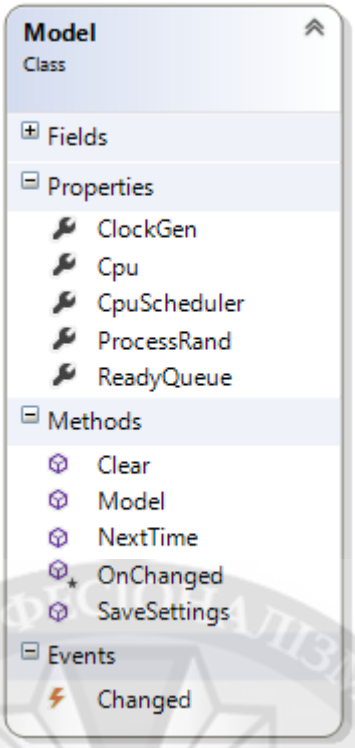

Рис. 3. Діаграма класу *Model*

«Процес» - центральне поняття обчислювальної системи, спроектований і реалізований у вигляді класу *Process*, що реалізує інтерфейс *IComparable*. Похідним від класу *Process* є клас *PriorityProcess*, який використовується в алгоритмах пріоритетного планування процесора. На рис. 4 представлена ієрархія класів *Process* - *PriorityProcess* з перерахуванням властивостей і методів класів.

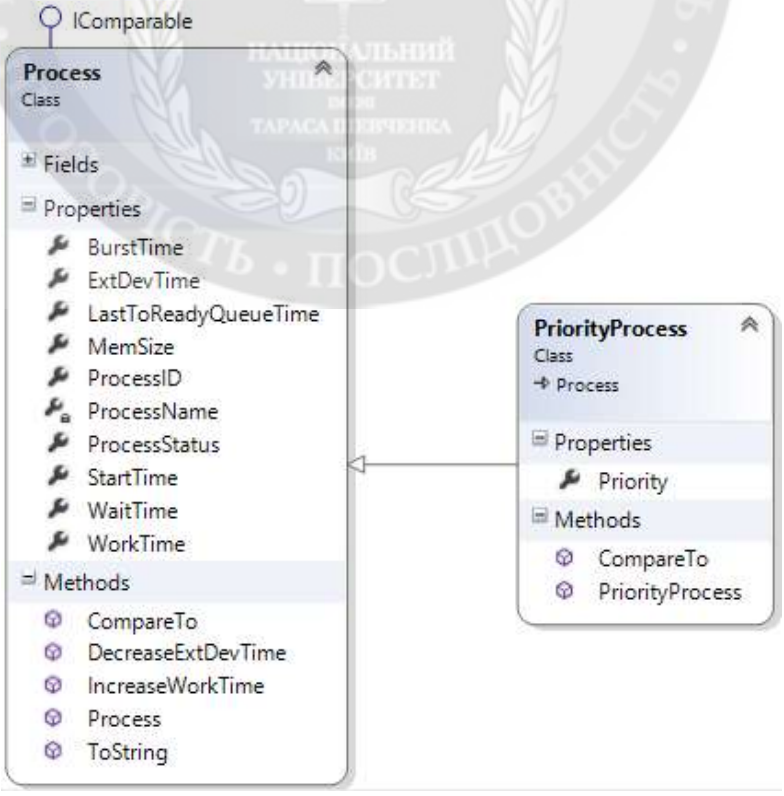

Рис. 4. Ієрархія класів *Process – PriorityProcess*

Методи *IncreaseWorkTime* і *DecreaseExtDevTime* збільшують на одиницю робочий час процесу, що займає центральний процесор, і зменшують на одиницю час використання зовнішнього пристрою (ресурсу) процесором відповідно. Інтерфейс *IComparable* містить єдиний метод *CompareTo*. Цей метод реалізований як віртуальний в класі *Process* і перевизначений в класі *PriorityProcess*. В рамках виконання лабораторних завдань студентам може знадобиться перевизначити цей метод.

Для організації управління процесами необхідно вибрати алгоритм планування. В реальності вибір конкретного алгоритму визначається класом завдань, що вирішуються обчислювальної системою, і цілями, яких потрібно досягти, використовуючи планування [12]. Симулятор, що описується в даній статті, є каркасною моделлю, в якій реалізуються деякі базові алгоритми планування:

–планування в порядку надходження (FCFS);

–пріоритетне планування (HPF) з витисненням і без витиснення і можливістю динамічної зміни пріоритету;

–стратегія SJF з витисненням і без витиснення (по суті окремий випадок HPF);

–карусельна стратегія (RR);

–змішана стратегія (HPF з квантуванням часу).

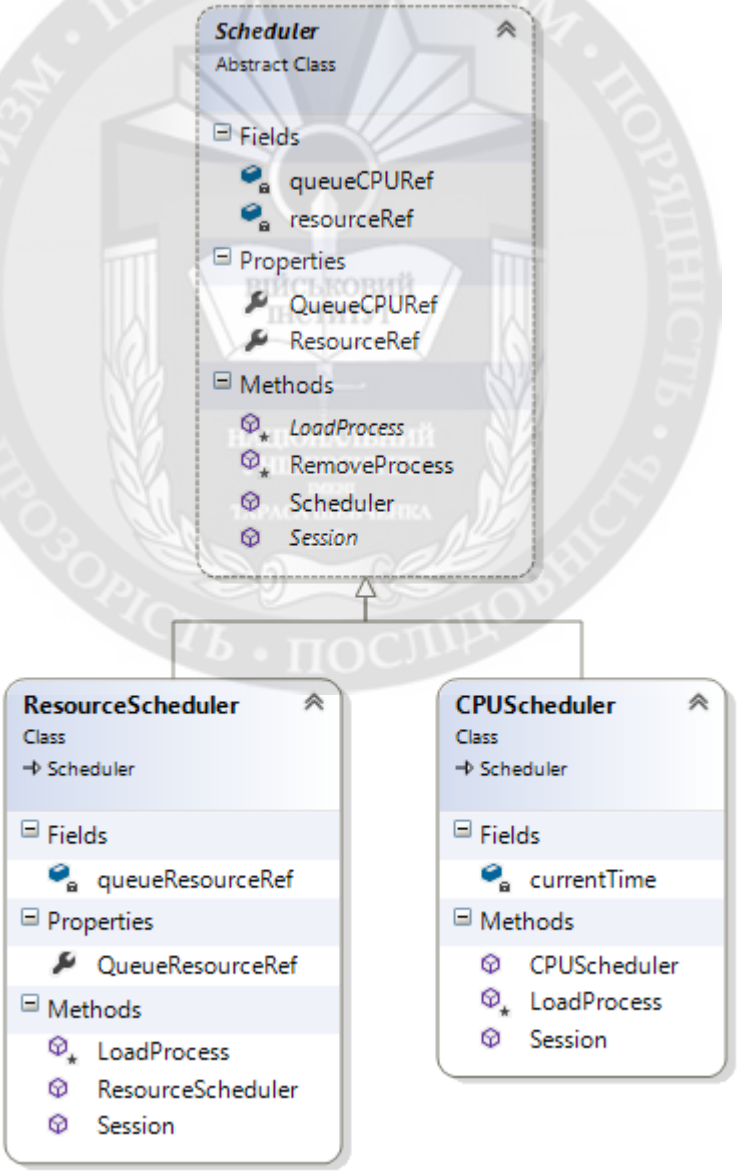

Рис. 5. Ієрархія класів «Планувальник»

Ці базові стратегії використовуються для демонстрації лекційного курсу і перевірки працездатності програмного продукту. В даному випадку, тому що симулятор є навчальним, вибір конкретного алгоритму визначається користувачем. Таким чином, в розглянутому симуляторі є можливість користувачеві задавати алгоритми планування.

Помітимо, що є можливість доповнення функцій симулятора іншими алгоритмами. Таке рішення дає можливість студентам вбудовувати в симулятор модифіковані варіанти алгоритмів або більш складні алгоритми.

Перераховані вище алгоритми реалізуються в класі *CPUScheduler* (Планувальник процесора), який успадкований від абстрактного класу *Scheduler*. Наявність ієрархії обумовлена тим, що в системі передбачені також планувальники ресурсів, які реалізуються об'єктами класу *ResourceScheduler*. Ієрархія класів представлена на рис. 5.

Алгоритм планування реалізується методом *Session* з використанням закритих методів *LoadProcess* і *RemoveProcess*.

Черга готових процесів, як правило, являє собою пріоритетну чергу, що, як відомо, може бути реалізована різними структурами даних. В базовому варіанті симулятора реалізована черга з пріорітетами на базі відсортованого масива. В завдання студентів входить власна реалізація цієї черги. Наприклад, вони можуть використовувати наступні структури даних:

- масив;
- двонаправлений список;
- відсортований двонаправлений список;
- бінарна купа (піраміда);
- хеш-таблиця;
- двійкове (бінарне) дерево пошуку;
- комбінація відсортованого і простого списків.

Для реалізації черги з пріоритетами, перш за все, був написаний інтерфейс *IQueueable <TItem>* (рис. 6), в якому перелічені стандартні операції над чергою (*Put, Remove* і *Item*), операції *Clear* і *Count*, а також операція *ToArray*, що повертає вміст черги в вигляді масиву елементів типу *TItem*. Необхідність в операції *ToArray* диктується двома причинами:

–системі потрібен доступ до всіх елементів черги (а не тільки до першого, що відповідає функціональній специфікації) для виявлення «завислих» процесів з метою підвищення їх пріоритетів;

–черга до процесору повинна повністю відображатися у вікні симулятора.

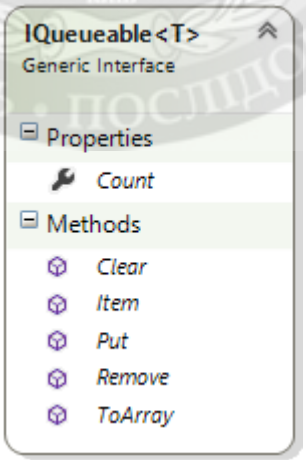

Рис. 6. Універсальний інтерфейс *IQueueable*

Для кожної операції були вказані передумови (якщо такі є) і постумови. Для цього була використана бібліотека *Code Contracts*. Зауважимо, що передумови і постумови були сформульовані для методів всіх розроблюваних класів за допомогою методів *Requires* і *Ensures* класу *Contract*. Використання *Code Contracts* забезпечує автоматичну перевірку

заявлених обмежень і гарантій під час виконання. Крім того, в середовищі розробки *Visual Studio* версії *Premium* або *Ultimate*, контракти можуть бути використані для перевірки правильності коду під час компіляції.

Універсальний (параметризований) клас *PriorityQueue* приймає в якості параметрів тип елементів черги і структуру даних, яка використовується під час її фізичної реалізації. При цьому тип елементів черги повинен реалізувати інтерфейс *IComparable*, а структура даних - інтерфейс *IQueueable <TItem>*. Вимога до структури *TStruct* реалізувати інтерфейс *IQueueable <TItem>* дає можливість реалізувати методи *PriorityQueue* простим викликом однойменних методів *TStruct.*

```
class PriorityQueue<TItem, TStruct>
where TItem: IComparable
where TStruct: IQueueable<TItem>
```

Оскільки інтерфейс *IQueueable <TItem>* заявляє операції, властиві і звичайним (*FIFO*) чергам, була побудована ієрархія, показана на рис. 7.

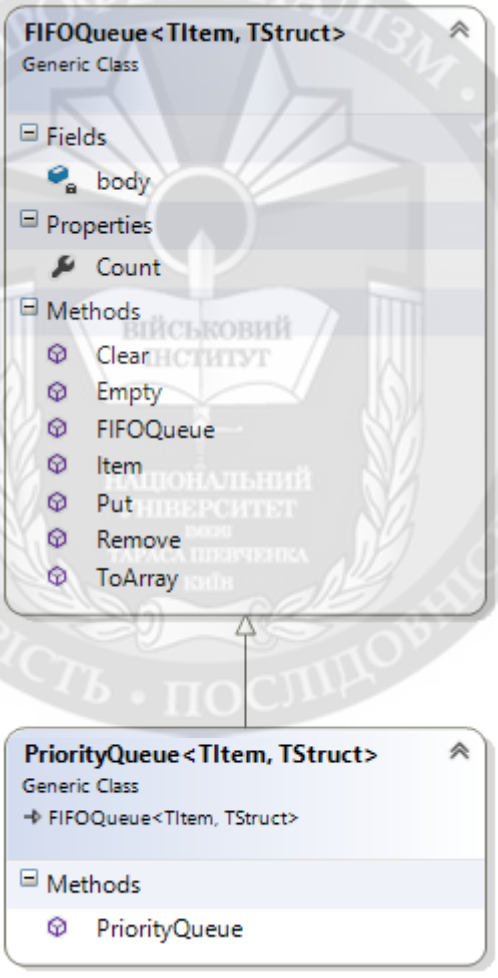

Рис. 7. Ієрархія класів *FIFOQueue – PriorityQueue*

Зауважимо, що формально пріоритетна черга відрізняється від звичайної лише вимогою до типу елементів черги реалізувати інтерфейс *IComparable*. Фактично ж в якості параметра *Tstruct FIFO*-черзі досить передати масив або зв'язний список.

Інтерфейс симулятора. Ще одним етапом реалізації симулятора є розробка інтерфейсу.

Поряд із загальними вимогами, що пред'являються до *Windows*-додатків, при проектуванні інтерфейсу симулятора врахована специфіка, яка в тому числі дозволяє користувачам додатка конфігурувати систему, отримувати оперативну інформацію про стан процесів і збирати статистику по закінченні певного часу. Тому вікно програми розбито на три функціональні частини: настройка конфігурації, відображення оперативної інформації і кнопкове меню. В результаті інтерфейс програми прийняв такий вигляд (рис. 1).

У середній частині вікна програми знаходяться блоки, що відповідають за відображення черг до ресурсів і журналів цих ресурсів (протоколюються дії з процесами, які займають ресурс на кожному тику), головного журналу системи, а також графічного представлення стану оперативної пам'яті. Кнопкове меню, що знаходиться в нижній частині вікна, дозволяє почати роботу системи, зупинити її, керувати роботою в ручному режимі. Статистика, зібрана під час роботи системи, відображається в модальному вікні при натисканні кнопки «Статистика» (рис. 8).

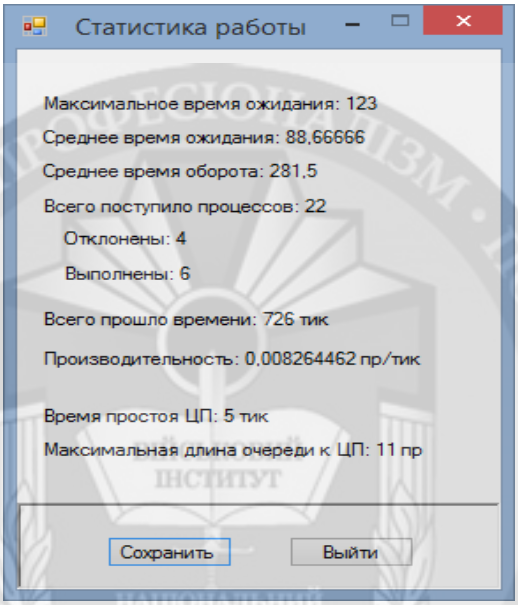

Рис. 8. Вікно статистики

Також передбачена можливість відображення інформації про обраний процес (рис. 9).

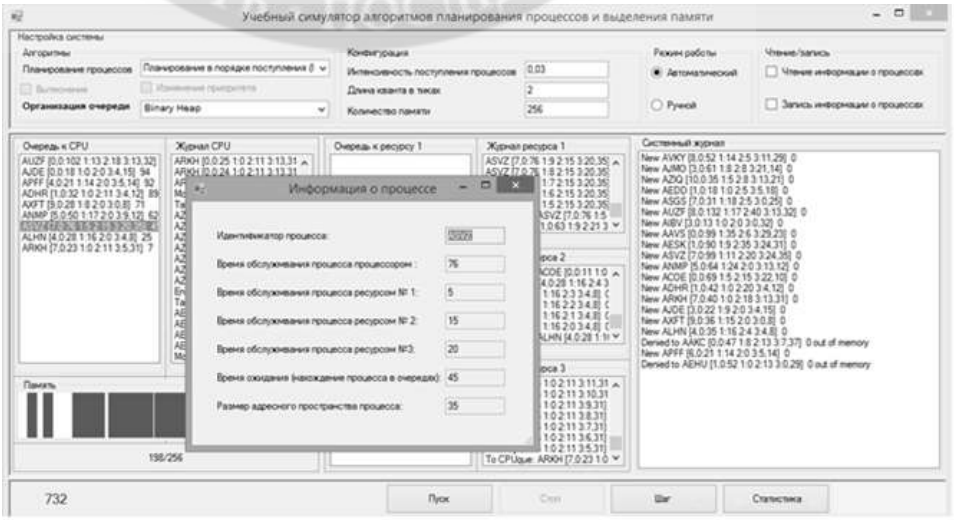

Рис. 9. Отримання інформації про процес під час роботи системи

Таким чином, розглянутий в статті симулятор поведінки процесів в комп'ютерних системах з урахуванням заданих алгоритмів планування та структур для реалізації черги з пріоритетами, є підтримуючим програмним забезпеченням курсу «Системне програмне забезпечення» з наступними характеристиками:

- кількість процесів, що надходять в систему, не обмежена;
- надана можливість завдання параметрів генератора процесів: інтенсивності надходження процесів, обмеження на максимальний час виконання кожного процесу, обмеження на максимальний розмір адресного простору процесу, при необхідності діапазону допустимих пріоритетів тощо;
- надана можливість завдання алгоритму планування процесора і структури для черги з пріоритетами;
- надана можливість налаштування параметрів алгоритмів планування процесора;
- дозволяє порівняння ефективності методів планування процесів на однакових наборах подій.

Використання симулятора для виконання лабораторних робіт можливо в двох напрямках. По-перше, оскільки симулятор надається з відкритим кодом, він може бути розширений додаванням нових алгоритмів планування процесора, а також додаванням нових структур даних для реалізації пріоритетної черги. По-друге, студентам може бути надано каркас додатка, в який необхідно, відповідно до варіанта, вбудувати конкретну структуру для реалізації пріоритетної черги, конкретний алгоритм планування процесора і конкретний метод розподілу пам'яті.

Представлена робота може бути розширена за багатьма напрямками:

- реалізація можливості планування віртуальної пам'яті;
- реалізація можливості вивчення продуктивності додатків в системах реального часу, де завдання мають пріоритети і терміни обмежень;
- реалізація можливості застосування методики планування на розподіленій системі.

ЛІТЕРАТУРА:

1. P. Maia, SOsim: A Simulator Supporting Lectures on Operating Systems, M.S. thesis, IM/NCE, Federal Univ. of Rio de Janeiro, Rio de Janeiro, Brazil, 2001.

2. Sami Khuri, Hsiu-Chin Hsu, ”Visualizing the CPU Scheduler and Page Replacement Algorithms”, ACM l-581 13.085-6/99/0003, 1999, pp.227-231.

3. Richard M. Reese, OSSIM: An Operating System Simulator: http://www.texasacet.org/journal/ACETJournalVol5/OperatingSystemSimulator.pdf.

4. PSSAV. Process Scheduling Simulation, Analysis, and Visualization http://code.google.com/p/pssav/

5. Sukanya Suranauwarat, A Visual and Interactive Learning Tool for CPU Scheduling Algorithms, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 2, March 2013, pp.509-518

6. Manuel Rodríguez-Cayetano. Design and Development of a CPU Scheduler Simulator for Educational Purposes Using SDL System Analysis and Modeling: About Models Lecture Notes in Computer Science Volume 6598, 2011, pp 72-90

7. Abraham Silberschatz and Peter Galvin. Operating System Concepts. Ninth Edition - Wiley Publishing Company. 2013. – 919 pages

8. Таненбаум Э. Современные операционные системы. 4-е изд. – Пер. с англ.– СПб.: Питер, 2017. – 1120 с.

9. Шеховцов В.А. Операционные системы. – СПб.: BHV, ПИТЕР, 2005, 400 с.

10.Лисицына И.Н., Трубина Н.Ф. Использование симулятора процессов вычислительной системы в качестве поддерживающего программного обеспечения // «Інформаційні і управляючі системи та технології». Матеріали VI Міжнародної науково-практичної конференції. – ОНМУ. – Одеса, 2017. – С. 90-92.

11.Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C# – СПб.: Символ-Плюс, 2011. – 768 с.

12.V. Singh, T. Gabba, “Comparative study of processes scheduling algorithms using simulator,” in Int‟l Journal of Computing and Business Research (IJCBR), vol. 4, 2013.

REFERENCES:

1. P. Maia, SOsim: A Simulator Supporting Lectures on Operating Systems, M.S. thesis, IM/NCE, Federal Univ. of Rio de Janeiro, Rio de Janeiro, Brazil, 2001.

2. Sami Khuri, Hsiu-Chin Hsu, ”Visualizing the CPU Scheduler and Page Replacement Algorithms”, ACM l-581 13.085-6/99/0003, 1999, pp.227-231.

3. Richard M. Reese, OSSIM: An Operating System Simulator: http://www.texasacet.org/journal/ACETJournalVol5/OperatingSystemSimulator.pdf.

4. PSSAV. Process Scheduling Simulation, Analysis, and Visualization http://code.google.com/p/pssav/

5. Sukanya Suranauwarat, A Visual and Interactive Learning Tool for CPU Scheduling Algorithms, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 2, March 2013, pp.509-518

6. Manuel Rodríguez-Cayetano. Design and Development of a CPU Scheduler Simulator for Educational Purposes Using SDL System Analysis and Modeling: About Models Lecture Notes in Computer Science Volume 6598, 2011, pp 72-90.

7. Abraham Silberschatz and Peter Galvin. Operating System Concepts. Ninth Edition - Wiley Publishing Company. 2013. – 919 pages

8. Andrew S. Tanenbaum, Herbert Bos. Modern operating systems. 4th Edition. – SPb.: Piter, 2015. – 1120 p.

9. Shexovcov V.A. Operating systems. – SPb.: BHV, PITER, 2005, 400 p.

10. Lisitsyna I.N., Trubina N.F. Using the simulator of the computing system processes as the supporting software // Materials of the VI International Scientific Conference «Information-Management Systems and Technologies» 20th – 22th September, 2017, Odessa, - p. 90-92

11. Martin R., Martin M. Agile Principles, Patterns, and Practices in C# - SPb.: Symbol-Plus, 2011. – 768 p.

12. V. Singh, T. Gabba, “Comparative study of processes scheduling algorithms using simulator,” in Int“l Journal of Computing and Business Research (IJCBR), vol. 4, 2013.

Рецензент: д.т.н., доц. Гунченко Ю.О., Одеський національний університет ім. І.І.Мечникова

Трубина Н.Ф., Лисицына И.Н., к.т.н., доц. Каменева А.В.

УЧЕБНЫЙ СИМУЛЯТОР ПЛАНИРОВАНИЯ РЕСУРСОВ ОПЕРАЦИОННОЙ СИСТЕМЫ

В статье представлен симулятор процессов вычислительной системы, предназначенный для использования в процессе преподавания в качестве поддерживающего программного обеспечения (ППО). Общей проблемой, с которой сталкиваются преподаватели и студенты в области компьютерных наук, является сложность в достижении правильного понимания реальной динамической природы вычислительных процессов. В процессе подготовки учебного курса «Системное программное обеспечение» появилась потребность в системе программного моделирования поведения процессов, воспроизводящей работу части операционной системы (ОС). Такая система, во-первых, позволяет наглядно иллюстрировать алгоритмы распределения ресурсов в многозадачной операционной системе, а, во-вторых, дает возможность студентам выполнять практические задания в рамках большого проекта, что способствует повышению интереса студентов к предмету изучения и улучшению усвоения материала.

Идея использования симулятора ОС с визуальным интерфейсом в качестве инструмента для лучшего изложения и усвоения концепций и технологий, применяемых в современных ОС не нова. Подобные симуляторы используются во многих зарубежных университетах. Однако известные системы имеют ряд недостатков, а, главное, отсутствие открытого кода. Предлагаемая программная система предоставляет возможность моделирования работы гипотетической ОС для использования в качестве учебного инструмента. Система дает возможность выбора и настройки методов планирования процессов и оперативной памяти, сбора статистики и анализа производительности алгоритмов. ППО представляет собой хорошо документированный образец проектирования и программного кода. Открытый программный код дает возможность использовать симулятор в качестве каркаса для выполнения лабораторных заданий. Представленная работа может быть расширена во многих направлениях.

Ключевые слова: симулятор процессов, операционные системы, планирование процессора, планирование ресурсов, поддерживающее программное обеспечение, очередь готовых процессов.

Trubina N.F., Lisitsyna I.N., Ph.D. Kamienieva A.V.

OPERATING SYSTEM RESOURCES SCHEDULER SIMULATOR FOR EDUCATIONAL PURPOSES

This paper presents a computing system resources scheduler simulator, designed as supporting software for educational purposes. The difficulty in achieving the right understanding of dynamic processes is a common problem encountered by computer science lecturers and students. A software process modeling system which would partially simulate an operating system was required for "System software" training course development. Such a system would allow to visually illustrate resources distribution algorithms in a multitasking operating system and give students an opportunity to perform practical assignments within a scope of a complex project, which contributes to greater students' appeal to the subject as well as improving learning results.

The idea to use an operating system simulator with a visual interface as a teaching tool for modern operating systems technologies and conceptions explanation is not a new one. Similar simulators are used by many foreign colleges and universities. However popular systems have a number of shortcomings for the task, the main one is the lack of open source. The presented software system grants an opportunity to model functioning of a hypothetical operating system in order to use as a teaching and training tool. The system allows to select and adjust processes and random-access memory scheduling methods, collect statistics, and analyze algorithms' efficiency. The supporting software represents a well-documented open source example of system design. Open source allows to use the simulator as a frame for practical assignments.

Keywords: resources scheduler simulator, operating system, CPU scheduling, resources scheduling, supporting software, completed processes queue.