МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Інститут математики, економіки та механіки

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

спеціаліста

(освітньо-кваліфікаційний рівень)

на тему Побудова моделі обчислювальної системи з використанням STL
Building a computer model using STL
Построение модели вычислительной системы с использованием STL

Виконав: студент 5 курсу, групи_____
напряму підготовки (спеціальності)
123 – Комп'ютерна інженерія

(шифр і назва напряму підготовки, спеціальності)

Кирило Андрійович
Голіцин К.А.

(прізвище та ініціали)

Керівник Лісіцина І.М.

(прізвище та ініціали)

Рецензент Пенко В.Г.

(прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри
№ 11 від «9» 06.17р.

Завідувач кафедри

(підпис) Є.В. Малахов (прізвище, ініціали)

Захищено на засіданні ЕК № 4

протокол №26 від «22» 06. 2017р.

Оцінка зараховано (70) D

(за 4-х бальною шкалою, за шкалою ECTS, бал.)

Голова ЕК

(підпис) О.О. Арсірій (прізвище, ініціали)

Одеса - 2017

АНОТАЦІЯ

Загальною проблемою, з якою стикаються викладачі та студенти в області комп'ютерних наук є складність в досягненні правильного розуміння реальної динамічної природи обчислювальних подій. У разі операційних систем, лекції, як правило, обмежуються тільки презентаціями концепцій і механізмів. У даній роботі реалізована модель обчислювальної системи з візуальним інтерфейсом, яка може використовуватися в якості ефективного інструменту підтримки теоретичної частини навчальних курсів, а, завдяки відкритому коду і можливості розширення, задіяна при виконанні лабораторних і курсових робіт. В роботі проаналізовано існуючі алгоритми планування процесів, виділені їхні переваги і недоліки.

АННОТАЦИЯ

Общей проблемой, с которой сталкиваются преподаватели и студенты в области компьютерных наук является сложность в достижении правильного понимания реальной динамической природы вычислительных событий. В случае операционных систем, лекции, как правило, ограничиваются только презентациями концепций и механизмов. В данной работе реализована модель вычислительной системы с визуальным интерфейсом, которая может использоваться в качестве эффективного инструмента поддержки теоретической части учебных курсов, а, благодаря открытому коду и возможности расширения, задействована при выполнении лабораторных и курсовых работ. В работе проанализированы существующие алгоритмы планирования процессов, выделены их достоинства и недостатки.

ABSTRACT

A common problem faced by teachers and students in computer science is the difficulty in achieving a correct understanding of the real dynamic nature of computational events. In the case of operating systems, lectures are usually limited to presentations of concepts and mechanisms. In this paper, a model of a computer system with a visual interface is implemented, which can be used as an effective tool for supporting the theoretical part of training courses, and, thanks to the open code and the possibility of expansion, is involved in the performance of laboratory and coursework. In the work, existing algorithms of process planning are analyzed, their advantages and disadvantages are highlighted.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ОС – операційна система

ОП – оперативна пам'ять

ППЗ – підтримуюче програмне забезпечення

Планування - визначення того, коли, в який час і якому процесу слід надавати ресурс.

HPF–highest priority first,« найвищий пріоритет - першим».

FCFS–First – Come , First – Served, процесор виділяється тому процесу, який раніше за всіх інших його запросив

RR–Round – Robin, карусельна стратегія або «кругообіг»

ВСТУП

Загальною проблемою, з якою стикаються викладачі та студенти в області комп'ютерних наук є складність в досягненні правильного розуміння реальної динамічної природи обчислювальних подій. Незалежно від того, наскільки надійні знання і здатність спілкування викладача, а також наскільки пильну увагу приділяється студентами вивчення предмета, правильне розуміння подаються концепцій порушується неявній статичної природою лекцій і презентацій. Тому, частина програми, як правило, резервується для лабораторних занять і практичних вправ. Найбільш поширені невеликі практичні проекти.

Невеликі проекти легко здійснити, і немає сумнівів, що вони дійсно допомагають студентам. Недоліком є те, що при такому підході, студенти не отримують достатнього досвіду з проектування великих програмних систем. Таке проектування досить складно вписати в навчальний процес. Виходом з положення є використання підтримуючого програмного забезпечення (ППЗ), яке, серед іншого, надасть студентам зразки хорошої якості для вивчення.

ППЗ в даному випадку має являти собою добре документований зразок проектування і програмного коду. Система повинна бути відкритою, що дозволить студентам виконувати завдання в рамках великої системи, відчувши себе її розробниками. Такий підхід, крім того, дасть студентам досвід командної роботи.

Оптимальним претендентом на роль такого ППЗ є модель обчислювальної системи у вигляді симулятора процесів у багатозадачній ОС. Вивчення концепцій, що лежать в основі розробки операційних систем (ОС), включається у більшість програм бакалаврату в області комп'ютерних наук, як в Україні, так і за кордоном.

Ідея використання симулятора ОС з візуальним інтерфейсом як інструмент для кращого викладу і, відповідно, засвоєння концепцій і технологій, що застосовуються в сучасних ОС не є новою. Зокрема, такий

підхід вдало застосовується у відділеннях 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. Найголовнішим з них є відсутність відкритого коду, що робить неможливою розширення такої системи.

Метою даної роботи є розробка програмної системи, яка надасть можливість моделювання роботи гіпотетичної операційної системи для використання в якості навчального інструменту. Така модель повинна дозволяти вибір і налаштування методів планування процесів і оперативної пам'яті. Для реалізації методів планування студентам повинен бути наданий набір абстрактних класів з чіткими специфікаціями.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести дослідження предметної області;
- провести огляд існуючих аналогів;
- сформулювати вимоги до створюваної системи;
- побудувати загальну архітектуру системи;
- провести проектування інтерфейсів програми;
- виконати програмну реалізацію системи;
- перевірити працездатність програмної реалізації.

ВИСНОВОК

Розроблено модель поведінки процесів в комп'ютерних системах з урахуванням заданих алгоритмів планування. Розглянуто основні алгоритми планування процесів і стратегії розміщення процесів в невіртуальної пам'яті.

Представлена робота може бути розширена за багатьма напрямками:

- для вивчення продуктивності програм в системах реального часу, де завдання мають пріоритети і терміни обмежень;
- для застосування методики планування на розподіленої системі. Розподілена система визначається як набір незалежних комп'ютерів, які з'являються на користувачів системи в одному комп'ютері.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

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 [pdf]. Available: 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. 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.
8. Abraham Silberschatz and Peter Galvin. Operating System Concepts. Seventh Edition - Wiley Publishing Company. 2005. – 944 pages
9. Таненбаум Э. Современные операционные системы. 2-е изд. – Пер. с англ.– СПб.: Питер, 2002. – 1040 с.
10. Шеховцов В.А. Операційні системи. – СПб.:BHV, ПИТЕР, 2005, 400 стр.
11. Стивен Прата.Язык программирование C++. Лекции и упражнения/шестое издание:Пер.с англ-М:,2014-1248 стр.

ДОДАТОК

```
private void MainForm_Load(object sender, EventArgs e)
        {
            lbCPU.DrawMode = DrawMode.OwnerDrawFixed;
            lbQueueCPU.DrawMode = DrawMode.OwnerDrawFixed;
            lbStatistics.DrawMode = DrawMode.OwnerDrawFixed;
            lbResource1.DrawMode = DrawMode.OwnerDrawFixed;
            lbResource2.DrawMode = DrawMode.OwnerDrawFixed;
            lbResource3.DrawMode = DrawMode.OwnerDrawFixed;
            lbQueueResource1.DrawMode = DrawMode.OwnerDrawFixed;
            lbQueueResource2.DrawMode = DrawMode.OwnerDrawFixed;
            lbQueueResource3.DrawMode = DrawMode.OwnerDrawFixed;
        }
        private void readParameters()
        {
            intensityThreshold = (double)nudIntens.Value;
            burstMin = (int)nudBurst.Minimum;
            burstMax = (int)nudBurst.Value;
        }
        private void btnStart_Click(object sender, EventArgs e)
        {
            readParameters();
            model = new Model(intensityThreshold, burstMin, burstMax);

            btnStart.Enabled = false;
            btnNext.Enabled = true;
            btnStop.Enabled = true;

            nudIntens.Enabled = false;
            nudBurst.Enabled = false;

            lblClock.Text = "0";
        }

        private void btnNext_Click(object sender, EventArgs e)
        {
            model.NextTime();

            // Интерфейс
            lblClock.Text = model.ClockGen.Clock.ToString();

            lbQueueCPU.Items.Clear();
            lbCPU.Items.Clear();
            lbResource1.Items.Clear();
            lbResource2.Items.Clear();
            lbResource3.Items.Clear();
            lbQueueResource1.Items.Clear();
            lbQueueResource2.Items.Clear();
            lbQueueResource3.Items.Clear();
            lbStatistics.Items.Clear();

            lbQueueCPU.Items.AddRange(model.CpuQueue.ToArray());
            Process activeProc = model.Cpu.RunningProcess;
            if (activeProc != null)
                lbCPU.Items.Add(activeProc);

            lbQueueResource1.Items.AddRange(model.ResourcesQueues[0].ToArray());
            activeProc = model.Resources[0].RunningProcess;
            if (activeProc != null)
                lbResource1.Items.Add(activeProc);
```

```
            lbQueueResource2.Items.AddRange(model.ResourcesQueues[1].ToArray());
            activeProc = model.Resources[1].RunningProcess;
            if (activeProc != null)
                lbResource2.Items.Add(activeProc);

            lbQueueResource3.Items.AddRange(model.ResourcesQueues[2].ToArray());
            activeProc = model.Resources[2].RunningProcess;
            if (activeProc != null)
                lbResource3.Items.Add(activeProc);

            lbStatistics.Items.AddRange(model.Stat.ToArray());
        }
        private void btnStop_Click(object sender, EventArgs e)
        {
            model.Clear();

            // интерфейс
            lblClock.Text = string.Empty;

            nudIntens.Value = (nudIntens.Minimum + nudIntens.Maximum) / 2;
            nudBurst.Value = nudBurst.Minimum;

            lbCPU.Items.Clear();
            lbQueueCPU.Items.Clear();
            lbStatistics.Items.Clear();

            nudIntens.Enabled = true;
            nudBurst.Enabled = true;

            btnStart.Enabled = true;
            btnNext.Enabled = false;
            btnStop.Enabled = false;
        }
        private void lbQueue_DrawItem(object sender, DrawItemEventArgs e)
        {
            /*if (e.Index != -1)
            {
                string text = this.lbQueueCPU.GetItemText(lbQueueCPU.Items[e.Index]);
                e.DrawBackground();
                using (SolidBrush br = new SolidBrush(e.ForeColor))
                {
                    e.Graphics.DrawString(text, e.Font, br, e.Bounds);
                }

                if ((e.State & DrawItemState.Selected) == DrawItemState.Selected)
                {
                    this.ttSchedQueue.Show(text, lbQueueCPU, e.Bounds.Right,
e.Bounds.Bottom);
                }
                else
                {
                    this.ttSchedQueue.Hide(lbQueueCPU);
                }
                e.DrawFocusRectangle();
            }*/
            DrawToolTip(e, this.lbQueueCPU);
        }
        private void lbCPU_DrawItem(object sender, DrawItemEventArgs e)
        {
            /*if (e.Index != -1)
            {
                string text = this.lbCPU.GetItemText(lbCPU.Items[e.Index]);
                e.DrawBackground();
```

```
                using (SolidBrush br = new SolidBrush(e.ForeColor))
                {
                    e.Graphics.DrawString(text, e.Font, br, e.Bounds);
                }

                if ((e.State & DrawItemState.Selected) == DrawItemState.Selected)
                {
                    this.ttSchedQueue.Show(text, lbCPU, e.Bounds.Right, e.Bounds.Bottom);
                }
                else
                {
                    this.ttSchedQueue.Hide(lbCPU);
                }
                e.DrawFocusRectangle();
            }*/
            DrawToolTip(e, this.lbCPU);
        }
        private void lbStatistics_DrawItem(object sender, DrawItemEventArgs e)
        {
            /*if (e.Index != -1)
            {
                string text = this.lbStatistics.GetItemText(lbStatistics.Items[e.Index]);
                e.DrawBackground();
                using (SolidBrush br = new SolidBrush(e.ForeColor))
                {
                    e.Graphics.DrawString(text, e.Font, br, e.Bounds);
                }

                if ((e.State & DrawItemState.Selected) == DrawItemState.Selected)
                {
                    this.ttSchedQueue.Show(text, lbStatistics, e.Bounds.Right,
e.Bounds.Bottom);
                }
                else
                {
                    this.ttSchedQueue.Hide(lbStatistics);
                }
                e.DrawFocusRectangle();
            }*/
            DrawToolTip(e, this.lbStatistics);
        }
        private void lbQueueResource1_DrawItem(object sender, DrawItemEventArgs e)
        {
            DrawToolTip(e, this.lbQueueResource1);
        }
        private void lbQueueResource2_DrawItem(object sender, DrawItemEventArgs e)
        {
            DrawToolTip(e, this.lbQueueResource2);
        }
        private void lbQueueResource3_DrawItem(object sender, DrawItemEventArgs e)
        {
            DrawToolTip(e, this.lbQueueResource3);
        }
        private void lbResource1_DrawItem(object sender, DrawItemEventArgs e)
        {
            DrawToolTip(e, this.lbResource1);
        }
        private void lbResource2_DrawItem(object sender, DrawItemEventArgs e)
        {
            DrawToolTip(e, this.lbResource2);
        }
        private void lbResource3_DrawItem(object sender, DrawItemEventArgs e)
        {
            DrawToolTip(e, this.lbResource3);
```

```
        }

        private void DrawToolTip(DrawItemEventArgs e, ListBox lb)
        {
            if (e.Index != -1)
            {
                string text = lb.GetItemText(lb.Items[e.Index]);
                e.DrawBackground();
                using (SolidBrush br = new SolidBrush(e.ForeColor))
                {
                    e.Graphics.DrawString(text, e.Font, br, e.Bounds);
                }

                if ((e.State & DrawItemState.Selected) == DrawItemState.Selected)
                {
                    this.ttSchedQueue.Show(text, lb, e.Bounds.Right, e.Bounds.Bottom);
                }
                else
                {
                    this.ttSchedQueue.Hide(lb);
                }
                e.DrawFocusRectangle();
            }
        }

        private Model model;

        // настройки
        private double intensityThreshold; // порог интенсивности поступления процессов
        private int burstMin;              // максимальная величина интервала
обслуживания
        private int burstMax;              // минимальная величина интервала обслуживания
    }
}
```

Model.cpp:

```
class Model
    {
        public Model(double it, int btMin, int btMax)
        {
            clockGen = new ClockGenerator();

            cpu = new Resource();
            resources = new Resource[RESOURCENUM];
            /*for (int i = 0; i < RESOURCENUM; i++)
                resources[i] = new Resource();*/

            cpuQueue = new SortedUnsortedListQueue(); //BinaryHeapQueue();
            resourcesQueues = new LinkedListQueue[RESOURCENUM];
            /*for (int i = 0; i < RESOURCENUM; i++)
                resourcesQueues[i] = new LinkedListQueue();*/

            cpuScheduler = new CPUScheduler(cpu, cpuQueue);
            resourceSchedulers = new ResourceScheduler[RESOURCENUM];
            for (int i = 0; i < RESOURCENUM; i++)
            {
                resources[i] = new Resource();
                resourcesQueues[i] = new LinkedListQueue();
                resourceSchedulers[i] = new ResourceScheduler(resources[i],
resourcesQueues[i]);
```

```
    }

    OnConnect();

    stat = new Statistics();

    intensityThreshold = it;
    burstMin = btMin;
    burstMax = btMax;
}
public void NextTime()
{
    clockGen.NextTime(); // увеличивается номер такта
    Random rnd = new Random();
    if (rnd.NextDouble() < intensityThreshold)
    {
        Process newProcess = new Process(clockGen.Clock);
        newProcess.PStatus = status.readySt;
        // случайным образом в соответствии с порогом получить burstTime;
        newProcess.PutBurstTime(burstMin, burstMax);
        // поставить процесс в очередь к процессору
        cpuQueue.put(newProcess);

        stat.IncreaseProcessNum();
    }
    cpuScheduler.NextTime();
    for (int i = 0; i < RESOURCENUM; i++)
        resourceSchedulers[i].NextTime();

    stat.EvaluateMaxQueueLength(cpuQueue.count());
    if (cpu.Free())
        stat.IncreaseCPUFree();
}
/*private void ToCPUQueue(Process proc)
{
    // случайным образом в соответствии с порогом получить burstTime;
    proc.PutBurstTime(burstMin, burstMax);
    // поставить процесс в очередь к процессору
    cpuQueue.put(proc);
}*/
/*public void Close()
{
    clockGen.Close();
}*/
public Resource Cpu
{
    get
    {
        return cpu;
    }
}
public ClockGenerator ClockGen
{
    get
    {
        return clockGen;
    }
}
public genProcessQueue CpuQueue
{
    get
    {
        return cpuQueue;
    }
```

```
        }
        public Resource[] Resources
        {
            get
            {
                return resources;
            }
        }
        public genProcessQueue[] ResourcesQueues
        {
            get
            {
                return resourcesQueues;
            }
        }
        public Statistics Stat
        {
            get
            {
                return stat;
            }
        }
        public void Clear()
        {
            clockGen.Clear();
            //scheduler.Clear();
            cpu.Clear();
            for (int i = 0; i < RESOURCENUM; i++)
                resources[i].Clear();
            cpuQueue.clear();
            for (int i = 0; i < RESOURCENUM; i++)
                resourcesQueues[i].clear();
            stat.Clear();
        }

        // компоненты модели
        private const int RESOURCENUM = 3;
        private ClockGenerator clockGen;
        private Scheduler cpuScheduler;
        private Scheduler[] resourceSchedulers;
        private Resource cpu;
        private Resource[] resources;
        private genProcessQueue cpuQueue;
        private genProcessQueue[] resourcesQueues;
        private Statistics stat;

        // параметры модели
        private readonly double intensityThreshold; // порог интенсивности поступления
процессов
        private readonly int burstMin;              // максимальная величина интервала
обслуживания
        private readonly int burstMax;              // минимальная величина интервала
обслуживания

        private void ProcessTerminateHandler(status newStatus, Resource res)
        {
            switch(newStatus)
            {
                case status.terminatedSt: // завершение работы процесса на процессоре
                    // корректировка компонента stat
                    stat.IncreaseProcessTerminatedNum();
                    stat.IncreaseWaitTimeForTerminatedProcesses(cpu.RunningProcess);
                    stat.IncreaseCommonTime(cpu.RunningProcess);
                    break;
```

```
                case status.waitingSt:    // переход процесса с процессора на ресурс
                    Random rnd = new Random();
                    int resourceNum = rnd.Next(0, RESOURCENUM);           //
сгенерировать номер ресурса
                    cpu.RunningProcess.ResetWorkTime();                   // сбросить
фактическое время занятости ресурса процессом
                    cpu.RunningProcess.PutBurstTime(burstMin, burstMax);  //
сгенерировать время занятости ресурса процессом
                    resourcesQueues[resourceNum].put(cpu.RunningProcess); // поместить
процесс в очередь к ресурсу
                    //resourceSchedulers[resourceNum].NextTime();
                    break;
                case status.readySt:       // переход процесса с ресурса на процессор
                    res.RunningProcess.ResetWorkTime();                   // сбросить
фактическое время занятости ресурса процессом
                    res.RunningProcess.PutBurstTime(burstMin, burstMax);  //
сгенерировать время занятости ресурса процессом
                    cpuQueue.put(res.RunningProcess);                     // поместить
процесс в очередь к процессору
                    break;
                default: break;
            }
        }
        private void OnConnect()
        {
            cpuScheduler.processTerminateEvent += new
ProcessTerminateEvent(ProcessTerminateHandler);
        }
    }
        }
```