На правах рукописи

Гусев Игорь Сергеевич

ЭФФЕКТИВНЫЕ АЛГОРИТМЫ ДЛЯ ОПЕРАЦИОННЫХ СИСТЕМ, ПОДДЕРЖИВАЮЩИХ РАСПРЕДЕЛЕННУЮ ОБРАБОТКУ И МЯГКОЕ РЕАЛЬНОЕ ВРЕМЯ

05.13.11– «Математическое и программное
обеспечение вычислительных машин,
комплектов и компьютерных сетей»

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Томск – 2004

Работа выполнена в Томском государственном университете

Научный руководитель:
Доктор технических наук, профессор Сущенко С.П.

Официальные оппоненты:
Доктор технических наук, профессор Матросова А.Ю.
Кандидат технических наук, Гриценко Ю.Б.

Ведущая организация: Томский государственный политехнический университет

Защита состоится 7 октября 2004 г. в 10:30 на заседании диссертационного совета Д 212.267.08 при Томском государственном университете (634050, г. Томск, пр. Ленина, 36).

С диссертацией можно ознакомиться в Научной библиотеке Томского государственного университета.

Отзывы на автореферат просьба высылать по адресу: 634050, г. Томск, пр. Ленина 36, Томский государственный университет, ученому секретарю университета Буровой Н.Ю.

Автореферат разослан « 2 » сентября 2004 г.

Ученый секретарь
Диссертационного Совета, д.т.н., доцент А.В. Скворцов

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность проблемы. Применение операционных систем в новых областях, таких как кластерные вычисления, системы реального времени, встроенные система, приводит к необходимости пересмотра функциональности операционных систем и алгоритмов, реализующих эту функциональность. В данной работе проводится исследование существующих алгоритмов функционирования операционных систем, выявляются их преимущества и недостатки, а также предлагаются более эффективные способы решения задач управления ресурсами вычислительной системы.

Целью работы является разработка методов повышения эффективности операционных систем, исследование возможности расширения их функций, а также реализация предложенных методов в модульной универсальной операционной системе, предназначенной как для индивидуального использования, так и в качестве серверной платформы, а также в системах реального времени и встроенных системах.

В рамках вышеуказанной цели выделяются следующие задачи.

1. Разработать подход к планированию задач и реализующий его алгоритм для прогнозируемого распределения процессорного времени при наличии нескольких участников, использующих вычислительную систему, и нескольких задач мягкого реального времени, которые можно было бы применить в выгружаемых операционных системах, в том числе и ВС с SMP-архитектурой.
2. Разработать базовый алгоритм распределения ресурсов по запросам от многих участников системы к нескольким устройствам в рамках всей вычислительной системы. Для алгоритма предъявляются следующие требования: не оказывать значительного не прогнозируемого влияния на остальную часть вычислительной системы, возможность использования в невыгружаемых операционных системах, возможность использования в вычислителях с SMP-архитектурой и проведения оптимизации очередей запросов с целью увеличения эффективности обслуживания.
3. Разработать подход к реализации расширенной терминальной подсистемы и основные организующие алгоритмы, удовлетворяющие следующим требованиям: возможность временного отключения от системы, возможность перехода на другой терминал, независимость от физических и технических возможностей терминала, устойчивость работы при обрыве канала связи с терминалом.
4. Разработать программное обеспечение, реализующее предложенные алгоритмы и подходы.

Методы исследования. При выполнении диссертационной работы использовались системный анализ, теория построения операционных систем, методы алгоритмизации и построения эффективных управляющих структур данных, практические эксперименты.

Научная новизна работы состоит в следующем:

1. Разработан метод планирования задач с прогнозируемым распределением времени процессоров, при интенсивном изменении состояния готовности последних. Данный метод в отличие от известных учитывает при распределении процессорного времени принадлежность потоков процессам, а процессов – пользователям, тем самым, исключая взаимное влияние процессов разных пользователей и разных типов друг на друга.
2. Предложена новая схема взаимодействия основных объектов ядра операционной системы для эффективного выполнения блокировок в выгружаемых системах, основанная на иерархических взаимосвязях между объектами. В отличие от широко распространенных одноранговых этот подход позволяет минимизировать время блокирования локальных объектов, в то время как необходимость блокировать глобальные объекты возникает значительно реже.
3. Предложен и реализован механизм активных интерфейсов для удобного манипулирования модулями ядра, позволяющий захватить ресурс еще до появления его в системе, а также убрать ресурс из системы, даже если он используется. Данный механизм в отличие от существующих позволяет «на лету» производить замену модулей (например, драйверов устройств), не внося значительных изменений в работу системы в целом.
4. Предложен алгоритм распределения частично разделяемых ресурсов для выгружаемых операционных систем с гарантированным максимальным временем пребывания в критических участках, основанный на использовании АВЛ-дерева. В отличие от классических алгоритмов, которые минимизируют среднее время и ничего не гарантируют в наихудшем случае, когда время работы потенциально может быть неопределенно большим, данный алгоритм гарантирует фиксированное время работы в наихудшем случае, что позволяет применять его в системах реального времени.
5. Предложена система расширенных терминалов для взаимодействия пользователя с вычислительной системой. Разработаны алгоритмы функционирования расширенного терминала для выгружаемых операционных систем. Данный подход в отличие от имеющихся сглаживает разницу между локальной и удаленной работой с вычислительной системой, включая в набор терминальных устройств любую периферию (в том числе и виртуальную), позволяет использование в архитектурах вычислительных систем, не поддерживают виртуализацию адресных пространств.

Практическая ценность диссертационной работы. В рамках данной работы была спроектирована и создана универсальная операционная система НИКОС на базе современных достижений и с учетом возросшего количества требований, предъявляемых к системам. В данной системе были опробованы предложенные технические решения, что позволило значительно увеличить гибкость системы и расширить потенциальный круг задач, для

которых эта система может использоваться. ОС НИКОС предназначена для индивидуального использования, а также для встроенных систем и систем реального времени.

Апробация работы. Основные положения диссертации и ее отдельные результаты докладывались и обсуждались на

1. Научно-практической конференции «Наука и образование: пути интеграции», 1998, г. Анжеро-Судженск.
2. Второй всероссийский симпозиум по прикладной и промышленной математике, 2001, г. Йошкар-Ола.
3. Всероссийская научно-практическая конференция «Новые технологии и комплексные решения: наука, образование, производство», 2001, Кемерово.

Объём работы. Диссертация состоит из введения, пяти глав, заключения и списка литературы. Общий объём работы составляет 144 страницы, из них, 10 страниц – список литературы, содержащий 94 наименования. Текст диссертационной работы иллюстрируется 17 рисунками и 5 таблицами.

СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность темы диссертационной работы, дается краткий анализ работ других авторов по оценке алгоритмов операционных систем, формулируется цель работы, приводится краткое изложение полученных в работе результатов.

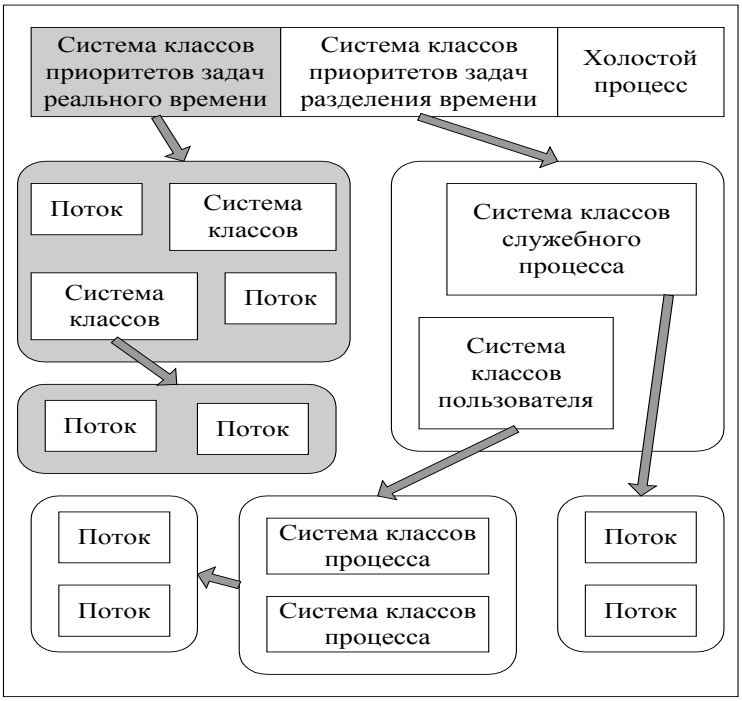

Рис. 1. Пул задач готовых к выполнению

В первой главе проводится анализ функциональности существующих операционных систем и их базовых алгоритмов. Обзор позволил классифицировать как сами системы, так и алгоритмы, а также помог выявить свойственные им недостатки. В существующих операционных системах общего назначения выявлен ряд узких мест, которые значительно ограничивают показатели этих систем при использовании для решения широкого круга задач. Это связанно с высокой трудоемкостью применяемых в них алгоритмов, высокими издержками при блокировках ресурсов и неустойчивой работой в условиях изменения нагрузки. В некоторых случаях применяемые алгоритмы просто не позволяют реализовать некоторые современные требования. Основными такими задачами являются задачи планирования процессорного времени с учетом пользователей, распределения ресурсов и поддержки модульности. В этой главе исследуется возможность построения операционной системы для широкого круга применений от рабочих станций и серверных платформ до встроенных систем и систем реального времени. В конце главы формулируются основные требования к современной операционной системе.

Во второй главе рассматривается задача распределения процессорного времени. Проводится сравнительный анализ существующих алгоритмов планирования. Предлагается алгоритм древовидного планирования с использованием системы классов элементов (структурированное множество элементов для выбора) для нового метода выбора элемента, где используются статические приоритеты. Схема предлагаемого метода приведена на рис. 1. Алгоритм планирования спускается по ветке дерева до потока, которому в последствии будет отдан квант процессорного времени. На верхнем уровне происходит выбор между пулами (системами классов) реального времени и разделения времени (фактически между виртуальными системами реального времени и разделения времени). Затем выбирается подпул (система классов), обычно это подпул задач пользователя. У пользователя выбирается система классов процесса, и в конце концов поток.

С помощью системы классов достигается прогнозируемый выбор элементов независимо от количества элементов. На рис. 2 приводится структура системы классов.

Для реализации предложенного метода используется итеративный алгоритм подсчета количества элементов для выбора и погрешностей. Все элементы разделаются на классы в соответствии с приоритетов, который может быть от 0 до *N*-1. Каждому классу назначается вес – некоторое положительное целое число. Классы сортируются по значению веса. Вводятся следующие переменные:

C_i – число элементов в *i*-м классе;

W_i – вес *i*-го класса;

C_{ij} – число элементов для выбора из *i*-го класса на *j*-й итерации;

O_{ij} – погрешности при вычислении числа элементов для выбора на *j*-й итерации;

i – текущий класс;

j – итерация для *i*-го класса;

$W_{\max}$ – максимальный вес непустого класса;

$C_{i0} = 0$, $O_{i0} = 0$.

Алгоритм планирования последовательно выбирает из классов количество элементов, определяемое следующими соотношениями:

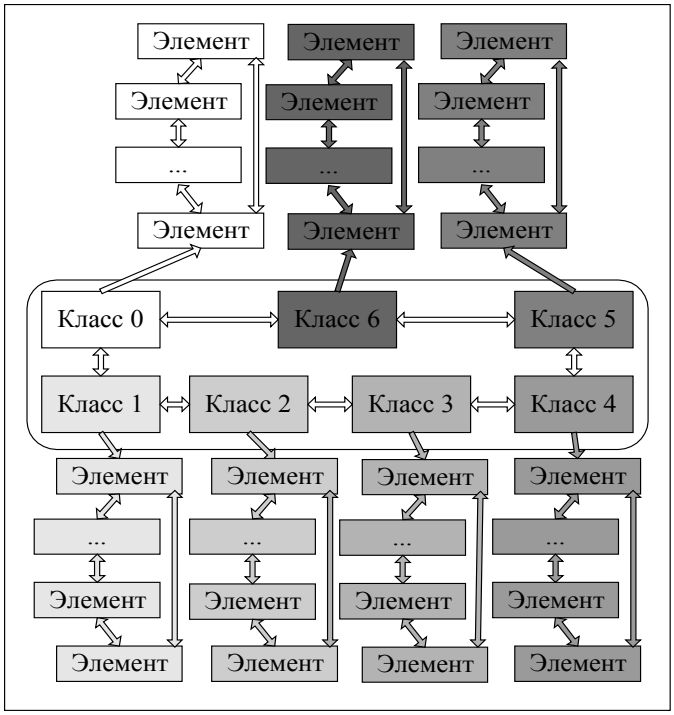

Рис. 2. Система классов приоритетов

$$C_{i(j+1)} = \left[\frac{W_i}{W_{\max}} \cdot C_i + O_{ij} \right];$$

$$O_{i(j+1)} = \frac{W_i}{W_{\max}} \cdot C_i + O_{ij} - C_{i(j+1)}.$$

Необходимость в использовании погрешности O_{ij} вызнана тем, что количество выбираемых элементов является целочисленным, а остаток должен использоваться при следующем выборе, чтобы не потерять точность, а соответственно не пропустить элементы.

При добавлении и удалении необходимо пересчитывать погрешности по следующему соотношению:

$$O_{i(j+1)} = O_{ij} \cdot \frac{W_{\max(new)}}{W_{\max}},$$

где $W_{\max(new)}$ – это новый максимальный вес.

Алгоритмы реализации приведенного выше общего подхода требуют достаточно много времени для выполнения, так как в них используется множество операций умножения и деления. С практической точки зрения выгоден частный случай алгоритма с экспоненциальной зависимостью между приоритетами, который позволяет добиться максимально быстрой скорости работы за счет замены операций умножения и деления битовыми сдвигами. Веса в этом случае задают возрастающими вдвое – 1, 2, 4, 8 и т.д. Алгоритм планирования фактиче

ски выбирает из первого класса все элементы, из второго половину, из третьего – четверть и т.д. В нем используются следующие соотношения для получения количества элементов выбранных из класса:

$$C_{i(j+1)} = \frac{C_i + \frac{O_{ij}}{2^N}}{2^{i-k}};$$

$$O_{i(j+1)} = (C_i \cdot 2^N + O_{ij}) \,\&\, (\frac{2^{2 \cdot N} - 1}{2^{N-i+k}});$$

где k – это номер непустого класса с максимальным весом,
& - побитовая операция «И».

Трудоемкость полученных алгоритмов установки задачи в очередь, удаления задачи из очереди и выбора задачи для выполнения составляет $O(1)$ и не зависит от числа планируемых задач. Предлагаются комбинированные алгоритмы планирования, где в качестве базы используется алгоритм древесного планирования, а для выбора элементов используются классические алгоритмы или базовым алгоритмом используется один из известных, а для выбора элементов используется система классов со статическими приоритетами.

В третьей главе рассматривается модульный принцип проектирования и реализации ядра операционной системы. Исследуется проблема блокировки объектов ядра и межмодульной блокировки. Предлагаются методы синхронизации использования и принципы взаимоотношения между разделяемыми ресурсами, обеспечивающие минимальное время нахождения в критической секции (рис. 3). Исследуется взаимное влияние на блокировки нескольких процессоров в SMP-системах, предлагается способ локализации блокировок, основной принцип которых – это частое блокирование локальных объектов и уменьшение количества блокировок глобальных объектов для доступа к локальным.

Автором предлагается структура и алгоритмы функционирования активного интерфейса (рис. 4), который позволяет выполнять замену модулей без удаления зависимых модулей, а также захватывать интерфейсы для работы еще до появления модуля с их реализацией в системе. Сам интерфейс состоит из набора таблиц функций, счетчиков использования этих функций и объекта синхронизации. Для работы с интерфейсом используют-

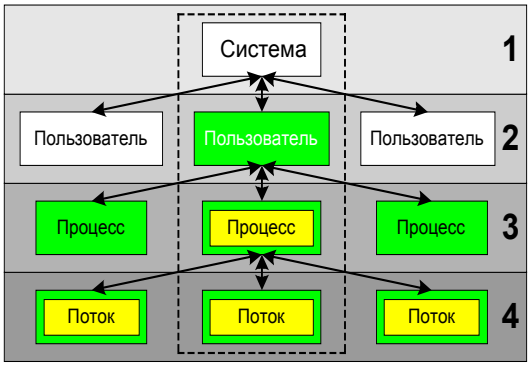

Рис. 3. Взаимодействие базовых объектов

ся пять функций: добавление набора функций (возможно просто задекларировать интерфейс), удаление набора, захват интерфейса, освобождение интерфейса и собственно функция вызова через захваченный интерфейс. Гарантируется, что вызов произойдет корректно, даже если все наборы функций будут удалены, хотя вместо результата вернется ошибка.

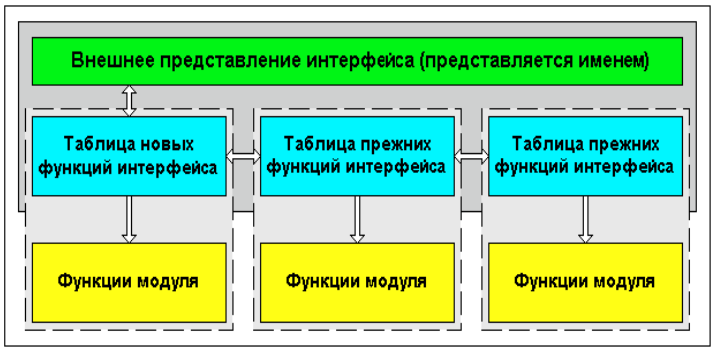

Рис. 4. Структура объекта-интерфейса

Далее исследуется важная задача распределения блоковых ресурсов. Блоковые ресурсы – это ресурсы, которые не требуются целиком, а только частями (блоками). Рассматриваются алгоритмы кэширования. Предлагаются алгоритмы функционирования блоковой подсистемы, которые в этой главе оптимизируются для оказания минимальной задержки при блокировке очередей, что является важным критерием для систем реального

времени. Для достижения поставленной цели применяется АВЛ-дерево, с помощью которого достигается быстрый поиск блоков. Теоретически трудоемкость поиска $1{,}4 \cdot \log n$, однако количество блоков в кэше ограничено и, как правило, не больше 4 миллионов (что уже требует 2 Гб ОЗУ), и соответственно максимальный количество итераций алгоритмов поиска, добавления и удаления будет меньше 50, что позволяет отказаться от использования мутексов и применить межпроцессорные блокировки.

В четвертой главе исследуется задача удаленного доступа к вычислительной системе. Предлагается «расширенная терминальная подсистема», где понятие терминала значительно расширяется: в него включаются дополнительные устройства, такие как: дисковод, принтер, устройство считывания компакт-дисков и др. Предлагается метод виртуализации терминальных устройств и алгоритмы его реализации.

На рис. 5 приводится взаимодействие внутренних объектов системы и физических устройств. Основными объектами данной подсистемы являются физический терминал (или просто терминал) и виртуальный терминал.

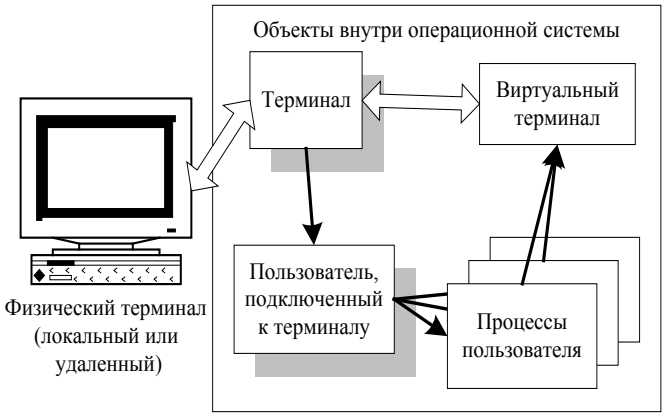

Рис. 5. Взаимодействие компонент терминала

Физический терминал используется для объединения терминальных устройств в единое целое, обеспечивает механизмы вызова драйверов устройств для локальных терминалов, а также осуществляет перенаправление и передачу данных на удаленный терминал. С каждым физическим терминалов ассоциируется системный процесс называемый «системным монитором». Этот процесс осуществляет регистрацию пользователя на термина

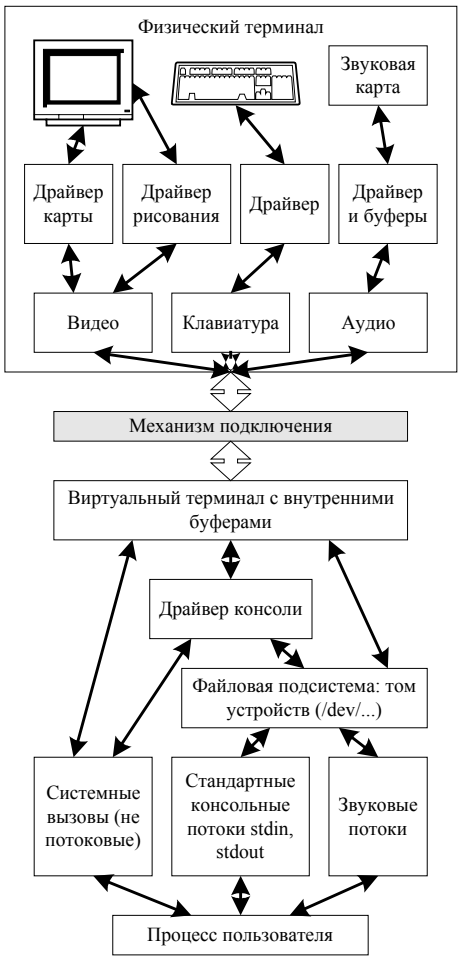

Рис. 6. Схема терминальной системы

ле и подключение группы виртуальных терминалов пользователя к физическому терминалу. Виртуальный терминал, предназначен для передачи запросов приложений к терминальным устройства, для буферизации входных сообщений, а также для хранения конфигурационных настроек устройств физического терминала, в соответствии с которыми настраивается физический терминал во время подключения к нему виртуального терминала.

На рис. 6 приводится схема взаимодействия программных компонент. Для каждого терминального устройства создается канал взаимодействия с ним. При работе с устройством приложения посылают данные в соответствующий канал и из него же принимают результат. Существуют виртуальные каналы, которые опираются не на устройства, а на другие каналы (консоль, одно в графической подсистеме и др.). В случае удаленного терминала, данные этих каналов могут быть напрямую переданы на удаленный терминал, что во многих случаях позволяет существенно понизить сетевой трафик. Например, приложение открывает звуковой виртуальный канал с форматом данных MP3, в случае локального терминала данные модулем поддержки этого канала будут развернуты в «сырой» поток и посланы в обычный звуковой канал, в случае же удаленного терминала данные сразу будут направлены в сеть, а развернуты они будут на удаленном терминале.

Так же в главе предлагаются алгоритмы подключения пользователя к терминалу, отключения, обмена данными с устройствами через виртуальные терминал и алгоритм функционирования терминала, который осуществляет настройку физических устройств по параметрам из виртуального терминала. Показано что трудоемкости этих алгоритмов линейные или меньше. Время нахождения в блокировках минимально.

В пятой главе кратко описана созданная на основе полученных теоретических результатов универсальная операционная система НИКОС. Структура ядра системы спроектирована с учетом минимальной сложности и универсальности. Для построения использовался модульный подход, позволяющий обеспечить новое качество системы при выборе тех или иных компонент ядра. Несмотря на новые подходы и реализацию новых алгоритмов, в операционной системе поддержаны основные промышленные стандарты программирования и интерфейсы, что позволяет использовать уже существующее программное обеспечение.

В табл. 1 приведено сравнение базовых функциональных возможностей реализованных в ОС НИКОС с наиболее распространенными в мире операционными системами. В таблице не приводится сравнение скоростных характеристик, так как в эталонных ОС компоненты, рассмотренные в данной работе, построены по другому принципу, и адекватного сравнения провести не возможно, подсистемы приемлемые для скоростного сравнения построены в ОС НИКОС по стандартным схемам и алгоритмам и не выносятся в качестве положений в данной работе.

Табл.1. Сравнение ОС НИКОС с другими ОС

Критерий сравнения	НИКОС	Free BSD	Linux	Solaris	Windows	Qnx
Выгружаемость	+	-	-	+	+	-
Учет пользователей при распределении ЦП	+	-	-	-	-	-
Поддержка мягкого реального времени	+	-	-	-	+	+
Многопотоковость на уровне ядра	+	-	-	+	+	+
Восстанавливаемое удаленное соединение	+	-	-	-	+	-
Разрыв жестких связей между модулями	+	-	-	-	-	-

В заключении кратко рассматривается возможность применения полученных результатов, а так же потенциальная область применения разработанной операционной системы.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Проведен анализ функциональности операционных систем, который позволил выяснить современные тенденции развития операционных систем, а также выявить присущие этим системам недостатки, часть которых не могут быть устранены в рамках упомянутых систем. Помимо этого выявлен ряд узких мест, которые требуют тщательной проработки при реализации новой среды. Сформулированы требования к универсальной операционной системе.
2. Предложен алгоритм древовидного планирования, который гарантирует справедливое распределение процессорного времени между потоками, процессами и пользователями, имеет трудоемкость $O(1)$ относительно числа планируемых задач и $O(n)$ относительно количества используемых приоритетов. Алгоритм гарантирует стабильный интервал получения квантов процессом и равномерно распределяет кванты процессов на всем периоде распределения квантов для всех процессов (время полного повторения цикла работы алгоритма). Предложены несколько комбинаций предложенного алгоритма с классическими алгоритмами планирования, что дает упрощение реализации с сохранением основных свойств предложенного алгоритма.

3. Предложена схема взаимодействия базовых и вспомогательных объектов ядра, в котором устанавливаются древовидные взаимосвязи между этими объектами, для обеспечения локализации частых обращений к объектам, что позволяет иметь минимальное время нахождения в блокировках.
4. Предложен подход к построению и алгоритмы функционирования активного интерфейса, который позволяется разорвать жесткую зависимость между модулями, что обеспечивает возможность обновления базовой функциональности системы без ее остановки, а также без остановки работающих приложений.
5. В алгоритмах функционирования подсистемы блоковых устройств предложено использование АВЛ-дерева для гарантированного быстрого поиска в блоковом кэше. Данный подход позволяет фиксировать максимальное время поиска и, следовательно, гарантирует поиск блока за фиксированное время (в классических подходах с хэш-функцией в наихудшем случае поиск блока может происходить неопределенно долго). Приводятся предлагаемые алгоритмы функционирования блоковой подсистемы и анализируются их трудоемкости. Показано, что трудоемкость данных алгоритмов составляет $1{,}4 \cdot \log(n)$. Это свойство позволяет применять эти алгоритмы в системах реального времени.
6. Предложено понятие «расширенного терминала» и свойства, которыми должен обладать этот терминал; дано определение терминальных устройств, которые должны функционировать через терминальную подсистему. Предложена схема взаимодействия вычислительной системы и терминальной подсистемы. Предложены алгоритмы функционирования терминальной подсистемы. Показано, что трудоемкость этих алгоритмов равна $O(1)$, а время нахождения в критических секциях постоянно и мало.
7. Разработана универсальная операционная система НИКОС, в которой воплощены полученные в работе теоретические результаты.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. *Гусев И.С.* Активный интерфейс для ядра ОС // Новые технологии и комплексные решения: наука, образование, производство. Материалы Всероссийской научно-практической конференции. Часть VI (Информатика).– КемГУ, 2001, с. 17-18.
2. *Гусев И.С.* Алгоритм планирования, основанный на справедливом распределении процессорного времени // Вестник Томского госуниверситета, 2002, № 275, с. 120-123.
3. *Гусев И.С.* Взаимодействие основных объектов ядра для эффективного выполнения блокировок в выгружаемых системах // Новые технологии и комплексные решения: наука, образование, производство. Материалы Всероссийской научно-практической конференции. Часть VI (Информатика).– КемГУ, 2001, с. 19-20.
4. *Гусев И.С.* Выполнение запросов от многих участников системы к нескольким устройствам // Вестник Томского госуниверситета, 2002, № 275, с. 124-126.
5. *Гусев И.С.* Использование криптографии на уровне операционной системы // Новые технологии и комплексные решения: наука, образование, производство. Материалы Всероссийской научно-практической конференции. Часть VI (Информатика).- КемГУ, 2001, с. 21.
6. *Гусев И.С.* Распределение процессорного времени с учетом современных требований к операционным системам // Тезисы докладов научно-практической конференции "Наука и образование: пути интеграции" г. Анжеро-Судженск, 1998, с. 13-17.
7. *Гусев И.С.* Распределение процессорного времени в узлах суперкомпьютера // Обозрение прикладной и промышленной математики, г. Йошкар-Ола, 2001, с. 582-583.
8. *Гусев И.С.* Расширенный терминал удаленного доступа // Новые технологии и комплексные решения: наука, образование, производство. Материалы Всероссийской научно-практической конференции. Часть VI (Информатика).- КемГУ, 2001, с. 16-17.
9. *Гусев И.С.* Расширенный терминал: алгоритмы функционирования и синхронизации // Вестник Томского госуниверситета, 2002, № 275, с. 117-119.
10. *Гусев И.С.* Эффективные методы синхронизации в выгружаемых операционных системах // Вестник Томского госуниверситета, 2002, № 275, с. 127-129.