Теоретические вопросы информатики, прикладной математики, компьютерных наук и когнитивно-информационных технологий

УДК 004.451

Назаров С.В. [1, 2]

[1] Московский научно-исследовательский телевизионный институт, г. Москва, Россия
[2] Финансовый университет при Правительстве РФ, г. Москва, Россия

ЭФФЕКТИВНОСТЬ СОВРЕМЕННЫХ ОПЕРАЦИОННЫХ СИСТЕМ

Аннотация

Многообразие современных универсальных и специализированных операционных систем (ОС) различных производителей, рассчитанных на применение в определенных предметных областях, ставит перед потребителями ряд вопросов: какую ОС выбрать для автоматизированной информационной системы или автоматизированной системы управления технологическими процессами критически важных объектов, как оценить ее эффективность. При этом известны общие требования, предъявляемые к операционным системам, достоинства и недостатки различных архитектурных схем ОС, что же касается вопросов оценки их эффективности, то кроме общих слов и рекомендаций, практически ничего не предложено. В данной работе формализуется понятие показатель эффективности функционирования ОС и рассматривается совокупность показателей, которая может быть использована для принятия решения по выбору ОС, учитывая предметную область ее использования, обращается внимание на показатели эффективности, которые характеризуют долю затрат компьютерных ресурсов, расходуемых на работу ОС. Дается математическая постановка задачи выбора ОС из совокупности возможных вариантов.

Ключевые слова

Операционная система; оценка эффективности; показатель эффективности ОС; обобщенные показатели; частные показатели; критерий эффективности.

Nazarov S.V. [1, 2]

[1] Moscow Research TV Institute Joint Stock Company, Moscow, Russia
[2] Finance University under the Government of the Russian Federation, Moscow, Russia

EFFICIENCY OF MODERN OPERATING SYSTEMS

Abstract

The variety of the modern multi-vendor universal and specialized operating systems (OS), designed for application in certain subject areas, poses a number of questions: which system should one choose for an automated information system, and for an automated process control system for critically important objects; and how to estimate its efficiency. At the same time, the general requirements for operating systems, the merits and demerits of various architectural schemes of the OS are known, and as for the estimation of efficiency except for vague generalities practically nothing has been proposed at this stage.

In this article the concept of the OS operational efficiency is formalized and a set of indicators that can be used for choosing the OS is considered. The attention is given to operational performance that characterizes the share of computer resources spent for operating the OS. The mathematical posing of the problem of choosing the OS from a set of possible variants is given.

Keywords

Operating system; efficiency assessment; indicator of efficiency of OS; the generalized indicators; private indicators; criterion of efficiency.

Введение

Важность современных операционных систем трудно переоценить. Информатизация общества, гигантский рост числа производственных, офисных, домашних, игровых компьютеров, интернет вещей обеспечивается широким спектром операционных систем различных зарубежных и отечественных производителей, рассчитанных на самые различные вычислительные мощности – от встраиваемых микроконтроллеров до суперкомпьютеров [1]. Операционные системы составляют основу различных автоматизированных систем управления (АСУ). К таким системам можно отнести ситуационные центры технологического назначения (например, ситуационно-аналитический центр ОАО «СО ЕЭС» [2], центры управления технологическими процессами атомных электростанций и др.,), АСУ управления критически важными объектами (финансово-банковская система, система спасения и оказания скорой помощи, гидротехнические сооружения, метрополитен и др.), центры управления информационной безопасностью различных объектов и систем, ситуационные центры органов государственной власти, национальный центр управления обороной Российской Федерации и др. [3]. В подобных системах используются как известные операционные системы реального времени (QNX, LynxOS и др.), так и универсальные, из которых следует выделить отечественную ОС Astra Linux Special Edition компании НПО «РусБИТех» и программный комплекс «Виртуализации и управления» (ПК «ВИУ») на ее основе [4].

Современные операционные системы (ОС), начиная с одной из первых – THE, разработанной Э. Дейкстрой, являются многоуровневыми [5]. По сути такие системы предоставляют пользователю виртуальную машину, с которой легче работать и проще писать программы, чем непосредственно с реальной физической машиной. Интерфейс прикладного программирования (API), образуемый ОС между приложением и аппаратурой компьютера, имеет много преимуществ. Он облегчает программирование, поскольку программистам не надо детально знать характеристики устройств компьютера. Кроме того, повышается безопасность системы, так как можно проверить корректность запроса приложения на уровне интерфейса до выполнения этого запроса. Такие интерфейсы делают программы переносимыми, позволяя компилировать и корректно выполнять их в каждом ядре, предлагающем такой же набор интерфейсов. Важная функция интерфейса – системные вызовы – механизм, который позволяет пользовательским программам обращаться к услугам ядра ОС. Посредством системных вызовов создаются, используются и удаляются различные объекты ОС – процессы, потоки, семафоры и файлы [6-9].

С другой стороны, операционную систему можно рассматривать как систему управления вычислительным процессом и ресурсами компьютерной системы. Современные операционные системы реализуют мультипрограммный вычислительный процесс, при котором десятки и сотни программ могут работать с одним и тем же набором ресурсов (процессоры, память, файлы и др.), распределяя его по временному (квантование) или пространственному (выделяя часть ресурса) принципам. Решение проблемы создания виртуальной машины с требуемыми свойствами в управлении и использовании, с обеспечением необходимой надежности, безотказности и защищенности приводит к росту сложности операционных систем и разрастанию программного кода. История развития операционных систем показывает, что с течением времени их сложность и объем только возрастают.

1. Требования, предъявляемые к операционным системам

К операционным системам в зависимости от сферы их использования предъявляется ряд требований. Некоторые, из перечисленных ниже, могут быть более или менее жесткими. Основное требование – функциональная полнота – зависит от предметной области использования системы. Функциональная полнота определяется выполнением основных функций эффективного управления ресурсами и обеспечения удобного интерфейса для пользователя и прикладных программ. Современная ОС должна поддерживать мультипрограммный вычислительный процесс, как правило, виртуальную память, возможно свопинг, требуемый интерфейс для приложений и, возможно для пользователя, высокую степень защиты, удобство работы, а также выполнять многие другие необходимые функции и услуги. Кроме требований функциональной полноты, к ОС предъявляется ряд важных эксплуатационных требований.

Эффективность. Под эффективностью вообще любой системы понимается степень соответствия системы своему назначению, техническое совершенство и экономическая целесообразность [7-10]. На показатели эффективности ОС влияет много различных факторов, среди которых основными являются архитектура ОС, многообразие ее функций, качество программного кода, аппаратная платформа (компьютер) и др. Эти вопросы рассмотрены ниже.

Надежность и отказоустойчивость. Операционная система должна быть не менее надежна, чем компьютер, на котором она работает.

Система должна быть защищена от внутренних и внешних сбоев и отказов. В случае ошибки в программе или аппаратуре система должна обнаружить ошибку, попытаться исправить ее или свести к минимуму ущерб, нанесенный этой ошибкой. Надежность и отказоустойчивость ОС определяются архитектурными решениями, положенными в ее основу, а также отлаженностью программного кода (основные отказы и сбои ОС в основном обусловлены программными ошибками в ее модулях) [11].

Безопасность (защищенность). ОС должна защищать приложения и пользователей и от воздействия чужих ошибок, и попыток злонамеренного вмешательства (несанкционированного доступа). Свойства безопасности особенно важны для сетевых ОС. В таких ОС к задаче контроля доступа добавляется задача защиты данных, передаваемых по сети.

Предсказуемость. Требования, которые пользователь может предъявить к системе, в большинстве случаев непредсказуемы. В то же время пользователь предпочитает, чтобы обслуживание не очень сильно менялось в течение предположительного времени. В частности, запуская свою программу в системе, пользователь должен иметь основанное на опыте работы с этой программной приблизительное представление, когда ему ожидать выдачи результатов. Требования со стороны приложений, как правило, могут быть сформулированы и достаточно полно учтены.

Расширяемость. В отличие от аппаратных средств компьютера полезная жизнь операционных систем измеряется десятками лет. ОС может быть расширяемой, если при ее создании руководствовались принципами модульности, функциональной избыточности, функциональной избирательности и параметрической универсальности.

Переносимость. В идеальном случае код ОС должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которые различаются не только типом процессора, но и способом организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа. Многоплатформенность достигается написанием большей части ОС на языке высокого уровня (например, C, C++ и др.). Меньшая часть ОС (программы ядра) является машиннозависимой и разрабатывается на машинном языке другого процессора.

Совместимость. Существует несколько "долгоживущих" популярных ОС, для которых наработана широкая номенклатура приложений. Если ОС имеет средства для выполнения приложений, написанных для других операционных систем, то она совместима с этими системами. Следует различать совместимость на уровне двоичных кодов и совместимость на уровне исходных текстов. Кроме того, понятие совместимости включает также поддержку пользовательских интерфейсов других ОС.

Удобство. Средства ОС должны быть простыми и гибкими, а логика ее работы ясна пользователю. Современные ОС ориентированы на обеспечение пользователю максимально возможного удобства при работе с ними. Необходимым условием этого стало наличие у ОС графического пользовательского интерфейса и всевозможных мастеров – программ, автоматизирующих активизацию функций ОС, подключение периферийных устройств, установку, настройку и эксплуатацию самой ОС.

Масштабируемость. Если ОС позволяет управлять компьютером с различным числом процессоров, обеспечивая линейное (или почти такое) возрастание производительности при увеличении числа процессоров, то такая ОС является масштабируемой. В масштабируемой ОС реализуется симметричная многопроцессорная обработка. С масштабируемостью связано понятие кластеризации – объединения в систему двух (и более) многопроцессорных компьютеров. Правда, кластеризация направлена не столько на масштабируемость, сколько на обеспечение высокой готовности системы.

2. Почему сложно создавать эффективные операционные системы

Как отмечалось выше, сложность и объем кода операционных систем отмечаются только ростом во времени, чего нельзя сказать о надежности и качестве операционных систем. Остановимся на причинах сложности создания эффективных операционных систем, удовлетворяющих перечисленным выше требованиям. Эти причины обусловлены как объективными, так и субъективными факторами. Главный объективный фактор – сложность и трудоемкость разработки, желание учесть максимально возможный перечень требований Субъективные факторы в основном связаны с конкуренцией на рынке ОС. Основные причины, сложности создания эффективных операционных систем, по мнению специалистов, достаточно обозначились на практике [5, 6, 11, 14, 15-17]. К ним относятся:

1. Операционные системы относятся к классу больших программных систем, которые характеризуются широкой функциональностью, высокой трудоемкостью и сложностью, рисками и длительностью процесса разработки. Например, создание OS/360 в шестидесятые годы прошлого век потребовало 5000 человеко-лет.
2. Высокая сложность и большой объем программного кода, например, ОС Windows

2000 содержит примерно 30 миллионов строк кода. Понятно, что ни один разработчик не может даже надеяться понять целиком эту систему. Следствие гигантского объема – большое количество программных ошибок (не менее 6 ошибок на 1000 строк кода), которые выявляются долгие годы.

3. Инерция и желание сохранить “обратную” совместимость – наличие в новой версии системы интерфейса и возможностей, присутствующих в старой версии, в результате чего ранее разработанные программы (или человек) могут работать с новой версией без переделки (или переучивания).
4. Желание создания системы “более дружественной по отношению к пользователю”, т.е. такой, которая не предъявляет к пользователю особых требований в части знания компьютера и программирования. Как правило, это приводит к снижению производительности компьютера, а отсюда принятие разработчикам специальных мер, компенсирующих этот фактор, например, реализация графического интерфейса Windows в режиме ядра ОС.
5. Стратегия производителей, направленная на повышение продаж. Разработчики зачастую не имеют четкого представления, как будет использоваться их система. Это приводит к тому, что они вынуждены увеличивать ее универсальность, добавляя множество разнообразных функций (достаточно вспомнить Windows с набором гаджетов, стандартных, игровых и прочих программ).
6. Агрессивная рыночная и рекламная политика производителей-конкурентов, периодически объявляющих о разработке новых более совершенных версий программного продукта. В такой ситуации компании-производители снимают поддержку и сопровождение популярных распространенных версий систем, чтобы вынудить потребителей системы перейти на новую версию. При этом необходимо обеспечить максимальную совместимость с предыдущими версиями.
7. Необходимое с течением времени усложнение организации мультипрограммного вычислительного процесса, связанное с прогрессом микроэлектроники и технологии построения вычислительных устройств. Отсюда возрастающая сложность параллельного управления большим числом параллельных заданий, процессов, потоков, волокон в сложных аппаратных многопроцессорных и многоядерных структурах.
8. Достижение требуемой надежности работы операционной системы. Особенность программных систем в отличие от аппаратуры вытекает из факта их неизменности по мере наработки. Если какие-то изменения происходят, особенно на начальных этапах работы, то это связано с установкой patch «заплаток» и сервисных пакетов. Однако в целом изменения в ОС происходят только в результате устранения программных ошибок.
9. Требование учета возможных непреднамеренных ошибок легальных пользователей или их незаконных действий, направленных на получение каких-либо преимуществ и даже хищений. Нужно учитывать также действия потенциально враждебных пользователей, желающих вмешаться в работу системы, выполняя незаконные действия.
10. Переносимость на другие платформы и поддержка широкой номенклатуры внешних устройств, которые проектируются независимо друг от друга и часто независимо от какой-либо ОС. Развитие технологии виртуализации несколько сглаживает эту проблему, но возникает другая – разработка эффективных гипервизоров.
11. Расчет на долгий срок жизни. Современные операционные системы – долгожители (первая версия Unix живет с 1970 года и поныне, Windows – с 1985 года). Проектировщики и разработчики ОС должны представлять, как могут измениться компьютеры и приложения в будущем и как к этому подготовиться.

Понятно, что только учет всех этих факторов позволит создавать эффективные операционные системы.

3. Формализация показателей эффективности функционирования операционной системы

В работах автора [7,12,13,17] достаточно подробно рассмотрены вопросы формализации понятия эффективность функционирования применительно к сложным техническим (программно-аппаратным) системам. Определены понятия: параметры системы, свойства, качество системы, показатель эффективности, критерий эффективности. Показана их связь и зависимости. Применительно к рассматриваемой в данной работе задачи эта связь и зависимость может быть представлена как показано на рис. 1.

Качество операционной системы

представляется совокупностью свойств, характеризующих ее использование по конкретному назначению. В то же время эффективность, в соответствии с положениями современной теории эффективности, свойством системы не является. Это обусловлено следующими причинами:

- оценка эффективности (уровня эффективности) связана не только со свойствами операционной системы, но и (даже, возможно, в большей степени) со свойствами результатов ее функционирования и ресурсами, затраченными на достижение этих результатов;
- эффективность функционирования операционной системы определяется не только ее свойствами, но и способом использования системы по целевому назначению (например, режимом работы, входящими потоками данных, характеристиками и требованиями функциональных задач к ресурсам и т.п.);
- рассмотрение эффективности как свойства операционной системы вступает в противоречие с понятием ее качества.

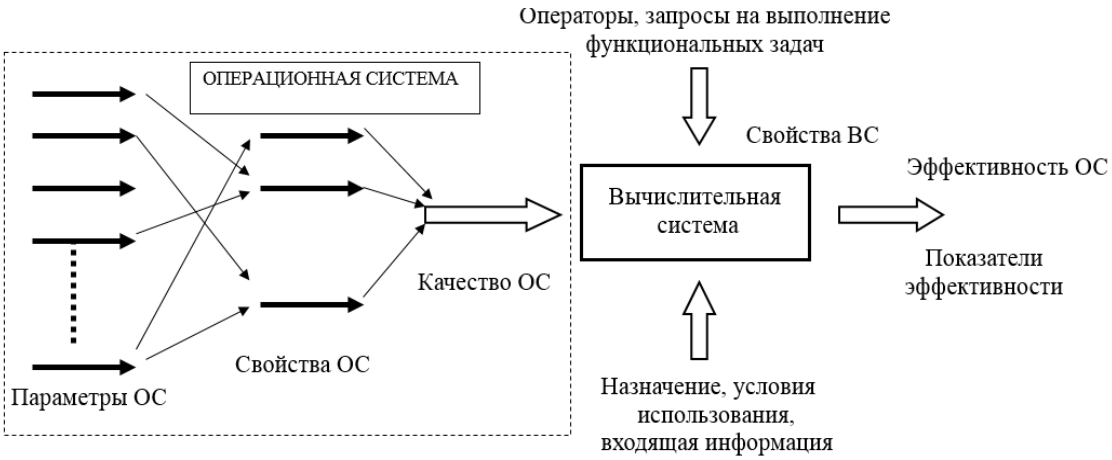

Рис. 1. Связь между параметрами, свойствами и показателями эффективности ОС

Таким образом, эффективность операционной системы следует понимать, как эффективность ее функционирования, т.е. свойства ее использования по назначению (т.е. свойство операции). Эффективность ОС зависит от ее качества, которое в свою очередь определяется свойствами системы, условиями использования системы, а также свойствами надсистемы, т.е. вычислительной системы в целом. Операционную систему формально (со степенью детализации до уровня, необходимого в данной работе) можно определить как четверку

$$OS = < S_{OS}, F_{OS}, P_S, P_F >,$$

где S_{OS} – структурная организация ОС (состав, взаимосвязь элементов и подсистем ОС);

F_{OS} – функциональная организация ОС (взаимодействие, поведение, процесс работы элементов и подсистем ОС); P_S, P_F – соответственно параметры структурной и функциональной организации операционной системы. Можно считать, что выражение (1) задает полный набор параметров, описывающих операционную систему, т.е.

$$P_{os} = P\left(S_{OS}, F_{OS}, P_S, P_F\right)$$

Как сложная система ОС характеризуется широким набором свойств

$$W_{os} = \left\{W_j\left(P_{os}\right)\right\}, j \in J, \left|J\right| = m \gg 1.$$

Численное значение функционала, отражающего некоторое свойство ОС, которое характеризует уровень качества ОС относительно этого свойства, называется показателем этого свойства. Свойства могут быть количественными (численными) и качественными, с нечеткими оценками или порядковой шкалой. Однако для практики наибольший интерес имеют количественные оценки. Совокупность численных значений показателей свойств ОС W_{os} представляет собой обобщенное свойство операционной системы как системы управления процессом функционирования вычислительной системы. Однако эта совокупность еще не характеризует эффективность функционирования ОС, так как необходимо учесть условия использования системы по целевому назначению.

Под целевым назначением (целью использования) ОС понимается организация процесса функционирования вычислительной системы, предназначенной для реализации определенного набора задач в заданных условиях функционирования в соответствии с предъявляемыми требованиями. Обозначим через $P_з$ совокупность параметров, характеризующих свойства решаемых задач, а через $P_{вх}$ – совокупность параметров, характеризующих запросы пользователей системы на решение функциональных задач, передачу сообщений,

взаимодействие с задачами и т.п. Параметры структуры и свойства аппаратных и программных средств вычислительной системы обозначим $P_{вс}$. Множество параметров окружающей среды, в том числе воздействия злоумышленников, обозначим $P_{вф}$ (внешние факторы).

Под эффективностью операционной системы следует понимать комплексное свойство, характеризующее степень соответствия процесса функционирования ОС своему назначению, ее техническое совершенство и экономическую целесообразность. С учетом введенных обозначений численное значение функционала

$$W_j = W_{os}, P_з, P_{вх}, P_{вс}, P_{вф}, t,$$

отражающего степень соответствия (пригодности) ОС своему назначению по тому или иному ее свойству, будем называть показателем эффективности ОС. Показатели эффективности являются функционалами, определенными во времени t на множестве процессов функционирования ОС. К показателям эффективности, выбираемым в задачах анализа и синтеза операционных систем, предъявляется ряд общих требований, таких как представительность, ясный физический смысл, критичность (чувствительность) и устойчивость к изменению параметров системы.

Поскольку эффективность ОС оценивается по некоторой совокупности показателей, обобщенно задача оценки эффективности заключается в установлении численных значений выбранных показателей эффективности в зависимости от параметров процесса функционирования системы, изменяемых в некотором выбранном диапазоне и времени *t*, т.е. в определении

$$W_{os} = \left\{ W_j\left(P_{os}, P_з, P_{вх}, P_{вс}, P_{вф}, t\right) \mid j = 1,2,\ldots,m \right\}.$$

Сложность решения задачи связана с выбором дискретности изменений параметров системы и трудностью сравнения векторных оценок $W_{os} = \left\{ W_1, W_2, \ldots, W_m \right\}$ различных вариантов организации операционных систем.

4. Показатели эффективности ОС

4.1. Виды показателей эффективности. Классификация показателей эффективности ОС как сложных систем возможна по различным признакам: степени учета назначения и свойств системы, способу получения, физической сущности и т.п. Анализ содержания процессов функционирования различных операционных систем приводит к заключению о целесообразности использования четырех групп показателей [12,13]: целевого использования (или целевого назначения) $W_ц$, технического совершенства $W_т$, эргономичности $W_э$ и экономической целесообразности $W_{эк}$. Показатели первой группы оценивают эффективность ОС с точки зрения соответствия системы ее целевому назначению (например, работе в режиме реального времени или максимума обработки транзакций). Показатели второй группы должны соответствовать отысканию наилучших архитектурных решений (например, по надежности и безотказности системы или минимальным затратам ресурсов на работу ОС).

Показатели эргономичности характеризуют возможности интерфейса системы и, наконец, показатели экономической целесообразности должны характеризовать ОС с точки зрения «эффективность – стоимость». Каждая названная группа может включать в себя достаточно большое количество показателей, которые принято называть частными. В практически решаемых задачах исследования ОС из множества возможных показателей эффективности

$$W_{os} = \left\{ W_ц \cup W_т \cup W_э \cup W_{эк} \right\} = \left\{ W_j \mid j = 1,2,\ldots,m \right\}$$

необходимо выбрать такое подмножество показателей $W'_{os}\left\{ W_j \mid j = 1,2,\ldots,m \right\}$, которое бы наиболее полно характеризовало различные свойства ОС с учетом предъявляемых к ней требований в соответствии с целевым назначением всей системы. При этом желательно стремиться к выполнению условия $\left| W'_{os} \right| \ll m$ [12,13].

Стремление к упрощению задач исследования ОС (и других сложных систем) привело к разработке методов объединения частных показателей в единый показатель, что очень удобно при решении различных задач поиска оптимальных решений. В зависимости от степени сравнимости частных показателей различают следующие типы объединения: объединение количественно соизмеримых показателей; объединение показателей, для которых известно предпочтение по важности; объединение несоизмеримых между собой показателей. Такие обобщенные показатели называются конструируемыми, так как они объективно не присущи исследуемой системе. Как правило, критикуемы и обладают существенными недостатками, не позволяющими чаще всего их использовать на практике.

4.2. Обобщённые показатели эффективности. В данном разделе идет речь не о конструируемых показателях, а о показателях, характеризующих эффективность функционирования всей вычислительной системы, памятуя о том, что именно ОС, являясь средством организации вычислительного процесса и использования ресурсов системы, определяет эффективность работы вычислительной системы в целом. На практике используются следующие основные показатели эффективности ОС [16,17]:

1. *Производительность или пропускная способность системы*, выражаемая количеством

вычислительных, информационных (транзакций, справок) или других работ (заданий), выполняемых вычислительной системой за определенный промежуток (или единицу времени). Этот показатель хорошо подходит для оценки эффективности ОС систем пакетной обработки, запросных информационно-справочных систем и других систем, в том числе систем гибкого реального времени, например, систем резервирования авиабилетов и др. Производительность может определяться по отдельным типам работ, а также по всем видам, т.е. интегрально:

$$B_i\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)=N_i\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)/T,$$
$$i\in I;$$
$$B_\Sigma\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)=$$
$$=\sum_{i\in I}\frac{N_i\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)}{T},$$

где i – тип работы (всего I типов); $N_i\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)$ – количество выполненных работ i-го типа; T – временной интервал функционирования системы; t – текущее время ($0 \le t \le T$).

Необходимо отметить, что эти показатели не характеризуют в явном виде возможности вычислительной системы по производительности, поскольку в значительной степени определяются загрузкой системы и с ее увеличением возрастают, достигая максимальных значений при полной загрузке системы ($\rho \to 1$). Загрузку системы в первом приближении можно определить по выражению

$$\rho=\sum_{i\in I}\tau_i\times\lambda_i,$$

где τ_i – время выполнения работы i –го типа; λ_i – интенсивность поступления работ i –го типа.

Производительность системы существенно зависит от ее загрузки и при росте загрузки растет, достигая максимального значения при ρ =1, а далее падает по причине роста очереди работ, затрат на работу ОС и появления заторов в системе. Наконец, еще замечание по показателю интегральной пропускной способности. Непосредственно по значению этого показателя нельзя сделать вывод о качестве или эффективности ОС. Но если замена одной операционной системы на другую приводит к увеличению этого показателя, можно сделать вывод о том, что вторая ОС оказалась более эффективной в данной ситуации рабочей загрузки системы.

2. *Относительная пропускная способность* системы, выражаемая отношением числа работ, выполненных системой за некоторый промежуток времени, к числу работ, поступивших с систему за этот же интервал времени:

$$\delta\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)=\frac{N_в\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)}{N_п\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)}.$$

Относительная пропускная способность системы характеризует динамику работы системы (на интервале Δt), ее способность работы при случайных возмущениях, вызванных поступлением незапланированной нагрузки. Поскольку

$$N_п\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)=$$
$$=N_в\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)+,$$
$$+N_o\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right),$$

где $N_o\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)$ – число работ еще не выполненных системой, т.е. находящихся в очереди, то

$$\delta\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)=$$
$$=\frac{N_в\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)}{N_в\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)+N_o\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},\Delta t,T\right)}.$$

Кратковременно возникающие очереди могут существенно снизить качество вычислительного процесса, свидетельствуя о перегрузке системы. Так для ОС Windows очередь двух и более потоков к процессору свидетельствует о перегрузке системы, а следовательно, ведет к задержке выполнения работ.

3. Время реакции системы – интервал времени от поступления запроса (например, от управляемого объекта) до выполнения первой инструкции в программе обработки этого события. Это время t_p в основном определяется системой прерывания компьютера и временем переключения контекста. Время реакции ОС – очень важная характеристика особенно для систем жесткого реального времени. Для любой автоматизированной системы время реакции является составной частью следующего показателя эффективности системы в целом.

4. *Время ответа* – промежуток времени между моментами поступления задания (запроса) на выполнение и получением результата решения. Этот ПЭ важен для интерактивных систем любого назначения и может быть представлен как математическое ожидание разности между временем завершения выполнения задания и моментом его поступления, т.е. это, по сути, время пребывания задания в системе:

$$t_o\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)=$$
$$=MO\left[t_p+t_в\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)-t_п\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)\right],$$

где $t_в\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)$ - время выдачи результата обработки задания пользователю (управляемому объекту), а $t_п\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)$ – время поступления этого задания в систему. Разность этих величин представляет собой время пребывания задания в системе.

5. *Коэффициент задержки выполнения работ* (решения задач, обработки справок, сообщений и т.п.) – отношение времени пребывания задания в системе ко времени выполнения этого задания в монопольном режиме использования компьютера:

$$K_з\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)=$$
$$=MO\left[\frac{t_{пр}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)}{t_м\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)}\right].$$

С ростом загрузки системы коэффициент задержки выполнения работ растет, поскольку в режиме мультипрограммной работы системы время выполнения задания превышает время выполнения этого задания в монопольном режиме.

6. *Время выполнения заданного набора функциональных задач (ЗНФЗ).* Этот ПЭ используется для оценки эффективности специализированных вычислительных систем с фиксированным набором прикладных программ (например, в системах управления технологическими процессами). Это может быть один пакет или различные программы, которые выполняются с определённой частотой на различных типовых периодах функционирования системы. В этих случаях время выполнения ЗНФЗ можно определить следующим образом. Пусть известен набор задач $P=\left\{\pi_i \vee i=1,2,\ldots,m\right\}$, каждая из которых реализуется на том или ином интервале функционирования системы $\left\{T_q\right\}, q\in Q$. Пусть далее $x_{iq}=1$, если задача π_i реализуется в периоде T_q, и 0 – в противном случае. Тогда набор функциональных задач, реализуемых на интервале T_q, представляет собой множество

$$P_q=\underset{i}{U}\left\{\pi_i, если\, x_{iq}=1, \varnothing\ , если\, x_{iq}=0\right\}.$$

Минимальное время выполнения ЗНФЗ на интервале T_q представляется выражением:

$$T_{знфз}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)=$$
$$=\sum_i \lambda_{iq}\times x_{iq}\times t_i\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right),$$

где λ_{iq} – интенсивность решения задачи π_i на интервале работы системы T_q, $t_i\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)$ – время выполнения этой задачи в монопольном режиме.

7. Вероятность выполнения ЗНФЗ в заданные сроки:

$$Pr_{знфз}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)=$$
$$=Pr_{знфз}\left(T_{знфз}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)\le T_д\right),$$

где $T_д$ – директивное (максимально допустимое) время выполнения ЗНФЗ.

8. *Коэффициент полезного действия ОС.* Подобный показатель (кпд) широко применяется при характеристике технических устройств и отображает долю передаваемой или преобразуемой энергии, или работы. Применительно к операционной системе этот показатель можно трактовать как долю компьютерного времени, которая ОС предоставляет для выполнения пользовательских приложений или функциональных задач. Любой интервал времени работы *T* автоматизированной системы под управлением ОС можно представить в виде суммы

$$T=T_п+T_{ос}+T_{пр},$$

где $T_п$ – время, выделенное ОС на выполнение задач пользователя (режим пользователя), $T_{ос}$ – время выполнения программ ОС (режим ядра и режим пользователя), $T_{пр}$ – время простоя системы (ожидание событий, например, завершения ввода-вывода, отсутствие готовых к выполнению работ и др.).

Тогда показатель

$$K_{os}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)=$$
$$=1-\left(T_{ос}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)+\right.$$
$$\left.+T_{пр}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)\right)/T$$

дает представление о степени затрат системного времени на работу операционной системы. Очевидно, что чем ближе к единице этот показатель, тем выше эффективность функционирования ОС. Однако нужно иметь ввиду, что часть программ ОС выполняется в пользовательском режиме, некоторые системные вызовы могут выполняться в пользовательском режиме, другие, хотя и выполняются в режиме ядра, большую часть работы выполняют в интересах пользовательских приложений, и только переключение режимов ядро-пользователь, смена контекста, диспетчеризация потоков, планирование, организация виртуальной памяти, обработка прерываний и некоторые другие функции ядра можно отнести к затратам системного времени на саму операционную систему.

Отметим, что важность показателя $K_{os}\left(P_{os},P_з,P_{вх},P_{вс},P_{вф},t,T\right)$ трудно переоценить – этот показатель эффективности ОС характеризует внутреннее устройство системы, совершенство и эффективность ее основных механизмов. В связи с важностью этого ПЭ ниже, в разделе 5, дается возможный подход к оценке затрат на работу операционной системы.

4.3. Частные показатели эффективности. Эти показатели характеризуют определенную сторону процесса функционирования операционной системы, ее техническое совершенство и экономическую целесообразность. Поэтому эта группа показателей более обширна по сравнению с группой обобщённых показателей. Не претендуя на исчерпывающую обоснованность, частные ПЭ ОС можно представить следующими группами:

1. *Показатели надежности*, характеризующие способность ОС сохранять хотя бы минимальную работоспособность в условиях аппаратных сбоев и программных ошибок. В качестве таких ПЭ может использоваться коэффициент готовности, вероятность безотказной работы в течение заданного интервала времени функционирования системы, среднее время восстановления работоспособности и др.;

2. *Показатели, характеризующие возможности организации и управления вычислительным процессом:* количество поддерживаемых процессов и процессоров (в одной системе и в кластерных системах), объем поддерживаемой оперативной памяти;

3. *Режимы работы ОС* – пакетная обработка, разделение времени, диалоговый режим, режим «запрос-ответ», универсальный режим. Возможности работы в системах мягкого и жесткого реального времени;

4. *Показатели, характеризующие эффективность мультизадачной и мультипроцессорной работы:* алгоритмы планирования у правления заданиями, процессами и потоками (волокнами – потоками в пользовательском пространстве) и их эффективность в однопроцессорных, многоядерных, многопроцессорных и кластерных системах и др.;

5. *Показатели, характеризующие качество выполнения тех или иных функций* ОС, например, эффективность алгоритмов планирования и распараллеливания вычислительного процесса, организации взаимодействия процессов, диспетчеризации потоков, распределения оперативной и дисковой памяти и т.п.;

6. *Показатели, характеризующие требования операционной системы к размеру оперативной памяти:* минимальный и рекомендуемый объем, величина выгружаемой и невыгружаемой памяти ядра. Требования к минимальному объему дисковой памяти;

7. *Показатели, характеризующие основную файловую систему:* скорость доступа, надёжность хранения данных, степень устойчивости при сбоях, максимальный размер файла, шифрование, резервное копирование и дополнительные возможности – поддержка журналирования и др. Совместимость и поддержка других файловых систем;

8. *Показатели, характеризующие возможности виртуализации О*С: механизмы организации и управления виртуальной памятью, использование средств аппаратной поддержки, возможности создания виртуальных сред и систем (вложенные средства виртуализации, например, Hyper-V хост);

9. *Показатели, характеризующие сетевые возможности ОС:* поддерживаемые сетевые протоколы, удаленный доступ к приложениям и защита доступа;

10. *Экономические показа*тели: затраты на закупку (лицензию) и сопровождение системы.

5. Оценка затрат на работу операционной системы

Интерфейс между ОС и программами пользователя определяется набором системных вызовов (СВ), предоставляемых ОС. Как отмечалось в разделе 4.2, собственно на работу непосредственно операционной системы нужно отнести часть работы системных вызовов, связанную с подготовкой и завершением вызываемых услуг ядра, переключением режимов ядро-пользователь, сменой контекста, диспетчеризацией потоков, планированием, организацией виртуальной памяти, обработкой прерываний и некоторыми другими функциями ядра, связанными с работой системных служб и сервисов. Рассмотрим, как соотносятся эти составляющие по временным затратам процессорного ресурса.

Ввод-вывод, печать, работа с файлами, создание и завершение процессов и потоков, взаимодействие между ними и т.п. связаны с реализацией начальных и завершающих этапов системных вызовов, переключений ядро-пользователь и сменами контекста. Частота выполнения системных вызовов может достигать в зависимости от типа реализуемой задачи десятки тысяч в секунду и более. Поэтому даже при малых затратах процессорного времени на эти операции суммарные затраты времени могут быть значительными.

Диспетчеризация потоков выполняется в основном при истечении кванта времени, выделяемого активному потоку, или при его переходе с состояние ожидания какого-либо события. Учитывая, что время кванта составляет порядка 15 миллисекунд, диспетчирование проходит примерно 67 раз в секунду, а, следовательно, затраты на него незначительны. Аналогична ситуация с планированием вычислительного процесса.

Затраты процессорного времени, связанные с организацией виртуальной памяти, определяются интенсивностью использования файла подкачки и во многом зависят от размеров оперативной памяти, файла подкачки и числа одновременно выполняемых заданий. При правильном выборе этих параметров время выполнения приложений практически не должно отличаться от времени их выполнения без файла подкачки. В противном случае выполнение приложений замедляется, однако в такой ситуации нельзя делать вывод о росте затрат на работу ОС, так как этот рост связан с недостаточным количеством ресурса памяти.

Наконец, о затратах процессорного времени на обработку прерываний. Они связаны с активностью устройств, формирующих аппаратные прерывания. Сюда относятся системный таймер, мышь, драйверы дисков, линии передачи данных, сетевые адаптеры и другие устройства. Они прерывают процессор при завершении своей работы или при возникновении необходимости обработки запроса. Системный таймер обычно прерывает работу процессора каждые 10-15 миллисекунд, что касается драйверов, то в основном их работа выполняется в

интересах приложений и не относится непосредственно к затратам на работу операционной системы. Таким образом, оценка затрат на работу ОС связана прежде всего с определением затрат процессорного времени на выполнение системных вызовов.

Как правило, большинство ОС имеют встроенные механизмы регистрации событий системы. Для ОС Windows – это утилиты Windows Performance Toolkit (WPT). Для перехвата и анализа событий ETW (Event Tracing for Windows) используются утилиты XPerf.exe и XPerfView.exe, а также более новые инструменты Windows Performance Toolkit (WPT) – wpr.exe (Windows Performance Recorder) и wpa.exe (Windows Performance Analyzer). Эти инструменты могут пригодиться в ситуациях, когда требуется детально вникнуть в поведение системы и отдельных процессов. Например, можно перехватить все события дисковых операций ввода-вывода, выявить операции, требующие значительных перемещений головок жесткого диска, получить информацию о состоянии стеков вызовов для процессоров в системе и др. Для оптимизации пользовательских приложений может быть использован инструмент командной строки PerfMonitor.exe, позволяющий анализировать события CLR и JIT-компилятора. В целом эти инструменты необходимы для глубокого анализа системы и пользовательских приложений.

Современные операционные системы имеют также в составе средств администрирования инструменты, позволяющие получить исчерпывающую информацию о состоянии и производительности оборудования компьютера и операционной системы. Например, системный монитор Windows можно использовать для анализа работы операционной системы как в реальном времени, так и посредством сбора данных с фиксацией в журнале для последующей обработки. Инструментами оценки состояния или активности системы являются счетчики производительности. Они входят в состав операционной системы или могут быть частью отдельных приложений. Используя счетчики системного монитора и достаточно простые тестовые программы, можно определить временные характеристики системных вызовов ОС.

```
class Test1
        { // Программа определения времени выполнения системного вызова
     // записи текста на дисплей
    static void Main()
    {
      double k1;
      Console.WriteLine("Введите число повторов цикла");
      k1 = Convert.ToDouble(Console.ReadLine());
      Console.WriteLine("Введите выводимый на экран текст");
      string t = Console.ReadLine();
      //Время начала системных вызовов в миллисек,
      //отсчитываемое от 00:00:00 1 января 0001 года
      DateTime now = DateTime.Now;
      double time = now.Ticks / 10000;
      for (double k = k1; k > 0; k--)
        Console.Write(t);
      //Время завершения системных вызовов в миллисек,
      //отсчитываемое от 00:00:00 1 января 0001 года
      DateTime now1 = DateTime.Now;
      double time1 = now1.Ticks / 10000;
      Console.WriteLine("\n");
      Console.WriteLine("\nЧисло повторов цикла  " + k1);
      Console.WriteLine("\nВыводимый на экран текст:  " + t);
      Console.WriteLine("\nВремя выполнения системных  "
      + "вызовов в миллисекундах " + (time1 - time));
            Console.WriteLine("\nВремя выполнения системного вызовова"
          + "в миллисекундах " + Convert.ToDouble((time1 - time) / k1));
      Console.ReadLine();
    }
 }
```

Рис. 2. Текст теста №1

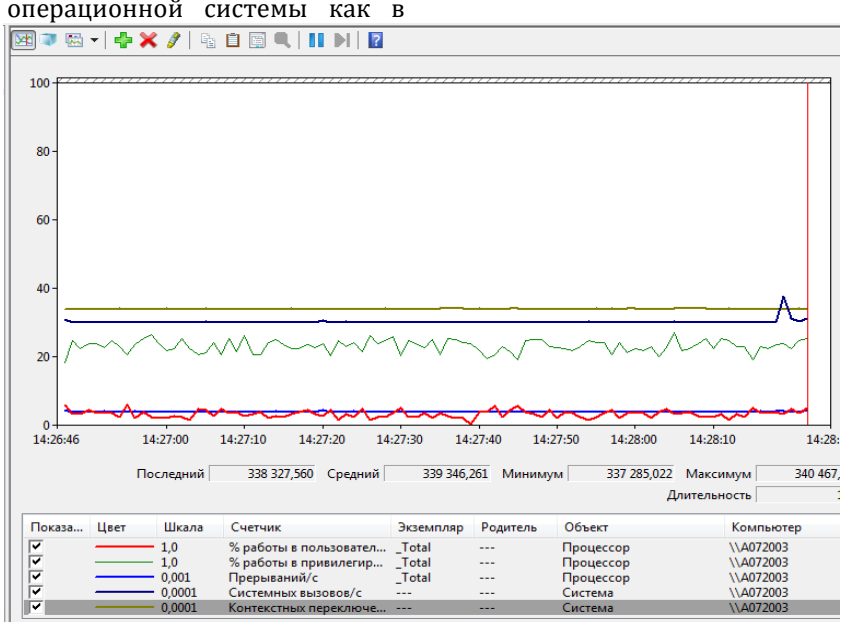

Рис. 3. Результат работы теста №1 в окне системного монитора

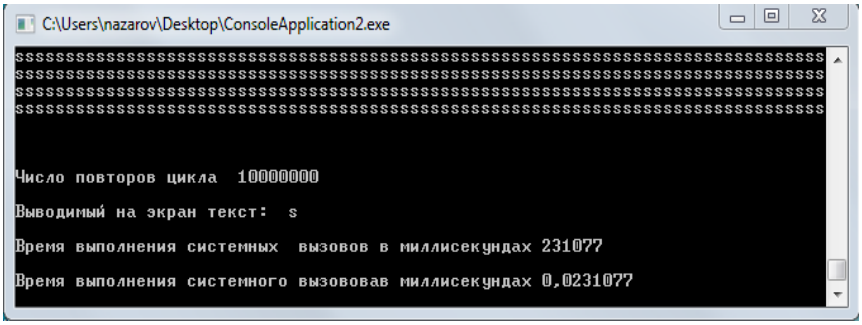

Рис.4. Результат работы теста № 1 в командном окне

	A	B	C	D	E	F	G
1	Время	\\A072003\Процессор(_Total)\% работы в пользовательском режиме	\\A072003\Процессор(_Total)\% работы в привилегированном режиме	\\A072003\Процессор(_Total)\Прерываний/с	\\A072003\Система\Контекстных переключений/с	\\A072003\Система\Системных вызовов/с	
2	13:35:00	2,73	23,33	3769,66	341172,39	301801,94	
3	13:35:05	3,59	22,89	3707,31	341888,59	303003,04	
4	13:35:10	2,41	23,83	3643,26	341639,57	302163,28	
5	13:35:15	3,04	23,31	3646,06	341378,23	301942,91	
6	13:35:20	3,36	23,13	3716,13	341461,23	301954,27	
7	13:35:25	2,81	24,02	3736,71	341626,42	302100,55	
8	13:35:30	3,27	23,05	3679,88	341445,66	301986,07	
9	13:35:35	3,12	23,71	3695,5	341663,79	302128,81	
10	13:35:40	2,97	22,57	3647,69	341295,39	301784,57	
11	13:35:45	3,59	23,09	3679,31	341681,26	302164,87	
12	13:35:50	3,44	23,83	3655,76	341210,21	301727,18	
13	13:35:55	3,67	22,93	3702,1	341812,9	302333,55	
14	13:36:00	3,27	23,83	3699,98	341157,18	301789,18	
15							

Рис.5. Фрагмент обработанных в Excel записей журнала результата работы теста № 1

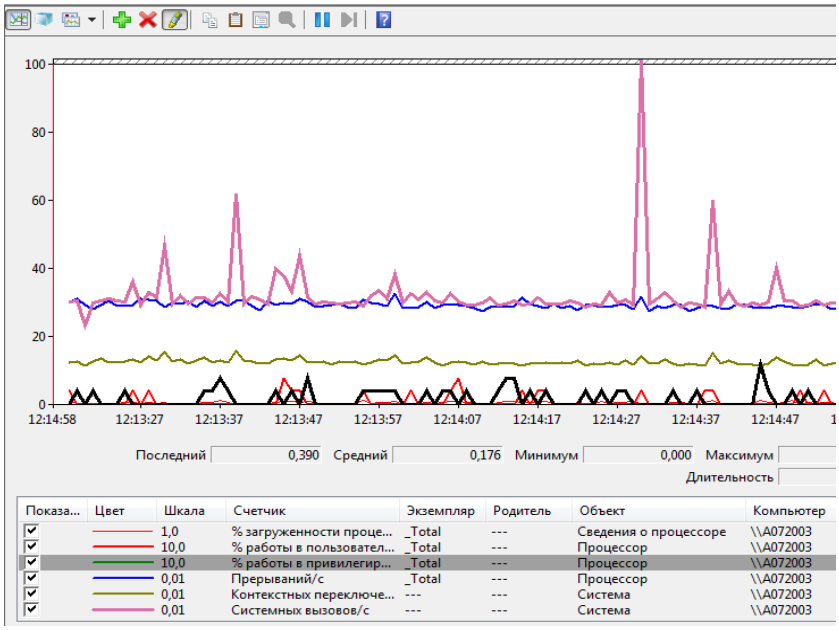

Рис. 6. Результат работы теста № 1 с "пустым циклом"

На рис. 2 приведен текст теста № 1 (Visual Studio, C#), с помощью которого определяется приближенно время выполнения системного вызова Read в операторе Console.ReadLine(). Последний выводит произвольный текст на экран монитора. Тест выполняется параллельно с системным монитором, в котором работает группа сборщиков данных. На рис. 3-5 приведены результаты работы системного монитора, где можно видеть использованные счетчики в группе сборщиков данных.

Заметим, что в затраты процессорного времени на выполнение системных вызовов в данном случае добавляется время работы по организации шагов цикла и преобразованию библиотечной функции WriteLine в системный вызов. Поскольку

системные вызовы выполняются в только режиме ядра, в нашем случае, как видно из таблицы (рис. 5), этот режим составляет примерно 85% общего процессорного времени. Поэтому время выполнения системного вызова необходимо соответственно уменьшить. Кроме того, сам системный монитор потребляет процессорное время, работают системные сервисы и службы. Этот “фон” искажает до определенной степени полученные результаты. Учет фона можно выполнить, удалив в цикле теста № 1 строку Console.Write(t). На рис. 6 можно видеть влияние фона, учитывая, что масштаб представления системных вызовов и контекстных переключений изменился в сто раз, можно сделать вывод о незначительном влиянии фона на результаты теста № 1.

Отметим еще один важный факт, который необходимо учитывать при определении затрат времени на работу ОС. В эксперименте в работу системного вызова включается работа драйвера. Эта работа осуществляется в интересах пользовательского приложения, она зависит целиком от специфики этого приложения, в нашем случае от размера, выводимого на экран текста. По этой причине эти затраты процессорного времени неразумно относить на счет операционной системы.

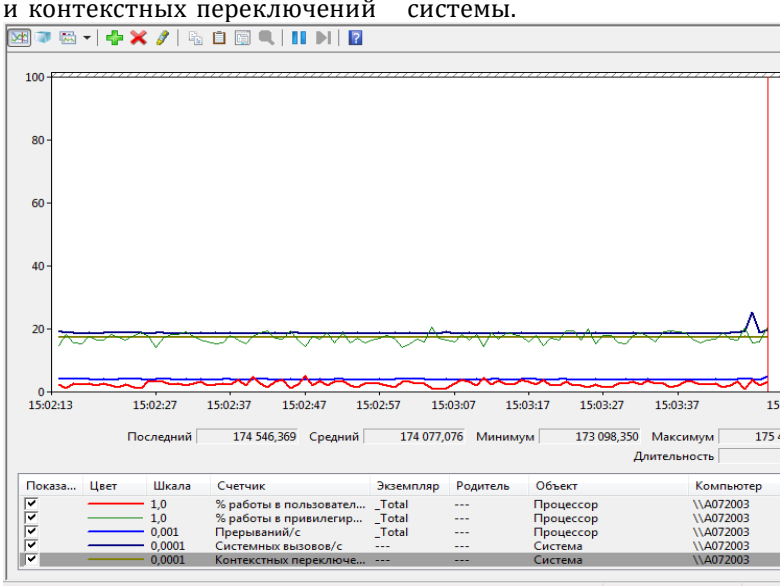

Рис. 7. Результат работы теста №1 в окне системного монитора (вывод на дисплей 64 байт)

	A	B	C	D	E	F
1	Время	\\A072003\Процессор(_Total)\% работы в пользовательском режиме	\\A072003\Процессор(_Total)\% работы в привилегированном режиме	\\A072003\Процессор(_Total)\Прерываний/с	\\A072003\Система\Контекстных переключений/с	\\A072003\Система\Системных вызовов/с
2	15:17:03	2,27	17,98	3834,98	173588,91	185919,67
3	15:17:08	2,57	17,30	3850,82	174036,80	187083,23
4	15:17:13	2,27	17,89	3847,49	173791,32	186255,66
5	15:17:18	1,87	16,82	3763,15	173787,20	186198,77
6	15:17:23	1,79	17,47	3837,32	173591,97	185864,73
7	15:17:28	1,95	17,94	3867,93	173537,52	185893,14
8	15:17:33	3,04	17,08	3797,71	173594,92	185869,02

Рис.8. Фрагмент обработанных в Excel записей журнала результата работы теста № 1 (вывод на дисплей 64 байт)

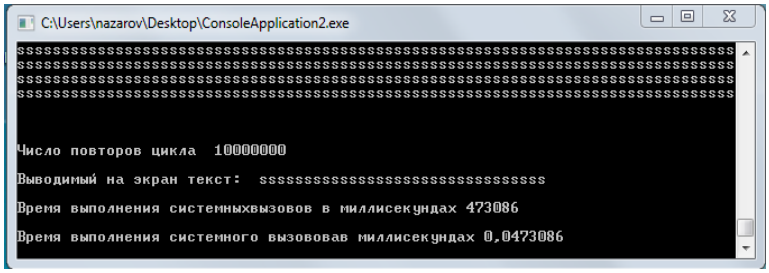

Рис.9. Результат работы теста № 1 в командном окне (вывод на дисплей 64 байт)

На рис. 7–9 представлен результат работы этого же теста, но для случая вывода на дисплей большего количества информации (64 байта вместо двух). В следующем эксперименте

количество выводимой информации увеличено 510 байт (рис. 10). Как следует из рис. 4 и 8, время системного вызова увеличилось с 0,0231 до 0,0473 миллисекунд. Увеличение количества текста до 510 байт приводит к росту системного вызова до 0,228 миллисекунд. Легко подсчитать время, затрачиваемое на выдачу одного байта информации. Во всех экспериментах оно составило 0,0004 миллисекунды. Таким образом, в данных экспериментах можно считать, что выполнение собственно системного вызова по команде WriteLine составляет примерно 0,023 миллисекунды. Это собственно и есть затраты на работу ОС, все остальное в данном случае надо отнести на работу приложения.

Следующий эксперимент (тест № 2) позволяет оценить примерно среднее время выполнения системных вызовов при работе с файлами (создание, запись в файл и удаление файла). Текст теста приведен на рис. 11.

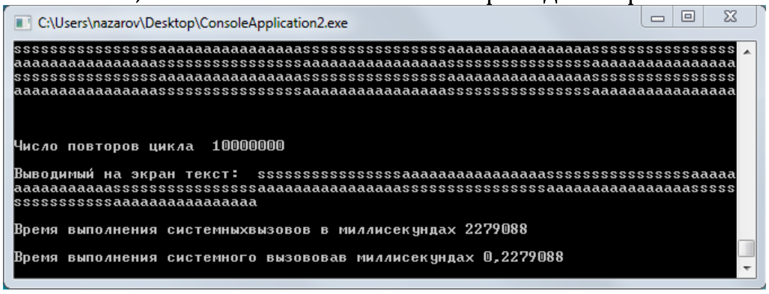

Рис. 10. Результат работы теста № 1 в командном окне (вывод на дисплей 510 байт)

```
class Test2
    { //Программа определения  времени выполнения 3-х системных вызовов:
  //создание файла, запись текста в файл и удаление файла
   static void Main()
   {
     string path = @"c:\temp\MyTest.txt";
       // Если файл существует, удаление файла
       if (File.Exists(path))
       File.Delete(path);
     double k1;
     int t;
     Console.WriteLine("Введите Введите задержку в цикле");
     t = Convert.ToInt32(Console.ReadLine());
     Console.WriteLine("Введите число повторов цикла");
     k1 = Convert.ToDouble(Console.ReadLine());
     Console.WriteLine("Введите текст для записи в файл");
     string Text = Console.ReadLine();
     //Преобразование текста в байты
     Byte[] info = new UTF8Encoding(true).GetBytes(Text);
              //Время начала системных вызовов в миллисек.
//отсчитываемое от 00:00:00, 1 января 0001 года
         DateTime now = DateTime.Now;
         double time = now.Ticks / 10000;
         Console.WriteLine("\nНачало " + time);
     for (double k = k1; k > 0; k--)
     {
       // Создание файла
       using (FileStream fs = File.Create(path))
       // Запись в файл
       fs.Write(info, 0, info.Length);
       Thread.Sleep(t);
       // Удаление файла
       if(k>1) File.Delete(path);
       Thread.Sleep(t);
     }
       //Время завершения системных вызовов в миллисек.
// отсчитываемое от 00:00:00, 1 января 0001 года
        DateTime now1 = DateTime.Now;
        double time1 = now1.Ticks / 10000;
     Console.WriteLine("\nКонец " + time1);
     Console.WriteLine("\nЗадержка в цикле " + t);
     Console.WriteLine("\nЧисло повторов цикла  " + k1);
     Console.WriteLine("\nВыводимый в файл текст:  " + Text);
           Console.WriteLine("\nВремя выполнения системных вызовов"  + "в миллисекундах " + (time1-time));
       Console.WriteLine("\nВремя выполнения трех системных вызовов"
            + "в миллисекундах " + Convert.ToDouble(time1-time-2*t*k1) / k1);
       Console.ReadLine();
     }
   }
```

Рис. 11. Текст теста № 2

Проанализируем, как меняется время выполнения системных вызовов работы с файлами в зависимости от объема записываемого текста. Как следует из рис. 12, после выполнения первого и второго системных вызовов (1 с.в. и 2 с.в.) в течение времени t выполняется запись текста в файл. После этого выполняется третий системный вызов, а затем удаляется созданный файл. На этот промежуток времени тест опять приостанавливается. Если значение времени t превышает время записи и время удаления файла, то результат теста дает "чистое" время выполнения трех основных системных вызовов работы с файлами с учетом неточностей, связанных с преобразованием библиотечных функций в системные вызовы. Если запись и удаление файла превышают время t, результат теста будет неправильным.

На рис. 13 приведен результат выполнения теста № 2 (все установочные параметры видны в командном окне), из которого следует, что время выполнения трех системных вызовов равно 2 миллисекунды. Увеличение объема записываемого текста с 2-х байт (рис. 14) до 510 байт (рис. 14) не меняет результата.

Таким образом, разрабатывая достаточно простые тесты, можно определить затраты процессорного времени на выполнение основных системных вызовов ОС при реализации приложений, а используя системный монитор, получить частоту (и общее количество) системных вызовов. Полученные данные можно использовать для определения коэффициента полезного действия операционной системы.

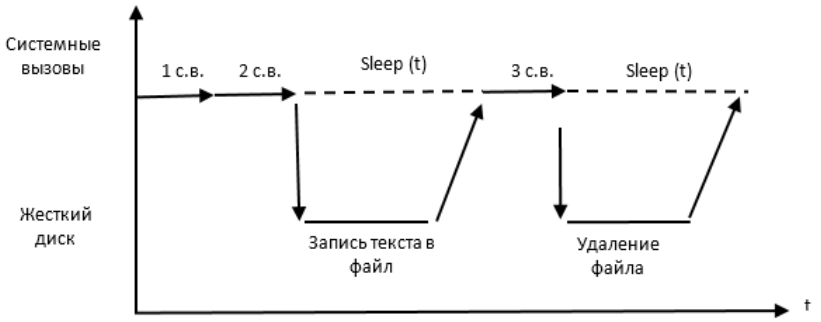

Рис. 12. Схема работы теста № 2

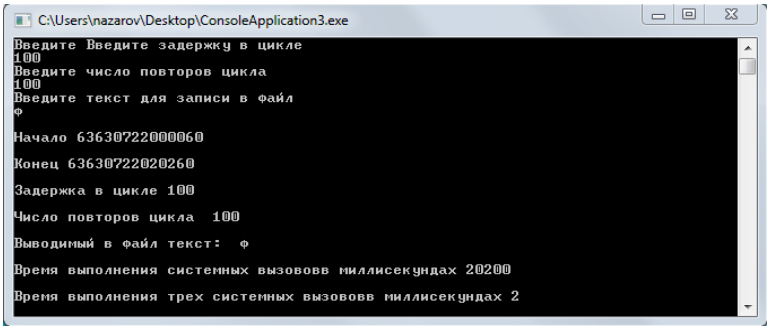

Рис. 13. Результат работы теста № 2 в командном окне (запись в файл одного символа)

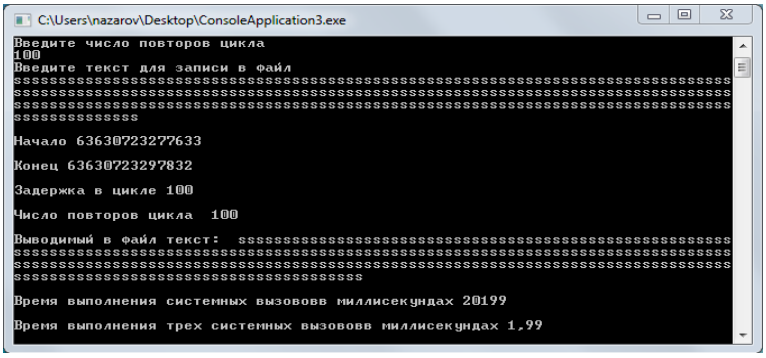

Рис.14. Результат работы теста № 2 в командном окне (запись в файл 255 символов)

6. Выбор оптимального варианта операционной системы

Задача выбора оптимального варианта ОС непосредственно связана с оценкой эффективности некоторого множества возможных вариантов. Подходы к выбору лучшего варианта или сравнению вариантов рассмотрены в статье автора [13]. Следуя формализации задачи оценки эффективности ОС по этой статье, будем считать, что выбор лучшего варианта производится последовательностью парных сравнений вариантов. Пусть имеется два варианта организации (реализации) ОС $(V_l, V_r) \in V$, где V – множество различных вариантов организации ОС, каждый из которых задан определенной совокупностью параметров. При этом полагаем, что все варианты выбора ОС оцениваются одним и тем же набором показателей эффективности. Не уменьшая общности, примем, что лучший вариант ОС характеризуется большим числовым значением каждого показателя. Тогда при выполнении определенных условий между вариантами могут быть установлены отношения строгог нестрогого предпочтения или безразличия.

На практике установить такие отношения не всегда возможно, поскольку если некоторый показатель ОС по варианту V_l больше этого же показателя по варианту V_r, то это не означает, что и другие показатели варианта V_l будут больше, чем в варианте V_r. В этом случае выбор оптимального (из сравниваемых) варианта невозможен без информации о предпочтениях лица, принимающего решение. Описание предпочтений осуществляется во множестве всех оценок W (в том числе гипотетических) с помощью бинарных предпочтений или функции ценности.

Если имеется возможность построения предпочтения во всем множестве показателей эффективности, например, построение лексикографического отношения предпочтения, то превосходство варианта V_l $(V_l \succ lex\, V_r)$ определяется выполнением хотя бы одного из условий:

$$\left.\begin{array}{l} 1)\ W_1(V_l) > W_1(V_r); \\ 2)\ W_1(V_l) = W_1(V_r), W_2(V_l) > W_2(V_r); \\ \quad \ldots\ldots \\ m)\ W_j(V_l) = W_j(V_r),\ j = 1,2,\ldots,m-1, \\ \quad W_m(V_l) > W_m(V_r) \end{array}\right\} \quad (1)$$

Однако на практике такой подход редко удается использовать в силу разнородности показателей и трудности установления предпочтений.

Как показано выше, для оценки эффективности системы целесообразно использовать четыре группы показателей: целевого (боевого) использования (или целевого назначения) $W_ц$, технического совершенства W_m, эргономичности $W_э$ и экономической целесообразности $W_{эк}$. В практически решаемых задачах исследования ОС из множества возможных показателей эффективности

$$W = \{W_ц \cup W_m \cup W_э \cup W_{эк}\} = \{W_j \mid j = 1,2,\ldots,m\} \quad (2)$$

необходимо выбрать такое подмножество показателей $W_{ос} \subset \{W_j \mid j = 1,2,\ldots,m\}$, которое бы наиболее полно характеризовало различные свойства ОС с учетом предъявляемых к ней требований в соответствии с целевым назначением всей системы. При этом желательно стремиться к выполнению условия

$$|W_{OS}| \ll m. \quad (3)$$

Стремление выполнить условие (3) приводит к рассмотрению вопроса возможности введения обобщенного показателя эффективности ОС. Такие обобщенные показатели называются конструируемыми. Подобные показатели обладают существенными недостатками, не позволяющими в большинстве случаев использовать их на практике. Во-первых, они объективно не присущи исследуемой системе. Во-вторых, трудно (и, пожалуй, нереально) определить правило формирования коэффициентов обобщенного показателя.

Возможным методом решения задачи выбора оптимального варианта ОС является метод анализа иерархий (МАИ). Основы метода были разработаны Беллманом Р., Бруком Б.Н. и Бурковым В.Н. в 70-е годы XX столетия, но получил он широкую известность по работам Саати Т. Хорошим подтверждением возможностей МАИ для решения сложных нестандартных задач является статья [18]. В то же время имеются публикации, ставящие под сомнение возможности и правомерность решения оптимизационных задач методом МАИ.

Заключение

Пример решения задачи выбора оптимального варианта СОИ КП методом анализа иерархий, опубликованный в [13] показывает, что в задачах многокритериальной оптимизации в случаях физической несовместимости показателей эффективности, имеющих разную природу, он особенно эффективен. Метод анализа иерархий (МАИ) может быть результативно использован для решения оптимизационных задач различного характера, как в части оценки эффективности различных вариантов структур операционных систем, так и для принятия решения по выбору оптимальной ОС для использования в автоматизированных системах управления различного назначения. Автором данной статьи метод анализа иерархий успешно использован для решения задачи выбора оптимального варианта системы отображения информации коллективного пользования. Однако следует отметить недостатки метода. Во-первых, декомпозиция задачи (разработка иерархии) – нетривиальная работа, требующая глубокого анализа проблемы. Во-вторых, составление таблиц парных сравнений требует наличия высококвалифицированных экспертов, информацию которых необходимо правильно учитывать (исключать какие-то данные, усреднять и т.п.). Таблицы должны быть согласованными. В-третьих, метод весьма трудоемкий – при большом количестве исследуемых вариантов электронные таблицы не спасут от большой работы. Использование пакета СППР “Expert Choice” ненамного облегчит ситуацию. Тем не менее в ряде задач метод анализа иерархий может быть единственно возможным.

Литература

1. Список операционных систем. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/
2. Организация работы Ситуационно-аналитического центра Системного оператора. [Электронный ресурс]. URL: http://so-ups.ru/index.php?id=press_view&tx_ttnews%5 Btt_news%5D=3476
3. Национальный центр управления обороной. [Электронный ресурс]. URL: РФhttp://sneg5.com/nacionalnyj_centr_upravlenija_oboronoj_rf
4. АО «НПО РусБИТех». [Электронный ресурс]. URL: http://astra-linux.com/
5. Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация. 3-е изд. Питер, СПб.: 2007. – 704 с.
6. Назаров С.В. Архитектура и проектирование программных систем: монография / С. В. Назаров. - 2-е изд., перераб. и доп. – М.: ИНФРА-М, 2016. – 376 с.
7. Назаров С.В. Операционные среды, системы и оболочки. Основы структурной и функциональной организации: Учеб. Пособие. – М.: Кудиц-Пресс, 2007. – 504 с.
8. Назаров С.В., Широков А.Г. Современные операционные системы: учебное пособие / С.В. Назаров, А.И. Широков. – 2-е изд., испр. и доп. – М.: Национальный Открытый Университет «ИНТУИТ»: БИНОМ. Лаборатория знаний, 2013. – 367 с.
9. Многопользовательские операционные системы: монография / С. В. Назаров, А. И. Широков; под ред. С. В. Назарова. – М.: Изд. дом МИСиС, 2010. – 193 с.
10. Технологии многопользовательских операционных систем: монография / С. В. Назаров, А. И. Широков; под ред. С. В. Назарова. – М.: Изд. Дом МИСиС, 2012. – 296 с.

11. Tanenbaum Andrew S., Herder Jorrit N., Bos Herbert. Vrije Universiteit, Amsterdam. Can We Make Operating Systems Reliable and Secure? Computer (IEEE Computer Society, V. 39, No 5, May 2006).
12. Назаров С.В., Вилкова Н.Н. Эффективность систем отображения информации коллективного пользования // Электросвязь. 2015. №9. С. 29 – 33.
13. Назаров С.В., Вилкова Н.Н. Выбор оптимального варианта системы отображения информации коллективного пользования // Электросвязь. 2017. №1. С. 60 – 65.
14. Таненбаум Э., Хердер Дж. и Бос Х. Построение надежных операционных систем, допускающих наличие ненадежных драйверов устройств http://citforum.ru/operating_systems/microkernel_tanenbaum/
15. Брукс Фредерик. Мифический человеко-месяц, или Как создаются программные комплексы. Пер. с англ. СПб. Символ-Плюс, 1999. – 304 с.
16. Назаров С.В., Барсуков А.Г. Генерация операционной системы ОС ЕС. – М.: Финансы и статистика,1985. – 175 с.
17. Назаров С.В. Операционные системы специализированных вычислительных комплексов: Теория построения и системного проектирования. М.: Машиностроение,1989. – 400 с.
18. Лобов С.А., Чемиренко В.П. Концепция и метод оценки степени влияния информационной безопасности на национальную безопасность // Электросвязь. – 2016. – № 9. С. 74- 82.

References

1. Spisok operazionnych sistem. [Jelektronnyj resurs]. URL: https://ru.wikipedia.org/wiki/
2. Organizatsiya raboty Situazionno-analiticheskogo centra Sistemnogo operatora. [Jelektronnyj resurs]. URL: http://so-ups.ru/index.php?id=press_view&tx_ttnews%5 Btt_news%5D=3476
3. Nazionalniy center upravleniya oboronoy. [Jelektronnyj resurs]. URL: РФhttp://sneg5.com/nacionalnyj_centr_upravlenija_oboronoj_rf
4. АО «NPO RusBITeh». [Jelektronnyj resurs]. URL: http://astra-linux.com/
5. Tanenbaum E., Vudhall A. Operazionnye sistemy. Razrabotka I realisaziya. 3-ye izd. Piter, SPb.: 2007. – 704 s.
6. Nazarov S.V. Architektura i proektirovanie programmnych sistem: monografia / C.V. Nazarov. – 2-ye izd., pererab. i dop. – M.: INFRA-M, 2016. – 376 s.
7. Nazarov S.V. Operazionnye sredy, sistemy i obolochri. Osnovy strukturnoy i funkzionalnoy organisazii: Ucheb. Posobie. – M.: Kudiz-Press, 2007. – 504 s.
8. Nazarov S.V., Shirokov A.I. Sovremennye operazionnye sistemy: uchebnoe posobie / S.V. Nazarov, A.I. Shirokov. – 2-ye izd., ispr. i dop. – M.: Nazionalniy Otkritiy Universitet «INTUIT»: BINOM. Laboratoria znaniy, 2013. – 367 s.
9. Mnogopolzovatelskie operazionnye sistemy: monografia / S.V. Nazarov, A.I. Shirokov; pod red. S.V. Nazarova. – M.: Izd. dom MISiS, 2010. – 193 s.
10. Technologii mnogopolzovatelskih operazionnyh sistem: monografia / S.V. Nazarov, A.I. Shirokov; pod red. S.V. Nazarova. – M.: Izd. dom MISiS, 2012. – 296 s.
11. Tanenbaum Andrew S., Herder Jorrit N., Bos Herbert. Vrije Universiteit, Amsterdam. Can We Make Operating Systems Reliable and Secure? Computer (IEEE Computer Society, V. 39, No 5, May 2006).
12. Nazarov S.V., Vikova N.N. Effektivnost sistem otobrazhenia informazii kollektivnogo polzovania // Elektrosviaz. 2015. №9. S. 29 – 33.
13. Nazarov S.V., Vikova N.N. Vybor optimalnogo varianta sistemy otobrazhenia informazii kollektivnogo polzovania я // Elektrosviaz. 2017. №1. S. 60 – 65.
14. Tanenbaum A., Herder Dzh. и Bos H. Postroenie nadezhnyh operazionnyh sistem, dopuskaushyh nalichie nenadezhnyh draiverov ustroistv [Jelektronnyj resurs]. URL: http://citforum.ru/operating_systems/microkernel_tanenbaum/
15. Bruks Frederik. Mificheskiy cheloveko-mesiaz, ili Kak sosdautsia programmnye kompleksy. Per. s angl. SPb., Simvol-Plus, 1999. – 304 s.
16. Nazarov S.V., Barsukov A.G. Generazia operazionnoy sistemy OS ES. – M.: Finansy i statistika, 1985. – 175 s.
17. Nazarov S.V. Operazionnye sistemy spezializirovannyh vychislitelnyh kompleksov: Teoria postroenia i sistemnogo proektirovania. M.: Mashinostroenie, 1989. – 400 s.
18. Lobov S,A., Chemirenko V.P. Konsepzia i metod ocenky stepeny vliyania informazionnoy besopasnosty na nazionalnuu besopasnost // Elektrosviaz. – 2016. – № 9. S. 74- 82.

Поступила: 1.06.2017

Об авторе:

Назаров Станислав Викторович, доктор технических наук, профессор, академик Международной академии информатизации, главный специалист, Московский научно-исследовательский телевизионный институт; профессор Департамента анализа данных, принятия решений и финансовых технологий, Финансовый университет при Правительстве РФ, nazarov@mniti.ru.

Note on the author:

Nazarov Stanislav V., doctor of technical sciences, professor, Academician of the International Informatization Academy, Chief Specialist, Moscow Research TV Institute Joint Stock Company; Professor of the Department of Data Analysis, Decision Making and Financial Technologies, Finance University under the Government of the Russian Federation, nazarov@mniti.ru.