Инженерный вестник Дона, №7 (2020)
ivdon.ru/ru/magazine/archive/n7y2020/6557

Алгоритм дополнительной диспетчеризации задач в многопрограммных информационных системах

О.Ф. Козырь

Старооскольский технологический институт им. А.А. Угарова (филиал) «Национальный исследовательский технологический университет «МИСиС»»

Аннотация: Предложена процедура планирования потоков заданий, устанавливающая очередность между группами равноценных задач на основе их сравнения по ряду характеристик. Целью планирования потоков задач являлась минимизация суммарного времени, затрачиваемого на простаивание задач в очередях в ожидании освобождения последовательно используемых ресурсов. В предложенном методе при выборе задач для выполнения приоритет отдается задачам, использующим минимум ресурсов. При этом метод позволяет избегать одновременного запуска на выполнение задач, использующих одни и те же ресурсы. Даны рекомендации по практическому применению процедуры планирования, разработана схема ее взаимодействия со встроенным планировщиком ОС. Для получения наибольшего эффекта от управления планированием заданий в различных средах предложены варианты изменения алгоритма диспетчеризации.
Ключевые слова: многопрограммные информационные системы, системы реального времени, алгоритм, блок-схема, диспетчеризация задач, планирование процессов по структурному критерию, группировка заданий, управление заданиями на основе приоритетов, сравнение многокритериальных задач, правило доминирования Парето, распределение ресурсов системы, дисциплины диспетчеризации, автономные сценарии.

В информационно-вычислительных системах планировщики (диспетчеры) [1, 2] координируют выполнение заданий и потребление ими ресурсов, встроенных в операционную систему. Алгоритм планирования зависит от версии и типа операционной системы (ОС), используемой на предприятии. Но даже самые современные алгоритмы не могут учитывать всех особенностей решаемых предприятием задач, то есть, имея вполне приемлемые результаты при работе в системах общего назначения, в то же время не дают желаемого эффекта в системах реального времени (СРВ) и наоборот [3].

Так в интерактивных системах диспетчер, основанный на многоуровневых очередях с обратной связью, постоянно пересчитывает приоритеты процессов [1] и производит перепланирование потоков в зависимости от происходящих в системе событий [4-6]. В системе же

реального времени абсолютные приоритеты задач не меняются, поэтому их целесообразно пересчитывать только в случае изменения состава задач [3].

А в режиме квантования времени величину кванта выбирают обычно такой, чтобы за этот интервал выполнилось как можно больше задач, что значительно превышает время выполнения служебных задач реального времени [1, 2].

Можно привести еще множество примеров [4- 6], но уже понятно, что планировщики в многопрограммных системах неэффективно используют ресурсы системы и не гарантируют выполнение заданий в назначенный срок. Последнее недопустимо для задач, решаемых в реальном масштабе времени, которые имеют строгие временные рамки на моменты их запуска и завершения.

Одним из способов устранения подобных недостатков является диспетчеризация задач с помощью управления их приоритетами. Разработчиками периодически предлагаются варианты планировщиков, реализующие приоритетные алгоритмы и эффективные для тех или иных информационно-вычислительных систем [7 - 9] .

Большинство алгоритмов планирования, применяемых в современных ОС (в том числе и распространенный метод планирования RR) или предлагаемых разработчиками, требуют линейного упорядочения приоритетов всех подлежащих выполнению задач. Такой подход не оправдывает себя, если для установления приоритетности требуется учитывать несколько, часто несравнимых между собой, характеристик задач. Возможность параллельного выполнения нескольких задач в ОС, например, класса Windows или Unix, позволяет упорядочивать задачи сразу по группе показателей. Поэтому предлагается использовать метод планирования потоков заданий по структурному критерию, устанавливающий очередность между группами равноценных задач на основе их сравнения по ряду

характеристик [10, 11]. При этом будем учитывать, что разрабатываемый планировщик задач должен работать также и в реальном времени, например, на устройствах, предназначенных для автоматического управления в промышленности.

При реализации предлагаемого метода планирования задач по структурному критерию нужно обеспечить выполнение всех задач, функционирующих в данной системе, к заданному сроку, то есть минимизировать суммарное время, затрачиваемое на простаивание задач в очередях в ожидании освобождения последовательно используемых ресурсов.

Процедура диспетчеризации задач с использованием структурного критерия будет использоваться в качестве дополнительного планировщика, работающего параллельно со встроенным диспетчером ОС. Роль такого дополнительного диспетчера задач состоит в определении приоритетов задач и запуске задач на выполнение с учетом совместимости по используемым ресурсам.

Для работы дополнительного планировщика нужно определить перечень интересующих нас характеристик задач и их значения, совместимость задач по ресурсам и назначить задачам приоритеты.

С целью определения значений свойств диспетчируемых задач требуется несколько раз запускать каждую задачу и фиксировать максимальную величину необходимых свойств в специальной таблице, которую назовем блоком управления задачами. Необходимо учесть, что в многопрограммной системе могут решаться задачи с различной периодичностью: от секунды до нескольких недель или месяцев, а также задачи реального времени.

В результате сравнения задач по нескольким числовым свойствам определяется рабочий приоритет задачи, который заносится в базовую

матрицу приоритетов. Первоначально в базовую матрицу приоритетов записывается приоритет, определенный для задачи пользователем при добавлении ее в систему. При повторном выполнении задачи планировщик заново рассчитывает ее приоритет уже с учетом всех интересующих пользователя характеристик, таких как время выполнения, число обращений к файлам БД и др. Рассчитанный приоритет запоминается в базовой таблице приоритетов, и далее задача будет запускаться на выполнение в соответствии с ним.

Чтобы предотвратить одновременный запуск на выполнение совместимых между собой задач поддерживается симметричная булева матрица совместимости $C=(c_{pq})$, каждый элемент c_{pq} которой равен 1, если задачи несовместимы по ресурсам (используют разные ресурсы), и 0 в противном случае:

$$c_{pq}=\begin{cases}1 \text{ - если p - ая задача не совместима по используемым ресурсам с q - ой задачей}\\ 0 - \text{если p - ая и q - ая задачи используют одни и тт же ресурсы}\end{cases}$$

Матрица совместимости заполняется планировщиком параллельно с блоком управления задачами.

При добавлении в систему новых задач, информация о них записывается в блок управления задачами, на основании чего рассчитывается, к какой группе приоритетов они относятся. Информация о совместимости новых задач по ресурсам с другими также добавляется в базовую матрицу совместимости. Базовые матрица приоритетов и матрица совместимости строятся для всех задач системы управления, независимо от заданных даты и времени их запуска. Эта предварительная работа выполняется после установки системы, а также всякий раз, когда пользователь меняет состав задач.

Теперь сформулируем алгоритм процедуры планирования:

1 шаг: На основе базовых матриц создаются текущие (рабочие) матрицы приоритетов и совместимости, куда копируется информация только о задачах, готовых к выполнению.

2 шаг: выполняется процедура сравнения задач по нескольким числовым свойствам. Для установления предпочтения воспользуемся отношением доминирования по Парето, которое предполагает, что сравнение задач из некоторого конечного множества (в нашем случае это блок управления задачами) производится попарно, и что задача z_p строго предпочтительнее задачи z_q, если задача z_p превосходит задачу z_q хотя бы по одному k-ому свойству ($z_{pk} > z_{qk}$), а по всем остальным не хуже неё ($z_{pj} \geq z_{qj};\ j = \overline{1,M};\ j \neq k$) [12]. Задачи z_p и z_q несравнимы между собой, если задача z_p превосходит задачу z_q по значениям одних свойств, а задача z_q превосходит задачу z_p по значениям других. Результатом попарного сравнения задач является сформированная булева матрица предпочтения $B = (b_{pq})$, каждый элемент b_{pq} которой:

$$b_{pq} = \begin{cases} 1, \text{ если p-ая задача предпочтительнее q-ой задачи;} \\ 0, \text{ если p-ая и q-ая задачи не сравнимы.} \end{cases}$$

3 шаг: задачи группируются по рангам, определяющим порядок их выполнения [110]. В матрице предпочтения выбираются задачи, которым соответствуют не содержащие единиц (нулевые) столбцы. Этим задачам присваивается ранг, равный 1. Далее из матрицы удаляются строки и столбцы, соответствующие выбранным задачам. На каждом следующем этапе ранжирования задач в откорректированной матрице снова выбирается группа задач с нулевыми столбцами, которой присваивается ранг, больший на единицу, чем предыдущей группе. Таким образом, в каждой группе оказываются несравнимые между собой задачи. Ранжирование завершается, когда список рассматриваемых задач исчерпан.

4 шаг: задачам одного ранга присваивается один и тот же приоритет, задачам разных рангов присваиваются приоритеты в порядке убывания их рангов.

5 шаг: среди задач с наивысшим приоритетом (наименьшим рангом) выбирается для выполнения группа задач, несовместимых по требуемым информационным ресурсам [10, 11].

С этой целью просматриваются строки матрицы совместимости, помеченные номерами выполняемых задач. Отмечаются все столбцы матрицы, в которых в выделенных строках находятся единичные значения. Номера этих столбцов определяют задачи, несовместимые с выполняемыми задачами. Среди них и выбираются задачи с наиболее высоким рангом для выполнения.

6 шаг: выбранные задачи запускаются на выполнение в режиме разделения времени (квантования) с помощью API-функций базовой ОС. Далее управлением задачами занимается встроенный планировщик ОС.

В каждом периоде повторения диспетчер проверяет, изменился ли набор задач по сравнению с предыдущим периодом. Если набор задач не изменился, то задачи запускаются планировщиком на выполнение с приоритетами в соответствии с их рангами, определенными в предыдущем периоде. Если же в текущем периоде набор задач изменился, тогда выполняется процедура определения новых рангов задач в соответствии с их структурными характеристиками (2 шаг алгоритма).

7 шаг: из текущих матриц приоритетов и совместимости удаляются строки и столбцы, соответствующие выполненным задачам, и для свободных процессоров выбираются задачи в соответствии с откорректированными матрицами. Дополнительный диспетчер проверяет, выполнились ли до конца в отведенный квант времени задачи с наивысшим приоритетом, если

выполнились все задачи, то запускает на выполнение все задачи следующего, более низкого, приоритета. т.е. переходим снова к шагу 5.

8 шаг: при появлении новых задач, время выполнения которых наступило, информация о них заносится в рабочие матрицы приоритетов и совместимости и алгоритм для нового набора задач повторяется, начиная с 1-го шага.

Блок-схема метода диспетчеризации представлена на рис. 1.

В разработанном методе планирования присваиваются наивысшие приоритеты задачам, использующим минимум ресурсов. Преимуществом данного метода является то, что приоритеты задач пересчитываются сравнительно редко, только при изменении состава задач. При этом метод позволяет избегать одновременного выполнения задач, использующих одни и те же ресурсы. Таким образом, разработанный планировщик позволяет экономить время на пересчете приоритетов, просто выбирая для выполнения несовместимые по ресурсам задачи с наивысшим приоритетом [11]. Разработанный диспетчер запускает только задачи из программного комплекса задач верхнего уровня системы управления. Управлением сервисами ОС занимается встроенный диспетчер ОС. На рис.2 схематически изображено взаимодействие разработанного планировщика с планировщиком операционной системы.

В системах, в которых процессы могут быть распределены по разным группам, может применяться следующая процедура планирования. Для каждой группы процессов создается своя очередь, в которой процессы находятся в состоянии «готовность». Этим очередям назначаются фиксированные приоритеты. Для планирования процессов, находящихся в этих очередях, могут применяться самые разные методы [1, 8]. Так,

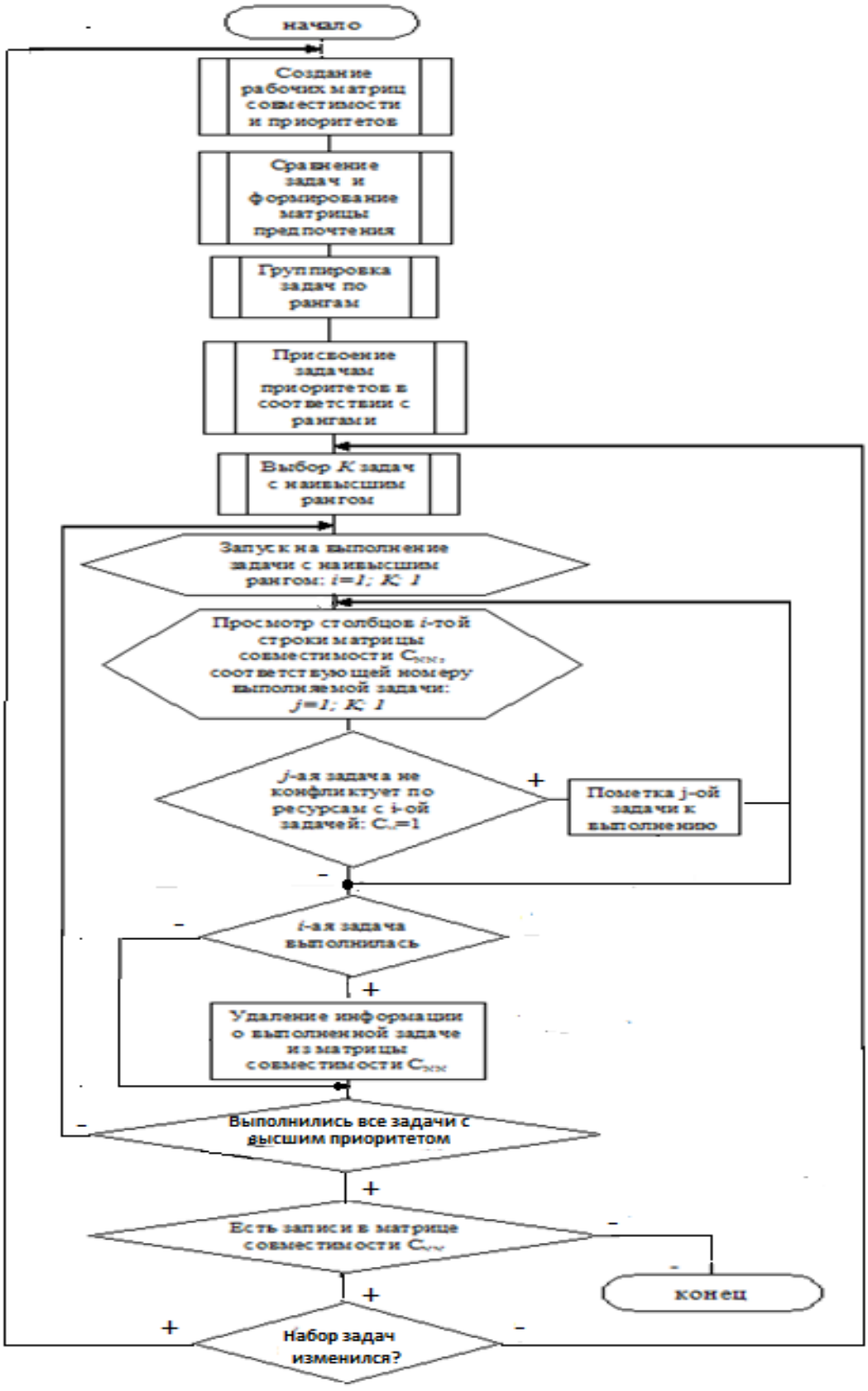

Рис. 1 - Блок-схема процедуры диспетчеризации задач по структурному критерию

например, для интерактивных процессов – карусельный алгоритм RR, а для больших счетных процессов, не требующих взаимодействия с пользователем (фоновых), может использоваться дисциплина многоуровневых очередей FCFS. Это расширяет возможности диспетчеризации в многопрограммных системах, когда для планирования процессов с различными характеристиками применяется наиболее подходящий им метод.

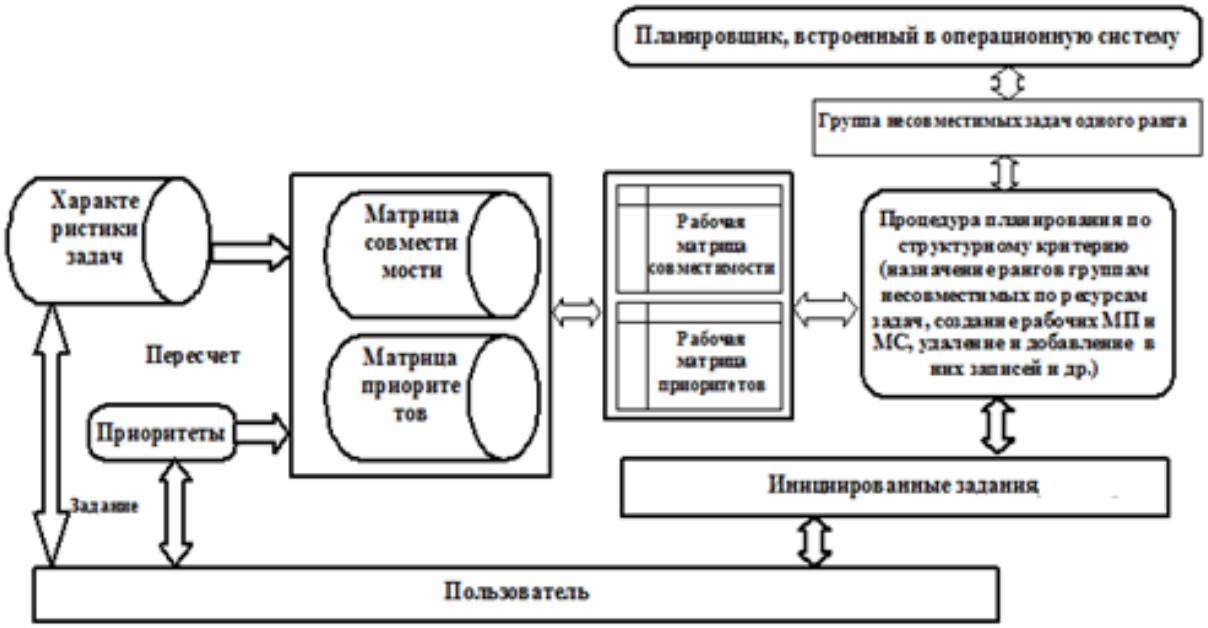

Рис. 2 - Схема взаимодействия дополнительного и встроенного в ОС планировщиков

Дальнейшим развитием метода многоуровневых очередей является добавление к нему механизма обратной связи (Multilevel Feedback Queue). Такие алгоритмы наиболее трудны в реализации, но в то же время, обладают наибольшей гибкостью. Они являются наиболее общим подходом к планированию процессов. Помимо приведенного выше варианта, существует много других разновидностей такого способа планирования [8, 13]. Изменяя такие показатели, как количество очередей для процессов, находящихся в состоянии «готовность», правила постановки активированного процесса в одну из очередей или перевода из одной очереди в другую, методы

планирования, действующие между очередями или внутри них, можно существенно менять поведение вычислительной системы.

Предлагаемая процедура планирования должна быть представлена в виде EXE-программы, написанной на каком-то языке программирования для конкретной ОС (Windows, UNIX, OS/2) , и в своей работе она может использовать практически все возможности как ОС, так и среды ее разработки. Посредством OPC-серверов или объектных моделей MS Office (для Windows), процедурой могут быть использованы все данные системы управления, включая и первичные данные, поступающие с контроллеров. В задачах, функционирующих под управлением диспетчера, может происходить событийное управление выполнением программ, в том числе обработка временных событий с точностью до долей секунды.

Разработанная процедура планирования была проверена на контрольном примере, в качестве которого был взят комплекс задач верхнего уровня слежения за технологическим процессом, которыми управляют три автономных сценария [14]. В результате чего выяснилось, что дополнительный планировщик «сглаживает» потребление ресурсов задачами, позволяя системе более эффективно их расходовать при выполнении последовательности задач.

В ходе имитационного моделирования в среде GPSS, был проведен сравнительный анализ наиболее популярного метода диспетчеризации RR с разработанным. Результаты моделирования показали, что применение разработанной процедуры по сравнению со встроенным в ОС планировщиком увеличивает пропускную способность рабочей станции. Результаты исследований и экспериментов опубликованы в статьях [11, 12].

Литература

1. Столингс В. Операционные системы Изд-во «Вильямс». 4 изд. М., 2002. 793 с.

2. Дейтел Х.М., Дейтел П. Дж., Чофнес Д.Р. Операционные системы. Основы и принципы; пер. с англ. 3-е издание. М.: ООО «Бином-Пресс», 2006. 1024 с.

3. Rajib Mall. 2006. Real-Time Systems: Theory and Practice. IGI Global, 2006. 242 p.

4. Сироткин А.В., Управление формированием информационных потоков в вычислительной системе // Инженерный вестник Дона. 2011. № 4. URL: ivdon.ru/ru/magazine/archive/n4y2011/593

5. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ – Петербург, 2002. 400 с.

6. Culler David E., Singh Jaswinder Pal, Gupta Anoop. 1998. Parallel computer architecture: a hardware / software / approach. // Morgan Kaufmann Publishers. 877 p.

7. Сироткин А.В. Приоритетное планирование процессов информационного обеспечения в АСУП // Инженерный вестник Дона. 2012. № 1. URL: ivdon.ru/ru/magazine/archive/n1y2012/629

8. Cooper K., Dasgupta A., Kennedy K. 2004. New grid scheduling and rescheduling methods in the GrADS project // In Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04). Pp. 199−206.

9. Xiaohui W., Zhaohui D., Shutao Y. 2006. CSF4: A WSRF Compliant Meta-Scheduler // In Proc. of World Congress in Computer Science Computer Engineering, and Applied Computing. Pp. 61-67.

10. Kozyr O.F. The optimization of use of the information recourses in real time systems // European Researcher. 2012. № 5-1 (20). pp. 503-506.

11. Козырь О.Ф. Процедура планирования выполнения автономных сценариев. // Сборник научных и научно-методических докладов Всероссийской научно-практической конференции преподавателей, сотрудников и аспирантов с международным участием 7-8 декабря 2011. Старый Оскол: СТИ НИТУ «МИСиС», 2011. том I, С. 215-220

12. Черноруцкий И.Г. Методы принятия решений. СПб: БХВ – Петербург, 2005. 416 с.

13. Курносов М.Г., Пазников А.А. Инструментарий децентрализованного обслуживания потоков параллельных MPI-задач в пространственно-распределенных мультикластерных вычислительных системах // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2011. № 3 (16). С. 78−85.

14. Филатов В.А., Кривоносов В.А., Козырь О.Ф. Адаптивные автономные сценарии в задачах управления информационными ресурсами предприятия // Инженерный вестник Дона. 2013. № 3. URL: ivdon.ru/ru/magazine/archive/n3y2013/1771

References

1. Stolings V. Operatsionnye sistemy [Operating systems]. Izdatel’stvo “Williams”. 4 izd. M. 2002. 793 p.

2. Dejtel H.M., Dejtel P.Dj., Chofnes D.R. Operatsionnye sistemy. Osnovy i printsipy. [Operating systems. The basics and principles]; per. s angl. 3-e izdanie. M.: OOO “Binom-Press”. 2006. 1024 p.

3. Rajib Mall. 2006. Real-Time Systems: Theory and Practice. IGI Global, 2006. 242 p.

4. Sirotkin A.V. Inzenernyj vestnik Dona. 2011. № 4. URL: ivdon.ru/ru/magazine/archive/n4y2011/593

5. Nemnjugin S.A., O.L. Stesik. Parallel'noe programmirovanie dlya mnogoprocessornykh vychislitel'nykh system [Parallel programming for multiprocessor computing systems]. SPb: BHV-Peterburg, 2002. 400 p.

6. Culler David E., Singh Jaswinder Pal, Gupta Anoop. 1998. Parallel computer architecture: a hardware / software / approach. Morgan Kaufmann Publishers. 877 p.

7. Sirotkin A.V. Inzenernyj vestnik Dona. 2012. № 1. URL: ivdon.ru/ru/magazine/archive/n1y2012/629

8. Cooper K., Dasgupta A., Kennedy K. 2004. In Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04). Pp. 199–206.

9. Xiaohui W., Zhaohui D., Shutao Y. 2006. In Proc. of World Congress in Computer Science Computer Engineering, and Applied Computing. Pp. 61-67.

10. Kozyr O.F. European Researcher. 2012. № 5-1 (20). Pp. 503-506.

11. Kozyr' O.F. Sbornik nauchnykh I nauchno-metodicheskikh dokladov Vserossiyskoy nauchno-prakticheskoy konferentsii prepodavateley, sotrudnikov i aspirantov s mezhdunarodnym uchastiem 7-8 dekabrya 2011. Staryj Oskol: STI NITU "MISiS", 2011. Tom I, Pp. 215-220

12. Chernorutskiy I.G. Metody prinyatiya resheniy [Decision making methods]. SPb: BHV. Peterburg, 2005. 416 p.

13. Kurnosov M.G., Paznikov A.A. Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitel'naya tekhnika i informatika техника и информатика. 2011. № 3 (16). Pp. 78–85.

14. Filatov V.A., Krivonosov V.A., Kozyr' O.F. Inzenernyj vestnik Dona. 2013. № 3. URL: ivdon.ru/ru/magazine/archive/n3y2013/1771