УДК 681.3.06 : 523.164

ОПЕРАЦИОННАЯ СИСТЕМА ДЛЯ АВТОМАТИЗАЦИИ ЭКСПЕРИМЕНТА В РАДИОАСТРОНОМИИ

В. В. Богданов, В. А. Леус

Проблемно-ориентированная операционная система ЭР (для мини-ЭВМ «Электроника-100И») предназначена для использования в нижнем уровне подразделения «Сбор» комплекса автоматизированного обеспечения радионаблюдений на радиотелескопе РАТАН-600. Основные функции ОС ЭР следующие: ведение диалога Пользователь—ЭВМ, реализация мультизадаточного режима, обеспечение мультипрограммирования, управление вводом—выводом данных.

ОС ЭР не производит планирования ресурсов памяти и внешних устройств, не осуществляет динамической загрузки программных модулей, не следит за адресацией операндов и выполняемых команд, не корректирует сбойных кодов и т. п. Подобная автоматизация уместна для мощных ЭВМ в условиях ВЦ коллективного пользования. Она требует дополнительной оперативной памяти, доступности всех регистров и «съедает» значительную часть вычислительного времени, так что представляется неоправданной на машине такого класса, как «Электроника-100И». Ядро ОС ЭР, главной функцией которого является обработка прерываний, заимствовано из работы [1], остальные части представляют собой оригинальную разработку и реализацию.

The problem-oriented operation system ER (for the minicomputer «Electronica-100I») is intended for using in the low level of the unit «acquisition» of the automatic complex for radio observations at the radiotelescope RATAN-600. The main functions of this system are the followings: conducting of the dialogue User—a computer, realization of the multitask regime, providing of multiprogramming, control of data input—output.

The system does not plan the resources of storage and outer devices, does not realize the dynamic load of the program modules, does not follow the operand addressings and the performed commands, does not correct the glitched codes etc. Such an automatization is good for large computers in the computing shops for collective using. It demands additional operative memory, availability of all the registers and «eats» a considerable part of the computer time, therefore it is unjustified for the computer of this class.

The core of this system, the main function of which is interruption processings, was taken from [1], the other parts are the original design and realization.

Общая архитектура. Укрупненная схема ОС ЭР представлена на рис. 1. Здесь выделяется два управляющих модуля — ядро ОС и ДИАЛОГОВЫЙ ЦЕНТР, которые взаимодействуют друг с другом. На ядро «навешены» блоки, обрабатывающие конкретные директивы Пользователя.

На рис. 2 приведена структурная схема ядра и его коммуникаций с другими частями ОС и программой. Одновременно в ведении ОС может находиться до четырех различных программ. Каждая программа вместе со всей вспомогательной, содержательной и управляющей информацией оформлена внутри операционной системы как секция № 1 (2, 3, 4).

Режим функционирования ОС, когда все загруженные программы работают независимо, мы называем мультизадачностью. Предусмотрен и более общий режим, позволяющий загруженным программам взаимодействовать посредством макрокоманд ОС. Именно этот режим естественно (на наш взгляд) называть мультипрограммированием. Режим разделения времени является частным случаем мультизадачности и достигается в рамках ОС ЭР за счет подключения (на правах ВУ) прерывающего таймера.

Когда в выполняемой программе попадается макрокоманда ОС, управление передается блоку ФУНКЦИОНЕР, который распознает макрокоманду и запу-

скает блок, реализующий соответствующую функцию. По прерыванию блок ФИКСАТОР заносит в стек слово состояния PSW текущей программы, а блок ДЕШИФРАТОР устанавливает причину прерывания и передает управление соответствующему обслуживающему блоку. В случае, если пришедшее прерывание оказалось паразитным, ДЕШИФРАТОР передает управление на АКТИВИЗАТОР, который восстанавливает содержимое активных регистров программы, прерванной последней, и запускает ее.

На рис. 3 дана схема организации общения Пользователя с операционной системой (а через нее и с собственными программами) посредством языка ди-

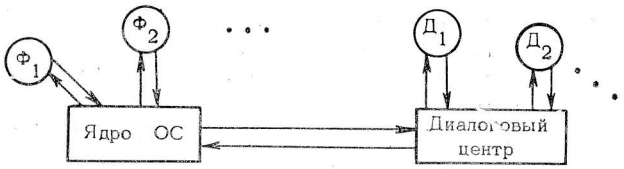

Рис. 1. Общая структура ОС ЭР.

ректив. Последовательность символов вводимой (с Консула или Видеотона) директивы поступает в ДИАЛОГОВЫЙ ЦЕНТР, где проверяется ее синтаксис. Если синтаксис соблюден, производится семантическое распознавание директивы. В случае системной директивы управление передается соответствующему разделу блока СИСТЕМА, реализующему обработку и выполнение. В случае директивы Пользователя управление переходит к блоку ПОЛЬЗОВАТЕЛЬ, где во взаимодействии с блоками РАСПОЗНАВАТЕЛЬ и ВЫБОРЩИК

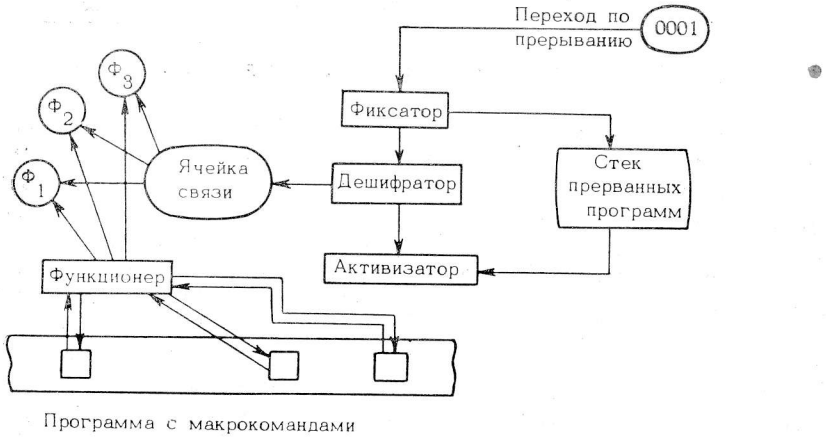

Рис. 2. Схема коммуникаций ядра ОС.

производится полный синтаксический и семантический анализ введенной директивы. При обнаружении ошибок директива передается на автоматическую корректировку блоку РЕДАКТОР, а в случае неисправимой ошибки она аннулируется.

Директива Пользователя, благополучно прошедшая контроль, подвергается преобразованию, и ее содержательная информация поступает в рабочее поле той секции, в которую загружена программа Пользователя. После этого ДИАЛОГОВЫЙ ЦЕНТР настраивается на ожидание приема следующей директивы, и продолжается выполнение программы, которая использует только что введенную информацию.

Функции операционной системы. За основными функциями ОС закреплены номера от 0 до 5. Функция № 0 — о ж и д а н и е и функция № 1 — и з в е щ е н и е в совокупности реализуют мультипрограммирование. Под мультипрограммированием здесь понимается возможность запланированного взаимо-

влияния программ, работающих в разных секциях. Макрокоманда с кодом запроса функции № 0, будучи выполненной при работе программы, приводит к тому, что эта программа прерывается, ее PSW_1 удаляется из стека задействованных процессов, но при этом переводится в единицу соответствующий бит W_1(WAIT) семафорного поля. После чего запускается программа, PSW которой лежало в стеке на втором месте. Если в этой второй программе встретится макрокоманда с запросом функции № 1, то операционная система прервет выполнение второй программы, переведет в единицу бит P_1(POST) семафорного поля первой программы и пустит вторую на продолжение работы. Как только теперь во второй программе встретится макрокоманда о ж и д а н и е, операционная система прервет эту программу, выведет ее из стека задействованных процессов и переведет в 1 бит W_2, а первую программу, поскольку $W_1=P_1=1$, введет в стек задействованных процессов ($PSW_1 \rightarrow$ стек) и запустит на продолжение.

О функции № 2 — обмен с НМЛ — подробно рассказывается ниже, а здесь мы вкратце коснемся остальных функций операционной системы.

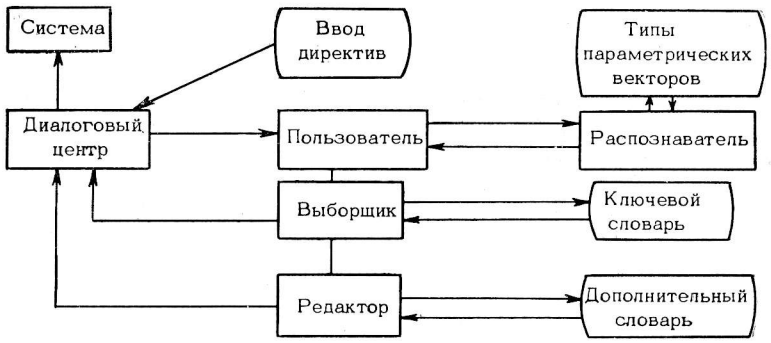

Рис. 3. Организация диалога.

Функция № 3 — запуск программы — осуществляется после того, как в выполняемой «старой» программе встретилась соответствующая макрокоманда. Выполнение старой программы прерывается, формируется слово состояния PSW_2 новой программы, причем данные для него берутся из параметров макрокоманды № 3. Сформированное PSW_2 вносится в стек задействованных процессов, после чего управление передается в старую программу вслед за макрокомандой № 3. Теперь любое прерывание старой программы приведет к запуску новой. В частности, роль такого прерывания может сыграть макрокоманда № 0, поставленная в старой программе после макрокоманды № 3.

Таким образом, перечисленные функции обеспечивают взаимный обмен управляющей информацией между независимо работающими программами и тем самым реализуют мультипрограммный режим. Следует отметить, что помимо чисто управленческих операций функции № 0, 1, 3 дают возможность передавать от программы к программе и небольшую содержательную информацию в виде так называемых кодов сообщения.

Функция № 4 осуществляется ДРАЙВЕРОМ ВВОДА, который разделен на две части — ПУСКАТЕЛЬ и КОНТИНУАТОР. Из ФУНКЦИОНЕРА управление передается ПУСКАТЕЛЮ, настраивающему драйвер на ожидание прерывания от устройства ввода. По прерыванию от ВУ ввода управление через ячейку связи переходит на КОНТИНУАТОР, который считывает в заданное Пользователем буферное поле очередной символ. Если это не последний символ вводимого текста, управление передается АКТИВИЗАТОРУ, запускающему очередную программу из стека прерванных. Таким образом, пока срабатывает медленное ВУ, процессор выполняет вычисления до прихода следующего сигнала прерывания, и простои сводятся к минимуму. По вводе последнего символа КОНТИНУАТОР возвращает драйвер в исходное состояние, и через блок ИЗВЕСТИТЕЛЬ, реализующий функцию № 1, управление передается АКТИВИЗАТОРУ, который запускает прерванную программу. Дело в том, что часто бывает необходимо задерживать выполнение программы с ма-

крокомандой № 4 до окончания ввода. С этой целью вслед за макрокомандой № 4 становится макрокоманда № 0 (ожидание). Блок ИЗВЕСТИТЕЛЬ как раз и «разблокирует» это ожидание.

Реализация функции № 5 — вывод на перфоратор, печать или дисплей — осуществляется ДРАЙВЕРОМ ВЫВОДА, структура и работа которого аналогичны вышеприведенным для случая ввода. И вообще по этому образцу к системе может быть подключено любое ВУ, не применяющее разрыв данными. Для этого нужно разместить в оперативной памяти соответствующий программный драйвер и расширить опрос флагов внешних устройств в ДЕШИФРАТОРЕ. Внешнее устройство, применяющее разрыв данными, подключается обычным образом.

Среди функций ОС ЭР нет настройки по месту загружаемых программ, которые написаны в относительных адресах. При страничной организации памяти потребность в этом не столь остра.

Обмен информацией с внешней памятью последовательного доступа. Функция № 2 осуществляет обмен информацией между ЭВМ и магнитной лентой (МЛ) и выполняется специальной программой ЛЕНТА, которая может работать как автономно, так и в качестве составной части ОС ЭР.

Основу программы ЛЕНТА составляет поисковая система, подчиненная определенному формату данных на МЛ, ввод и вывод которых производятся программой ввода—вывода через КАМАК-систему, а в качестве модуля, управляющего магнитофоном, используется КАМАК-модуль «Драйвер ИЗОТ-5003» 5Р.301.85.

Данные, выводимые на МЛ, упаковываются в файлы, которые располагаются последовательно, разделяются маркером файла (МФ) и имеют специальную структуру. А именно, файл состоит из зон, разделенных межзонными промежутками, причем первая зона в файле имеет номер 0 и является паспортной для всего файла. Все зоны на ленте имеют одинаковую длину, назначенную пользователем в пределах от 16 до 382 машинных слов. Каждое машинное двенадцатизарядное слово записывается на МЛ в две строки по шесть битов в каждой. Таким образом, первые шесть пар строчек зоны содержат служебную информацию, а оставшаяся часть зоны предназначена для информации Пользователя. Нулевая зона хранит имя файла и контрольную сумму информации пользователя. Все остальные зоны хранят только свой порядковый номер в файле и контрольную сумму. Служебную информацию использует программа направленного поиска. Конец всех файлов помечается записанными подряд двумя МФ. Начальная разметка МЛ состоит во внесении МФ сразу после физического маркера начала ленты. Программа ЛЕНТА имеет средства приведения МЛ к требуемому формату. Использование неразмеченной МЛ приводит в автономном режиме к останову, а в системе ОС ЭР — к передаче управления на ДИАЛОГОВЫЙ ЦЕНТР. При этом проводится диагностика ошибок, и соответствующие сообщения выдаются на экран.

В связи с тем что кодировка информации в формате ЕС требует больших затрат времени и ресурсов памяти, оказалось целесообразным проводить процесс перекодировки вне эксперимента.

Поисковая программа позволяет извлекать необходимую справочную информацию из тех лент, которые используются в работе. Эта информация оформляется в виде каталогов, причем каждой МЛ соответствует ровно один каталог, размещенный по усмотрению Пользователя в любом месте ОП, кроме нулевого и первого кубов.

Каталог представляет собой последовательность записей, каждая из которых содержит номер, имя и число зон имеющегося на МЛ файла. Последовательность записей в каталоге отражает естественный порядок расположения файлов на МЛ. Формирование каталога осуществляется автоматически при записи новой информации на соответствующую МЛ. ЛЕНТА размещает каталог только в одном кубе ОП, поэтому число файлов (каждый файл имеет имя, отличное от других) на МЛ не должно превышать 682.

Основная цель программы ЛЕНТА заключается в автоматическом выводе информации из ОП в определенное Пользователем место на МЛ и вводе ее с МЛ порциями размером, кратным длине зон, в любое место ОП, объявленное Поль-

зователем в качестве буфера обмена. Для удобства работы со справочной информацией в системном варианте ЛЕНТЫ предусмотрено использование двух системных директив: справка с МЛ и составление каталогов МЛ. Программа ЛЕНТА контролирует правильность обращения к ней из программы Пользователя, а также ведет контроль функционирования стойки «КАМАК» и магнитофонов, подключенных через нее к ЭВМ. Все ошибки программной и аппаратной частей обнаруживаются, распознаются и выдаются Пользователю в виде сообщений. Сообщения о программных ошибках начинаются литерами «LEN», сообщения об аппаратных ошибках начинаются литерами «PD», после чего следует код ошибки. В системном варианте использования ЛЕНТЫ управление после сообщения об ошибке передается в ДИАЛОГОВЫЙ ЦЕНТР.

Рис. 4. Обмен информацией с МЛ.

Выполнение функции № 2 начинается настройкой ОС на программу ЛЕНТА, во время которой приводится в готовность «КАМАК»-устройство, ЛЕНТА переводится в режим наивысшего приоритета и управление передается блоку УПРАВЛЕНИЕ (рис. 4). Здесь очищаются служебные ячейки и ячейки связи, проверяется правильность обращения к программе ЛЕНТА. В случае обнаружения ошибки выдается сообщение, и управление возвращается ДИАЛОГОВОМУ ЦЕНТРУ. Если ошибки не обнаружены, то начинает работать указанный пользователем блок: СПРАВКА, КАТАЛОГ, РАЗМЕТКА, ЗАПИСЬ, ЧТЕНИЕ. Если задание выполнено без ошибок, то происходит передача управ-

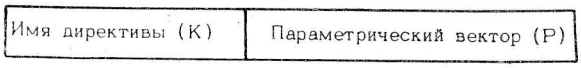

Рис. 5. Строение директивы.

ления АКТИВИЗАТОРУ, который запускает последний прерванный процесс, и далее ДИАЛОГОВОМУ ЦЕНТРУ, который уже готов к приему новых директив.

Язык общения в диалоговом режиме. Семантически завершенными единицами языка общения являются директивы. На рис. 5 дано строение директивы. Возглавляет директиву имя, а далее идет параметрический вектор директивы. Параметрический вектор представляет собой последовательность из произвольного (но фиксированного для данной директивы) числа параметров, каждый из которых является либо числом (*ч* — вещественное число), либо номером (*н* — целое число), либо словом (*с* — цепочка символов, где первый должен быть цифрой).

Системные директивы имеют имена, начинающиеся буквой S. Вот несколько примеров системных директивов: «SECTION 2» — транзитивная директива, указывающая, что последующие директивы относятся к программе, «загруженной» в секции № 2; «SITUATE 31600» — загрузить программу в 3-й куб с адреса 1600. В частности, параметрический вектор директив может быть пустым, как, например, в директиве «START», запускающей загруженную программу на выполнение, или в директиве «STATE», требующей сообщения о состоянии операционной системы.

Имена директив Пользователя не должны начинаться с буквы S. В рамках грамматики языка общения эти директивы составляются самим Пользователем в зависимости от текущих потребностей. Словарь имен и типы параметрических векторов директив Пользователя вводятся в систему заранее, и она, таким образом, оказывается настроенной на диалог в соответствии с лексикой Пользователя. Удобно в качестве имен директив Пользователя применять русские слова, которые будут хорошо выделяться на фоне энглизированных системных директив.

В ОС ЭР используется конкретный вариант языка директив D, ориентированного на диалоговое общение с автоматизированной системой обработки данных. В языке D каждая директива состоит из имени K и следующего за ним параметрического вектора P. Имя директивы представляет собой осмысленную фразу (в нашем случае это одно слово) на естественном языке в алфавите $\mathscr{T}$ и несет качественную информацию. Параметрический вектор есть последовательность качественных и количественных информационных единиц языка директив, для обозначения типов которых используется алфавит A (в нашем случае $A=\{ч, н, с\}$). Компонентами вектора P могут быть целые и действительные числа, адреса, идентификаторы, модификаторы и т. п.

Четверка $\langle V_{\mathscr{T}}, V_A, I, S\rangle$ есть порождающая безконтекстная грамматика G языка директив D (G) [2]. Здесь $V_{\mathscr{T}}$ — множество терминальных символов, частью которого является алфавит $\mathscr{T}$, V_A — множество нетерминальных символов, частью которого является алфавит A, I — начальный (исходный) символ и S — комплекс правил вывода.

Автоматическое редактирование директив. Автоматическое редактирование символьной информации становится теперь все более актуальным в связи с постепенной эволюцией баз данных в базы знаний. Как известно, машины пятого поколения предоставят пользователю возможность общения на языке, приближенном к естественному. В текстах этого языка особую роль будут выполнять ключевые слова — дескрипторы, ошибки в написании которых могут исказить смысл всего задания в целом. На примере языка директив ОС ЭР мы покажем, как повысить информационную помехоустойчивость, поручив машине не только выявление, но и исправление ошибок.

Совокупность используемых директив составляет ничтожную долю всех возможных (при ограничении на длину директивы) текстов языка D (G), поэтому возможна автоматическая коррекция некоторых ошибок, допускаемых при вводе. С формальной точки зрения директива состоит из двух цепочек, одна из которых принадлежит множеству $\mathscr{T}^*$ (множество всех цепочек в алфавите $\mathscr{T}$), другая — множеству A^*. Предлагаемый метод коррекции основывается на введении расстояния ρ_Q в множестве $\mathscr{T}^*$ (соответственно $\rho_{\mathscr{P}}$ в A^*). Это превращает множество цепочек в метрическое (топологическое) пространство [3], где определен критерий близости между цепочками и можно говорить об аппроксимации произвольной цепочки посредством конечного подмножества $Q\subset\mathscr{T}^*$ (соответственно $\mathscr{P}\subset A^*$). Аппроксимировать цепочку $k\in\mathscr{T}^*$ — значит найти в Q такую цепочку q_0, что $\rho\ (k, q_0)=\min \rho\ (k, q)$. Аналогично для значений цепочки в терминальном алфавите параметра некоторого восстанавливаемого типа можно ввести свой критерий близости ρ_R и аппроксимирующее подмножество R.

Зададим частичное отображение $F:\mathscr{T}^*\to A^*$, которое ставит в соответствие каждому имени некоторый параметрический вектор (быть может, пустой). Область определения отображения F назовем к л ю ч е в ы м словарем и обозначим Q, а область значений $\mathscr{P}$ назовем списком типов. Все правила вывода из $\mathscr{P}$, ставящие в соответствие нетерминальному символу P цепочки из A, заменим одним правилом следующего вида: $KP\to KF$ (K). В результате получим новый комплекс правил вывода S' и новую грамматику $G'=\langle V_{\mathscr{T}}, V_A, I, S'\rangle$. Заданное на Q отображение F определяет контекстную зависимость грамматики G', а язык D (G') является подмножеством языка D (G).

После того как осуществлено синтаксическое распознавание директивы в грамматике G, она (директива), тем не менее, может содержать ошибки в смысле грамматики G'. Опишем схему дальнейшего контроля с редактиро-

ванием, позволяющего в удачных случаях восстанавливать верные директивы по искаженным.

1. Проверяется, входит ли K (имя директивы) в ключевой словарь Q. Если $K \in Q$, производится аппроксимация цепочки K посредством Q согласно критерию близости ρ_Q. При однозначном результате K_1 ($K_1 \in Q$) это K_1 вставляется в директиву на место K и производится переход к пункту 2. Если кандидатов несколько — $K_1, K_2, \ldots, K_j, \ldots, K_J$, фактическая цепочка P (последовательность нетерминальных символов, обозначающих типы параметров в параметрическом векторе введенной директивы) аппроксимируется посредством $\mathscr{P}$ согласно критерию $\rho_{\mathscr{P}}$. При однозначном результате, например $F(K_j)$, на место K вставляется K_j, после чего — переход к пункту 2. В случае неоднозначности восстановление считается невозможным и директива аннулируется.

2. Имеем: $K \in Q$ либо K восстановлен действиями пункта 1. Проверяется, будет ли фактическая цепочка P введенной директивы значением отображения F от аргумента K. Если $P = F(K)$, производится переход к пункту 4. Если $P \neq F(K)$, на фактической цепочке отмечаются места ошибок.

3. С учетом отметок пункта 2 восстанавливаемые параметры аппроксимируются согласно своим критериям ρ_R посредством соответствующих множеств R. При любой неоднозначности аппроксимаций директива аннулируется.

4. Исправленная директива предлагается Пользователю в качестве возможной версии. Если восстановленный вариант удовлетворяет Пользователя, то одного нажатия клавиши достаточно, чтобы директива выполнялась. В противном случае директиву нужно ввести заново.

В ОС ЭР реализована упрощенная схема редактирования (рис. 3), использующая лишь аппроксимацию подмножеством Q (ключевой словарь) и $\mathscr{P}$ (типы параметрических векторов). Указанный на рис. 3 дополнительный словарь предназначен для хранения списка кандидатов $\{K_1, K_2, \ldots, K_J\}$.

Выводы. Характерными чертами ОС ЭР являются следующие: 1) мультипрограммирование, 2) направленный поиск информации на носителе последовательного доступа (МЛ), 3) двойное общение — программное и диалоговое, 4) возможность выбора Пользователем лексики диалога, 5) автоматическое редактирование

Операционные системы, которыми снабжаются централизованно поставляемые ЭВМ, конечно же не в состоянии удовлетворить все многообразие требований Пользователя, особенно тогда, когда таковым является научный исследователь. Множество различных конкретных вариантов условий и специфических особенностей применения ЭВМ в научном эксперименте необозримо. По-видимому, адаптация стандартного математического обеспечения и создание оригинальных программных средств автоматизации для экспериментальной астрономии также будут долго еще считаться насущными задачами. Здесь мы позволим себе процитировать интересную работу [4], целиком посвященную этой теме: «разрыв между on-line и off-line видами обработки будет всегда варьировать с переходом от одного эксперимента к другому и в зависимости от имеющихся вычислительных средств. . . В деле заполнения этого разрыва более успешной будет скорее система, которая помогает астроному осмысливать его наблюдения, нежели та, которая берется думать за него». При создании ОС ЭР разработчики старались руководствоваться именно этой идеей.

В заключение мы сердечно благодарим П. Г. Сафонова за большую бескорыстную помощь и З. Е. Петрова за доброжелательность и постоянное внимание к работе.

Литература

1. Седухин С. Г., Желтов М. П., Кашун И. Н. Ядро операционной системы СУММА. — В кн.: Вопросы теории и построения вычислительных систем. Новосибирск, 1977, с. 113—129. (Вычислит. системы, вып. 70).
2. Братчиков И. Л. Синтаксис языков программирования. М.: Наука, 1975. 232 с.
3. Леус В. А. Формальные критерии близости слов. — Проблемы управления и теории информации, 1979, т. 8, № 4, с. 313—325.
4. Davies J. G. Automation and astronomy. — Astronomy a. Astrophysics, 1974, vol. 15, N 3, p. 331—350.

Поступила в редакцию 27.05.85