РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 367 006** (13) **C2**

(51) МПК
G06F 9/48 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2004131022/09, 22.10.2004**

(24) Дата начала отсчета срока действия патента:
22.10.2004

(30) Конвенционный приоритет:
24.10.2003 US 10/693,749

(43) Дата публикации заявки: **10.04.2006**

(45) Опубликовано: **10.09.2009** Бюл. № **25**

(56) Список документов, цитированных в отчете о поиске: **US 6490720 B1, 03.12.2002. RU 2000111466 A, 10.07.2002. US 6006332 A, 21.12.1999. FR 279243 A1, 20.10.2000. US 5327557 A, 05.07.1994. US 6205465 B1, 20.03.2001.**

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
РЕЙ Кеннет Д. (US),
ПЕЙНАДО Маркус (US),
ИНГЛЕНД Пол (US),
КУРИЕН Тхекктхалахал Варугис (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ КОРПОРЕЙШН (US)

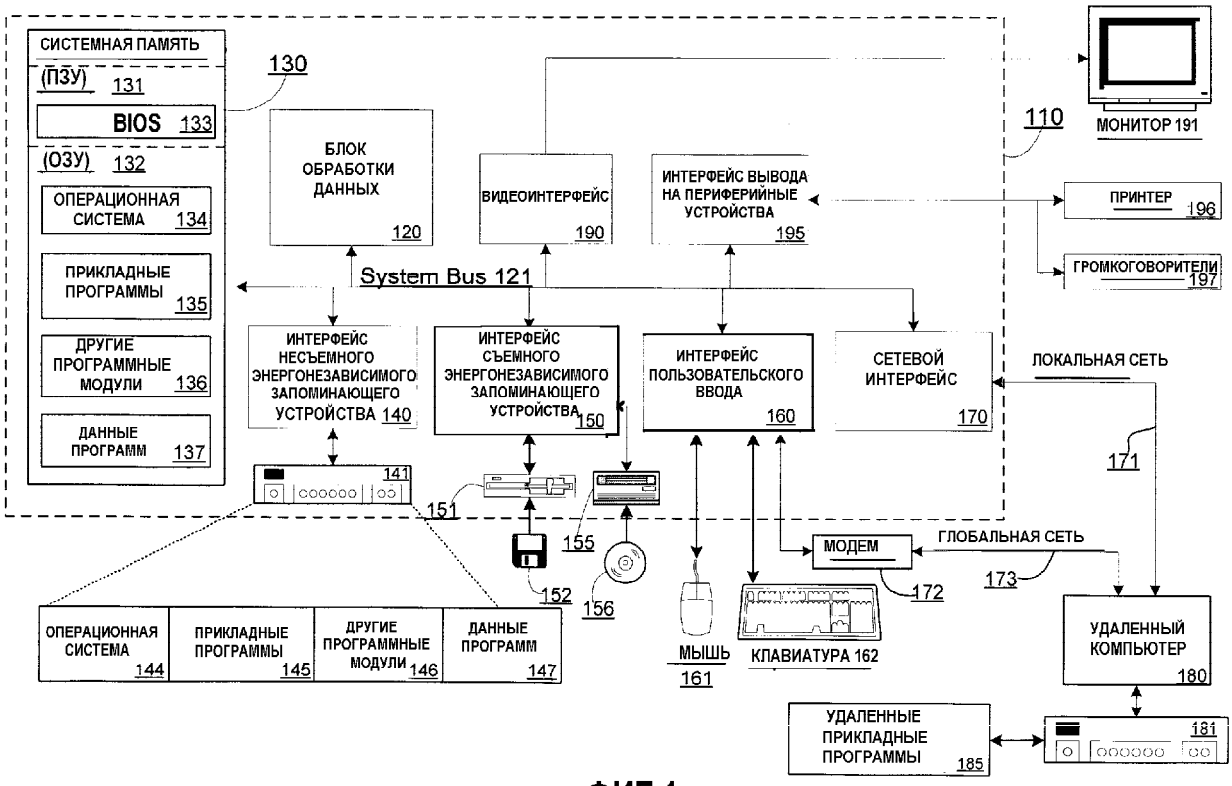
(54) ИНТЕГРИРОВАНИЕ ВЫСОКОНАДЕЖНЫХ ФУНКЦИЙ В ПРИЛОЖЕНИЕ ПОСРЕДСТВОМ РАЗЛОЖЕНИЯ ПРИЛОЖЕНИЯ

(57) Реферат:

Изобретение относится к средствам обработки данных. Техническим результатом является обеспечение универсальной среды обработки данных для выполнения большинства функций. Функции приложения разбивают на две группы в соответствии с тем, содержит или нет заданное действие обработку уязвимых данных, создают отдельные программные объекты (процессоры) для выполнения действий, относящихся к двум упомянутым группам, доверенный процессор обрабатывает защищенные данные и исполняет в высоконадежной среде. Когда другой процессор встречает защищенные данные, эти данные передаются в доверенный процессор. Данные заключают в оболочку таким способом, чтобы можно было направлять данные в доверенный процессор и предотвратить дешифрование данных любым другим объектом, кроме доверенного процессора. Инфраструктура заключает объекты в оболочки, направляет объекты в надлежащий процессор и позволяет заверять целостность объектов в обратной доверенной последовательности до базового компонента, который известен как доверенный компонент. 4 н. и 29 з.п. ф-лы, 8 ил.

RU 2367006 C2

RU 2367006 C2



ФИГ.1

RU 2367006 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 367 006** (13) **C2**

(51) Int. Cl.
G06F 9/48 (2006.01)

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2004131022/09, 22.10.2004**

(24) Effective date for property rights:
22.10.2004

(30) Priority:
24.10.2003 US 10/693,749

(43) Application published: **10.04.2006**

(45) Date of publication: **10.09.2009 Bull. 25**

Mail address:
129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
REJ Kennet D. (US),
PEJNADO Markus (US),
INGLEND Pol (US),
KURIEN Tkhekktkhalakhal Varugis (US)

(73) Proprietor(s):
MAJKROSOFT KORPOREJShN (US)

(54) INTEGRATION OF HIGHLY RELIABLE FUNCTIONS INTO APPLICATION THROUGH APPLICATION DISSECTION

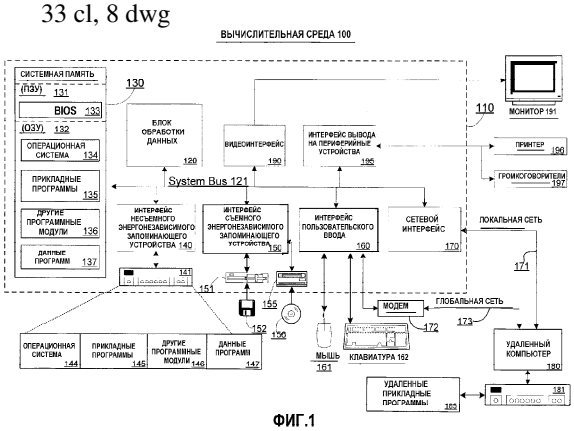
(57) Abstract:

FIELD: physics; computer engineering.

SUBSTANCE: invention relates to means of processing data. An application function is divided into two groups according to whether it involves given a sensitive data processing action. Separate program objects (processors) are created for executing actions, related to the two groups. A confidential processor processes protected data and executes in a highly reliable medium. When the other processor encounters protected data, the data are sent to the confidential processor. The data are encased such that, they can be sent to the confidential processor and decoding of data using any other object except the confidential processor is prevented. The infrastructure encases objects, sends the objects to the appropriate processor and verifies integrity in reverse confidential sequence up to the basic component, known as the confidential component.

EFFECT: provision for a universal medium for processing data for most functions.

33 cl, 8 dwg



ФИГ.1

RU 2367006 C2

RU 2367006 C2

Область техники, к которой относится изобретение

Настоящее изобретение относится, в общем, к области обработки данных. В частности, настоящее изобретение обеспечивает механизм, который поддерживает такое разбиение или разложение приложений, которое позволяет встраивать операции, требующие определенной надежности или безопасности, в обычное незащищенное программное обеспечение.

Предшествующий уровень техники

В области обработки данных с использованием вычислительной техники существует конфликт между системами, которые обеспечивают высокую степень безопасности, с одной стороны, и системами, которые обеспечивают множество функциональных возможностей и высокую степень расширяемости, с другой стороны. Беопасность в области обработки данных с использованием вычислительной техники зависит от способности понимать и прогнозировать поведение компьютерной системы (т.е. поведение как программных, так и аппаратных средств) с высокой степенью достоверности, т.е. способность гарантировать, что система не будет вследствие неосторожного неправильного применения или преднамеренной попытки нарушения защиты проявлять несколько иное поведение, чем запланировано. Например, компьютерная система, которая предназначена защищать от копирования охраняемые авторскими правами материалы, надежна лишь настолько, насколько можно гарантировать, что система будет реально выполнять то, для чего она предназначена. Однако большие открытые архитектуры обычно бывают громоздкими и сложными, что затрудняет анализ их поведения из-за наличия большого числа параметров, которые способны повлиять на поведение. В настоящее время представляется маловероятным, чтобы поведение большой сложной программы, например, универсальной операционной системы или текстового процессора можно было верифицировать с высокой степенью достоверности. Можно написать небольшую программу, поведение которой можно тестировать и верифицировать в разнообразных режимах и для различных видов попыток нарушения защиты, однако такие программы способны выполнять только ограниченный набор функций. Следовательно, существует конфликт между задачами обеспечения большого числа функций и обеспечения высокой степени защиты.

Одно из предложенных ранее решений состоит в том, чтобы исполнять две системы бок о бок - одну большую многофункциональную систему и другую малую систему с высокой степенью защиты. Следовательно, универсальную операционную систему, например WINDOWS XP, можно было бы исполнять совместно с малой высоконадежной операционной системой. Каждый раз, когда в универсальной операционной системе происходит событие, которое необходимо выполнять под строгим управлением и с высокой степенью доверия, соответствующую задачу можно было передавать высоконадежной операционной системе.

Операционные системы обеспечивают среды, в которых можно исполнять другие программы. Однако тот простой факт, что операционные системы могут существовать совместно, не решает напрямую проблему, заключающуюся в том, как заданное приложение может использовать обе среды. В оптимальном варианте приложение должно использовать универсальную среду для выполнения большинства функций (т.е. функций, которые не нуждаются в высокой степени защиты), а высоконадежную среду для выполнения функций, которые требуют высокой степени защиты. Кроме того, желательно использовать две упомянутые среды таким способом, который обеспечивает пользователю комплексное восприятие.

В связи с вышеизложенным, существует потребность в системе, которая лишена недостатков известных систем.

Краткая формулировка сущности изобретения

Настоящее изобретение обеспечивает механизм, при посредстве которого функции приложения можно разложить или разбить на несколько составных частей, т.е. на такие действия, которые нуждаются в безопасности или защите до определенной степени, и на такие действия, которые в этом не нуждаются. В соответствии с настоящим изобретением, приложение осуществляется в виде, по меньшей мере, двух программных объектов: одного программного объекта, который исполняется в полнофункциональной (но ограниченно надежной) среде, и другого программного объекта, который исполняется в высоконадежной (но функционально ограниченной) среде. Когда исполняется объект в полнофункциональной среде, он может встретить данные, которые нуждаются в определенной степени защиты (например, данные могут требовать засекречивания или могут нуждаться в верификации, заключающейся в определении того, что неправомерного использования данных не случилось). Когда программный объект, который исполняется в полнофункциональной среде, встречает такие данные, этот программный объект предписывает передать данные высоконадежной среде. После этого с данными работает программный объект, исполняемый в высоконадежной среде. Любые операции ввода, вывода или хранения данных, которые требуется выполнять во время обработки данных, выполняются с использованием высоконадежной среды, чтобы противодействовать перехвату или неправомерному использованию данных событиями, возникающими вне высоконадежной среды.

Упомянутые две среды обеспечиваются базовым компонентом, который предоставляет инфраструктуру (или «средства связи»), необходимые для информационного взаимодействия двух сред. Например, информационный объект, который следует обрабатывать в высоконадежной среде, может обладать оболочкой, созданной базовым компонентом. Данная оболочка может идентифицировать среду, в которую следует направить информационный объект для обработки, а также может обеспечивать так называемое «клеймо», которое позволяет проверить тот факт, что информационный объект не изменялся с момента его заключения в оболочку базовым компонентом. В соответствии с вышеизложенным, программный объект может передавать информационный объект в базовый компонент, а базовый компонент выполнен с возможностью: (1) определить среду, в которую следует передать информационный объект, и (2) проверить, что неправомерного использования информационного объекта не происходило с момента его создания. Упомянутое последнее из двух действий обеспечивает инфраструктуру, которая позволяет устанавливать достоверность объекта на всех платформах: если объект создается на первой машине (с базовым компонентом), то первый базовый компонент подписывает объект в качестве свидетельства того, что объект является точным объектом, который создан в высоконадежной среде, известной данному базовому компоненту. Когда объект после этого открывается на другой машине, то эта другая машина может проверить подпись и принять решение о том, следует ли доверять подписавшемуся. (Например, некоторые машины и/или базовые компоненты могут лучше обеспечивать надлежащее поведение размещаемых на них сред, чем другие машины; причем каждая машина может решать самостоятельно, доверяет ли она платформе, на которой создан заданный информационный объект).

Ниже приведено описание других признаков настоящего изобретения.

Перечень фигур чертежей

Более полное понимание вышеприведенной краткой формулировки сущности изобретения, а также нижеследующего подробного описания предпочтительных вариантов осуществления изобретения достигается ссылками на прилагаемые чертежи. В целях иллюстрации изобретения на чертежах представлены типичные конструкции, осуществляющие изобретение, однако, настоящее изобретение не ограничивается конкретными описанными способами и средствами. На чертежах:

фиг.1 - блок-схема типичной вычислительной среды, в которой можно осуществить аспекты настоящего изобретения;

фиг.2 - блок-схема приложения, которое разложено на составляющие функции;

фиг.3 - блок-схема, показывающая маршрутизацию данных в разные компоненты приложения;

фиг.4 - блок-схема иллюстративного пользовательского интерфейса для допускающего разложение приложения;

фиг.5 - блок-схема иллюстративной архитектуры, которая поддерживает использование разложенных на компоненты приложений;

фиг.6 - блок-схема иллюстративной иерархии сред, в которых можно использовать разложенное на компоненты приложение;

фиг.7 - блок-схема иллюстративного информационного объекта для использования в среде, которая поддерживает использование разложенного на компоненты приложения;

фиг.8 - блок-схема последовательности операций процесса, посредством которого можно обрабатывать данные в разложенном на компоненты приложении.

Подробное описание изобретения

Общие сведения

Настоящее изобретение обеспечивает механизм, который позволяет разбивать или «разлагать» приложение на незащищенные компоненты и защищенные компоненты и который позволяет этим компонентам работать совместно, чтобы обеспечить пользователю комплексное восприятие приложения. Например, программу обработки текстов можно разбить на незащищенный компонент, который выполняет большую часть компоновки, редактирования, печати, орфографической проверки, грамматической проверки и других функций, которые относятся к текстовому процессору, и защищенный компонент, который обеспечивает отображение и редактирование информационных объектов, которые нуждаются в определенной степени защиты. Незащищенный компонент может исполняться в обычной открытой среде, например в типичной коммерческой операционной системе. Защищенный компонент может исполняться в высоконадежной среде, которая обеспечивает то, что исполнение программного обеспечения определенных типов с высокой степенью надежности будет соответствовать правильному поведению программного обеспечения. Изобретение обладает различными признаками для реализации описанного сценария. Во-первых, изобретение предусматривает для пользователя комплексное восприятие обоих компонентов, чтобы пользователю представлялось, насколько возможно, что он использует единое приложение. Во-вторых, изобретение предусматривает инфраструктуру (или «средства связи»), которая позволяет приложению манипулировать как требующими, так и не требующими защиты данными, а также позволяет использовать такие данные в разных частях.

Что касается пользовательского восприятия, следует учитывать, что пользователь обычно начинает работу с использования незащищенной части приложения. Данные,

создаваемые в процессе такого использования (например, уязвимые текстовые элементы в документах, созданных текстовыми процессорами, уязвимые финансовые показатели электронных таблиц и т.д.), целесообразно интегрировать для пользовательского восприятия определенным способом при посредстве незащищенной части приложения, даже несмотря на то, что незащищенная часть фактически не может отображать данные. Например, для уязвимого или секретного текстового элемента в документе, созданном текстовым процессором, незащищенная часть текстового процессора может отображать окно, которое представляет этот элемент совместно с графическими объектами определенного типа (например, волнистыми линиями, служащими индикатором текста), которая отражает факт существования текста даже несмотря на то, что текст нельзя отобразить. Например, пользователь может щелкнуть мышью на окне или графическом объекте и тогда будет вызвана защищенная часть приложения для отображения в той позиции на экране, где появляется окно. Следовательно, защищенная и незащищенная части приложения становятся интегрированными с точки зрения пользовательского восприятия.

Что касается инфраструктуры, изобретение предлагает механизмы, которые позволяют направлять информационные объекты, находящиеся в одной среде, в другую среду. Как правило, информационный объект, который является секретным или уязвимым и потому должен обрабатываться защищенной частью, будет шифроваться, чтобы его не могла прочитать незащищенная часть. Следовательно, такой объект обычно будет заключаться в набор оболочек каждым компонентом, который участвовал в создании информационного объекта. В предпочтительном варианте каждая оболочка содержит идентификатор компонента, который присоединил оболочку, а также «клеймо», которое обеспечивает возможность верификации сохранности данных внутри оболочки. В предпочтительном варианте внешнюю оболочку присоединяет корневой доверенный элемент на машине, создавшей оболочку, т.е. базовый компонент, который обеспечивает различные среды в отдельной машине и который проверен определенным известным органом на достоверность его поведения. Эта внешняя оболочка позволяет базовому компоненту выполнять функцию маршрутизатора. Следовательно, когда незащищенная часть встречает объект, она может не прочитать объект, но может идентифицировать объект как объект, который следует передать в другую среду для обработки. Следовательно, незащищенная часть передает объект в базовый компонент, который затем направляет объект в надлежащую среду на основании идентификатора среды, предусмотренного в оболочке.

Кроме того, если объект создан на первой машине, а открыт на второй машине, то вторая машина может определить достоверность объекта проверкой подписи, которая составляет часть внешней оболочки. Следует понимать, что, когда базовый компонент заключает объект в оболочку и ставит «клеймо», базовый компонент, по существу, свидетельствует, что, при создании объекта базовый компонент надлежащим образом выполнил свои функции по защите процедуры создания от внешнего неправомерного использования. (Базовый компонент обычно обеспечивает механизмы, которые позволяют высоконадежной среде защищаться от неправомерного использования, например доверенный процессорный модуль (TPM), механизм изоляции памяти и т.д.). Некоторые базовые компоненты могут выполнять данную функцию лучше других, поэтому любая машина, на которой открывается информационный объект, может принимать решение, доверять ли тому, что объект

действительно создан в соответствии с требованиями безопасности, применимыми к объекту. Например, если объект представляет текст, который введен с клавиатуры, то высоконадежная среда может принимать данные ввода с клавиатуры безопасным способом, но способность среды безопасно принимать клавиатурные данные может зависеть от способности базового компонента защищать путь от клавиатуры до высоконадежной среды. Поскольку оболочка содержит «клеймо» или подпись, сформированные конкретным базовым компонентом, то оболочка поддерживает доверенную модель, в которой разные машины могут принимать решения о том, насколько надежным является информационный объект, на основании предположения о безопасности машины, на которой был создан информационный объект.

Далее приведено описание систем, способов и механизмов, которые поддерживают использование разложенных или разбитых на компоненты приложений.

Иллюстративная вычислительная система

Фиг.1 представляет типичную вычислительную среду, в которой можно осуществить аспекты настоящего изобретения. Вычислительная среда 100 представляет собой лишь один пример подходящей вычислительной среды и не должна предполагать никаких ограничений области применения или функций настоящего изобретения. Кроме того, вычислительную среду 100 нельзя трактовать как связанную какой-либо зависимостью или каким-либо требованием, имеющими отношение к какому-либо компоненту или сочетанию компонентов, показанному в иллюстративной операционной среде 100.

Настоящее изобретение способно работать с другими многочисленными средами или конфигурациями вычислительных систем общего или специального назначения. Примеры широко известных вычислительных систем, сред и/или конфигураций, которые могут подходить для использования с изобретением, включают в себя, но не в ограничительном смысле, персональные компьютеры, компьютеры-серверы, карманные или носимые устройства, мультипроцессорные системы, микропроцессорные системы, телевизионные приставки, программируемую бытовую электронику, сетевые персональные компьютеры (ПК), миникомпьютеры, универсальные компьютеры (мейнфреймы), встроенные системы, распределенные вычислительные среды, которые содержат любые вышеперечисленные системы или устройства, и т.п.

Изобретение можно изложить в общем контексте машиноисполняемых команд, таких как программные модули, исполняемые в компьютере. В общем случае, программные модули содержат процедуры, программы, объекты, компоненты, структуры данных и т.д., которые выполняют конкретные задания или реализуют определенные абстрактные типы данных. Изобретение можно также практически реализовать в распределенных вычислительных средах, в которых задания выполняются удаленными устройствами обработки данных, которые связаны через сеть связи или другую среду передачи данных. В распределенных вычислительных средах программные модули и другие данные могут находиться как на локальных, так и на удаленных компьютерных носителях информации, в том числе в запоминающих устройствах.

Как видно из фиг.1, иллюстративная система для реализации настоящего изобретения содержит вычислительное устройство общего назначения в виде компьютера 110. Компонентами компьютера 110 могут быть, но не в ограничительном смысле, блок обработки 120 данных (процессор), системная

память 130 и системная шина 121, которая связывает различные компоненты системы, включая системную память, с блоком обработки 120 данных. Блок обработки 120 данных может представлять собой несколько логических блоков обработки данных, например таких, которые поддерживаются в многопоточном процессоре. Системной шиной 121 может быть любая из шин с разного типа структурами, включая шину памяти или контроллер памяти, периферийную шину и локальную шину, которые используют любую из множества шинных архитектур. В качестве примера, но не ограничения, к шинным архитектурам относятся шина промышленной стандартной архитектуры (ISA), шина микроканальной архитектуры (MCA), шина расширенной промышленной стандартной архитектуры (EISA), локальная шина Ассоциации по стандартизации в области видеоэлектроники (VESA) и шина межсоединения периферийных компонентов (PCI) (называемая также шиной расширения (Mezzanine)). Системную шину 121 можно также реализовать как двухточечное соединение, коммутирующую структуру и т.п. между взаимодействующими устройствами.

Компьютер 110 обычно содержит набор машиночитаемых носителей информации. Машиночитаемые носители информации могут представлять собой любые доступные носители информации, к которым может обращаться компьютер 110, и содержат как энергозависимые, так и энергонезависимые носители информации, как съемные, так и несъемные осители информации. В качестве примера, но не ограничения, машиночитаемые носители информации могут содержать компьютерные носители данных и среды передачи данных. Компьютерные носители данных включают в себя энергозависимые и энергонезависимые, съемные и несъемные носители информации, реализованные любым способом или технологией для хранения такой информации, как машиночитаемые команды, структуры данных, программные модули или другие данные. Компьютерные носители данных включают в себя, но не в ограничительном смысле, оперативное запоминающее устройство (RAM), постоянное запоминающее устройство (ROM), электрически стираемое перепрограммируемое постоянное запоминающее устройство (EEPROM), флэш-память или память, использующую другую технологию, постоянное запоминающее устройство на компакт-диске (CDROM), запоминающее устройство на цифровом многофункциональном диске (DVD) или запоминающее устройство на других оптических дисках, запоминающие устройства на магнитных кассетах, магнитной ленте и магнитных дисках или другие магнитные запоминающие устройства, или любой другой носитель информации, который можно использовать для хранения необходимой информации и к которому может обращаться компьютер 110. Среда передачи данных обычно воплощает машиночитаемые команды, структуры данных, программные модули или другие данные в виде модулированного информационного сигнала, например, несущей волны или другого транспортного механизма и содержит любые среды доставки информации. Термин «модулированный информационный сигнал» означает сигнал, у которого, по меньшей мере, одна характеристика устанавливается или изменяется таким образом, чтобы кодировать информацию в этом сигнале. В качестве примера, но не ограничения, среды передачи данных включают в себя проводные среды, такие как проводная сеть или прямое проводное соединение, и беспроводные среды, такие как акустические, радиочастотные, инфракрасные и другие беспроводные среды. Комбинации любых вышеперечисленных сред и носителей также охватываются понятием «машиночитаемый носитель».

Системная память 130 содержит компьютерный носитель данных в виде энергозависимой и/или энергонезависимой памяти, например постоянного

запоминающего устройства (ROM) 131 и оперативного запоминающего устройства (RAM) 132. Базовая система ввода-вывода 133 (BIOS), содержащая основные процедуры, которые обеспечивают передачу информации между элементами компьютера 110, например, при пуске, обычно хранится в ROM 131. RAM 132 обычно содержит данные и/или программные модули, к которым может непосредственно обращаться и/или с которыми на текущий момент работает блок обработки 120 данных. В качестве примера, но не ограничения, на фиг.1 показаны операционная система 134, прикладные программы 135, другие программные модули 136 и данные 137 программ.

Компьютер 110 может также содержать другие съемные и несъемные, энергозависимые и энергонезависимые компьютерные носители данных. Исключительно для примера на фиг.1 показан накопитель 141 на жестких дисках, который считывает или записывает на несъемный энергонезависимый магнитный носитель, дисковод 151 для магнитного диска, который считывает или записывает на съемный энергонезависимый магнитный диск 152, и дисковод 155 для оптического диска, который считывает или записывает на съемный энергонезависимый оптический диск 156, например на диск CD ROM или на другой оптический носитель. К другим съемным и несъемным, энергозависимым и энергонезависимым средам, компьютерным носителям данных, которые можно использовать в составе иллюстративной операционной среды, относятся, но не в ограничительном смысле, кассеты с магнитными лентами, платы флэш-памяти, цифровые многофункциональные диски, цифровые видеоленты, твердотельные RAM, твердотельные ROM и т.п. Накопитель 141 на жестких дисках обычно подсоединен к системной шине 121 через интерфейс несъемного запоминающего устройства, например интерфейс 140, а дисковод 151 для магнитного диска и дисковод 155 для оптического диска обычно подсоединены к системной шине 121 через интерфейс съемного запоминающего устройства, например интерфейс 150.

Рассмотренные выше и показанные на фиг.1 накопители и дисководы и относящиеся к ним компьютерные носители данных обеспечивают хранение машиночитаемых команд, структур данных, программных модулей и других данных для компьютера 110. Например, на фиг.1 показано, что накопитель 141 на жестких дисках хранит операционную систему 144, прикладные программы 145, другие программные модули 146 и данные 147 программ. Следует отметить, что указанные компоненты могут быть идентичны операционной системе 134, прикладным программам 135, другим программным модулям 136 и данным 137 программ или отличаться от них. В изображенном случае операционной системе 144, прикладным программам 145, другим программным модулям 146 и данным программы 147 присвоены другие номера, чтобы показать, что они представляют собой, по меньшей мере, другие копии. Пользователь может вводить команды и информацию в компьютер 110 посредством таких устройств ввода, как клавиатура 162 и указательный манипулятор 161, обычно называемый мышью, трекболом или сенсорным планшетом. Другими устройствами ввода (не показаны) могут быть микрофон, джойстик, игровой планшет, спутниковая тарелка, сканер и т.п. Упомянутые и другие устройства ввода часто подсоединены к блоку обработки 120 данных через интерфейс 160 пользовательского ввода, который связан с системной шиной, но могут быть подсоединены с использованием других структур интерфейсов и шин, например параллельного порта, игрового порта или универсальной последовательной шины (USB). Монитор 191 или устройство отображения другого

типа также подсоединены к системной шине 121 через интерфейс, например видеоинтерфейс 190. Кроме монитора, компьютеры могут также содержать другие периферийные устройства вывода, например громкоговорители 197 и принтер 196, которые могут быть подсоединены через интерфейс 195 вывода периферийных устройств.

Компьютер 110 может работать в сетевой среде с использованием логических соединений, по меньшей мере, с одним удаленным компьютером, например с удаленным компьютером 180. Удаленный компьютер 180 может представлять собой персональный компьютер, сервер, маршрутизатор, сетевой персональный компьютер (PC), одноранговое устройство или другой обычный сетевой узел и обычно содержит многие или все элементы, указанные выше применительно к компьютеру 110, хотя на фиг.1 показано только запоминающее устройство 181. Показанные на фиг.1 логические соединения включают в себя локальную сеть (LAN) 171 и глобальную сеть (WAN) 173, но могут также включать в себя использование других сетей. Такие сетевые среды широко применяются в офисных и корпоративных компьютерных сетях, внутрикорпоративных сетях и сети Internet.

При использовании в сетевой среде LAN компьютер 110 подсоединен к LAN 171 через сетевой интерфейс или адаптер 170. При использовании в сетевой среде WAN компьютер 110 обычно содержит модем 172 или другое устройство для организации связи через WAN 173, такую как Internet. Модем 172, который может быть внутренним или внешним, может быть подсоединен к системной шине 121 через интерфейс 160 пользовательского ввода или с использованием другого соответствующего механизма. В сетевой среде программные модули, упомянутые применительно к компьютеру 110, или их составные части могут храниться в удаленном запоминающем устройстве. В качестве примера, но не ограничения, на фиг.1 показано, что удаленные прикладные программы 185 хранятся в запоминающем устройстве 181. Специалистам очевидно, что показанные сетевые соединения приведены для примера и что можно применить другие средства организации линии связи между компьтерами.

Разбитое или «разложенное» на компоненты приложение

Программное обеспечение, например прикладная программа, обычно выполняет различные функции и работает с данными разных типов. С этой позиции приложение можно рассматривать как набор различных функций. Возможно, целесообразно разбить приложение на несколько разных выполняемых им функций, чтобы эти различные функции можно было выполнять по отдельности (например, с распределением по средам, обеспечивающим разные уровни безопасности). На фиг.2 показано, как приложение можно разбить на несколько разных выполняемых им функций.

Приложение 135 содержит несколько разных функций 202(1), 202(2), …, 202(n), 202(n+1), 202(n+2), …, 202(n+m). Например, приложение 135 может представлять собой программу обработки текстов, а отдельными функциями могут быть редактирование, просмотр, печать, отслеживание изменений и т.д. Следует отметить, что хотя приложение 135 показано на фиг.2 как обладающее n+m дискретными функциями, существует определенная свобода действий при принятии решения в отношении того, что есть дискретная функция. Например, все операции печати можно считать одной функцией, либо печать можно считать содержащей две или более разных функций (например, функцию печати одной страницы и функцию печати документа в целом). Кроме того, как показано ниже, одну и ту же базовую операцию можно рассматривать как набор из нескольких функций, зависящих от типа

обрабатываемых данных. (Например, базовую операцию просмотра документа можно считать одной из двух функций в зависимости от того, нуждаются ли просматриваемые данные в защите или нет). Следовательно, могут быть две отдельных функции просмотра, одна - защищенная в определенном отношении, а другая - нет. По сути, разбиение программы на составляющие функции является вопросом проведения границ, окружающих разные действия, выполняемые программой (и/или различных типов данных, с которыми программа работает), и конкретные требования к тому, как проводить упомянутые границы, обычно отсутствуют.

В соответствии с одним примером, функции можно сгруппировать вместе, чтобы функции 202(1)-202(n) составляли часть первого «сегмента» 206(1), а функции 202(n+1)-202(n+m) составляли часть второго сегмента 206(2). Возможно, функции было бы удобно сгруппировать так, чтобы единообразно работать с функциями, относящимися к одному сегменту. Например, сегмент 206(1) может содержать такие функции приложения 135, которые относятся к обычным незащищенным данным, а сегмент 206(2) может содержать такие функции приложения 135, которые относятся к секретным или защищенным данным (или данным, которые иным образом требуют определенной степени защиты). Следовательно, функции, составляющие сегмент 206(1), могут выполняться в обычной открытой среде (например, среде, которую обеспечивает обычная коммерческая операционная система), а функции, составляющие сегмент 206(2), могут исполняться в высоконадежной среде. (Ниже приведено более подробное описание высоконадежных сред).

На фиг.3 приведен пример возможного использования сегментов для того, чтобы одно приложение 135 могло обрабатывать как защищенные, так и незащищенные данные. В примере по фиг.3 приложение 135 выполняет операции 302, которые нуждаются в высокой степени защиты (например, операции с секретными данными), а также выполняет операции 304, которые нуждаются в защите более низкой степени или совсем не нуждаются в защите (например, операции без использования секретных данных). («Операции» с данными, в данном примере, могут содержать любые типы обработки данных, например выполнение ввода или вывода данных, выполнение вычислений с данными и т.д.). В приведенном примере функцией, не содержащей работы с секретными данными, манипулирует сегмент 206(1) приложения 135, а функцией, содержащей работу с секретными данными (или данными, которые иным образом требуют определенной степени защиты) манипулирует сегмент 206(2). Например, если приложение 135 является программой обработки текстов, то отображение обычного (незащищенного) документа может выполняться сегментом 206(1), а отображение секретного документа может выполняться сегментом 206(2). В соответствии с другим примером, если приложение 135 представляет собой программу продажи акций, сегмент 206(1) может содержать функцию, которая позволяет пользователю просматривать текущую цену акций, а сегмент 206(2) может позволить пользователю покупать и продавать акции после аутентификации пользователя (в данном случае, пользовательские полномочия для аутентификации и другие данные финансовых счетов пользователя относятся к типу секретных данных).

Следует понимать, что на фиг.3 показано, что различные сегменты приложения можно использовать для работы с секретными и несекретными данными, однако содержание фиг.3 не ограничивается каким-либо конкретным механизмом реализации или использования разбитого на сегменты приложения. Описание примерной

архитектуры приведено ниже со ссылками на фиг.4 - 8, на которых показано разбиение приложения и использование сегментов разбиения таким способом, который обеспечивает пользователю комплексное восприятие.

Пример разбитого на сегменты приложения

В соответствии с вышеизложенным, различные функции приложения можно разбивать в соответствии с любыми критериями. В примере по фиг.3, функции приложения разбиты соответственно тому, работает ли приложение с защищенными или незащищенными данными. Если приложение разбито, то различные сегменты желательно интегрировать так, чтобы обеспечить пользователю единое восприятие. На фиг.4 приведен пример пользовательского интерфейса для разбитого приложения. (Следует понимать, что термин «защищенные» данные относится к данным, которые нуждаются в определенной степени защиты, хотя изобретение не ограничивается каким-либо конкретным типом защиты. Например, данные могут нуждаться в защите в смысле «секретности», и тогда данные могут шифроваться. В другом случае, данные могут нуждаться в защите в смысле возможности верификации, т.е. возможности определения того, что данные не подвергались изменению с некоторого момента начала отсчета времени, и тогда данные можно подписывать).

Пользовательский интерфейс 400 представляет собой интерфейс для программы обработки текстов. (Следует понимать, что программа обработки текстов является удобным примером приложения, хотя и ни в коем случае не единственным примером). В примере по фиг.4 пользовательский интерфейс 400 отображает различные элементы 402(1)-402(5), которые являются частью документа для обработки текста. Элемент 402(1) представляет собой обычный незащищенный текст. Элемент 402(2) представляет собой незащищенный графический объект 404. Элементы 402(3), 402(4) и 402(5) представляют собой элементы защищенного текста. Незащищенный сегмент приложения выполняет всю обработку, которая не включает в себя манипулирование защищенными данными. В настоящем примере к указанной обработке относится отображение (и, возможно, редактирование, печать, сохранение и т.д.) незащищенных элементов 402(1) и 402(2), а также компоновки всех элементов (даже защищенных элементов). Незащищенный сегмент способен компоновать защищенные элементы, поскольку предполагается, что хотя контент (информационно значимое содержимое) элементов 402(3), 402(4) и 402(5) может быть секретным, само существование таких элементов не секретно. Следовательно, незащищенный сегмент способен отображать элементы 402(3)-402(5) как недешифруемые волнистые линии 405, тем самым обеспечивая пользователю, по меньшей мере, какое-то восприятие контента, который не доступен для обработки незащищенным сегментом.

Если пользователь намерен просмотреть контент элементов 402(3), 402(4) и 402(5), то пользователь может, например, щелкнуть мышью на волнистых линиях 405, которые символически представляют контент данного элемента. Указанное действие может вызвать защищенный сегмент. Тогда контент одного из защищенных элементов может быть передан в защищенный сегмент для работы. В соответствии с одним из примеров, защищенный сегмент может после этого отобразить фактический контент в окне, которое налагается на участок для контента, который размещается незащищенным сегментом, например, если защищенный сегмент отображает контент элемента 402(3), то данный сегмент может выполнять указанное действие размещением изображения упомянутого контента элемента поверх зоны, в которой ранее были показаны волнистые линии 405, которые представляют контент элемента 402(3). Хотя вышеописанный сценарий предусматривает взаимодействие

пользователя с двумя различными частями программного обеспечения (т.е. защищенным и незащищенным сегментами), пользователю может представляться, что он взаимодействует с одним приложением, что обеспечивает пользователю комплексное восприятие при взаимодействии с двумя сегментами.

На фиг.4 приведен пример, в котором разбитое на сегменты приложение является программой обработки текстов. К другим примерам относятся:

Электронная таблица, в которую пользователь может вводить формулы, встроенные таблицы и т.д. в незащищенном приложении, но в которой подлежащие обработке реальные данные доступны только защищенному сегменту. Следовательно, незащищенный сегмент отображает некоторые данные метками-заполнителями (например, "xxxx") в каждой ячейке. Вычисление и визуализация данных в ячейках выполняется защищенным сегментом приложения. Если пользователь нажимает кнопку «вычислить», то появляется окно (созданное защищенным сегментом) и отображает в соответствующих строках и столбцах значения ячеек, которые вычислены с использованием базовых данных, доступных защищенному сегменту. В соответствии с данным вариантом осуществления, защищенный сегмент будет обладать вычислительным средством и (простой) стандартной программой визуализации, но не будет нуждаться во встраивании других аспектов пользовательского интерфейса электронных таблиц, окон справки, построителей формул и т.д.

Приложение продажи ценных бумаг является приложением, в котором функции соответственно распределены защищенному и незащищенному сегментам. Например, незащищенный сегмент может отображать графики продаж и информацию об акциях (т.е. общедоступные сведения). Защищенный сегмент, с другой стороны, позволяет пользователю вводить символ продажи акций, цену и число акций, подлежащий покупке или продаже пользователем, и передает указанную информацию в удаленный сервер продаж после предварительной верификации подлинности сервера.

Текстовый процессор, который выдает документ в формате изображения (например, в формате переносимого документа или "PDF"). Защищенный сегмент способен читать и отображать изображение. Такая визуализация документа называется «факсимиле» документа. Базовый документ для обработки текста (т.е. версия документа, которая содержит коды форматирования и текстовые символы вместо самого изображения) и его факсимиле передаются незащищенным сегментом в защищенный сегмент, а защищенный сегмент отображает факсимиле. Пользователь подписывает факсимиле (или дайджест документа и факсимиле) с использованием защищенного подписывающего агента в защищенной среде. Подписанное факсимиле (называемое «факсимиле-0») и базовый документ в формате блока двоично-кодированных данных возвращаются в незащищенный сегмент. Когда верификатор принимает блок двоично-кодированных данных, содержащий документ, факсимиле-0 и подпись факсимиле, верификатор сначала визуализирует новую копию факсимиле (называемую факсимиле-1) на основе базового документа для обработки текста. Верификатор передает факсимиле-0, факсимиле-1, документ редактора Word и подпись в защищенный сегмент, который проверяет подлинность факсимиле и достоверность подписи на факсимиле-0. Хотя описанная модель не обеспечивает секретности документа (поскольку незащищенный сегмент по-прежнему может визуализировать факсимиле-1 документа), она обеспечивает верификацию целостности документа, т.е. проверку того, что документ является первоначально созданным документом и не подвергался изменению.

На фиг.5 показана архитектура, которая поддерживает разбиение приложения. На фиг.5 показаны две среды 502(1) и 502(2), в которых может исполняться программное обеспечение. В данном примере представлены четыре отдельных процессора 504(1), 504(2), 504(3) и 504(4). Здесь термин «процессор» относится не к микропроцессору, а к программному компоненту, который определенным способом обрабатывает данные для приложения. Процессоры 504(1) и 504(2) исполняются в среде 502(1), а процессоры 504(3) и 504(4) исполняются в среде 502(2). Например, процессоры 504(1) и 504(3) могут находиться, соответственно, в незащищенном и защищенном сегментах одного приложения для обработки текстов. Базовый компонент 508 обеспечивает и контролирует ресурсы для сред 502(1) и 502(2), чтобы две среды могли существовать в одной машине. Изобретение не ограничивается каким-либо конкретным типом базового компонента 508; например, базовый компонент 508 может представлять собой монитор (средство мониторинга) виртуальных машин (VMM), внешнее ядро, микроядро или гипервизор (программу управления операционными системами). Например, среды 502(1) и 502(2) могут быть отдельными операционными системами, управляемыми VMM или гипервизором. В соответствии с другим примером, базовый компонент 508 может представлять собой одну из операционных систем, например высоконадежную операционную систему, которая обеспечивает некоторые пользовательские функции, а также обеспечивает разделение между собой и другой (ограниченно надежной) операционной системой.

Одним из компонентов, который исполняется как составная часть базового компонента 508, является монитор 510 ссылок. Монитор 510 ссылок выполняет функцию направления заданного информационного объекта в надлежащую среду для обработки. Например, во время исполнения приложения могут встретиться помеченные данные 506 (либо созданные, либо введенные), и монитор 510 ссылок предписывает направить помеченные данные 506 в надлежащую среду. Помеченные данные 506 связаны с идентификационным признаком, который позволяет монитору 510 ссылок определять, в какую среду следует направить помеченные данные. Кроме того, помеченные данные 506 могут также содержать дополнительные признаки, которые позволяют среде назначения определять, в какой процессор следует передать данные для обработки. Ниже, со ссылкой на фиг.7, приведено описание иллюстративной структуры помеченных данных 506.

Помеченные данные 506 можно встретить (или создать, или ввести) в любом месте, хотя, как правило, помеченные данные будут встречаться в одной среде и передаваться в другую среду. Например, незащищенная часть приложения для обработки текстов может встретить помеченные данные, которые содержат какой-то секретный текст. В этом случае, приложение для обработки текстов может отобразить представление текста, как показано на фиг.4 (например, в виде недешифруемых волнистых линий). Если пользователю потребуется отобразить реальный текст (например, щелчком мышью на области, зарезервированной за данными), то незащищенный сегмент приложения (например, процессор 504(1), исполняемый в среде 502(1)) может представлять помеченные данные 506 в монитор 510 ссылок, который после этого будет передавать данные в среду 502(2). Затем среда 502(2) может направлять данные в надлежащий процессор (например, процессор 504(3)), который после этого может отображать данные в соответствующей зоне на экране в соответствии с описанием, приведенным со ссылкой на фиг.4.

Следует понимать, что в рамках модели, изображенной на фиг.5, приложения состоят, по существу, из процессоров, т.е. компонентов, которые могут работать с

данными различных заданных типов или выполнять задачи определенных категорий, при этом разные сегменты приложения используют разные процессоры. Например, процессоры 504(1) и 504(3) могут представлять собой защищенный и незащищенный процессоры одного приложения (например, приложения для обработки текстов), в котором данные направляются в один из процессоров, 504(1) или 504(3), в зависимости от того, являются данные защищенными или незащищенными.

Следует также понимать, что требования к обязательному наличию сред какого-то типа не существует, но наиболее целесообразно, чтобы одна среда была высоконадежной операционной системой, а другая среда была обычной операционной системой с полным набором услуг. «Высоконадежной» операционной системой является операционная система, которая обеспечивает сравнительно высокую степень надежности в отношении того, что данная система будет правильно выполнять ожидаемые от нее функции. При условии, что все программное обеспечение (включая операционную систему) соответствует спецификации, которая описывает ожидаемую функцию программного обеспечения, и при условии, что все программное обеспечение в какой-то степени подвержено ошибкам или уязвимо к попыткам нарушения защиты, которые вынуждают программное обеспечение вести себя непредвиденным образом, «высоконадежная» операционная система представляет собой операционную систему, которая обеспечивает сравнительно высокую степень доверия тому, что операционная система обеспечит поведение в соответствии со спецификацией. (Большая часть программного обеспечения поступает с точной документально оформленной спецификацией, хотя в контексте настоящего описания под спецификацией может также пониматься не записанное непосредственно понимание поведения программного обеспечения). Понятия высокой надежности и безопасности имеют разный смысл, хотя высокую надежность можно использовать для достижения безопасности. Например, процессор для обработки секретных данных можно спроектировать с расчетом на исполнение в высоконадежной операционной системе, а именно в той степени, насколько способность процессора защитить секретные данные зависит от правильного поведения среды, в которой исполняется процессор (например, от правильного исполнения системных вызовов, правильной изоляции процессов и т.д.), процессор может обеспечить уровень безопасности, который был бы недостижим, если бы процессор должен был исполняться в обычной операционной системе с полным набором услуг. (Проблема высокой надежности не решается без поиска компромисса, поскольку высоконадежная среда может обладать весьма ограниченными функциональными возможностями; т.е. чем больше программа, тем труднее убедиться верификацией в том, что программа будет обеспечивать правильное поведение в самых разных обстоятельствах. Следовательно, коммерческую операционную систему с полным набором услуг, например WINDOWS XP, невозможно считать высоконадежной системой).

С учетом вышеприведенного описания высокой надежности для специалистов очевидно, что зачастую полезно разбить приложение на ограниченно защищенный процессор, который обрабатывает данные, которые не нуждаются в особой защите, и надежно защищенный процессор, который обрабатывает данные, которые нуждаются в усиленной защите. Надежно защищенный процессор может исполняться в высоконадежной среде, а ограниченно защищенный процессор может исполняться в ограниченно защищенной среде.

Типичная архитектура для поддержки разбитого на сегменты приложения

На фиг.6 представлена иллюстративная архитектура 600, в которой может

исполняться разбитое на сегменты приложение. Архитектура 600 содержит базовый компонент 508, который в функциональном отношении предназначен обеспечивать и контролировать различные среды, в которых будут исполняться сегменты приложения. Ранее упоминалось, что базовый компонент 508 может быть осуществлен в разных формах, таких как монитор виртуальных машин (VMM), внешнее ядро, микроядро, гипервизор и т.д. Функции базового компонента 508 заключаются в том, чтобы обеспечивать и контролировать несколько сред (например, операционных систем), а также управлять взаимодействием (ограничивать взаимодействие) между этими средами. Обеспечение и контроль нескольких сред может выполняться с использованием различных методик. Например, базовый компонент 506 может предоставлять автономную «виртуальную машину» для использования каждой из нескольких операционных систем; затем операционные системы управляют «виртуальными аппаратными средствами» виртуальной машины, а базовый компонент 506 выдает в «реальные» аппаратные средства команды, которые основаны на командах (но не обязательно идентичны командам), которые операционные системы направляют в виртуальные машины. (Вообще говоря, таким образом работает традиционная VMM). В соответствии с другим примером, базовый компонент 506 может распределять определенные устройства и определенные сегменты физического адресного пространства машины разным операционным системам и может обеспечивать указанное распределение посредством разрешения каждой операционной системе управлять только выделенными ей устройствами и выделенной ей частью адресного пространства. Изобретение не ограничивается каким-либо конкретным вариантом осуществления базового компонента, а просто допускает применительно к фиг.6, что существует базовый компонент, который способен обеспечивать несколько сред так, чтобы эти несколько сред могли совместно существовать на отдельной машине в некоторой степени изоляции друг от друга.

В примере по фиг.6 базовый компонент 506 обеспечивает и контролирует операционные системы 602(1)-602(4). Операционная система 602(1) представляет собой экземпляр операционной системы WINDOWS XP или другой универсальной операционной системы; операционная система 602(2) представляет собой экземпляр операционной системы Linux; операционная система 603(3) представляет собой «связующую систему», т.е. высоконадежную операционную систему (в вышеописанном смысле), которая обеспечивает ограниченные функции, но при этом высокую надежность того, что эти функции будут выполняться правильно; операционная система 602(4) может представлять собой другую универсальную операционную систему, например OS/2. В целом, базовый компонент 508 может обеспечивать и контролировать ресурсы для произвольного количества операционных систем (или сред другого типа), а четыре операционные системы показаны на фиг.6 просто для примера.

В определенной операционной системе можно исполнять часть программного обеспечения прикладного уровня. Например, программа обработки текстов MICROSOFT WORD (604) и программа обработки электронных таблиц MICROSOFT EXCEL (606) исполняются под управлением операционной системы WINDOWS XP 602(1). Операционная система Linux 602(2), связующая система 602(3) и операционная система OS/2 602(4) могут также исполнять свои собственные прикладные программы. В приведенном примере программы, названные "Word-let" (608) и "Excel-let" (610) исполняются под управлением связующей

системы 602(3). Программа Word-let 608 представляет собой защищенную программу, которая работает с программой WORD 604, когда программа WORD нуждается в обработке защищенных данных. Аналогично, программа Excel-let 610 работает с защищенными данным из программы EXCEL 606. Следовательно, в примере по фиг.4 программа WORD 604 может быть программой (или «процессором»), который обрабатывает незащищенные элементы данных, компонует элементы данных в окне и отображает недешифруемые волнистые линии, которые представляют защищенные элементы. Программа Word-let 608 может быть программой, которая открывает защищенный элемент и визуализирует элемент в зоне окна, которую программа WORD 604 резервирует за данным элементом. По существу, программы WORD 604 и Word-let 608 совместно отражают разбиение (или «разложение») функций приложения для обработки текстов в различных процессорах, которые исполняются в разных средах. Аналогичная взаимосвязь может существовать между программами EXCEL 606 и Excel-let 610, т.е. программа EXCEL 606 работает в электронных таблицах с большинством функций, которые не связаны с защищенными данными, а программа Excel-let 610 работает с защищенными данными по поручению программы EXCEL.

В соответствии с вышеизложенным связующая система 602(3) представляет собой высоконадежную операционную систему, которая может обеспечивать среду, в которой может исполняться доверенное приложение (т.е. приложение, чье поведение является достаточно надежным для того, чтобы данному приложению можно было доверить такие уязвимые операции, как открытие секретных данных). Однако связующая система 602(3) вероятно должна обеспечивать весьма ограниченные функциональные возможности. Следовательно, целесообразно разбить приложение для обработки текстов, например, на программу WORD 604 и программу Word-let 608, чтобы программа WORD могла использовать расширенные функциональные возможности, доступные в универсальной оперативной системе, чтобы обеспечить выполнение расширенных функций, а программа Word-let могла использовать высоконадежный характер среды, предоставляемой связующей системой 602(3), чтобы выполнять (вероятно, ограниченный) набор уязвимых функций с высокой степенью достоверности.

На фиг.6 представлен базовый компонент, который обеспечивает инфраструктуру для поддержки разбиения приложений. Ниже приведено описание некоторых возможностей, которыми может располагать инфраструктура:

Служба регистрации и каталогов: Базовый компонент может обеспечивать интерфейс, посредством которого среды, обеспечиваемые базовым, компонентом могут регистрироваться в базовом компоненте. Базовый компонент может также обеспечивать способы, с использованием которых можно запускать обеспечиваемые среды (либо базовый компонент может быть одной из обеспечиваемых сред). Степень безопасности среды является метаинформацией, относящейся к среде, а базовый компонент может обеспечивать дополнительную услугу доступа к этой информации и ее просмотра структурированным способом.

Разделение: обеспечиваемые среды содержат контекст безопасности, присвоенный им базовым компонентом. Контекст безопасности среды может содержать такие компоненты, как:

a. контекст DAC (избирательного управления доступом), например код-идентификатор среды;

b. контекст (полномочного управления доступом) MAC. В случае с

контекстом MAC, контекст MAC среды содержит метку уязвимости и категорию.

Связь: Базовый компонент регулирует связь между средами с использованием однонаправленных средств передачи данных, принадлежащих базовому компоненту.

a. Средства передачи данных обеспечивают последовательную и надежную доставку сообщений между средами. Для уведомления среды о поступлении данных для транспорта предусмотрен объект синхронизации.

b. Управление доступом: Средства передачи данных являются объектами, принадлежащими базовому компоненту, а способность обеспечиваемой среды считывать из средства передачи данных и записывать в него регулируется моделью управления доступом, которую обеспечивает базовый компонент. В случае DAC, управление указанными действиями осуществляется посредством списка управления доступом (ACL) в средстве передачи данных для действий «считывание» и «запись». В случае MAC, управление указанными действиями осуществляется меткой, присоединяемой к средству передачи данных. Чтобы обеспечить экспорт в многоуровневые устройства, базовый компонент обеспечивает реализацию интерфейсных фильтров защиты данных. Интерфейсные фильтры защиты данных (FESF) представляют собой такие функции, регистрируемые в базовом компоненте, которые могут быть присоединены на сторонах записи и считывания средства передачи данных. В случае с фильтрами FESF записи и считывания, фильтр FESF записи вызывается монитором VMM перед тем, как данные записываются в средство передачи данных (считываются из средства передачи данных) виртуальной машиной (VM). Примером фильтра FESF записи является Seal() (Клеймо()), которое обычно прилагается к данным в высоконадежной среде перед тем, как данные записываются в обычную (невысоконадежную) среду, если действует контекст MAC. Если контекст MAC не действует, то среда должна будет шифровать данные, секретность которых среда полагает уязвимой, перед передачей этих данных в другую среду.

Сообщения: Средства передачи данных транспортируют сообщения по разным средам. Сообщение представляет собой блок двоично-кодированных данных, созданный базовым компонентом и обладающий следующей структурой:

a. Контекст безопасности, присвоенный базовым компонентом: Содержит контекст DAC и MAC. Контекст DAC содержит код-идентификатор создающей среды. Контекст MAC содержит метку уязвимости и категорию создающей среды.

b. Контент: Данные (включая контекст безопасности, присвоенный создащей средой).

Высокоуровневые абстракции: Среды (или приложения, или другие программные объекты, исполняемые в среде) могут самостоятельно реализовать интерфейс вызова удаленных процедур (RPC) с использованием коммуникационной абстракции, предоставляемой базовым компонентом. Описанную услугу может обеспечивать среда или просто код библиотеки. Другой абстракцией, которая может быть предоставлена, является создаваемый между средами интерфейс вызова сонета, который не обязательно должен быть реализован базовым компонентом.

Управление фокусом ввода: Во многих примерах разбитых на сегменты приложений поступление сообщения из одной среды в другую может привести к изменению текущего владельца защищенного устройства ввода или вывода. Управление фокусом ввода обеспечивается базовым компонентом по требованию сред, например, среда может запросить изменения фокуса ввода при поступлении сообщения из другой среды. Затем среда проводит сеанс с защищенным устройством

ввода-вывода, а затем отдает управление. Базовый компонент может осуществить инициированное средой прерывание сеанса ввода/вывода.

Пример информационного объекта для использования с разбитыми на сегменты приложениями

На фиг.7 приведен пример структуры информационного объекта, которая допускает использование информационного объекта с разбитыми на сегменты приложениями. В соответствии с вышеизложенным, когда приложение разбито на сегменты, первый сегмент приложения может встретить информационный объект, который невозможно обработать данным сегментом, а следует передать во второй сегмент. В том случае, когда первый сегмент является «незащищенной» частью приложения, а второй сегмент является «защищенной» частью, целесообразно, чтобы первый сегмент мог распознавать, нуждается ли информационный объект в передаче в другое место для обработки, даже если первый сегмент не может определить никакой другой информации об объекте (например, даже если первый сегмент не может прочитать содержание объекта). Кроме того, в некоторых обстоятельствах, может быть, важно было бы выполнить верификацию того, что информационный объект действительно был создан в защищенных условиях и с тех пор не подвергался модификации (например, следует обеспечить возможность определения того, вводились ли данные в среду, в которой данные на пути от клавиатуры до принадлежащего приложению потока ввода-вывода являются защищенными от «спуфинга» (обманного доступа)). Следовательно, целесообразно также, чтобы способ представления информационного объекта позволял верифицирвать последовательность его обработки.

Информационный объект 700 содержит элемент 702 данных. Элемент 702 данных представляет собой базовый действительный контент, для передачи которого информационный объект 700 является, например, засекреченным документом для текстовой обработки (или частью такого документа), и содержит уязвимые финансовые данные для электронной таблицы, информацию о пароле для аутентификации пользователя при финансовой транзакции и т.д.

Элемент 702 данных последовательно заключается в несколько оболочек каждым уровнем архитектуры, которая осуществляет обработку данных. Каждая оболочка выполняет две функции: (1) оболочка идентифицирует объект (например, приложение, среду и т.д.), в котором элемент был заключен в оболочку, что помогает последующему направлению элемента в данный объект; (2) оболочка содержит такое клеймо элемента, которое позволяет выполнить верификацию того, что элемент не изменялся с момента заключения в оболочку. Например, если программа Word-let 608 создает элемент 702 данных, то программа Word-let может присоединить как цифровую подпись объекта элемента 702 данных, так и заголовок, который идентифицирует программу Word-let как процессор, в который элемент 702 данных должен быть направлен впоследствии, когда элемент потребуется обработать определенным образом. Таким образом, оболочка 704 состоит из подписи и заголовка. Следует отметить, что подписи и заголовки и любые конкретные варианты исполнения не налагают ограничение на оболочки, и существуют различные известные методики, которые позволяют идентифицировать элемент и, следовательно, позволяют выполнить верификацию целостности (т.е. отсутствие изменений) элемента.

В соответствии с вышеизложенным, безопасность такого процессора, как программа Word-let, основана на иерархии доверенных компонентов, т.е. программа Word-let является доверенной потому, что она базируется на высокой

надежности связующей системы; связующая система является доверенной потому, что она использует как основу изолирующие возможности базового компонента и т.д. Следовательно, каждый входящий в иерархию компонент, который находится в последовательности, ведущей до приложения, которое создает элемент 702 данных, заключает элемент в собственную оболочку компонента. Элемент 702 данных, который уже заключен в оболочку, созданную в программе Word-let, затем заключается в оболочку 706 связующей системы. После этого элемент (с двумя оболочками) заключается в оболочку 708 базового компонента 708. Каждая оболочка, по существу, идентифицирует копонент, создавший данную оболочку, а также составляет доступное верификации утверждение (в той степени, насколько доверенным является создавший оболочку компонент), что заключенный в оболочку контент не претерпел изменений с момента заключения в оболочку.

Кроме того, поскольку оболочки вложены одна в другую согласно иерархическому порядку, каждая оболочка пригодна для целей маршрутизации. Следовательно, когда приложение встречает информационный объект 700, внешняя оболочка 708 идентифицирует объект как объект, который следует направить в другой процессор, поэтому объект передается в базовый компонент. Затем базовый компонент использует оболочку, чтобы выполнить верификацию целостности контента внутри оболочки, и использует оболочку 706 следующего уровня для определения того, в какую среду необходимо направить контент (в настоящем случае, в связующую систему). Затем связующая система выполняет верификацию целостности контента внутри оболочки 706, а также использует оболочку 704, чтобы определить программный объект (в настоящем случае, программа Word-let), который будет обрабатывать контент. Затем программа Word-let выполняет верификацию целостности контента внутри оболочки 704. Упоминаемый контент представляет собой элемент 702 данных. Следовательно, структура информационного объекта 700 позволяет направлять и верифицировать элемент 702 данных на каждом этапе в последовательности, ведущей к процессору, который будет обрабатывать элемент 702 данных.

Пример процедуры использования разбитого на сегменты приложения

В процессе по фиг.8 предполагается, что приложение содержит два процессора, обозначенных процессор 1 и процессор 2.

Сначала исполняется процессор 1; в то время как процессор 1 исполняется, процессор 1 встречает информационный объект (802). В соответствии с вышеизложенным, процессор 1, возможно, не сможет прочитать информационный объект, но может распознать информационный объект как некий объект, который следует передать для обработки в другое место. Следовательно, процессор 1 передает информационный объект в монитор (804) ссылок.

Монитор ссылок по самой внешней оболочке информационного объекта определяет, куда следует передать объект данных. В соответствии с вышеизложенным, монитор ссылок (который в предпочтительном варианте представляет собой часть вышеописанного базового компонента) по самой внешней оболочке информационного объекта выполняет верификацию целостности контента внутри данной оболочки. В настоящем примере предполагается, что целостность контента поддерживалась и что монитор ссылок определяет, что информационный объект следует обрабатывать в процессоре 2 (806). (В соответствии с вышеизложенным, монитор ссылок может не знать прямо о существовании процессора 2, но может использовать оболочку для направления объекта в конкретную среду, где объект

затем направляется в процессор 2 получившей его средой. Поскольку описание использования вложенных оболочек с данной целью приведено выше в связи с фиг.7, то в примере по фиг.8 для простоты предполагается, что информационный объект можно как-либо направить в процессор 2 без учета промежуточных этапов маршрутизации, которые могут выполняться при направлении объекта в процессор 2).

После того как выполнено определение, что информационный объект предназначен для процессора 2, то определяется (808), исполняется ли процессор 2. Например, указанное определение может быть выполнено средой, для исполнения в которой предназначен процессор 2 (например, связующей системой, в вышеприведенных примерах). Если процессор 2 не исполняется, то процессор 2 запускается (810). После запуска процессора 2 (или, в ином случае, определения того, что процессор уже исполняется), процессору 2 предоставляется фокус ввода (т.е. возможность выполнять ввод-вывод с устройств ввода и вывода), если процессор 2 предназначен для обработки любых данных ввода или вывода.

Следует отметить, что вышеприведенные примеры приведены только для пояснения и ни в коем случае не подразумевают ограничение настоящего изобретения. Выше настоящее изобретение описано со ссылками на различные варианты его осуществления, однако специалистам в данной области техники очевидно, что использованные в тексте формулировки являются формулировками, предназначенными для описания и пояснения, а не для ограничения. Кроме того, хотя изобретение изложено со ссылками на конкретные устройства, материалы и варианты осуществления, изобретение не должно ограничиваться специфическими деталями, приведенными в настоящем описании; более того, изобретение охватывает все функционально эквивалентные структуры, способы и применения, которые не выходят за пределы объема, определяемого прилагаемой формулой изобретения. Специалисты в данной области техники на основании приведенных в настоящем описании идей изобретения могут вносить в него многочисленные модификации и изменения, не выходящие за пределы объема и сущности изобретения во всех его аспектах.

Формула изобретения

1. Компьютерно-реализованная система для поддержки разбиения приложения, содержащая:

базовый компонент, который обеспечивает функционирование первой среды и второй среды, при этом приложение содержит: первый программный объект, который исполняется в первой среде, причем первый программный объект обрабатывает множество элементов данных и содержит логические средства для идентификации первого элемента из упомянутого множества элементов данных как необрабатываемого первым программным объектом, и второй программный объект, который исполняется во второй среде и который обрабатывает первый элемент из упомянутого множества элементов данных так, чтобы противодействовать неправомерному использованию первого элемента из упомянутого множества элементов данных,

при этом базовый компонент содержит или обеспечивает логические средства, которые принимают первый элемент из упомянутого множества элементов данных от первого программного объекта, и направляет первый элемент из упомянутого множества элементов данных во вторую среду.

2. Система по п.1, в которой первый программный объект предписывает

отображать представление первого элемента из упомянутого множества элементов данных на устройстве отображения, при этом упомянутое представление содержит одно или несколько недешифруемых графических изображений.

3. Система по п.2, в которой упомянутые одно или несколько недешифруемых графических изображений имеют либо одинаковый размер, либо размеры, которые не связаны с контентом первого элемента из упомянутого множества элементов данных.

4. Система по п.1, в которой противодействие неправомерному использованию, обеспечиваемое вторым программным объектом, заключается в том, что вторая среда противодействует помехам отображению первого элемента из упомянутого множества элементов данных посредством записи представления первого элемента из упомянутого множества элементов данных в видеопамять, относящуюся к устройству отображения, чтобы благодаря этому упомянутое представление заменяло любое изображение в той позиции на устройстве отображения, где должно отображаться упомянутое представление.

5. Система по п.1, в которой первый элемент из упомянутого множества элементов данных вводится с клавиатуры, а противодействие неправомерному использованию, обеспечиваемое вторым программным объектом, содержит противодействие неправомерному использованию первого элемента из упомянутого множества элементов данных при передаче от клавиатуры до входного потока второго программного объекта.

6. Система по п.5, в которой второе приложение подписывает первый элемент из упомянутого множества элементов данных, чтобы предотвратить последующее еправомерное использование первого элемента из упомянутого множества элементов данных.

7. Система по п.6, в которой вторая среда подписывает первый элемент из упомянутого множества элементов данных, и подпись, созданная вторым приложением, является указанием на то, что первый элемент из упомянутого множества элементов данных и упомянутая подпись созданы во второй среде.

8. Система по п.1, в которой базовый компонент содержит компонент, который присваивает первый идентификатор второй среде.

9. Система по п.8, в которой первый элемент из упомянутого множества элементов данных содержит или сопровождается первым идентификатором и вторым идентификатором, который идентифицирует второй программный объект.

10. Система по п.1, в которой первая среда соответствует первой спецификации, которая описывает поведение первой среды, а вторая среда соответствует второй спецификации, которая описывает поведение второй среды, при этом уровень надежности того, что вторая среда будет соответствовать второй спецификации, выше уровня надежности того, что первая среда будет соответствовать первой спецификации.

11. Система по п.10, в которой второй программный объект использует как основу поведение второй среды, чтобы противодействовать неправомерному использованию первого элемента из упомянутого множества элементов данных.

12. Система по п.1, в которой базовый компонент представляет собой вторую среду или содержится во второй среде.

13. Компьютерно-реализуемый способ поддержки разбиения приложения, при этом приложение содержит первый программный объект, который исполняется в первой среде, и второй программный объект, который исполняется во второй среде, обеспечивающей первый уровень надежности того, что действия, выполняемые во

второй среде, будут выполнены правильно, при этом способ содержит этапы, на которых

первый программный объект получает данные и определяет, что обработка этих данных не может быть выполнена первым программным объектом,

первый программный объект предписывает передать эти данные второму программному объекту,

второй программный объект обрабатывает упомянутые данные с использованием упомянутого уровня надежности для обеспечения противодействия неправомерному использованию этих данных действиями, совершаемыми вне второй среды.

14. Способ по п.13, в котором противодействие неправомерному использованию содержит противодействие изменению упомянутых данных.

15. Способ по п.14, в котором упомянутые данные должны оображаться на устройстве визуального отображения, а противодействие неправомерному использованию содержит отображение представления упомянутых данных в некоторой позиции на устройстве визуального отображения и замену любого изображения, отличающегося от упомянутого представления, которое визуализируется в упомянутой позиции.

16. Способ по п.13, в котором первый программный объект предписывает отображать представление данных на устройстве визуального отображения, при этом упомянутое представление содержит одно или несколько недешифруемых графических изображений.

17. Способ по п.16, в котором упомянутое представление имеют либо одинаковый размер, либо размеры, которые не связаны с контентом упомянутых данных.

18. Способ по п.16, в котором первый программный объект, или второй программный объект, или комбинация первого программного объекта и второго программного объекта предписывает изменить элементы, отображаемые на устройстве визуального отображения, по меньшей мере, в одном отношении, чтобы позволить просмотреть изображение данных, вырабатываемых вторым программным объектом.

19. Способ по п.14, в котором упомянутые данные вводятся с клавиатуры, а противодействие неправомерному использованию содержит противодействие изменению этих данных при передаче от клавиатуры до входного потока второго программного объекта.

20. Способ по п.13, в котором упомянутые данные содержат первую метку или связаны с первой меткой, которая идентифицирует вторую среду как местоположение, в котором должны обрабатываться упомянутые данные.

21. Способ по п.20, в котором упомянутые данные содержат вторую метку или связаны со второй меткой, которая идентифицирует второй программный объект как процессор для упомянутых данных, а вторая среда направляет упомянутые данные во второй программный объект на основании второй метки.

22. Способ по п.13, в котором вторая среда соответствует первой спецификации, которая описывает поведение второй среды.

23. Способ по п.13, в котором первая среда соответствует второй спецификации, которая описывает поведение первой среды, и первая среда обеспечивает второй уровень надежности того, что действия, выполняемые в первой среде, будут выполняться правильно, при этом второй уровень надежности ниже по сравнению с первым уровнем надежности.

24. Машиночитаемый носитель информации, на котором хранятся код и данные,

чтобы позволить пользователю манипулировать первым и вторым типами данных, причем второй тип данных нуждается в более высокой степени защиты от неправомерного использования по сравнению с первым типом данных, при этом упомянутые код и данные содержат:

первый программный объект, соответствующий первой спецификации, которая описывает поведение первого программного объекта, при этом первый программный объект содержит инструкции для того, чтобы:

работать с элементами первого типа данных,

распознавать элемент второго типа данных как не обрабатываемый первым программным объектом и

предписывать направление упомянутого элемента второго типа данных во второй программный объект; и

второй программный объект, соответствующий второй спецификации, которая описывает поведение второго программного объекта, при этом уровень надежности того, что второй программный объект будет соответствовать второй спецификации, выше уровня надежности того, что первый программный объект будет соответствовать первой спецификации, и второй программный объект содержит инструкции для того, чтобы работать с элементами второго типа данных.

25. Машиночитаемый носитель информации по п.24, в котором первый программный объект работает в первой среде, а второй программный объект работает во второй среде, при этом первая среда соответствует третьей спецификации, которая описывает поведение первой программной среды, а вторая среда соответствует четвертой спецификации, которая описывает поведение второй среды, при этом уровень надежности того, что вторая среда будет соответствовать четвертой спецификации, выше по сравнению с уровнем надежности того, что первая среда будет соответствовать первой спецификации, а надежность того, что второй программный объект будет соответствовать второй спецификации, определяется из уверенности второго программного объекта в поведении второй среды.

26. Машиночитаемый носитель информации по п.24, в котором каждый элемент второго типа данных содержит: первую метку, указывающую на то, что упомянутый элемент второго типа должен обрабатываться во второй среде, и вторую метку, присвоенную второй средой и указывающую на то, что упомянутый элемент второго типа должен обрабатываться вторым программным объектом.

27. Машиночитаемый носитель информации по п.26, в котором первый программный объект предписывает направлять упомянутый элемент второго типа во второй программный объект посредством посылки упомянутого элемента второго типа базовому компоненту, причем базовый компонент присваивает первую метку, вторая метка может быть распознана второй средой, но не базвым компонентом.

28. Машиночитаемый носитель информации по п.24, в котором первый программный объект отображает выходные данные на устройстве визуального отображения, при этом упомянутые выходные данные содержат одну или более позиций на устройстве визуального отображения, в которых следует отображать упомянутый элемент второго типа, а второй программный объект отображает представление упомянутых данных второго типа в упомянутых одной или более позициях.

29. Машиночитаемый носитель информации по п.28, в котором упомянутое представление отображается в упомянутых одной или более позициях второй средой, предписывающей записывать упомянутое представление в видеопамять, относящуюся

к устройству визуального отображения.

30. Машиночитаемый носитель информации по п.24, в котором упомянутый элемент второго типа содержит данные, вводимые с клавиатуры, а действие, предписывающее направлять упомянутый элемент второго типа данных во второй программный объект, заключается в том, что вторая среда транспортирует упомянутый элемент второго типа от клавиатуры во второй программный объект таким способом, который противодействует неправомерному использованию упомянутого элемента второго типа событиями, возникающими вне второй среды.

31. Компьютерно-реализованная система для поддержки разбиения приложения, по меньшей мере, на первый программный объект и второй программный объект, при этом система обеспечивает первую среду и вторую среду, причем первый программный объект исполняется в первой среде, второй программный объект исполняется во второй среде, и система содержит интерфейс прикладного программирования, который предоставляет, по меньшей мере, один из следующих методов:

первый метод, который принимает от первого программного объекта первый информационный объект, который содержит: данные, обрабатываемые вторым программным объектом, и первый идентификатор, присвоенный системой второй среде; и который направляет первый информационный объект во вторую среду на основе первого идентификатора;

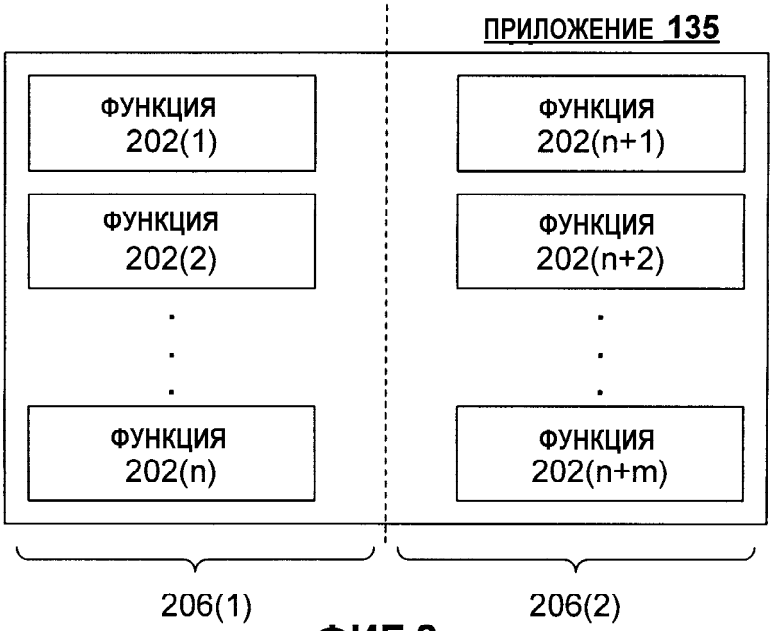
второй метод, который создает второй информационный объект, который содержит: данные, обрабатываемые вторым программным объектом, первый идентификатор и данные аутентификации, которые позволяют в последующем определить, что неправомерного использования второго информационного объекта не происходило с момента его создания вторым методом;

третий метод, который принимает от первой среды второй идентификатор, соответствующий второму программному объекту, и который предписывает создать экземпляр второго программного объекта; и четвертый метод, который:

принимает от первой программной среды третий информационный объект и третий идентификатор, соответствующий первому программному объекту, предписывает создать экземпляр первого программного объекта на основании принятого третьего идентификатора и предписывает первому программному объекту работать с третьим информационным объектом.

32. Система по п.31, в которой первая среда соответствует первой спецификации, которая описывает поведение первой среды, при этом вторая среда соответствует второй спецификации, которая описывает поведение второй среды, причем существует первый уровень надежности того, что первая среда будет соответствовать первой спецификации, существует второй уровень надежности того, что вторая среда будет соответствовать второй спецификации, и второй уровень надежности выше по сравнению с первым уровнем надежности.

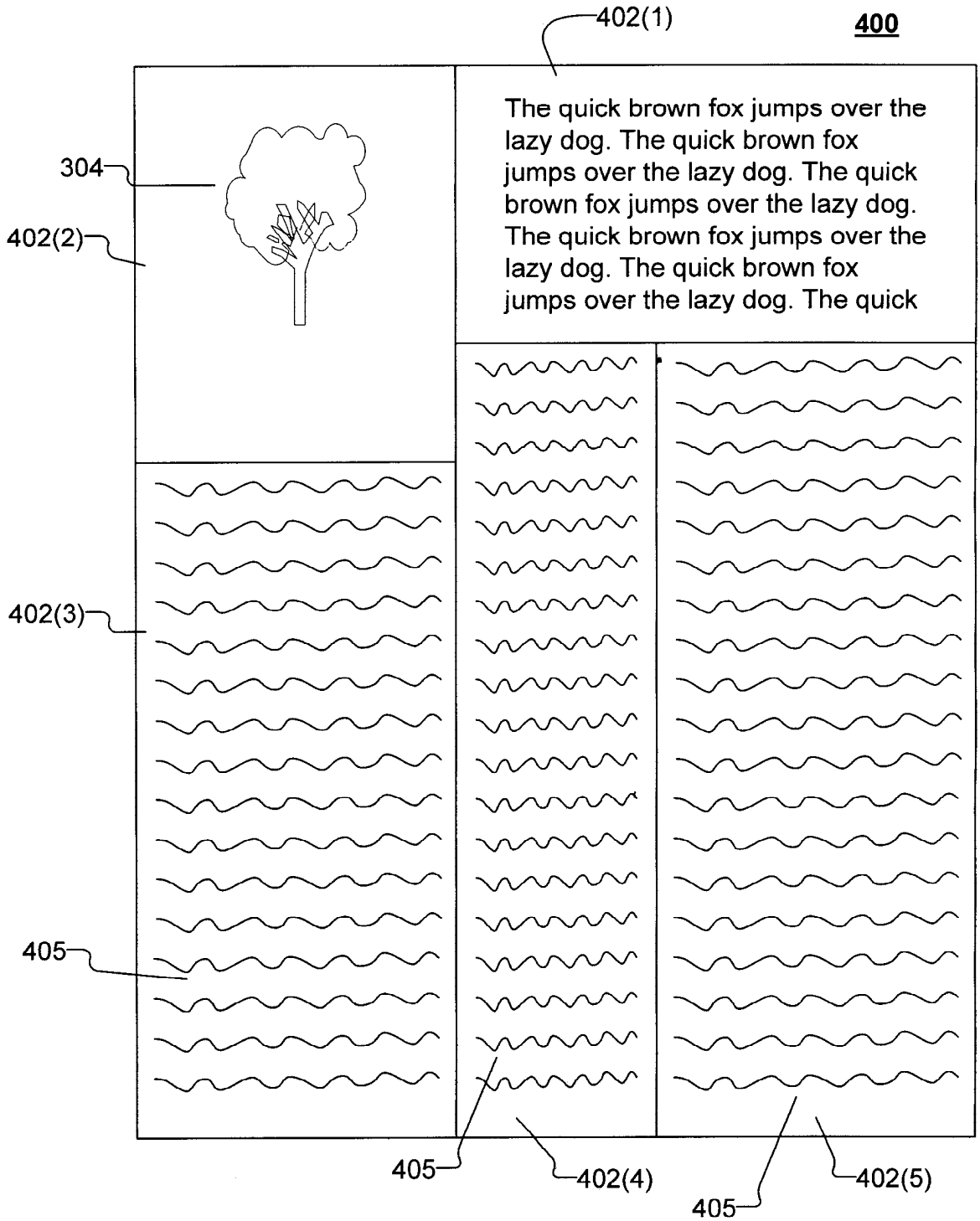
33. Система по п.32, в которой второй программный объект обеспечивает надежность того, что второй программный объект осуществит защиту данных, при этом упомянутая надежность обеспечивается, по меньшей мере, частично на основе поведения второй среды.


ПРИЛОЖЕНИЕ 135
ФУНКЦИЯ 202(1)
ФУНКЦИЯ 202(2)
ФУНКЦИЯ 202(n)
ФУНКЦИЯ 202(n+1)
ФУНКЦИЯ 202(n+2)
ФУНКЦИЯ 202(n+m)
206(1)
206(2)

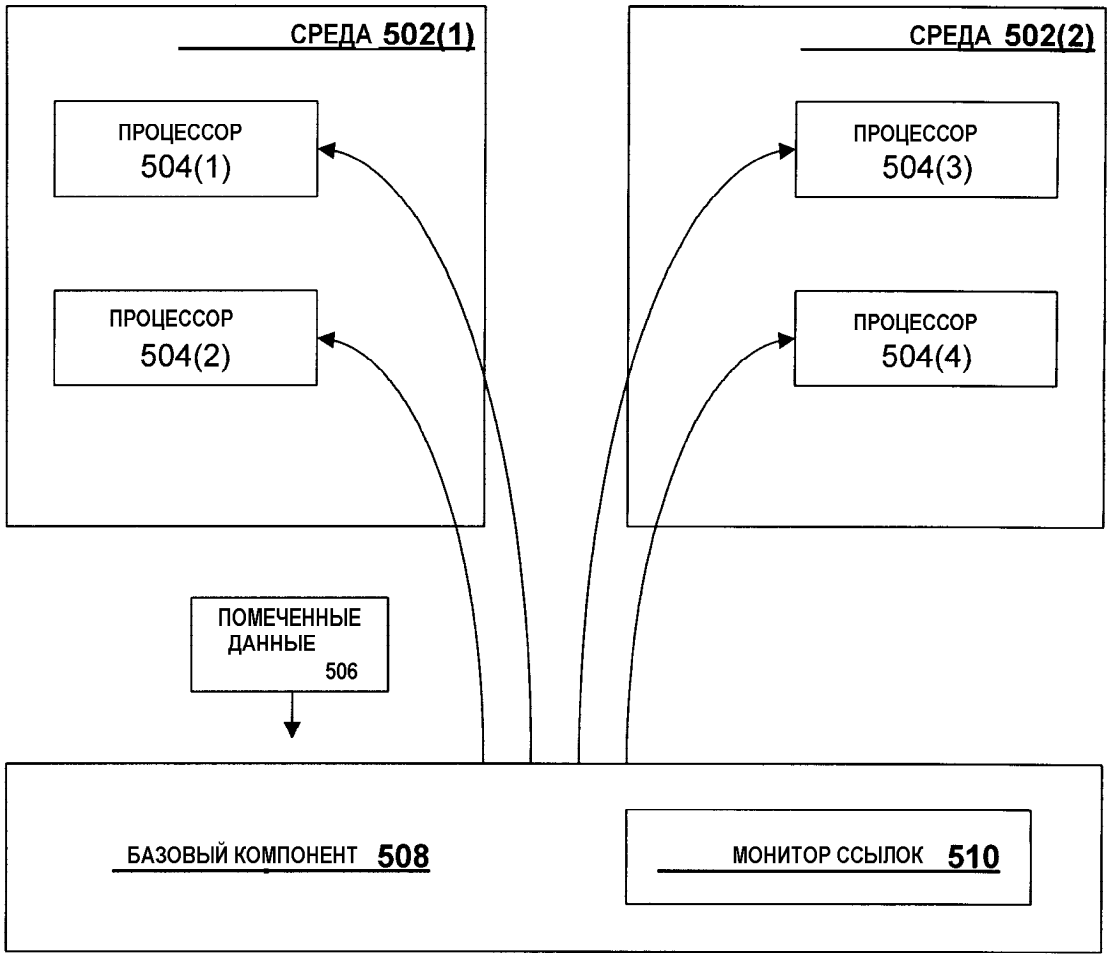
ФИГ.2


ПРИЛОЖЕНИЕ 135
206(1)
206(2)
304
ОПЕРАЦИИ, КОТОРЫЕ НУЖДАЮТСЯ В ЗАЩИТЕ НИЗКОЙ СТЕПЕНИ ИЛИ НЕ НУЖДАЮТСЯ В ЗАЩИТЕ
302
ОПЕРАЦИИ, КОТОРЫЕ НУЖДАЮТСЯ В ВЫСОКОЙ СТЕПЕНИ ЗАЩИТЫ

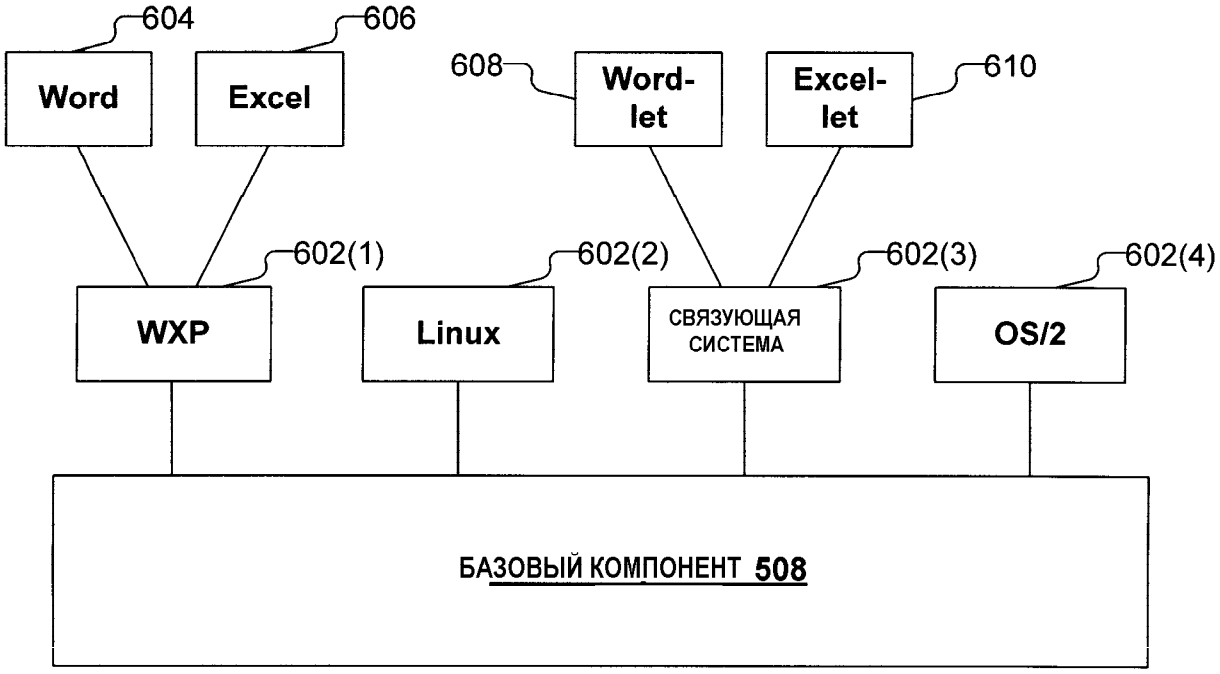
ФИГ.3


402(1)
400
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick
304
402(2)
402(3)
405
405
402(4)
405
402(5)

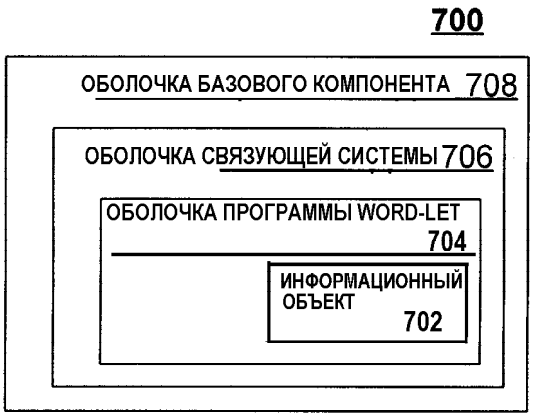
ФИГ.4


СРЕДА 502(1)
ПРОЦЕССОР 504(1)
ПРОЦЕССОР 504(2)
СРЕДА 502(2)
ПРОЦЕССОР 504(3)
ПРОЦЕССОР 504(4)
ПОМЕЧЕННЫЕ ДАННЫЕ 506
БАЗОВЫЙ КОМПОНЕНТ 508
МОНИТОР ССЫЛОК 510

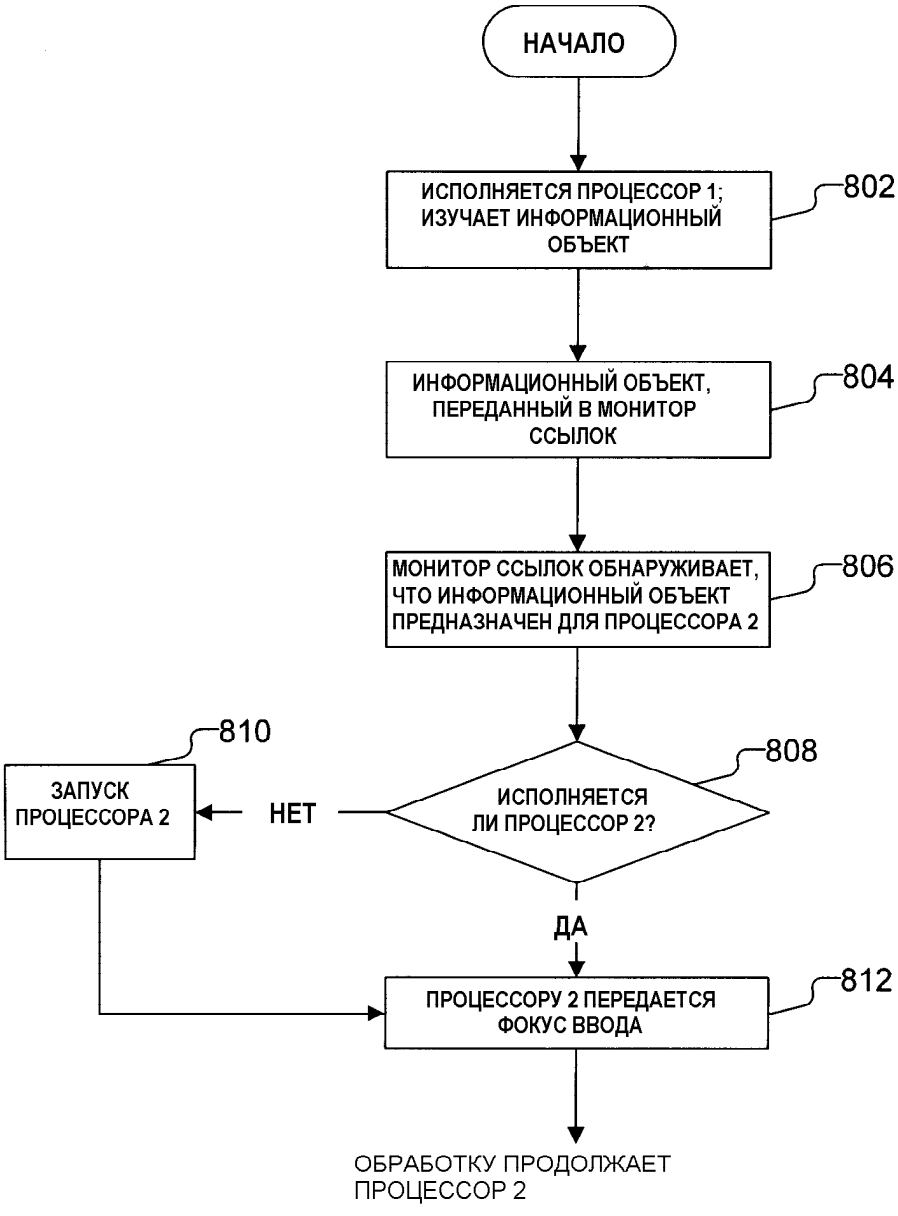
ФИГ.5


604
606
608
610
Word
Excel
Word-let
Excel-let
602(1)
602(2)
602(3)
602(4)
WXP
Linux
СВЯЗУЮЩАЯ СИСТЕМА
OS/2
БАЗОВЫЙ КОМПОНЕНТ 508

ФИГ.6


700
ОБОЛОЧКА БАЗОВОГО КОМПОНЕНТА 708
ОБОЛОЧКА СВЯЗУЮЩЕЙ СИСТЕМЫ 706
ОБОЛОЧКА ПРОГРАММЫ WORD-LET 704
ИНФОРМАЦИОННЫЙ ОБЪЕКТ 702

ФИГ.7


НАЧАЛО
ИСПОЛНЯЕТСЯ ПРОЦЕССОР 1; ИЗУЧАЕТ ИНФОРМАЦИОННЫЙ ОБЪЕКТ
802
ИНФОРМАЦИОННЫЙ ОБЪЕКТ, ПЕРЕДАННЫЙ В МОНИТОР ССЫЛОК
804
МОНИТОР ССЫЛОК ОБНАРУЖИВАЕТ, ЧТО ИНФОРМАЦИОННЫЙ ОБЪЕКТ ПРЕДНАЗНАЧЕН ДЛЯ ПРОЦЕССОРА 2
806
ИСПОЛНЯЕТСЯ ЛИ ПРОЦЕССОР 2?
808
НЕТ
ЗАПУСК ПРОЦЕССОРА 2
810
ДА
ПРОЦЕССОРУ 2 ПЕРЕДАЕТСЯ ФОКУС ВВОДА
812
ОБРАБОТКУ ПРОДОЛЖАЕТ ПРОЦЕССОР 2

ФИГ.8