DOI 10.36622/VSTU.2022.25.3.001
УДК 004.8

РАСШИРЕННАЯ МОДЕЛЬ ОТКРЫТЫХ СИСТЕМ (ЧАСТЬ 2)

К.А. Бугайский, И.С. Перескоков, Александр О. Петров, Андрей О. Петров

Во второй части статьи выполнена разработка базовой плоскости расширенной модели открытых систем, которая содержит все компоненты модели. Дано определение слоев базовой плоскости в привязке к физическим носителям информационных единиц, а также дано определение основных видов программных сущностей модели. Рассмотрено взаимодействие программных сущностей и информационных единиц через реализацию функций отображения, в основе которых лежит концепция «поставщик-потребитель». Показано, что это взаимодействие представляет собой рекуррентное отношение механизмов коммуникаций, возникающее в силу формирования путей доступа программных сущностей к информационным единицам. Для учета современных тенденций построения вычислительных систем в модели введено понятие монитора обращений и показано его место в процессе взаимодействия программных сущностей и информационных единиц.

Ключевые слова: открытые системы, информационная безопасность, модель, информационная система.

Введение

В первой части статьи [1] была предложена расширенная модель открытых систем (далее – РМОС, модель), к основным компонентам которой относятся:

AE – программные сущности (далее – ПС),

IC – механизмы коммуникаций (далее – МК),

IE – информационные единицы (далее – ИЕ).

В данной статье рассматривается базовая плоскость как неотъемлемая часть РМОС, определяющая основные механизмы взаимодействия ПС с ИЕ и ПС между собой, что, в частности, необходимо при дальнейшем рассмотрении плоскости администрирования и плоскости защиты.

Взаимодействие ПС и ИЕ

Как было показано в первой части статьи [1] ИЕ являются пассивными элементами модели. Но вместе с тем обработка ИЕ программными сущностями составляет основу функционирования любой вычислительной системы (далее – ВС). Можно считать общепринятой точку зрения, когда отношения ПС и ИЕ представляются в виде двудольного графа (рис. 1).

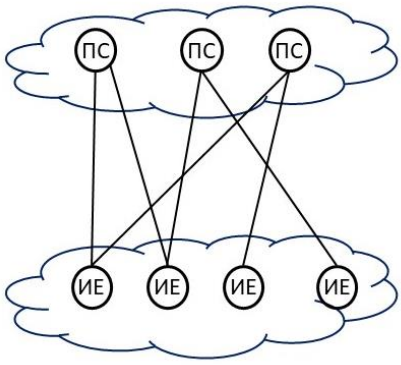

Рис. 1. Граф отношений ПС и ИЕ

Для любых операций с ИЕ данная информационная единица должна быть поименована на логическом уровне [2, 3], причем имя $name$ должно быть уникально в пределах ВС. Кроме того, каждая ИЕ должна иметь уникальный адрес adr на физическом носителе. Без потери общности случай, когда одной логически поименованной ИЕ соответствует несколько физических адресов (фрагментов на носителе), рассматривать не будем. Обозначим ПС как $a \in AE$, а ИЕ как $e \in IE$. В рамках модели отношения между ПС и ИЕ с учетом логической и физической адресации можно рассматривать как коммутативную диаграмму (рис. 2).

Морфизм $f^d: a \to e$ представляет собой «классическое» отношение ПС и ИЕ, представленное на рис. 1.

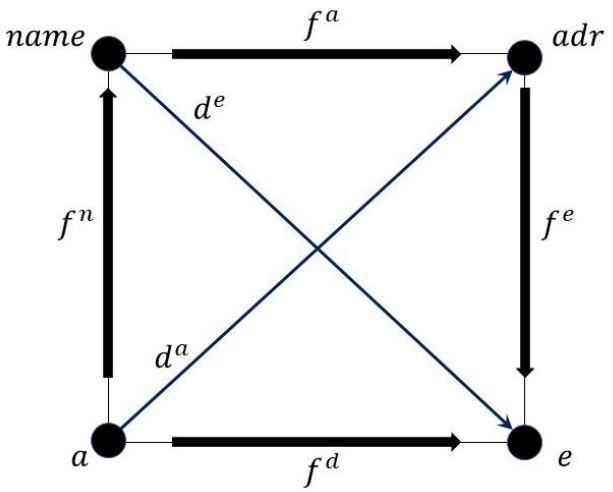

Рис. 2. Коммутативная диаграмма

Морфизм $f^e: adr \to e$ можно рассматривать как “firmware” – внутреннюю программную составляющую физического устройства – носителя ИЕ. Аналогично, морфизм $f^n: a \to name$ представляет собой внутреннюю реализацию в ПС обращения к ИЕ. Наконец, морфизм $f^a: name \to adr$ можно трактовать как функцию отображения логической адресации ИЕ на физическую. Приведенные трактовки позволяют в рамках РМОС сделать ряд допущений.

Д1. Для разных типов физических носителей необходимы разные морфизмы f^a и f^e. Описание уровня HW в стандартной МОС представляется достаточным с функциональной и структурной точек зрения и может рассматриваться как набор следующих компонент:

CPB – процессор и системная шина;

RAM – энергозависимая память (ОЗУ);

DD – энергонезависимая память (диски);

DIO – устройства ввода-вывода;

NIC - устройства сетевого обмена данными.

Д2. Морфизмы f^* в зависимости от используемого набора из общего состава компонент $K = \{\text{CPB}, \text{RAM}, \text{DD}, \text{DIO}, \text{NIC}\}$ так или иначе программно реализованы во всех ПС данной ВС, что необходимо для обращения к соответствующим ИЕ в процессе их обработки.

Д3. Корректность программной реализации морфизмов f^* является ключевым условием обеспечения работоспособности ВС, что позволяет не рассматривать вопросы их конкретной реализации в РМОС.

Выделим на диаграмме рис. 2 функции отображения $d^e = f^e \circ f^a$ и $d^a = f^a \circ f^n$. Первая из них – d^e – обеспечивает взаимно-однозначное соответствие между ИЕ и его логическим именованием в рамках ВС. Вторая – d^a – может рассматриваться как непосредственное обращение ПС к ИЕ. Поскольку любая операция в ВС может рассматриваться с точки зрения ее программной реализации, определим в рамках модели специальные ПС, соответствующие этим функциям отображения:

- диспетчер, реализующий функцию отображения d^e;
- драйвер, реализующий функцию отображения d^a.

В соответствии с диаграммой на рис. 2 целесообразно возложить в РМОС подобные функции на диспетчеров компонент K ВС. Таким образом, введение в модель функций отображения d^e и d^a и, соответственно, диспетчеров и драйверов, дает основание для рассмотрения в дальнейшем таких свойств как конфиденциальность, целостность и доступность ИЕ в модели.

Д4. Пользуясь тем, что в стандартной МОС определение слоев HW, OW, MW и AW имеют выраженный функциональный характер, будем обозначать этими буквами отдельные типы ПС. Обозначим драйверы как hw^k, диспетчеры – как ow^k, где $k \in K$.

Тогда диаграммы на рис. 1 и 2 можно представить в обобщенном виде как базовую схему взаимодействия ПС и ИЕ, представленную на рис. 3.

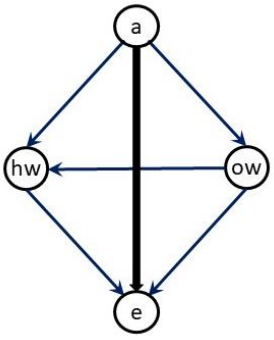

Рис. 3. Базовая схема взаимодействия

С целью учета архитектурных решений современных ВС [4] целесообразно представить взаимодействие ПС и ИЕ в обобщенном виде как показано на рис. 4, где ПС промежуточного (по месту на схеме) уровня обозначены как mw, а «терминальные» ПС – как aw. Стрелками на рис. 4 обозначено направление «движения» ИЕ к тем или иным ПС.

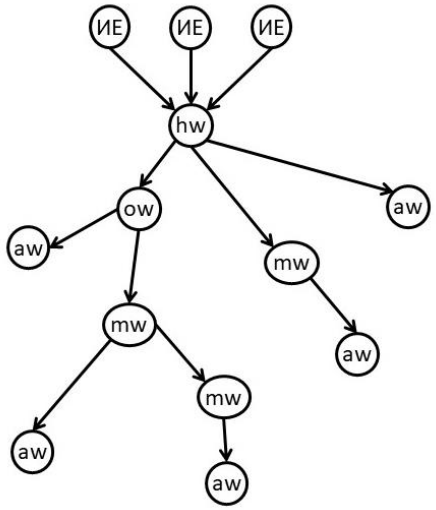

Рис. 4. Взаимодействие ПС и ИЕ в ВС

Д5. Поставим в соответствие каждому из морфизмов f^* (и набору компонент K) некоторый тип операции, реализуемый программно в ПС. Предлагается использование следующий перечень (обязательных, но не исчерпывающих) типов операций, связанных с получением ИЕ для обработки и сохранения результатов обработки:

а) op^{mem} – операции с имеющимися в распоряжении программы структурами памяти (компонента ВС K^{RAM});

б) op^{proc} – операции по созданию, запуску, останову или удалению потоков или процессов (компонента ВС K^{CPB});

в) op^{file} – операции по созданию, удалению, записи, чтения или модификации файлов (компонента ВС K^{DD});

г) op^{net} – операции сетевого обмена данными (компонента ВС K^{NIC});

д) op^{io} – операции обмена данными через устройства ввода-вывода (компонента ВС K^{DIO}).

Данный перечень операций может быть расширен, например за счет выделения в качестве отдельного физического носителя информации графической подсистемы в составе видеокарты и монитора.

Взаимодействие ПС

Как показано на схеме взаимодействия (рис. 4), все перечисленные типы операций по сути являются вызовами преимущественно соответствующих диспетчеров ВС. И именно в таком качестве и будут рассматриваться далее. Вместе с тем, возможно выделение в отдельный тип операций прямого обращения к драйверам. Что можно рассматривать как норму для встраиваемых средств защиты информации и устройств интернета вещей. Приведенный выше перечень операций целесообразно рассматривать как множество

$$OPS = \{op^{mem}, op^{proc}, op^{file}, op^{net}, op^{io}\},$$

которое может использоваться для описания взаимосвязей ПС в РМОС с точки зрения их доступа к ИЕ. Напомним, что диспетчеры и драйверы, реализующие эти операции непосредственно в отношении ИЕ обозначены hw^m и ow^m, $m \in OPS$. В рамках модели положим, что:

- диспетчеры и драйверы hw^m и ow^m для каждой из операций изолированы и взаимодействуют между собой, а также с другими драйверами и диспетчерами только через выделенные сущности (OE*);
- диспетчеры и драйверы hw^m и ow^m взаимодействуют с ПС mw и aw – независимо от реализации любой из операций OPS через системные вызовы (S*).

Такой подход позволяет выделить для каждой из операций монитор обращений M, состоящий их драйвера и диспетчера непосредственно отвечающих за выполнение той или иной операции с ИЕ. При этом положим, что рассмотрение вопроса непосредственного взаимодействия монитора обращений с ИЕ выходит за рамки РМОС. Структурная схема монитора обращений приведена на рис. 5.

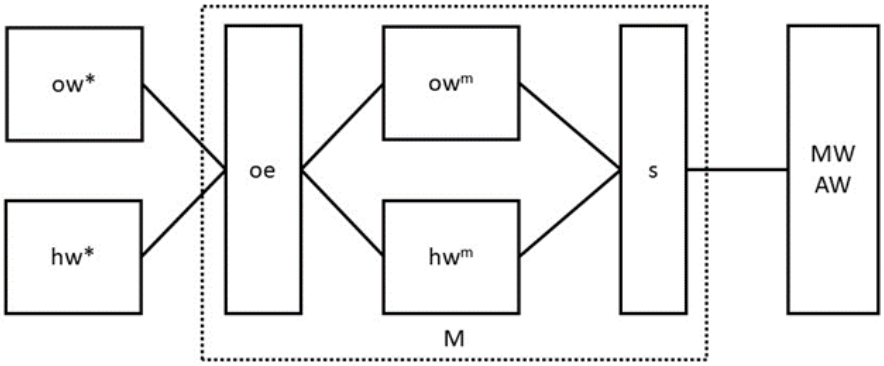

Рис. 5. Структурная схема монитора обращений

Д6. В рамках модели можно положить, что во всех ПС современных ВС реализованы операции, связанные с запросами на выделение и освобождение памяти или процессов в ходе работы данной ПС – $OP_{min} = \{op^{mem}, op^{proc}\}$. То есть, минимально необходимый набор операций, обеспечивающий работоспособность ПС. Что позволяет рассматривать набор операций OP_{min}, реализуемый в ПС hw^k и ow^k, $k \in OP_{min}$ как особый случай в РМОС. Обозначим совокупность таких драйверов и диспетчеров в РМОС как ядро:

$$kernel = \bigcup_{k \in OP_{min}} (hw^k, ow^k).$$

С учетом того, что даже в операционных системах, построенных по микроядерной архитектуре, ядро представляет собой монолитную ПС, состав и внутреннюю структуру $kernel$ в модели рассматривать не будем. Так же поступим и в отношении остальных мониторов, реализующих операции $OP^* = OPS \backslash OP_{min}$. То есть в дальнейшем в рамках модели ядро и мониторы будем считать отдельными ПС в составе ВС.

Д7. Подход за счет учета связей между ядром и мониторами позволяет моделировать разные типы архитектуры операционных систем. Например:

- структура $Kernel \to mw \to M$ соответствует микроядерной архитектуре;
- структура $Kernel \to M \to mw \to Kernel \to M$ соответствует механизмам виртуализации в ВС.

Поскольку ядро и мониторы представлены в модели в виде отдельных ПС, то целесообразно говорить о программной реализации алгоритмов, определяющих их взаимодействие через OE* и S* с ИЕ и другими ПС в составе ВС, то есть о функциях. При этом можно выделить два типа функций:

φ^t - функции типа take, обеспечивающие по результатам внутренних вычислений запрос на выполнение необходимых операций в другой ПС;

φ^g - функции типа grant, обеспечивающие инициализацию выполнения необходимых вычислений в самой ПС в ответ на запрос от другой ПС.

Каждая функция φ^* имеет набор $B = \{b_1, \dots, b_n\}$ параметров вызова и возвращаемых значений и может быть представлена как $\varphi^t(B^t)$ и $\varphi^g(B^g)$. Необходимо подчеркнуть, что аргументы функции с полным основанием могут быть отнесены к параметрам конфигурации ПС [1], то есть $C = \{B^t, B^g\}$.

На примере мониторов и ядра, а также схемы взаимодействия (рис. 4) можно утверждать, что каждая ПС в рамках РМОС $a \in AE$ описывается двумя множествами функций:

$\Phi_a^t = \{\varphi_1^t, \dots, \varphi_n^t\}$ – множество программных функций ПС для выполнения отдельных операций, реализованных в другой ПС (take-функции);

$\Phi_a^g = \{\varphi_1^g, \dots, \varphi_n^g\}$ – множество программных функций ПС для выполнения отдельных операций в ответ на запрос от другой ПС (grant-функции).

В дальнейшем, для сокращения числа индексов будем при необходимости записывать $\varphi_a^*(B^*) \in \Phi_a^*$ как $\varphi^*(a_i)$, $C(a_i)$ или $\Phi^*(a_i)$, $i \in [1, \dots, |AE|]$ соответственно. ПС в рамках модели может быть представлена кортежем: $a = \{\Phi^t, \Phi^g, C\}$.

Заметим, что из введенных выше допущений следует отношение: $\varphi^g > \varphi^t$. Действительно, рассмотрим взаимодействие двух ПС – А и Б. Пусть А имеет в своем составе take-функцию, способную обмениваться данными с grant-функцией Б. Связь между ПС возникает тогда, когда А обращается к Б – посылает соответствующий запрос. В ответ на запрос ПС Б должна выполнить некоторый набор действий, возможно связанный с обращениями к другим ПС. Результат выполненных действий Б возвращает в виде ответа на запрос А. Таким образом, ПС Б в некотором смысле на момент взаимодействия «важнее» ПС А, поскольку ПС А зависит от результатов деятельности ПС Б. Следствием является возможность рассмотрения взаимодействий ПС А и Б как направленной от grant- к take-функции связи между ПС.

Приведенные рассуждения позволяют сделать следующие предположения относительно структуры базовой плоскости РМОС (рис. 6).

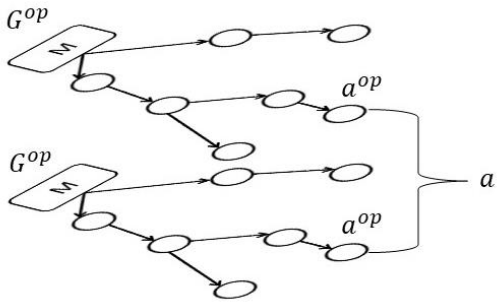

Рис. 6. Структура базовой плоскости РМОС

То есть, базовую плоскость РМОС необходимо рассматривать как набор слоев, каждый из которых представлен графом G^{op}, $op \in OPS$. Каждый такой граф является деревом, имеющим в качестве корня монитор обращений M, а в качестве вершин a^{op} необходимо рассматривать отображение одной и той же ПС $a \in AE$ при реализации ею тех или иных операций по доступу к ИЕ, что соответствует take- и grant-функциям φ^* ПС.

Такой подход дает возможность рассматривать в дальнейшем вопросы обеспечения целостности, доступности и конфиденциальности при работе ПС применительно к различным физическим носителям ИЕ. Поскольку одна и таже единица информации может одновременно находиться на разных физических носителях. В качестве подтверждения приведем пример с редактированием файла данных. В процессе редактирования одна и таже информация отображается на экране монитора (DIO), находится в памяти (RAM) и на диске (DD) ВС.

В рамках РМОС целесообразно положить, что взаимодействие ПС не ограничивается операциями по доступу к ИЕ. Как минимум, необходимо рассматривать непосредственный обмен данными между двумя ПС. Кроме того, в силу многопользовательской структуры современных ВС необходимо учитывать операции, связанные с формированием пространства (и прав) пользователей – userspace. Отсюда вытекает необходимость в дополнительных операциях, связанных непосредственно с взаимодействием ПС в процессе работы ВС. Предлагается следующий перечень (обязательных, но не исчерпывающих) типов операций на уровне программной реализации:

е) op^{api} – операции обмена информацией с другими ПС в процессе ее обработки на основе реализованных в ПС программных интерфейсах и не относящиеся к предыдущим операциям а) – д);

ж) op^{aaa} – вычислительные операции, связанные с обработкой внутри ПС информации по идентификации, аутентификации и авторизации пользователей ВС.

Следуя логике построения РМОС необходимо рассматривать ПС, выполняющие данные операции и их связи на основе функций φ^* как вершины и ребра соответствующих графов G^{api} и G^{aaa}, которые вводят в состав базовой плоскости РМОС еще два слоя. В силу архитектурных особенностей современных ВС, графы G^{api} и G^{aaa} будут представлять собой набор компонент связности, которые могут трактоваться как модули, подсистемы и другие подобные архитектурные компоненты ВС. В рамках модели такие компоненты связности могут рассматриваться как уровни абстракции. На уровне описания базовой плоскости РМОС целесообразно выделить следующие уровни абстракции как

обобщенные характеристики ПС $a^n \in AE$, $n \in OPA$, $OPA = \{op^{api}, op^{aaa}\}$:

- $Lib = \{a^n | \Phi^t(a^n) = \emptyset, \Phi^g(a^n) \neq \emptyset\}$ – библиотеки (ПС, не имеющие take-функций);
- $App = \{a^n | \Phi^t(a^n) \neq \emptyset, \Phi^g(a^n) = \emptyset\}$ – приложения (ПС, не имеющие grant-функций);
- $Gap = \{a_i^n, a_j^n | \exists(\varphi^t(a_i^n), \varphi^g(a_j^n)) \wedge \exists(\varphi^g(a_j^n), \varphi^t(a_i^n))\}$, $i \neq j$, $i, j \in [1, \dots, |AE|]$ – точки доступа (ПС, имеющая одновременно take- и grant-функции с другой ПС);
- $Prg = \{a^n | a^n \notin Lib \vee a^n \notin App \vee a^n \notin Gap\}$ – программы (это все остальные ПС).

Отметим, что данные уровни абстракции позволяют описать ядро и модули обращения РМОС, что может стать предметом дальнейших исследований.

Таким образом, собственно базовая плоскость РМОС может быть представлена как набор слоев, каждый из которых является графом G^{op}, $op \in OP$, $OP = OPS \cup OPA$. При этом одна и таже ПС $a \in AE$ может иметь отображение на вершины любого их этих графов.

Завершая описание взаимодействия ПС, отметим, что схема взаимодействия (рис. 4) и предположения П1-П5 позволяют говорить о наличии для слоев OPS определенной иерархии среди ПС. Которая определяется тем, насколько далеко ПС отстоит от ядра и мониторов обращений. С точки зрения информационной безопасности при доступе конкретной ПС к определенной ИЕ наличие промежуточных ПС можно рассматривать как потенциальный источник угроз. То есть иерархия ПС представляет собой иерархию уровней доверия. При этом одна и таже ПС может относиться к различным уровням доверия в различных слоях базовой плоскости РМОС. Целесообразно на основе введенных выше допущений Д6-Д7 в качестве нулевого (высшего) уровня доверия рассматривать именно ядро $Kernel$ РМОС.

Рассмотрение перечисленных аспектов модели будет сделано в других статьях, посвященных плоскости администрирования и плоскости защиты РМОС.

Механизмы коммуникаций

Из приведенного в предыдущем разделе описания базовой плоскости РМОС следует, что взаимодействие ПС и ИЕ может быть представлено как перемещение по графу операций. То есть взаимодействие состоит из нескольких шагов и при выполнении любого типа операции на любом шаге во взаимодействии участвуют только две ПС. Пусть у нас множество ПС AE упорядочено произвольным образом. Как было показано ранее, взаимодействие между ПС имеет направление, определяемое типом функций – take или grant. Обозначим связь, возникающую при взаимодействии двух ПС как $l(i,j)\colon \varphi^g(a_i) \rightarrow \varphi^t(a_j)$, $a \in AE$, $i \neq j$, $i, j \in [1, \dots, |AE|]$. Положим, без потери общности, что ИЕ может быть представлен как ПС типа драйвер hw, а доступ к ИЕ осуществляет ПС типа aw. В этом случае взаимодействие ПС и ИЕ это путь определенной длины, который может быть представлен как сумма (конкатенация) связей $p = l_1(hw, ow) + l_2(ow, mw) + l_3(mw, aw)$. Иначе говоря, взаимодействие ПС a и ИЕ e можно представить как $R\colon aw \xrightarrow{p} hw$ и рассмотреть это как отношение двух ПС. Данное отношение является:

- *антирефлексивным* в силу того, что вызов ПС самой себя равносильно выполнению внутренних операций, что в случае их безошибочной реализации не влечет за собой необходимости для ПС вступать во взаимодействие с другими ПС;
- *асимметричным*, в силу направленности связей между различными ПС на пути доступа конкретной ПС к ИЕ;
- *транзитивным*, что вытекает из необходимости обеспечения работоспособности ПС, составляющих ВС.

Если рассматривать путь p как функцию отображения ИЕ в ПС, то можно утверждать, что она монотонна. Поскольку, с учетом свойств отношения R, если в процессе взаимодействия ПС на пути произойдет обратный вызов (например, $l_2(mw, ow)$), то образуется цикл (бесконечный в общем случае). Или иначе – нарушение работоспособности вычислительной системы при отсутствии внешнего деструктивного воздействия.

Также, исходя из принципа обеспечения работоспособности ВС, можно положить, что:

- при фиксированных ПС и ИЕ путь p также фиксирован (неизменен) во времени, то есть при каждом обращении ПС к ИЕ будет использоваться один и тот же путь;

- каждая связь $l(i,j)$ из состава пути может быть представлена как функция взаимодействия двух ПС;

- выполнение функций взаимодействия (реализация связей $l(i,j)$ пути p) происходит поочередно во времени.

Взаимодействие ПС в составе ВС [2-6] определяется совместно используемыми:

- коммуникационными каналами (сокет, именованный и неименованный канал, механизмы типа IPC и RPC и т. п.);

- структурами данных (файл, общая (разделяемая) память, стек и т. п.);

- соглашениями – форматами и протоколами обмена.

Совместно используемые для взаимодействия каналы, структуры и соглашения определим как механизм коммуникаций ПС (МК). В рамках РМОС будем говорить о программной реализации различных МК в ПС. Опираясь на [2, 3, 5, 6], можно положить, что в обобщенном виде в основе МК лежит особый тип ИЕ – буфер (buffer), который представляет собой определенную структуру непосредственно доступную (являющуюся частью) программной реализации МК в ПС. При этом буфер зависит от типа операции – $buffer^{op}$ и всегда находится под управлением ядра ($Kernel$). На рис. 7 приведена структура взаимодействия двух ПС (ПС a^{op} с монитором обращений M^{op}) с длиной пути $p = 1$, которая имеет минимальную сложность для конкретного слоя операций G^{op} (рис. 6).

Тогда МК может быть представлен как:

$$t = \{\varphi^*(a_i), \varphi^*(a_j), \varphi^*(Kernel), buffer, C, \Theta\},$$

где: C – конфигурация МК, определяющая форматы и протоколы обмена, а Θ – множество связей, возникающих при взаимодействии ПС.

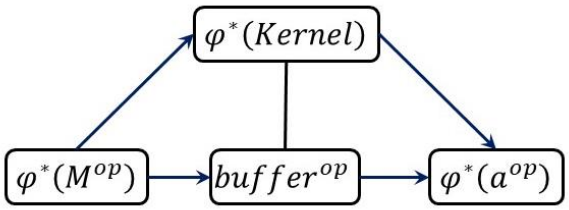

Рис. 7. Структура взаимодействия ПС

В примере на рис. 7 это будут связи: $\theta(a, Kernel)$, $\theta(a, buffer)$, $\theta(M, Kernel)$, $\theta(Kernel, buffer)$, а также $\theta(M, buffer)$.

С учетом широкого распространения механизмов виртуализации и, соответственно, возрастающей роли коммуникационных каналов при взаимодействии ПС в современных ВС, обобщим приведенную на рис. 7 структуру взаимодействия. Обобщенная структура взаимодействия описывает взаимодействие двух ПС через общую ИЕ, такую как сетевой сокет или служебный файл именованного канала.

В этом случае, как показано на рис. 8, раздельные буферы создаются ядром и соответствующим монитором обращений для каждой из взаимодействующих ПС. Синхронизация обмена данными между буферами также выполняются ядром и монитором обращений. Для обобщенной структуры взаимодействия МК может быть представлен как:

$$t = \{\varphi^*(a_i), \varphi^*(a_j), \varphi^*(Kernel), buffer_i, buffer_i, C_i, C_j, \Theta_i, \Theta_j\}.$$

Пример рис. 8 показывает, что в определенных случаях взаимодействие ПС целесообразно представлять в виде двух информационных потоков:

- поток данных, состоящий из связей $\theta(a, buffer)$, $\theta(M, buffer)$ – для каждой из ПС;

- поток управления, состоящий из связей $\theta(a, Kernel)$ – для каждой из ПС, $\theta(M, Kernel)$ – общей для обеих ПС.

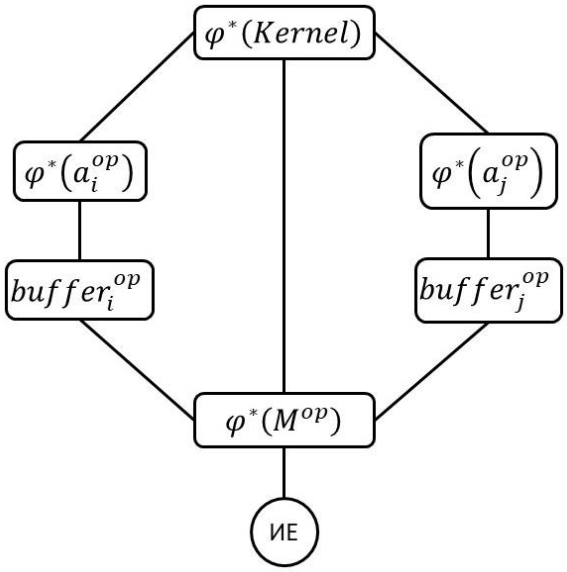

Рис. 8. Обобщенная структура взаимодействия

Отметим, что конфигурации C (как множества параметров конфигурации) для каждой из ПС могут различаться, но должны иметь общие (равные) элементы $C_i \cap C_j \neq \emptyset$.

Примеры на рис. 7 и 8 показывают, что МК должен быть соотнесен с типом операции или с типом физического носителя ИЕ – t^{op}. В общем виде в рамках РМОС взаимодействие ПС на одном шаге может быть описано как $l(a_i^{op}, a_j^{op}) = \{a_i^{op}, t^{op}, a_j^{op}\}$. Как показано на рис. 4 длина пути (число шагов) от ИЕ до ПС в значительном числе случаев превышает единицу, что вызывает необходимость рассмотрения взаимодействия ПС и ИЕ при длине пути $p > 1$.

Пример подобного взаимодействия ПС (a3) и ИЕ приведен на рис. 9.

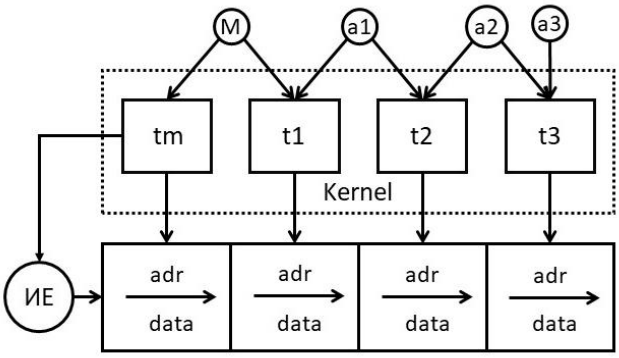

Рис. 9. Взаимодействие ПС и ИЕ при $p > 1$

Как было отмечено ранее, взаимодействие монитора обращений с ИЕ (МК типа tm), ядра и монитора обращений, а также ядра и МК (t1-t3) в данной статье не рассматривается. Если положить, что в качестве ПС a1, a2 используются программные средства виртуализации или такие фреймворки как Java, dotNet, то обращение ПС a3 к ИЕ может сопровождаться перевычислением адреса доступа к ИЕ или модификации (например, сериализации/десериализации) данных.

Отметим, что это могут быть «штатные» процедуры, обусловленные архитектурой ВС или применяемыми методами обработки данных. Но даже если положить, что МК на пути от ИЕ до ПС a3 выполняют только трансляцию информации (адрес или данные ИЕ), то необходимо сделать еще один промежуточный вывод: работа каждого последующего МК на пути зависит от качества работы предыдущего МК. То есть, путь от ИЕ до ПС в РМОС должен быть представлен как рекуррентная функция $p = \psi(t_1, \dots, t_n)$, где n – число МК (длина пути). Что позволяет, в свою очередь, рассматривать ПС как совокупность путей, обеспечивающих ее функционирование: $a = \{\Psi, C\}$, $\Psi = \{\psi^{op} | op \in OP\}$.

Представляется целесообразным вопросы трансляции и преобразования рассмотреть при описании плоскостей администрирования и защиты РМОС.

Приведенные структуры взаимодействия позволяют описывать различные виды взаимодействия ПС и ИЕ (рис. 10).

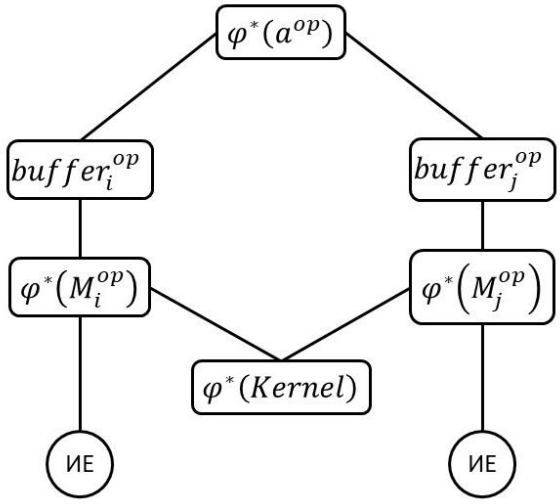

Рис. 10. Пример взаимодействия ПС и ИЕ

А также и обмен данными между различными ВС (рис. 11).

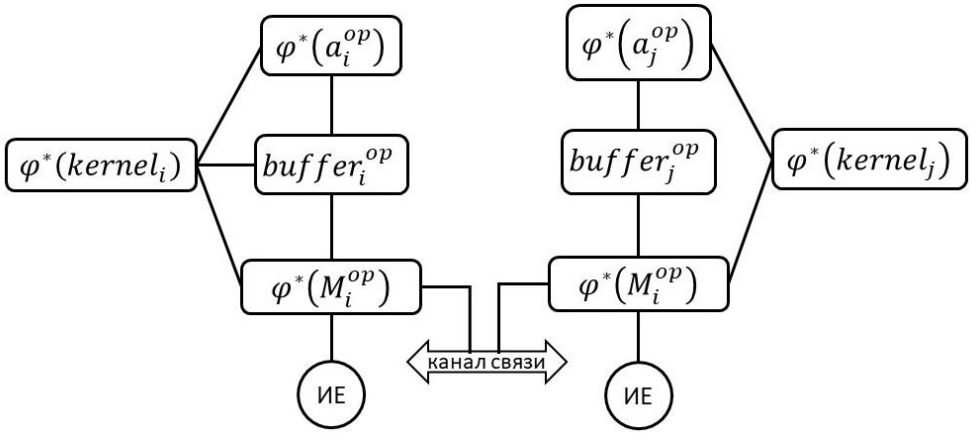

Рис. 11. Пример взаимодействия ВС

Заключение

В основу разработки элементов базовой плоскости модели положено рассмотрение отображения обрабатываемых ИЕ на ПС. Результаты разработки позволяют описать взаимодействие ПС при обработке данных, в том числе распределенной и совместной, на основе МК. Введение понятий слоев абстракции базовой плоскости, типов ПС, а также ядра и монитора обращений дает возможность учитывать различный характер данных и выполняемых операций при разработке плоскостей администрирования и защиты РМОС. Показан рекуррентный характер взаимодействия ПС. Результаты разработки элементов базовой плоскости могут быть применены для описания современных ВС.

Список литературы

1. Бугайский К.А. Расширенная модель открытых систем (Часть1) / К.А. Бугайский, Д.С. Бирин, Б.О. Дерябин, С.О. Цепенда // Информация и безопасность. 2022. Т. 25. № 2. С. 169-178.

2. Робачевский А.М. Операционные системы UNIX. / А.М. Робачевский, С.А. Немнюгин, О.Л. Стесик. СПб: BHV-Петербург, 2010. 65 с.

3. Танненбаум Э. Современные операционные системы. / Э. Танненбаум. СПб.: Питер, 2006. 1038 с.

4. Калашников А.О. Инфраструктура как код: формируется новая реальность информационной безопасности / А.О. Калашников, К.А. Бугайский // Информация и безопасность. 2019. Т. 22. № 4. С. 495-506.

5. Батаев, Л.В. Операционные системы и среды: учеб. пособ. / Л.В. Батаев, Н.Ю Налютин, А.В. Батаев. М.: Академия, 2014. 272 с.

6. Вдовикина Н. В. Операционные системы: взаимодействие процессов / Н.В. Вдовикина, И.В. Машечкин, А.Н. Терехин, А.Н. Томилин. М.: МАКС Пресс, 2008. 216 с.

Институт проблем управления им. В.А. Трапезникова РАН
V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences

Поступила в редакцию 30.06.2022

Информация об авторах

Бугайский Константин Алексеевич – младший научный сотрудник, институт проблем управления им. В.А. Трапезникова РАН, e-mail: kabuga@ipu.ru

Перескоков Илья Сергеевич – младший научный сотрудник, институт проблем управления им. В.А. Трапезникова РАН, e-mail: pereskokov@phystech.edu

Петров Александр Олегович –младший научный сотрудник, институт проблем управления им. В.А. Трапезникова РАН, e-mail petrovalexandr@ipu.ru
Петров Андрей Олегович – младший научный сотрудник, институт проблем управления им. В.А. Трапезникова РАН, e-mail: petrovaojob@gmail.com

EXTENDED MODEL OF OPEN SYSTEMS (PART 2)

K.A. Bugayskiy, I.S. Pereskokov, Aleksandr O. Petrov, Andrei O. Petrov

In the second part of the article, the development of the base plane of the extended model of open systems, which contains all the components of the model, is carried out. The definition of the layers of the base plane in relation to the physical carriers of information units is given, as well as the definition of the main types of program entities of the model is given. The interaction of software entities and information units through the implementation of display functions based on the concept of "supplier-consumer" is considered. It is shown that this interaction is a recurrent relationship of communication mechanisms that arises due to the formation of access paths of program entities to information units. To take into account modern trends in the construction of computing systems, the concept of a call monitor is introduced in the model and its place in the process of interaction of program entities and information units is shown.

Keywords: open systems, information security, model, information system.

Submitted 30.06.2022

Information about the authors
Konstantin A. Bugayskiy – junior researcher, V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: kabuga@ipu.ru
Ilia S. Pereskokov – junior researcher, V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: pereskokov @phystech.edu
Andrei O. Petrov – junior researcher, V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: petrovaojob@gmail.com
Aleksandr O. Petrov – junior researcher, V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: petrovalexandr@ipu.ru