Глава 4. ЗАЩИТА ОПЕРАЦИОННЫХ СИСТЕМ

4.1 Введение

В данной главе рассмотрены проблемы защиты программного обеспечения. Они охватывают широкий диапазон: от законодательных аспектов защиты интеллектуальной собственности до конкретных технических устройств. Последние помогают в тех ситуациях, когда очень трудно решить эти проблемы юридическими методами из-за высокой степени скрытности улики и несоизмеримости судебных издержек по сравнению с потерями от самих нарушений.

Установление авторского права на компьютерные программы изменило взгляд на защиту программных продуктов. Возникли две крайние точки зрения. Одну представляют изготовители программных продуктов, которые считают, что технические средства обеспечивают высокую степень защиты; другую – пользователи, для которых такие средства создают определенные трудности. Некоторые авторы считают, что авторское право должно быть единственным средством защиты.

Когда ряд фирм приняли авторское право в качестве единственного средства защиты программ, то оказалось коммерчески нецелесообразным продавать программы без соответствующей защиты на рынках тех стран мира, где авторское право не признается. Даже на территории стран, где соблюдается авторское право, необходима защита дешевых массовых программ; в противном случае продажа таких программ оказывается коммерчески невыгодной.

Кроме того, необходимо повысить степень скрытности защищаемой информации и методов обработки. В терминах информатики методы обработки, вероятно, должны быть реализованы в виде алгоритмов и соответствующих программных модулей, которые, хотя и являются объектом патентной защиты, если включены в состав вычислительной системы, тем не менее могут оказаться защищенными неадекватно, если содержат существенно новые элементы («ноу-хау»), которые представляют самостоятельный интерес для конкурентов.

Технические методы защиты, начиная от дешевых простых устройств и кончая мощными методами, включающими сложные способы шифрования, описаны ниже. Наша цель – рассмотреть все возможные виды компьютерного пиратства – от простых нарушений до злостных действий. Если для выявления нарушений требуются значительные усилия, это с большой вероятностью означает, что злоумышленник не является случайным лицом, однако обеспокоенность и постоянное чувство страха разоблачения делает его уязвимым по отношению к юридическим действиям, направленным на возмещение ущерба.

При чисто коммерческом отношении к оценке ущерба уменьшение числа посягательств на копирование программы, защищенной техническими средствами, должно быть сопоставлено с неприятием обычным покупателем того, что купленный товар имеет ограничения на его использование. Например, это может быть связано с тем, что пользователю запрещено создавать резервные копии программы; или один из портов компьютера должен быть выделен для подключения устройства защиты и тем самым ограничиваются его функциональные возможности; кроме того, защита может сделать программу более уязвимой и вызвать конфликт при взаимодействии с другими программами. Эти критические ситуации хорошо известны, и поэтому продолжаются поиски методов, которые могут удовлетворить таким противоречивым требованиям, как высокая степень защищенности и удобство применения. В идеале пользователь не должен «замечать» механизмов защиты, до тех пор пока он не попытается скопировать и выполнить программу на компьютере, на котором эти операции не разрешены.

Существует важное различие между методами защиты программного обеспечения и методами защиты компьютера и данных. Защита компьютера, основанная на использовании паролей и ограничении физического доступа к аппаратуре, совпадает с интересами законопослушного пользователя; в то же время, если программа успешно эксплуатируется и ее целостность не нарушена, пользователь не проявляет большой заинтересованности в защите прав автора программы.

Вычислительную установку разумно размещать на небольшой площади, и тогда упрощается контроль за ее защи-

той, а время и возможности на преодоление средств защиты оказываются весьма ограниченными. Программное обеспечение часто пересылается по почте, по каналам связи или просто продается в розницу, и это определяет способы его защиты. Программное средство, которое оказалось в руках пользователя, можно неограниченное время испытывать на преодоление механизмов защиты.

Когда говорят о превентивных мерах, обычно имеют в виду технические методы защиты, хотя и пассивные методы играют важную роль. Превентивные меры предполагают ограничения на использование программы, запрещение ее копирования или просмотра. Многие из них могут быть реализованы либо за счет создания условий защиты в самом компьютере, либо за счет включения защиты в программу, требующую, например, для своей работы разрешения от специального устройства. Однако такое устройство может оказаться экономически невыгодным из-за малого тиража его выпуска или из-за того, что его стоимость может увеличить существенно стоимость копии программы. Программа может быть сделана зависимой от характеристик конкретной вычислительной системы.

Другой важный механизм защиты – подтверждение подлинности программного кода. Он позволяет защитить от копирования, обеспечить конфиденциальность при выполнении финансовых сделок, в управленческой деятельности, при поддержании секретности, а также, например, в процессе автоматизированного проектирования, когда формальное доказательство целостности программы используется для разрешения входа в систему. Важно также обеспечить подтверждение подлинности и в тех случаях, когда в программу вносятся несущественные изменения, которые автор не в состоянии проконтролировать; это достигается тем, что такие изменения не учитываются. При этом программа должна сохранять совместимость с текущими версиями, имеющимися в продаже.

Предостережения обычно делаются с целью указать на неотвратимость возмещения ущерба по закону, но и простые словесные напоминания во многих случаях могут быть полезны. Лицензионная политика также позволяет достичь цели, но отношение к ней зависит от общественного мнения. Например, одноразовая продажа

программы с разрешением копирования может окончательно подорвать позицию конкурента. Некоторые пользователи считают, что они имеют право копировать другие программы в зависимости от того, получен или не получен автором гонорар при передаче программы в общественное пользование.

Одна из форм предостережения заключается в выводе на экран авторской этикетки (обращается внимание на юридическую ответственность). Аналогично включение в программу имени покупателя и вывод его на экран предостерегают покупателей от последующего копирования ее даже своим коллегам, поскольку имя покупателя будет присутствовать во всех копиях, которые будут сделаны и переданы другим лицам. Такие меры могут оказаться достаточными для законопослушного пользователя, но чтобы защититься от тех, кто не страдает угрызениями совести и способен выявить и уничтожить элементы защиты, в программу должны быть включены более мощные средства.

Пассивные методы могут во многих случаях помочь выявить улику, подтверждающую несанкционированное копирование, и это может сыграть свою роль в судебном разбирательстве. Однако следует определить цену такого пути, прежде чем последовать ему. Предпочтительно использовать улику, которую дают пассивные методы, для того чтобы вынудить компьютерного нарушителя (хакера), затеявшего игру, прекратить свою деятельность. Было бы достаточно, если бы нарушитель, которому предъявлены отличительные метки или другие улики в программе, прекратил дальнейшие попытки распространения и согласился возместить автору нанесенный ущерб.

Психология предостережения весьма важна при защите программного обеспечения. Хотя хакер может овладеть большим числом приемов раскрытия защиты, к услугам автора имеются разнообразные активные и пассивные методы защиты, использующие отличительные метки для ссылки на владельца авторского права или лицензию непосредственно в коде программы. Многообразие авторских приемов ставит компьютерного нарушителя в сложное положение, поскольку, хотя он и может выявить отдельные приемы, но не может быть абсолютно уверен, что все ключи и ловушки удалены из программы.

Для того чтобы обсуждение методов носило содержательный характер, эти методы классифицированы по категориям. Следует отметить, что использованные термины для обозначения категории не всегда связаны с устоявшейся терминологией.

Одно из направлений защиты – использование неожиданных или маловероятных приемов. Полезно уяснить, как можно комбинировать методы, чтобы создать прием, который из-за его неожиданности не может быть раскрыт хакером.

Выделены следующие категории средств защиты программного обеспечения:

- средства собственной защиты;
- средства защиты в составе вычислительной системы;
- средства защиты с запросом информации;
- средства активной защиты;
- средства пассивной защиты.

4.2 Средства собственной защиты

Собственная защита программ – это термин, определяющий те элементы защиты, которые присущи самому программному обеспечению или сопровождают его продажу и препятствуют незаконным действиям пользователя. Средства собственной защиты представлены на рис. 4.1.

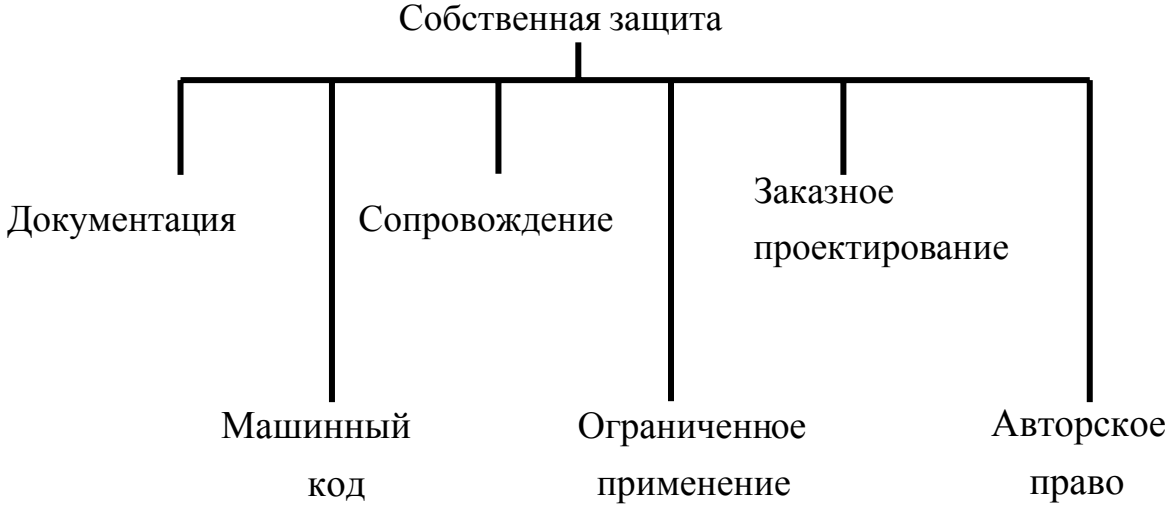

Рис. 4.1

Документация, сопровождающая любое программное обеспечение, является субъектом авторского права и может выполнять

функции защиты. Этому способствуют следующие факторы: ее репродуцирование стоит достаточно дорого, особенно если оригинал выполнен в цвете и не может быть качественно воспроизведен одноцветным копировальным устройством; обычно программы распространяются, будучи представленными в машинном коде, что затрудняет анализ их структуры и обеспечивает определенную степень защиты. В последнем случае весьма важно, чтобы сохранялось сопровождение программы со стороны разработчика, особенно в тех случаях, когда программа не полностью отлажена.

Ограниченное применение как способ защиты реализуется в том случае, когда программное обеспечение используется небольшим числом пользователей, каждый из которых известен по имени. Эта ситуация относительно легко контролируется в окружении, пользующемся доверием, хотя могут возникать проблемы с отдельными работниками, нанятыми на ограниченный срок. В этих случаях следует оговорить условия работы с программными средствами в заключаемом контракте.

Заказное проектирование предполагает разработку программного обеспечения для специальных целей. Если программа используется редко, то ее кража в коммерческих целях маловероятна; однако если кража произошла, то именно эти детали дают ключ к источнику несанкционированного копирования.

Рекомендуется также расставлять отличительные метки в стандартных программных модулях для того, чтобы идентифицировать программы, поставляемые добросовестным покупателям. Цена индивидуальной разметки каждой копии программы должна быть тщательно соразмерена с ожидаемой коммерческой прибылью.

4.3 Средства защиты в составе вычислительной системы

Эта категория средств защиты включает защиту дисков и аппаратуры, замки защиты, изменение функций штатных устройств. При использовании таких средств операционная среда вычислительной системы в отличие от штатного режима постоянно изменяется, поскольку выполнение программы зависит от определенных действий, специальных мер предосторожности и

условий, гарантирующих защиту. Перечень таких средств представлен на рис. 4.2.

4.3.1 Защита магнитных дисков

Методы защиты от копирования прежде всего были разработаны для IBM PC и подобных персональных компьютеров. Методы защиты гибких дисков используют два принципа: либо помешать копированию программы на другой диск, либо воспрепятствовать просмотру или операции обратного ассемблирования (реассемблирования).

О таких программах можно говорить как о «защищенных от копирования» и «защищенных от просмотра». Метод первого типа защищает программу от несанкционированного воспроизведения, а второго – от несанкционированной проверки.

Эти два типа защиты не взаимосвязаны: вполне возможно, что допускающая копирование программа защищена от просмотра и наоборот.

Весьма уязвимой является программа, размещенная на диске, защищенном от копирования, но допускающая просмотр. Если пользователь достаточно разобрался в программе и может обеспечить ее просмотр, то с большой долей вероятности он сможет найти способ переписать на незащищенный диск. В крайнем случае он может сделать распечатку и затем ввести программу заново. И наоборот, если программа не допускает просмотра, но находится на незащищенном диске, квалифицированный специалист может, используя стандартные средства проверки и изменения содержимого диска, обеспечить просмотр программы. По этой причине программные продукты обычно имеют оба вида защиты.

Защита дисков выполняется различными способами. Основная техника заключается в форматировании диска специальными способами, которые предохраняют операционную систему от копирования. Это – нестандартное определение форматов данных или каталогов, изменение размеров секторов, увеличение числа синхронизирующих битов и замена информационных заголовков. Поскольку время обращения к секторам различно, то программным способом можно определить время запаздывания

при чтении различных секторов, а поскольку при копировании с помощью стандартной DOS расположение секторов изменится, то запаздывания не будут более соответствовать запаздываниям исходной копии.

Рис. 4.2

Перечисленные методы становятся неэффективными при использовании систем побитового копирования. Побитовый копировщик – это электронная система копирования, которая осуществляет непосредственное считывание информации, бит за битом. Таким способом можно скопировать диск независимо от того, форматирован он или нет.

Средства защиты от побитовых копировщиков используют некоторые интересные приемы. Например, введение битов защиты, которые читаются по-разному в разное время и, таким образом, мешают верификатору копий; или запись исходной копии со скоростью ниже стандартной, что увеличивает плотность записи. В этом случае копирование на другой диск со стандартной скоростью вызывает увеличение длины записи и, следовательно, начало дорожки будет испорчено.

4.3.2 Защитные механизмы устройств вычислительной системы

Использование специальных характеристик аппаратуры для защиты программ всегда рассматривалось как весьма мощное, но дорогостоящее средство. Практический интерес представляет идея уникального диска, поскольку стоимость ее реализации достаточно низка. Принцип состоит в том, чтобы придать магнитной поверхности диска уникальность, запрещающую запись информации в некоторые секции дорожки. Сначала это достигалось механическим стиранием элементов поверхности, а позднее – использованием лазерного пучка. Таким образом, требуемый формат каждого диска оказывается уникальным. Это позволяет, сравнивая скорости чтения разных дисков, различать оригинальный диск от его копии.

В то время как специализированный компьютер экономически невыгоден для обеспечения защиты программ, специализированный микропроцессор на одном чипе оказывается пригодным для этих целей. Разработаны методы проектирования архитектуры микропроцессора, позволяющей защитить программу от считывания на шину данных для просмотра или копирования. Выполнение программы реализуется в чипе. В этом случае чип играет роль интеллектуального аппаратного модуля,

предназначенного для реализации наиболее важных процедур, требующих защиты, например, процедур шифрования и дешифрирования программ или данных, используемых исполняемой программой.

4.3.3 Замки защиты

Замки защиты используются для того, чтобы запретить доступ к программе, если при попытке обращения к ней не выполнены некоторые проверки. Например, можно осуществлять контроль предельного времени или даты использования согласно лицензии; при этом эталоном служат часы компьютера. Во многих системах установка часов доступна оператору, и это позволяет подстраивать часы под временной интервал действия лицензии. Защиту часов компьютера, т.е. предотвращение доступа к ним, можно выполнить, используя либо программный модуль, либо дополнительное оборудование в составе центрального процессора для контроля доступа к часам.

Контроль можно реализовать, если построить замок на основе уникального для каждого компьютера серийного номера. Такой замок защиты относят к классу «мобильных» (настраиваемых, уникальных) замков. В этом случае программа функционирует только на тех компьютерах, серийные номера которых включены в лицензию. Доступ к серийному номеру невозможен ни на одном компьютере, но хакеру это доставляет незначительные трудности – достаточно внести изменения в машинные коды объектного модуля, чтобы обойти или имитировать проверку.

Такая возможность обхода присуща многим методам контроля; данный недостаток можно устранить, если предусмотреть проверки в многочисленных и случайно назначаемых местах программы. Эти точки контроля должны быть проверены на подлинность перед переходом к счету, а еще лучше и во время счета, так чтобы никогда не возникали ситуации, когда вычисления идут без подтверждения проверки.

Интересный метод, который позволяет придать уникальную характеристику каждому компьютеру, состоит в записи с частичным разрушением памяти. Блоки динамической памяти в отличие от статической характеризуются тем, что данные в ней должны

периодически восстанавливаться путем регенерации. Данные исчезают, если регенерация, связанная с периодической перезаписью, по каким-либо причинам приостанавливается. Это свойство изменчивости структуры памяти можно использовать в целях идентификации. Если сигнал перезаписи прерывать на некоторое время, можно убедиться, что элементы памяти разрушаются специфическим образом для каждого конкретного модуля памяти. Условие функционирования программы можно связать с уникальной структурой памяти. Запись с частичным разрушением дает уникальный ключ защиты, который предохраняет программу от функционирования на другом компьютере.

Ключи защиты позволяют контролировать использование программного средства в течение заданных интервалов времени с последующим продолжением. Допустим, пользователь выплачивает периодическую (например, ежемесячную) арендную плату и получает на этот срок определенный пароль. Схема защиты запрещает доступ к программе, если пароль или ключ не будут соответствовать показаниям внутренних часов.

Другое применение таких ключей защиты заключается в том, что не период аренды, а ресурс программного продукта служит критерием. В этом случае необходимо определить единицу ресурса, которой может быть время функционирования программы в секундах или объем данных, извлеченных из базы данных. В этом методе ключевые слова присваиваются в соответствии с различными номерами устройств, так что пользователь может покупать блоки устройств в соответствии с работой, которая должна быть выполнена. В дальнейшем длительность использования может быть установлена заново и перезаписана в памяти.

4.3.4 Изменение функций

Существует ряд методов защиты, которые основаны на чередовании действий ключей или функций системы. Эти чередования могут предотвратить просмотр программного листинга или приостановить выполнение подпрограмм копирования. Изменения в визуальном представлении данных могут быть незаметными для пользователя, и он может не осознавать, что ин-

формация изменена или скрыта. Так, например, если изменить имена файлов, выводимых на экран монитора, то случайный пользователь не сможет вызвать такие файлы.

Функциональные особенности аппаратуры могут использоваться для защиты программ. Любая программа, размещенная в ПЗУ, будет отражать присущее этому устройству свойство, разрешающее только чтение информации. Попытки обойти это свойство программным способом будут безуспешны, если только не скопировать программу в память, допускающую запись, где незаконная копия может быть изменена.

4.4 Средства защиты с запросом информации

Включение защиты в программу связано с разработкой программ с запросом информации, т.е. требующих для своей работы ввода дополнительной информации, такой, как пароли, номера ключей и т.п.

4.4.1 Пароли

Обычные пароли не являются в полном смысле средствами защиты, они скорее относятся к механизмам управления доступом. Пароли обеспечивают сохранение целостности программного обеспечения в составе вычислительной системы, но для поддержания системы паролей требуется высокая дисциплинированность. Пароли должны быть просты для запоминания, чтобы не записывать их, и не должны быть столь очевидными, чтобы нарушитель мог угадать – они не должны быть связаны с адресом или названием фирмы. Вопросно-ответные системы, которые запрашивают место рождения, девичью фамилию и т.п., обеспечивают высокий уровень защиты, но требуют значительных ресурсов и времени работы вычислительной системы. Использование в качестве пароля отдельных элементов условного слова, например первой и пятой буквы, предотвращает ситуацию, когда целое слово могло бы быть случайно услышано. Так называемый одноразовый блокнот – более надежный механизм формирования паролей; в этом случае пароль является составным, наподобие листков блокнота, которые открываются одновременно (такой пароль

очень трудно определить). Однако защита с помощью пароля может оказаться неэффективной, если требуется хранение копии пароля, которая может быть похищена хакером.

Пароль в обычном смысле этого слова не является средством защиты программного обеспечения, поскольку законный пользователь, которому вручен пароль, может оказаться хакером. При формировании пароля можно прибегнуть к помощи специального устройства, которое генерирует последовательности чисел или букв в зависимости от данных, которые задает пользователь. Такое устройство называется преобразователем информации. В действительности вычислительная система генерирует последовательность случайных чисел и требует, чтобы пользователь в течение короткого промежутка времени присвоил ей некоторое число.

4.4.2 Сигнатуры

Сигнатура – уникальная характеристика компьютера или других устройств системы, которая может быть использована для защиты и проверена программным способом.

Уникальность гибких дисков проявляется прежде всего в форматировании. Уникальное форматирование позволяет закрепить за таким диском каталог файлов, требуемых для данной программы, чтобы установить нужную вычислительную среду. При этом копирование отдельных участков на ту же дискету гарантирует правильность, чего нельзя утверждать при копировании на другую дискету. Список испорченных секторов зависит от конкретной дискеты и отличается от списка для дубликата. К другим возможным сигнатурам относятся длина незаписанных участков магнитной ленты, неиспользованные дорожки на дискете и т.п. Техника частичного разрушения диска является примером, где сигнатура определяется уникальными характеристиками блока памяти компьютера.

В общем случае следует отыскивать такие характеристики аппаратуры или системы, которые не подвержены изменениям и сами не влияют на нормальное функционирование программного обеспечения. Если характеристики уникальны для данной вычис-

лительной системы, нормальное прохождение программы может быть выполнено только на ней.

4.4.3 Аппаратура защиты

Ожидалось, что производители компьютеров первыми проявят интерес к аппаратуре защиты и начнут встраивать ее в стандартные устройства. Но выяснилось, что никаких запросов на обеспечение компьютеров средствами защиты от пользователей не поступало. Другим потенциальным заказчиком должен быть изготовитель программных средств, особенно при продаже сложных и дорогих программных продуктов, когда защищенность прямо связана с получением дохода.

Принцип защиты программ с использованием аппаратуры защиты состоит в том, что при несанкционированном копировании программы из ПЗУ в оперативную память вырабатывается сигнал на самоуничтожение программы. Часть программного обеспечения обычно размещается в ПЗУ либо из-за недостатка памяти компьютера, либо из-за желания поставить под контроль операционной системы операцию копирования ПЗУ. Такая защита, возможно, и будет успешной от неискушенного нарушителя, но совершенно недостаточна для защиты от хакера.

Преобразователь информации, о котором мы кратко упомянули при обсуждении паролей, использует некоторые особенности преобразования данных. В одной из возможных реализаций преобразователя используется микропроцессор, генерирующий в соответствии с алгоритмом псевдослучайное число при нажатии некоторой клавиши клавиатуры. Если на вычислительной установке имеется такой же алгоритм, оператору достаточно задать правильное число, чтобы подтвердить требуемую последовательность.

Другие возможности были реализованы при использовании оптических устройств для выделения исходного образа из искаженных, которые поступают от компьютера. Это может выполнить только оператор, имеющий соответствующее оптическое устройство. Такое устройство можно построить на основе оптических материалов с двойным лучепреломлением, которые по-

зволяют получать наборы цветовых сигналов в соответствии с числами, вводимыми в устройство. Оптические системы, как правило, просты в изготовлении и имеют низкую стоимость. Достоинство описанных преобразователей информации состоит в том, что они независимы от назначения вычислительной системы и в контуре управления используют человека для оценки ответа от компьютера и задания уставок с помощью клавиатуры или другого устройства.

Электронные устройства защиты (ЭУЗ) обычно подсоединяются через стандартный интерфейс RS-232 и откликаются на запрос в виде некоторого числа или последовательности чисел. Недостаток этого устройства связан с тем, что необходимо управлять доступом к этому устройству из программы, и поэтому хакер может предусмотреть обход такого запроса. Обход может, например, заключаться в том, что включается обращение к подпрограмме, которая имитирует функцию ЭУЗ и затем возвращается к исполнению основной программы, обходя запрос. Можно также следить за линией связи и фиксировать числа, а затем генерировать таблицу для вторжения в программу. Для предотвращения таких попыток вторжения необходимо повторять запрос на доступ несколько раз и случайным образом. Кроме того, подлинность исходной программы должна подтверждаться в случайные моменты времени и предусматривать самоуничтожение программы при обнаружении обходов.

Устройства защиты с элементами интеллекта представляют собой одну из форм ЭУЗ с встроенным микропроцессором для реализации сложных алгоритмов защиты. В этом случае связь выхода ЭУЗ с входом оказывается непредсказуемой, обход устройства осуществить нельзя, если только этот обход не удалось встроить непосредственно в защищенную программу. Чтобы обойти эту защиту, надо скопировать программу из памяти микропроцессора. Действия против нарушителя – проектирование микропроцессора на кристалле со сверхвысокой степенью интеграции и встроенной памятью.

Альтернативные устройства используют генератор случайных чисел для реализации механизма подтверждения. Защищаемая программа сначала обращается к датчику случайных чисел для генерации некоторой случайной последовательно-

сти, которая и запоминается в программе; при подтверждении программа генерирует некоторую последовательность байтов, которая заставляет датчик случайных чисел воспроизвести последовательность, уникальную для данной программы.

Разработаны также устройства, которые обеспечивают защиту и подтверждение подлинности при взаимодействии двух терминалов. Такое устройство имеется на каждом терминале и предназначено для шифрования. Оно шифрует некоторое случайное число, сгенерированное на одном терминале, а использует его в диалоге с другим. На следующем шаге эта процедура осуществляется со второго терминала.

Средства непосредственной защиты основаны на уничтожении данных, если модуль, содержащий секретные данные, например ключи шифрования, подвергся взлому. Наиболее часто блокируется питание динамической памяти путем разрыва проводов, и информация исчезает. Возможные способы преодоления такой защиты засекречены. Очевидно, что эти методы широко используются в военных применениях, однако информация об этом недоступна для обычных коммерческих целей.

Программу, которую следует защитить, можно зашифровать, используя стандарт шифрования данных и зашифрованный открытый ключ. Особенность шифра открытого ключа состоит в том, что ключ к стандарту шифрования можно дешифрировать только секретным ключом, который находится в специальном аппаратном модуле, а защищенная программа может быть дешифрирована и выполнена только внутри этого модуля. Такие системы первоначально разрабатывались для перевода капиталов между банками и были реализованы в виде специализированной программы. Например, модуль защиты программного обеспечения, разработанный Национальной физической лабораторией, использует как симметричный, так и асимметричный шифры внутри модуля непосредственной защиты; это удобно при защите нескольких программ одним устройством.

Вероятно, неразумно ни с позиций функционирования, ни из экономических соображений размещать в устройстве защиты целые программы или пакеты программ. Должны быть зашифрованы лишь наиболее важные программные модули, которые будут выполняться внутри такого устройства, а вызываться

будут из незашифрованной главной программы, выполняющейся на центральном компьютере. Такое устройство может включить часы для контроля времени и даты и, поскольку оно защищено, такие часы могут фиксировать допустимую длительность работы, например, при аренде или пересылке программы пользователю для проведения вычислений в течение ограниченного времени.

Проблема, которую следует решить при выполнении нескольких программ в одном устройстве, – это отразить вторжение «троянского коня», т.е. включение в состав законной программы подпрограммы, которая оказалась бы вынужденной обратиться к данным, хранящимся в памяти микропроцессора. Новшество, предложенное в одном из патентов, состоит в том, чтобы в защищенную область памяти нельзя было обратиться, не вызвав сброса микропроцессора. В результате перезагрузки микропроцессор стартует заново и, следовательно, оказывается под управлением защищенной программы, размещенной в новой области памяти.

Следует отметить, что защита системы с открытым ключом требует сохранения регистра законных ключей с целью воспрепятствовать возможному появлению парного ключа. Если этого не предусмотрено, зашифрованное программное обеспечение может быть дешифрировано оставшимся ключом, принадлежащим нарушителю. Почему же регистр хранения открытого ключа обеспечивает необходимую защиту? Дело в том, что длина этих чисел, будучи порядка 150 десятичных цифр, позволяет сгенерировать невероятно большое число таких ключей. Число законных пар, размещаемых в регистре, существенно меньше и вероятность того, что нарушитель подберет пару, которая соответствует регистру защиты, чрезвычайно мала.

Специальная архитектура плат со сверхвысокой степенью интеграции позволяет защитить память, содержащую программу, от чтения через шину данных. Память для хранения программ – это либо программируемое (ППЗУ), либо стираемое программируемое постоянное запоминающее устройство (СППЗУ), которое размещено отдельно от памяти данных и недоступно со стороны входных и выходных портов (чтобы предотвратить прямой доступ к содержимому памяти). Плата

помещается в резервные гнезда компьютера. Считается, что преодоление такой защиты по трудоемкости аналогично технологии воспроизведения архитектуры микропроцессора с помощью шлифовки платы, как это делается при расслоении интегральной микросхемы.

Степень защиты, обеспечиваемая рядом описанных выше подсистем, поднимает интересные проблемы о потенциальных возможностях такой защиты. Секретный ключ обычно можно снова восстановить из записей и, вероятно, он будет разглашен при судебном разбирательстве. Однако можно организовать модуль так, чтобы секретный ключ порождался генератором случайных чисел в обстановке секретности и помещался в защищаемый модуль без участия человека и знания ключа. В этом случае невозможно проверить содержимое модуля. То же справедливо и по отношению к архитектуре специального микропроцессора.

4.5 Средства активной защиты

Средства активной защиты делятся на две группы: внутренние и внешние, используемые в составе компьютера и вне его соответственно. Средства защиты инициируются при возникновении особых обстоятельств – вводе неправильного пароля, указания неправильной даты или времени при запуске программы на выполнение или других подобных условий. Попытки получить доступ к точной информации без разрешения на это могут также служить инициирующим обстоятельством для приведения защиты в действие. Внутренние средства активной защиты характеризуются тем, что их обычно не рекламируют хакерам: они либо блокируют программу, либо уничтожают ее.

4.5.1 Внутренние средства активной защиты

Ключи защиты для блокирования выполнения программы могут быть настроены на любое недозволенное действие, которое будет обнаружено. Обычно это ключи, настроенные на дату, определенное время или на перечень разрешенных ресурсов; в наибольшей степени это относится к арендуемым лицен-

зионным программам, для которых период использования и требуемые ресурсы бывают однозначно определены. Проверка уровня авторских полномочий необходима, чтобы блокировать доступ к точной информации или другим ресурсам, запрещенным для использования. Реакция на несанкционированный доступ может быть реализована в виде предупреждения, дружеского напоминания либо служить поводом для организации наблюдения.

Инициализация наблюдения может начаться с регистрации в системном журнале использования терминала или реализовываться в виде подтверждения подлинности структуры программы; следует проверить, не подверглись ли средства защиты, включенные в программу, изменению или удалению.

Искажение программы представляет собой интересный прием изменения функций, хотя возможны и более решительные действия, например стирание памяти. Например, программы-вирусы вызывают постепенное разрушение программы.

4.5.2 Внешние средства активной защиты

В группе этих средств общепринятые сигналы тревоги, которые известны или неизвестны хакеру, приводят в состояние готовности средства защиты, что может быть вызвано различными ситуациями. Они могут быть активизированы при возникновении многих условий, уже описанных выше. Такие внешние факторы включают также использование ключевых слов, чтобы вызвать распечатку названия программы или имени ее владельца.

Распечатка авторской этикетки важна, поскольку большинство людей считают, что они действуют законно, и напоминание им о праве собственности владельца вызывает у них некоторую обеспокоенность. Хотя описанное и не является защитой от пиратства, это тем не менее способствует увеличению объема продаж, если число случайных копирований уменьшится. Общепринятые сигналы тревоги более сродни созданию среды защиты компьютера, когда требуется подтверждение подлинности операции, особенно при копировании.

Замечено, что чувство беспокойства может оказаться эффективным механизмом сокращения активности по крайней мере не-

которых хакеров. Законность совместного (коллективного) пользования программами должна быть подтверждена приобретением лицензии, что может служить эффективным методом борьбы с нарушителями авторского права.

Запуск распечатки этикетки или других деталей из защищенных участков программы осуществляется только при наличии ключевых слов. В то время как этикетка, появившаяся на листинге, может быть вырезана из него, защищенные данные, о которых мы еще будем говорить при обсуждении методов пассивной защиты не так просто, во-первых, найти, а во-вторых, декодировать для получения распечатки, используя подпрограмму, которая инициируется при вводе нужного ключевого слова или другой операцией (активная защита). Метод применим и в иных случаях, не обязательно связанных с анализом программного листинга.

4.6 Средства пассивной защиты

К средствам пассивной защиты относятся предостережения, контроль, а также методы, направленные на поиск улик и доказательство копирования, чтобы создать обстановку неотвратимости раскрытия.

4.6.1 Идентификация программ

Идентификация программы или отдельного модуля представляет интерес в том случае, когда другие методы защиты не приносят успех. Эти вопросы слабо освещены в литературе, за исключением обсуждения нескольких программных процедур и ряда отчетов о судебных тяжбах в США. Широко обсуждаются проблемы авторского права для отдельной процедуры программы и взаимосвязь между идеей и способом ее реализации.

Выделение объективных характеристик программы – довольно сложная процедура, тем не менее признаки подобия двух программ или модулей, содержащихся в больших программах, указать можно. Проблема заключается в том, чтобы уметь идентифицировать программы, которые изменены хакером, погружены в другую программу или откомпилированы в машинный

код. Оценка относительной частоты появления операторов или машинных команд – практический способ количественной оценки характеристики программы. Эта величина изменяется при внесении хакером изменений в программу, однако в большой программе для существенного изменения характеристики требуется выполнить значительную работу; к этому следует добавить возможность появления дополнительных ошибок или не согласующихся процедур, которые уменьшают надежность программы.

Для получения корреляционных характеристик, связанных с вставкой программного модуля в большую программу, требуются трудоемкие расчеты, хотя можно указать ряд важных признаков, которые указывали бы на целесообразность более детальных исследований.

Понятие «родимые пятна» используется для описания характеристик, появляющихся в результате естественного процесса разработки программы и относящихся к особенностям стиля программирования, ошибкам и избыточностям, которые не должны иметь места.

Каждое из них может служить убедительной уликой нарушения авторского права. Наоборот, отличительные метки относятся к таким признакам, которые не являются случайными, а вводятся специально, чтобы дать информацию об авторе или владельце авторского права. Другое использование идентификационных меток – выявление путей незаконного копирования или других злоумышленных действий. Термин «отличительная метка» относится к пассивным средствам защиты, которые при нормальном функционировании не «проявляют» себя по отношению к пользователю.

Одно из убедительных доказательств копирования – наличие скопированных ошибок. Маловероятно, чтобы в точном аналоге, который создан, как утверждается, независимо, содержались те же ошибки. В каждой программе остаются избыточные части, например подпрограммы, которые были необходимы для отладки в процессе проектирования программного продукта, а затем не были удалены. Таким образом, в любой программе содержится встроенная улика, которая тем или иным способом сохраняет следы разработки. Отсюда вытекает практический совет – сохранять документацию, которая сопровож-

дала процесс проектирования, чтобы потом иметь улику, подтверждающую авторское право.

Существует точка зрения, что убедительность улики повышается, если отличительная метка, содержащая информацию о владельце авторского права, закодирована. Известно много способов включения такой улики, особенно в программы на языках высокого уровня. Диапазон возможностей сокращается, если необходимо, чтобы отличительная метка сохранялась после компиляции в машинном коде программы. Эта проблема особенно актуальна при использовании оптимизирующих трансляторов. Использование закодированных отличительных меток – довольно распространенная практика, поскольку при этом они остаются доступными и в машинном коде. Существенно, что отличительные метки не являются в полной мере избыточными для того, кто организует контроль за данными и в состоянии отделить на их фоне действительно избыточные данные.

Очевидно, что можно разработать методы, которые позволят использовать закодированные в программе данные. Один из них связан с форматированием выходных данных в закодированной форме, что обусловлено необходимостью проверки кодирования при удаленной передаче, и это требует дополнительной работы.

Важная особенность отличительных меток заключается в том, что они неизвестны нарушителю. Поскольку в прошлом на программы покушались в основном бывшие служащие фирмы, существует организационная проблема, связанная с тем, чтобы отличительные метки были неизвестны руководству фирмы. Даже если существуют многочисленные версии программы, необходимо учитывать относительную несложность процедуры оценки корреляции двух программ, чтобы обнаружить различия в отличительных метках. Методы идентификации машинного кода с целью установления факта копирования довольно-таки надежны, но проблема с процедурами, встроенными в программу, значительно сложнее. Степень подобия процедур, которая обеспечивает правильное функционирование, представляет значительный интерес.

В то время как элементы интеллектуальной собственности, содержащиеся в программе, должны быть защищены, требование совместимости, особенно по диалоговому интерфейсу, имеет важное значение. Существует общий подход к проблеме диалогового взаимодействия (иногда определяемой как проблема «человек – машина»). Сообщество пользователей не должно изучать различные процедуры диалога при переходе к другим системам. Считается, что пользователю необходимо около трех месяцев, чтобы изучить приемы работы со сложной системой, например автоматизированного проектирования. О несогласованности процедур диалога в различных системах можно судить по тому, что более шести недель требуется для переобучения. Аналогичные проблемы возникают при использовании клавиатур с различающимся расположением клавиш.

4.6.2 Устройства контроля

Устройства регистрации событий, процедур или доступа к данным могут рассматриваться как часть общей системы защиты, причем как программ, так и данных. Подтверждение подлинности программы охватывает проблемы: от установления идентичности функционирования текущей программы и ее оригинала до подтверждения адекватности средств защиты. Сохранение выполняемой функции наиболее важно при выполнении финансовых сделок, а также для систем автоматизированного проектирования, когда целостность процедуры проектирования не должна быть нарушена. Последнее весьма важно, если используются устройства с низким уровнем защищенности, когда возможен обход проверок, связанных с защитой.

4.6.3 Водяные знаки

Использование водяных знаков как метода выявления подделки занимает особое место, поскольку препятствует созданию точной копии, которую пользователь не мог бы отличить от оригинала. В большинстве методов, предложенных для анализа проблемы идентификации программ, считается, что программы либо замаскированы, либо не полностью открыты для просмотра.

4.6.4 Психологические методы защиты

Эти методы основаны на том, чтобы создать у нарушителя чувство неуверенности и психологического напряжения, заставляя его все время помнить, что в похищенном программном продукте могут сохраняться средства защиты. Поэтому полезно было бы дать объявление, что в программное обеспечение встроены механизмы защиты (независимо от того, так ли это на самом деле). Во многих странах распространено мнение, что защита авторского права на программы усиливает психологическую обеспокоенность. Существует огромное число хитроумных способов расстановки отличительных меток в программе и никакой хакер не может быть уверен, что ему удалось уничтожить все ключи и механизмы защиты.

Методы защиты программного обеспечения имеют широкий диапазон действия, и пользователь должен выбрать тот или иной механизм, учитывая стоимость его реализации. Фактор неудобства для покупателя может несколько снизить цену.

Стратегия, выбираемая изготовителем, будет зависеть от объема программных средств, которые следует защитить. Выбранный способ должен быть относительно дешев для изготовителя, чтобы можно было его включить в большое число программных продуктов, но одновременно он должен быть сложным и дорогостоящим для преодоления нарушителем, т.е. не должен относиться к одной-единственной программе. Угроза законного возмездия против квалифицированного хакера должна быть поддержана убедительными уликами.

Идеальные методы защиты должны позволять пользователю делать резервные копии для собственного использования и не должны ограничивать возможности компьютера (такие, как число строк ввода-вывода или выбор приоритета при работе в многозадачном режиме). Исключительно важное значение приобретают устройства зашиты, которые эффективны при работе в сетях.

Психологические и социальные факты должны способствовать защите и поддерживать в сознании нарушителя обеспокоенность, и это будет полезное дополнение к методам защиты, которыми мы располагаем.

4.7 Электронные ключи

Среди средств так называемых AAA (authentication, authorization, administration – аутентификация, авторизация, администрирование) важное место занимают программно-аппаратные инструменты контроля доступа к компьютерам – электронные замки, устройства ввода идентификационных признаков (УВИП) и соответствующее программное обеспечение (ПО) [9]. В этих средствах контроля доступа к компьютерам идентификация и аутентификация, а также ряд других защитных функций, выполняются с помощью электронного замка и УВИП до загрузки ОС.

По способу считывания современные УВИП подразделяются на контактные, дистанционные и комбинированные.

Контактное считывание идентификационных признаков осуществляется непосредственным взаимодействием идентификатора и считывателя.

При бесконтактном способе считывания идентификатор может располагаться на некотором расстоянии от считывателя, а сам процесс считывания осуществляется радиочастотным или инфракрасным методом.

УВИП могут быть электронными, биометрическими и комбинированными.

Электронные УВИП содержат микросхему памяти идентификационного признака.

Анализ новых технологий защиты информации показывает, что одним из наиболее мощных инструментов защиты ПО и БД от несанкционированного использования и нелегального копирования являются системы защиты на базе электронных ключей, в частности инструментальная система **Hardlock** с ее основным компонентом – электронным ключом Hardlock.

В целом система защиты Hardlock базируется на трех компонентах:

- электронном ключе Hardlock;
- криптокарте для программирования ключей Cripto-Programmer Card;

- программном обеспечении Hardlock Bistro, позволяющем быстро создать систему защиты для приложений и связанных с ними файлов данных.

Основой ключей Hardlock является заказной ASIC-чип (Application Specific Integrated Circuit) со встроенной EEPROM-памятью, разработанной компанией Aladdin. Чип имеет достаточно сложную внутреннюю организацию и специфический алгоритм работы, причем он программируется только с использованием специальной платы Cripto-Programmer Card, так как любой другой метод программирования ключей в настоящее время не обеспечивает безопасное хранение ключевой информации.

Каждый экземпляр этой платы является уникальным и позволяет задавать 43 680 вариантов работы алгоритма шифрования, которые могут быть использованы при программировании ASIC-чипа ключа. Логику работы чипа практически невозможно реализовать с помощью наборов микросхем, его практически невозможно воспроизвести, а содержащийся в его памяти микрокод – считать, расшифровать либо сэмулировать.

Использование подобного чипа позволяет работать на всех типах ПК. Последняя модель ключа Hardlock Twin может работать как с параллельным портом, так и с последовательным, позволяя подключать через него практически любые устройства, в том числе модемы, сканеры, принтеры и т.п.

В настоящее время ключи Hardlock выпускаются в следующих конфигурациях:

- внешний ключ на параллельный порт (Hardlock EYE);
- внешний ключ на параллельный и на последовательный порт (Hardlock Twin);
- внутренний ключ на шину ISA/MCA (Hardlock Internal);
- сетевой ключ (HL-Server) – внешний или внутренний;
- USB-порт (Hardlock USB);
- PC Card (Hardlock PCMCIA) – для лэптопов и ноутбуков.

Все модели ключей совместимы между собой и могут работать с большинством ОС, в различных сетях (с протоколами IPX, TCP/IP, Net-BIOS), осуществлять защиту 16- и 32-разрядных приложений (com, exe, dll) DOS и Windows и связанных с ними файлов данных в прозрачном режиме. При записи данные авто-

матически шифруются с использованием заданного аппаратно реализованного алгоритма, при чтении – расшифровываются. Симметричное шифрование производится блоками по 64 бита, причем для каждого нового блока ASIC генерирует новый сеансовый ключ длиной 48 бит.

Главным отличием ключей Hardlock от известных является высокий уровень защиты программ и аппаратное шифрование файлов данных.

Компания ALADDIN SOFTWARE SECURITY R.D. получила сертификат Госкомиссии РФ на продукт для защиты информации под названием Secret Disk. Этот продукт позволяет создавать на диске компьютера защищенные разделы, доступ к которым невозможен без установленного в USB-порт компьютера индивидуального электронного брелока. Перед началом работы секретный диск необходимо отпереть, вставив в порт электронный брелок, и ввести пароль, а по окончании – отключить. Если во время работы с конфиденциальными документами необходимо прерваться, не выключая компьютера, достаточно вынуть ключ. При этом экран гаснет, а клавиатура блокируется. Допускается применение ключа также при работе с документами, составляющими государственную тайну.

Дальнейшим шагом в развитии Secret Disk стал продукт Secret Disk Server, предназначенный для шифрования корпоративной информации на серверах Windows NT/2000.

Устройства ввода идентификационных признаков на базе идентификаторов **Proximity** (от английского слова proximity – близость, соседство) относится к классу электронных бесконтактных радиочастотных устройств. Они выпускаются в виде карточек, ключей, брелоков и т.п. Каждый из них имеет собственный уникальный серийный номер. Основными составляющими устройств являются интегральная микросхема для связи со считывателем и встроенная антена. В составе микросхемы находятся приемо-передатчик и запоминающее устройство, хранящее идентификационный код и другие данные. Внутри Proximity может быть встроена литиевая батарейка (активные идентификаторы). Активные идентификаторы могут считывать информацию на расстоянии нескольких метров. Расстояние считывания пассив-

ными идентификаторами (не имеющих батарейки) составляет десятки сантиметров.

Устройство считывания постоянно излучает радиосигнал, который принимается антенной и передается на микросхему. За счет принятой энергии идентификатор излучает идентификационные данные, принимаемые считывателем.

4.8 Технология защиты информации на основе смарт-карт

Появление новой информационной технологии смарт-карт (СК), основанной на картах со встроенным микропроцессором, позволило удобнее решать вопросы использования пластиковых денег. Однако уникальные возможности СК с микропроцессором, состоящие в высокой степени защиты от подделки, поддержке базовых операций по обработке информации, обеспечении высоких эксплуатационных характеристик, сделали СК одним из лидеров среди носителей конфиденциальной информации. Следует отметить отличительные особенности таких карт. СК содержит микропроцессор и ОС, которые обеспечивают уникальные свойства защиты, имеют контактное и бесконтактное исполнение (на рис. 4.3 показана схема бесконтактной СК). СК могут быть произведены только промышленным путем и, следовательно, не могут быть скопированы. Каждая СК имеет уникальный код, определенный на производстве, и если на другую карту будут записаны те же данные, что и на оригинале, то различия во внутренних параметрах дают возможность системе отличить одну карту от другой. СК может быть запрограммирована так, что она выходит из строя при попытке НСД. Данные шифруются с помощью различных алгоритмов, в том числе ГОСТ 28147 – 89 или DES и секретных ключей, которые содержатся на микросхеме карты. Если карта обнаруживает несоответствие введенного пользователем pin-кода – персонального идентификационного номера – со своим личным pin-кодом несколько раз подряд (обычно 3 раза), она может самоуничтожиться, т.е. стать непригодной для использования, записав в память вместо системных ключей случайно сгенерированные числа.

Таким образом, технология СК обеспечивает надежное хранение ключей и доступ к различным информационным ресурсам.

Персональные идентификаторы **iKey** компании Rainbow являются недорогими брелоками, которые могут использоваться на любой рабочей станции, имеющей универсальную последовательную шину (USB). Они обеспечивают надежность, простоту и безопасность в такой же степени, как и смарт-карты, но без сложностей и лишних затрат, связанных с с использованием считывателя [9]. iKey являются идеальным инструментом для контроля доступа к сетевым службам. iKey 2000 поддерживает и интегрируется со всеми основными прикладными системами, работающими по технологии PKI и используемыми в сетях отдельной организации, нескольких взаимодействующих организаций. Указанные системы включают Microsoft Internet Exolorer и Outlook, Netscape, Entrust, Baltimore, Xcert, Verisign и др. iKey 2000 разрабатывался для защиты цифровой идентичности в рамках инфраструктуры открытых ключей (PKI). iKey 2000 способен с помощью аппаратных средств генерировать и сохранять в памяти пары открытых ключей и цифровые сертификаты, а также производить цифровую подпись. Личный PKI-ключ недоступен компьютеру клиента.

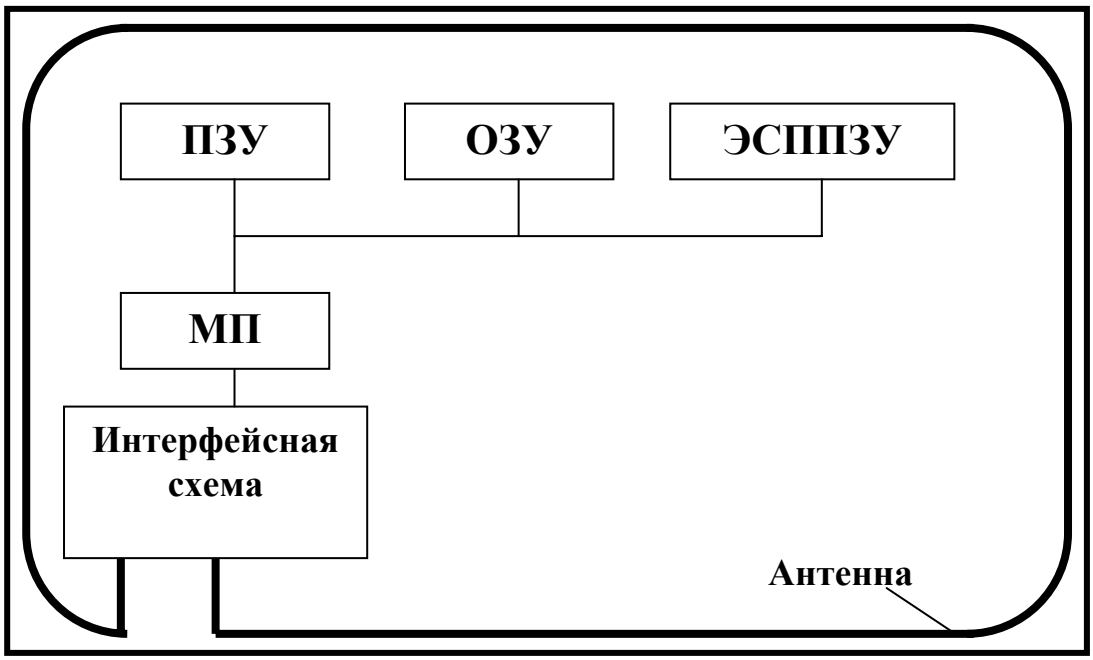

Рис. 4.3

iKey 2000 создает мощную систему защиты и криптографического кодирования непосредственно внутри аппаратного устройства. Для iKey 2000 пользователю поставляется программное обеспечение. Устройство содержит полный набор криптографических библиотек для броузеров Netscape и Internet Exolorer, а также для клиентов электронной почты. iKey 2000 действует одновременно как смарт-карта и считыватель, находящиеся в едином устройстве с конструктивом USB. Для активизации прикладной программы достаточно вставить iKey 2000 в USB-порт.

iKey 2000 реализует более простой метод обеспечения привилегий пользователя, чем пароли или чисто программные сертификаты. Чтобы запрограммировать ключ, администратору потребуется всего несколько минут. Потерянные ключи могут быть дезактивированы и изменены.

4.9 Создание защищенной операционной системы

Анализ архитектуры и характеристик сертифицированных защищенных систем показывает [1]:

1. В настоящее время имеются два принципиально различных метода к проектированию защищенных информационных систем. Первый метод предполагает разработку защищенной системы с нуля, и в этом случае для нее специально разрабатываются защищенные приложения. Вследствие значительной трудоемкости и стоимости такой подход приемлем для состоятельных производителей. Второй подход заключается в доработке системы-прототипа с целью улучшения ее характеристик защиты.

2. Принципы построения защищенных операционных систем изменялись с развитием сферы применения компьютерных систем. Защищенные ОС стали разрабатываться для рабочих станций и персональных компьютеров, для сетевых систем. Особое значение при этом имела ОС UNIX, которая наиболее часто служила основой для разработки защищенных ОС.

3. Для соответствия требованиям сертификации ОС должна обеспечивать произвольный и нормативный доступ. Для реализации произвольного доступа защищенные ОС используют как традиционный механизм битов защиты, так и списки контроля доступа (ACL). Для организации нормативного контроля доступа

в большинстве случаев используется модель Белла и Лападула. Обеспечение целостности данных в защищенных ОС обеспечивается применением нормативной модели Биба.

4. Так как при сертификации защищенные ОС подлежат формальному анализу, то защищенные системы должны разрабатываться на основе иерархического и модульного проектирования.

Все защищенные ОС в общем реализуют один и тот же набор защитных функций – управление доступом и контроль за его осуществлением, идентификация и аутентификация, аудит, прямое взаимодействие и т.д. Это связано с тем, что все разработчики ориентируются на соответствующие разделы стандартов информационной безопасности. Однако способы реализации этих функций, как и реализуемый ими уровень защиты, различаются от системы к системе даже в рамках одного класса требований [1].

Построению защищенной системы должно предшествовать определение класса требований стандарта информационной безопасности к данной системе. На современном уровне развития информационных технологий необходимо разрабатывать системы с жесткими требованиями по защищенности. В обязательном порядке должны быть реализованы произвольное и нормативное управление доступом, а структура ТСВ системы должна позволять применение формальных методов анализа.

Поэтому будем ориентироваться на требования безопасности уровня В3 «Оранжевой книги». Эти требования содержат поддержку формально определенной модели безопасности, предусматривают произвольное и нормативное управление доступом, метки безопасности, контроль доступа к субъектам и объектам. ТСВ в такой системе должна быть структурирована с целью исключения из нее подсистем, не отвечающих за функции защиты, и быть компактной для надежного тестирования и анализа. ТСВ должна быть минимизирована по сложности. Средства аудита должны содержать механизмы оповещения администратора о событиях, влияющих на безопасность системы. Требуется наличие средств восстановления работоспособности после сбоев и контроля скрытых каналов утечки информации.

С точки зрения обеспечения безопасности самой прогрессивной на сегодняшний день технологией построения ОС является технология микроядра. В отличие от традиционной архитектуры, в которой ОС представляет собой монолитное ядро, реализующее основные функции по управлению аппаратными ресурсами и организующее среду для выполнения пользовательских процессов, микроядерная архитектура распределяет функции ОС между микроядром и входящими в состав ОС системными сервисами (процессы, равноправные с пользовательскими приложениями).

Микроядро выполняет базовые функции операционной системы, на которые опираются эти системные сервисы и приложения. В итоге такие важные компоненты ОС как файловая система, сетевая поддержка и т.д. превращаются в независимые модули, которые функционируют как отдельные процессы и взаимодействуют с ядром и друг с другом на общих основаниях. Имевшее раньше место четкое разделение программного обеспечения на системные и прикладные программы размывается, т.к. между процессами, реализующими функции ОС, и прикладными процессами, выполняющими программы пользователя, нет никаких различий. Все компоненты системы используют средства микроядра для обмена сообщениями, но взаимодействуют непосредственно. Микроядро лишь проверяет законность сообщений, пересылает их между компонентами и обеспечивает доступ к аппаратуре.

Другое изменение в технологии построения ОС, связанное исключительно с внедрением технологии микроядра, это организация взаимодействий между процессами и ядром с помощью универсального механизма передачи информации – обмена сообщениями, пришедшего на смену технике системных вызовов. При этом десятки или даже сотни вызовов, различающихся числом и типом параметров, можно заменить несколькими типами сообщений, которые содержат компактные порции информации и могут передаваться от одного обработчика к другому.

На современном этапе развития ОС эта технология является самой перспективной, т.к. позволяет преодолеть самые заметные недостатки существующих систем – отсутствие мобильности, громоздкость, ресурсоемкость. Реализация многих традиционных

функций ОС за пределами ядра способствует построению на базе этого ядра операционных систем с недостижимым раннее уровнем модульности и расширяемости.

Чтобы иметь представление о базовом наборе понятий, необходимом для изложения деталей реализации средств защиты, рассмотрим некоторые понятия и основные принципы построения архитектуры современных микроядерных операционных систем на примере лежащего в основе Trusted Mach микроядра MK++[1].

4.9.1 Основные положения архитектуры микроядерных ОС

В основе архитектуры микроядерных ОС лежат следующие базовые концепции:

- минимизация набора функций, поддерживаемых микроядром, и реализация традиционных функций ОС (файловая система, сетевая поддержка) вне микроядра;
- организация синхронного и асинхронного взаимодействия между процессами через механизм обмена сообщениями;
- все отношения между компонентами строятся на основе модели клиент/сервер;
- применение объектно-ориентированного подхода при разработке архитектуры и программирования системы.

Минимизация функций микроядра дает возможность сконцентрировать в нем код, зависящий от аппаратной платформы, что позволяет повысить переносимость ОС до максимума. Таким образом, микроядро реализует только жизненно важные функции, лежащие в основе операционной системы, являющиеся базисом для всех системных служб, сервисов и прикладных программ.

Использование механизма передачи сообщений позволяет установить единый интерфейс для взаимодействия между всеми компонентами системы независимо от их уровня и назначения, что дает возможность строить все информационные связи в системе по модели клиент/сервер.

В модели клиент/сервер все компоненты рассматриваются либо как потребители (клиенты), либо как поставщики (серверы) некоторых ресурсов или сервисов. Стандартизированные протоколы предоставления сервиса или ресурсов позволяют серверу обслуживать клиентов независимо от деталей их реализации, что открывает перед разработчиками широкие возможности для построения распределенных систем. Инициатором обмена обычно является клиент, который посылает запрос на обслуживание серверу, находящемуся в состоянии ожидания запроса. Один и тот же процесс может являться клиентом по отношению к одним ресурсам и быть сервером для других. Данная модель успешно применяется не только при построении ОС, но и при создании программного обеспечения любого уровня. Применение модели клиент/сервер по отношению к ОС состоит в реализации не вошедших в состав ядра компонентов ОС, в виде множества серверов, каждый из которых предназначен для обслуживания определенного ресурса (например, управление памятью, процессами, контроль доступа и т.д.).

Наиболее полно раскрыть преимущества технологии клиент/сервер позволяет применение методов объектно-ориентированного проектирования и программирования. Если каждый сервер обслуживает только один тип ресурсов и представляет его клиентам в виде некоторой абстрактной модели, то такой сервер можно рассматривать как объект, т.к. он обладает всеми необходимыми для этого качествами. Объект должен обладать состоянием, поведением и индивидуальностью. Для каждого сервера существует четко определенная модель состояний и переходов между ними. И, наконец, «поведение» каждого сервера однозначно регламентируется протоколом его взаимодействия с клиентами. Соответственно, можно строить модель ОС, построенной по этим принципам, в виде иерархии серверов и моделей, представляемых ими ресурсов, а также описывать существующие между ними взаимосвязи с помощью объектных отношений наследования, использования и включения.

С точки зрения создания защищенных операционных систем, использование объектно-ориентированного подхода в сочетании с микроядром и технологией клиент/сервер позволяет разработчику реализовать взаимодействие субъектов и объектов, а

также контроль за информационными потоками с помощью ограниченного числа простых и понятных механизмов, что облегчает адекватность реализации модели безопасности и позволяет применять формальные методы анализа.

4.9.2 Микроядерная архитектура с точки зрения создания защищенных систем

Благодаря принципам, на которых основаны микроядерные ОС, их компоненты функционируют на основе очень небольшого и сравнительно простого набора абстракций, составляющих базис системы и компактно реализованных в микроядре. Можно сказать, что для защищенных систем такая архитектура является оптимальной, т.к. она позволяет достаточно просто и эффективно разрешить целый ряд вопросов, неизбежно возникающих при реализации защищенных систем:

1. Выявление потоков информации в системе. Поскольку все взаимодействия осуществляются исключительно посредством механизма передачи сообщений, очевидно, что, контролируя потоки сообщений, можно быть уверенным в том, что контролируются все информационные потоки в системе;

2. Определение субъектов и объектов взаимодействия. Как уже говорилось, все задачи в микроядерных системах связаны между собой отношениями клиент/сервер. Соответственно, субъектом взаимодействия всегда является задача-клиент, а объектом – ресурс, обслуживаемый задачей-сервером.

3. Размещение подсистемы контроля доступа. Поскольку единственным механизмом взаимодействия является передача сообщений, очевидно, что функция контроля за информационными потоками должна быть возложена на ту часть ядра системы, которая реализует этот механизм. Контроль и управление доступом к ресурсам и объектам могут быть реализованы, как в составе серверов, отвечающих за обслуживание этих ресурсов и объектов, так и на уровне всей системы в целом. Так как многие сервера обслуживают однотипные ресурсы и объекты (файлы, устройства и т.д.), то контроль доступа к ним с целью унификации реализуется на глобальном уровне, с помощью Сервера имен,

который формирует глобальное пространство имен системы и организует взаимодействие между клиентами и серверами.

4. Минимизация объема программного кода, отвечающего за контроль доступа. Как следует из предыдущего пункта, в системе существует всего две процедуры, реализующие контроль за осуществлением доступа: на уровне передачи сообщений и глобальная система на уровне именованных объектов. Таким образом, объем программ, корректность функционирования которых критична для безопасности всей системы, сокращен до минимума.

5. Использование объектно-ориентированных технологий программирования. Контроль за потоками сообщений и доступом процессов к ресурсам из глобального пространства имен можно осуществлять на основе унифицированного набора свойств сообщений (источник, приемник) и ресурсов (идентификатор процесса, имя ресурса). За счет этого достигается абстрагирование системы защиты от специфики информационных взаимодействий, но в то же время сохраняется ее гибкость за счет возможности использования в задачах-серверах специализированных механизмов защиты, адаптированных и конкретизированных к ресурсам, которые обслуживают эти сервера.

6. Верификация и анализ защиты. Достигнутая с помощью применения описанных решений простота и компактность средств контроля за осуществлением доступа очевидным образом, за счет структуризации системы, способствует применению формальных методов верификации и анализа программного кода средств защиты.

4.9.3 Микроядро как основа для создания защищенной ОС нового поколения – МК++

Микроядро МК++ отвечает всем требованиям, предъявляемым к ОС нового поколения (многопоточность, расширяемость, мобильность и т.д.) [1]. Разработчики преследовали следующие цели:

- отработать технологии реализации современных требований к операционным системам;

- создать основу для разработки будущих поколений защищенных ОС, рассчитанных на достаточно высокий класс требований безопасности;
- обеспечить возможность работы микроядра и приложений в режиме реального времени;
- предусмотреть максимальное использование параллельности, как для приложений, так и для самих компонентов операционной системы в расчете на распределенные и массово-параллельные системы будущего;
- обеспечить переносимость микроядра с одной аппаратной платформы на другую;
- обеспечить совместимость с существующим программным обеспечением.

Рассмотрим МК++ только с точки зрения создания основы для защищенной ОС.

Поскольку разработчики МК++ стремились создать классическую микроядерную систему, в которой практически все традиционные для ОС функции вынесены за пределы ядра, само микроядро осуществляет только следующий набор функций:

- управление физической аппаратурой (оперативной памятью, процессорами, внешними устройствами);
- распределение ресурсов аппаратной платформы между процессами (время процессора, память и т.д.);
- изоляция процессов;
- организация взаимодействия между процессами;
- управление процессами (создание, уничтожение, переключение).

Ядро является своеобразным арбитром, роль которого сводится к поддержанию некоторого набора «правил игры» внутри операционной системы, все остальные традиционные функции ОС должны быть реализованы вне ядра.

Для описания микроядерной архитектуры необходимо перечислить набор основных понятий, используемых в микроядерной технологии построения ОС:

Задача. В микроядерных системах это понятие заменяет традиционное для ОС понятие *процесс*. Задача представляет собой обобщение понятия процесс и обозначает набор ресурсов,

который образует среду для выполнения *потоков* (см. далее). Эта среда включает в себя:

- изолированное от других задач адресное пространство;
- среду выполнения прикладного процесса;
- атрибуты безопасности;
- средства взаимодействия с ядром;
- средства взаимодействия с другими задачами.

Каждая задача имеет свое собственное пространство имен *портов*. Задача может выступать в роли потребителя ресурсов (клиент) или предоставлять определенные ресурсы другим задачам (сервер). Одна и та же задача может являться одновременно и сервером, и клиентом, потребляя ресурсы, контролируемые одними задачами, и предоставляя свои ресурсы другим. Доступ к ресурсам, как и взаимодействие с другими задачами и ядром, осуществляется только с помощью обмена сообщениями через порты.

Поток (Thread) – логически связанный поток выполняемых команд. Каждый поток выполняется в контексте какой-либо *задачи* и может осуществлять непосредственный доступ только к ее среде. Потоки являются основной единицей вычислений и единственными активными элементами в системе. Поток представляет собой последовательность команд, выполняемых в рамках задачи. Его единственным атрибутом является состояние процессора. Все потоки внутри задачи совместно используют адресное пространство и наследуют атрибуты безопасности задачи.

Порт – однонаправленный информационный канал, с помощью которого задачи обмениваются информацией друг с другом и с ядром, осуществляя операции посылки и получения *сообщений*. Задачи могут получить доступ к портам только при наличии у них прав на посылку/прием сообщений.

Сообщение – логически связанный набор данных, предаваемый через порт за одно обращение. Для осуществления контроля доступа ядро снабжает все сообщения специальной меткой, идентифицирующей отправителя сообщения.

Пример взаимодействия между сервером, клиентом и микроядром с помощью портов и сообщений показан на рис. 4.4.

Рассмотрим внутреннюю структуру МК++, назначение и основные функции составляющих его компонентов. Как видно из рис. 4.5, структура МК++, представляет собой иерархию, в которой каждый уровень опирается на сервисы, реализуемые нижестоящими уровнями, и, в свою очередь, обслуживает уровни, лежащие выше.

Для того чтобы пояснить, как функционирует микроядро, рассмотрим в общих чертах тот сервис, который реализуется компонентами МК++ и некоторыми задачами, обеспечивающими важные для функционирования системы функции.

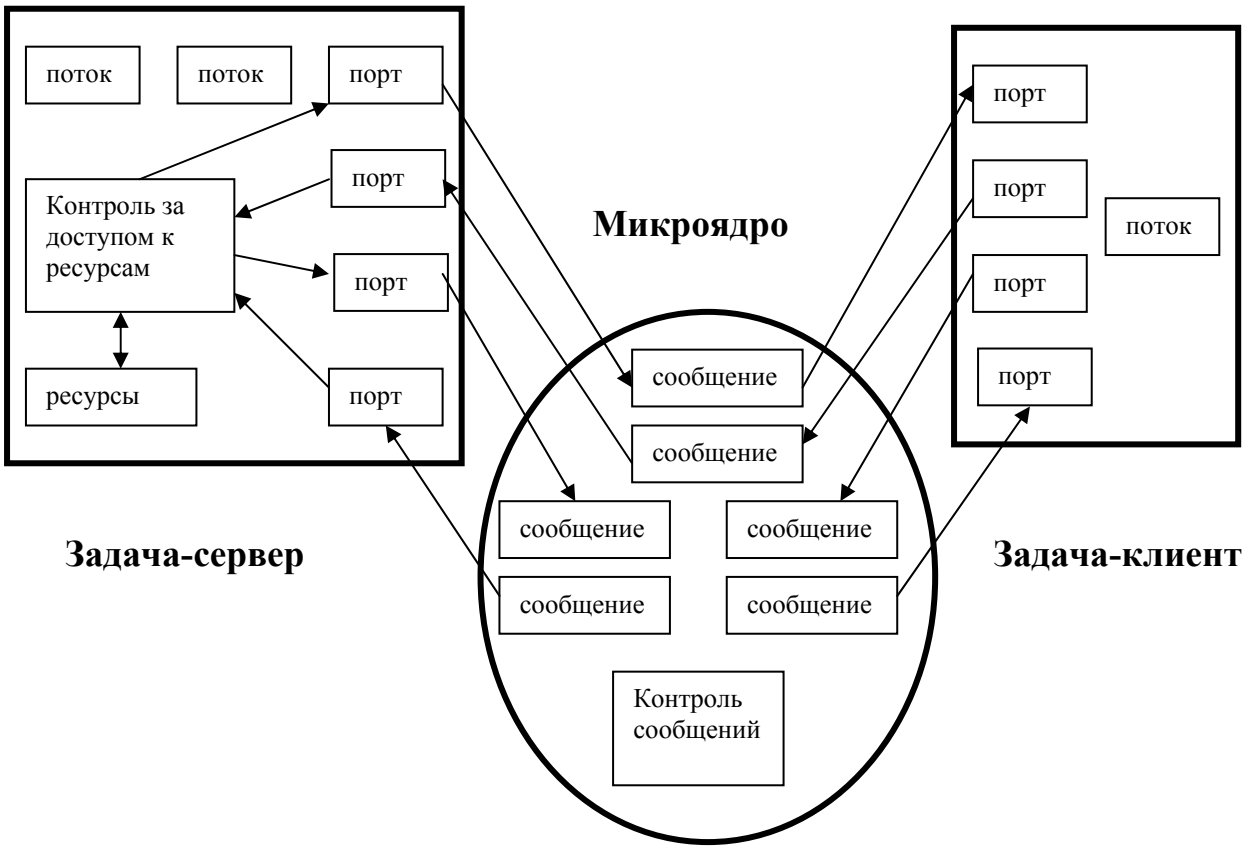

Рис. 4.4

1. Подсистема управления процессорами представляет собой самую низкоуровневую и в наибольшей степени зависимую от аппаратной платформы подсистему во всей структуре микроядра. Подсистема управления процессорами контролирует все переходы между различными режимами работы процессоров, в том числе изменение уровня привилегий и переходы между различными кольцами защиты, а также переключение с одного по-

тока команд на другой, и также обработку прерываний и исключений. Данный уровень скрывает от вышестоящих все особенности аппаратной архитектуры, такие как порядок сохранения и восстановления регистров, формат таблиц трансляции адресов, способы управления приоритетами прерываний и т.д. Таким образом, подсистема управления процессорами реализует следующие функции:

- управление состоянием процессоров;
- обработка аппаратных прерываний и исключений;
- синхронизация совместной работы нескольких процессоров;
- предоставление процессоров в распоряжение потоков и контроль за использованием процессорного времени.

2. Управление ресурсами микроядра. Служебные структуры (сообщения, порты и т.д.) требуют выделения определенных ресурсов (памяти) и при этом должны быть доступны как для приложений, так и из микроядра, что влечет за собой необходимость размещения их в адресном пространстве микроядра. Подсистема управления ресурсами памяти микроядра отвечает за выделение памяти в адресном пространстве микроядра, ее освобождение и контроль за использованием.

3. Подсистема организации взаимодействий реализует для вышележащих уровней сервис, позволяющий посылать и принимать элементы данных. Данная подсистема включает в свой состав универсальные механизмы, которые могут быть использованы для передачи абстрактных элементов (непосредственно или с буферизацией в очереди) от некоторого источника к некоторому приемнику. Тип элементов не имеет значения. Самым распространенным примером использования этого сервера является механизм передачи сообщений, обслуживающий информационный обмен между задачами и ядром, однако этот сервис может быть использован для других целей.

4. Подсистема управления физической памятью отвечает за порядок распределения и использования физической памятью системы. Кроме того, данная подсистема выполняет машинно-зависимые операции трансляции адресов для различных адресных пространств.


Загрузчик
Сервер свопинга
Интерфейс микроядра
Управление ресурсами микроядра
Управление портами
Управление адресными пространствами задач
Виртуальные устройства
Реальное время
Управление виртуальной памятью
Идентификация объектов микроядра
Управление вводом-выводом и таймерами
Управление физической памятью
Организация взаимодействий
Управление ресурсами микроядра
Управление процессорами
Аппаратная платформа

Рис. 4.5

5. Подсистема ввода-вывода выполняет низкоуровневый ввод/вывод информации, специфичный для каждого конкретного устройства. Кроме того, эта подсистема управляет аппаратными таймерами, а также осуществляет обработку прерываний от внешних устройств.

6. Подсистема идентификации ставит в соответствие каждой сущности уровня микроядра (задачи, потоки, порты и т.д.) уникальный идентификатор.

7. Идентификатор никогда не используется повторно, даже если объект, на который он ссылается, уничтожен. Эта подсистема обеспечивает механизм взаимодействий на уровне всей системы. Все манипуляции с объектами микроядра опираются на этот сервис.

8. Подсистема виртуальной памяти отвечает за отображение виртуальных адресных пространств ядра и задач в физическую память и обеспечивает инициализацию и очистку объектов памяти.

9. Подсистема реального времени обеспечивает функции работы с сигналами и таймерами и осуществляет доступ к аппаратным часам.

10. Подсистема управления виртуальными устройствами поддерживает абстракции, представляющие их на уровне задач. Данная подсистема ответственна за присваивание имен и реализацию операций открытия/закрытия для всех устройств.

11. Подсистема управления адресными пространствами организует виртуальные адресные пространства задач и пользуется сервисом виртуальной памяти для манипулирования областями памяти, находящимися в адресном пространстве задач. Эта подсистема также осуществляет низкоуровневое управление доступом к пространствам памяти.

12. Подсистема управления портами отвечает за организацию пространства имен портов для задач и осуществляет отображение имен портов во внутренние идентификаторы микроядра с помощью подсистемы идентификации. Именно эта подсистема реализует контроль за информационными потоками и определяет, обладает правом приема сообщений из порта и отправки в него.

13. Подсистема управления ресурсами обеспечивает высокоуровневые средства управления ресурсами ядра, в частности

задачами и потоками. Управление состоит в отслеживании событий, связанных с ресурсами, и манипуляциях с ними, например, создание, завершение и приостановка выполнения потоков. Специальный механизм данной подсистемы позволяет реализовать различные политики управления ресурсами ядра.

14. Интерфейс микроядра определяет границу между выполнением потоков в контексте ядра и в пользовательском контексте. Подсистема управления процессорами осуществляет переключение процессора таким образом, что вход в контекст микроядра из прикладной задачи и выход из него возможны только через интерфейс микроядра.

15. Сервер загрузки находится вне микроядра и представляет собой обычную пользовательскую задачу, создаваемую процессом инициализации ОС. Целью этой задачи является загрузка и запуск серверов, реализующих системные сервисы, в том числе входящих в состав ТСВ.

16. Сервер свопинга также является обычной задачей. Он обеспечивает сохранение на диске областей виртуальной памяти, вытесненных из физической памяти.

В микроядре отсутствуют средства, отвечающие за реализацию политики безопасности и управления доступом к информационным ресурсам системы. Однако МК++ включает два механизма, необходимые для реализации этих функций, а именно: изоляцию задач и контроль за передачей сообщений. Все остальные механизмы защиты, опирающиеся на этот сервис, могут быть реализованы в составе серверов. В совокупности микроядро и эти серверы образуют ТСВ системы.