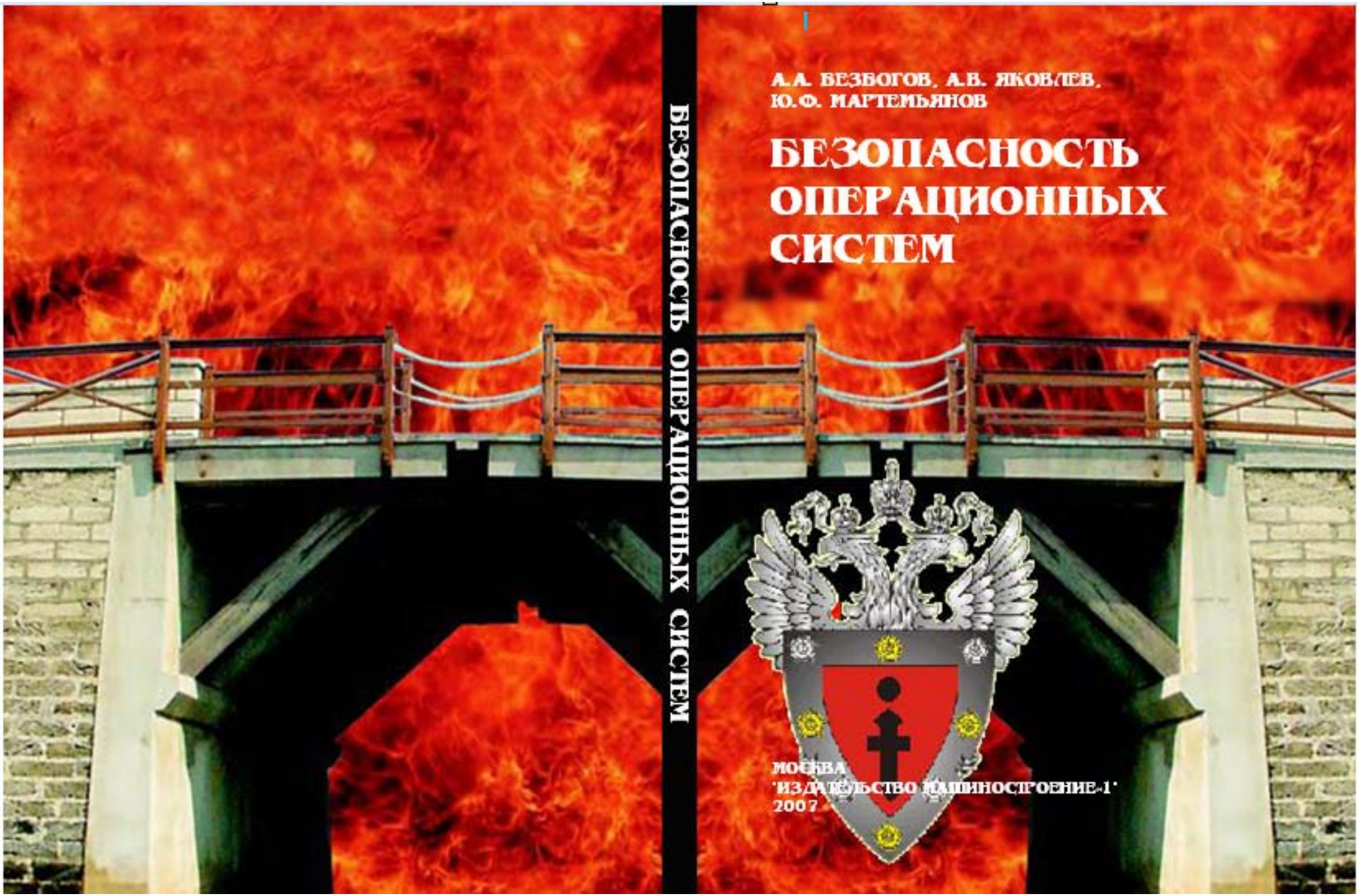
БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ
А.А. БЕЗБОГОВ, А.В. ЯКОВЛЕВ,
Ю.Ф. МАРТЕМЬЯНОВ
БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ
МОСКВА
"ИЗДАТЕЛЬСТВО МАШИНОСТРОЕНИЕ-1"
2007

УДК 004.451(075)
ББК ←973-018.3я73
Б391

Р е ц е н з е н т ы :

Кандидат физико-математических наук, доцент,
заместитель председателя УМС, начальник кафедры
программирования и компьютерной безопасности ИКСИ
А.В. Черемушкин

Кандидат технических наук, доцент
кафедры программирования и компьютерной безопасности ИКСИ
В.Г. Проскурин

Безбогов, А.А.

Б391 Безопасность операционных систем : учебное пособие / А.А. Безбогов, А.В. Яковлев, Ю.Ф. Мартемьянов. – М. : "Издательство Машиностроение-1", 2007. – 220 с. – 400 экз. ISBN 978-5-94275-348-1

Учебное пособие содержит материал в соответствии с Государственным образовательным стандартом по специальности 090105.

Рассматриваются принципы построения и концептуальные основы операционных систем, методы, средства и алгоритмы управления процессами, памятью и вводом-выводом в операционных системах. Приведены понятия и основные положения в информационно-вычислительных системах, стандарты и спецификации в области информационной безопасности, а также модели основных типов политик безопасности. Подробно рассмотрены модели и механизмы защиты операционных систем, программного обеспечения, а также протоколирование и аудит.

Пособие может быть полезно при курсовом и дипломном проектировании, аспирантам, а также кругу читателей, интересующихся современными проблемами безопасности операционных систем.

УДК 004.451(075)
ББК ←973-018.3я73

ISBN 978-5-94275-348-1

А.А. Безбогов, А.В. Яковлев, Ю.Ф. Мартемьянов

БЕЗОПАСНОСТЬ ОПЕРАЦИОННЫХ СИСТЕМ

Допущено Учебно-методическим объединением по образованию в области информационной безопасности в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальности 090105 "Комплексное обеспечение информационной безопасности автоматизированных систем"

МОСКВА
"ИЗДАТЕЛЬСТВО МАШИНОСТРОЕНИЕ-1"
2007

Учебное издание

БЕЗБОГОВ Александр Александрович,
ЯКОВЛЕВ Алексей Вячеславович,
МАРТЕМЬЯНОВ Юрий Федорович

Безопасность операционных систем

Учебное пособие

Редактор З.Г. Ч е р н о в а
Инженер по компьютерному макетированию М.Н. Р ы ж к о в а
Дизайн обложки А.А. П о г о р е л о в а

Подписано в печать 31.05.2007.
Формат 60 × 84/16. 12,79 усл. печ. л. Тираж 400 экз. Заказ № 396.

"Издательство Машиностроение-1"
107076, Москва, Стромынский пер., 4

Подготовлено к печати и отпечатано в Издательско-полиграфическом центре
Тамбовского государственного технического университета
392000, Тамбов, Советская, 106, к. 14
Контактный телефон (4752)71-81-08

СОДЕРЖАНИЕ

предисловие

Стремительное развитие информационных технологий привело к формированию информационной среды, оказывающей влияние на все сферы человеческой деятельности. Однако с развитием информационных технологий возникают и стремительно растут риски, связанные с их использованием, появляются совершенно новые угрозы, с последствиями, от реализации которых человечество раньше не сталкивалось.

Одним из главных инструментов для реализации конкретных информационных технологий являются информационные системы, задача обеспечения безопасности которых является приоритетной, так как от сохранения конфиденциальности, целостности и доступности информационных ресурсов зависит результат деятельности информационных систем.

Операционная система является важнейшим программным компонентом любой вычислительной машины, поэтому от уровня реализации политики безопасности в каждой конкретной операционной системе во многом зависит и общая безопасность информационной системы.

В связи с этим знания в области современных методов и средств обеспечения безопасности операционных систем являются необходимым условием для формирования специалиста по информационной безопасности.

Данное учебное пособие предназначено для студентов, обучающихся по специальности 090105 – "Комплексное обеспечение информационной безопасности автоматизированных систем" и может быть использовано при изучении курсов "Компьютерная безопасность", "Защита компьютерной информации" и других дисциплин, будет также полезно для студентов, аспирантов и преподавателей вузов соответствующих специальностей.

Авторы выражают глубокую признательность коллегам из ИКСИ Академии ФСБ России Е.Б. Белову, В.П. Лосю, А.В. Черемушкину и В.Г. Проскурину, чьи замечания и пожелания способствовали улучшению качества учебного пособия. Предложения и замечания по книге будут с благодарностью приняты по адресу: fit@mail.nnn.tstu.ru.

Р а з д е л I

АРХИТЕКТУРА ОПЕРАЦИОННЫХ СИСТЕМ

Т Е М А 1. Принципы построения операционных систем

1.1. ПОНЯТИЕ ОБ АРХИТЕКТУРЕ АППАРАТНЫХ СРЕДСТВ

1.1.1. Вводные замечания

Появление общего программного обеспечения в ЭВМ относят к 1953 г., когда в СССР появилась одна из первых теоретических работ по автоматизации программирования для цифровых ЭВМ (А.П. Ершов), а в Массачусетском технологическом институте (США) была создана экспериментальная "операционная система", применявшаяся в учебных целях. Затем появились специализированные операционные системы (ОС) для обслуживания оборонных вычислительных систем реального времени. Однако эти разработки имели экспериментальный, исследовательский характер и широкого распространения в то время не получили. Тем не менее, потребности практического использования ЭВМ в различных предметных областях, необходимость более эффективного использования ЭВМ, повышение производительности труда разработчиков программного обеспечения, а также стремление расширить рынок сбыта ЭВМ вызвали стремительный прогресс в создании теории и инструментальных средств общего программного обеспечения вычислительных систем.

Построение вычислительных машин основано на трех принципах:

1. Принцип цифрового представления данных (чисел, команд, обозначение операций, букв, слов и т.д.). Единицами данных в ЭВМ являются бит, байт, слово и т.п.
2. Принцип адресности данных – все данные и любые объекты программы хранятся в ячейках памяти, имеющих адрес.
3. Принцип программного управления (Ч. Беббидж, 1834 г.) – управление вычислительным процессом осуществляется с помощью программы, находящейся в памяти ЭВМ.

Все универсальные вычислительные машины, в том числе и персональные компьютеры, имеют структуру, показанную на рис. 1.1.

Впервые такую структуру вычислительных машин предложил Джон фон Нейман в 1945 г., поэтому ЭВМ с такой структурой называют машинами фон Неймана.

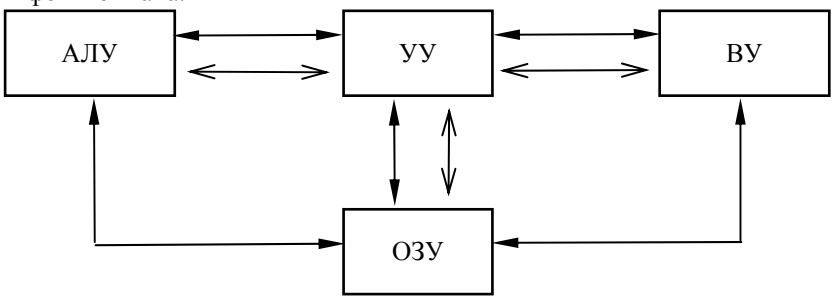

Рис. 1.1. Общая структура универсальной ЭВМ:
АЛУ – арифметическо-логическое устройство; УУ – устройство управления; ВУ – внешние устройства; ОЗУ – оперативное запоминающее устройство

Конкретная аппаратурная реализация схемы изменялась от поколения к поколению ЭВМ. Например, в современных компьютерах АЛУ и УУ объединены в единое устройство – *центральный процессор*. Кроме того, в ЭВМ ввели *систему прерываний*. Появились многопроцессорные ЭВМ, позволяющие осуществлять параллельную обработку данных в компьютере. Тем не менее, функциональная структура существующих компьютеров в основном соответствует структуре машины фон Неймана.

Архитектура вычислительной системы – общая логическая организация цифровой вычислительной системы, определяющая процесс обработки данных в конкретной вычислительной системе и включающая методы кодирования данных, состав, назначение, принципы взаимодействия технических средств и программного обеспечения.

Большинство из выпускаемых сейчас компьютеров выполнено в соответствии с принципом *открытой архитектуры*, впервые примененном в персональной ЭВМ IBM PC (фирма IBM, 1981 г.).

1.1.2. Классификация программных средств

Программное обеспечение вычислительных систем принято делить на следующие виды:

1) общее (системное) программное обеспечение (ОПО);
2) специальное программное обеспечение (СПО).

Введем ряд определений.

В состав общего программного обеспечения вычислительных систем входят:

- программные средства управления обработкой данных, включая операционные системы;
- обслуживающие (сервисные) программы (утилиты);
- инструментальные программные средства.

Специальное ПО делят на следующие виды:

- прикладные программы (приложение) общего назначения;
- прикладные программы пользователя.

Прикладные программы общего назначения можно разделить на следующие группы:

– программа офисного назначения;
– программа экономического назначения;
– издательские системы;
– компьютерная графика, видео, анимация и звук;
– системы управления базами данных;
– прочие прикладные программы общего назначения.

Можно видеть, что современные компьютеры и их программное обеспечение глубоко внедрилось практически во все сферы человеческой деятельности: науку, производство, экономику, право и т.д.

Функционирование прикладных программ любого назначения происходит под управлением и при участии программ, относящихся к категории системного программного обеспечения.

1.1.3. Место и функции системного программного обеспечения

Системное ПО играет роль "прослойки" между пользователем и техническими средствами вычислительной системы. На различных этапах работы с компьютером в качестве такой "прослойки" выступают разные программы и пакеты программ системного ПО, выполняя при этом отличающиеся назначением функции.

Основой системного ПО является операционная система.

Операционная система **(ОС)** ***цифровой вычислительной системы*** – система программ, предназначенная для обеспечения определенного уровня эффективности цифровой вычислительной системы за счет автоматизированного управления ее работой и предоставляемого пользователям набора услуг.

Основными функциями ОС являются:

1) автоматическое выполнение действий по запуску задач в обработку и их завершению;
2) диспетчеризация (планирование обработки задач);
3) распределение памяти между различными задачами;
4) управление ходом выполнения задач в вычислительной системе;
5) распределение задачам необходимых ресурсов ВС;
6) синхронизация выполнения задач;
7) поддержка выполнения операций ввода/вывода данных;
8) ведение учета работы системы (при необходимости).

Выполнение своих функций ОС осуществляется с помощью соответствующих программных комплексов управления, которые носят название супервизорных программ (супервизоров или менеджеров).

Супервизорная программа – машинная программа, являющаяся обычно частью операционной системы, которая управляет выполнением других машинных программ и регулирует поток работ в системе управления данными.

Супервизор – часть управляющей программы, координирующая распределение ресурсов вычислительной системы.

В целом современные операционные системы представляют собой иерархическую структуру (рис. 1.2).

В основе иерархии находится аппаратура вычислительной машины, называемая иногда "чистой машиной" или "голым железом". На следующем уровне иерархии (иногда на следующих нескольких уровнях) находятся некоторые функции ядра операционной системы. В совокупности с этими функциями ядра (называемыми еще "примитивами") компьютер становится *расширенной машиной*, т.е. машиной, которая представляет для операционной системы и пользователей не только свой машинный язык, но и ряд дополнительных возможностей.

Выше над ядром расположены программы ОС для обеспечения выполнения задач пользователя (управления внешними устройствами, обслуживание операций ввода/вывода и т.п.). На вершине иерархии находятся программы пользователя. В подобных иерархических системах принято, как правило, следующее ограничение: допускается обращение только сверху вниз в иерархии, т.е. средства каждого уровня могут обращаться только к тем функциям, которые находятся на ближайшем нижележащем уровне.

Рис. 1.2. Структура операционной системы

Обслуживающие* (*сервисные*) *программы (утилиты) предназначены для выполнения различных вспомогательных функций и разделяются на следующие типы: программы-упаковщики (архиваторы); антивирусные программы; программы резервирования; программы диагностики компьютера; программы оптимизации дисков; программы динамического сжатия дисков.

Инструментальные программные средства, называемые также средствами разработки приложений и системами программирования, являются орудием автоматизации разработок программного обеспечения ЭВМ, обеспечивающим повышение производительности труда разработчиков и надежности ПО.

К инструментальным программным средствам относятся:

- компиляторы и интерпретаторы;
- автономные отладчики (дебагеры, от англ. Debug "удаление насекомых");
- интегрированные оболочки;
- средства создания приложений типа клиент-сервер и т.п.

Существующие инструментальные программные средства обеспечивают разработчиков ПО всем необходимым набором функций для создания мощного программного обеспечения решения прикладных задач любой мощности для практически всех предметных областей.

1.2. ПРИНЦИПЫ РАБОТЫ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

Всякая вычислительная система создается для решения некоторого множества вычислительных или информационных задач, которые в совокупности называются задачами обработки данных. Для успешного решения любой задачи в вычислительной системе необходимо иметь:

1) программу, реализующую алгоритм решения задачи;

2) аппаратные средства ВС для ввода программы, выполнения программы, получения дополнительной информации и вывода результатов;

3) дополнительные программные средства, необходимые для решения прикладной задачи (стандартные программы).

Существует три вида систем обработки данных (СОД), отличающихся друг от друга требованиями к скорости получения результатов решения задач:

1) *системы реального времени* (СРВ), в которых требования к скорости обработки информации очень высокие из-за необходимости решения задач в темпе реального времени (примером являются системы навигации и управления летательными аппаратами);

2) *системы оперативной обработки* (СОО), в которых планирование заданий на обработку данных осуществляется, исходя из требования минимальности времени выполнения каждого полученного задания. Примером такого вида систем является система обработки данных для персонала боевых расчетов пунктов управления;

3) *системы пакетной обработки* (СПО), в которых основным требованием является минимизация простоя оборудования при решении поставленных задач.

Запуск прикладной программы в работу, предоставление ей необходимых аппаратных мощностей и программных средств осуществляется операционной системой.

1.3. РЕЖИМЫ РАБОТЫ ОПЕРАЦИОННЫХ СИСТЕМ

1.3.1. Режимы обработки данных

Порядок представления прикладной программе перечисленных средств определяется *режимом обработки данных*, реализованных в операционной системе ЭВМ. Различают однопрограммные и мультипрограммные режимы обработки данных и, соответственно, работы ОС.

К однопрограммным режимам относятся:

- режим непосредственного доступа (РНД);
- пакетный однопрограммный режим (П1П).

Мультипрограммными режимами обработки данных являются:

- пакетный мультипрограммный режим (ПМП);
- режим разделения времени (РРВ).

1.3.1.1. ОДНОПРОГРАММНЫЕ РЕЖИМЫ ОБРАБОТКИ ДАННЫХ

Режим непосредственного доступа широко применялся в ЭВМ первого поколения и используется при работе с современными персональными компьютерами. РНД характерен тем, что ЭВМ предоставляется только одному пользователю, который осуществляет взаимодействие с машиной посредством пульта управления (сейчас – клавиатура, мыши и дисплеи). Время решения каждой задачи в РНД складывается из времени $T_{ВВ}$ ввода программы и данных в ЭВМ, времени $T_{Р}$ работы процессора над решением задачи, времени $T_{ВУ}$ обмена данными с внешними устройствами (включая вывод результатов и обработки), времени $T_{ОП}$ обслуживания ЭВМ и задачи оператором ЭВМ при ее подготовке к запуску и по окончании решения задачи:

$$T_{НД} = T_{ВВ} + T_{ЦП} + T_{ВУ} + T_{ОП}\,.$$

Коэффициент загрузки процессора при одной задаче составляет

$$\eta_{НД} = T_{ЦП} / T_{НД}.$$

Полное время решения N задач и коэффициент загрузки:

$$T_{НД}(N) = \sum_{i=1}^{N} [T_{ВВ}(i) + T_{ЦП}(i) + T_{ВУ}(i) + T_{ОП}(i)];$$

$$\eta_{НД}(N) = \left[\sum_{i=1}^{N} T_{ЦП}(i) \right] \Big/ T_{НД},$$

где i – номер задачи.

В РНД наличие ОС не обязательно.

Недостатками РНД являются:

1) аппаратура и программы ЭВМ используются не эффективно;
2) велики затраты времени программиста на управление машиной;
3) предъявляются высокие требования к подготовке пользователя как оператора вычислительной машины.

Пакетный однопрограммный режим применяется в ВС, начиная с ЭВМ второго поколения. Несколько заданий для решения задач обработки собираются в один пакет, называемый пакет заданий (ПЗ). Пакет заданий оператор ЭВМ вводит в ЭВМ, где ПЗ сначала записывается во внешнюю память (магнитные диски, магнитные барабаны и т.п.). Затем операционная система машины последовательно считывает задания, входящие в ПЗ, и осуществляет выполнение необходимых в соответствиями с заданиями действий для решения задач пользователей. После завершения очередного задания происходит обращение к ОС, которая активирует начало выполнения следующего. После завершения последнего задания пакета оператор ЭВМ загружает в машину новый пакет заданий.

Режим П1П обладает следующими положительными чертами:

1) более высокая пропускная способность;
2) отсутствие специальных требований к аппаратуре ЭВМ;
3) возможна его реализация на любой ЭВМ.

К недостаткам режима П1П относятся:

1) необходимо наличие операционной системы;
2) пользователь физически отделен от ЭВМ и решаемой им задачи;
3) увеличивается реакция пользователя на полученные результаты решения;
4) последовательный порядок выполнения заданий пакетов не позволяет увеличить загрузку оборудования вычислительной системы.

1.3.1.2. МНОГОПРОГРАММНЫЕ РЕЖИМЫ ОБРАБОТКИ ДАННЫХ

Пакетный мультипрограммный режим широко применяется в ЭВМ третьего и последующих поколений. ПМП является режимом классического мультипрограммирования, при котором в вычислительной системе находятся в обработке сразу несколько заданий. На входе в систему формируется набор пакетов заданий, которые оператор ЭВМ загружает в систему. После окончания ввода первого ПЗ операционная система начинает его обработку, не дожидаясь ввода второго и последующих ПЗ. Задания, принадлежащие одному пакету, выполняются последовательно (т.е. в режиме П1П). Задания, принадлежащие разным пакетам, выполняются параллельно. Первым начинает выполняться первое задание первого пакета. По мере освобождения ресурсов ОС активизирует выполнение заданий из других пакетов в порядке их следования внутри ПЗ.

Пакетный мультипрограммный режим обеспечивает наивысшую пропускную способность вычислительной системы, что достигается при наличии в ЭВМ следующих аппаратных средств:

1) автономно управляемые внешние устройства;
2) развитая система прерывания программ;
3) средства защиты памяти от взаимного влияния программ.

Основным недостатком режима ПМП является практически полное устранение пользователя из системы и, как следствие, отсутствие связи пользователя со своей задачей.

Режим разделения времени существенно отличается от классического мультипрограммирования, реализованного в ПМП, и является в настоящее время основным режимом функционирования операционных систем. Главное в режиме разделения времени – это предоставление каждой задаче (или пользователю, работающему в диалоге с машиной) ресурсов ЭВМ на некоторый ограниченный интервал времени (квант). По истечении кванта времени данная программа свертывается операционной системой, развертывается следующая по очереди программа (или подключается следующий терминал пользователя), которой предоставляются ресурсы ЭВМ, и т.д.

1.3.2. Режимы и дисциплины обслуживания

Порядок обслуживания заданий (заявок на работу) в операционных системах с мультипрограммированием, т.е. реализующих режимы ПМП или РРВ, определяются принятыми в них *режимами обслуживания и дисциплинами обслуживания*.

1.3.2.1. РЕЖИМЫ ОБСЛУЖИВАНИЯ

Режимом обслуживания называется правило отбора заявок на обслуживание.

Режимы обслуживания делятся на три вида:

1) режим одиночного отбора заявок:
2) режим группового отбора, когда на обслуживание отбирается вся очередь заявок определенного типа;
3) смешанный режим отбора, когда для одних классов заявок производится одиночный отбор, а для других групповой.

1.3.3.2. ДИСЦИПЛИНЫ ОБСЛУЖИВАНИЯ

Дисциплиной обслуживания называется правило отбора заявок на обслуживание при заданном режиме обслуживания.

Для каждого из режимов обслуживания может быть применен один из следующих видов дисциплин обслуживания:

- бесприоритетное обслуживание;
- обслуживание с приоритетом;
- обслуживание по расписанию.

Разновидности дисциплины бесприоритетного обслуживания:

1) ОПП – обслуживание в порядке поступления ("первый пришел – первый обслужен", FIFO);
2) ООП – обслуживание в обратном порядке ("первый пришел – последний обслужен", LIFO);
3) ОСП – обслуживание в случайном порядке.

При бесприоритетном обслуживании считается, что все заявки имеют равное право на обслуживание.

Если требуется, чтобы заявки некоторого типа имели преимущества перед другими на их обслуживание операционной системой, то применяется дисциплина обслуживания с приоритетами:

1) ДОП – дисциплина обслуживания с относительными приоритетами, когда приоритет заявки влияет только на ее место в очереди заявок на обслуживание;
2) ДАП – дисциплина с абсолютными приоритетами, когда высоко приоритетная заявка получает преимущества не только перед заявками, стоящими в очереди, но и перед заявкой, получающей обслуживание;
3) ДСП – дисциплина со смешанными приоритетами, при которой к одним группам заявок применяются относительные приоритеты, а к другим – абсолютные;
4) ДДП – дисциплина обслуживания с динамическими приоритетами, когда значение приоритетов заявок может изменяться (расти) по мере их нахождения в очереди, обеспечивая тем самым первоочередное обслуживание заявок, долго находящихся в системе.

Дисциплина обслуживания по расписанию обеспечивает заданный пользователем порядок обработки заданий независимо от очередности их поступления в систему. Она применяется в тех случаях, когда результаты решения одной задачи являются входными данными для другой.

1.4. КЛАССИФИКАЦИЯ ОПЕРАЦИОННЫХ СИСТЕМ

Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами), особенностями использованных методов проектирования, типами аппаратных платформ, областями использования и многими другими свойствами.

Ниже приведена классификация ОС по нескольким наиболее основным признакам.

1.4.1. Особенности алгоритмов управления ресурсами

От эффективности алгоритмов управления локальными ресурсами компьютера во многом зависит эффективность всей сетевой ОС в целом. Поэтому, характеризуя сетевую ОС, часто приводят важнейшие особенности реализации функций ОС по управлению процессорами, памятью, внешними устройствами автономного компьютера. Так, например, в зависимости от особенностей использованного алгоритма управления процессором, операционные системы делят на многозадачные и однозадачные, многопользовательские и однопользовательские, на системы, поддерживающие многонитевую обработку и не поддерживающие ее, на многопроцессорные и однопроцессорные системы.

1.4.1.1. ПОДДЕРЖКА МНОГОЗАДАЧНОСТИ

По числу одновременно выполняемых задач операционные системы могут быть разделены на два класса:

1) однозадачные (например, MS DOS, MSX);
2) многозадачные (ОС ЕС, OS/2, Unix, Windows 95).

Однозадачные ОС в основном выполняют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером. Однозадачные ОС включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем.

Многозадачные ОС, кроме вышеперечисленных функций, управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства.

Поддержка многопользовательского режима. По числу одновременно работающих пользователей ОС делятся на:

1) однопользовательские (MS DOS, Windows 3.x, ранние версии OS/2);
2) многопользовательские (Unix, Windows NT).

Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей. Следует заметить, что не всякая многозадачная система является многопользовательской, и не всякая однопользовательская ОС является однозадачной.

Вытесняющая и невытесняющая многозадачность. Важнейшим разделяемым ресурсом является процессорное время. Способ распределения процессорного времени между несколькими одновременно существующими в системе процессами (или нитями) во многом определяет специфику ОС. Среди множества существующих вариантов реализации многозадачности можно выделить две группы алгоритмов:

1) невытесняющая многозадачность (NetWare, Windows 3.x);
2) вытесняющая многозадачность (Windows NT, OS/2, Unix).

Основным различием между вытесняющим и невытесняющим вариантами многозадачности является степень централизации механизма планирования процессов. В первом случае механизм планирования процессов целиком сосредоточен в операционной системе, а во втором – распределен между системой и прикладными программами. При невытесняющей многоза-

дачности активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению процесс. При вытесняющей многозадачности решение о переключении процессора с одного процесса на другой принимается операционной системой, а не самим активным процессом.

1.4.1.2. ПОДДЕРЖКА МНОГОНИТЕВОСТИ

Важным свойством операционных систем является возможность распараллеливания вычислений в рамках одной задачи. Многонитевая ОС разделяет процессорное время не между задачами, а между их отдельными ветвями (нитями).

Многопроцессорная обработка. Другим важным свойством ОС является отсутствие или наличие в ней средств поддержки многопроцессорной обработки – мультипроцессирование. Мультипроцессирование приводит к усложнению всех алгоритмов управления ресурсами.

В наши дни становится общепринятым введение в ОС функций поддержки многопроцессорной обработки данных. Такие функции имеются в операционных системах Solaris 2.x фирмы Sun, Open Server 3.x компании Santa Crus Operations, OS/2 фирмы IBM, Windows NT фирмы Microsoft и NetWare 4.1 фирмы Novell.

Многопроцессорные ОС могут классифицироваться по способу организации вычислительного процесса в системе с многопроцессорной архитектурой: асимметричные ОС и симметричные ОС. Асимметричная ОС целиком выполняется только на одном из процессоров системы, распределяя прикладные задачи по остальным процессорам. Симметричная ОС полностью децентрализована и использует весь пул процессоров, разделяя их между системными и прикладными задачами.

Выше были рассмотрены характеристики ОС, связанные с управлением только одним типом ресурсов – процессором. Важное влияние на облик операционной системы, в целом, на возможности ее использования в той или иной области оказывают особенности и других подсистем управления локальными ресурсами – подсистем управления памятью, файлами, устройствами ввода-вывода.

Специфика ОС проявляется и в том, каким образом она реализует сетевые функции: распознавание и перенаправление в сеть запросов к удаленным ресурсам, передача сообщений по сети, выполнение удаленных запросов. При реализации сетевых функций возникает комплекс задач, связанных с распределенным характером хранения и обработки данных в сети: ведение справочной информации о всех доступных в сети ресурсах и серверах, адресация взаимодействующих процессов, обеспечение прозрачности доступа, тиражирование данных, согласование копий, поддержка безопасности данных.

1.4.2. Особенности аппаратных платформ

На свойства операционной системы непосредственное влияние оказывают аппаратные средства, на которые она ориентирована. По типу аппаратуры различают операционные системы персональных компьютеров, мини-компьютеров, мейнфреймов, кластеров и сетей ЭВМ. Среди перечисленных типов компьютеров могут встречаться как однопроцессорные варианты, так и многопроцессорные. В любом случае специфика аппаратных средств, как правило, отражается на специфике операционных систем.

Очевидно, что ОС большой машины является более сложной и функциональной, чем ОС персонального компьютера. Так в ОС больших машин функции по планированию потока выполняемых задач, очевидно, реализуются путем использования сложных приоритетных дисциплин и требуют большей вычислительной мощности, чем в ОС персональных компьютеров. Аналогично обстоит дело и с другими функциями.

Сетевая ОС имеет в своем составе средства передачи сообщений между компьютерами по линиям связи, которые совершенно не нужны в автономной ОС. На основе этих сообщений сетевая ОС поддерживает разделение ресурсов компьютера между удаленными пользователями, подключенными к сети. Для поддержания функций передачи сообщений сетевые ОС содержат специальные программные компоненты, реализующие популярные коммуникационные протоколы, такие как IP, IPX, Ethernet и другие.

Многопроцессорные системы требуют от операционной системы особой организации, с помощью которой сама операционная система, а также поддерживаемые ею приложения могли бы выполняться параллельно отдельными процессорами системы. Параллельная работа отдельных частей ОС создает дополнительные проблемы для разработчиков ОС, так как в этом случае гораздо сложнее обеспечить согласованный доступ отдельных процессов к общим системным таблицам, исключить эффект гонок и прочие нежелательные последствия асинхронного выполнения работ.

Другие требования предъявляются к операционным системам кластеров. Кластер – слабо связанная совокупность нескольких вычислительных систем, работающих совместно для выполнения общих приложений, и представляющихся пользователю единой системой. Наряду со специальной аппаратурой для функционирования кластерных систем необходима и программная поддержка со стороны операционной системы, которая сводится в основном к синхронизации доступа к разделяемым ресурсам, обнаружению отказов и динамической реконфигурации системы. Одной из первых разработок в области кластерных технологий были решения компании Digital Equipment на базе компьютеров VAX. Недавно этой компанией заключено соглашение с корпорацией Microsoft о разработке кластерной технологии, использующей Windows NT. Несколько компаний предлагают кластеры на основе Unix-машин.

Наряду с ОС, ориентированными на совершенно определенный тип аппаратной платформы, существуют операционные системы, специально разработанные таким образом, чтобы они могли быть легко перенесены с компьютера одного типа на компьютер другого типа, так называемые мобильные ОС. Наиболее ярким примером такой ОС является популярная система Unix. В этих системах аппаратно-зависимые места тщательно локализованы, так что при переносе системы на новую платформу переписываются только они. Средством, облегчающим перенос остальной части ОС, является написание ее на машинно-независимом языке, например на C, который и был разработан для программирования операционных систем.

1.4.3. Особенности областей использования

Многозадачные ОС подразделяются на три типа в соответствии с использованными при их разработке критериями эффективности:

1) системы пакетной обработки (например, ОС ЕС);
2) системы разделения времени (Unix, VMS);
3) системы реального времени (QNX, RT/11).

Системы пакетной обработки предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является максимальная пропускная способность, т.е. решение максимального числа задач в единицу времени. Для достижения этой цели в системах пакетной обработки используются следующая схема функционирования: в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из этого пакета заданий формируется мультипрограммная смесь, т.е. множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие отличающиеся требования к ресурсам, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины; так, например, в мультипрограммной смеси желательно одновременное присутствие вычислительных задач и задач с интенсивным вводом-выводом. Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, т.е. выбирается "выгодное" задание. Следовательно, в таких ОС невозможно гарантировать выполнение того или иного задания в течение определенного периода времени. В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит только в случае, если активная задача сама отказывается от процессора, например, из-за необходимости выполнить операцию ввода-вывода. Поэтому одна задача может надолго занять процессор, что делает невозможным выполнение интерактивных задач. Таким образом, взаимодействие пользователя с вычислительной машиной, на которой установлена система пакетной обработки, сводится к тому, что он приносит задание, отдает его диспетчеру-оператору, а в конце дня после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок снижает эффективность работы пользователя.

Системы разделения времени призваны исправить основной недостаток систем пакетной обработки – изоляцию пользователя-програм-миста от процесса выполнения его задач. Каждому пользователю системы разделения времени предоставляется терминал, с которого он может вести диалог со своей программой. Так как в системах разделения времени каждой задаче выделяется только квант процессорного времени, ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым. Если квант выбран достаточно небольшим, то у всех пользователей, одновременно работающих на одной и той же машине, складывается впечатление, что каждый из них единолично использует машину. Ясно, что системы разделения времени обладают меньшей пропускной способностью, чем системы пакетной обработки, так как на выполнение принимается каждая запущенная пользователем задача, а не та, которая "выгодна" системе, и, кроме того, имеются накладные расходы вычислительной мощности на более частое переключение процессора с задачи на задачу. Критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.

Системы реального времени применяются для управления различными техническими объектами такими, например, как станок, спутник, научная экспериментальная установка или технологическими процессами такими, как гальваническая линия, доменный процесс и т.п. Во всех этих случаях существует предельно допустимое время, в течение которого должна быть выполнена та или иная программа, управляющая объектом, в противном случае может произойти авария: спутник выйдет из зоны видимости, экспериментальные данные, поступающие с датчиков, будут потеряны, толщина гальванического покрытия не будет соответствовать норме. Таким образом, критерием эффективности для систем реального времени является их способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия). Это время называется временем реакции системы, а соответствующее свойство системы – реактивностью. Для этих систем мультипрограммная смесь представляет собой фиксированный набор заранее разработанных программ, а выбор программы на выполнение осуществляется, исходя из текущего состояния объекта или в соответствии с расписанием плановых работ.

Некоторые операционные системы могут совмещать в себе свойства систем разных типов, например, часть задач может выполняться в режиме пакетной обработки, а часть – в режиме реального времени или в режиме разделения времени. В таких случаях режим пакетной обработки часто называют фоновым режимом.

1.4.4. Особенности методов построения

При описании операционной системы часто указываются особенности ее структурной организации и основные концепции, положенные в ее основу.

К таким базовым концепциям относятся. *Способы построения ядра системы* – монолитное ядро или микроядерный подход. Большинство ОС использует монолитное ядро, которое компонуется как одна программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую, не требующие переключения из привилегированного режима в пользовательский и наоборот. Альтернативой является построение ОС на базе микроядра, работающего также в привилегированном режиме и выполняющего только минимум функций по управлению аппаратурой, в то время как функции ОС более высокого уровня выполняют специализированные компоненты ОС – серверы, работающие в пользовательском режиме. При таком построении ОС работает более медленно, так как часто выполняются переходы между привилегированным режимом и пользовательским, зато система получается более гибкой – ее функции можно наращивать, модифицировать или сужать, добавляя, модифицируя или исключая серверы пользовательского режима. Кроме того, серверы хорошо защищены друг от друга, как и любые пользовательские процессы.

Построение ОС на базе объектно-ориентированного подхода дает возможность использовать все его достоинства, хорошо зарекомендовавшие себя на уровне приложений, внутри операционной системы, а именно аккумуляцию удачных решений в форме стандартных объектов, возможность создания новых объектов на базе имеющихся с помощью механизма наследования, хорошую защиту данных за счет их инкапсуляции во внутренние структуры объекта, что делает данные недоступными для несанкционированного использования извне, структурированность системы, состоящей из набора хорошо определенных объектов.

Наличие нескольких прикладных сред дает возможность в рамках одной ОС одновременно выполнять приложения, разработанные для нескольких ОС. Многие современные операционные системы поддерживают одновременно прикладные среды MS DOS, Windows, Unix (POSIX), OS/2 или хотя бы некоторого подмножества из этого популярного набора. Концепция множественных прикладных сред наиболее просто реализуется в ОС на базе микроядра, над которым работают различные серверы, часть которых реализуют прикладную среду той или иной операционной системы.

Распределенная организация операционной системы позволяет упростить работу пользователей и программистов в сетевых средах. В распределенной ОС реализованы механизмы, которые дают возможность пользователю представлять и воспринимать сеть в виде традиционного однопроцессорного компьютера. Характерными признаками распределенной организации ОС являются: наличие единой справочной службы разделяемых ресурсов, единой службы времени, использование механизма вызова удаленных процедур (RPC) для прозрачного распределения программных процедур по машинам, многонитевой обработки, позволяющей распараллеливать вычисления в рамках одной задачи и выполнять эту задачу сразу на нескольких компьютерах сети, а также наличие других распределенных служб.

Кроме того, операционные системы можно также разделить на группы, используя различные признаки классификации (см. табл. 1.1).

1.1. Классы операционных систем

Признак классификации	Описание класса
1. Мощность аппаратных средств	персональные компьютеры; мини ЭВМ; большие ЭВМ; суперЭВМ
2. Количество ЭВМ, обслуживаемых ОС	автономная ЭВМ; многомашинная ВС; сети ЭВМ (локальные и глобальные)
3. Тип системы обработки данных	система оперативной обработки; система пакетной обработки; система реального времени
4. Режим обработки данных	однопрограммный; пакетный мультипрограммный; разделения времени
5. Режим обслуживания заявок	одиночный отбор; групповой отбор; смешанный отбор
6. Дисциплина обслуживания заявок	без приоритетов; с приоритетами

1.5. ОСНОВНЫЕ ПРИНЦИПЫ ПОСТРОЕНИЯ ОПЕРАЦИОННЫХ СИСТЕМ

Частотный принцип реализации системных программ основан на выделении в алгоритмах и в обрабатываемых массивах ОС действий и данных по частоте их использования. Следствием применения частотного принципа в современных ОС – наличие многоуровневого планирования при организации работы ОС.

Принцип модульности отражает технологические и эксплуатационные свойства системы, предусматривая оформление функционально законченных компонентов ОС в виде отдельных модулей.

Принцип функциональной избирательности предусматривает выделение некоторого множества важных модулей, которые должны быть постоянно в "горячем" режиме для обеспечения эффективного управления вычислительным процессом. Этот выделенный набор модулей называют *ядром* ОС. При формировании состава ядра ОС ищут компромисс между двумя разноречивыми требованиями: в состав ядра должны войти наиболее часто используемые модули; объем памяти, занимаемый ядром ОС, должен быть как можно меньше. Программы ядра ОС постоянно находятся в оперативной памяти ЭВМ и называются *резидентными*. Программы ОС, подгружаемы в ОЗУ по мере необходимости из внешней памяти, называются *транзитными.*

Принцип генерируемости определяет такой способ исходного представления системной программы ОС, который позволяет настраивать эту системную программу, исходя из конкретной конфигурации аппаратных средств и круга решаемых проблем.

Принцип функциональной избыточности предусматривает обеспечение возможности выполнения одной и той же работы различными средствами.

Принцип перемещаемости предусматривает такое построение модулей ОС, при котором результаты работы не зависят от места их расположения.

Принцип защиты информации определяет необходимость разработки мер, ограждающих программы и данные пользователя от искажений или нежелательных влияний друг от друга, а также пользователей на ОС и обратно.

Принцип независимости программ от внешних устройств заключается в том, что связь программ с конкретными внешними устройствами осуществляется не на уровне подготовки программных устройств (трансляции или компиляции исходного кода, генерации выполняемого модуля), а в период планирования операционной системой ее выполнения.

Принцип открытости и наращиваемости ОС предусматривает возможность доступа к ней для анализа пользователями, специалистами, обслуживающим персоналом, а также изменения конфигурации ОС и ее мощности без осуществления процессов генерации.

1.6. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС ОПЕРАЦИОННЫХ СИСТЕМ

Как любое техническое устройство, компьютер обменивается информацией с человеком посредством набора определенных правил, обязательных как для машины, так и для человека. Эти правила в компьютерной литературе называются интерфейсом. Интерфейс может быть понятным и непонятным, дружественным и нет. К нему подходят многие прилагательные. Но в одном он постоянен: он есть, и никуда от него не денешься.

Интерфейс, по определению – это правила взаимодействия операционной системы с пользователями, а также соседних уровней в сети ЭВМ. От интерфейса зависит технология общения человека с компьютером.

1.6.1. Классификация интерфейсов

Как уже указывалось выше, интерфейс – это, прежде всего, набор правил. Как любые правила, их можно обобщить, собрать в "кодекс", сгруппировать по общему признаку. Таким образом, мы пришли к понятию "вид интерфейса" как объединение по схожести способов взаимодействия человека и компьютеров. Вкратце можно предложить следующую схематическую классификацию различных интерфейсов общения человека и компьютера (см. рис. 1.3).

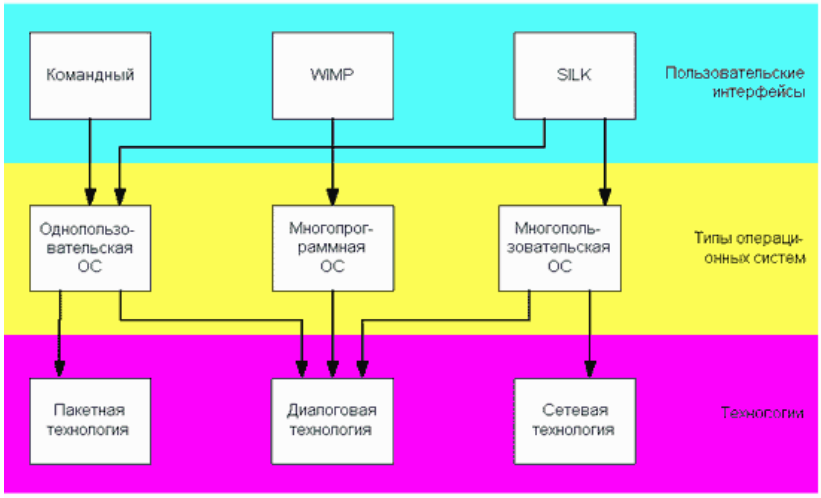

Рис. 1.3. Типы интерфейсов, операционных систем и технологий реализаций

Современными видами интерфейсов являются:

1. *Командный интерфейс* – называется так по тому, что в этом виде интерфейса человек подает "команды" компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки.
2. *WIMP-интерфейс* (Window – окно, Image – образ, Menu – меню, Pointer – указатель). Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов – меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается "опосредственно", через графические образы. Этот вид интерфейса реализован на двух уровнях технологий: простой графический интерфейс и "чистый" WIMP – интерфейс.
3. *SILK-интерфейс* (Speech – речь, Image – образ, Language – язык, Knowlege – знание). Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный "разговор" человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяют в основном для военных целей.
4. *Общественный интерфейс* – основан на семантических сетях.

1.6.2. Пакетная технология

Исторически вид пакетной технологии появился первым. Она существовала уже на релейных машинах Зюса и Цюзе (Германия, 1937 г.). Идея ее проста: на вход компьютера подается последовательность символов, в которых по определенным правилам указывается последовательность запущенных на выполнение программ. После выполнения очередной программы запускается следующая и т.д. Машина по определенным правилам находит для себя команды и данные. В качестве этой последовательности может выступать, например, перфолента, стопка перфокарт, последовательность нажатия клавиш электрической пишущей машинки (типа CONSUL). Машина также выдает свои сообщения на перфоратор, алфавитно-цифровое печатающее устройство (АЦПУ), ленту пишущей машинки.

Такая машина представляет собой "черный ящик" (точнее "белый шкаф"), в который постоянно подается информация и которая также постоянно "информирует" мир о своем состоянии. Человек здесь имеет малое влияние на работу машины – он может лишь приостановить работу машины, сменить программу и вновь запустить ЭВМ. Впоследствии, когда машины стали помощнее и могли обслуживать сразу нескольких пользователей, вечное ожидание пользователей типа: "Я послал данные машине. Жду, что она ответит. И ответит ли вообще?" – стало, мягко говоря, надоедать. К тому же вычислительные центры, вслед за газетами, стали вторым крупным "производителем" макулатуры. Поэтому с появлением алфавитно-цифровых дисплеев началась эра по-настоящему пользовательской технологии – командной строки.

1.6.3. Технология командной строки

При технологии командной строки в качестве единственного способа ввода информации от человека к компьютеру служит клавиатура, а компьютер выводит информацию человеку с помощью алфавитно-цифрового дисплея (монитора). Эту комбинацию (монитор + клавиатура) стали называть терминалом, или консолью.

Команды набираются в командной строке, которая представляет собой символ приглашения и мигающий прямоугольник – курсор. При нажатии клавиши на месте курсора появляются символы, а сам курсор смещается вправо. Это очень похоже на набор команды на пишущей машинке. Однако, в отличие от нее, буквы отображаются на дисплее, а не на бумаге, и неправильно набранный символ можно стереть. Команда заканчивается нажатием клавиши Enter (или Return.) После этого осуществляется переход в начало следующей строки. Именно с этой позиции компьютер выдает на монитор результаты своей работы. Затем процесс повторяется.

Технология командной строки уже работала на монохромных алфавитно-цифровых дисплеях. Поскольку вводить позволялось только буквы, цифры и знаки препинания, то технические характеристики дисплея были не существенны. В качестве монитора можно было использовать телевизионный приемник и даже трубку осциллографа.

Обе эти технологии реализуются в виде командного интерфейса – машине подаются на вход команды, а она как бы "отвечает" на них.

Преобладающим видом файлов при работе с командным интерфейсом стали текстовые файлы – их и только их можно было создать при помощи клавиатуры. На время наиболее широкого использования интерфейса командной строки приходится появление операционной системы Unix и появление первых восьмиразрядных персональных компьютеров с многоплатформенной операционной системой CP/M.

1.6.4. Графический интерфейс

Идея графического интерфейса зародилась в середине 70-х гг., когда в исследовательском центре Xerox Palo Alto Research Center (PARC) была разработана концепция визуального интерфейса. Предпосылкой графического интерфейса явилось уменьшение времени реакции компьютера на команду, увеличение объема оперативной памяти, а также развитие технической базы компьютеров. Аппаратным основанием концепции, конечно же, явилось появление алфавитно-цифровых дисплеев на компьютерах, причем на этих дисплеях уже имелись такие эффекты, как "мерцание" символов, инверсия цвета (смена начертания белых символов на черном фоне обратным, т.е. черных символов на белом фоне), подчеркивание символов. Эти эффекты распространились не на весь экран, а только на один или более символов. Следующим шагом явилось создание цветного дисплея, позволяющего выводить, вместе с этими эффектами, символы в 16 цветах на фоне с палитрой (т.е. цветовым набором) из 8 цветов. После появления графических дисплеев, с возможностью вывода любых графических изображений в виде множества точек на экране различного цвета, фантазии в использовании экрана вообще не стало границ! Первая система с графическим интерфейсом 8010 Star Information System группы PARC, таким образом, появилась за четыре месяца до выхода в свет первого компьютера фирмы IBM в 1981 г. Первоначально визуальный интерфейс использовался только в программах. Постепенно он стал переходить и на операционные системы, используемые сначала на компьютерах Atari и Apple Macintosh, а затем и на IBM-совместимых компьютерах.

С более раннего времени и под влиянием также и этих концепций проходил процесс по унификации в использовании клавиатуры и мыши прикладными программами. Слияние этих двух тенденций и привело к созданию того пользовательского интерфейса, с помощью которого, при минимальных затратах времени и средств на переучивание персонала, можно работать с любым программным продуктом. Описание этого интерфейса, общего для всех приложений и операционных систем, и посвящена данная часть.

Графический интерфейс пользователя за время своего развития прошел две стадии. Об эволюции графического интерфейса с 1974 г. по настоящее время будет рассказано ниже.

1.6.4.1. ПРОСТОЙ ГРАФИЧЕСКИЙ ИНТЕРФЕЙС

На первом этапе графический интерфейс очень походил на технологию командной строки. Отличия от технологии командной строки заключались в следующем:

1. При отображении символов допускалось выделение части символов цветом, инверсным изображением, подчеркиванием и мерцанием. Благодаря этому повысилась выразительность изображения.

2. В зависимости от конкретной реализации графического интерфейса курсор может представляться не только мерцающим прямоугольником, но и некоторой областью, охватывающей несколько символов и даже часть экрана. Эта выделенная область отличается от других, невыделенных частей (обычно цветом).

3. Нажатие клавиши Enter не всегда приводит к выполнению команды и переходу к следующей строке. Реакция на нажатие любой клавиши во многом зависит от того, в какой части экрана находится курсор.

4. Кроме клавиши Enter, на клавиатуре все чаще стали использоваться "серые" клавиши управления курсором.

5. Уже в этой редакции графического интерфейса стали использоваться манипуляторы (типа мыши, трекбола и т.п.). Они позволяли быстро выделять нужную часть экрана и перемещать курсор.

Подводя итоги, можно привести следующие отличительные особенности этого интерфейса:

1) выделение областей экрана;
2) переопределение клавиш клавиатуры в зависимости от контекста;
3) использование манипуляторов и серых клавиш клавиатуры для управления курсором;
4) широкое использование цветных мониторов.

Появление этого типа интерфейса совпадает с широким распространением операционной системы MS DOS. Именно она внедрила этот интерфейс в массы, благодаря чему 80-е годы прошли под знаком совершенствования этого типа интерфейса, улучшения характеристик отображения символов и других параметров монитора.

Типичным примером использования этого вида интерфейса является файловая оболочка Norton Commander (о файловых оболочках смотри ниже) и текстовый редактор Multi-Edit. А текстовые редакторы Лексикон, ChiWriter и текстовый процессор Microsoft Word for Dos являются примером, как этот интерфейс превзошел сам себя.

1.6.4.2. WIMP-ИНТЕРФЕЙС

Вторым этапом в развитии графического интерфейса стал "чистый" интерфейс WIMP. Этот подвид интерфейса характеризуется следующими особенностями:

1. Вся работа с программами, файлами и документами происходит в окнах – определенных очерченных рамкой частях экрана.

2. Все программы, файлы, документы, устройства и другие объекты представляются в виде значков – иконок. При открытии иконки превращаются в окна.

3. Все действия с объектами осуществляются с помощью меню. Хотя меню появилось на первом этапе становления графического интерфейса, оно не имело в нем главенствующего значения, а служило лишь дополнением к командной строке. В чистом WIMP – интерфейсе меню становится основным элементом управления.

4. Широкое использование манипуляторов для указания на объекты. Манипулятор перестает быть просто игрушкой – дополнением к клавиатуре, а становится основным элементом управления. С помощью манипулятора УКАЗЫВАЮТ на любую область экрана, окна или иконки, ВЫДЕЛЯЮТ ее, а уже потом через меню или с использованием других технологий осуществляют управление ими.

Следует отметить, что WIMP требует для своей реализации цветной растровый дисплей с высоким разрешением и манипулятор. Также программы, ориентированные на этот вид интерфейса, предъявляют повышенные требования к производительности компьютера, объему его памяти, пропускной способности шины и т.п. Однако этот вид интерфейса наиболее прост в усвоении и интуитивно понятен. Поэтому сейчас WIMP-интерфейс стал стандартом де-факто.

Ярким примером программ с графическим интерфейсом является операционная система Microsoft Windows.

1.6.5. Речевая технология

С середины 90-х гг. XX в., после появления недорогих звуковых карт и широкого распространения технологий распознавания речи, появилась так называемая "речевая технология" – SILK-интерфейс. При этой технологии команды подаются голосом путем произнесения специальных зарезервированных слов – команд, например:

- "Проснись" – включение голосового интерфейса;
- "Отдыхай" – выключение речевого интерфейса;
- "Открыть" – переход в режим вызова той или иной программы. Имя программы называется в следующем слове;
- "Буду диктовать" – переход из режима команд в режим набора текста голосом;
- "Режим команд" – возврат в режим подачи команд голосом и некоторые другие.

Слова должны выговариваться четко, в одном темпе. Между словами обязательна пауза. Из-за неразвитости алгоритма распознавания речи такие системы требует индивидуальной предварительной настройки на каждого конкретного пользователя.

"Речевая" технология является простейшей реализацией SILK-интерфейса.

1.6.6. Биометрическая технология

Биометрическая технология возникла в конце 90-х гг. XX в. и на момент написания книги еще разрабатывается. Для управления компьютером используется выражение лица человека, направление его взгляда, размер зрачка и другие признаки. Для идентификации пользователя используется рисунок радужной оболочки его глаз, отпечатки пальцев и другая уникальная информация. Изображения считываются с цифровой видеокамеры, а затем с помощью специальных программ распознавания образов из этого изображения выделяются команды. Эта технология, по-видимому, займет свое место в программных продуктах и приложениях, где важно точно идентифицировать пользователя компьютера.

1.6.7. Семантический интерфейс

Семантический интерфейс возник в конце 70-х гг. XX в., с развитием искусственного интеллекта. Его трудно назвать самостоятельным видом интерфейса – он включает в себя и интерфейс командной строки, и графический, и речевой, и мимический интерфейс. Основная его отличительная черта – это отсутствие команд при общении с компьютером. Запрос формируется на естественном языке, в виде связанного текста и образов. По своей сути это трудно называть интерфейсом – это уже моделирование "общения" человека с компьютером.

С середины 90-х гг. XX в. автор уже не встречал публикаций, относящихся к семантическому интерфейсу. Похоже, что в связи с важным военным значением этих разработок (например, для автономного ведения современного боя машинами-роботами, для "семантической" криптографии) эти направления были засекречены. Информация, что эти исследования продолжаются, иногда появляется в периодической печати (обычно в разделах компьютерных новостей).

Контрольные вопросы к теме 1

1. Дать определение и характеристику основных режимов работы, дисциплин и режимов обслуживания заявок в вычислительных системах.
2. Дать определение и характеристику классов программных средств.
3. Изложить классификацию ОС.
4. Охарактеризовать основные принципы построения ОС.
5. Перечислить виды интерфейсов ОС. Охарактеризовать пакетную технологию как интерфейс. Дать описание интерфейса командной строки.
6. Дать описание графических интерфейсов. В каких ОС они применяются?
7. Охарактеризовать речевую технологию как интерфейс.
8. Охарактеризовать биометрическую технологию как интерфейс.
9. Охарактеризовать семантический интерфейс.

Т Е М А 2. Концептуальные основы операционных систем

2.1. КОНЦЕПЦИЯ ПРОЦЕССА

Процесс – это система действий, реализующая определенную функцию в вычислительной системе и оформленная так, что управляющая программа вычислительной системы может перераспределять ресурсы этой системы в целях обеспечения мультипрограммирования.

Понятие процесса тесно связано с понятием задача.

Задача – в режиме мультипрограммирования или мультипроцессорной обработки одна или более последовательностей команд, обрабатываемых управляющей программой как элемент работы, которая выполняется вычислительной машиной.

Выполнение задачи реализуется в вычислительной системе запуском не менее одного процесса. Можно говорить, что задача – это один или несколько процессов, обеспечивающих достижение поставленных пользователем целей.

Следует отличать понятия процесс и задача от понятий программа и задание.

Программа (для ЭВМ) – упорядоченная последовательность команд, подлежащих обработке.

Задание (вычислительной системе) – единица работы, возлагаемой на вычислительную систему пользователем, оформленная для ввода в вычислительную систему независимо от других таких же единиц.

Отношение программы и задания аналогично отношению процесса и задачи, т.е. каждое задание содержит не менее одной программы, предназначенной для обработки в ЭВМ.

Об отношении процесса и программы можно сказать, что *процесс – это программа во время ее выполнения*. Всякая программа становится процессом, когда начинает выполняться в ЭВМ.

В период своего существования процесс может находиться в одном из следующих основных состояний (рис. 2.1):

- *порождение*, во время которого подготавливаются условия для первого исполнения на центральном процессоре;
- *активное* состояние (выполнение), когда процессу принадлежит центральный процессор;
- *ожидание*, во время которого процесс блокирован по причине занятости каких-либо необходимых ему ресурсов;
- *готовность*, при котором процесс получил все необходимые ему ресурсы, кроме центрального процессора;
- *окончание*, во время которого выполняются завершающие работу операции, после чего ресурсы процессу больше не предоставляются.

Возможно также представление переходов между состояниями в таблицы – так называемой матрицей смежностей графа (см. табл. 2.1).

Для построения средств управления процессами необходимо знать их свойства и классифицировать процессы в соответствии с этими свойствами (см. табл. 2.2).

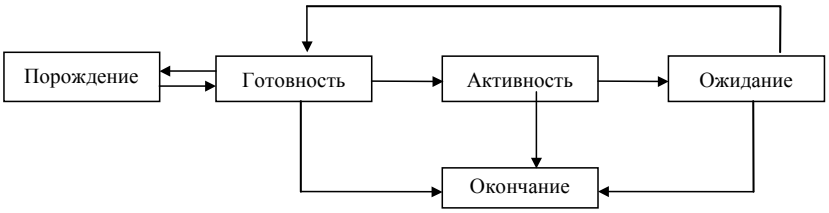

Рис. 2.1. Граф существования процесса

2.1. Матрица существования процесса

Состояние	*Порож-дение*	*Готовность*	*Активность*	*Ожидание*	*Окончание*
Порождение	0	1	0	0	0
Готовность	0	0	1	0	1
Активность	0	1	0	1	1
Ожидание	0	1	0	0	1
Окончание	0	0	0	0	0

2.2. Классификация процессов

Классификационный признак	Содержание классов
1. Время существования	А. Реального времени Б. Интерактивные В. Пакетные
2. Генеалогический признак	А. Порождающие Б. Порожденные
3. Принадлежность к ОС	А. Системные Б. Пользовательские
4. Принадлежность к ЦП	А. Внутренние Б. Внешние
5. Порядок выполнения	А. Последовательные Б. Параллельные В. Комбинированные
6. Наличие связи	А. Изолированные Б. Информационно-независимые В. Взаимодействующие ВГ. Конкурирующие
7. Результативность	А. Различные Б. Эквивалентные В. Тождественные Г. Равные

2.2. КОНЦЕПЦИЯ РЕСУРСА

Одной из важнейших задач, решаемых современными ОС, является обеспечение эффективного и бесконфликтного распределения ресурсов вычислительной системы между процессами.

Ресурс вычислительной системы – средство вычислительной системы, которое может быть выделено процессу обработки данных на определенный интервал времени.

Основными ресурсами вычислительной системы являются: процессоры, области основной памяти, наборы данных, периферийные устройства, программы.

Для осуществления планирования и распределения ресурсов ОС необходима информация об их свойствах, что достигается введением упорядочения ресурсов по ряду классификационных признаков. Классификация ресурсов ВС представлена в табл. 2.3 в направлении от наиболее абстрактных свойств в сторону более детального раскрытия в отношении реализации и использования этих свойств.

2.3. Классификация ресурсов

Классификационный признак	Содержание класса
1. Реальность существования	А. Физический Б. Виртуальный
2. Возможность расширения средств	А. Эластичный Б. Жесткий
3. Степень активности	А. Активный Б. Пассивный

4. Время существования	А. Постоянный Б. Временный
5. Степень важности	А. Главный Б. Второстепенный
6. Стоимость	А. Дорогой Б. Дешевый
7. Структура	А. Простой Б. Сложный
8. Восстанавливаемость	А. Воспроизводимый Б. Потребляемый
9. Характер использования	А. Параллельно-используемый Б. Последовательно-используемый
10. Форма реализации	А. Мягкий Б. Твердый

Признак "Реальность существования" ресурса разделяет ресурсы на *физические и виртуальные* (от англ. Virtual – возможный). Под физическим понимают ресурс, который реально существует и при распределении его между процессами в ВС обладает всеми присущими ему физическими характеристиками. Виртуальный ресурс – это некоторая модель физического ресурса.

Признак "Возможность расширения свойств" характеризует ресурс с точки зрения возможности построения на его основе некоторого виртуального ресурса. Физический ресурс, допускающий виртуализацию, т.е. размножение или расширение свойств, называют *эластичным.* В противном случае ресурс называется *жестким.*

Признак "Степень активности" отражает способность ресурса воздействовать на другие ресурсы ВС. Ресурс называется *активным*, если при его использовании он способен выполнять действия по отношению к другим ресурсам. В противном случае ресурс называется *пассивным*.

Различие ресурсов по признаку "Время существования" обусловлено динамикой ресурсов в отношении процессов, использующих их. Если ресурс существовал в системе до момента порождения процесса и доступен для использования на всем интервале времени существования процесса, то такой ресурс рассматривают как *постоянный* для данного процесса. *Временный* ресурс может появляться или уничтожаться в системе динамически в течение времени существования данного процесса.

Необходимость различать ресурсы по признаку "Степень важности" обусловлена двумя причинами: во-первых, необходимостью обеспечения должной работоспособности и, во-вторых, требованиями увеличения гибкости управления процессами и распределения ресурсов. Различают *главные* и *второстепенные* ресурсы. Ресурс является главным по отношению к конкретному процессу, если без его выделения процесс принципиально не может развиваться. Ресурсы, в отсутствие которых возможно некое альтернативное развитие процесса, называются второстепенными.

Разделение ресурсов по признаку "Стоимость" на *дорогие* и *дешевые* связано с реализацией принципа функциональной избыточности при распределении ресурсов.

Классификационным признаком "Структура" ресурсы разделяются на *простые* и *составные*. Ресурс является простым, если он, с точки зрения управляющей программы ВС, не содержит составных элементов и должен быть выделен процессу как единое целое. *Составной* ресурс характеризуется некоторой структурой, и при каждом акте распределения процесс может получить один или несколько составных частей такого ресурса. Простой и составной ресурсы различаются количеством своих состояний. Простой ресурс может находиться в одном из двух состояний – "занят" и "свободен". Составной ресурс имеет более двух состояний: "свободен" – все элементы ресурса свободны; "занят" – все элементы ресурса распределены процессом; "частично занят" – часть элементов ресурса распределена процессом, остальные свободны. Иногда говорят о доле занятости составного ресурса ("занято 20 %", "свободно 50 %" и т.п.).

Характер использования ресурсов устанавливается признаком "Восстанавливаемость". Если при распределении системой некоторого ресурса допускается многократное выполнение действий последовательности ЗАПРОС-ВЫДЕЛЕНИЕ-ИСПОЛЬЗОВАНИЕ-ОСВОБОЖДЕНИЕ (З-В-И-О), то такой ресурс называют *воспроизводимым*. Иногда такого вида ресурсы называют также постоянными, поскольку они всегда находятся в составе ресурсов ВС. В отношении определенной категории ресурсов многократное применение последовательности З-В-И-О невозможно, поскольку на каком-либо цикле работы с ними может наступить ситуация исчерпания ресурса, т.е. обрыв последовательности на шаге ИСПОЛЬЗОВАНИЕ, после чего такой ресурс изымается из использования. Ресурсы с таким свойством называются *потребляемыми*.

Природа ресурса отражается классификационным признаком "Характер использования". Ресурс называется *последовательно-используемым*, если он может использоваться одновременно только одним процессом. Если ресурс одновременно может быть выделен и использован несколькими параллельно работающими процессами, то он называется *параллельно-используемым* ресурсом. Для обозначения последовательно-используемого ресурса употребляют также термины "монопольно-используемый ресурс" и "критический ресурс". Параллельно-используемые ресурсы называют разделяемыми ресурсами. Заметим, что один и тот же ресурс может выступать как последовательно-используемый, так и параллельно-используемый.

По признаку "Форма реализации" различают *твердые* и *мягкие* ресурсы. Под твердыми ресурсами понимают аппаратные компоненты вычислительной системы (от англ. Hardware – твердое изделие, дословно – скобяной товар). Остальные ресурсы называют мягкими (от англ. Software – мягкое изделие).

2.3. КОНЦЕПЦИЯ ВИРТУАЛЬНОСТИ

Виртуальность в операционных системах, впрочем, как и в других предметных областях, основана на том, что потребитель (в ОС-процесс) взаимодействует с необходимым ему ресурсом не прямо, а опосредованно (рис. 2.2).

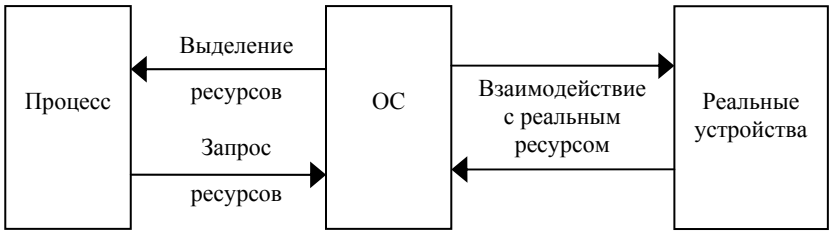

Рис. 2.2. Организация распределения ресурсов

Виртуальный ресурс вычислительной системы – это такой ресурс ВС, который реально может либо не существовать в вычислительной системе, либо существовать с худшими, чем необходимо процессу обработки данных, свойствами и формируемый управляющей программой ВС в виде модели ресурса с требуемыми потребительскими свойствами с использованием реально существующих ресурсов вычислительной системы.

Наиболее законченным и естественным проявлением концепции виртуальности является понятие виртуальной машины. Любая операционная система, будучи посредником между пользователем и аппаратной частью вычислительной системы, создает у пользователя определенное представление о вычислительной системе и ее ресурсах, т.е. формирует у пользователя видимость виртуальной машины. Восприятие характеристик виртуальной машины различными пользователями неоднозначно. Чаще всего виртуальная машина, предоставляемая пользователю, воспроизводит архитектуру реальной машины, однако архитектурные элементы этого представления выступают с новыми, зачастую улучшенными, качествами.

Чем больше машина, реализуемая средствами конкретной ОС на базе конкретной аппаратной части, приближена к "идеальной" по характеристикам машине, чем больше ее архитектурно-логические характеристики отличаются от реально существующих, тем больше уровень виртуальности получаемой пользователем виртуальной машины.

Концепция виртуальности машины нашла широкое применение при проектировании и реализации операционных систем, позволяя наиболее рационально представить структуру системы в виде определенного набора планировщиков и распределителей ресурсов.

2.4. КОНЦЕПЦИЯ ПРЕРЫВАНИЯ

Реализация мультипрограммного режима работы вычислительных систем возможна только на применении концепции прерываний, которая состоит в том, что любой процесс, обслуживаемый операционной системой, может быть прерван процессом, имеющим более высокий приоритет.

Прерывание – временное прекращение процесса, такого как выполнение программы вычислительной машины, вызванное событием, внешним по отношению к этому процессу, и совершенное таким образом, что процесс может быть продолжен (СТ ИСО 2382/10–79).

Приведенное определение исчерпывающе характеризует суть понятия прерывания, оставляя за рамками рассмотрения физическую природу аппаратного средства, где возникает это прерывание.

В вычислительной машине прерывание – это событие, при котором меняется нормальная последовательность команд, выполняемых процессором. Сигнал "прерывание" сначала отрабатывается аппаратурой вычислительной машины – системой прерываний.

Если произошло прерывание, то в вычислительной системе выполняются последовательно следующие действия (рис. 2.3):

- управление передается операционной системе;
- операционная система запоминает состояние прерванного процесса;
- операционная система анализирует тип прерывания и передает управление соответствующей программе обработки этого прерывания;
- программа обработки прерывания выполняет предписанные действия и передает управление операционной системе;
- операционная система по результатам работы программы обработки прерываний либо восстанавливает состояние прерванного процесса и позволяет развиваться ему дальше, либо аварийно заканчивает его.

Следует иметь в виду, что инициатором прерывания может быть также и выполняющийся процесс.

Количество источников сигналов прерывания достигает в современных вычислительных системах нескольких сотен и даже тысяч. Все возможные в системе прерывания можно классифицировать по месту (причине) их возникновения.

Различают шесть основных классов прерываний: прерывания от схем контроля ЭВМ; прерывания по рестарту (повторному пуску); прерывания ввода/вывода; внешние прерывания; прерывания по вызову супервизора; программные прерывания.

1. *Прерывание от схем контроля* возникает в случае появления любой аппаратной ошибки в ЭВМ. Продолжение работы машины становится невозможным, и процесс аварийно заканчивает свое существование.

2. *Прерывание по рестарту* может наступить в следующих случаях: на пульте управления была нажата кнопка (клавиша, сочетание клавиш) повторного пуска ЭВМ; процесс, выполняющий в данной ЭВМ, выдал команду рестарта; в многомашинной системе получена команда рестарта от другого компьютера. В любом случае в ЭВМ, получившей команду рестарта, выполняются действия по загрузке операционной системы.

Рис. 2.3. Циклограмма действий при возникновении прерывания

3. *Прерывания ввода/вывода* инициируются аппаратурой, обеспечивающей операции ввода и вывода данных. Они сигнализируют центральному процессору об изменении состояния устройств ввода/вывода.

4. *Внешнее прерывание* может возникнуть по самым различным причинам. Типичными источниками внешних прерываний являются таймер, выдающий сигнал об истечении кванта времени, параллельный синхронный или асинхронный процесс, другой процессор, другой компьютер.

5. *Прерывание по вызову супервизора* появляется в том случае, когда работающий процесс выполняет команду обращения к супервизору. Этой командой (SVC-командой) программа пользователя генерирует запрос на предоставление конкретной системной услуги, например, на выполнение операции ввода/вывода, увеличение объема выделенной памяти и т.п. Механизм SVC-команд позволяет защитить операционную систему от пользовательских процессов.

6. *Программное прерывание* может возникнуть по двум причинам: процесс пытается выполнить ошибочную операцию (например, деление на нуль, операция с неправильным кодом и т.п.); процесс выполнил заранее подготовленную команду прерывания для обеспечения перехода к выполнению других действий (например, для синхронизации нескольких процессов в вычислительной системе).

Для обработки каждого из типов прерываний в составе операционной системы предусмотрены специальные *программы обработки прерываний* (или обработчики прерываний, Interrupt Handler – IH). В ОС имеется шесть основных обработчиков прерываний (по количеству классов прерываний). Когда происходит прерывание, операционная система запоминает состояние прерванного процесса и передает управление соответствующему обработчику прерываний. Это делается способом, получившим название *переключение контекста.* Суть этого способа состоит в следующем. С каждым прерыванием связывают слова, называемые словами состояния программы ССП (или PSW от англ. Programs State Word), которые управляют порядком выполнения команд и содержат различную информацию относительно состояния процесса. Различают три типа слов состояния программы: 1) текущее ССП; 2) новое ССП; 3) старое ССП.

Центральный процессор реагирует только на разрешенные прерывания. Обработка запрещенных прерываний либо *задерживается,* либо игнорируется. Процессору нельзя запретить реагировать на прерывания по вызову супервизора, по рестарту и на некоторые виды программных прерываний. Количество ССП совпадает с количеством процессоров в мультипроцессорный ВС. Количество старых ССП и новых ССП совпадают с количеством типов прерываний.

Текущее ССП содержит информацию о выполняющемся процессе и постоянно обновляется по мере развития процесса. Новое ССП содержит информацию о маске прерываний и постоянный адрес обработчика данного типа прерываний.

Когда происходит прерывание, то (если процессору не запрещено обрабатывать прерывание данного типа) производится автоматическое, выполняемое аппаратурой, переключение слов состояния программы в следующей последовательности:

1) текущее ССП становится старым ССП для прерывания этого типа;
2) новое ССП для прерывания этого типа становится текущим ССП.

После такого замещения текущее ССП содержит адрес соответствующего обработчика прерываний, который и начинает выполняться.

Когда обработка прерывания окончена, то происходит обратное переключение слов состояния программы, при котором старое ССП для прерывания данного типа становится текущим ССП, после чего прерванный процесс продолжает свою работу. Существует большое количество различных схем обработки прерываний, соответствующих разным режимам и дисциплинам обслуживания. С ними можно ознакомиться в литературе по архитектуре вычислительных машин и систем.

2.5. ПОНЯТИЕ ЯДРА И МИКРОЯДРА ОС

2.5.1. Понятие ядра ОС

Понятие ядра ОС непосредственно вытекает из принципов построения ОС, конкретно – из принципа избирательности.

Принцип функциональной избирательности предусматривает выделение некоторого множества важных модулей, которые должны быть постоянно в "горячем" режиме для обеспечения эффективного управления вычислительным процессом. Этот выделенный набор модулей называют *ядром* ОС. При формировании состава ядра ОС ищут компромисс между двумя разноречивыми требованиями: в состав ядра должны войти наиболее часто используемые модули; объем памяти, занимаемый ядром ОС, должен быть как можно меньше. Программы ядра ОС постоянно находятся в оперативной памяти ЭВМ и называются *резидентными. Программы* ОС, подгружаемые в ОЗУ по мере необходимости из внешней памяти, называются *транзитными.*

2.5.2. Понятие микроядра ОС

Ядро любой современной ОС представляет собой набор очень большого количества функций, с запутанными взаимосвязями и очень расплывчатыми границами между основными подсистемами. В результате любая модификация организованной таким образом системы дается тяжело и приводит к появлению в новых версиях большого количества ошибок. Кроме того, не во всех инсталляциях нужны все компоненты ядра, а при монолитном его построении удаление ненужных функций затруднено. Недостатки, присущие операционным системам с большим монолитным ядром (а это, в первую очередь, различные версии Unix), породили интерес к системам, построенным на основе микроядра.

Микроядерный подход заключается в том, что базовые функции ядра оформляются в виде отдельной небольшой компоненты, выполняемой в привилегированном режиме, а остальные функции ОС выполняются в пользовательском режиме с использованием примитивов микроядра. Ввиду больших потенциальных преимуществ, которые сулит этот подход, можно предположить, что в ближайшее время большинство новых операционных систем будет строиться на основе микроядра. Наиболее известными реализациями этого подхода являются микроядра Mach и Chorus.

Основной сложностью использования микроядерного подхода на практике является замедление скорости выполнения системных вызовов при передаче сообщений через микроядро по сравнению с классическим подходом.

Микроядро по определению содержит базовые механизмы, имеющиеся внутри любой операционной системы, поэтому знакомство с этими механизмами в чистом виде полезно и для изучения любой конкретной ОС.

Микроядра лицензируются и используются как готовый программный продукт в качестве основы для построения коммерческой сетевой операционной системы. Сейчас имеется несколько коммерческих реализаций операционных систем на основе, например, микроядра Mach (NextStep фирмы Next, Unix BSD, OSF/1 v.1.3).

Контрольные вопросы к теме 2

1. Дать определение понятия процесса. Зачем оно требуется?
2. Дать определение понятия прерывания. Зачем оно требуется?
3. Дать определение понятия виртуальности. Зачем оно требуется?
4. Дать определение понятия ресурса. Зачем оно требуется?
5. Охарактеризуйте понятие "ядро ОС"?
6. Охарактеризуйте понятие "микроядро ОС"?

Т Е М А 3. Понятие управления задачами

3.1. ОРГАНИЗАЦИЯ УПРАВЛЕНИЯ ЗАДАЧАМИ

Теория и практика создания операционных систем, концентрированно выраженных в принципах и концепциях построения ОС (см. темы 1 и 2), позволили в качестве основы применять *двухуровневую схему управления* прохождением пакетов заданий в вычислительной системе. Применение не менее двух уровней в схеме управления опирается на использование частотного принципа при построении ОС. Различают уровни долгосрочного (внешнего) и краткосрочного (внутреннего) планирования.

Уровень внешнего планирования выполняет действия, относительно редкие в системе, но требующие больших системных затрат.

Уровень внутреннего планирования выполняет частые и более короткие действия по управлению процессами.

Объектами управления уровня долгосрочного планирования являются задачи и их объединения – работы.

Работа – совокупность задач обработки данных, объединенных для достижения заданных целей обработки. Каждая работа является независимой единицей обработки данных и связана с исполнением одной или нескольких программ (пользовательских и/или системных) на одном или нескольких процессорах для достижения определенного результата.

Объектом управления на уровне краткосрочного планирования являются процессы, которые выступают как конкурирующие друг с другом потребители ресурсов вычислительной системы.

Пакет заданий представляет собой заявку пользователя на выполнение вычислительной системой обработки данных и состоит из следующих компонентов:

- данные для идентификации пользователя;
- общая управляющая информацию пакета заданий (максимально необходимый объем ОЗУ, приоритет, вид выдачи сообщений ОС и т.п.);
- одно или несколько заданий на обработку.

Каждое задание на обработку, будучи компонентом заявки пользователя, содержит данные, передаваемые операционной системе:

- программы заданий (или их имена) и параметры запуска программ;
- управляющую информацию задания;
- требования на аппаратные и информационные ресурсы (ОЗУ, внешние устройства, библиотеки программ и т.д.);
- исходные данные обработки.

Обслуживание заявок на обработку данных в соответствии с требованиями пользователей осуществляется операционной системой с привлечением различных программно-аппаратных средств.

3.2. СРЕДСТВА И МЕХАНИЗМЫ УПРАВЛЕНИЯ ЗАДАЧАМИ

3.2.1. Средства управления задачами на уровне внешнего планирования

Информация, передаваемая операционной системе пакетом заданий, записывается по вполне определенным правилам. Эти правила представлены в операционных системах описаниями языков, называемых либо командными языками, либо языками управления заданиями, которые появились одновременно с мультипрограммными ОС и предназначены только для описания заданий. Примерами командных языков являются язык пакетных файлов в MS DOS и JCL в ОС ЕС (JCL – Job Control Language, язык управления работами). Командные языки универсальны в рамках одной архитектуры, т.е. могут быть применены любым пользователем в любой модели ЭВМ данной архитектуры. Обработка JCL-операторов выполняется в ОС с помощью "языковых процессоров управления заданиями" (JCL-процессоров). Существуют два принципа реализации JCL-процессоров: компиляционный и интерпретационный.

Компиляционный принцип построения JCL-процессора предусматривает наличие в составе ОС специальной программы-компилятора с языка управления заданиями. Все операторы пакета заданий, введенного в ЭВМ, сначала компилируются (т.е. переводятся в некоторое внутреннее представление), и только затем используются операционной системой для целей управления процессами обработки данных. Такая схема обработки данных применена в ОС ЕС ЭВМ.

Интерпретационный принцип требует наличия программы-оболочки (Shell-программы), которая рассматривается в системе как интерактивная программа-утилита, расположенная между пользователем и ядром системы. Shell-программа воспринимает операторы командного языка, интерпретирует их и обеспечивает выполнение пользовательских программ. По этой схеме построен JCL-процессор ОС Unix.

Рассмотрим одну из распространенных схем долгосрочного уровня планирования, в которой нашли отражение практически все необходимые средства для управления заданиями на этом уровне: системный ввод, инициатор, терминатор, системный вывод.

В функции системного ввода входят:

- считывание управляющих операторов пакетов заданий с устройств системного ввода, их анализ и формирование управляющих таблиц;
- создание входных очередей работ и задач в соответствии с принятыми в ОС дисциплинами и режимами обслуживания заявок;
- запись входных данных и программ каждого задания в память прямого доступа (магнитные диски);
- создание основы выходных очередей для выходных наборов данных и системных сообщений.

Программы системного ввода формируют управляющие таблицы. Совокупность управляющих таблиц образует базу данных управляющей информации, необходимой для осуществления процессов долгосрочного и краткосрочного планирования в ОС. Связь различных таблиц этой базы осуществляется с помощью ключевых полей, в которые заносятся ссылки (адреса) на записи в этой же или другой таблице.

Следующим этапом работы системного ввода является создание входной очереди. Сформированная очередь работ записывается на устройство прямого доступа (магнитный диск). Одновременно с входной очередью осуществляется подготовка и выходной очереди для выходных наборов данных и системных сообщений. По окончании своей работы системный ввод передает управление инициатору.

Перечень действий, выполняемых инициатором, особенности программной реализации, способы расположения этого системного компонента зависят от режима мультипрограммирования, реализуемого в операционной системе. Основной функцией инициализатора является подготовка к исполнению (фаза инициализации) работы и задач. Для реализации этой функции инициализатор выполняет основные виды действий:

1) распределение необходимых для исполнения работы и задач ресурсов системы;
2) выполнение дополнительных действий, связанных с порождением процессов.

При выполнении действий первого вида используются управляющие таблицы, сформированные системным вводом. На данном этапе осуществляется:

- распределение внешних устройств:
- проверка доступности необходимых наборов данных;
- выделение требуемой оперативной и внешней памяти.

Совокупность действий второго вида существенно различается для разных операционных систем. Завершающей операцией является создание первого (главного) процесса очередной задачи, после чего управление передается системе управления процессами.

После окончания выполнения задачи начинает работать терминатор, который реализует основные функции:

- освобождение всех ресурсов, ранее закрепленных за задачей (или за работой, если оканчивающаяся задача является последней в работе);

– выполнение действий по уничтожению задачи (или работы) и коррекция входных очередей;

– выдача разрешения системному выводу на вывод результатов и системных сообщений;

– подготовка условий для выведения инициатора из состояния ожидания для последующей инициализации следующего задания обрабатываемого пакета.

Функционирование терминатора может происходить во взаимодействии с оператором ЭВМ, сообщая ему о необходимости подготовки тех или иных внешних устройств, возникающих конфликтных ситуациях и т.п.

Системный вывод обеспечивает вывод выходных наборов данных и системных сообщений. Наборы выходных данных сначала записываются на устройствах прямого доступа, образуя выходные очереди в соответствии с заготовками, сформированными при работе системного ввода.

3.2.2. Средства управления задачами на уровне внутреннего планирования

По окончании своей работы инициатор передает управление системе управления процессами, называемую также *супервизором* (*менеджером*) *процессов*, которая отвечает за распределение ресурсов и координирует развитие всех процессов, находящихся в системе.

Управление процессами осуществляется с помощью внутреннего планирования, целью которого является организация оптимальной обработки задач в рабочей смеси.

Рабочей смесью называется совокупность задач, одновременно находящихся на обслуживании операционной системой.

С каждой задачей в ОС связана одна или несколько программ обработки данных, оформленных в виде загрузочных модулей, при выполнении которых в системе порождается один или несколько процессов. Порождение процессов осуществляется, как правило, с учетом внутренней структуры загрузочных модулей и способа их использования.

Загрузочные (выполняемые) модули прикладных программ подготавливаются с применением системных обрабатывающих программ (компиляторов, редакторов связей и т.п.).

По *структуре* различают простые, оверлейные и динамические загрузочные модули.

- *Модуль простой структуры* – загружается в ОЗУ как единое целое, выполняется как единое целое, выполняется и после своего завершения сразу освобождает память.
- *Модуль оверлейной структуры* – создается редактором связей как загрузочный модуль, где определены сегменты кода, которые не обязательно должны одновременно находиться в ОЗУ во время выполнения программы.
- *Модуль динамической структуры* – может загружаться в ОЗУ полностью или частично. Он может иметь переходы к другим загрузочным модулям. Различают модули с динамической последовательной и с динамической параллельной структурами.

По *способу использования* модули разделяют на *однократно используемые, повторно* (*многократно*) *используемые и реентерабельные* (от англ. *re-enter* – многовходовый). Реентерабельные модули сейчас называют также разделяемыми (от англ. *share* – разделять).

Представителем процесса в операционной системе является *дескриптор процесса*. Это структура данных, содержащая определенную важную информацию о процессе, в том числе: текущее состояние процесса, уникальный идентификатор процесса, приоритет процесса и т.п.

Системы управления процессами должны иметь возможность выполнять над процессами следующие основные операции:

1) создание (порождение, образование) процесса;
2) уничтожение процесса;
3) приостановка процесса;
4) возобновление процесса;
5) изменение приоритета процесса.

Реализация системы краткосрочного планирования представлена по разному в различных ОС. Основными компонентами краткосрочного планировщика являются: диспетчер процессов, супервизор процессов, супервизор памяти, супервизор таймера.

- *диспетчер процессов* – для планирования использования центрального процессора вычислительной системы, работает совместно с системой прерывания и обслуживает две очереди – очередь готовности и очередь ожидания;
- *супервизор процессов* – осуществляет функции по созданию, уничтожению и переводу процессов из состояния в состояние. Кроме того, он поддерживает механизмы синхронизации параллельных процессов и механизмы статического и динамического назначения приоритетов процессам, учитываемых при распределении главного ресурса системы – центрального процессора;
- *супервизор памяти* – выполняет распределение оперативной памяти между процессами в соответствии с принятой в данной ОС дисциплиной;
- *супервизор таймера* – обеспечивает управление интервальным таймером системы, исходя из требований параллельных процессов, обслуживаемых системой.

В некоторых ОС в отдельную подсистему выделяют супервизор ресурсов, задачей которого является распределение ресурсов (в первую очередь, внешних устройств) между процессами.

3.3. АЛГОРИТМЫ УПРАВЛЕНИЯ ЗАДАЧАМИ

3.3.1. Алгоритмы управления задачами на уровне внешнего планирования

Все алгоритмы внешнего планирования, применяемые в ОС, имеют эвристический характер. При разработке этих алгоритмов стараются улучшить качество планирования, показатели которого зависят от режимов использования вычислительной системы. Планирование может осуществляться с использованием любых дисциплин и режимов обслуживания заявок (с учетом приоритетов и без их учета, с неизменными и динамическими приоритетами и т.п.).

В системах с разделением времени внешнее планирование осуществляется согласно одного из двух алгоритмов: алгоритм бесприоритетного планирования; алгоритм приоритетного планирования.

Алгоритм бесприоритетного планирования предусматривает постановку заданий в очередь в порядке их поступления в систему. Выбор заданий на обработку из очереди может быть организован либо по принципу FIFO, либо по принципу LIFO.

Алгоритм приоритетного планирования предусматривает занесение заданий в очередь на обслуживание в соответствии с их приоритетами. Выборка заданий из очереди осуществляется из непустой очереди с наивысшим приоритетом.

В системах с пакетной мультипрограммной обработкой внешнее планирование осуществляется с применением одного из следующих критериев:

- минимум простое оборудование;
- максимум пропускной способности вычислительной системы;
- минимум времени реакции системы на задания пользователей.

Внешнее планирование в режиме ПМП может выполняться, например, по следующим алгоритмам:

- алгоритм минимизации простоя оборудования;
- алгоритм упорядочивания по требованиям на ресурсы;
- алгоритм упорядочивания по требованиям на загрузку центрального процессора (ЦП).

Алгоритм минимизации простоя оборудования включает в рабочую смесь только те задачи, которые обеспечивают равномерную нагрузку на все устройства системы. Рассмотрим минимизацию времени простоя ЦП и внешних устройств (ВУ).

Пусть N – количество задач в рабочей смеси. Каждому i-му заданию поставим в соответствие пару чисел: $T_{\text{ЦП}\,i}$ – запрос пользователя на время выполнения задания на ЦП; $T_{\text{ВУ}\,i}$ – запрос пользователя на длительность загрузки внешнего устройства. Пусть заданы пороговые интервалы времени $T_{j\,\text{п}}$ обслуживания заданий j-м ресурсом. Тогда i-е задание является *значимым*, относительно j-го ресурса, если длительность времени T_{ji} его обслуживания этим ресурсом больше порогового, т.е. $T_{ji} > T_{j\,\text{п}}$.

Введем следующие обозначения:

$N_{\text{о. ЦП}}$, $N_{\text{о. ВУ}}$ – оптимальное количество в рабочей смеси задач, значимых относительно ЦП и ВУ, соответственно;

$N_{\text{ф. ЦП}}$, $N_{\text{ф. ВУ}}$ – фактическое количество в рабочей смеси задач, значимых, соответственно, относительно ЦП и ВУ.

Тогда алгоритм минимизации простоя оборудования для случая ЦП и ВУ может быть представлен двумя информационно связанными процедурами:

- процедура формирования очередей;
- процедура включения задачи в рабочую смесь.

Процедура формирования очередей обеспечивает распределение всех поступающих в систему заданий в одну из следующих четырех очередей:

z_0 – для которых $T_{\text{ЦП}\,i} < T_{\text{ЦП п}}$ и $T_{\text{ВУ}\,i} < T_{\text{ВУ п}}$;

z_1 – для которых $T_{\text{ЦП}\,i} \geq T_{\text{ЦП п}}$ и $T_{\text{ВУ}\,i} < T_{\text{ВУ п}}$;

z_2 – для которых $T_{\text{ЦП}\,i} < T_{\text{ЦП п}}$ и $T_{\text{ВУ}\,i} \geq T_{\text{ВУ п}}$;

z_3 – для которых $T_{\text{ЦП}\,i} \geq T_{\text{ЦП п}}$ и $T_{\text{ВУ}\,i} \geq T_{\text{ВУ п}}$.

Процедура включения задачи в рабочую смесь обеспечивает выборку задания из очередей z_0, ..., z_3, сформированных процедурой формирования очередей в соответствии со следующими правилами:

- если $N_{\text{ф. ЦП}} < N_{\text{о. ЦП}}$ и $N_{\text{ф. ВУ}} < N_{\text{о. ВУ}}$, то задание выбирается из очередей z_1, z_2 или z_3 в соответствии с алгоритмом выбора из очередей z_1, z_2, z_3;
- если $N_{\text{ф. ЦП}} < N_{\text{о. ЦП}}$ и $N_{\text{ф. ВУ}} \geq N_{\text{о. ВУ}}$, то задание выбирается из очереди z_1, а при ее истощении – из z_0;
- если $N_{\text{ф. ЦП}} \geq N_{\text{о. ЦП}}$ и $N_{\text{ф. ВУ}} < N_{\text{о. ВУ}}$, то задание выбирается из очереди z_2, а при ее истощении – из очереди z_0;
- если $N_{\text{ф. ЦП}} \geq N_{\text{о. ЦП}}$ и $N_{\text{ф. ВУ}} \geq N_{\text{о. ВУ}}$, то задание выбирается из очереди z_0, а при ее истощении – из очередей z_1, z_2 или z_3 по алгоритму выбора из очередей z_1, z_2, z_3.

Алгоритм выбора из очередей z_1, z_2, z_3 должен обеспечить бессбойную выборку заданий из указанных очередей. Здесь возможно применение различных подходов, например, кольцевая процедура обхода очередей.

Алгоритм упорядочивания по требованиям на ресурсы разбивает задания на очереди в зависимости от объемов требуемых ими ресурсов.

Алгоритм упорядочивания по требованиям на загрузку ЦП для каждого i-го задания устанавливает приоритет (p_i) в зависимости от требуемого времени центрального процессора $T_{\text{ЦП}\,i}$.

Несмотря на то, что рассмотренные три алгоритма внешнего планирования достаточно легко реализуемы в ОС, их применение может привести к недопустимо большой задержке обслуживания бесприоритетных или "неудобных" заданий.

3.3.2. Алгоритмы управления задачами на уровне внутреннего планирования

3.3.2.1. МУЛЬТИЗАДАЧНОСТЬ, ПРОЦЕССЫ И НИТИ

Мультизадачность. В современном мультипрограммировании различают два типа мультизадачности: кооперативная и вытесняющая.

Кооперативная мультизадачность – это такой режим работы ОС, когда активный процесс, которому ОС выделило центральный процессор, монопольно использует его до тех пор, пока ему не потребуется выполнить какие-либо операции с внешними устройствами.

Вытесняющая мультизадачность – это такой режим работы ОС, когда операционная система в любой момент времени может приостановить развитие активного процесса, сохранив его состояние во внешней памяти ("вытеснить" процесс), и активизировать вместо него другой процесс.

Процессы. Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами. *Процесс* (или по-другому, задача) – абстракция, описывающая выполняющуюся программу. Для операционной системы процесс представляет собой единицу работы, заявку на потребление системных ресурсов. Подсистема управления процессами планирует выполнение процессов, т.е. распределяет процессорное время между несколькими одновременно существующими в системе процессами, а также занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает взаимодействие между процессами.

Контекст и дескриптор процесса. На протяжении существования процесса его выполнение может быть многократно прервано и продолжено. Для того чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды отображается состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок выполняемых данным процессом системных вызовов и т.д. Эта информация называется *контекстом процесса*.

Кроме этого, операционной системе для реализации планирования процессов требуется дополнительная информация: идентификатор процесса, состояние процесса, данные о степени привилегированности процесса, место нахождения кодового сегмента и другая информация. В некоторых ОС (например в ОС Unix) информацию такого рода, используемую ОС для планирования процессов, называют *дескриптором процесса*.

Дескриптор процесса по сравнению с контекстом содержит более оперативную информацию, которая должна быть легко доступна подсистеме планирования процессов. Контекст процесса содержит менее актуальную информацию и используется операционной системой только после того, как принято решение о возобновлении прерванного процесса.

Очереди процессов представляют собой дескрипторы отдельных процессов, объединенные в списки. Таким образом, каждый дескриптор, кроме всего прочего, содержит, по крайней мере, один указатель на другой дескриптор, соседствующий с ним в очереди. Такая организация очередей позволяет легко их переупорядочивать, включать и исключать процессы, переводить процессы из одного состояния в другое.

Программный код только тогда начнет выполняться, когда для него операционной системой будет создан процесс. Создать процесс – это значит:

- создать информационные структуры, описывающие данный процесс, т.е. его дескриптор и контекст;
- включить дескриптор нового процесса в очередь готовых процессов;
- загрузить кодовый сегмент процесса в оперативную память или в область свопинга.

Нити. Многозадачность является важнейшим свойством ОС. Для поддержки этого свойства ОС определяет и оформляет для себя те внутренние единицы работы, между которыми и будет разделяться процессор и другие ресурсы компьютера. Эти внутренние единицы работы в разных ОС носят разные названия: задача, задание, процесс, нить. В некоторых случаях сущности, обозначаемые этими понятиями, принципиально отличаются друг от друга.

Говоря о процессах, мы отмечали, что операционная система поддерживает их обособленность: у каждого процесса имеется свое виртуальное адресное пространство, каждому процессу назначаются свои ресурсы – файлы, окна, семафоры и т.д. Такая обособленность нужна для того, чтобы защитить один процесс от другого, поскольку они, совместно используя все ресурсы машины, конкурируют с друг другом. В общем случае процессы принадлежат разным пользователям, разделяющим один компьютер, и ОС берет на себя роль арбитра в спорах процессов за ресурсы.

При мультипрограммировании повышается пропускная способность системы, но отдельный процесс никогда не может быть выполнен быстрее, чем если бы он выполнялся в однопрограммном режиме (всякое разделение ресурсов замедляет работу одного из участников за счет дополнительных затрат времени на ожидание освобождения ресурса). Однако задача, решаемая в рамках одного процесса, может обладать внутренним параллелизмом, который в принципе позволяет ускорить ее решение. Например, в ходе выполнения задачи происходит обращение к внешнему устройству, и на время этой операции можно не блокировать полностью выполнение процесса, а продолжить вычисления по другой "ветви" процесса.

Для этих целей современные ОС предлагают использовать сравнительно новый механизм *многонитевой обработки* (*multithreading*). При этом вводится новое понятие "нить" (thread), а понятие "процесс" в значительной степени меняет смысл.

Мультипрограммирование теперь реализуется на уровне нитей, и задача, оформленная в виде нескольких нитей в рамках одного процесса, может быть выполнена быстрее за счет псевдопараллельного (или параллельного в мультипроцессорной системе) выполнения ее отдельных частей. Например, если электронная таблица была разработана с учетом возможностей многонитевой обработки, то пользователь может запросить пересчет своего рабочего листа и одновременно продолжать заполнять таблицу. Особенно эффективно можно использовать многонитевость для выполнения распределенных приложений, например, многонитевый сервер может параллельно выполнять запросы сразу нескольких клиентов.

Нити, относящиеся к одному процессу, не настолько изолированы друг от друга, как процессы в традиционной многозадачной системе, между ними легко организовать тесное взаимодействие. Действительно, в отличие от процессов, которые принадлежат разным, вообще говоря, конкурирующим приложениям, все нити одного процесса всегда принадлежат одному приложению, поэтому программист, пишущий это приложение, может заранее продумать работу множества нитей процесса таким образом, чтобы они могли взаимодействовать, а не бороться за ресурсы.

В традиционных ОС понятие "нить" тождественно понятию "процесс". В действительности часто бывает желательно иметь несколько нитей, разделяющих единое адресное пространство, но выполняющихся квазипараллельно, благодаря чему нити становятся подобными процессам (за исключением разделяемого адресного пространства).

Нити иногда называют облегченными процессами или мини-процессами. Действительно, нити во многих отношениях подобны процессам. Каждая нить выполняется строго последовательно и имеет свой собственный программный счетчик и стек. Нити, как и процессы, могут, например, порождать нити-потомки, могут переходить из состояния в состояние. Подобно традиционным процессам (т.е. процессам, состоящим из одной нити), нити могут находится в одном из следующих со-

стояний: *выполнение*, *ожидание* и *готовность*. Пока одна нить заблокирована, другая нить того же процесса может выполняться. Нити разделяют процессор так, как это делают процессы, в соответствии с различными вариантами планирования.

Однако различные нити в рамках одного процесса не настолько независимы, как отдельные процессы. Все такие нити имеют одно и то же адресное пространство. Это означает, что они разделяют одни и те же глобальные переменные. Поскольку каждая нить может иметь доступ к каждому виртуальному адресу, одна нить может использовать стек другой нити. Между нитями нет полной защиты, потому что, во-первых, это невозможно, а во-вторых, не нужно. Все нити одного процесса всегда решают общую задачу одного пользователя, и аппарат нитей используется здесь для более быстрого решения задачи путем ее распараллеливания. При этом программисту очень важно получить в свое распоряжение удобные средства организации взаимодействия частей одной задачи. Кроме разделения адресного пространства, все нити разделяют также набор открытых файлов, таймеров, сигналов и т.п.

Итак, нити имеют собственные:

- программный счетчик;
- стек;
- регистры;
- нити-потомки;
- состояние.

Нити разделяют:

- адресное пространство;
- глобальные переменные;
- открытые файлы;
- таймеры;
- семафоры;
- статистическую информацию.

Многонитевая обработка повышает эффективность работы системы по сравнению с многозадачной обработкой. Например, в многозадачной среде Windows можно одновременно работать с электронной таблицей и текстовым редактором. Однако, если пользователь запрашивает пересчет своего рабочего листа, электронная таблица блокируется до тех пор, пока эта операция не завершится, что может потребовать значительного времени. В многонитевой среде в случае, если электронная таблица была разработана с учетом возможностей многонитевой обработки, предоставляемых программисту, этой проблемы не возникает, и пользователь всегда имеет доступ к электронной таблице.

Широкое применение находит многонитевая обработка в распределенных системах.

Некоторые прикладные задачи легче программировать, используя параллелизм, например задачи типа "писатель-читатель", в которых одна нить выполняет запись в буфер, а другая считывает записи из него. Поскольку они разделяют общий буфер, не стоит их делать отдельными процессами. Другой пример использования нитей – это управление сигналами, такими как прерывание с клавиатуры (del или break). Вместо обработки сигнала прерывания одна нить назначается для постоянного ожидания поступления сигналов. Таким образом, использование нитей может сократить необходимость в прерываниях пользовательского уровня. В этих примерах не столь важно параллельное выполнение, сколь важна ясность программы.

Наконец, в мультипроцессорных системах для нитей из одного адресного пространства имеется возможность выполняться параллельно на разных процессорах. Это действительно один из главных путей реализации разделения ресурсов в таких системах. С другой стороны, правильно сконструированные программы, которые используют нити, должны работать одинаково хорошо как на однопроцессорной машине в режиме разделения времени между нитями, так и на настоящем мультипроцессоре.

3.3.2.2. СОСТАВ АЛГОРИТМОВ ВНУТРЕННЕГО ПЛАНИРОВАНИЯ

Важнейшим фактором, влияющим на эффективность вычислительных систем, являются потери времени на организацию мультизадачного режима, поэтому в состав алгоритмов внутреннего планирования входят:

- алгоритм управления количеством процессов в рабочей смеси;
- алгоритм планирования очередности выбора задач для исполнения их на ЦП;
- алгоритмы выбора величины кванта времени, в течение которого процесс, получивший ЦП, может использовать его.

Особой функцией, возлагаемой на внутреннее планирование, является синхронизация параллельных процессов, протекающих в системе.

3.3.2.3. АЛГОРИТМ УПРАВЛЕНИЯ КОЛИЧЕСТВОМ ПРОЦЕССОВ В РАБОЧЕЙ СМЕСИ

Управление служит для повышения производительности системы на основе рационального использования аппаратуры ЭВМ. Для режима вытесняющей мультизадачности количество процессов, одновременно допускаемых в систему, представляет собой объем рабочей смеси N. Выбор значения N осуществляется с учетом величины кванта и длительности цикла. Введем следующие обозначения: q_i – величина кванта процессорного времени, отводимого i-му процессу; T_i – длительность одного цикла обработки для i-го процесса; $T_{з\,i}$, $T_{в\,i}$ – длительность перемещения i-го процесса из ОЗУ в ВЗУ и обратно при его вытеснении и восстановлении; $T_{с\,i}$ – затраты времени ОС на организацию работы i-го процесса.

Тогда имеем:

$$T_{ц\,i} = T_{з\,i} + T_{в\,i} + q_i + T_{с\,i}, \quad T = \sum_{i=1}^{N} (T_{з\,i} + T_{в\,i} + T_{с\,i} + q_i).$$

Если $T_{з\,i} = T_{в\,i} = T_п$; $T_{с\,i} = T_с$; $q_i = q$, $i = 1, ..., N$, то

$$T = (2T_п + q + T_с)\,N; \quad N = T / (2T_п + q + T_с). \qquad (3.1)$$

Нетрудно рассчитать также эффективность загрузки центрального процессора на одном цикле для одного процесса: полезное время процессора – q_i; непроизводительные затраты времени – $T_{нi} = T_{зi} + T_{пi} + T_{сi}$; показатель загрузки процессора полезной работой

$$\eta_i = \frac{q_i}{T_{нi} + q_i}. \qquad (3.2)$$

Для всех N процессов получим:

$$Q = \sum_{i=1}^{N} q_i; \quad T_н = \sum_{i=1}^{N} (T_{нi} + T_{пi} + T_{сi}); \quad \eta = \frac{Q}{T_н + Q} = \frac{1}{1+\beta}. \qquad (3.3)$$

где $\beta = T_н / Q$ – относительная величина потерь времени на переключение между процессами.

Увеличение количества задач, одновременно решаемых системой, связанное с увеличением значения N (выражение (3.1)), можно осуществить четырьмя методами: увеличением общей длительности интервала цикла T; уменьшением затрат времени ОС на организацию мультипрограммирования $T_с$; уменьшением затрат времени на перемещение процессов из ОЗУ в ВЗУ и обратно при их вытеснении и восстановлении $T_з$ и $T_в$; сокращением длительности квантов времени, отводимых для выполнения процессов q.

Сокращение длительности квантов времени q_i, отводимых для выполнения процессов, автоматически вызывает увеличение длительности нахождения процессов в системе (см. выражение (3.3.)), из-за чего этот способ имеет ограниченное применение.

Наиболее продуктивными являются методы, приводящие к сокращению затрат времени операционной системы $T_с$ и, в особенности, затрат времени на перемещение процессов из ОЗУ в ВЗУ и обратно при их вытеснении и восстановлении.

Уменьшение значения $T_с$ достигается путем создания эффективных системных программ, имеющих наименьшую длину программного кода и наивысшую скорость выполнения своих функций.

В современных вычислительных системах применяются все перечисленные способы увеличения эффективности обработки задач.

3.3.2.4. АЛГОРИТМЫ ВЫБОРА ОЧЕРЕДНОСТИ ОБРАБОТКИ

Планирование очередности обработки осуществляется на основе очереди процессов, находящихся в состоянии готовности. Все эти процессы в мультипрограммных системах конкурируют между собой, прежде всего, из-за времени центрального процессора. В течение кванта времени, выделенного процессу, возможно наступление одного или нескольких событий: выполнение процесса завершено; процесс перешел в состояние ожидания; центральный процессор потребовался для обслуживания процесса с более высоким приоритетом; завершился выделенный процессу квант времени; произошла ошибка в системе.

Решение о порядке выбора процессов из очереди осуществляется в соответствии с реализованными в ОС алгоритмами планирования очередности обработки. В настоящее время наиболее известными являются:

- алгоритм циклической обработки;
- алгоритм очередей с обратной связью;
- алгоритм выбора по характеру использования предыдущего кванта;
- алгоритм выбора с приоритетом по характеру блокировки.

Практически все алгоритмы планирования очередности обработки имеют эвристический характер. Сигналом к началу работы этих алгоритмов служат указанные выше события, наступившие в системе:

- если предшествующий процесс закончился, то выполняются действия по его выводу из системы:
- если предшествующий процесс перешел в состояние ожидания, то он перемещается в очередь блокированных процессов;
- если предшествующий процесс прерван операционной системой из-за того, что ЦП потребовался для обслуживания другого процесса с более высоким приоритетом, то прерванный процесс помещается в очередь прерванных процессов;
- если предшествующий процесс исчерпал выделенный ему квант времени, то он поступает в очередь готовых процессов;
- если алгоритм планирования активизирован из-за какого-либо другого события, то выполняются действия согласно реакции на ошибочную ситуацию, возникшую в системе.

Логика работы всех алгоритмов планирования очередности обработки практически совпадает. Различаются они лишь реализациями блоков "Выбор длины кванта" и "Выбор очередного процесса".

Рассмотрим алгоритмы реализации блока "выбор очередного процесса".

Алгоритм циклической обработки процессов не использует никакой информации о приоритетах обрабатываемых процессов. Все процессы, находящиеся в очереди, упорядочиваются по времени их поступления. Стоящий первым в очереди процесс получает квант времени q центрального процессора. Существует много разновидностей алгоритмов циклической обработки, которые называют также алгоритмами круговорота процессов, например, круговорот со смещением и эгоистический круговорот.

Алгоритм очередей с обратной связью организует некоторое количество M очередей, каждая из которых обслуживается в порядке поступления. Новый процесс, поступивший в систему, попадает в очередь № 1. После окончания использования очередного кванта времени процесс переходит из очереди с номером m в очередь с номером $m + 1$.

Алгоритм выбора по характеру использования предыдущего кванта различает два типа состояния готовности процессов: низкоприоритетная и высокоприоритетная готовность.

Если процесс полностью использовал предыдущий выделенный квант времени ЦП, то ему присваивается состояние "Низкоприоритетная готовность". Если процесс использовал не весь выделенный квант времени ЦП из-за перехода в состояние ожидание по вводу-выводу, то ему присваивается состояние "Высокоприоритетная готовность". На обслуживание сначала выбираются процессы, находящиеся в состоянии высокоприоритетной готовности, а затем, если их нет, процессы, находящиеся в состоянии "Низкоприоритетная готовность".

3.3.2.5. АЛГОРИТМЫ ВЫБОРА ВЕЛИЧИНЫ КВАНТА

Выбор величины кванта является принципиально необходимым в режиме разделения времени. Процедура квантования выполняется каждый раз, когда ОС выбирает процесс из рабочей смеси и активизирует его. В настоящее время наиболее распространенными являются:

- алгоритм равномерного квантования;
- алгоритм квантования по приоритету процесса;
- алгоритм минимизации количества переключений между процессами.

Алгоритм равномерного квантования – самый простой из упомянутых выше алгоритмов. В соответствии с этим алгоритмом каждому процессу, находящемуся в рабочей смеси, выделяется квант времени, длительность которого может изменяться на отрезке времени существования процесса в системе. Длительности квантов времени всех процессов равны между собой

Алгоритм квантования по приоритету процесса осуществляет регулирование длительности кванта q_i для i-го процесса в зависимости от его текущего приоритета p_i. Функциональная зависимость $q_i = q_i(p_i)$ может иметь любой допустимый вид и должна иметь следующие основные свойства: монотонность, положительная определенность, ограниченность.

Алгоритм квантования с минимизацией количества переключений заключается в том, что активизируемый процесс занимает не только свой квант, но и остатки квантов процессов, перешедших к этому моменту времени в состояние ожидания.

На практике применяют также различные сочетания описанных алгоритмов.

3.4. ВЗАИМОСВЯЗАННЫЕ И КОНКУРИРУЮЩИЕ ЗАДАЧИ

3.4.1. Средства управления ресурсами

Под управлением ресурсами в ОС понимается распределение ресурсов системы между различными задачами и процессами, одновременно функционирующими в ней.

В ОС, как правило, отсутствует отдельный супервизор ресурсов, поскольку функции распределения ресурсов реализуются как на уровне внешнего планирования, так и на уровне внутреннего планирования.

Основными функциями управления ресурсами являются:

- учет наличия и состояния ресурсов;
- прием и учет заявок на ресурсы от задач и процессов;
- распределение ресурсов между задачами и процессами;
- организация использования ресурсов, выделенных каждой задаче или процессу;
- возврат ресурса в систему по мере его освобождения потребителем.

Для реализации функций управления ресурсами в ОС формируются информационные таблицы, в которых отражаются следующие основные данные:

- *для ресурсов*:
  - учетная информация о ресурсе (идентификатор, класс, количество каналов и т.п.);
  - код состояния ресурса;
  - идентификатор процесса-владельца и т.п.;
- *для заявок на ресурсы*:
  - идентификатор процесса-заявителя;
  - приоритет процесса;
  - идентификатор и требуемый объем ресурса и т.п.

В ходе организации использования ресурсов формируются таблицы, в которых указываются списки распределенных и свободных ресурсов, связи между ресурсами и процессами.

Наряду с проблемой рационального распределения ресурсов между процессами существует также проблема синхронизации протекания параллельных процессов и исключение возникновения тупиков в вычислительной системе.

Процесс в мультипрограммной системе находится в состоянии *тупика*, если он ожидает некоторого события, которое никогда не произойдет.

Системная тупиковая ситуация, или ситуация "зависания" системы – это ситуация, когда один или более процессов оказываются в состоянии тупика.

В операционных системах тупики возникают в большинстве случаев как результат конкуренции процессов за обладание монопольно используемыми ресурсами (рис. 3.1).

Сформулированы следующие четыре необходимых условия наличия тупика:

1) процессы требуют предоставления им монопольного права управления ресурсом, которые им выделяются (условие *взаимоисключения*);

2) процессы удерживают за собой ресурсы, уже выделенные им, ожидая в то же время выделения дополнительных ресурсов (условие *ожидания* ресурсов);

3) ресурсы нельзя отобрать у процессов, удерживающих их, эти ресурсы не будут использованы для завершения работы (условие *неперераспределяемости*);

4) существует кольцевая связь процессов, в которой каждый процесс удерживает за собой один или более ресурсов, требующихся следующему процессу цепи (условие *кругового ожидания*).

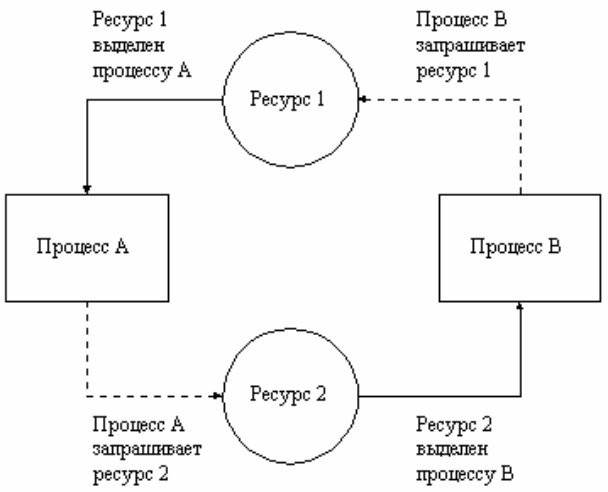

Рис. 3.1. Простая тупиковая ситуация

В проблеме тупиков выделяют следующие четыре основных направления: предотвращение тупиков, обход тупиков, обнаружение тупиков, восстановление после тупиков.

При *предотвращении тупиков* целью является обеспечение условий, исключающих возможность возникновения тупиковых ситуаций. Такой подход является вполне корректным решением о том, что касается самого тупика, однако он часто приводит к нерациональному использованию ресурсов.

Обход тупиков заключается в том, чтобы можно было предусматривать менее жесткие ограничения, чем в случае предотвращения тупиков, и тем самым обеспечить лучшее использование ресурсов. При наличии средств обхода тупиков не требуется такой реализации системы, при которой опасность тупиковых ситуаций даже не возникает. Методы обхода учитывают подобную возможность, однако в случае увеличения вероятности возникновения тупиковой ситуации здесь принимаются меры по аккуратному обходу тупика.

Методы ***обнаружения тупиков*** применяются в системах, которые допускают возможность возникновения тупиковой ситуации как следствие умышленных или неумышленных действий программистов. Целью средств обнаружения тупиков является установить сам факт возникновения тупиковой ситуации и точно определить те процессы и ресурсы, которые оказались вовлеченными в эту тупиковую ситуацию.

Методы ***восстановления после тупиков*** применяются для устранения тупиковых ситуаций с тем, чтобы система могла продолжать работать, а процессы, попавшие в тупиковую ситуацию, могли завершиться с освобождением занимаемых ими ресурсов.

3.4.2. Механизмы синхронизации процессов

3.4.2.1. ПАРАЛЛЕЛЬНЫЕ ПРОЦЕССЫ И КРИТИЧЕСКИЕ УЧАСТКИ

Рабочая смесь, созданная механизмами высшего и внутреннего планирования, представляет собой совокупность параллельных процессов.

Процессы называются *параллельными*, если они существуют в системе одновременно.

Параллельные процессы называются *независимыми*, если они работают без целенаправленной передачи сигналов информации друг другу.

Параллельные процессы называются *связанными*, если осуществляется направленный обмен сигналами (информацией) между ними.

Связанные параллельные процессы бывают *синхронными*, когда обеспечиваются согласование скоростей их развития в системе, и *асинхронными*, если скорости протекания процессов не регулируются в системе.

В механизмах синхронизации нуждаются все виды параллельных процессов, функционирующих в мультипрограммной вычислительной системе.

Не связанные между собой процессы также нуждаются в синхронизации своей работы. Это объясняется тем, что они используют во время функционирования одни и те же физические и логические внешние устройства, которые в каждый конкретный момент времени могут обслуживать только один процесс (критические ресурсы).

Ресурс системы называется *критическим*, если он допускает в каждый момент времени обслуживание только одного процесса.

Общим принципом, положенным в основу всех механизмов синхронизации процессов, является принцип взаимоисключения.

Принцип взаимоисключения: каждый процесс, обращающий к разделяемым (критическим) ресурсам, должен исключить возможность для всех других процессов одновременного с ним использования этого ресурса.

Использование принципа взаимоисключения требует встраивания в программы процессов механизмов синхронизации, обеспечивающих выполнение следующих условий:

- при обращении нескольких процессов к одному разделяемому ресурсу только одному из них разрешено воспользоваться этим ресурсом;
- в каждый момент времени только один процесс должен владеть критическим ресурсом.

Все механизмы синхронизации, реализующие принцип взаимоисключения, основаны на применении концепции критического участка программы.

Критическим участком, критической областью программы процесса называется тот отрезок программного кода процесса, на котором этот процесс образуется к критическому ресурсу.

Количество критических участков в процессе зависит только от того, к ресурсам какого вида он обращается при своем функционировании.

Когда некоторый процесс находится на своем критическом участке, другие процессы могут продолжать выполнение, но без входа в их критические участки (занятым критическим ресурсам), когда процесс выходит из критического участка, то должно быть разрешено использование освобожденного критического ресурса. Обеспечение взаимоисключения является основной проблемой параллельного программирования.

3.4.2.2. ПРИМИТИВЫ ВЗАИМОИСКЛЮЧЕНИЯ

Общим подходом к построению механизмов синхронизации с использованием концепции критических участков является применение примитивов взаимоисключения.

Примитивами взаимоисключения называется программная конструкция, обеспечивающая реализацию взаимоисключений для параллельных процессов. В обобщенном виде можно указать два примитива взаимоисключений:

1) примитив *вход_взаимоисключения* – с его помощью фиксируется захват критического ресурса данным процессом и устанавливается запрет на использование его другими процессами;

2) примитив *выход_взаимоисключения* – с его помощью процесс сообщает об освобождении им критического ресурса.

Простейший алгоритм синхронизации с применением примитивов взаимоисключения состоит в следующем (рис. 3.2). Главный процесс запускает в работу два параллельных процесса $П_1$ и $П_2$, после чего он может закончить свою работу. Каждый из параллельных процессов в своем теле имеет области работы с критическим ресурсом. Эти области обязательно обрамляются примитивами *вход_взаимоисключения* и *выход_взаимоисключения*.

В рассматриваемом случае двух процессов эти примитивы работают следующим образом. Когда процесс $П_1$ выполняет примитив *вход_взаимоисключения* и если при этом процесс $П_2$ находится вне своего критического участка, то $П_1$ входит в свой критический участок, выполняет работу с критическим ресурсом, а затем выполняет примитив выход_взаимоисключения, показывая тем самым, что он вышел из своего критического участка. Если $П_1$ выполняет *вход_взаимоисключения*, в то время как $П_2$ находится на своем критическом участке, то процесс $П_1$ переводится в состояние ожидания, пока процесс $П_2$ не выполнит *выход_взаимоисключения*. Только после этого процесс $П_1$ может выйти из состояния ожидания и войти в свой критический участок.

Если процессы $П_1$ и $П_2$ выполняют *вход_взаимоисключения* одновременно, то одному из них операционная система разрешает продолжить работу, а другой переводит в состояние ожидания.

3.4.2.3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИМИТИВОВ ВЗАИМОИСКЛЮЧЕНИЯ

Программная реализация примитивов взаимоисключения осуществляется с соблюдением следующих ограничений:

1) задача должна быть решена чисто программным способом на машине, не имеющей специальных команд взаимоисключения;

2) не должно быть никаких предположений об относительных скоростях выполнения асинхронных параллельных процессов;

3) процессы, находящиеся вне своих критических участков, не могут препятствовать другим процессам входить в их собственные критические участки;

4) не должно быть бесконечного откладывания момента входа процессов в их критические участки.

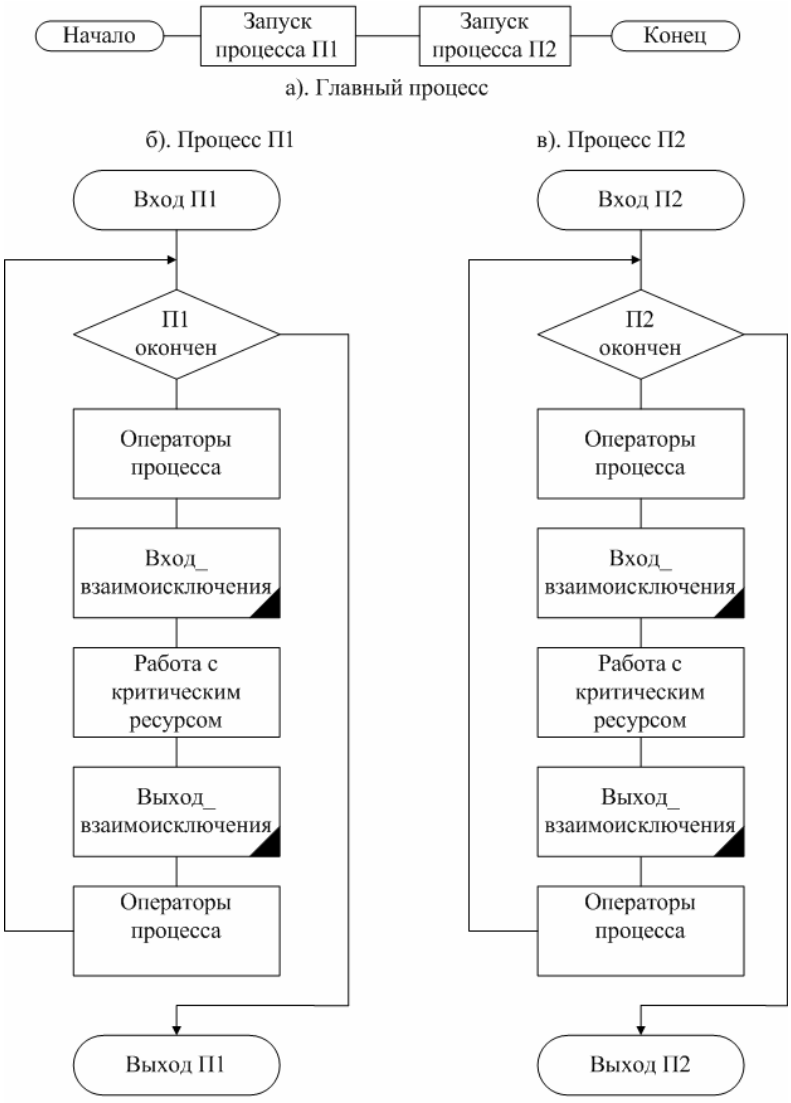

Рис. 3.2. Схема применения примитивов взаимоисключения

Впервые изящную программную реализацию механизма взаимоисключения предложил голландский математик Деккер. Впоследствии алгоритм Деккера был существенно усовершенствован Е. Дейкстрой. Программа реализации алгоритма Деккера-Дейкстры приведена на рис. 3.3.

```
program Алгоритм Деккера-Дейкстры;
var избранный_процесс: (первый, второй);
        П1_хочет_войти, П2_хочет_войти: логический;
procedure П1;
begin
        while истина do
        begin
                П1_хочет_войти:=истина;
                while П2_хочет_войти do
                begin
                        if избранный_процесс=второй then
                        begin
                                П1_хочет_войти:=ложь;
                                while избранный_процесс=второй
        do ;
                                П1_хочет_войти:=истина;
                        end;
                end;
                критический участок процесса П1;
                избранный_процесс:=второй;
                П1_хочет_войти:=ложь;
                прочие операторы процесса П1;
        end;
end;
procedure П2;
begin
        while истина do
        begin
                П2_хочет_войти:=истина;
                while П1_хочет_войти do
                begin
                        if избранный_процесс=первый then
                        begin
                                П2_хочет_войти:=ложь;
                                while избранный_процесс=первый
        do ;
                                П2_хочет_войти:=истина;
                        end;
                end;
                критический участок процесса П2;
                избранный_процесс:=первый;
                П2_хочет_войти:=ложь;
                прочие операторы процесса П2;
        end;
end;
{  Основной процесс  }
begin
        П1_хочет_войти:=ложь;
        П2_хочет_войти:=ложь;
        Избранный_процесс:=первый;
        {  Запуск параллельных процессов  }
        П1; П2;
end;
```

Рис. 3.3. Программная реализация примитивов взаимоисключения

3.4.2.4. РЕАЛИЗАЦИЯ ПРИМИТИВОВ ВЗАИМОИСКЛЮЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ АППАРАТНЫХ СРЕДСТВ

Для синхронизации с использованием аппаратных средств необходимо наличие в системе команд ЭВМ команды, которая обеспечивает выполнение следующих операций: чтение переменной, запись ее значения в область сохранения, установка нужного конкретного значения этой переменной.

Подобная команда в настоящее время существует практически во всех ЭВМ и обычно имеет мнемоническое обозначение *testandset* (*проверить_и_установить*). Пример применения *testandset* приведен на рис. 3.4.

```
program Пример_TestAndSet;
var активный: логический;
{  Процесс П1  }
procedure П1;
var первому_входить_нельзя: логический;
begin
  while истина do
  begin
    первому_входить_нельзя:=истина;
    while первому_входить_нельзя do
      testandset(первому_входить_нельзя, активный);
    Критический участок процесса П1;
    активный:=ложь;
    Прочие операторы процесса П1;
  end;
end;
{  Процесс П2  }
```

```
procedure П2;
var второму_входить_нельзя: логический;
begin
  while истина do
  begin
    второму_входить_нельзя:=истина;
    while второму_входить_нельзя do
      testandset(второму_входить_нельзя, активный);
    Критический участок процесса П2;
    активный:=ложь;
    Прочие операторы процесса П2;
  end;
end;
{  Основной процесс  }
begin
  активный:=ложь;
  П1;  { запуск процесса П1 }
  П2;  { запуск процесса П2 }
end;
```

Рис. 3.4. Реализация взаимоисключения с применением команды *проверить_и_установить*

Неделимая команда с двумя операндами *testandset*(*a*, *b*) читает значения логической переменной *b*, копирует его в *a*, а затем устанавливает для *b* значение истины, и все это в рамках одной непрерывной операции.

3.4.2.5. СЕМАФОРЫ

Семафор – это защищенная переменная, значение которой можно опрашивать и изменять только при помощи операции *инициализация_семафора* и двух специальных операций P и V:

- операция P над семафором S записывается как P(S) и выполняется по правилам:

если S > 0 **то** S: = S–1 **иначе** (ожидать на S);

- операция V над семафором S записывается как V(S) и выполняется по правилам:

если (один или более процессов ожидают на S) **то** (разрешить одному из процессов продолжить работу) **иначе** S: = S+1.

```
program Пример_семафора;
var активный: семафор;
procedure П1;
begin
        while истина do
        begin
                Предшествующие операторы процесса П1;
                P(активный);
                Критический участок процесса П1
                V(активный);
                Прочие операторы процесса П1;
        end;
end;
procedure П2;
begin
        while истина do
        begin
                Предшествующие операторы процесса П2;
                P(активный);
                Критический участок процесса П2
                V(активный);
                Прочие операторы процесса П2;
        end;
end;
{  Основной процесс  }
begin
        инициализация семафора(активный, 1);
        П1;
        П2;
end;
```

Рис. 3.5. Обеспечение взаимоисключения с помощью семафора

Различают два вида семафоров: двоичные семафоры, у которых S может принимать значения 0 и 1, и считающие семафоры, где S может принимать неотрицательные целые значения.

Подобно операции *проверить_и_установить,* операции *инициализация_семафора*, P(S) и V(S) являются неделимыми. Участки взаимоисключения по семафору S обрамляются операциями P(S) и V(S). Если одновременно несколько процессов попытаются выполнить операцию P(S), то только одному из них будет позволено это сделать, а остальным придется ждать.

Семафоры и операции над ними могут быть реализованы как программно (рис. 3.5), так и аппаратно. При программной реализации они размещаются в той части ядра операционной системы, где осуществляется управление сменой состояний процессов.

3.4.2.6. МОНИТОРЫ

Монитор – это механизм организации параллелизма, который содержит как данные, так и процедуры, необходимые для реализации динамического распределения конкретного общего ресурса или группы общих ресурсов.

Чтобы обеспечить выделение нужного ему ресурса, процесс должен обратиться к конкретной процедуре монитора.

Необходимость входа в монитор в различные моменты времени может возникать у многих процессов. Однако вход в монитор находится под жестким контролем: здесь осуществляется взаимоисключение процессов, так что в каждый момент времени только одному процессу разрешается войти в монитор. Процессам, которые хотят войти в монитор, когда он занят, приходится ждать, причем режимом ожидания автоматически управляет сам монитор.

Если процесс обращается к некоторой процедуре монитора и обнаруживается, что соответствующий ресурс уже занят, то эта процедура монитора выдает процессу команду WAIT (ЖДАТЬ). Процесс, получивший такую команду, переводится в режим ожидания вне монитора.

Со временем процесс, занимавший нужный ресурс, освобождает его, обращаясь для этого к монитору. Соответствующая процедура монитора может просто принять уведомление о возвращении ресурса, а затем ждать, пока не поступит запрос от другого процесса, которому потребуется этот ресурс. Однако может оказаться, что уже имеются процессы, ожидающие освобождение данного ресурса. В этом случае монитор выполняет примитив *оповещение* SIGNAL, чтобы один из ожидающих процессов мог занять данный ресурс и покинуть очередь к монитору. Если процесс сигнализирует об освобождении ресурса и в это время нет процессов, ожидающих этот ресурс, то подобное оповещение не вызывает никаких других последствий, кроме того, что монитор вновь вносит ресурс в список свободных.

```
monitor Распределитель_ресурса;
var ресурс_занят: логический;
    ресурс_свободен: условие;
procedure захватить_ресурс;
begin
  if ресурс_занят then
    WAIT(ресурс_свободен);
  ресурс_занят:=истина;
end;
procedure возвратить_ресурс;
begin
  ресурс_занят:=ложь;
  SIGNAL(ресурс_свободен);
end;
begin
  ресурс_занят:=ложь;
end;
```

Рис. 3.6. Распределение ресурса при помощи монитора

На практике у процессов может возникнуть необходимость находиться в режиме ожидания вне монитора по различным причинам. Поэтому было введено понятие *переменной-условие*. Для каждой отдельно взятой причины, по которой процесс может быть переведен в состояние ожидания, назначается свое условие. В связи с этим команды ожидания и оповещения модифицируются. В них включают имена условий, так что они приобретают вид:

WAIT (имя_условие)

SIGNAL (поле_условие)

Переменные-условия совершенно не похожи на обычные переменные. Когда определяется переменная-условие, заводится очередь. Процесс, выдавший команду ожидания, включается в эту очередь, а процесс, выдавший команду оповещения, тем самым позволяет ожидающему процессу выйти из очереди и войти в монитор (рис. 3.6).

3.4.3. Алгоритмы управления ресурсами

3.4.3.1. ВВОДНЫЕ ЗАМЕЧАНИЯ

Для реализации функций управления ресурсами операционная система осуществляет постоянный учет их наличия и состояния. В современных ОС наиболее употребительным является идентификация ресурса с помощью файла его описания.

Способы формирования запросов на ресурсы различаются местом их выдачи в операционную систему – в пакете заданий или в программе во время ее выполнения. В первом случае ОС осуществляет статическое распределение ресурсов, а во втором – динамическое. Операционная система может принять, отложить или проигнорировать запрос на ресурс, что зависит от реализованного в ОС алгоритма управления ресурсами.

Существуют также комбинированные способы формирования запросов на ресурсы с предварительным заказом в пакете заданий или в программе и исполнительным запросом во время выполнения программы.

3.4.3.2. АЛГОРИТМ ПРЕДОСТАВЛЕНИЯ РЕСУРСА ПО ПЕРВОМУ ОБРАЩЕНИЮ

Алгоритм представления ресурса по первому обращению к нему является самым простым из алгоритмов управления ресурсами. Суть данного алгоритма состоит в следующем.

Независимо от места выдачи запроса на ресурс (пакет заданий или выполняющаяся в системе программа) ОС выполняет следующие действия:

1) фиксирует первое упоминание имени запрашиваемого ресурса;

2) распределяет конкретное физическое устройство, соответствующее этому имени, источнику запроса;

3) помечает ресурс в соответствующей таблице как выделенный (занятый, распределенный).

Распределенный ресурс не может быть выделен другим процессам, выполняющимся в системе, пока процесс, которому этот ресурс выделен, не освободит его или не прекратит свое существование.

Данный алгоритм прост в реализации, однако обладает серьезным недостатком – при его применении возможно появление тупиковых ситуаций в вычислительной системе. Поэтому этот алгоритм называют также алгоритмом с возможным возникновением тупиков.

3.4.3.3. АЛГОРИТМЫ ПРЕДОТВРАЩЕНИЯ ТУПИКОВ

Все алгоритмы предотвращения тупиковых ситуаций при управлении ресурсами основаны на нарушении хотя бы одного из необходимых условий наличия тупика Д. Хавендером (Havender J.W.) предложены следующие стратегии:

1) каждый процесс должен запрашивать все требуемые ему ресурсы сразу, причем не может начать выполняться до тех пор, пока все они не будут ему предоставлены;

2) если процесс, удерживающий определенные ресурсы, получает отказ в удовлетворении запроса на дополнительные ресурсы, этот процесс должен освободить свои первоначальные ресурсы и при необходимости запросить их снова вместе с дополнительными;

3) введение линейной упорядоченности по типам ресурсов для всех процессов; другими словами, если процессу выделены ресурсы данного типа, то в дальнейшем он может запросить только ресурсы более далеких по порядку типов.

Можно видеть, что каждая из этих стратегий нарушает одно из необходимых условий существования тупика. Сознательно оставлено условие монопольного владения процессом выделенными ему ресурсами, поскольку нужно предусмотреть возможность работы с *закрепленными* ресурсами.

Алгоритм предварительного распределения ресурсов в соответствии с первым стратегическим принципом Хавендера требует, чтобы процесс сразу же запрашивал все ресурсы, которые ему понадобятся. Система в ответ на эти запросы должна представлять ресурсы по принципу "все или ничего", т.е. если набор ресурсов, необходимый процессу, имеется, то система предоставляет процессу все эти ресурсы сразу же так, что он может продолжить свою работу. Если в данный момент времени полного набора ресурсов нет, то этому процессу придется ждать, пока они не освободятся.

В ***алгоритме распределения ресурсов с освобождением при отказе*** процесс, имеющий в своем распоряжении некоторые ресурсы, если он получает отказ на запрос о выделении дополнительных ресурсов, должен освободить все принадлежащие ему ресурсы и при необходимости запрашивать их снова вместе с дополнительными. Такая ситуация действенно ведет к нарушению условия неперераспределяемости, подобным образом можно забирать ресурсы у удерживающих их процессов до завершения этих процессов.

Алгоритм распределения с линейным упорядочением по типам ресурсов предусматривает предварительное присвоение всем ресурсам вычислительной системы уникальных номеров.

Процессы в ходе своего выполнения должны запрашивать необходимые им ресурсы строго в порядке возрастания номеров этих ресурсов, так что ситуация кругового ожидания возникнуть не может.

3.4.3.4. АЛГОРИТМЫ ОБХОДА ТУПИКОВ

Если даже необходимые условия возникновения тупиков не нарушены, то все же имеется возможность избежать тупиковой ситуации, рационально распределяя ресурсы с соблюдением определенных правил. Наиболее известными алгоритмами, позволяющими осуществить обход тупиков, являются алгоритм Дейкстры, называемый также алгоритмом банкира, и алгоритм Хабермана, называемый алгоритмом регулируемого распределения.

Алгоритм регулируемого распределения основан на представлении распределения ресурсов, запросов и процессов в виде ориентированного графа (орграфа), называемого графом распределения ресурсов (ГРР).

***Правило Хабермана*:** Если граф распределения ресурсов имеет циклы, то такое распределение ресурсов по процессам может привести к образованию тупиковой ситуации.

Это правило позволяет построить следующий алгоритм распределения ресурсов:

- при поступлении очередного запроса проверить наличие свободного ресурса;
- если свободный ресурс отсутствует, то отказ, процесс блокировать, иначе сделать предварительное распределение и скорректировать ГРР;
- проверить ГРР на наличие циклов;
- если в ГРР имеются циклы, то отменить предварительное распределение, восстановить ГРР, сформировать отказ и блокировать процесс, иначе закрепить ресурс за процессом.

Алгоритм Хабермана позволяет обойти тупики, однако он сложен в реализации из-за необходимости работы с графом распределения
ресурсов.

Контрольные вопросы к теме 3

1. Описать организацию управления в ОС.
2. Перечислить дисциплины обслуживания.
3. Перечислить режимы обслуживания.
4. Описать средства управления задачами на уровне внешнего планирования.
5. Дать определение понятия "контекст процесса".

6. Пояснить понятия "нить" и "процесс".
7. Назвать состав алгоритмов внутреннего планирования.
8. Охарактеризовать алгоритмы управления количеством процессов в рабочей смеси.
9. Охарактеризовать алгоритмы выбора очередности обработки.
10. Охарактеризовать алгоритмы выбора величины кванта
11. Дать определение понятий параллельных процессов, критического ресурса, критического участка.
12. Что такое "примитивы взаимоисключения"?
13. Каковы механизмы реализации примитивов взаимоисключения?
14. Описать алгоритмы предотвращения тупиков.
15. Описать алгоритмы обхода тупиков.

Т Е М А 4. Управление памятью в операционных системах

4.1. ПОНЯТИЕ ОБ ОРГАНИЗАЦИИ И УПРАВЛЕНИИ ФИЗИЧЕСКОЙ ПАМЯТЬЮ

В операционных системах различают два вида памяти: основная (первичная) и внешняя (вторичная).

Основная память (main storage) – оперативная память центрального процессора или ее часть, представляющая собой единое пространство памяти.

Внешняя память (external storage) – память, данные в которой доступны центральному процессору посредством операций ввода-вывода.

Для непосредственного выполнения программ или обращения к данным необходимо, чтобы они размещались в основной памяти. Внешняя память имеет, как правило, гораздо большую емкость, чем основная, стоит дешевле и позволяет хранить данные и программы, которые должны быть наготове для обработки.

Кроме основной и внешней памяти в современных ЭВМ существует дополнительная быстродействующая память, называемая ***кэш-памятью*.**

Все три вида памяти образуют ***иерархию памяти*** вычислительной машины (рис. 4.1).

Операционным системам с несколькими уровнями иерархии памяти свойственна высокая интенсивность челночных обменов программами и данными между физическими устройствами памяти различных уровней. Такие обмены отнимают системные ресурсы (например, время центрального процессора), которые можно было бы использовать более продуктивно.

Основная память представляет собой один из самых дорогостоящих ресурсов. Главной задачей при разработке ОС считается оптимальное использование основной памяти на основе рациональной организации и управления.

Под *организацией памяти* понимается то, каким образом представляется основная память.

Под *управлением памятью* понимается то, как используется основная память.

В ОС применяются следующие виды представления основной памяти:

1) фиксированными блоками равного размера;
2) фиксированными разделами неодинакового размера;
3) динамическими разделами, размеры которых изменяются в ходе работы вычислительной системы.

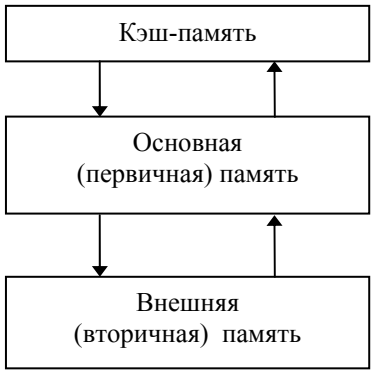

Рис. 4.1. Иерархия памяти ЭВМ

Использование основной памяти может осуществляться следующими способами:

1) размещение в памяти единовременно только одной программы пользователей;
2) размещение в памяти одновременно нескольких программ пользователей;
3) размещение программ пользователей в конкретном заранее заданном разделе основной памяти;
4) размещение каждой программы пользователя в одном непрерывном (односвязном) пространстве основной памяти;
5) размещение программы пользователя в несмежных областях оперативной памяти.

В операционных системах может применяться любая комбинация перечисленных видов представления и способов использования основной памяти ЭВМ.

Независимо от того, какая схема организации памяти принята для конкретной ОС, необходимо решить, какие стратегии следует применять для достижения оптимальных характеристик.

Стратегии управления памятью определяют, как будет работать память с конкретной схемой организации при различных подходах к решению следующих вопросов:

1) когда следует поместить новую программу в память;
2) в какое место основной памяти будет размещаться очередная программа;
3) как разместить очередную программу в памяти (с минимизацией потерь памяти или с максимизацией скорости размещения);
4) какую из находящихся в памяти программ следует вывести из памяти, если необходимо обязательно разместить новую программу, а память уже заполнена.

В существующих ОС реализованы стратегии управления, по-разному отвечающие на перечисленные выше вопросы, что в немалой степени обусловлено имеющимися в распоряжении разработчиков аппаратурными и программными средствами.

Стратегии управления памятью делятся на следующие категории: стратегии выборки; стратегии размещения; стратегии замещения.

В свою очередь, стратегии выборки разделяют на две подкатегории: стратегии выборки по запросу (по требованию), стратегии упреждающей выборки.

Стратегии выборки ставят своей целью определить, когда следует "втолкнуть" очередную программу (или блок программы) или данные в основную память.

Стратегии размещения ставят своей целью определить, в какое место основной памяти следует размещать поступающую программу. Наиболее распространенными являются стратегии размещения, реализующие принципы занятия "первого подходящего", "наиболее подходящего" и "наименее подходящего" по размерам свободного участка памяти.

Стратегии замещения ставят своей целью определить, какой блок программы или данных следует вывести ("вытолкнуть") из основной памяти, чтобы освободить место для размещения вновь поступающих программ или данных.

При реализации стратегий размещения операционные системы часто учитывают требования связного распределения памяти для программ и данных.

Связное распределение памяти – такое распределение основной памяти ЭВМ, при котором каждая программ занимает один непрерывный (связный) блок ячеек памяти.

Несвязное распределение памяти – такое распределение основной памяти ЭВМ, при котором программа пользователя разбивается на ряд блоков (сегментов, страниц), которые могут размещаться в основной памяти в участках, не обязательно соседствующих друг с другом (в несмежных участках). В этом случае обеспечивается более эффективное использование пространства основной памяти.

Эффективность той или иной стратегии размещения можно оценить с помощью коэффициента использования памяти η:

$$\eta = \frac{V_{\text{п}}}{V_{\text{оп}} - V_{\text{ос}}} = \frac{V_{\text{п}}}{V_{\text{о}}}, \tag{4.1}$$

где $V_{\text{п}}$ – объем памяти, занимаемый программами пользователя; $V_{\text{оп}}$ – полный объем основной памяти; $V_{\text{ос}}$ – объем памяти, занимаемый операционной системой; $V_{\text{о}}$ – объем памяти, доступный для распределения.

4.2. МЕТОДЫ СВЯЗНОГО РАСПРЕДЕЛЕНИЯ ОСНОВНОЙ ПАМЯТИ

(без использования дискового пространства)

4.2.1. Связное распределение памяти для одного пользователя

Связное распределение памяти для одного пользователя, называемое также одиночным непрерывным распределением, применяется в ЭВМ, работающих в пакетном однопрограммном режиме под управлением простейшей ОС.

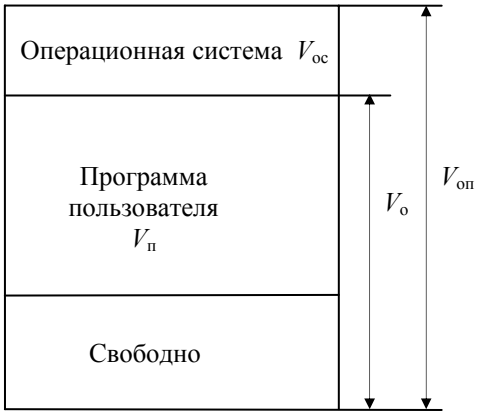

Рис. 4.2. Связное распределение памяти для одного пользователя

Вся основная часть ЭВМ, не занятая программами операционной системы, выделяется программе единственного на данном отрезке времени пользователя (рис. 4.2).

Размер программы в этом случае ограничивается размером доступной основной памяти, однако существует возможность выполнения программ, размер которых превышает размер основной памяти, используя механизм оверлеев.

Коэффициент использования памяти вычисляется по формуле

$$\eta_{c1} = V_{\text{п}} / V_{\text{о}}, \tag{4.2}$$

где $V_{\text{п}}$ – размер программы пользователя; $V_{\text{о}}$ – объем доступной для распределения основной памяти ЭВМ.

Функциями ОС являются: выделение программе необходимого пространства памяти; защита памяти; освобождение памяти.

Функция выделения памяти сводится к предоставлению программе всей доступной памяти ЭВМ.

Защита памяти в однопрограммных системах заключается в установке защиты областей памяти, занятых операционной системой, от воздействия программ пользователя. Эта функция реализуется при помощи одного *регистра границы*, встроенного в центральный процессор. Регистр границы содержит либо старший адрес команды, относящийся к операционной сис-

теме, либо младший адрес доступной программе основной памяти (адрес начала программы). Если программа пользователя пытается войти в область операционной системы, то вырабатывается прерывание по защите памяти, и программа аварийно завершается.

4.2.2. Связное распределение памяти при мультипрограммной обработке

При мультипрограммной обработке в памяти компьютера размещается сразу несколько заданий. Распределение памяти между заданиями в этом случае может быть выполнено следующими способами: распределение фиксированными разделами; распределение переменными разделами; распределение со свопингом.

Распределение фиксированными разделами имеет две модификации:

1) с загрузкой программ в абсолютных адресах;
2) с загрузкой перемещаемых модулей.

При загрузке перемещаемых модулей вся оперативная память машины разбивается на некоторое количество разделов фиксированного размера (рис. 4.3). Размеры разделов могут не совпадать. В каждом разделе может быть размещено только одно задание.

В случае загрузки программ в абсолютных адресах при их подготовке указывается начальный адрес загрузки программ, совпадающий с начальным адресом раздела, в котором эта программа будет выполняться.

В случае загрузки перемещаемых модулей раздел, в котором будет размещено задание, либо автоматически определяется операционной системой в соответствии с реализованной в нем стратегией выбора раздела ("первый подходящий", "самый подходящий", "самый неподходящий"), либо указывается операционной системе специальными командами языка управления заданиями.

В обоих случаях задание монопольно владеет всем объемом оперативной памяти раздела, в который оно было помещено операционной системой.

Коэффициенты использования памяти при распределении с фиксированными разделами вычисляется по формулам:

$$\eta_{\text{см}} = \frac{1}{V_{\text{о}}} \sum_{i=1}^{N_{\text{ф}}} V_{\text{п}\,i} \ , \qquad (4.3)$$

где $\eta_{\text{см}\,i}$ – коэффициент использования памяти i-го раздела; $V_{\text{о}\,i}$ – размер i-го раздела; $V_{\text{п}\,i}$ – длина программы, помещенной в i-й раздел; $N_{\text{ф}}$ – количество разделов; $V_{\text{о}}$ – общий объем оперативной памяти, доступной для распределения.

Основным недостатком распределения памяти фиксированными разделами является неэффективное использование ресурсов вычислительной системы из-за возможного появления длинных очередей заданий, ожидающих освобождения конкретного раздела в то время, как остальные разделы пусты (рис. 4.3).

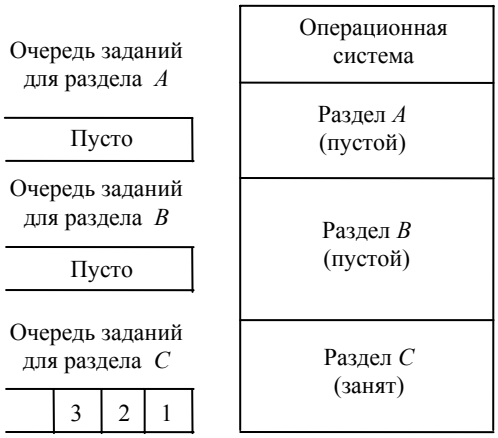

Рис. 4.3. Распределение памяти фиксированными разделами

Защита памяти при распределении фиксированными разделами выполняется аналогично защите памяти для одного пользователя, только теперь необходимо наличие нескольких *граничных регистров* – по два регистра на каждый раздел. В одном из граничных регистров указывается нижняя граница раздела, а во втором – его верхняя граница. Если программа пользователя пытается обратиться к данным, расположенным вне области адресов данного раздела, то вырабатывается прерывание по защите памяти.

В мультипрограммных системах с фиксированными разделами наблюдается явление фрагментации памяти.

Фрагментация памяти – появление в памяти вычислительной машины чередования занятых и незанятых (свободных) участков оперативной памяти.

При распределении фиксированными разделами появление фрагментации обусловлено тем, что либо задания пользователей не полностью занимают выделенные им разделы, либо часть разделов остается незанятой (рис. 4.4).

Уровень фрагментации памяти можно оценить *коэффициентом фрагментации* $K_{\text{ф}}$, вычисляемым по формуле

$$K_{\text{ф}} = \frac{1}{V_{\text{о}}} \sum_{i=1}^{N_{\text{д}}} V_{\text{д}} \ ,$$

где $V_{д\,i}$ – размер i-й "дыры", т.е. i-го участка свободной памяти, ограниченного программами пользователей; $N_д$ – количество "дыр", т.е. участков свободной памяти, лежащих между программами пользователей; $V_о$ – объем оперативной памяти, доступной для распределения.

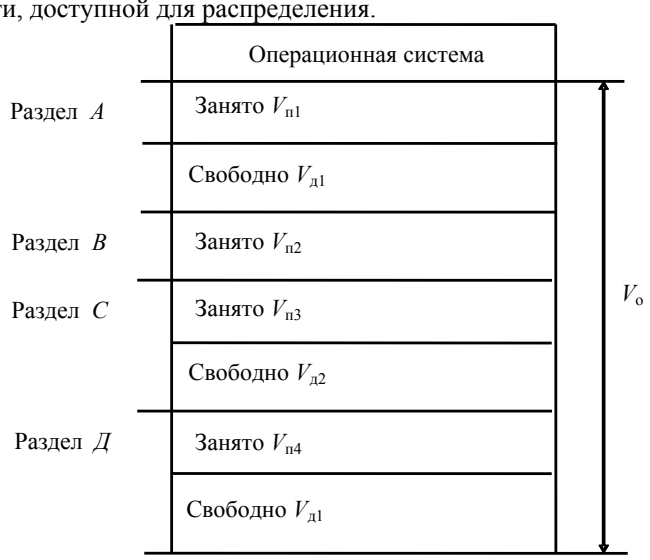

Рис. 4.4. Фрагментация памяти

Фрагментация памяти представляет собой нарушение односвязности пространства свободной памяти ЭВМ, что приводит к снижению эффективности использования памяти.

Распределение памяти переменными разделами предназначено для повышения эффективности использования оперативной памяти ЭВМ. Суть способа распределения памяти переменными разделами состоит в том, что заданиям, когда они поступают, выделяется такой объем памяти, который им требуется, т.е. размер раздела оперативной памяти, выделяемой каждому заданию, в точности соответствует размеру этого задания. Поэтому "перерасхода" памяти, как это происходит при распределении фиксированными разделами, в данном способе не наблюдается.

Имеется две модификации способа распределения переменными разделами: распределение переменными неперемещаемыми разделами; распределение переменными перемещаемыми разделами.

При распределении памяти *переменными неперемещаемыми разделами* (динамическими разделами) операционная система создает две таблицы: таблицу учета распределенных областей памяти и таблицу учета свободных областей памяти ("дыр").

При поступлении очередного задания память для него отводится на этапе долгосрочного планирования, причем выделение памяти осуществляется по информации из таблицы учета "дыр" в соответствии с принятой в ОС стратегией размещения ("первый подходящий", "самый подходящий", "самый неподходящий"). При успешном распределении ОС корректирует обе таблицы – распределенных и свободных областей.

После окончания какого-либо задания занимаемый им участок памяти освобождается, и операционная система корректирует таблицу распределенных областей, вычеркивая из нее информацию о закончившемся задании, а также заносит в таблицу свободных областей данные о вновь появившейся "дыре" (рис. 4.5, *б*).

При распределении памяти *переменными перемещаемыми разделами* операционная система осуществляет действия, называемые *уплотнением памяти,* состоящими в перемещении всех занятых участков к одному или другому краю основной памяти (рис. 4.5, *в*). Благодаря этому вместо большого количества небольших "дыр", образующихся при использовании распределения переменными неперемещаемыми разделами, формируется единый (связный) участок свободной памяти. Этот процесс называют также *дефрагментацией* памяти.

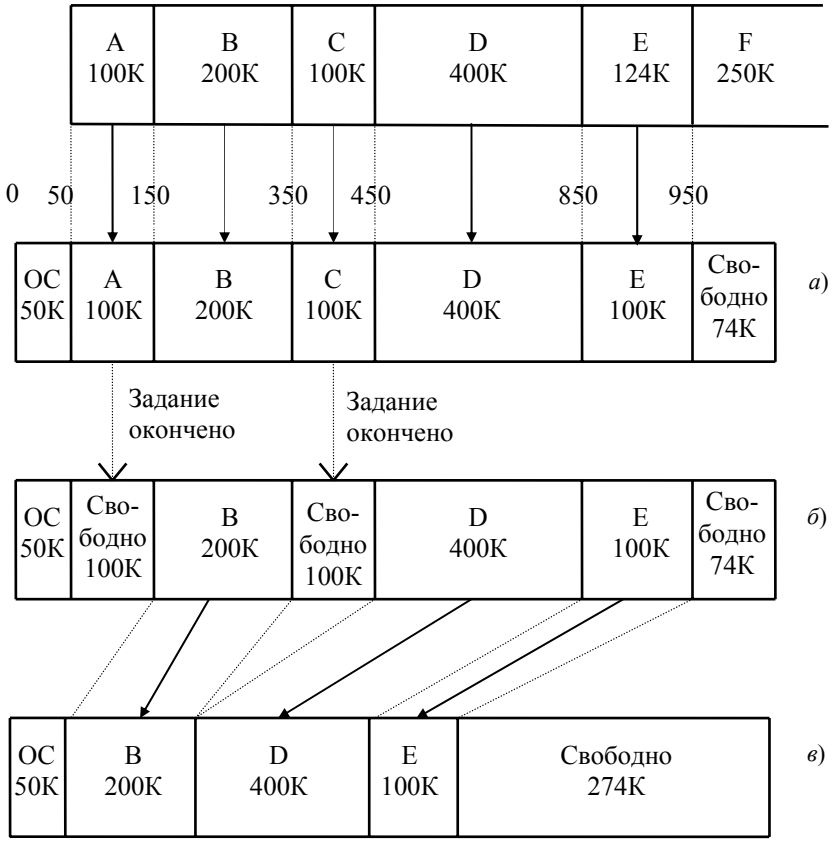

Рис. 4.5. Распределение памяти разделами

Дефрагментация памяти, применяемая при распределении перемещаемыми разделами, имеет свои недостатки:

1) требуются дополнительные затраты времени;

2) во время уплотнения памяти система должна прекращать (приостанавливать) все другие работы, что зачастую может оказаться неприемлемым;

3) необходимость перемещения заданий в памяти требует хранения значительного объема информации, связанной с размещением программ в памяти, что увеличивает требования к памяти со стороны ОС;

4) при интенсивном потоке коротких программ может возникнуть необходимость частой дефрагментации памяти, так что затачиваемые на эти цели системные ресурсы могут оказаться неоправданными получаемой выгодой.

Распределение памяти со свопингом (от англ. swapping – подкачка) характеризуется тем, что в отличие от рассмотренных ранее способов распределения программы пользователей не остаются в основной памяти до момента их завершения. Вся память целиком на короткий период выделяется одному заданию, затем в некоторый момент времени это задание выводится (выталкивается, т.е. осуществляется "откачка"), а очередное задание вводится (вталкивается, т.е. осуществляется "подкачка"). В обычном случае каждое задание, еще до своего завершения, будет много раз перекачиваться из внешней памяти в основную и обратно. Для обеспечения свопинга во внешней памяти ОС создает один или несколько *файлов подкачки*, где хранятся образы оперативной памяти находящихся в работе заданий пользователей. Способ распределения памяти со свопингом применяется в простейших ОС, работающих в режиме разделения времени.

4.2.3. Стратегии размещения информации в памяти

Стратегии размещения информации в памяти предназначены для того, чтобы определить, в какое место основной памяти следует помещать поступающие программы и данные при распределении памяти неперемещаемыми разделами. Наиболее часто применяются следующие стратегии:

- размещение с выбором первого подходящего (стратегия "первый подходящий");
- размещение с выбором наиболее подходящего (стратегия "самый подходящий");
- алгоритм с выбором наименее подходящего (стратегия "самый неподходящий").

Стратегия "первый подходящий": состоит в выполнении следующих шагов:

1) упорядочить таблицу свободных областей в порядке возрастания адресов;
2) поместить информацию в первый встретившийся участок основной памяти размером не менее требуемого.

Стратегия "самый подходящий": реализует следующую последовательность действий:

1) упорядочить таблицу свободных областей в порядке возрастания размеров свободных областей;
2) поместить информацию в первый встретившийся участок свободной памяти размером не менее требуемого.

Стратегия "самый неподходящий": выполняет следующие действия:

1) упорядочить таблицу свободных областей в порядке убывания размеров областей;
2) поместить информацию в первый встретившийся участок свободной памяти размером не менее требуемого.

Строгих доказательств преимуществ той или иной стратегии перед остальными не существует, так что их применение в операционных системах основано на интуитивных аргументах разработчиков ОС.

4.3. ОРГАНИЗАЦИЯ ВИРТУАЛЬНОЙ ПАМЯТИ

(с использованием дискового пространства)

4.3.1. Основные концепции виртуальной памяти

Термин ***виртуальная память*** обычно ассоциируется с возможностью адресовать пространство памяти, гораздо большее, чем емкость первичной (реальной, физической) памяти конкретной вычислительной машины. Концепция виртуальной памяти впервые была реализована в машине, созданной в 1960 г. в Манчестерском университете (Англия). Однако широкое распространение системы виртуальной памяти получили лишь в ЭВМ четвертого и последующих поколений.

Существует два наиболее известных способа реализации виртуальной памяти – ***страничный*** и ***сегментный***. Применяется также их комбинация – ***странично-сегментная*** организация виртуальной памяти.

Все системы виртуальной памяти характеризуются тем, *что адреса*, *формируемые выполняемыми программами*, *не обязательно совпадают с адресами первичной памяти*. Виртуальные адреса, как правило, представляют гораздо большее множество адресов, чем имеется в первичной памяти.

Суть концепции виртуальной памяти заключается в том, что адреса, к которым обращается выполняющийся процесс, отделяются от адресов, реально существующих в первичной памяти.

Адреса, на которые делает ссылки выполняющийся процесс, называются ***виртуальными адресами***. Адреса, которые реально существуют в первичной памяти, называются ***реальными (физическими) адресами***.

Диапазон виртуальных адресов, к которым может обращаться выполняющийся процесс, называется ***пространством виртуальных адресов*** *V* этого процесса. Диапазон реальных адресов, существующих в конкретной вычислительной машине, называется ***пространством реальных адресов*** *R* этой ЭВМ.

Несмотря на то, что процессы обращаются только к виртуальным адресам, в действительности они должны работать с реальной памятью. Для установления соответствия между виртуальными и реальными адресами разработаны механизмы динамического преобразования адресов ДПА (или DAT – от англ. Dynamics Adress Transformation), обеспечивающие преобразование виртуальных адресов в реальные во время выполнения процесса. Все подобные системы обладают общим свойством – *смежные адреса виртуального адресного пространства процесса не обязательно будут смежными в реальной памяти*. Это свойство называют "искусственной смежностью". Тем самым пользователь освобождается от необходимости рассматривать физическую память с ее уникальными характеристиками.

Виртуальная память строится, как правило, по двухуровневой схеме.

Первый уровень – это реальная память, в которой находятся выполняемые процессы и в которой должны размещаться данные, к которым обращаются эти процессы.

Второй уровень – это внешняя память большой емкости, например накопители на магнитных дисках, способные хранить программы и данные, которые не могут все сразу уместиться в реальной памяти из-за ограниченности ее объема. Память второго уровня называют *вторичной* или *внешней.*

Механизм динамического преобразования адресов ведет учет того, какие ячейки виртуальной памяти в данный момент находятся в реальной памяти и где именно они размещаются. Это осуществляется с помощью таблиц отображения, ведущихся механизмом ДПА.

Информация, перемещаемая из виртуальной памяти в реальную, механизмом ДПА группируется в *блоки*, и система следит за тем, в каких местах реальной памяти размещаются различные блоки виртуальной памяти. Размер блока влияет на то, какую долю реальной памяти ДПА будет использовать непроизводительно, для своих целей.

Если блоки имеют одинаковый размер, то они называются *страницами*, а соответствующая организация виртуальной памяти называется *страничной.* Если блоки могут быть различных размеров, то они называются *сегментами*, а соответствующая организация виртуальной памяти называется *сегментной*. В некоторых системах оба подхода комбинируются, т.е. сегменты реализуются как объекты переменных размеров, формируемые из страниц фиксированного размера. Такая организация виртуальной памяти называется либо *сегментно-страничной*, либо *странично-сегментной*.

4.3.2. Схема прямого отображения адресов

Адреса в системе поблочного отображения являются *двухкомпонентными* (*двумерными*). Чтобы обратиться к конкретному элементу данных, программа указывает блок, в котором расположен этот элемент, и смещение элемента относительно начала блока. Виртуальный адрес ν указывает при помощи упорядоченной пары (b, d), где b – номер блока, в котором размещается соответствующий элемент данных, а d – смещение относительно начального адреса этого блока.

Преобразование адреса виртуальной памяти $\nu = (b, d)$ в адрес реальной памяти r осуществляется следующим образом (рис. 4.6).

Каждый процесс имеет собственную *таблицу отображения блоков,* которую операционная система ведет в реальной памяти. Реальный адрес в этой таблицы загружается в специальный регистр центрального процессора, называемый *регистром начального адреса таблицы отображения блоков процесса*.

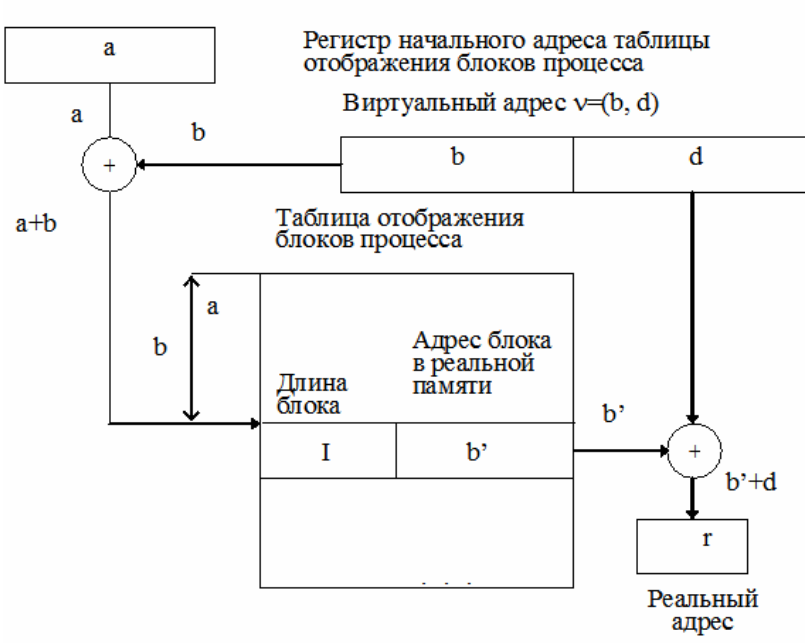

Рис. 4.6. Преобразование виртуального адреса

Таблицы отображения блоков содержат по одной строке для каждого блока процесса, причем эти блоки идут последовательно: сначала блок 0, затем блок 1 и т.д. Номер блока *b* суммируется с начальным адресом *a* таблицы, образуя реальный адрес строки таблицы для блока *b*. Найденная строка содержит реальный адрес *b* начала блока *b* в реальной памяти. К этому начальному адресу *b′* прибавляется смещение *d*, так что образуется искомый реальный адрес $r = b' + d$.

Все методы поблочного отображения, применяемые в системах с сегментной, страничной и комбинированной странично-сегментной организациями, подобные рассмотренной схеме отображения, называемой *схемой прямого отображения*.

4.3.3. Отображения адресов при страничной организации виртуальной памяти

Виртуальный адрес при чисто страничной организации памяти – это упорядоченная пара (*p*, *d*), где *p* – номер страницы в виртуальной памяти, а *d* – смещение в рамках страницы *p*. Процесс может выполняться, если его текущая страница находится в первичной памяти. Страницы переписываются из внешней памяти в первичную и размещаются в ней в блоках, называемых *страничными кадрами* и имеющих точно такой же размер, как у поступающих страниц. Страничные кадры начинаются в реальной памяти с адресов, кратных фиксированному размеру страниц. *Поступающая страница может быть помещена в любой свободный страничный кадр*.

Для обеспечения работы механизма отображения страниц формируется таблица отображения страниц, каждая строка которой содержит информацию об отображаемой странице виртуальной памяти; *r* – признак наличия страницы в первичной памяти (*r* = 0 – страницы в первичной памяти нет; 1 – страница находится в первичной памяти); *S* – адрес страницы во внешней памяти (при *r* = 0); *p′* – номер страничного кадра в первичной памяти, где размещена виртуальная страница с номером *p*.

4.3.4. Отображения адресов при сегментной организации виртуальной памяти

Виртуальный адрес при сегментной организации виртуальной памяти – это упорядоченная пара $v = (s, d)$, где *s* – номер сегмента виртуальной памяти, а *d* – смещение в рамках этого сегмента. Процесс может выполняться только в том случае, если его текущий сегмент находится в первичной памяти, Сегменты передаются из внешней памяти в первичную целиком. Все ячейки, относящиеся к сегменту, занимают смежные адреса первичной памяти. Для размещения поступающих из внешней памяти сегментов в свободные участки первичной памяти применяются те же стратегии размещения, как и при распределении переменными неперемещаемыми разделами – "первый подходящий", "самый подходящий", "самый неподходящий". Динамическое преобразование виртуальных адресов в реальные адреса осуществляется в соответствии со схемой прямого отображения.

4.3.5. Отображения адресов при странично-сегментной организации виртуальной памяти

Системы со странично-сегментной организацией обладают достоинствами обоих способов реализации виртуальной памяти. Сегменты обычно содержат целое число страниц, причем не обязательно, чтобы все страницы сегмента находились в первичной памяти одновременно, а смежные страницы виртуальной памяти не обязательно должны оказаться смежными в первичной памяти. В системе со странично-сегментной организацией применяется трехкомпонентная (трехмерная) адресация. Виртуальный адрес v здесь определяется как упорядоченная тройка $v = (s, p, d)$, где *s* – номер сегмента, *p* – номер страницы, а *d* – смещение в рамках страницы, где находится нужный элемент.

Операционная система для каждого процесса формирует, во-первых, одну таблицу сегментов процесса, и, во-вторых, таблицы страниц сегментов (по одной на каждый сегмент процесса).

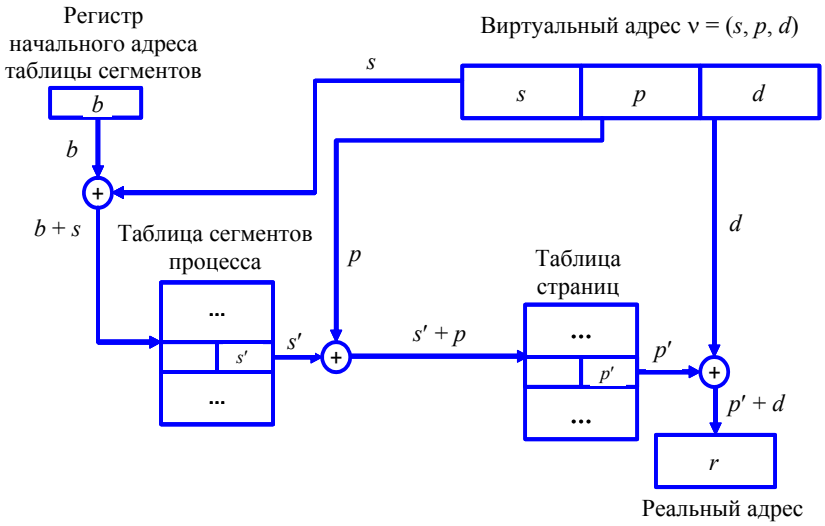

Рис. 4.7. Отображения адресов при странично-сегментной организации виртуальной памяти

Таблица сегментов процесса содержит в своих строках информацию о количестве страниц в сегменте и о начальных адресах s' размещения таблиц страниц сегментов в первичной памяти ЭВМ.

Каждая страница таблиц сегмента содержит в своих строках информацию о начальном адресе p' размещения в первичной памяти страничного кадра для данной страницы виртуальной памяти.

Динамическое преобразование виртуальных адресов в системах со странично-сегментной организацией отличается от преобразования по схеме наличием еще одного уровня вычисления адреса (см. рис. 4.7) и появлением таблиц страниц для каждого сегмента процесса.

4.4. УПРАВЛЕНИЕ ВИРТУАЛЬНОЙ ПАМЯТЬЮ

4.4.1. Стратегии управления виртуальной памятью

Стратегии управления виртуальной памятью так же, как и стратегии управления физической памятью, разделяются на три категории: стратегии вталкивания, стратегии размещения и стратегии выталкивания.

Целью ***стратегий вталкивания*** является определить, в какой момент следует переписать страницу или сегмент из вторичной памяти в первичную.

Целью ***стратегий размещения*** является определить, в какое место первичной памяти помещать поступающую страницу или сегмент.

Целью ***стратегий выталкивания*** является решить, какую страницу или сегмент следует удалить из первичной памяти, чтобы освободить место для помещения поступающей страницы или сегмента, если первичная память полностью занята.

Большинство стратегий управления виртуальной памятью базируется на концепции ***локальности***.

Суть концепции *локальности* заключается в том, что *распределение запросов процессов на обращение к памяти имеет, как правило, неравномерный характер с высокой степенью локальной концентрации*.

Свойство локальности проявляется как во времени, так и в пространстве.

Локальность во времени означает, что к ячейкам памяти, к которым недавно производилось обращение, с большой вероятностью будет обращение в ближайшем будущем.

Локальность в пространстве означает, что обращения к памяти, как правило, концентрируются так, что в случае обращения к некоторой ячейке памяти с большой вероятностью можно ожидать обращение к близлежащим ячейкам.

Свойство локальности наблюдается не только в прикладных программах, но и в работе программ операционной системы. Свойство это скорее эмпирическое (наблюдаемое на практике), чем теоретически обоснованное. Локальность никак нельзя гарантировать, однако ее вероятность достаточно велика. Самым важным следствием локализации является то, что программа может эффективно работать, если в первичной памяти находится подмножество, включающее наиболее "популярные" ее страницы или сегменты.

4.4.2. СТРАТЕГИИ ВТАЛКИВАНИЯ (подкачки)

Для управления вталкиванием применяются следующие стратегии:

– вталкивание (подкачка) по запросу (по требованию);
– вталкивание (подкачка) с упреждением (опережением).

Вталкивание (подкачка) по запросу предполагает, что система ждет ссылки на страницу или сегмент от выполняющегося процесса и только после появления такой ссылки начинает переписывать данную страницу или сегмент в первичную память. Подкачка по запросу имеет положительные и отрицательные стороны.

К положительным сторонам относятся:

1) гарантировано, что в первичную память будут переписываться только те страницы (сегменты), которые необходимы для работы процесса;

2) накладные расходы на то, чтобы определить, какие страницы или сегменты следует передавать в первичную память, минимальны.

К недостаткам подкачки по запросу относится тот факт, что процесс в этом случае накапливает в первичной памяти требуемые ему страницы (сегменты) по одной. При появлении ссылки на каждую новую страницу (сегмент) процессу приходится ждать, когда эта страница (или сегмент) будет передана в первичную память. В зависимости от того, сколько страниц (сегментов) данного процесса уже находится в первичной памяти, эти периоды ожидания будут обходиться все более дорого, поскольку ожидающие процессы будут занимать все больший объем памяти.

Вталкивание (подкачка) с упреждением предполагает, что система пытается заблаговременно определить, к каким страницам или сегментам будет обращаться процесс. Если вероятность обращения высока и в первичной памяти имеется свободное место, то соответствующие страницы или сегменты будут переписываться в первичную память еще до того, как к ним будет явно производиться обращение. При правильном выборе страниц (сегментов) для упреждающей подкачки удается существенно сократить общее время выполнения данного процесса и уменьшить значение показателя "пространство-время".

К недостаткам стратегии подкачки с упреждением можно отнести тот факт, что точно предсказать путь, по которому будет развиваться процесс, в общем случае невозможно, поэтому вполне возможны ситуации, когда решения о выборе станиц (сегментов) для упреждающей подкачки будет в большинстве случаев приниматься неверно для одного или нескольких процессов, развивающихся в системе, что, в свою очередь, приведет к резкому снижению скорости работы этих процессов из-за увеличения времени ожидания необходимых им страниц или сегментов.

4.4.3. Стратегии размещения

В системах со *страничной* организацией виртуальной памяти решение о размещении вновь загружаемых страниц принимается достаточно просто: *новая страница может быть помещена в любой свободный страничный кадр*.

Для систем с *сегментной* организацией виртуальной памяти применяются такие же стратегии размещения, какие используются в системах распределения памяти переменными разделами (см. п. 4.2): размещение с выбором первого подходящего свободного участка; размещение с выбором самого подходящего свободного участка; размещение с выбором наименее подходящего свободного участка.

4.4.4. Стратегии выталкивания

В мультипрограммных системах вся первичная память бывает, как правило, занята. В этом случае программа управления памятью должна решать, какую страницу или какой сегмент следует удалить из первичной памяти, чтобы освободить место для поступающей страницы или сегмента. В настоящее время применяются следующие стратегии выталкивания (откачки) страниц (сегментов):

- выталкивание случайных страниц или сегментов;
- выталкивание первой пришедшей страницы или сегмента (FIFO);
- выталкивание дольше всего не использовавшихся страниц или сегментов (LRU – Least Recently Used);
- выталкивание наименее часто использовавшихся страниц или сегментов (LFU – Least Frequently Used);
- выталкивание не использовавшихся в последнее время страниц или сегментов (NRU – Not Recently Used).

Стратегия выталкивания случайных страниц или сегментов является наиболее простой в реализации, обладает малыми издержками и не является дискриминационной по отношению к каким-либо процессам, работающим в системе. В соответствии с этой стратегией любые страницы или сегменты, находящиеся в первичной памяти, могут быть выбраны для выталкивания с равной вероятностью, в том числе даже следующая страница или сегмент, к которым будет производиться обращение (и которые, естественно, удалять из памяти наиболее нецелесообразно). Поскольку подобная стратегия, по сути, рассчитана на "слепое" везение, в реальных системах она применяется редко.

Стратегия выталкивания первой пришедшей страницы или сегмента (FIFO-стратегия) реализует принцип "первый пришел – первый ушел". В этом случае в момент поступления каждой страницы (сегмента) в первичную память ей (ему) присваивается метка времени. Когда появляется необходимость удалить из первичной памяти какую-либо страницу (сегмент), выбирается та страница (сегмент), у которой метка времени имеет наименьшее значение.

Стратегия выталкивания дольше всего не использовавшихся страниц или сегментов (LRU-стратегия) предусматривает, что для выталкивания следует выбирать те страницы (сегменты), которые не использовались дольше других. Стратегия LRU требует, чтобы при каждом обращении к страницам (сегментам) их метки времени обновлялись. Это может быть сопряжено с существенными издержками, поэтому LRU-стратегия, несмотря на свою привлекательность, в современных операционных системах реализуется достаточно редко. Кроме того, при реализации LRU-стратегии может быть так, что страница (сегмент), к которой дольше всего не было обращений, в действительности станет следующей используемой страницей (сегментом), если программа к этому моменту очередной раз пройдет большой цикл, охватывающий несколько страниц или сегментов.

Стратегия выталкивания реже всего используемых страниц или сегментов (LFU-стратегия) является одной из наиболее близких к рассмотренной выше LRU-стратегии. В соответствии с LFU-стратегией из первичной памяти выталкиваются наименее часто (наименее интенсивно) использовавшиеся к данному времени страницы или сегменты. Здесь контролируется интенсивность использования страниц (сегментов). Для этого каждой странице (сегменту) назначается счетчик, значение которого увеличивается на единицу при каждом обращении к данной странице (сегменту). LFU-стратегия, будучи интуитивно оправданной, имеет те же недостатки, что и стратегия LRU: во-первых, велика вероятность того, что из первичной памяти будут удалены страницы или сегменты, которые потребуются процессам при следующем обращении к памяти, и,

во-вторых, ее реализация может быть сопряжена со значительными затратами на организацию контроля интенсивности использования страниц или сегментов.

Стратегия выталкивания не использовавшихся в последнее время страниц или сегментов (NRU-стратегия) также является близкой к стратегии LRU и характеризуется относительно небольшими издержками на свою реализацию. Согласно NRU-стратегии из первичной памяти выталкиваются те страницы (сегменты), к которым не было обращений в последнее время. В соответствии со свойством локальности во времени (см. п. 4.4.1) к страницам (сегментам), не использовавшимся в последнее время, вряд ли будет обращение в ближайшем будущем, так что их можно заменить на вновь поступающие страницы.

Поскольку желательно заменять те страницы (сегменты), которые в период нахождения в основной памяти не изменялись, реализация NRU-стратегии предусматривает введение двух аппаратных бит-признаков на страницу (сегмент):

- бит-признак b_0 обращения к странице (сегменту);
- бит-признак b_1 модификации страницы (сегмента).

Первоначально все b_0 и b_1 устанавливаются в 0. При обращении к странице (сегменту) соответствующий бит-признак b_0 устанавливается в 1. В случае изменения содержимого страницы (сегмента) соответствующий бит-признак b_1 устанавливается в 1. NRU-стратегия предусматривает существование четырех групп страниц (сегментов), показанных в табл. 4.5.

В первую очередь из первичной памяти выталкиваются страницы (сегменты), принадлежащие группам с меньшими номерами. Учет времени, в течение которого к страницам (сегментам) не было обращений, осуществляется периодическим сбрасыванием в 0 всех битов-признаков, выполняемым операционной системой. Практически любая стратегия выталкивания страниц (сегментов) не исключает опасности нерациональных решений. Это объясняется тем, что операционная система не может точно прогнозировать будущее поведение любого из процессов, поступивших к ней на обработку.

4.5. Группы страниц (сегментов)

Группа	b_0	b_1
1	0	0
2	1	0
3	0	1
4	1	1

Контрольные вопросы к теме 4

1. Описать иерархию организации памяти.
2. Перечислить и охарактеризовать методы связного распределения памяти. Где они применяются?
3. Охарактеризовать стратегии размещения при связном распределении памяти.
4. Что такое виртуальная память? Каковы свойства различных видов организации виртуальной памяти?
5. Описать способы вычисления адреса при страничной, сегментной и странично-сегментной организации виртуальной памяти.
6. Перечислить и охарактеризовать стратегии управления виртуальной памятью.
7. Сформулировать и пояснить принцип локальности.
8. Перечислить и охарактеризовать стратегии вталкивания, размещения и выталкивания.

Т Е М А 5. Управление файлами и вводом-выводом в операционных системах

5.1. МЕТОДЫ ОРГАНИЗАЦИИ ДАННЫХ В ОПЕРАЦИОННЫХ СИСТЕМАХ

Данные – это информация, представленная в виде, пригодном для обработки автоматическими средствами при возможном участии человека.

Любые данные, подлежащие обработке, находятся на том или ином носителе.

Носитель данных – это материальный объект, предназначенный для хранения данных.

Носителями данных могут служить магнитные диски и ленты, перфокарты и перфоленты, компакт-диски и т.п. Кроме того, данные могут поступать в систему обработки от источников данных, не являющихся средствами хранения информации. Примерами такого рода источников данных являются линии передачи информации, аналого-цифровые преобразователи, к входам которых подключены выходы измерительных приборов и систем и т.п.

Источник данных – функциональное устройство, являющееся источником передаваемых данных.

Источник информации – часть коммуникационной системы, которая порождает сообщение.

Любые процессы обработки информации представляют собой преобразование данных из одного представления (структуры) в другое представление (структуру). При этом структуры данных могут анализироваться в двух аспектах: с позиций различных категорий пользователей и с позиции самой системы обработки данных.

В первом случае понятие структуры данных есть не что иное, как представление пользователя о данных вне связи с методами хранения данных.

Во втором случае на первое место выдвигаются вопросы, связанные с представлением данных в памяти ЭВМ, т.е. вопросы отображения структур данных в физическую запоминающую среду, определяемые конкретным набором структур данных, характеристиками устройств памяти, взаимосвязями между структурами данных и структурами хранения.

Операционные системы манипулируют информацией, представляющей собой отображение данных на заданную вычислительную среду и организованную соответствующим образом.

Организация данных – это представление данных и управление данными в соответствии с определенными соглашениями.

Одна и та же структура данных может отображаться в среду хранения несколькими способами, каждый из которых будет соответствовать определенной структуре хранения. Каждая структура хранения материализуется в среде хранения в виде одного или нескольких наборов данных.

Набор данных – идентифицированная совокупность физических записей, организованная одним из установленных в системе обработки данных способов и представляющая файлы или части файлов в среде хранения.

В ОС используются в основном последовательная, древовидная и прямая организация данных.

При последовательной организации данных ОС манипулирует с последовательным набором данных (рис. 5.1).

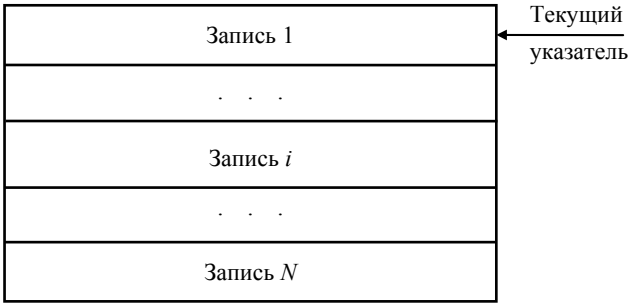

Рис. 5.1. Последовательная организация данных

Последовательный набор данных – набор данных, к физическим записям которого обеспечивается лишь последовательный доступ в порядке их размещения во внешней памяти.

Последовательный доступ к порции данных – доступ к порции данных, при котором операции чтения или запись порции данных, к которым осуществляется доступ, проводится после чтения или записи всех порций, расположенных до этой порции в соответствии с порядком, фиксированным для определенной совокупности порций данных.

Различают физически последовательные и логически последовательные наборы данных.

Физически последовательные наборы данных предполагают размещение составляющих в памяти сплошным массивом.

Логически последовательные наборы данных могут не иметь в памяти представления в виде сплошного массива.

В качестве носителей последовательных наборов данных применяются, например, магнитные диски, магнитные ленты, перфоленты, перфокарты.

Древовидная организация данных основана на представлении структуры данных в виде деревьев, причем каждый узел дерева является частью структуры (рис. 5.2).

Методы представления древовидных структур в памяти ЭВМ могут быть разделены на две группы:

1) методы физически последовательного размещения данных;
2) методы логически последовательного размещения данных.

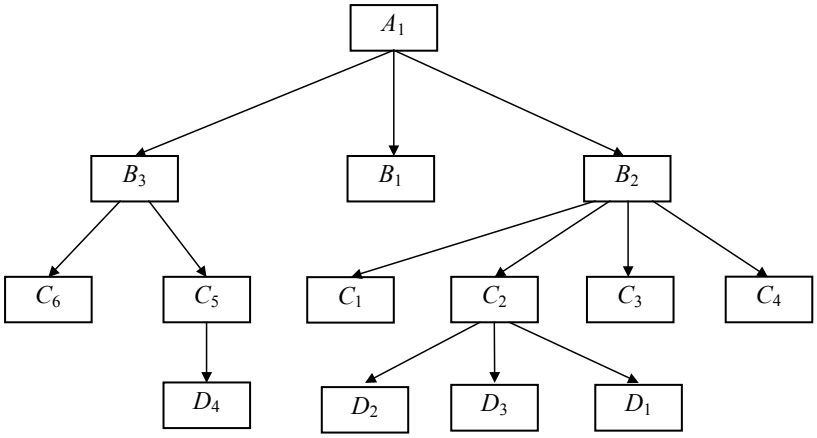

Рис. 5.2. Древовидная организация данных

A_1	B_3	C_6	C_5	D_4	B_1	B_2	C_1	C_2	D_2	D_3	D_1	C_3	C_4

Рис. 5.3. Физически последовательное отображение древовидных структур

При *физически последовательном размещении* древовидной структуры данных структура хранения начинается узлом, соответствующем вершине дерева (рис. 5.3). По иерархии за ним следуют узлы на самой левой ветви дерева. В пределах узлов одного уровня они отображаются слева направо.

Подобная организация структуры хранения является единственной при использовании чисто последовательных сред хранения, например, магнитных лент. При размещении составляющих древовидной структуры данных они могут иметь метки или специальные разграничители, позволяющие установить, к какому уровню дерева относится та или иная составляющая структуры данных.

При ***логически последовательном размещении*** древовидной структуры возможно два способа отображения дерева на среду хранения.

В первом способе (рис. 5.4) порции данных, соответствующие каждому узлу дерева, снабжаются ссылочной информацией на порции данных, связанные с ним и находящиеся ниже по иерархии (т.е. на порожденные этим узлом данные).

При втором способе для отображения дерева формируют специальные таблицы, в которые заносится информация об узлах дерева и связях между ними.

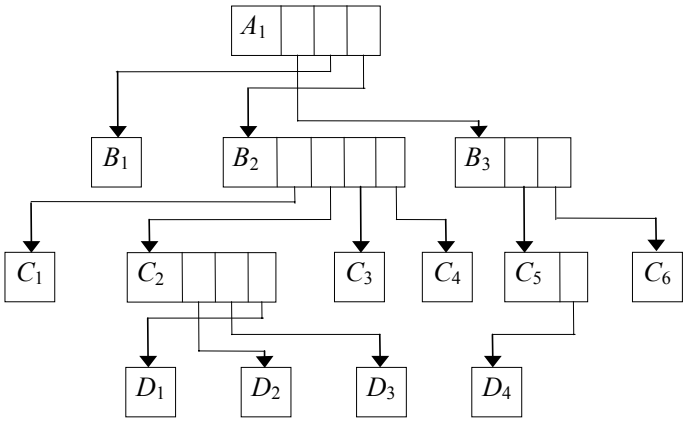

Рис. 5.4. Логически последовательное отображение древовидной структуры с указателями на порожденные записи

Оба эти способа используют представление информации в виде связных списков, каждый элемент которых наряду с данными о текущем узле дерева содержит информацию о связанных с этим узлом других узлов древовидной структуры.

Прямая организация данных рассчитана на произвольную обработку информации. Основными отличительными особенностями прямой организации данных являются: не используется система индексов; способ размещения данных на носителе определяет пользователь в своих рабочих программах. Прямая организация данных характеризуется тем, что идентификатор порции данных (ключ) используется для установления адреса этой порции. В наборах данных с прямой организацией для указания местоположения порций данных применяется два способа адресации: абсолютная адресация порций данных; относительная адресация порций данных.

Способ абсолютной адресации основан на том, что адрес каждой порции данных в наборе является ее физическим адресом на носителе данных. Для наборов данных на магнитном диске абсолютный адрес включает в себя, например номер цилиндра, номер дорожки, номер сектора, номер блока и т.п. – в зависимости от типа накопителя на магнитных дисках. Недостатком способа прямой адресации является жесткость привязки порций данных к месту на носителе и, следовательно, проблематичность их перемещения на внешних запоминающих устройствах.

Способ относительной адресации основан на использовании порядковых номеров порций данных, отсчитываемых от начала набора данных. Эти порядковые номера используются либо самостоятельно, либо в сочетании с номерами порций данных на дорожках магнитных дисков. Существует три разновидности реализации способа относительной адресации: метод ключа (метод прямой адресации); метод косвенной адресации; метод адресных таблиц (метод перекрестных ссылок).

Метод прямой адресации является наиболее распространенным и эффективным, так как он реализуется самым простым образом при произвольном порядке работы с порциями набора данных прямой организации. В этом методе адресом любой порции данных является некоторое натуральное число, представляющее собой порядковый номер порции относительно начала набора данных. Очень часто в качестве такого номера используется некоторое поле порции данных (ключ или признак) либо в "чистом" виде, либо после минимальных преобразований. Например, из текущего значения признака вычитается его минимально возможное значение. Результат вычитания принимается как относительный номер порции в наборе данных. Прямой метод обеспечивает взаимно-однозначное соответствие между относительным номером порции данных и ее относительным физическим адресом в наборе.

Основными недостатками метода прямой адресации являются:

1) необходимость использования порций данных одинаковой длины;

2) перед созданием набора данных необходимо выполнить подготовительные действия по разметке отведенного для него участка на носителе данных;

3) если значения признака, задающего адрес порции данных, расположены неравномерно, то набор данных будет неэффективно использовать выделенное ему пространство памяти на носителе, поскольку внутри набора данных образуются пустоты из-за наличия неиспользованных значений признака.

Метод косвенной адресации основан на применении преобразований, выполняемых над ключом порции данных, для получения адреса этой порции. Эти преобразования значительно сложнее, чем в методе прямой адресации, так как имеют более общий характер. Преобразования ключа в методе косвенной адресации носят название *рандомизации.* Косвенная адресация эффективна, когда диапазон изменения значения ключей значительно шире диапазона количества порций данных в наборе и, соответственно, диапазона возможных адресов. Несомненным достоинством метода косвенной адресации является высокая плотность заполнения памяти на носителе даже при существенно неравномерном распределении значений ключей в их диапазоне изменения. Недостатками этого метода являются, во-первых, высокая вероятность появления синонимов адресов, т.е. ситуации, когда на одно и то же место в памяти претендуют две и более порции данных с различными значениями ключей, и, во-вторых, отсутствие возможности восстановить значение ключа по значению адреса порции данных, как это можно сделать в методе прямой адресации.

Метод адресных таблиц универсален и характеризуется тем, что порции данных заносятся в набор в произвольном порядке, а в памяти создается таблица соответствия значений ключей этих порций их физическим адресам. Вся дальнейшая работа с данными реализуется через эти таблицы.

Логической единицей любого набора данных является запись.

Запись – совокупность данных, которая используется средствами системы как единое целое.

Запись может быть фиксированной, переменной или неопределенной длины.

Запись фиксированной длины – это логическая запись, длина которой задана вне этой записи.

Запись переменной длины – логическая запись, длина которой определяется значением одного из ее полей.

Запись неопределенной длины – логическая запись, в которой отсутствует специализированное поле для описания ее длины, а длина определяется в момент обработки этой записи.

5.2. МЕТОДЫ ДОСТУПА К ДАННЫМ

Управление доступом к данным, размещенным на внешних запоминающих устройствах (ВЗУ), состоящим в выполнении операций передачи данных от внешних устройств в основную память (операция ввода) или пересылки данных из основной памяти на внешнее устройство (операция вывода), может быть выполнено с применением методов прямого и косвенного управления доступом.

Метод прямого управления доступом к данным основан на наличии непосредственной связи между центральным процессором и внешним запоминающим устройством (см. рис. 5.5, *а*).

На центральный процессор возлагается обязанность непосредственно управления работой устройства, что предполагает наличие в составе команд процессора специальных команд по управлению работой этого устройства (инициирование ВЗУ, проверка готовности к работе, остановка ВЗУ, запись/чтение данных и т.п.).

Главным недостатком метода прямого управления доступом является невозможность реализация на его основе режимов мультипрограммной обработки данных.

Метод косвенного управления доступом (рис. 5.5, *б*) основан на том, что между центральным процессором и внешними запоминающими устройствами помещается специальный процессор, называемый каналом ввода-вывода (контроллер ввода-вывода), который осуществляет фактическое управление внешним запоминающим устройством при выполнении операций ввода и вывода данных. На центральный процессор теперь возлагается функция управления каналом ввода-вывода. Синхронизация параллельной работы центрального процессора и канала ввода/вывода осуществляется с применением системы прерываний. Канал через систему прерываний прерывает работу центрального процессора всякий раз при завершении операции ввода-вывода или при условии возникновения ошибок ввода/вывода. Сигнал пребывания является по смыслу синхронизирующим.

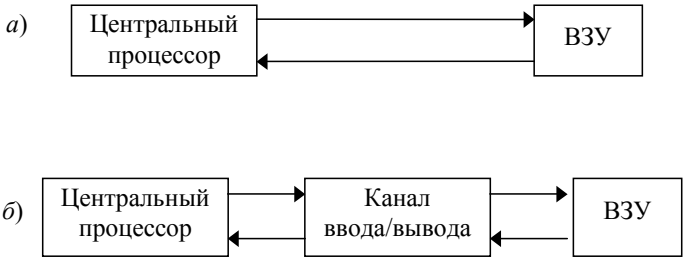

Рис. 5.5. Методы доступа к данным:
а – прямой; *б* – косвенный

Независимо от принятого в системе метода управления доступ к данным в программах пользователя может осуществляться разными методами. Наиболее распространенными в настоящее время являются два доступа к данным из программ пользователя: доступ к данным на низком уровне; доступ к данным на высоком уровне.

Способ доступа на низком уровне, называемый также низкоуровневым доступом или физическим способом доступа, характеризуется тем, что для обеспечения выполнения операции ввода-вывода пользователь обязан самостоятельно создать программу управления соответствующим устройством ввода/вывода.

Доступ к данным на высоком уровне, называемый также высоко уровневым доступом или способом доступа на логическом уровне, предназначен для повышения уровня автоматизации выполнения операций ввода-вывода в программах пользователя. В этом случае пользователь работает с необходимыми ему данными опосредованно, через драйверы операционной системы.

В системах с виртуальной памятью могут применяться ***виртуальные методы доступа***, которые предназначены для обработки наборов данных с виртуальной организацией.

Виртуальная организация наборов данных предполагает разбиение набора на блоки данных размером в один страничный кадр. На носителе данных набор хранится в виде страниц того же размера. В каждой странице размещается один блок набора данных с виртуальной организацией.

5.3. ОБЪЕДИНЕНИЕ ЗАПИСЕЙ В БЛОКИ И БУФЕРИЗАЦИЯ

Для сглаживания эффекта несоответствия скоростей между внутренними и внешними процессами в системах управления вводом-выводом применяют три основных метода: синхронизация по прерываниям ввода-вывода; буферизация ввода-вывода; блокирование данных.

Буферизация ввода-вывода основана на размещении между внешним и внутренним процессами одного или нескольких буферов, роль которых выполняют, как правило, непрерывные области первичной памяти.

Буфер является критическим ресурсом для связанных с ним внутренних и внешних процессов (рис. 5.6). Введение буферов как средства информационного взаимодействия выдвинуло задачу управления буферами, которая решается средствами супервизора ввода-вывода. На супервизор ввода-вывода возлагаются функции по выделению и уничтожению буферов в первичной памяти, синхронизации обращения к буферам внутренних и внешних процессов, устранения одновременного обращения к буферу этих процессов и т.п.

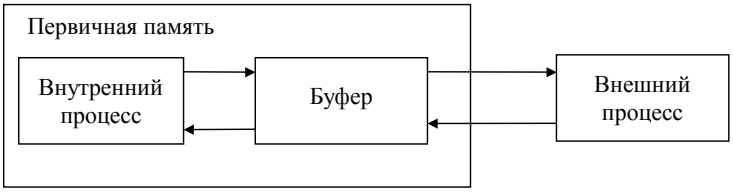

Рис. 5.6. Буферизация ввода-вывода

При решении задачи буферизации важным является определение количества буферов, закрепляемых за отдельным каналом или устройством, а также размер области первичной памяти, отводимой под каждый буфер.

Важным фактором при управлении буферами является также оперативность обновления информации в буфере. Не всегда является необходимым стремление наиболее быстрой передачи данных из заполненного буфера внешнему процессу. Иногда такую передачу целесообразно задержать на достаточно длительный интервал времени, поскольку хранимые в буфере данные могут понадобиться внутреннему процессу. В этом случае буфер ввода-вывода превращается в программный аналог кэш-памяти при работе с внешними устройствами. Задержка информации в буфере предполагает возможность обращения к ней со стороны программных (внутренних) процессов. Если бы эта информация была "сброшена" на внешнее запоминающее устройство сразу по мере заполнения буфера, то повторное обращение к ней со стороны внутреннего процесса потребовало бы обращение к этому устройству с операцией чтения данных. Оперативность в этом случае была бы существенно ниже, чем в случае использования программной кэш-памяти.

Блокирование данных – это операция объединения порций данных, которыми оперирует внутренний процесс (логические записи), в более крупные образования – блоки логических записей. Такие блоки логических записей называются физическими записями. Физическая запись является единицей обмена данными между первичной памятью и внешним устройством. Каждая физическая запись (блок) представляется на внешнем устройстве непрерывной областью. Чем больше длина блока, тем меньше непроизводительных затрат времени на выполнение операций ввода-вывода больших массивов данных и тем большее время канал работает автономно от центрального процессора. Однако с увеличением количества логических записей в блоке возрастают затраты времени на выполнение дополнительных операций по блокированию данных (при выводе) и деблокированию данных (при вводе). Кроме того, при этом возрастают требования к объему необходимой для реализации этих операций первичной памяти.

С другой стороны, длинные блоки "выгодны" для хранения на внешних запоминающих устройствах, так как количество памяти, отводимое для фиксации межблочных промежутков, в этом случае меньше, чем для коротких блоков (при одном и том же количестве хранимых полезных данных).

Противоречивые требования к длине физических записей делают проблематичным однозначный выбор длины блока при организации обмена данными между первичной памятью и внешним устройством. Поэтому в операционных системах решение этой проблемы нашли в следующем. В некоторых ОС, например, в MS DOS, Windows, Unix, размер блока определяется исходя из технических характеристик внешнего запоминающего устройства и контроллеров ввода-вывода. В других операционных системах определение параметров блокирования данных может быть возложено на программиста.

5.4. УПРАВЛЕНИЕ ФАЙЛАМИ

5.4.1. Понятие файлового способа хранения данных и файловой системы

С появлением в составе ЭВМ внешних запоминающих устройств, способных хранить огромные массивы информации в течение длительного времени, привело к необходимости разработки такого способа хранения и управления данными, при котором затраты на доступ к информации со стороны разработчиков прикладных систем и программ были бы сведены к минимуму.

В многопользовательских вычислительных системах вторичная (внешняя) память должна быть так же разделяема между пользователями, как и первичная. Такое разделение в современных операционных системах обеспечивается с использованием файлового способа хранения данных.

Файловый способ хранения данных – это способ хранения данных, при котором каждый набор данных представляются как именованное, возможно, защищенное, собрание записей, называемой файлом.

Файл – идентифицированная совокупность экземпляров полностью описанного в конкретной программе типа данных, находящихся вне программы во внешней памяти и доступных программе посредством специальных операций.

Файловая система – система управления данными с файловым способом хранения.

В результате применения файлового способа хранения данных пользователь получает виртуальное представление внешней памяти и работает с ней не в терминах команд управления конкретными физическими устройствами внешней памяти, а в терминах, обусловленных особенностями структуры и состава его конкретных наборов данных. Пользователь видит виртуальную внешнюю память как среду, способную хранить его обособленные и поименованные информационные объекты, имеющие определенную внутреннюю структуру. Среда должна обеспечить возможность хранения произвольного количества файлов без ограничения объема, причем пользователь должен иметь возможность доступа как к отдельным файлам, так и к их составным частям, с учетом логической структуры.

Файловые системы могут быть простыми и сложными. Их природа зависит от разнообразия применений и среды, в которой будет использоваться операционная система. В общем случае к файловой системе предъявляют следующие основные требования:

- каждый пользователь должен иметь возможность создавать, удалять и изменять файлы;
- каждый пользователь может иметь контролируемый доступ к файлам других пользователей;
- каждый пользователь может контролировать, какие типы доступа разрешены к его файлам;
- каждый пользователь должен иметь возможность переструктурировать свои файлы к форме, соответствующей его задачи;
- каждый пользователь должен иметь возможность пересылать данные между файлами;
- каждый пользователь должен иметь возможность копировать и восстанавливать свои файлы в случае их повреждения;
- каждый пользователь должен иметь возможность доступа к своим файлам по их символическим именам.

Для того чтобы удовлетворить перечисленные выше требования, программная часть файловых систем должна содержать следующие компоненты:

1) средства взаимодействия с процессами пользователей, обеспечивающие прием и интерпретацию запросов от пользователя на обработку файлов и сообщающие ему о результатах выполненной обработки;
2) средства реализации методов доступа к файлу и к его составным элементам;
3) средства распределения внешней памяти для хранения файлов, а также ее освобождения по мере уничтожения файлов;
4) средства учета расположения файлов и их составных элементов.

Все перечисленные средства составляют логический уровень управления данными в файловой системе. Физическим уровнем в ней является система ввода-вывода. В таком обобщенном виде файловая система выступает как интерфейс между программными процессами и файлами (рис. 5.7).

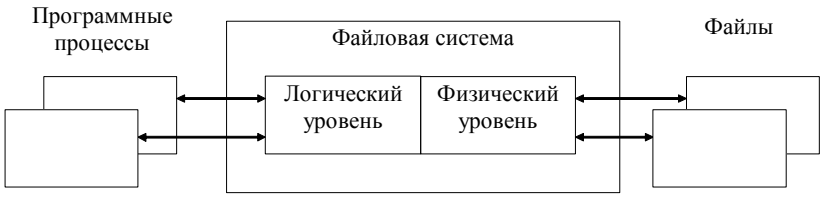

Рис. 5.7. Взаимодействие программных процессов и файлов

Различные подходы к построению файловых систем отличаются уровнем автоматизации действий по управлению данными. Несмотря на имеющиеся различия в построении, все файловые системы имеют совпадающие способы организации хранения файлов во внешней памяти.

5.4.2. Организация файлов

Физическая организация файлов зависит от физических характеристик внешнего устройства. Существуют внешние устройства, которые можно рассматривать как последовательные файлы, где обмен записями доступен только в линейном порядке. К ним относятся накопители на магнитных лентах, стримеры, принтеры, модемы и т.п. По сравнению с ними накопители на магнитных дисках допускают огромную гибкость организации файлов. На поверхности дисковой пластины расположена серия концентрических окружностей, называемых дорожками или цилиндрами диска. Каждая дорожка подразделяется на секторы. Сектор – это минимальный объем информации, обычно передаваемый в дисковой системе.

Физические записи могут располагаться в любом месте диска. Из соображений эффективности при поиске и передаче данных предпринимаются всевозможные попытки расположить связанные физические записи в смежных секторах или на соседних пластинах той же дорожки диска.

Структура диска позволяет системе управления файлами организовать файлы тремя различными способами: последовательным, непрерывным, сегментированным. Каждая организация файлов обладает своими ограничениями и характеристиками производительности.

Последовательная организация файла предполагает создание на диске последовательного файла.

Последовательный файл – файл, к компонентам которого обеспечивается лишь последовательный доступ в соответствии с упорядоченностью этих компонентов.

В последовательных файлах размещают наборы данных с последовательной организацией. Обычно в последовательных файлах используют один указатель от одного блока к другому (рис. 5.8). Иногда для ускорения доступа применяют двойные ссылки.

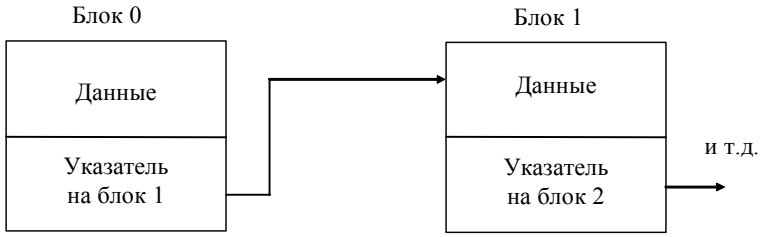

Рис. 5.8. Последовательная организация файла

Последовательный файл состоит из связанной последовательности блоков. В одном блоке последовательного файла может быть размещена одна или несколько записей последовательного набора данных. Доступ к последовательному файлу отличается простотой и достаточно быстр. Для того, чтобы найти нужную запись, файл должен быть просмотрен блок за блоком от начала файла, пока не будет найдена требуемая запись. При обработке последовательного файла в файловых системах рассматривают следующие случаи размещения логической записи в физических блоках:

1) один физический блок под одну логическую запись;
2) несколько физических блоков под одну логическую запись;
3) один физический блок под несколько логических записей.

В первом случае необходимо считывать в первичную память только один физический блок. Логическая запись может быть обновлена на том же месте путем записи новых данных поверх старых. Включить новую запись довольно просто, так как система выделяет еще один физический блок и настраивает соответствующие указатели.

Во втором случае для того, чтобы получить логическую запись целиком, все физические блоки, занятые ею, должны быть считаны в память. Обновление логической записи производится так же, как и в первом случае – путем записи новых данных поверх старых в найденном физическом блоке. Относительно несложно добавить новую логическую запись, так как для этого система должна просто выделить необходимое число блоков для новой записи и настроить соответствующие указатели.

В третьем случае для обеспечения доступа к требуемой записи программа управления файлами должна сначала найти блок, где она находится. После того, как этот блок считан в первичную память, выполняется выделение нужной записи из содержимого блока. Добавить новую запись значительно сложнее, чем в первых двух случаях, так как если блок заполнен, необходимо перенести в новый блок одну или несколько записей из старого блока с соответствующей коррекцией указателей.

Доступ к записи в последовательном файле требует, в среднем, чтения и просмотра половины блоков файла.

Непрерывная организация файла предполагает создание на диске непрерывного файла.

Непрерывный файл – файл на носителе, состоящий из ряда физических блоков, которые все расположены в одной сплошной области дискового пространства.

Непрерывная организация файла представляет собой жесткую структуру, что обеспечивает самый быстрый доступ к данным. Размер непрерывного файла фиксирован и задается в момент создания файла. Файл не может быть увеличен в размере, однако уменьшение длины непрерывного файла допускается. Главный недостаток непрерывной организации файла в том, что во внешней памяти не всегда может быть выделено нужное количество смежных физических блоков.

Доступ к записям непрерывного файла достаточно прост. Если обозначить через r – номер записи, l – длину записи и L – длину блока, то номер b блока, где находится запись, вычисляется по формуле

$$b = lr / L.$$

Значение b используется затем для считывания нужного блока в первичную память. Поскольку не нужно проходить по указателям, как в первом случае, то достигается существенное сокращение затрат времени на доступ к данным.

Обновление записи r совершается очень просто – она просто перекрывается новыми данными. При этом происходит обращение к нужному количеству физических блоков.

Для непрерывного файла имеют место те же три случая размещения логических записей в физических блоках, что и для последовательного файла. Включение новой записи в непрерывный файл очень сложно. Для этого системе, в среднем, необходимо переместить половину записей длиной l, прежде чем освободить место для новой записи. Включение новой записи в непрерывный файл может окончиться неудачей, если при сдвиге записей последняя запись будет передвинута за границу файла.

Сегментированная организация файла предполагает создание сегментированного файла.

Сегментированный файл – это файл, состоящий из последовательного индекса, содержащего адреса соответствующих блоков данных.

Сегментированный файл называют также ***индексным файлом***.

Такая организация предназначена для преодоления недостатков доступа в последовательных файлах. Индекс представляет собой меру произвольности доступа и средство обслуживания добавления в файл.

Индекс строится как множество блоков указателей сегментов (БУС) и может быть организован последовательным или непрерывным способами. При непрерывном подходе максимальный размер файла считается известным (например, один дисковый том) и в соответствии с этим выделяется требуемое количество индексных блоков. В случае последовательного подхода размер файла не ограничивается. Однако поиск по последовательному индексу ставит те же проблемы, что и при

поиске в последовательном файле. Каждый БУС в индексном файле содержит *n* элементов или указателей на блоки данных и, возможно, указатель на следующий индексный блок. Указатель на дисковый блок должен иметь размер, достаточный для представления всего диапазона адресов на запоминающем устройстве.

Получение записи из сегментированного файла требует не более двух обращений к диску. В первую очередь вычисляется адрес индексного блока, и он считывается в память. Затем из него извлекается адрес блока данных, и тот считывается в память. Включение новой записи в сегментированный файл также не вызывает больших затруднений.

5.4.3. Организация хранения файлов

Современные файловые системы ЭВМ предназначены для обслуживания многих тысяч файлов. Поэтому роль механизмов учета в составе файловой системы чрезвычайно важна. В настоящее время общепринятым приемом реализации механизма учета файлов является сведение всей учетной информации о расположении файлов в одну таблицу, называемую каталогом.

Каталог (Catalog) – справочник файлов и каталогов с ссылками на их расположение.

Справочник (Directory) – таблица идентификаторов и ссылок к соответствующим элементам данных.

Каталог может также содержать другую информацию, например, типы устройств, на которых хранятся файлы, пароли, коэффициенты блокирования и т.п.

Доступ к файлам организован в файловых системах только через каталоги. Каталоги обычно располагают на тех же носителях (томах, разделах дисков и т.п.), где расположены файлы (локальная стратегия ведения каталогов).

Структура и содержание каталогов определяется структурой и содержанием учитываемых в них файлов, а также стремление достигнуть должной гибкости в способах доступа к файлам, обеспечить допустимое время доступа.

Современные файловые системы работают с каталогами, имеющими древовидную структуру. При этом различают так называемый *главный* (*корневой*) *каталог* и *подчиненные* (*вложенные*) *каталоги.*

Корневой каталог размещается так, что его местоположение всегда заранее известно файловой системе. Корневой каталог содержит идентификаторы и адреса размещения файлов и каталогов, хранящихся на данном томе внешней памяти. Используя данные, находящиеся в корневом каталоге, файловая система осуществляет доступ к файлу или к вложенному каталогу (если необходимый файл не поименован в корневом каталоге).

Наиболее развитой файловой системой в настоящее время обладает ОС Unix, в которой поддерживается иерархия каталогов, логически связанная в древовидную структуру. Каждый пользователь может работать в составе этой структуры со своей системой каталогов (со своим поддеревом), а также осуществлять переход к другим каталогам. Полное имя файла в данной структуре задает траекторию перехода между каталогами в данной логической структуре каталогов. Каждый каталог рассматривается как файл, имеет собственное имя, доступен непосредственно пользователям для чтения. Запись в каталоги осуществляется по требованию пользователя с помощью специальных системных процедур. Продвижение по каталогу при поиске некоторого каталога или файла возможно как вниз, по дереву от текущего узла, так и вверх, в направлении к корню. Система обеспечивает гибкий и одновременно защищенный доступ к файлам.

Каждый файл, размещенный в рамках некоторого каталога, должен отличаться от другого файла этого же каталога. Поэтому каждому файлу дается *имя,* выражаемое строкой символов. Кроме того, файловые системы наряду с именем поддерживают *расширение имени файла,* которое записывается справа от имени и отделяется от него символом-разделителем (как правило, символ "**.**" – точка). Полное имя файла, содержащее имя и расширение, записывается в качестве идентификатора в каталог файлов.

Наряду с именем и расширением имени файлы могут отличаться друг от друга и другими характеристиками – наборами атрибутов, специфицирующих его тип, защиту, способ буферизации и т.п. Состав набора атрибутов файлов и способ их кодирования не совпадают в различных операционных системах.

Для эффективного управления файлами требуется также несколько блоков структурированных данных, главным из которых является блок управления файлом (БУФ), называемый также дескриптором файла.

Дескриптор файла содержит различные данные, необходимые для выполнения операций с файлом, например, атрибуты файла, дату и время создания файла, размер файла, размер записи и т.п. Дескриптор файла может храниться как вместе с файлом, так и в каталоге файлов.

5.4.4. Операции над файлами

Во многих операционных системах операции над файлами имеют много общего, хотя отличаются по форме выражения требований.

Общие операции над файлами можно разделить на три группы:

1) операции над файлами как над единым целым;
2) операции для обмена данными между файлом и программой, инициирующей обмен;
3) служебные операции.

К *первой группе* относятся следующие файловые операции: открытие файла, закрытие файла, копирование файла, объединение файлов, перемещение файла, удаление файла.

Операции открытия и закрытия файла являются наиболее важными среди операций первой группы, поэтому рассмотрим их подробно.

Для процесса, который желает работать с файлом, файл выступает как ресурс, который необходимо получить, прежде чем начать с ним работать. Поэтому пользователь обязан выполнить операцию открытия файла в своей программе, прежде чем начать выполнять с файлом некоторые допустимые действия.

Действия, выполняемые операционной системой при обработке операции открытия файла, чрезвычайно разнообразны и объемны. По отношению к другим операциям над файлами операция открытия файла является самой длительной операцией,

обеспечивающей возможность выполнения последующих действий над файлом достаточно быстро (в предположении их частого выполнения). Действия по открытию файла связаны с проведением подготовительных работ в составе файловой системы к дальнейшему взаимообмену между данными и программой или по управлению файлом. Одними из основных являются действия по подготовке к пересылке информации, которые осуществляются средствами программного или аппаратного канала. Чтобы составить программу для работы канала, система должна располагать информацией о соответствующих характеристиках открываемого файла. Прежде всего необходимо знать его месторасположение. Поэтому, если файл каталогизирован, то осуществляется обращение к системе каталогов, находится и считывается необходимая информация о файле (в каталоге или дескрипторе). Если система не обнаруживает при поиске требуемый файл, то она выдает об этом соответствующее сообщение.

Характерным приемом, применяемым при открытии файла, является копирование учетной информации о нем из каталога или из дескриптора файла в некоторую структуру данных (различную в разных файловых системах), расположенную в первичной памяти. Это позволяет далее оперативно работать с учетной информацией при организации фактического доступа пользователя к файлу без многократного обращения к внешней памяти.

Операция закрытия файла противоположна по смыслу операции открытия. Ее назначение – разорвать логическую связь между файлом и соответствующей программой.

Ко *второй группе* относятся следующие операции над файлами: чтение составных элементов файла, запись составных элементов файла.

Если файловая система поддерживает файлы с различной структурой, то команды записи и чтения будут разными по форме и содержанию. Можно использовать асинхронный и синхронный способ доступа к данным.

Синхронный способ чтения или записи данных в файл возможен только в случае, если файл имеет последовательную структуру. После выдачи запроса, реализованного в виде макрокоманды, на ввод или вывод одной логической записи (минимально различимый элемент файла) программный процесс переводится системой принудительно в состояние ожидания. Перевод программного процесса обратно в активное состояние операционная система выполняет только тогда, когда файловая система ОС закончит пересылку данных между файлом и указанной в макрокоманде областью первичной памяти.

При асинхронном способе файловая система после восприятия запроса от программного процесса на ввод или вывод данных передает управление программному процессу без перевода его в состояние ожидания. Вместе с тем файловая система предоставляет программному процессу средства для последующей проверки окончания операции ввода-вывода. Используя асинхронный способ, можно работать с любой структурой файлов, допустимой в файловой системе.

Состав операций, составляющих *третью группу*, в разных файловых системах различен. Например, к этой группе относят операции по переименованию файлов, изменению атрибутов файлов, управлению правами доступа к файлам, созданию и удалению каталогов. К этой же группе можно отнести операции, обеспечивающие просмотр и печать содержимого каталогов и файлов, поиска файлов и каталогов по дереву каталогов и т.п.

5.4.5. Файловая система

5.4.5.1. Общая модель файловой системы

Функционирование любой файловой системы можно представить многоуровневой моделью (рис. 5.9), в которой каждый уровень предоставляет некоторый интерфейс (набор функций) вышележащему уровню, а сам, в свою очередь, для выполнения своей работы использует интерфейс (обращается с набором запросов) нижележащего уровня.

Рис. 5.9. Общая модель файловой системы

Задачей *символьного уровня* является определение по символьному имени файла его уникального имени. В файловых системах, в которых каждый файл может иметь только одно символьное имя (например MS DOS), этот уровень отсутствует, так как символьное имя, присвоенное файлу пользователем, является одновременно уникальным и может быть использовано операционной системой. В других файловых системах, в которых один и тот же файл может иметь несколько символьных имен, на данном уровне просматривается цепочка каталогов для определения уникального имени файла. В файловой системе Unix, например, уникальным именем является номер индексного дескриптора файла (i-node).

На следующем, *базовом уровне* по уникальному имени файла определяются его характеристики: права доступа, адрес, размер и другие. Как уже было сказано, характеристики файла могут входить в состав каталога или храниться в отдельных таблицах. При открытии файла его характеристики перемещаются с диска в оперативную память, чтобы уменьшить среднее время доступа к файлу. В некоторых файловых системах (например HPFS) при открытии файла вместе с его характеристиками в оперативную память перемещаются несколько первых блоков файла, содержащих данные.

Следующим этапом реализации запроса к файлу является проверка прав доступа к нему (*уровень проверки прав доступа*). Для этого сравниваются полномочия пользователя или процесса, выдавших запрос, со списком разрешенных видов доступа к данному файлу. Если запрашиваемый вид доступа разрешен, то выполнение запроса продолжается, если нет, то выдается сообщение о нарушении прав доступа.

На *логическом уровне* определяются координаты запрашиваемой логической записи в файле, т.е. требуется определить, на каком расстоянии (в байтах) от начала файла находится требуемая логическая запись. При этом абстрагируются от физического расположения файла, он представляется в виде непрерывной последовательности байт. Алгоритм работы данного уровня зависит от логической организации файла.

На *физическом уровне* файловая система определяет номер физического блока, который содержит требуемую логическую запись, и смещение логической записи в физическом блоке. Для решения этой задачи используются результаты работы логического уровня – смещение логической записи в файле, адрес файла на внешнем устройстве, а также сведения о физической организации файла, включая размер блока. Задача физического уровня решается независимо от того, как был логически организован файл.

После определения номера физического блока, файловая система обращается к системе ввода-вывода для выполнения операции обмена с внешним устройством. В ответ на этот запрос в буфер файловой системы будет передан нужный блок, в котором на основании полученного при работе физического уровня смещения выбирается требуемая логическая запись.

5.4.5.2. Современные архитектуры файловых систем

Разработчики новых операционных систем стремятся обеспечить пользователя возможностью работать сразу с несколькими файловыми системами. В современном понимании файловая система состоит из многих составляющих, в число которых входят и файловые системы в традиционном понимании.

Современная файловая система имеет многоуровневую структуру (рис. 5.10), на верхнем уровне которой располагается так называемый переключатель файловых систем. Он обеспечивает интерфейс между запросами приложения и конкретной файловой системой, к которой обращается это приложение. Переключатель файловых систем преобразует запросы в формат, воспринимаемый следующим уровнем – уровнем файловых систем.

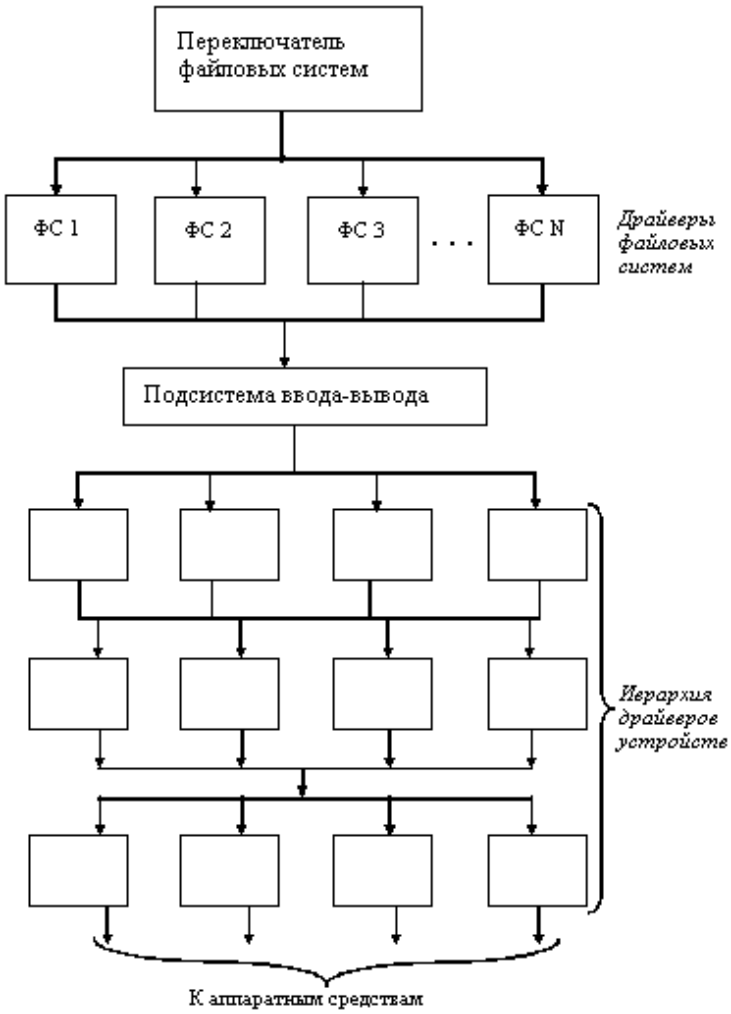

Рис. 5.10. Архитектура современной файловой системы

Каждый компонент уровня файловых систем выполнен в виде драйвера соответствующей файловой системы и поддерживает определенную организацию файловой системы. Переключатель является единственным модулем, который может обращаться к драйверу файловой системы. Приложение не может обращаться к нему напрямую.

Драйвер файловой системы может быть написан в виде реентерабельного кода, что позволяет сразу нескольким приложениям выполнять операции с файлами. Каждый драйвер файловой системы в процессе собственной инициализации регистрируется у переключателя, передавая ему таблицу точек входа, которые будут использоваться при последующих обращениях к файловой системе.

Для выполнения своих функций драйверы файловых систем обращаются к подсистеме ввода-вывода, образующей следующий слой файловой системы новой архитектуры. Подсистема ввода вывода – это составная часть файловой системы, которая отвечает за загрузку, инициализацию и управление всеми модулями низших уровней файловой системы. Обычно эти модули представляют собой драйверы портов, которые непосредственно занимаются работой с аппаратными средствами. Кроме этого подсистема ввода-вывода обеспечивает некоторый сервис драйверам файловой системы, что позволяет им осуществлять запросы к конкретным устройствам. Подсистема ввода-вывода должна постоянно присутствовать в памяти и организовывать совместную работу иерархии драйверов устройств. В эту иерархию могут входить драйверы устройств определенного типа (драйверы жестких дисков или накопителей на лентах), драйверы, поддерживаемые поставщиками (такие драйверы перехватывают запросы к блочным устройствам и могут частично изменить поведение существующего драйвера этого устройства, например зашифровать данные), драйверы портов, которые управляют конкретными адаптерами.

Большое число уровней архитектуры файловой системы обеспечивает авторам драйверов устройств большую гибкость – драйвер может получить управление на любом этапе выполнения запроса – от вызова приложением функции, которая занимается работой с файлами, до того момента, когда работающий на самом низком уровне драйвер устройства начинает просматривать регистры контроллера. Многоуровневый механизм работы файловой системы реализован посредством цепочек вызова.

В ходе инициализации драйвер устройства может добавить себя к цепочке вызова некоторого устройства, определив при этом уровень последующего обращения. Подсистема ввода-вывода помещает адрес целевой функции в цепочку вызова устройства, используя заданный уровень для того, чтобы должным образом упорядочить цепочку. По мере выполнения запроса, подсистема ввода-вывода последовательно вызывает все функции, ранее помещенные в цепочку вызова.

Внесенная в цепочку вызова процедура драйвера может решить передать запрос дальше – в измененном или в неизменном виде – на следующий уровень, или, если это возможно, процедура может удовлетворить запрос, не передавая его дальше по цепочке.

5.5. СИСТЕМА ВВОДА-ВЫВОДА

5.5.1. Общие положения

В состав любой операционной системы входят программные модули, обеспечивающие управление устройствами ввода-вывода ЭВМ. Эти программные модули называют драйверами устройств, а совокупность драйверов ввода-вывода образует систему ввода-вывода, входящую в состав операционной системы.

Драйвер устройства (Device driver) – программа, обеспечивающая взаимодействие операционной системы с физическим устройством.

Система ввода-вывода (Input-Output System) – часть операционной системы, обеспечивающая управление внешними устройствами, подключенными к ЭВМ.

Основной задачей системы ввода-вывода является обеспечение непрерывной организации (планирования, управления) и двусторонней передачи данных между основной памятью и внешними устройствами с целью достижения максимального перекрытия во времени работы этой аппаратуры и процессора.

Состав систем ввода-вывода и, следовательно, перечень драйверов устройств в различных операционных системах не совпадают, что объясняется имеющимися отличиями в аппаратуре ввода-вывода, а также множеством методов, используемых для управления этой аппаратурой. Вместе с тем в большинстве операционных систем существует некоторое ядро системы ввода-вывода, получившее название базовой системы ввода-вывода.

Базовая система ввода-вывода (BIOS – Basic Input Output System) – часть программного обеспечения ЭВМ, поддерживающая управление адаптерами внешних устройств и представляющая стандартный интерфейс для обеспечения переносимости операционных систем между ЭВМ с одинаковым процессором. Базовая система ввода-вывода, как правило, разрабатывается изготовителем ЭВМ, хранится в постоянном запоминающем устройстве и рассматривается как часть ЭВМ.

При построении систем ввода-вывода аппаратура ввода-вывода рассматривается как совокупность аппаратурных процессоров, которые способны работать параллельно и независимо друг от друга, а также относительно центрального процессора. На таких процессорах развиваются так называемые внешние процессы.

Внешние процессы, используя аппаратуру ввода-вывода, могут взаимодействовать как между собой, так и с внутренними процессами, которые развиваются на центральном процессоре. Важным фактом является то, что скорости развития внешних и внутренних процессов существенно различаются, причем эти различия могут достигать нескольких порядков.

Система управления вводом-выводом представляет собой один или несколько системных процессов (т.е. процессов, принадлежащих операционной системе), обеспечивающих информационное и управляющее взаимодействие внутренних и внешних процессов. Через эту систему происходит инициализация, управление развитием и уничтожение внешних процессов.

С точки зрения внутренних (программных) процессов-пользователей система управления вводом-выводом представляет собой программный интерфейс с необходимыми для этих процессов внешними устройствами. В составе этого интерфейса пользователь имеет возможность выражать запросы на выполнение действий в отношении внешних устройств. При этом различают три типа действий: операции чтения и записи данных, операции управления устройством, операции по проверке состояния устройств. При построении систем управления вводом-выводом руководствуются стремлением сделать большинство ее компонентов "невидимыми" для пользователей, что достигается созданием развитых драйверов внешних устройств с понятным интерфейсом и доступными из любой системы программирования.

Для сглаживания эффекта несоответствия скоростей между внутренними и внешними процессами в системах управления вводом-выводом применяют три основных метода: синхронизация по прерываниям ввода-вывода, буферизация ввода-вывода, блокирование данных.

Для синхронизации параллельной работы могут применяться различные методы, среди которых наиболее совершенными являются средства, основанные на использовании системы прерывания. Канал ввода-вывода через систему прерываний прерывает работу центрального процессора всякий раз при завершении операции ввода-вывода или при возникновении ошибки. Такие сигналы прерывания являются по своему смыслу синхронизирующими, так как они используются для оповещения определенного внутреннего процесса о событии, которое произошло при работе канала ввода-вывода или внешнего устройства.

Одной из главных функций ОС является управление всеми устройствами ввода-вывода компьютера. ОС должна передавать устройствам команды, перехватывать прерывания и обрабатывать ошибки; она также должна обеспечивать интерфейс между устройствами и остальной частью системы. В целях развития интерфейс должен быть одинаковым для всех типов устройств (независимость от устройств).

5.5.2. Физическая организация устройств ввода-вывода

Устройства ввода-вывода делятся на два типа: блок-ориентированные и байт-ориентированные устройства. Блок-ориентированные устройства хранят информацию в блоках фиксированного размера, каждый из которых имеет свой собственный адрес. Самое распространенное блок-ориентированное устройство – диск. Байт-ориентированные устройства не адресуемы и не позволяют производить операцию поиска, они генерируют или потребляют последовательность байтов. Примерами являются терминалы, строчные принтеры, сетевые адаптеры. Однако некоторые внешние устройства не относятся ни к одному классу, например часы, которые, с одной стороны, не адресуемы, а с другой – не порождают потока байтов. Это устройство только выдает сигнал прерывания в некоторые моменты времени.

Внешнее устройство обычно состоит из механического и электронного компонента. Электронный компонент называется контроллером устройства или адаптером. Механический компонент представляет собственно устройство. Некоторые кон-

троллеры могут управлять несколькими устройствами. Если интерфейс между контроллером и устройством стандартизован, то независимые производители могут выпускать совместимые как контроллеры, так и устройства.

Операционная система обычно имеет дело не с устройством, а с контроллером. Контроллер, как правило, выполняет простые функции, например, преобразует поток бит в блоки, состоящие из байт, и осуществляет контроль и исправление ошибок. Каждый контроллер имеет несколько регистров, которые используются для взаимодействия с центральным процессором. В некоторых компьютерах эти регистры являются частью физического адресного пространства. В таких компьютерах нет специальных операций ввода-вывода. В других компьютерах адреса регистров ввода-вывода, называемых часто портами, образуют собственное адресное пространство за счет введения специальных операций ввода-вывода (например команд IN и OUT в процессорах i86).

ОС выполняет ввод-вывод, записывая команды в регистры контроллера. Например, контроллер гибкого диска IBM PC принимает
15 команд, таких как READ, WRITE, SEEK, FORMAT и т.д. Когда команда принята, процессор оставляет контроллер и занимается другой работой. При завершении команды контроллер организует прерывание для того, чтобы передать управление процессором операционной системе, которая должна проверить результаты операции. Процессор получает результаты и статус устройства, читая информацию из регистров контроллера.

5.5.3. Организация программного обеспечения ввода-вывода

5.5.3.1. Уровни организации программного обеспечения ввода-вывода

Основная идея организации программного обеспечения ввода-вывода состоит в разбиении его на несколько уровней, причем нижние уровни обеспечивают экранирование особенностей аппаратуры от верхних, а те, в свою очередь, обеспечивают удобный интерфейс для пользователей.

Ключевым принципом является независимость от устройств. Вид программы не должен зависеть от того, читает ли она данные с гибкого или жесткого диска.

Очень близкой к идее независимости от устройств является идея единообразного именования, т.е. для именования устройств должны быть приняты единые правила.

Другим важным вопросом для программного обеспечения ввода-вывода является обработка ошибок. Вообще говоря, ошибки следует обрабатывать как можно ближе к аппаратуре. Если контроллер обнаруживает ошибку чтения, то он должен попытаться ее скорректировать. Если же это ему не удается, то исправлением ошибок должен заняться драйвер устройства. Многие ошибки могут исчезать при повторных попытках выполнения операций ввода-вывода, например ошибки, вызванные наличием пылинок на головках чтения или на диске. И только если нижний уровень не может справиться с ошибкой, он сообщает об ошибке верхнему уровню.

Еще один ключевой вопрос – это использование блокирующих (синхронных) и неблокирующих (асинхронных) передач. Большинство операций физического ввода-вывода выполняется асинхронно – процессор начинает передачу и переходит на другую работу, пока не наступает прерывание. Пользовательские программы намного легче писать, если операции ввода-вывода блокирующие – после команды READ программа автоматически приостанавливается до тех пор, пока данные не попадут в буфер программы. ОС выполняет операции ввода-вывода асинхронно, но представляет их для пользовательских программ в синхронной форме.

Последняя проблема состоит в том, что одни устройства являются разделяемыми, а другие – выделенными. Диски – это разделяемые устройства, так как одновременный доступ нескольких пользователей к диску не представляет собой проблему. Принтеры – это выделенные устройства, потому что нельзя смешивать строчки, печатаемые различными пользователями. Наличие выделенных устройств создает для операционной системы некоторые проблемы.

Для решения поставленных проблем целесообразно разделить программное обеспечение ввода-вывода на четыре слоя (рис. 5.11):

1) обработка прерываний;
2) драйверы устройств;
3) независимый от устройств слой операционной системы;
4) пользовательский слой программного обеспечения.

Рис. 5.11. Многоуровневая организация программного обеспечения системы ввода-вывода

5.5.3.2. ОБРАБОТКА ПРЕРЫВАНИЙ

Прерывания должны быть скрыты как можно глубже в недрах операционной системы, чтобы как можно меньшая часть ОС имела с ними дело. Наилучший способ состоит в разрешении процессу, инициировавшему операцию ввода-вывода, блокировать себя до завершения операции и наступления прерывания. Процесс может блокировать себя, используя, например, вызов DOWN для семафора, или вызов WAIT для переменной условия, или вызов RECEIVE для ожидания сообщения. При наступлении прерывания процедура обработки прерывания выполняет разблокирование процесса, инициировавшего операцию ввода-вывода, используя вызовы UP, SIGNAL или посылая процессу сообщение. В любом случае эффект от прерывания будет состоять в том, что ранее заблокированный процесс теперь продолжит свое выполнение.

5.5.3.3. ДРАЙВЕРЫ УСТРОЙСТВ

Весь зависимый от устройства код помещается в драйвер устройства. Каждый драйвер управляет устройствами одного типа или, может быть, одного класса.

В операционной системе только драйвер устройства знает о конкретных особенностях какого-либо устройства. Например, только драйвер диска имеет дело с дорожками, секторами, цилиндрами, временем установления головки и другими факторами, обеспечивающими правильную работу диска.

Драйвер устройства принимает запрос от устройств программного слоя и решает, как его выполнить. Типичным запросом является чтение *n* блоков данных. Если драйвер был свободен во время поступления запроса, то он начинает выполнять запрос немедленно. Если же он был занят обслуживанием другого запроса, то вновь поступивший запрос присоединяется к очереди уже имеющихся запросов, и он будет выполнен, когда наступит его очередь.

Первый шаг в реализации запроса ввода-вывода, например для диска, состоит в преобразовании его из абстрактной формы в конкретную. Для дискового драйвера это означает преобразование номеров блоков в номера цилиндров, головок, секторов; проверку, работает ли мотор, находится ли головка над нужным цилиндром. Короче говоря, он должен решить, какие операции контроллера нужно выполнить и в какой последовательности.

После передачи команды контроллеру драйвер должен решить, блокировать ли себя до окончания заданной операции или нет. Если операция занимает значительное время, как при печати некоторого блока данных, то драйвер блокируется до тех пор, пока операция не завершится, и обработчик прерывания не разблокирует его. Если команда ввода-вывода выполняется быстро (например, прокрутка экрана), то драйвер ожидает ее завершения без блокирования.

5.5.3.4. НЕЗАВИСИМЫЙ ОТ УСТРОЙСТВ СЛОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ

Большая часть программного обеспечения ввода-вывода является независимой от устройств. Точная граница между драйверами и независимыми от устройств программами определяется системой, так как некоторые функции, которые могли бы быть реализованы независимым способом, в действительности выполнены в виде драйверов для повышения эффективности или по другим причинам.

Типичными функциями для независимого от устройств слоя являются:

- обеспечение общего интерфейса к драйверам устройств;
- именование устройств;
- защита устройств;
- обеспечение независимого размера блока;
- буферизация;
- распределение памяти на блок-ориентированных устройствах;
- распределение и освобождение выделенных устройств;
- уведомление об ошибках.

Остановимся на некоторых функциях данного перечня. Верхним слоям программного обеспечения неудобно работать с блоками разной величины, поэтому данный слой обеспечивает единый размер блока, например, за счет объединения нескольких различных блоков в единый логический блок. В связи с этим верхние уровни имеют дело с абстрактными устройствами, которые используют единый размер логического блока независимо от размера физического сектора.

При создании файла или заполнении его новыми данными необходимо выделить ему новые блоки. Для этого ОС должна вести список или битовую карту свободных блоков диска. На основании информации о наличии свободного места на диске может быть разработан алгоритм поиска свободного блока, независимый от устройства и реализуемый программным слоем, находящимся выше слоя драйверов.

5.5.3.5. ПОЛЬЗОВАТЕЛЬСКИЙ СЛОЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Хотя большая часть программного обеспечения ввода-вывода находится внутри ОС, некоторая его часть содержится в библиотеках, связываемых с пользовательскими программами. Системные вызовы, включающие вызовы ввода-вывода, обычно делаются библиотечными процедурами. Если программа, написанная на языке С, содержит вызов

count = write (fd, buffer, nbytes),

то библиотечная процедура write будет связана с программой. Набор подобных процедур является частью системы ввода-вывода. В частности, форматирование ввода или вывода выполняется библиотечными процедурами. Примером может служить функция printf языка С, которая принимает строку формата и, возможно, некоторые переменные в качестве входной информации, затем строит строку символов и делает вызов write для вывода этой строки. Стандартная библиотека ввода-вывода содержит большое число процедур, которые выполняют ввод-вывод и работают как часть пользовательской программы.

Другой категорией программного обеспечения ввода-вывода является подсистема спулинга (spooling). Спулинг – это способ работы с выделенными устройствами в мультипрограммной системе. Рассмотрим типичное устройство, требующее спулинга – принтер. Хотя технически легко позволить каждому пользовательскому процессу открыть специальный файл, связанный с принтером, такой способ опасен из-за того, что пользовательский процесс может монополизировать принтер на произвольное время. Вместо этого создается специальный процесс – монитор, который получает исключительные права на использование этого устройства. Также создается специальный каталог, называемый каталогом спулинга. Для того чтобы напечатать файл, пользовательский процесс помещает выводимую информацию в этот файл и помещает его в каталог спулинга. Процесс-монитор по очереди распечатывает все файлы, содержащиеся в каталоге спулинга.

Контрольные вопросы к теме 5

1. Что такое данные, источник данных, организация данных?
2. Перечислите методы организации данных. В чем их различия?
3. Опишите способы организации файлов.
4. Как можно хранить файлы на носителе?
5. Перечислите основные операции над файлами.
6. Перечислите и опишите уровни многоуровневой модели файловой системы.
7. Каковы основные компоненты архитектуры современных файловых систем?
8. Дайте определения системе ввода-вывода.
9. Что такое драйвер ввода-вывода?
10. Перечислите и охарактеризуйте типы устройств ввода-вывода.
11. На какие слои (уровни) разбито программное обеспечение ввода-вывода, каково их назначение?

Р а з д е л 2

ЗАЩИТА ИНФОРМАЦИИ В СОВРЕМЕННЫХ ОПЕРАЦИОННЫХ СИСТЕМАХ

Т е м а 6. ОСНОВНЫЕ ПОНЯТИЯ И ПОЛОЖЕНИЯ ЗАЩИТЫ ИНФОРМАЦИИ В ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Информационная безопасность – сравнительно молодая, быстро развивающаяся область информационных технологий. Под ***информационной безопасностью*** будем понимать защищенность информации и поддерживающей инфраструктуры от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести неприемлемый ущерб субъектам информационных отношений, в том числе владельцам и пользователям информации и поддерживающей инфраструктуры.

Защита информации – это комплекс мероприятий, направленных на обеспечение информационной безопасности.

С методологической точки зрения правильный подход к проблемам информационной безопасности начинается с выявления ***субъектов информационных отношений*** и интересов этих субъектов, связанных с использованием информационных систем (ИС). Угрозы информационной безопасности – это оборотная сторона использования информационных технологий.

Информационная безопасность – многогранная область деятельности, в которой успех может принести только систематический, комплексный подход. Для решения данной проблемы рассматриваются меры ***законодательного, административного, процедурного и программно-технического уровня***.

Спектр интересов субъектов, связанных с использованием информационных систем, можно разделить на следующие категории: обеспечение ***доступности, целостности*** и ***конфиденциальности*** информационных ресурсов и поддерживающей инфраструктуры.

6.1. ПРЕДМЕТ ЗАЩИТЫ ИНФОРМАЦИИ

Успех практически любой деятельности в немалой степени зависит от умения распоряжаться такой ценностью, как информация.

В законе РФ "Об информации, информатизации и защите информации" определено:

- "***информационные ресурсы*** являются объектами собственности граждан, организаций, общественных объединений, государства";
- "***информация*** – сведения о лицах, предметах, событиях, явлениях и процессах (независимо от формы их представления), отраженные на материальных носителях, используемые в целях получения знаний и практических решений".

Информация имеет ряд особенностей:

- не материальна;
- хранится и передается с помощью материальных носителей;
- любой материальный объект содержит информацию о самом себе либо о другом объекте.

Информации присущи следующие свойства:

Ценность информации определяется степенью ее полезности для владельца. Законом РФ "Об информации, информатизации и защите информации" гарантируется право собственника информации на ее использование и защиту от доступа к ней других лиц (организаций). Если доступ к информации ограничен, то такая информация называется *конфиденциальной*. Конфиденциальная информация может содержать *государственную* или *коммерческую* тайну.

Достоверность информации определяется достаточной для владельца точностью отражать объекты и процессы окружающего мира в определенных временных и пространственных рамках. Информация, искаженно представляющая действительность, может нанести владельцу значительный материальный и моральный ущерб. Если информация искажена умышленно, то ее называют *дезинформацией.*

Своевременность информации, т.е. соответствие ценности и достоверности определенному временному периоду, может быть выражена формулой

$$C(t) = C_0 e^{-2,3t/\tau},$$

где C_0 – ценность информации в момент ее возникновения; t – время от момента возникновения информации до момента определения ее стоимости; τ – время от момента возникновения информации до момента ее устаревания.

Предметом защиты является информация, хранящаяся, обрабатываемая и передаваемая в компьютерных (информационных) системах. Особенностями данного вида информации являются:

- двоичное представление информации внутри системы, независимо от физической сущности носителей исходной информации;
- высокая степень автоматизации обработки и передачи информации;
- концентрация большого количества информации в КС.

6.2. ОБЪЕКТ ЗАЩИТЫ ИНФОРМАЦИИ

Объектом защиты информации является компьютерная (информационная) система или автоматизированная система обработки информации (АСОИ).

Информационная система – это организационно-упорядоченная совокупность информационных ресурсов, технических средств, технологий и персонала, реализующих информационные процессы в традиционном или автоматизированном режиме для удовлетворения информационных потребностей пользователей.

Информационная безопасность АСОИ – состояние рассматриваемой автоматизированной системы, при котором она, с одной стороны, способна противостоять дестабилизирующему воздействию внешних и внутренних информационных угроз, а с другой − ее наличие и функционирование не создает информационных угроз для элементов самой системы и внешней среды.

Информационная безопасность достигается проведением соответствующего уровня политики информационной безопасности.

Под ***политикой информационной безопасности*** понимают совокупность норм, правил и практических рекомендаций, регламентирующих работу средств защиты АСОИ от заданного множества угроз безопасности.

Система защиты информации – совокупность правовых норм, организационных мер и мероприятий, технических, программных и криптографических средств и методов, обеспечивающих защищенность информации в системе в соответствии с принятой политикой безопасности.

6.2.1. ОСНОВНЫЕ ПОЛОЖЕНИЯ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ

Что касается подходов к реализации защитных мероприятий по обеспечению безопасности информационных систем, то сложилась трехэтапная (трехстадийная) разработка таких мер.

Первая стадия – ***выработка требований*** – включает:

- определение состава средств информационной системы;
- анализ уязвимых элементов ИС;
- оценка угроз (выявление проблем, возникающих при наличия уязвимых мест);
- анализ риска (прогноз возможных последствий, вызывающих эти проблемы).

Вторая стадия – ***определение способов защиты*** – включает ответы на следующие вопросы:

Какие угрозы должны быть устранены и в какой мере?

Какие ресурсы системы должны быть защищаемы и в какой степени?

С помощью каких средств должна быть реализована защита?

Какова должна быть полная стоимость реализации защиты и затраты на эксплуатацию с учетом потенциальных угроз?

Третья стадия – ***определение функций, процедур и средств безопасности, реализуемых в виде некоторых механизмов защиты***.

6.2.2. ОСНОВНЫЕ ПРИНЦИПЫ ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ В АС

Для защиты АС на основании руководящих документов Гостехкомиссии сии могут быть сформулированы следующие положения.

1. Информационная безопасность АС основывается на положениях требованиях существующих законов, стандартов и нормативно-методических документов.
2. Информационная безопасность АС обеспечивается комплексом программно-технических средств и поддерживающих их организационных мер.
3. Информационная безопасность АС должна обеспечиваться на всех технологических этапах обработки информации и во всех режимах функционирования, в том числе при проведении ремонтных и регламентных работ.
4. Программно-технические средства защиты не должны существенно ухудшать основные функциональные характеристики АС (надежность, быстродействие, возможность изменения конфигурации АС).
5. Неотъемлемой частью работ по ИБ является оценка эффективности средств защиты, осуществляемая по методике, учитывающей всю совокупность технических характеристик оцениваемого объекта, включая технические решения и практическую реализацию средств защиты.
6. Защита АС должна предусматривать контроль эффективности средств защиты. Этот контроль может быть периодическим либо инициироваться по мере необходимости пользователем АС или контролирующим органом.

Рассмотренные подходы могут быть реализованы при обеспечении следующих основных принципов:

Принцип системности. Системный подход к защите информационных систем предполагает необходимость учета всех взаимосвязанных, взаимодействующих и изменяющихся во времени элементов, условий и факторов:

- при всех видах информационного проявления и деятельности;
- во всех структурных элементах;
- при всех режимах функционирования;
- на всех этапах жизненного цикла;
- с учетом взаимодействия объекта защиты с внешней средой.

Система защиты должна строиться не только с учетом всех известных каналов проникновения, но и с учетом возможности появления принципиально новых путей реализации угроз безопасности.

Принцип комплексности. В распоряжении специалистов по компьютерной безопасности имеется широкий спектр мер, методов и средств защиты компьютерных систем (современные СВТ, ОС, инструментальные и прикладные программные средства, обладающие теми или иными встроенными элементами защиты). Комплексное их использование предполагает

согласование разнородных средств при построении целостной системы защиты, перекрывающей все существенные каналы реализации угроз и не содержащей слабых мест на стыках отдельных ее компонентов.

Принцип непрерывности защиты. Защита информации – это не разовое мероприятие и даже не конкретная совокупность уже проведенных мероприятий и установленных средств защиты, а непрерывный целенаправленный процесс, предполагающий принятие соответствующих мер на всех этапах жизненного цикла АС. Разработка системы защиты должна вестись параллельно с разработкой самой защищаемой системы. Это позволит учесть требования безопасности при проектировании архитектуры и, в конечном счете, позволит создать более эффективные (как по затратам ресурсов, так и по стойкости) защищенные системы. Большинству физических и технических средств защиты для эффективного выполнения своих функций необходима постоянная организационная поддержка (своевременная смена и обеспечение правильного хранения и применения имен, паролей, ключей шифрования, переопределение полномочий и т.п.). Перерывы в работе средств защиты могут быть использованы злоумышленниками для анализа применяемых методов и средств защиты, внедрения специальных программных и аппаратных "закладок" и других средств преодоления системы защиты после восстановления ее функционирования.

Разумная достаточность. Создать абсолютно непреодолимую систему защиты принципиально невозможно, при достаточных времени и средствах можно преодолеть любую защиту. Поэтому имеет смысл вести речь только о некотором приемлемом уровне безопасности. Высокоэффективная система защиты стоит дорого, использует при работе существенную часть мощности и ресурсов ИС и может создавать ощутимые дополнительные неудобства пользователям. Важно правильно выбрать тот достаточный уровень зашиты, при котором затраты, риск и размер возможного ущерба были бы приемлемыми.

Гибкость системы защиты. Часто приходится создавать систему защиты в условиях большой неопределенности. Поэтому принятые меры и установленные средства защиты, особенно в начальный период их эксплуатации, могут обеспечивать как чрезмерный, так и недостаточный уровень защиты. Для обеспечения возможности варьирования уровнем защищенности средства защиты должны обладать определенной гибкостью. Особенно важно это свойство в тех случаях, когда средства защиты необходимо устанавливать на работающую систему, нарушая процесс ее нормального функционирования.

6.1. Этапы развития концепций обеспечения безопасности данных

Этапы развития концепций	Характеристика этапа
1 этап 1960 – 1970 гг.	Попытки обеспечить безопасность данных чисто формальными механизмами, содержащими, главным образом, технические и программные средства. Сосредоточение программных средств в рамках операционных систем и систем управления базами данных
2 этап 1970 – 1976 гг.	Развитие формальных механизмов защиты данных. Выделение управляющего компонента защиты данных – ядра безопасности. Развитие неформальных средств защиты. Формирование основ системного подхода к обеспечению безопасности данных
3 этап 1976 – 1990 гг.	Дальнейшее развитие механизмов второго этапа. Формирование взгляда на обеспечение безопасности данных как на непрерывный процесс. Развитие стандартов на средства защиты данных. Усиление тенденции аппаратной реализации средств защиты данных. Формирование вывода о взаимосвязи обеспечения безопасности данных, архитектуры ИВС и технологии ее функционирования. Формирование системного подхода к проблеме обеспечения безопасности данных
4 этап 1990 г. – по настоящее время	Дальнейшее развитие механизмов третьего этапа. Формирование основ теории обеспечения безопасности данных в ИВС. Разработка моделей, методов и алгоритмов управления защитой данных в ИВС

Открытость алгоритмов и механизмов защиты. Суть принципа открытости алгоритмов и механизмов защиты состоит в том, что защита не должна обеспечиваться только за счет секретности структурной организации и алгоритмов функционирования ее подсистем. Знание алгоритмов работы системы защиты не должно давать возможности ее преодоления. Но это вовсе не означает, что информация конкретной системы защиты должна быть общедоступна – необходимо обеспечивать защиту от угрозы раскрытия параметров системы.

Принцип простоты применения средств защиты. Механизмы защиты должны быть интуитивно понятны и просты в использовании, применение средств защиты не должно быть связано со знанием специальных языков или с выполнением

действий, требующих значительных дополнительных трудозатрат при обычной работе законных пользователей, а также не должно требовать от пользователя выполнения рутинных непонятных ему операций (ввод нескольких паролей и имен и т.д.).

В настоящее время выделяют этапы развития концепций обеспечения безопасности данных, сущность которых приведена в табл. 6.1.

Контрольные вопросы к теме 6

1. Охарактеризуйте информацию и ее свойства.
2. Что является предметом и объектом защиты информации?
3. Чем определяется ценность информации? Приведите классификацию конфиденциальной информации.
4. Охарактеризуйте свойства достоверности и своевременности информации.
5. Дайте определения информационной безопасности АСОИ и политики информационной безопасности.

Т е м а 7. УГРОЗЫ БЕЗОПАСНОСТИ ИНФОРМАЦИИ В ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

7.1. Анализ угроз информационной безопасности

Угроза – это потенциальная возможность определенным образом нарушить информационную безопасность.

Угроза – это потенциально возможно событие, действие (воздействие), процесс или явление, которое может привести к нанесению ущерба чьим-либо интересам.

Угрозой информационной безопасности АС называется возможность реализации воздействия на информацию, обрабатываемую АС, приводящего к искажению, уничтожению, копированию, блокированию доступа к информации, а также возможность воздействия на компоненты АС, приводящего к утрате, уничтожению или сбою функционирования носителя информации, средства взаимодействия с носителем или средства его управления.

Попытка реализации угрозы называется *атакой*, а тот, кто предпринимает такую попытку, – *злоумышленником*. Потенциальные злоумышленники называются *источниками угрозы*.

Чаще всего угроза является следствием наличия *уязвимых* мест в защите информационных систем (таких как возможность доступа посторонних лиц к критически важному оборудованию или ошибки в программном обеспечении).

Промежуток времени от момента, когда появляется возможность использовать слабое место, и до момента, когда пробел ликвидируется, называется *окном опасности*, ассоциированным с данным уязвимым местом. Пока существует окно опасности, возможны успешные атаки на ИС.

Если речь идет об ошибках в ПО, то окно опасности "открывается" с появлением средств использования ошибки и ликвидируется при наложении заплат, ее исправляющих.

Для большинства уязвимых мест окно опасности существует сравнительно долго (несколько дней, иногда – недель), поскольку за это время должны произойти следующие события:

- должно стать известно о средствах использования пробела в защите;
- должны быть выпущены соответствующие заплаты;
- заплаты должны быть установлены в защищаемой ИС.

Новые уязвимые места и средства их использования появляются постоянно; это значит, во-первых, что почти всегда существуют окна опасности, и, во-вторых, что отслеживание таких окон должно производиться постоянно, а выпуск и наложение заплат – как можно более оперативно.

Некоторые угрозы нельзя считать следствием каких-то ошибок или просчетов, они существуют в силу самой природы современных ИС. Например, угроза отключения электричества или выхода его параметров за допустимые границы существует в силу зависимости аппаратного обеспечения ИС от качественного электропитания.

Рассмотрение наиболее распространенных угроз, которым подвержены современные информационные системы, дает представление о возможных угрозах, а также об уязвимых местах, которые эти угрозы обычно эксплуатируют, необходимо для того, чтобы выбирать наиболее экономичные средства обеспечения безопасности.

Угрозы, как и все в ИБ, зависят от интересов субъектов информационных отношений (и от того, какой ущерб является для них неприемлемым). Задание возможных угроз информационной безопасности проводится с целью определения полного перечня требований к разрабатываемой системе защиты. Перечень угроз, оценки вероятностей их реализации, а также модель нарушителя служат основой для анализа риска реализации угроз и формулирования требований к системе защиты АС. Кроме выявления возможных угроз, должен быть проведен их анализ на основе классификационных признаков. Каждый из признаков классификации отражает одно из обобщенных требований к системе защиты. При этом угрозы, соответствующие каждому признаку классификации, позволяют детализировать отражаемое этим признаком требование.

***Угрозы можно классифицировать по нескольким критериям*:**

1) *по аспекту информационной безопасности* (доступность, целостность, конфиденциальность), против которого угрозы направлены в первую очередь;

2) *по компонентам информационных систем*, на которые угрозы нацелены (данные, программы, аппаратура, поддерживающая инфраструктура);

3) *по способу осуществления* (случайные/преднамеренные действия природного/техногенного характера);

4) *по расположению источника угроз* (внутри/вне рассматриваемой ИС).

Необходимость классификации угроз ИБ АС обусловлена тем, что архитектура современных средств автоматизированной обработки информации, организационное, структурное и функциональное построение информационно-вычислительных систем и сетей, технологии и условия автоматизированной обработки информации такие, что накапливаемая, хранимая и

обрабатываемая информация подвержена случайным влияниям чрезвычайно большого числа факторов, в силу чего становится невозможным формализовать задачу описания полного множества угроз. Как следствие, для защищаемой системы определяют не полный перечень угроз, а перечень классов угроз.

Классификация всех возможных ***угроз информационной безопасности*** АС может быть проведена по ряду базовых признаков.

1. *По природе возникновения*.

1) *Естественные угрозы* – угрозы, вызванные воздействиями на АС и ее компоненты объективных физических процессов или стихийных природных явлений, независящих от человека.

2) *Искусственные угрозы* – угрозы информационной безопасности АС, вызванные деятельностью человека.

2. *По степени преднамеренности проявления*.

1) *Угрозы случайного действия и/или угрозы, вызванные ошибками или халатностью персонала*. Например:

– проявление ошибок программно-аппаратных средств АС;

– некомпетентное использование, настройка или неправомерное отключение средств защиты персоналом службы безопасности;

– неумышленные действия, приводящие к частичному или полному отказу системы или разрушению аппаратных, программных, информационных ресурсов системы (неумышленная порча оборудования, удаление, искажение файлов с важной информацией или программ, в том числе системных и т. п.);

– неправомерное включение оборудования или изменение режимов работы устройств и программ;

– неумышленная порча носителей информации;

– пересылка данных по ошибочному адресу абонента (устройства);

– ввод ошибочных данных;

– неумышленное повреждение каналов связи.

2) *Угрозы преднамеренного действия* (например, угрозы действий злоумышленника для хищения информации).

3. *По непосредственному источнику угроз*.

1) *Угрозы, непосредственным источником которых является природная среда* (стихийные бедствия, магнитные бури, радиоактивное излучение и т.п.).

2) *Угрозы, источником которых является человек*:

– внедрение агентов в число персонала системы (в том числе, возможно, и в административную группу, отвечающую за безопасность);

– вербовка (путем подкупа, шантажа и т.п.) персонала или отдельных пользователей, имеющих определенные полномочия;

– угроза несанкционированного копирования секретных данных пользователем АС;

– разглашение, передача или утрата атрибутов разграничения доступа (паролей, ключей шифрования, идентификационных карточек, пропусков и т.п.).

3) *Угрозы, непосредственным источником которых являются санкционированные программно-аппаратные средства*:

– запуск технологических программ, способных при некомпетентном пользовании вызывать потерю работоспособности системы (зависания) или зацикливания) или необратимые изменения в системе (форматирование или реструктуризацию носителей информации, удаление данных и т.п.);

– возникновение отказа в работе операционной системы.

4) *Угрозы, непосредственным источником которых являются несанкционированные программно-аппаратные средства*:

– нелегальное внедрение и использование неучтенных программ (игровых, обучающих, технологических и других программ, не являющихся необходимыми для выполнения нарушителем своих служебных обязанностей) с последующим необоснованным расходованием ресурсов (загрузка процессора, захват оперативной памяти и памяти на внешних носителях);

– заражение компьютера вирусами с деструктивными функциями.

4. *По положению источника угроз*.

1) *Угрозы, источник которых расположен вне контролируемой зоны территории* (*помещения*), *на которой находится АС*:

– перехват побочных электромагнитных, акустических и других излучений устройств и линий связи, а также наводок активных излучений на вспомогательные технические средства, непосредственно не участвующие в обработке информации (телефонные линии, сети питания, отопления и т.п.);

– перехват данных, передаваемых по каналам связи, и их анализ с целью выяснения протоколов обмена, правил вхождения в связь и авторизации пользователя и последующих попыток их имитации для проникновения в систему;

– дистанционная фото- и видеосъемка.

2) *Угрозы, источник которых расположен в пределах контролируемой зоны территории* (*помещения*), *на которой находится АС*:

– хищение производственных отходов (распечаток, записей, списанных носителей информации и т.п.);

– отключение или вывод из строя подсистем обеспечения функционирования вычислительных систем (электропитания, охлаждения и вентиляции, линий связи и т.д.);

– применение подслушивающих устройств.

3) *Угрозы, источник которых имеет доступ к периферийным устройствам АС* (терминалам).

4) *Угрозы, источник которых расположен в АС*:

– проектирование архитектуры системы и технологии обработки данных, разработка прикладных программ, которые представляют опасность для работоспособности системы и безопасности информации;

– некорректное использование ресурсов АС.

5. *По степени зависимости от активности АС.*

1) *Угрозы*, *которые могут проявляться независимо от активности АС*:

– вскрытие шифров криптозащиты информации;

– хищение носителей информации (магнитных дисков, лент, микросхем памяти, запоминающих устройств и компьютерных систем).

2) *Угрозы*, *которые могут проявляться только в процессе автоматизированной обработки данных* (например, угрозы выполнения и распространения программных вирусов).

6. *По степени воздействия на АС.*

1) *Пассивные угрозы*, которые при реализации ничего не меняют в структуре и содержании АС (угроза копирования секретных данных).

2) *Активные угрозы*, которые при воздействии вносят изменения в структуру и содержание АС:

– внедрение аппаратных спецвложений, программных "закладок" и "вирусов" ("троянских коней" и "жучков"), т.е. таких участков программ, которые не нужны для выполнения заявленных функций, но позволяют преодолеть систему защиты, скрытно и незаконно осуществить доступ к системным ресурсам с целью регистрации и передачи критической информации или дезорганизации функционирования системы;

– действия по дезорганизации функционирования системы (изменение режимов работы устройств или программ, забастовка, саботаж персонала, постановка мощных активных радиопомех на частотах работы устройств системы и т.п.);

– угроза умышленной модификации информации.

7. *По этапам доступа пользователей или программ к ресурсам АС.*

1) *Угрозы*, *которые могут проявляться на этапе доступа к ресурсам АС* (например, угрозы несанкционированного доступа в АС).

2) *Угрозы*, *которые могут проявляться после разрешения доступа к ресурсам АС* (например, угрозы несанкционированного или некорректного использования ресурсов АС).

8. *По способу доступа к ресурсам АС.*

1) *Угрозы*, *направленные на использование прямого стандартного пути доступа к ресурсам АС*:

– незаконное получение паролей и других реквизитов разграничения доступа (агентурным путем, используя халатность пользователей, подбором, имитацией интерфейса системы и т.д.) с последующей маскировкой под зарегистрированного пользователя ("маскарад");

– несанкционированное использование терминалов пользователей, имеющих уникальные физические характеристики, такие как номер рабочей станции в сети, физический адрес, адрес в системе связи, аппаратный блок кодирования и т.п.

2) *Угрозы*, *направленные на использование скрытого нестандартного пути доступа к ресурсам АС*:

– вход в систему в обход средств защиты (загрузка посторонней операционной системы со сменных магнитных носителей и т.п.);

– угроза несанкционированного доступа к ресурсам АС путем использования недокументированных возможностей ОС.

9. *По текущему месту расположения информации, хранимой и обрабатываемой в АС.*

1) *Угрозы доступа к информации на внешних запоминающих устройствах* (например, угроза несанкционированного копирования секретной информации с жесткого диска).

2) *Угрозы доступа к информации в оперативной памяти*:

– чтение остаточной информации из оперативной памяти;

– чтение информации из областей оперативной памяти, используемых операционной системой (в том числе подсистемой защиты) или другими пользователями, в асинхронном режиме, используя недостатки мультизадачных АС и систем программирования;

– угроза доступа к системной области оперативной памяти со сторон прикладных программ.

3) *Угрозы доступа к информации*, *циркулирующей в линиях связи*:

– незаконное подключение к линиям связи с целью работы во время пауз в действиях законного пользователя от его имени с вводом ложных сообщений или модификацией передаваемых сообщений;

– незаконное подключение к линиям связи с целью прямой подмены законного пользователя путем его физического отключения после входа в систему и успешной аутентификации с последующим вводом дезинформации и навязыванием ложных сообщений;

– перехват всего потока данных с целью дальнейшего анализа не в реальном масштабе времени.

4) *Угрозы доступа к информации, отображаемой на терминале или печатаемой на принтере* (например, угроза записи отображаемой информации на скрытую видеокамеру). Вне зависимости от конкретных видов угроз или их проблемно-ориентированной классификации АС удовлетворяет потребности эксплуатирующих ее лиц, если обеспечиваются следующие свойства информации систем ее обработки.

В качестве основного критерия будем использовать аспект ИБ, привлекая при необходимости остальные.

Угроза доступности (отказа служб) возникает всякий раз, когда в результате преднамеренных действий, предпринимаемых другим пользователем или злоумышленником, блокируется доступ к некоторому ресурсу вычислительной системы. Реально блокирование может быть постоянным – запрашиваемый ресурс никогда не будет получен, или оно может вызывать

только задержку запрашиваемого ресурса, достаточно долгую для того, чтобы он стал бесполезным. В этих случаях говорят, что ресурс исчерпан.

Доступность информации – свойство системы (среды, средств и технологии обработки), в которой циркулирует информация, характеризующаяся способностью обеспечивать своевременный беспрепятственный доступ субъектов к интересующей их информации и готовность соответствующих автоматизированных служб к обслуживанию поступающих от субъектов запросов всегда, когда возникает в этом необходимость.

Самыми частыми и самыми опасными (с точки зрения размера ущерба) являются *непреднамеренные ошибки* штатных пользователей, операторов, системных администраторов и других лиц, обслуживающих ИС. Эти ошибки и являются собственно угрозами (неправильно введенные данные или ошибка в программе), иногда они создают уязвимые места, которые используют злоумышленники (по данным до 65 % потерь – от непреднамеренных ошибок).

Другие *угрозы доступности* классифицируем по *компонентам ИС*, на которые нацелены угрозы:

- отказ пользователей;
- внутренний отказ информационной системы;
- отказ поддерживающей инфраструктуры.

Обычно, применительно к *пользователям*, рассматриваются следующие угрозы: нежелание работать с информационной системой; невозможность работать с системой в силу отсутствия соответствующей подготовки; невозможность работать с системой в силу отсутствия технической поддержки (неполнота документации, недостаток справочной информации и т.п.).

Основными *источниками внутренних отказов* являются: отступление от установленных правил эксплуатации; выход системы из штатного режима эксплуатации в силу случайных или преднамеренных действий пользователей или обслуживающего персонала (превышение расчетного числа запросов, чрезмерный объем обрабатываемой информации и т.п.); ошибки при (пере)конфигурировании системы; отказы программного и аппаратного обеспечения; разрушение данных; разрушение или повреждение аппаратуры.

По *отношению к поддерживающей инфраструктуре* рекомендуется рассматривать следующие угрозы: нарушение работы (случайное или умышленное) систем связи, электропитания, водо- и/или теплоснабжения, кондиционирования; разрушение или повреждение помещений; невозможность или нежелание обслуживающего персонала и/или пользователей выполнять свои обязанности.

Приведем некоторые примеры угроз и программных атак на доступность.

В качестве средства вывода системы из штатного режима эксплуатации может использоваться *агрессивное потребление ресурсов* (обычно – полосы пропускания сетей, вычислительных возможностей процессоров или ОЗУ). По расположению источника угрозы такое *потребление* подразделяется на *локальное* и *удаленное*. При просчетах в конфигурации системы локальная программа способна практически монополизировать процессор и/или физическую память, сведя скорость выполнения других программ к нулю.

Удаленное потребление ресурсов в последнее время проявляется в особенно опасной форме – как скоординированные распределенные атаки, когда на сервер с множества разных адресов с максимальной скоростью направляются вполне легальные запросы на соединение и/или обслуживание.

Угроза нарушения целостности включает в себя любое умышленное изменение информации, хранящейся в вычислительной системе или передаваемой из одной системы в другую, в том числе и несанкционированное изменение информации при случайных ошибках программного или аппаратного обеспечения. Санкционированными изменениями являются те, которые сделаны уполномоченными лицами с обоснованной целью (например, периодическая запланированная коррекция некоторой базы данных).

Целостность информации – существование информации в неискаженном виде (неизменном по отношению к некоторому фиксированному ее состоянию). Обычно субъектов интересует обеспечение более широкого свойства – достоверности информации, которое складывается из адекватности (полноты и точности) отображения состояния предметной области и непосредственно целостности информации, т.е. ее неискаженности.

Существует различие между *статической* и *динамической целостностью*. С целью нарушения статической целостности злоумышленник может: ввести неверные данные, изменить данные.

Угрозой целостности является не только фальсификация или изменение данных, но и отказ от совершенных действий. Если нет средств обеспечить *"неотказуемость"*, компьютерные данные не могут рассматриваться в качестве доказательства.

Потенциально уязвимы с точки зрения нарушения *целостности* не только *данные*, но и *программы*. Внедрение рассмотренного выше вредоносного ПО – пример подобного нарушения.

Угрозами динамической целостности являются нарушение атомарности транзакций, переупорядочение, кража, дублирование данных или внесение дополнительных сообщений (сетевых пакетов и т.п.). Соответствующие действия в сетевой среде называются активным прослушиванием.

Угроза нарушения конфиденциальности заключается в том, что информация становится известной тому, кто не располагает полномочиями доступа к ней. В связи с данной угрозой используется термин "утечка".

Конфиденциальность информации – субъективно определяемая (приписываемая) характеристика (свойство) информации, указывающая на необходимость введения ограничений на круг субъектов, имеющих доступ к данной информации, и обеспечиваемая способностью системы (среды) сохранять указанную информацию в тайне от субъектов, не имеющих полномочий доступа к ней. Объективные предпосылки подобного ограничения доступности информации для одних субъектов заключены в необходимости защиты их законных интересов от других субъектов информационных отношений.

Конфиденциальную информацию можно разделить на предметную и служебную. Служебная информация (например, пароли пользователей) не относится к определенной предметной области, в информационной системе она играет техническую роль, но ее раскрытие особенно опасно, поскольку оно чревато получением несанкционированного доступа ко всей информации, в том числе предметной.

Многим субъектам приходится выступать в качестве пользователей не одной, а целого ряда систем (информационных сервисов). Если для доступа к таким системам используются многоразовые пароли или иная конфиденциальная информация, то наверняка эти данные будут храниться не только в голове, но и в записной книжке или на листках бумаги, которые пользователь часто оставляет на рабочем столе, а то и попросту теряет. И дело здесь не в неорганизованности людей, а в изначальной непригодности парольной схемы. Невозможно помнить много разных паролей; рекомендации по их регулярной смене только усугубляют положение, заставляя применять несложные схемы чередования или стараться свести дело к двум-трем легко запоминаемым паролям.

Описанный класс уязвимых мест можно назвать размещением конфиденциальных данных в среде, где им не обеспечена необходимая защита. Угроза же состоит в том, что кто-то не откажется узнать секреты, которые сами просятся в руки. Помимо паролей, хранящихся в записных книжках пользователей, в этот класс попадает передача конфиденциальных данных в открытом виде (в разговоре, в письме, по сети), которая делает возможным *перехват данных*. Для атаки могут использоваться разные технические средства (подслушивание или прослушивание разговоров, пассивное прослушивание сети и т.п.), но идея одна – получить доступ к данным в тот момент, когда они наименее защищены.

Перехват данных – очень серьезная угроза, и если конфиденциальность действительно является критичной, а данные передаются по многим каналам, их защита может оказаться весьма сложной и дорогостоящей. Технические средства перехвата хорошо проработаны, доступны, просты в эксплуатации, а установить их, например на кабельную сеть, может кто угодно, так что эту угрозу нужно принимать во внимание по отношению не только к внешним, но и к внутренним коммуникациям.

Опасной нетехнической угрозой конфиденциальности являются *методы морально-психологического воздействия*, такие как *маскарад* – выполнение действий под видом лица, обладающего полномочиями для доступа к данным

К неприятным угрозам, от которых трудно защищаться, можно отнести *злоупотребление полномочиями*. На многих типах систем привилегированный пользователь (например системный администратор) способен прочитать любой (незашифрованный) файл, получить доступ к почте любого пользователя и т.д. Другой пример – нанесение ущерба при сервисном обслуживании. Обычно сервисный инженер получает неограниченный доступ к оборудованию и имеет возможность действовать в обход программных защитных механизмов.

Таковы основные угрозы, которые наносят наибольший ущерб субъектам информационных отношений.

На современном этапе развития информационных технологий подсистемы или функции защиты являются неотъемлемой частью комплекса по обработке информации. Информация не представляется "в чистом виде", на пути к ней имеется хотя бы какая-нибудь система защиты, и поэтому чтобы угрожать, атакующая сторона должна преодолеть эту систему. Однако не существует абсолютно стойкой системы защиты, вопрос лишь во времени и средствах, требующихся на ее преодоление. Исходя из данных условий, примем следующую модель: ***защита информационной системы считается преодоленной, если в ходе ее исследования определены все уязвимости системы***. Поскольку преодоление защиты также представляет собой угрозу, для защищенных систем будем рассматривать ее четвертый вид – ***угрозу раскрытия параметров АС***, включающей в себя систему защиты. С точки зрения практики любое проводимое мероприятие предваряется этапом разведки, в ходе которого определяются основные параметры системы, ее характеристики, в результате чего уточняется поставленная задача и выбираются оптимальные технические средства.

Угрозу раскрытия можно рассматривать как опосредованную. Последствия ее реализации не причиняют какой-либо ущерб обрабатываемой информации, но дают возможность реализоваться первичным или непосредственным угрозам, перечисленным выше. Введение данного вида угроз позволяет описывать с научно-методологической точки зрения отличия защищенных информационных систем от открытых. Для последних угроза разведки параметров системы считается реализованной.

7.2. Методы обеспечения информационной безопасности

7.2.1. СТРУКТУРИЗАЦИЯ МЕТОДОВ ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

При рассмотрении вопросов защиты АС целесообразно использовать четырехуровневую градацию доступа к хранимой, обрабатываемой и защищаемой АС информации, которая поможет систематизировать как возможные угрозы, так и меры по их нейтрализации и парированию, т.е. может систематизировать и обобщить весь спектр методов обеспечения защиты, относящихся к информационной безопасности:

1) уровень носителей информации;
2) уровень средств взаимодействия с носителем;
3) уровень представления информации;
4) уровень содержания информации.

Данные уровни были введены исходя из того, что, во-первых, информация для удобства манипулирования чаще всего фиксируется на некотором материальном носителе, которым может быть бумага, дискета, CD-диск и т.п. Во-вторых, если способ представления информации таков, что она не может быть непосредственно воспринята человеком, возникает необходимость в преобразованиях информации в доступный для человека способ представления. Например, для чтения информации с дискеты необходим компьютер, оборудованный дисководом соответствующего типа. В-третьих, информация может быть охарактеризована способом своего представления. Язык жестов, язык символов и другие – все это способы представления информации. В-четвертых, человеку должен быть доступен смысл представленной информации (семантика).

Защита носителей информации должна обеспечивать парирование всех возможных угроз, направленных как на сами носители, так и на зафиксированную на них информацию, представленную в виде изменения состояний отдельных участков, блоков, полей носителя. Применительно к АС защита носителей информации в первую очередь подразумевает защиту машинных носителей. Вместе с тем, необходимо учитывать, что носителями информации являются также каналы связи, документаль-

ные материалы, получаемые в ходе эксплуатации АС, и т.п. Защита средств взаимодействия с носителем охватывает спектр методов защиты программно-аппаратных средств, входящих в состав АС, таких как средства вычислительной техники, операционная система, прикладные программы. В основном защита на данном уровне рассматривается как защита от несанкционированного доступа, обеспечивающая разграничение доступа пользователей к ресурсам системы. Защита представления информации, т.е. некоторой последовательности символов, обеспечивается средствами криптографической защиты. Защита содержания информации обеспечивается семантической защитой данных.

7.1. Основные методы реализации угроз информационной безопасности АС

Уровень доступа к информации в АС	Основные методы реализации угроз информационной безопасности			
	Угроза раскрытия параметров системы	Угроза нарушения конфиденциальности	Угроза нарушения целостности	Угроза отказа доступа к информации (отказа служб)
Носителей информации	Определение злоумышленником типа и параметров носителей информации	Хищение (копирование) носителей информации, имеющих конфиденциальные данные. Использование специальных технических средств для перехвата побочных электромагнитных излучений и наводок (ПЭМИН) – конфиденциальные данные перехватываются злоумышленником изменением информативных сигналов из ЭМИ и наводок по цепям питания средств ВТ, входящей в АС	Уничтожение машинных носителей информации	Выведение из строя машинных носителей информации без уничтожения информации – выведение из строя электронных блоков в ней на жестких дисках и т.п.
Средств взаимодействия с носителем	Получение злоумышленником информации о программно-аппаратной среде: типе и параметрах средств ВТ, типе и версии ОС, составе прикладного ПО.	Несанкционированный доступ пользователя к ресурсам АС путем преодоления систем защиты с использованием спецсредств, приемов, методов.	Уничтожение средств ВТ. Внесение пользователем несанкционированных изменений в программно-аппаратные компоненты АС и обрабатываемые данных.	Проявление ошибок проектирования и разработки программно-аппаратных компонентов АС.
Средств взаимодействия с носителем	Получение злоумышленником детальной информации о функциях, выполняемых АС. Получение злоумышленником данных о применяемых системах защиты	Несанкционированное превышение пользователем полномочий. Несанкционированное копирование программного обеспечения. Перехват данных, передаваемых по каналам связи	Установка и использование нештатного аппаратного и/или программного обеспечения. Заражение программными вирусами	Обход (отключение) механизмов защиты – загрузка злоумышленника нештатной операционной системы с дискеты, использование режимов программно-аппаратных компонент АС и т.п.
Представления информации	Определение способа представления информации	Визуальное наблюдение – конфиденциальные данные считываются с экранов терминалов, распечаток в процессе их печати и т.п. Раскрытие представления информации (дешифрование данных)	Внесение искажений в представление данных, уничтожение на уровне представления, искажение информации при передаче по ЛС. Уничтожение данных	Искажение соответствия синтаксических и семантических конструкций языка – установление новых значений слов, выражений и т.п.
Содержания информации	Определение злоумышленником содержания данных, обрабатываемых в АС, на качественном уровне (мониторинг дешифрования сообщений)	Раскрытие содержания информации на семантическом уровне к смысловой составляющей информации, хранящейся в АС	Внедрение дезинформации	Запрет на использование информации – имеющаяся информация по каким-либо причинам не может быть использована

7.2.2. КЛАССИФИКАЦИЯ ЗЛОУМЫШЛЕННИКОВ

Возможности осуществления вредительских воздействий зависят от статуса злоумышленника по отношению к ИВС. Злоумышленником может быть:

- ***разработчик ИВС*** (владеет наиболее полной информацией о программных и аппаратных средствах ИВС и имеет возможность внедрения "закладок" на этапах создания и модернизации систем, но не получает доступа на эксплуатируемые объекты ИВС);
- ***сотрудник из числа обслуживающего персонала*** (работники службы безопасности информации, системные и прикладные программисты, инженерно- технический персонал);

- ***пользователь*** (имеет общее представление о структуре ИВС и механизмах ее защиты, но может осуществлять сбор информации методами традиционного шпионажа и попытками НСДИ);
- ***постороннее лицо*** (дистанционные методы шпионажа и диверсионная деятельность).

7.2.3. ОСНОВНЫЕ НАПРАВЛЕНИЯ И МЕТОДЫ РЕАЛИЗАЦИИ УГРОЗ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

К ***основным направлениям*** реализации злоумышленником информационных угроз относятся:

- непосредственное обращение к объектам доступа;
- создание программных и технических средств, выполняющих обращение к объектам доступа в обход средств защиты;
- модификация средств защиты, позволяющая реализовать угрозы ИБ;
- внедрение в технические средства АС программных или технических механизмов, нарушающих предполагаемую структуру и функции АС.

К числу *основных методов реализации угроз* ИБ АС относятся методы, приведенные в табл. 7.1.

Контрольные вопросы к теме 7

1. Что понимается под угрозой информации? Назовите разновидности угроз информации.
2. Приведите классификацию угроз информации.
3. Какие основные направления и методы реализации угроз вам известны?
4. Поясните классификацию злоумышленников.
5. Охарактеризуйте причины и виды утечки информации.
6. Назовите и приведите примеры каналов утечки информации.

Т е м а 8. ПРОГРАММНО-ТЕХНИЧЕСКИЙ УРОВЕНЬ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

8.1. ОСНОВНЫЕ ПОНЯТИЯ ПРОГРАММНО-ТЕХНИЧЕСКОГО УРОВНЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Программно-технические меры, т.е. меры, направленные на контроль компьютерных сущностей – оборудования, программ и/или данных – образуют последний и самый важный рубеж информационной безопасности. Напомним, что ущерб наносят в основном действия легальных пользователей, по отношению к которым процедурные регуляторы малоэффективны. Главные враги – некомпетентность и неаккуратность при выполнении служебных обязанностей, и только программно-технические меры способны им противостоять.

В связи с этим именно компьютер (особенно находящийся в составе сети) следует в первую очередь рассматривать в качестве объекта защиты, а конечного пользователя – в качестве ее наиболее вероятного потенциального нарушителя. Как следствие, под сомнение ставится обоснованность концепции реализованной системы защиты в современных универсальных ОС. Эта система защиты заключается в построении распределенной схемы администрирования механизмов защиты, элементами которой, помимо администратора, выступают пользователи, имеющие возможность назначать и изменять права доступа к создаваемым ими файловым объектам.

На практике сегодня существует два подхода к обеспечению компьютерной безопасности:

1) использование только встроенных в ОС и приложения средств защиты;

2) применение, наряду со встроенными, дополнительных механизмов защиты. Этот подход заключается в использовании так называемых технических средств добавочной защиты – программных, либо программно-аппаратных комплексов, устанавливаемых на защищаемые объекты.

Существующая статистика ошибок, обнаруженных в ОС, а также сведения о недостаточной эффективности встроенных в ОС и приложения механизмов защиты, заставляет специалистов сомневаться в достижении гарантированной защиты от НСД, при использовании встроенных механизмов, и все большее внимание уделять средствам добавочной защиты информации.

Таким образом, важнейшим условием защищенности компьютерной информации является квалификация администраторов безопасности и сотрудников эксплуатирующих служб, которая, по крайней мере, не должна уступать квалификации злоумышленников, в противном случае не помогут никакие средства защиты.

Центральным для программно-технического уровня является понятие ***сервиса безопасности***.

Далее рассмотрим следующие ***основные*** и ***вспомогательные*** сервисы:

- ***идентификация*** и ***аутентификация***;
- ***управление доступом***;
- ***протоколирование*** и ***аудит***;
- ***шифрование***;
- ***контроль целостности***;
- ***экранирование***;
- ***анализ защищенности***;
- обеспечение ***отказоустойчивости***;
- обеспечение ***безопасного восстановления***;
- ***туннелирование***;
- ***управление***.

Совокупность перечисленных выше сервисов безопасности называют полным набором. Считается, что его, в принципе, достаточно для построения надежной защиты на программно-техническом уровне, правда, при соблюдении целого ряда дополнительных условий (отсутствие уязвимых мест, безопасное администрирование и т.д.).

Для проведения классификации сервисов безопасности и определения их места в общей архитектуре меры безопасности можно разделить на следующие виды:

- ***превентивные***, препятствующие нарушениям ИБ;
- меры ***обнаружения нарушений***;
- ***локализующие***, сужающие зону воздействия нарушений;
- меры по ***выявлению нарушителя***;
- меры ***восстановления*** режима безопасности.

Большинство сервисов безопасности попадает в число превентивных. Аудит и контроль целостности способны помочь в обнаружении нарушений; активный аудит, кроме того, позволяет запрограммировать реакцию на нарушение с целью локализации и/или прослеживания. Направленность сервисов отказоустойчивости и безопасного восстановления очевидна. Управление играет инфраструктурную роль, обслуживая все аспекты ИС.

Идентификацию и ***аутентификацию*** можно считать основой программно-технических средств безопасности, поскольку остальные сервисы рассчитаны на обслуживание именованных субъектов. Идентификация и аутентификация – это первая линия обороны, "проходная" информационного пространства организации.

Идентификация позволяет субъекту (пользователю, процессу, действующему от имени определенного пользователя, или иному аппаратно-программному компоненту) назвать себя (сообщить свое имя). Посредством аутентификации вторая сторона убеждается, что субъект действительно тот, за кого он себя выдает. В качестве синонима слова "аутентификация" иногда используют словосочетание "проверка подлинности".

Аутентификация бывает односторонней (обычно клиент доказывает свою подлинность серверу) и *двусторонней* (взаимной). Пример односторонней аутентификации – процедура входа пользователя в систему.

В сетевой среде, когда стороны идентификации/аутентификации территориально разнесены, у рассматриваемого сервиса есть два основных аспекта:

- что служит *аутентификатором* (т.е. используется для подтверждения подлинности субъекта);
- как организован (и защищен) обмен данными идентификации/аутентификации.

Субъект может подтвердить свою подлинность, предъявив, по крайней мере, одну из следующих сущностей:

- нечто, что он знает (пароль, личный идентификационный номер, криптографический ключ и т.п.);
- нечто, чем он владеет (личную карточку или иное устройство аналогичного назначения);
- нечто, что есть часть его самого (голос, отпечатки пальцев и т.п., т.е. свои биометрические характеристики).

В открытой сетевой среде между сторонами идентификации/аутентификации не существует доверенного маршрута; это значит, что в общем случае данные, переданные субъектом, могут не совпадать с данными, полученными и использованными для проверки подлинности. Необходимо обеспечить защиту от пассивного и активного прослушивания сети, т.е. от *перехвата*, *изменения* и/или *воспроизведения* данных. Передача паролей в открытом виде, очевидно, неудовлетворительна; не спасает положение и шифрование паролей, так как оно не защищает от воспроизведения. Нужны более сложные протоколы аутентификации.

Надежная идентификация и аутентификация затруднена не только из-за сетевых угроз, но и по целому ряду причин. Во-первых, почти все аутентификационные сущности можно узнать, украсть или подделать. Во-вторых, имеется противоречие между надежностью аутентификации, с одной стороны, и удобствами пользователя и системного администратора – с другой. В-третьих, чем надежнее средство защиты, тем оно дороже.

Современные средства идентификации/аутентификации должны поддерживать концепцию единого входа в сеть. *Единый вход в сеть* – это, в первую очередь, требование удобства для пользователей. Если в корпоративной сети много информационных сервисов, допускающих независимое обращение, то многократная идентификация/аутентификация становится слишком обременительной. К сожалению, пока нельзя сказать, что единый вход в сеть стал нормой, доминирующие решения пока не сформировались.

Парольная аутентификация – проста и уже давно встроена в операционные системы и иные сервисы. При правильном использовании пароли могут обеспечить приемлемый для многих организаций уровень безопасности. Тем не менее, по совокупности характеристик их следует признать самым слабым средством проверки подлинности. Чтобы пароль был запоминающимся, его зачастую делают простым (имя подруги, название спортивной команды и т.п.). Иногда пароли с самого начала не хранятся в тайне, так как имеют стандартные значения, указанные в документации, и далеко не всегда после установки системы производится их смена.

Ввод пароля можно подсмотреть. Иногда для подглядывания используются даже оптические приборы. Пароли нередко сообщают коллегам, чтобы те могли, например, подменить на некоторое время владельца пароля. Пароль можно угадать "методом грубой силы", используя, скажем, словарь. Если файл паролей зашифрован, но доступен для чтения, его можно скачать к себе на компьютер и попытаться подобрать пароль, запрограммировав полный перебор (предполагается, что алгоритм шифрования известен).

Тем не менее, следующие меры позволяют значительно повысить надежность парольной защиты:

- ***наложение технических ограничений*** (пароль должен быть не слишком коротким, он должен содержать буквы, цифры, знаки пунктуации и т.п.);
- ***управление сроком действия паролей***, их периодическая смена;
- ограничение доступа к файлу паролей;
- ограничение числа неудачных попыток входа в систему (это затруднит применение "метода грубой силы");
- обучение пользователей;

– использование программных *генераторов паролей* (программа, основываясь на несложных правилах, порождает только благозвучные и, следовательно, запоминающиеся пароли).

Рассмотренные выше *пароли можно назвать многоразовыми*; их раскрытие позволяет злоумышленнику действовать от имени легального пользователя. Гораздо более сильным средством, устойчивым к пассивному прослушиванию сети, являются *одноразовые пароли*.

Наиболее известным программным генератором одноразовых паролей является система **S/KEY** компании Bellcore. Идея этой системы состоит в следующем. Пусть имеется *односторонняя функция f* (т.е. функция, вычислить обратную которой за приемлемое время не представляется возможным). Эта функция известна и пользователю, и *серверу аутентификации*. Пусть, далее, имеется *секретный ключ* K, известный только пользователю.

На этапе начального администрирования пользователя функция *f* применяется к ключу K *n* раз, после чего результат сохраняется на сервере. После этого процедура проверки подлинности пользователя выглядит так:

– сервер присылает на пользовательскую систему число $(n-1)$;

– пользователь применяет функцию *f* к секретному ключу K $(n-1)$ раз и отправляет результат по сети на сервер аутентификации;

– сервер применяет функцию *f* к полученному от пользователя значению и сравнивает результат с ранее сохраненной величиной.

В случае совпадения подлинность пользователя считается установленной, сервер запоминает новое значение (присланное пользователем) и уменьшает на единицу счетчик (*n*).

Поскольку функция *f* необратима, перехват пароля, равно как и получение доступа к серверу аутентификации, не позволяют узнать секретный ключ K и предсказать следующий одноразовый пароль.

Система S/KEY имеет статус Internet-стандарта (RFC 1938).

Другой подход к надежной аутентификации состоит в генерации нового пароля через небольшой промежуток времени (например, каждые 60 секунд), для чего могут использоваться программы или специальные интеллектуальные карты. Серверу аутентификации должен быть известен алгоритм генерации паролей и ассоциированные с ним параметры; кроме того, часы клиента и сервера должны быть синхронизированы.

Биометрическая аутентификация представляет собой совокупность автоматизированных методов идентификации и/или аутентификации людей на основе их физиологических и поведенческих характеристик. К числу физиологических характеристик принадлежат особенности отпечатков пальцев, сетчатки и роговицы глаз, геометрия руки и лица и т.п. К поведенческим характеристикам относятся динамика подписи (ручной), стиль работы с клавиатурой. На стыке физиологии и поведения находятся анализ особенностей голоса и распознавание речи.

В общем виде работа с биометрическими данными организована следующим образом. Сначала создается и поддерживается база данных характеристик потенциальных пользователей. Для этого биометрические характеристики пользователя снимаются, обрабатываются, и результат обработки (называемый биометрическим шаблоном) заносится в базу данных (исходные данные, такие как результат сканирования пальца или роговицы, обычно не хранятся).

В дальнейшем для идентификации (и одновременно аутентификации) пользователя процесс снятия и обработки повторяется, после чего производится поиск в базе данных шаблонов. В случае успешного поиска личность пользователя и ее подлинность считаются установленными. Для аутентификации достаточно произвести сравнение с одним биометрическим шаблоном, выбранным на основе предварительно введенных данных.

На наш взгляд, к биометрии следует относиться весьма осторожно. Необходимо учитывать, что она подвержена тем же угрозам, что и другие методы аутентификации. Во-первых, биометрический шаблон сравнивается не с результатом первоначальной обработки характеристик пользователя, а с тем, что пришло к месту сравнения. А, как известно, за время пути... много чего может произойти. Во-вторых, биометрические методы не более надежны, чем база данных шаблонов. В-третьих, следует учитывать разницу между применением биометрии на контролируемой территории, под бдительным оком охраны, и в "полевых" условиях, когда, например к устройству сканирования роговицы могут поднести муляж и т.п. В-четвертых, биометрические данные человека меняются, так что база шаблонов нуждается в сопровождении, что создает определенные проблемы и для пользователей, и для администраторов.

Но главная опасность состоит в том, что любая "пробоина" для биометрии оказывается фатальной. Пароли, при всей их ненадежности, в крайнем случае можно сменить. Утерянную аутентификационную карту можно аннулировать и завести новую. Палец же, глаз или голос сменить нельзя. Если биометрические данные окажутся скомпрометированы, придется как минимум производить существенную модернизацию всей системы.

8.2. ТРЕБОВАНИЯ К ЗАЩИТЕ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ

8.2.1. ОБЩИЕ ПОЛОЖЕНИЯ

Естественным ходом развития информационных технологий явился принципиальный переход от открытости к защищенности при построении информационных систем. На сегодняшний день большинство программных продуктов, применяющихся для построения информационных систем, обладают встроенными средствами защиты.

Данную тенденцию наглядно иллюстрирует развитие ОС MS Windows. Можно четко проследить развитие встроенных в ОС механизмов защиты от Windows 3.1 (где механизмы защиты практически отсутствовали), к Windows NT (где механизмы защиты интегрированы в ядро ОС) и к Windows 2000 (где интеграция захватывает и внешние по отношению к разработчикам технологии защиты, такие как Kerberos, IP-тунелирование и т.д.). То же самое можно сказать и о других семействах ОС. Например, в ОС FreeBSD в каждой из новых версий появляются новые механизмы защиты (firewall, nat). Такое же развитие средств защиты касается и прикладного программного обеспечения (ПО).

С учетом сказанного возникает ряд вопросов:

Достаточно ли встроенных в современные ОС и приложения механизмов защиты для обеспечения гарантированной защиты информации от НСД?

В предположении, что встроенных механизмов защиты недостаточно, то чем это вызвано?

Почему защищенность компьютерной информации остается недостаточной, несмотря на устойчивую тенденцию к усилению встроенных в современные универсальные ОС и приложения механизмов защиты?

Каким образом следует усиливать встроенные механизмы добавочными средствами защиты?

Какие функции должны обеспечивать и какими характеристиками должны обладать системы встроенной и добавочной защиты, чтобы обеспечить надежное противодействие попыткам НСД?

В предположении, что добавочные механизмы защиты необходимы, каким образом комплексировать в защищаемой вычислительной системе встроенные и добавочные механизмы защиты?

Используемые в настоящее время на практике подходы к защите компьютерной информации определяются следующим характеристиками:

- формализованными требованиями к набору и параметрам механизмов защиты, регламентирующими современные требования к обеспечению компьютерной безопасности;
- реальными механизмами защиты, реализуемыми при защите компьютерной информации (к ним относятся, прежде всего, средства защиты ОС, так как большинство приложений используют встроенные в ОС механизмы защиты);
- существующей статистикой угроз компьютерной безопасности – существующими успешными атаками на информационные компьютерные ресурсы.

Рассмотрим формализованные требования, т.е. требования соответствующих нормативных документов, регламентирующих требования к защите компьютерной информации от несанкционированного доступа. Затем рассмотрим механизмы защиты, реализованные в современных ОС и проанализируем, в какой мере ими выполняются требования соответствующих нормативных документов.

Этот анализ необходим, чтобы определиться с причиной низкой защищенности компьютерной информации. Если встроенные в современные ОС механизмы защиты в полной мере соответствуют формализованным требованиям к ним, то, следовательно, эти требования необходимо усиливать. В противном случае необходимо усиливать механизмы защиты с целью выполнения соответствующих требований.

Следующим шагом будет рассмотрение и анализ существующей статистики угроз компьютерной информации и определение причины уязвимости современных ОС. На основе этого, а также на основе нормативных документов определим дополнительные требования к защите компьютерной информации, которые не выполняются встроенными в ОС механизмами защиты.

Затем рассмотрим непосредственно методы и механизмы, которые должны быть реализованы средствами добавочной защиты в дополнение к встроенным в ОС защитным механизмам. Особое внимание уделим методам и механизмам добавочной защиты, позволяющим функционально расширять встроенные механизмы защиты в предположении возможности потенциальных угроз. Также отдельно будут рассмотрены архитектурные и технические решения по реализации добавочной защиты.

8.2.2. КЛАССИФИКАЦИЯ ТРЕБОВАНИЙ К СИСТЕМАМ ЗАЩИТЫ

В общем случае следует говорить о необходимости учета двух (дополняющих друг друга) групп требований к системе защиты.

Первая группа требований (необходимые требования) заключается в необходимости реализации системой защиты формализованных мер безопасности.

Формализованные требования к необходимым механизмам защиты информационных систем, в части их защиты от НСД, сформулированы в руководящих документах Гостехкомиссии России. Для других стран эти требования сформулированы в документах соответствующих организаций, например:

– "Оранжевая книга" Критерии оценки надежных компьютерных систем Министерства обороны США;

– "Согласованные критерии оценки безопасности информационных технологий" (Information Technology Security Evaluation Criteria, ITSEC, Европейские страны).

При этом отметим, что данные документы носят общий характер. В них не в полной мере предусматривается классификация объектов, для которых должна быть реализована защита. В частности, эти документы предъявляют единые требования для всех семейств ОС. И это несмотря на то, что ОС различных семейств имеют принципиально отличные принципы построения, а значит, для них различаются и способы НСД.

В указанных руководящих документах не дается рекомендаций по способам построения и администрирования защищенных систем, т.е. не сказано, как их строить. В этих документах лишь сформулированы требования к механизмам защиты информации и, отчасти, требования к их количественным характеристикам.

Данный подход к заданию формализованных требований, наверное, в целом оправдан, так как невозможно учесть в нормативных документах все тонкости построения и проектирования средств защиты сложных информационных систем, особенно при существующей динамике их развития.

Формализованные требования носят основополагающий характер – в том смысле, что в них формализованы требования к основополагающим механизмам защиты информации. Поэтому их возможное со временем изменение не может быть сколько-нибудь существенным без появления новых научно обоснованных технологий защиты информации. Само же изменение неизбежно и естественно ввиду меняющейся со временем статистики угроз информационной безопасности.

Вторая группа требований (дополнительные требования) заключается в необходимости учета существующей (текущей) статистики угроз для конкретного типа защищаемого объекта, а также потенциально возможных угроз. Необходимость этой группы требований обусловлена тем, что формализованные требования не могут учитывать все возможные угрозы объектам всех типов, требующих защиты.

С учетом сказанного может быть сделан вывод о целесообразности рассмотрения условий необходимости и достаточности требований к защите информации. Необходимыми являются формализованные требования, определяемые соответствующими нормативными документами в области защиты информации. Достаточной является совокупность формализованных и дополнительных требований, формулируемых на основе анализа текущей статистики угроз защищаемому объекту, а также потенциально возможных угроз.

8.2.3. ФОРМАЛИЗОВАННЫЕ ТРЕБОВАНИЯ К ЗАЩИТЕ ИНФОРМАЦИИ ОТ НСД. ОБЩИЕ ПОДХОДЫ К ПОСТРОЕНИЮ СИСТЕМ ЗАЩИТЫ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ

Как было сказано ранее, общие подходы в системах защиты целесообразно рассматривать относительно соответствия их принятым формализованным требованиям. Эти требования предъявляются к механизмам защиты, которые в той или иной мере реализуются современными ОС, приложениями и добавочными средствами защиты.

Защиту информации от НСД в нашей стране на сегодняшний день регламентируют нормативные документы Гостехкомиссии РФ:

- требования к защите средств вычислительной техники (СВТ), формализуют условия защищенности отдельно взятого средства – ОС, СУБД, приложения;
- требования к защите автоматизированных систем (АС), формализуют условия защищенности объекта с учетом:

– совокупности механизмов защиты, реализуемых установленными на защищаемом объекте средствами, включая ОС, СУБД, приложениями, добавочными механизмами защиты;

– дополнительные организационные меры, принимаемые для безопасного функционирования АС.

При этом отметим, что, как правило, приложения используют механизмы защиты ОС. Что касается СУБД, то механизмы защиты СУБД дополняют защитные механизмы ОС, так как возникает необходимость защиты дополнительных объектов доступа – "таблиц". И действительно, для ОС защищаемыми файловыми объектами являются: логические диски (тома), каталоги, файлы. При работе с базами данных сложность обусловлена тем, что таблицы различных пользователей, к которым должен разграничиваться доступ, могут находиться в одном файле. Таким образом, получается, что в общем случае невозможно управлять доступом к базам данных только механизмами защиты ОС. Однако это относится только к СУБД. Остальные приложения обычно довольствуются защитными механизмами ОС, т.е. можно сказать, что, как правило, все механизмы защиты автоматизированных систем (АС) по умолчанию реализуются собственно ОС.

В общем случае формализованные требования к обеспечению защиты компьютерной информации от НСД задаются формализованными требованиями к защите СВТ. Однако, в связи с вышесказанным, будем для оценки эффективности защитных механизмов ОС также использовать формализованные требования, применяемые к автоматизированным системам (АС). Это следует делать, потому что именно защитные механизмы ОС призваны обеспечивать необходимый уровень защиты автоматизированной системы (АС) в целом.

Аналогичные рассуждения могут быть проведены и относительно средства добавочной защиты. При этом, во-первых, оно может рассматриваться как отдельное СВТ, выполняющее формализованные требования, при этом встроенные в ОС механизмы защиты не должны рассматриваться; во-вторых, оно может рассматриваться как средство в составе АС (наряду с механизмами ОС, дополняя их). В этом случае встроенные в ОС и добавочные механизмы защиты должны в совокупности выполнять формализованные требования. Кроме того, для определенных классов защиты встроенные механизмы защиты должны быть сертифицированы.

Рассмотрим формализованные требования к защите компьютерной информации АС. При этом будем рассматривать первую группу АС (в соответствии с используемой классификацией), как включающую в себя наиболее распространенные многопользовательские АС, в которых одновременно обрабатывается и/или хранится информация разных уровней конфиденциальности. Причем не все пользователи имеют право доступа ко всей информации АС.

Первая группа АС содержит пять классов – 1Д, 1Г, 1В, 1Б и 1А. Деление АС на соответствующие классы по условиям их функционирования с точки зрения защиты информации необходимо в целях разработки и применения обоснованных мер по достижению требуемого уровня защиты. Дифференциация подхода к выбору методов и средств защиты определяется важностью обрабатываемой информации, а также различием АС по своему составу, структуре, способам обработки информации, количественному и качественному составу пользователей и обслуживающего персонала. Требования к АС сведены в табл. 8.1.

Как видно, рассматриваемыми требованиями выделяются следующие основные группы механизмов защиты:

– механизмы управления доступом;

– механизмы регистрации и учета;

– механизмы криптографической защиты;

– механизмы контроля целостности.

Отметим, что первая группа "Подсистема управления доступом" является основополагающей для реализации защиты от НСД, так как именно механизмы защиты данной группы призваны непосредственно противодействовать несанкционированному доступу к компьютерной информации.

Остальные же группы механизмов реализуются в предположении, что механизмы защиты первой группы могут быть преодолены злоумышленником. В частности, они могут использоваться:

- для контроля действий пользователя – группа "Подсистема регистрации и учета";
- для противодействия возможности прочтения похищенной информации – группа "Криптографическая подсистема";
- для контроля осуществленных злоумышленником изменений защищаемых объектов (исполняемых файлов и файлов данных) при осуществлении к ним НСД и для восстановления защищаемой информации из резервных копий – группа "Под-

система обеспечения целостности".

Кроме того, эти группы механизмов могут использоваться для проведения расследования по факту НСД.

8.1. Требования к защищенности автоматизированных систем

Подсистемы и требования	КЛАССЫ								
	3Б	3А	2Б	2А	1Д	1Г	1В	1Б	1А
1. Подсистема управления доступом.									
1.1. Идентификация, проверка подлинности и контроль доступа субъектов:									
в систему;	+	+	+	+	+	+	+	+	+
к терминалам, ЭВМ, узлам сети ЭВМ, каналам связи, внешним устройствам ЭВМ;				+		+	+	+	+
к программам;				+		+	+	+	+
к томам, каталогам, файлам, записям, полям записей				+		+	+	+	+
1.2.Управление потоками информации				+			+	+	+
2. Подсистема регистрации и учета.									
2.1. Регистрация и учет:									
входа/выхода субъектов в/из системы (узла сети);			+	+	+	+	+	+	+
выдача печатных (графических) выходных документов;				+		+	+	+	+
запуска/завершения программ и процессов (заданий, задач);				+		+	+	+	+
доступа программ субъектов к защищаемым файлам, включая их создание и удаление, передачу по линиям и каналам связи;				+		+	+	+	+
доступа программ субъектов, доступа к терминалам, ЭВМ, узлам сети ЭВМ, внешним устройствам ЭВМ, программам, томам, каталогам, файлам, записям, полям записей;				+		+	+	+	+
изменения полномочий субъектов доступа;							+	+	+
создаваемых защищаемых объектов доступа				+			+	+	+
2.2.Учет носителей информации			+	+	+	+	+	+	+
2.3. Очистка (обнуление, обезличивание) освобождаемых областей оперативной памяти ЭВМ и внешних накопителей				+		+	+	+	+
2.4. Сигнализация попыток нарушения защиты							+	+	+
3. Криптографическая защита.									
3.1. Шифрование конфиденциальной информации				+				+	+
3.2. Шифрование информации, принадлежащей различным субъектам доступа (группам доступа) на разных ключах									+
3.3. Использование аттестованных (сертифицированных) криптографических средств					+			+	+
4. Подсистема обеспечения целостности.									
4.1. Обеспечение целостности программных средств и обрабатываемой информации			+	+	+	+	+	+	+
4.2. Физическая охрана средств вычислительной охраны и носителей информации			+	+	+	+	+	+	+
4.3. Наличие администратора (службы) защиты информации в АС				+			+	+	+
4.4. Периодическое тестирование СЗИ НСД			+	+	+	+	+	+	+
4.5. Наличие средств восстановления СЗИ НСД			+	+	+	+	+	+	+
4.6. Использование сертифицированных средств защиты				+			+	+	+

Обозначение: "+" – есть требования к данному классу;
СЗИ НСД – система защиты информации от несанкционированного доступа.

Рассмотрим более подробно требования различных групп, а также соответствующие им основные подходы к защите компьютерной информации, реализуемые на сегодняшний день на практике. При этом имеет смысл остановиться лишь на двух классах:

1) 1Г – задающим необходимые (минимальные) требования для обработки конфиденциальной информации;

2) 1В – задающим необходимые (минимальные) требования для обработки информации, являющейся собственностью государства и отнесенной к категории секретной.

8.2.3.1. ТРЕБОВАНИЯ К ЗАЩИТЕ КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ

Подсистема управления доступом должна удовлетворять следующим требованиям:

1. Идентифицировать и проверять подлинность субъектов доступа при входе в систему. Причем это должно осуществ-

ляться по идентификатору (коду) и паролю условно-постоянного действия длиной не менее шести буквенно-цифровых символов.

2. Идентифицировать терминалы, ЭВМ, узлы компьютерной сети, каналы связи, внешние устройства ЭВМ по их логическим адресам (номерам).

3. По именам идентифицировать программы, тома, каталоги, файлы, записи и поля записей.

4. Осуществлять контроль доступа субъектов к защищаемым ресурсам в соответствии с матрицей доступа.

Подсистема регистрации и учета должна:

1. Регистрировать вход (выход) субъектов доступа в систему (из системы), либо регистрировать загрузку и инициализацию операционной системы и ее программного останова. При этом в параметрах регистрации указываются:

- дата и время входа (выхода) субъекта доступа в систему (из системы) или загрузки (останова) системы;
- результат попытки входа – успешная или неуспешная (при НСД);
- идентификатор (код или фамилия) субъекта, предъявленный при попытке доступа;
- код или пароль, предъявленный при неуспешной попытке.

Регистрация выхода из системы или останова не проводится в моменты аппаратурного отключения АС.

2. Регистрировать выдачу печатных (графических) документов на "твердую" копию. При этом в параметрах регистрации указываются:

- дата и время выдачи (обращения к подсистеме вывода);
- краткое содержание документа (наименование, вид, код, шифр) и уровень его конфиденциальности;
- спецификация устройства выдачи (логическое имя (номер) внешнего устройства);
- идентификатор субъекта доступа, запросившего документ.

3. Регистрировать запуск (завершение) программ и процессов (заданий, задач), предназначенных для обработки защищаемых файлов. При этом в параметрах регистрации указывается:

- дата и время запуска;
- имя (идентификатор) программы (процесса, задания);
- идентификатор субъекта доступа, запросившего программу (процесс, задание);
- результат запуска (успешный, неуспешный – несанкционированный).

4. Регистрировать попытки доступа программных средств (программ, процессов, задач, заданий) к защищаемым файлам. В параметрах регистрации указывается:

- дата и время попытки доступа к защищаемому файлу с указанием ее результата (успешная, неуспешная – несанкционированная);
- идентификатор субъекта доступа;
- спецификация защищаемого файла.

5. Регистрировать попытки доступа программных средств к следующим дополнительным защищаемым объектам доступа: терминалам, ЭВМ, узлам сети ЭВМ, линиям (каналам) связи, внешним устройствам ЭВМ, программам, томам, каталогам, файлам, записям, полям записей. При этом в параметрах регистрации указывается:

- дата и время попытки доступа к защищаемому файлу с указанием ее результата: успешная, неуспешная, несанкционированная;
- идентификатор субъекта доступа;
- спецификация защищаемого объекта [логическое имя (номер)].

6. Проводить учет всех защищаемых носителей информации с помощью их маркировки и с занесением учетных данных в журнал (учетную карточку).

7. Регистрировать выдачу (приемку) защищаемых носителей.

8. Осуществлять очистку (обнуление, обезличивание) освобождаемых областей оперативной памяти ЭВМ и внешних накопителей. При этом очистка должна производиться однократной произвольной записью в освобождаемую область памяти, ранее использованную для хранения защищаемых данных (файлов).

Подсистема обеспечения целостности должна:

1. Обеспечивать целостность программных средств системы защиты информации от НСД (СЗИ НСД), обрабатываемой информации, а также неизменность программной среды. При этом:

- целостность СЗИ НСД проверяется при загрузке системы по контрольным суммам компонент СЗИ;
- целостность программной среды обеспечивается использованием трансляторов с языка высокого уровня и отсутствием средств модификации объектного кода программ в процессе обработки и (или) хранения защищаемой информации.

2. Осуществлять физическую охрану СВТ (устройств и носителей информации). При этом должны предусматриваться контроль доступа в помещение АС посторонних лиц, а также наличие надежных препятствий для несанкционированного проникновения в помещение АС и хранилище носителей информации. Особенно в нерабочее время.

3. Проводить периодическое тестирование функций СЗИ НСД при изменении программной среды и персонала АС с помощью тест программ, имитирующих попытки НСД.

4. Иметь в наличии средства восстановления СЗИ НСД. При этом предусматривается ведение двух копий программных средств СЗИ НСД, а также их периодическое обновление и контроль работоспособности.

8.2.3.2. ТРЕБОВАНИЯ К ЗАЩИТЕ СЕКРЕТНОЙ ИНФОРМАЦИИ

Подсистема управления доступом должна:

1. Идентифицировать и проверять подлинность субъектов доступа при входе в систему. Причем это должно осуществляться по идентификатору (коду) и паролю условно-постоянного действия длиной не менее шести буквенно-цифровых символов.

2. Идентифицировать терминалы, ЭВМ, узлы компьютерной сети, каналы связи, внешние устройства ЭВМ по их логическим адресам (номерам).

3. По именам идентифицировать программы, тома, каталоги, файлы, записи и поля записей.

4. Осуществлять контроль доступа субъектов к защищаемым ресурсам в соответствии с матрицей доступа.

5. Управлять потоками информации с помощью меток конфиденциальности. При этом уровень конфиденциальности накопителя должен быть не ниже уровня конфиденциальности записываемой на него информации.

Подсистема регистрации и учета должна:

1. Регистрировать вход (выход) субъектов доступа в систему (из системы) либо регистрировать загрузку и инициализацию операционной системы и ее программного останова. При этом в параметрах регистрации указываются:

- дата и время входа (выхода) субъекта доступа в систему (из системы) или загрузки (останова) системы;
- результат попытки входа – успешная или неуспешная (при НСД);
- идентификатор (код или фамилия) субъекта, предъявленный при попытке доступа;
- код или пароль, предъявленный при неуспешной попытке.

Регистрация выхода из системы или останова не проводится в моменты аппаратурного отключения АС.

2. Регистрировать выдачу печатных (графических) документов на "твердую" копию. Выдача должна сопровождаться автоматической маркировкой каждого листа (страницы) документа порядковым номером и учетными реквизитами АС с указанием на последнем листе документа общего количества страниц. В параметрах регистрации указываются:

- дата и время выдачи (обращения к подсистеме вывода);
- краткое содержание документа (наименование, вид, код, шифр) и уровень его конфиденциальности;
- спецификация устройства выдачи (логическое имя (номер) внешнего устройства);
- идентификатор субъекта доступа, запросившего документ;
- объем фактически выданного документа (количество страниц, листов, копий) и результат выдачи (успешный – весь объем, неуспешный).

3. Регистрировать запуск (завершение) программ и процессов (заданий, задач), предназначенных для обработки защищаемых файлов. В параметрах регистрации указывается:

- дата и время запуска;
- имя (идентификатор) программы (процесса, задания);
- идентификатор субъекта доступа, запросившего программу (процесс, задание);
- результат запуска (успешный, неуспешный – несанкционированный).

4. Регистрировать попытки доступа программных средств (программ, процессов, задач, заданий) к защищаемым файлам. В параметрах регистрации указывается:

- дата и время попытки доступа к защищаемому файлу с указанием ее результата: (успешная, неуспешная – несанкционированная);
- идентификатор субъекта доступа;
- спецификация защищаемого файла;
- имя программы (процесса, задания, задачи), осуществляющих доступ к файлам;
- вид запрашиваемой операции (чтение, запись, удаление, выполнение, расширение и т.п.).

5. Регистрировать попытки доступа программных средств к следующим дополнительным защищаемым объектам доступа: терминалам, ЭВМ, узлам сети ЭВМ, линиям (каналам) связи, внешним устройствам ЭВМ, программам, томам, каталогам, файлам, записям, полям записей. В параметрах регистрации указывается:

- дата и время попытки доступа к защищаемому файлу с указанием ее результата (успешная, неуспешная – несанкционированная);
- идентификатор субъекта доступа;
- спецификация защищаемого объекта [логическое имя (номер)];
- имя программы (процесса, задания, задачи), осуществляющих доступ к файлам;
- вид запрашиваемой операции (чтение, запись, монтирование, захват и т.п.).

6. Регистрировать изменения полномочий субъектов доступа, а также статуса объектов доступа. В параметрах регистрации указывается:

- дата и время изменения полномочий;
- идентификатор субъекта доступа (администратора), осуществившего изменения.

7. Осуществлять автоматический учет создаваемых защищаемых файлов с помощью их дополнительной маркировки, используемой в подсистеме управления доступом. Маркировка должна отражать уровень конфиденциальности объекта.

8. Проводить учет всех защищаемых носителей информации с помощью их маркировки и с занесением учетных данных в журнал (учетную карточку).

9. Проводить учет защищаемых носителей с регистрацией их выдачи (приема) в специальном журнале (картотеке).

10. Проводить несколько видов учета (дублирующих) защищаемых носителей информации.

11. Осуществлять очистку (обнуление, обезличивание) освобождаемых областей оперативной памяти ЭВМ и внешних накопителей. Причем очистка должна осуществляться двукратной произвольной записью в освобождаемую область памяти, ранее использованную для хранения защищаемых данных (файлов).

12. Сигнализировать о попытках нарушения защиты.

Подсистема обеспечения целостности должна:

1. Обеспечивать целостность программных средств СЗИ НСД, обрабатываемой информации, а также неизменность программной среды. При этом:

- целостность СЗИ НСД проверяется при загрузке системы по контрольным суммам компонент СЗИ;

- целостность программной среды обеспечивается использованием трансляторов с языка высокого уровня и отсутствием средств модификации объектного кода программ в процессе обработки и (или) хранения защищаемой информации.

2. Осуществлять физическую охрану СВТ (устройств и носителей информации). При этом должно предусматриваться постоянное наличие охраны на территории здания и помещений, где находится АС. Охрана должна производиться с помощью технических средств охраны и специального персонала, а также с использованием строгого пропускного режима и специального оборудования в помещении АС.

3. Предусматривать наличие администратора или целой службы защиты информации, ответственных за ведение, нормальное функционирование и контроль работы СЗИ НСД. Администратор должен иметь свой терминал и необходимые средства оперативного контроля и воздействия на безопасность АС.

4. Проводить периодическое тестирование функций СЗИ НСД при изменении программной среды и персонала АС с помощью специальных программных средств не реже одного раза в год.

5. Иметь в наличии средства восстановления СЗИ НСД, предусматривающие ведение двух копий программных средств СЗИ НСД и их периодическое обновление и контроль работоспособности.

6. Использовать только сертифицированные средства защиты. Их сертификацию проводят специальные сертификационные центры или специализированные предприятия, имеющие лицензию на проведение сертификации средств защиты СЗИ НСД.

8.2.4. РАЗЛИЧИЯ ТРЕБОВАНИЙ И ОСНОВОПОЛАГАЮЩИХ МЕХАНИЗМОВ ЗАЩИТЫ ОТ НСД

Сравним две рассмотренные группы требований и их особенности для защиты информации различных категорий (конфиденциальной и секретной). Ясно, что ключевыми механизмами защиты, образующими основную группу механизмов защиты от НСД ("Подсистема управления доступом") являются:

1) идентификация и проверка подлинности субъектов доступа при входе в систему по идентификатору (коду) и паролю условно-постоянного действия;

2) контроль доступа субъектов к защищаемым ресурсам в соответствии с матрицей доступа.

Дополнительным требованием и принципиальным отличием при защите секретной информации является то, что механизмом защиты должно осуществляться управление потоками информации с помощью меток конфиденциальности. При этом уровень конфиденциальности накопителя должен быть не ниже уровня конфиденциальности записываемой на него информации.

Все три перечисленных механизма являются основополагающими. Связаны они следующим образом: все права доступа к ресурсам (разграничительная политика доступа к ресурсам) задаются для конкретного субъекта доступа (пользователя). Поэтому субъект доступа (пользователь) должен быть идентифицирован при входе в систему, соответственно, должна быть проконтролирована его подлинность (аутентичность). Реализацию основ защиты компьютерной информации от НСД – разграничительную политику доступа к ресурсам осуществляют механизмы контроля доступа.

Следуя формализованным требованиям к системе защиты информации, основу реализации разграничительной политики доступа к ресурсам при обработке сведений конфиденциального характера является дискреционный механизм управления доступом. При этом реализуется дискреционная модель доступа к ресурсам.

Принципы организации и функционирования этой модели основаны на том, что права доступа задаются матрицей доступа, элементами которой являются разрешенные права доступа субъекта к объекту. Что касается контроля доступа, то он осуществляется непосредственно путем анализа прав доступа к объекту запрашивающего доступ субъекта. При этом анализируется, есть ли в матрице информация о разрешении доступа данного субъекта к данному объекту, или нет.

При защите секретной информации в основе разграничительной политики доступа (РПД) к ресурсам должен лежать помимо дискреционного (дискреционная модель управления доступом), мандатный механизм управления доступом (мандатная модель управления доступом).

В рамках мандатного механизма каждому субъекту (пользователю, приложению и т.д.) и каждому объекту (файлу, каталогу и т.д.) ставятся в соответствие специальные классификационные метки. Посредством этих меток субъектам и объектам назначаются классификационные уровни (уровни уязвимости, категории секретности и т.п.), являющиеся комбинациями иерархических и неиерархических категорий. Сам контроль и управление доступом осуществляется путем сопоставления классификационных меток субъекта и объекта доступа, отражающих их место в соответствующей иерархии. В общих чертах в этом и заключается мандатный механизм управления доступом, т.е. право доступа дается на основе сравнения меток объекта и субъекта. При этом, чтобы субъект получил доступ к объекту, его уровень конфиденциальности должен быть не ниже уровня конфиденциальности объекта.

Требования к практической реализации дискретного и мандатного механизмов управления доступом тесно связаны с требованиями к обеспечению целостности и очистки памяти.

Для системы защиты выполнение требования "Должна быть обеспечена целостность программных средств системы защиты информации от НСД (СЗИ НСД), обрабатываемой информации, а также замкнутость программной среды" обеспечивает возможность противодействия скрытым действиям пользователей, направленных на получение НСД к информации в обход РПД. При этом не важно, чем будет пользоваться потенциальный нарушитель – программами собственной разработки, всевозможными отладочными средствами или иным ПО.

Требование к очистке памяти "Должна осуществляться очистка (обнуление, обезличивание) освобождаемых областей оперативной памяти ЭВМ и внешних накопителей. Очистка осуществляется двукратной произвольной записью в освобождаемую область памяти, ранее использованную для хранения защищаемых данных (файлов)" обусловливает невозможность доступа пользователя к остаточной информации. Его необходимость вызвана тем, что при удалении файла на внешнем накопителе системными средствами осуществляется изменение разметки накопителя, но собственно информация остается. Она так и называется "остаточная информация". При этом ввиду того, что остаточная информация уже не является объектом дос-

тупа (она соответствующим образом не размечена на диске – не является файлом), дискреционный и мандатный механизмы контроля доступа не могут осуществлять РПД к этой информации. А это значит, что доступ к ней может быть осуществлен в обход подсистемы управления доступом.

Контрольные вопросы к теме 8

1. Охарактеризуйте подходы к обеспечению компьютерной безопасности.
2. Перечислите основные и вспомогательные сервисы безопасности, дайте их классификацию.
3. Дайте характеристику групп требований к системе защиты.
4. Перечислите основные требования к защите конфиденциальной информации.
5. Перечислите основные требования к защите секретной информации.
6. Опишите основные различия требований и механизмов защиты от НСД для конфиденциальной и секретной информации.

Т е м а 9. МОДЕЛИ БЕЗОПАСНОСТИ ОСНОВНЫХ ОПЕРАЦИОННЫХ СИСТЕМ

9.1. МЕХАНИЗМЫ ЗАЩИТЫ ОПЕРАЦИОННЫХ СИСТЕМ

Операционная система есть специально организованная совокупность программ, которая управляет ресурсами системы (ЭВМ, вычислительной системы, других компонентов ИВС) с целью наиболее эффективного их использования и обеспечивает интерфейс пользователя с ресурсами.

Операционные системы, подобно аппаратуре ЭВМ, на пути своего развития прошли несколько поколений.

ОС первого поколения были направлены на ускорение и упрощение перехода с одной задачи пользователя на другую задачу (другого пользователя), что поставило проблему обеспечения безопасности данных, принадлежащих разным задачам.

Второе поколение ОС характеризовалось наращиванием программных средств обеспечения операций ввода-вывода и стандартизацией обработки прерываний. Надежное обеспечение безопасности данных в целом осталось нерешенной проблемой.

К концу 60-х гг. XX в. начал осуществляться переход к мультипроцессорной организации средств ВТ, поэтому проблемы распределения ресурсов и их защиты стали более острыми и трудноразрешимыми. Решение этих проблем привело к соответствующей организации ОС и широкому применению аппаратных средств защиты (защита памяти, аппаратный контроль, диагностика и т.п.).

Основной тенденцией развития вычислительной техники была и остается идея максимальной доступности ее для пользователей, что входит в противоречие с требованием обеспечения безопасности данных.

Под ***механизмами защиты ОС*** будем понимать все средства и механизмы защиты данных, функционирующие в составе ОС. Операционные системы, в составе которых функционируют средства и механизмы защиты данных, часто называют защищенными системами.

Под ***безопасностью ОС*** будем понимать такое состояние ОС, при котором невозможно случайное или преднамеренное нарушение функционирования ОС, а также нарушение безопасности находящихся под управлением ОС ресурсов системы. Укажем следующие особенности ОС, которые позволяют выделить вопросы обеспечения безопасности ОС в особую категорию:

- управление всеми ресурсами системы;
- наличие встроенных механизмов, которые прямо или косвенно влияют на безопасность программ и данных, работающих в среде ОС;
- обеспечение интерфейса пользователя с ресурсами системы;
- размеры и сложность ОС.

Большинство ОС обладают дефектами с точки зрения обеспечения безопасности данных в системе, что обусловлено выполнением задачи обеспечения максимальной доступности системы для пользователя.

Рассмотрим ***типовые функциональные дефекты ОС***, которые могут привести к созданию каналов утечки данных.

1. ***Идентификация.*** Каждому ресурсу в системе должно быть присвоено уникальное имя – идентификатор. Во многих системах пользователи не имеют возможности удостовериться в том, что используемые ими ресурсы действительно принадлежат системе.
2. ***Пароли.*** Большинство пользователей выбирают простейшие пароли, которые легко подобрать или угадать.
3. ***Список паролей.*** Хранение списка паролей в незашифрованном виде дает возможность его компрометации с последующим НСД к данным.
4. ***Пороговые значения.*** Для предотвращения попыток несанкционированного входа в систему с помощью подбора пароля необходимо ограничить число таких попыток, что в некоторых ОС не предусмотрено.
5. ***Подразумеваемое доверие.*** Во многих случаях программы ОС считают, что другие программы работают правильно.
6. ***Общая память.*** При использовании общей памяти не всегда после выполнения программ очищаются участки оперативной памяти (ОП).
7. ***Разрыв связи.*** В случае разрыва связи ОС должна немедленно закончить сеанс работы с пользователем или повторно установить подлинность субъекта.
8. ***Передача параметров по ссылке, а не по значению*** (при передаче параметров по ссылке возможно сохранение параметров в ОП после проверки их корректности, нарушитель может изменить эти данные до их использования).
9. ***Система может содержать много элементов*** (например, программ), имеющих различные привилегии.

Основной проблемой обеспечения безопасности ОС является проблема создания механизмов контроля доступа к ресурсам системы. Процедура контроля доступа заключается в проверке соответствия запроса субъекта предоставленным ему правам доступа к ресурсам. Кроме того, ОС содержит вспомогательные средства защиты, такие как средства мониторинга, профилактического контроля и аудита. В совокупности механизмы контроля доступа и вспомогательные средства защиты образуют механизмы управления доступом.

Средства профилактического контроля необходимы для отстранения пользователя от непосредственного выполнения критичных с точки зрения безопасности данных операций и передачи этих операций под контроль ОС. Для обеспечения безопасности данных работа с ресурсами системы осуществляется с помощью специальных программ ОС, доступ к которым ограничен.

Средства мониторинга осуществляют постоянное ведение регистрационного журнала, в который заносятся записи о всех событиях в системе. В ОС могут использоваться средства сигнализации о НСД, которые используются при обнаружении нарушения безопасности данных или попыток нарушения.

Контроль доступа к данным. При создании механизмов контроля доступа необходимо, прежде всего, определить множества субъектов и объектов доступа. Субъектами могут быть, например, пользователи, задания, процессы и процедуры. Объектами – файлы, программы, семафоры, директории, терминалы, каналы связи, устройства, блоки ОП и т.д. Субъекты могут одновременно рассматриваться и как объекты, поэтому у субъекта могут быть права на доступ к другому субъекту. В конкретном процессе в данный момент времени субъекты являются активными элементами, а объекты – пассивными.

Для осуществления доступа к объекту субъект должен обладать соответствующими полномочиями. Полномочие есть некий символ, обладание которым дает субъекту определенные права доступа по отношению к объекту, область защиты определяет права доступа некоторого субъекта ко множеству защищаемых объектов и представляет собой совокупность всех полномочий данного субъекта.

При функционировании системы необходимо иметь возможность создавать новые субъекты и объекты. При создании объекта одновременно создается и полномочие субъектов по использованию этого объекта. Субъект, создавший такое полномочие, может воспользоваться им для осуществления доступа к объекту или же может создать несколько копий полномочия для передачи их другим субъектам.

С традиционной точки зрения средства управления доступом позволяют специфицировать и контролировать действия, которые субъекты (пользователи и процессы) могут выполнять над объектами (информацией и другими компьютерными ресурсами). В данном разделе речь пойдет о логическом управлении доступом, которое, в отличие от физического, реализуется программными средствами. Логическое управление доступом – это основной механизм многопользовательских систем, призванный обеспечить конфиденциальность и целостность объектов и, до некоторой степени, их доступность (путем запрещения обслуживания неавторизованных пользователей).

Рассмотрим формальную постановку задачи в традиционной трактовке. Имеется совокупность субъектов и набор объектов. Задача логического управления доступом состоит в том, чтобы для каждой пары "субъект-объект" определить множество допустимых операций и контролировать выполнение установленного порядка.

Отношение "субъекты-объекты" можно представить в виде матрицы доступа, в строках которой перечислены субъекты, в столбцах – объекты, а в клетках, расположенных на пересечении строк и столбцов, записаны дополнительные условия (например, время и место действия) и разрешенные виды доступа. Фрагмент матрицы может выглядеть, например, как показано в табл. 9.1.

9.1. Фрагмент матрицы доступа

	Файл	Программа	Линия связи	Реляционная таблица
Пользователь 1	o r w с системной консоли	e	rw с 8:00 до 18:00	
Пользователь 2				a

О б о з н а ч е н и е : "o" – разрешение на передачу прав доступа другим пользователям, "r" – чтение, "w" – запись, "e" – выполнение, "a" – добавление информации.

Тема логического управления доступом – одна из сложнейших в области информационной безопасности. Дело в том, что само понятие объекта (а тем более видов доступа) меняется от сервиса к сервису. Для операционной системы к объектам относятся файлы, устройства и процессы. Применительно к файлам и устройствам обычно рассматриваются права на чтение, запись, выполнение (для программных файлов), иногда на удаление и добавление. Отдельным правом может быть возможность передачи полномочий доступа другим субъектам (так называемое право владения). Процессы можно создавать и уничтожать. Современные операционные системы могут поддерживать и другие объекты.

Для систем управления реляционными базами данных объект – это база данных, таблица, представление, хранимая процедура. К таблицам применимы операции поиска, добавления, модификации и удаления данных, у других объектов. В результате при задании матрицы доступа нужно принимать во внимание не только принцип распределения привилегий для каждого сервиса, но и существующие связи между сервисами (приходится заботиться о согласованности разных частей матрицы). Аналогичная трудность возникает при экспорте/импорте данных, когда информация о правах доступа, как правило, теряется (поскольку на новом сервисе она не имеет смысла). Следовательно, обмен данными между различными сервисами представляет особую опасность с точки зрения управления доступом, а при проектировании и реализации разнородной кон-

фигурации необходимо позаботиться о согласованном распределении прав доступа субъектов к объектам и о минимизации числа способов экспорта/импорта данных.

Матрицу доступа, ввиду ее разреженности (большинство клеток – пустые), неразумно хранить в виде двухмерного массива. Обычно ее хранят по столбцам, т.е. для каждого объекта поддерживается список "допущенных" субъектов вместе с их правами. Элементами списков могут быть имена групп и шаблоны субъектов, что служит большим подспорьем администратору. Некоторые проблемы возникают только при удалении субъекта, когда приходится удалять его имя из всех списков доступа; впрочем, эта операция производится нечасто.

Списки доступа – исключительно гибкое средство. С их помощью легко выполнить требование о гранулярности прав с точностью до пользователя. Посредством списков несложно добавить права или явным образом запретить доступ (например, чтобы наказать нескольких членов группы пользователей). Безусловно, списки являются лучшим средством произвольного управления доступом.

Подавляющее большинство операционных систем и систем управления базами данных реализуют именно произвольное управление доступом. Основное достоинство произвольного управления – гибкость. К сожалению, у "произвольного" подхода есть ряд недостатков. Рассредоточенность управления доступом ведет к тому, что доверенными должны быть многие пользователи, а не только системные операторы или администраторы. Из-за рассеянности или некомпетентности сотрудника, владеющего секретной информацией, эту информацию могут узнать и все остальные пользователи. Следовательно, произвольность управления должна быть дополнена жестким контролем за реализацией избранной политики безопасности.

Второй недостаток, который представляется основным, состоит в том, что права доступа существуют отдельно от данных. Ничто не мешает пользователю, имеющему доступ к секретной информации, записать ее в доступный всем файл или заменить полезную утилиту ее "троянским" аналогом. Подобная "разделенность" прав и данных существенно осложняет проведение несколькими системами согласованной политики безопасности и, главное, делает практически невозможным эффективный контроль согласованности.

Возвращаясь к вопросу представления матрицы доступа, укажем, что для этого можно использовать также функциональный способ, когда матрицу не хранят в явном виде, а каждый раз вычисляют содержимое соответствующих клеток. Например, при принудительном управлении доступом применяется сравнение меток безопасности субъекта и объекта.

Удобной надстройкой над средствами логического управления доступом является *ограничивающий интерфейс*, когда пользователя лишают самой возможности попытаться совершить несанкционированные действия, включив в число видимых ему объектов только те, к которым он имеет доступ. Подобный подход обычно реализуют в рамках системы меню (пользователю показывают лишь допустимые варианты выбора) или посредством ограничивающих оболочек, таких как restricted shell в ОС Unix.

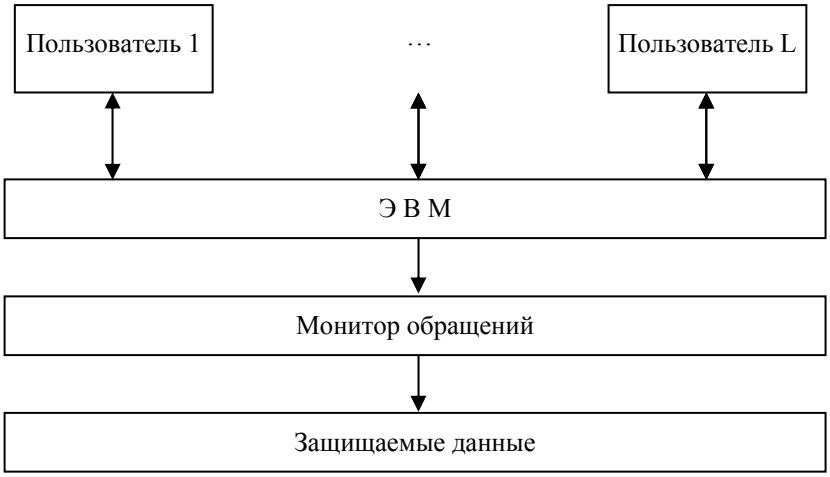

Рис. 9.1. Схема модели Харрисона, Руззо и Ульмана

При принятии решения о предоставлении доступа обычно анализируется следующая информация:

1) идентификатор субъекта (идентификатор пользователя, сетевой адрес компьютера и т.п.). Подобные идентификаторы являются основой произвольного (или дискреционного) управления доступом;

2) атрибуты субъекта (метка безопасности, группа пользователя и т.п.). Метки безопасности – основа мандатного управления доступом.

Непосредственное управление правами доступа осуществляется на основе одной из моделей доступа:

- матричной модели доступа (модель Харрисона-Руззо-Ульмана);
- многоуровневой модели доступа (модель Белла-Лападулы).

Разработка и практическая реализация различных защищенных ОС привела Харрисона, Руззо и Ульмана к построению формальной модели защищенных систем. Схема модели Харрисона, Руззо и Ульмана (HRU-модели) приведена на рис. 9.1.

9.2. АНАЛИЗ ЗАЩИЩЕННОСТИ СОВРЕМЕННЫХ ОПЕРАЦИОННЫХ СИСТЕМ

9.2.1. АНАЛИЗ ВЫПОЛНЕНИЯ СОВРЕМЕННЫМИ ОС ФОРМАЛИЗОВАННЫХ ТРЕБОВАНИЙ К ЗАЩИТЕ ИНФОРМАЦИИ ОТ НСД

Анализировать выполнение современными универсальными ОС требований, задаваемых для класса защищенности АС 1В, не имеет смысла в принципе. Для большинства ОС либо полностью не реализуется основной для данных приложений мандатный механизм управления доступом к ресурсам, либо не выполняется его важнейшее требование "Должно осуществляться управление потоками информации с помощью меток конфиденциальности. При этом уровень конфиденциальности накопителя должен быть не ниже уровня конфиденциальности записываемой на него информации". В связи с этим далее будем говорить лишь о возможном соответствии средств защиты современных ОС классу АС 1Г (защита конфиденциальной информации).

В качестве альтернативных реализаций ОС рассмотрим семейства Unix и Windows (естественно, Windows NT/2000, так как о встроенных механизмах защиты ОС Windows 9x/Me говорить вообще не приходится).

Сначала остановимся на принципиальном, даже, можно сказать, концептуальном противоречии между реализованными в ОС механизмами защиты и принятыми формализованными требованиями. Концептуальном в том смысле, что это противоречие характеризует не какой-либо один механизм защиты, а общий подход к построению системы защиты.

Противоречие состоит в принципиальном различии подходов (соответственно требований) к построению схемы администрирования механизмов защиты и, как следствие, это коренным образом сказывается на формировании общих принципов задания и реализации политики безопасности в организации, распределения ответственности за защиту информации, а также на определении того, кого относить к потенциальным злоумышленникам (от кого защищать информацию).

Для иллюстрации из совокупности формализованных требований к системе защиты конфиденциальной информации рассмотрим следующие два требования:

1) право изменять правила разграничения доступа (ПРД) должно предоставляться выделенным субъектам (администрации, службе безопасности и т.д.);
2) должны быть предусмотрены средства управления, ограничивающие распространения прав на доступ.

Данные требования жестко регламентируют схему (или модель) администрирования механизмов защиты. Это должна быть централизованная схема, единственным элементом которой выступает выделенный субъект, в частности, администратор (администратор безопасности). При этом конечный пользователь исключен в принципе из схемы администрирования механизмов защиты.

При реализации концепции построения системы защиты, регламентируемой рассматриваемыми требованиями, пользователь не наделяется элементом доверия, так как он может считаться потенциальным злоумышленником, что и имеет место на практике.

Теперь в общих чертах рассмотрим концепцию, реализуемую в современных универсальных ОС. Здесь "владельцем" файлового объекта, т.е. лицом, получающим право на задание атрибутов (или ПРД) доступа к файловому объекту, является лицо, создающее файловый объект. Так как файловые объекты создают конечные пользователи, то именно они и назначают ПРД к создаваемым им файловым объектам. Другими словами, в ОС реализуется распределенная схема назначения ПРД, где элементами схемы администрирования являются собственно конечные пользователи.

В данной схеме пользователь должен наделяться практически таким же доверием, как и администратор безопасности, при этом нести наряду с ним ответственность за обеспечение компьютерной безопасности. Отметим, что данная концепция реализуется и большинством современных приложений, в частности СУБД, где пользователь может распространять свои права на доступ к защищаемым ресурсам. Кроме того, не имея в полном объеме механизмов защиты компьютерной информации от конечного пользователя, в рамках данной концепции невозможно рассматривать пользователя в качестве потенциального злоумышленника. А как мы увидим далее, именно с несанкционированными действиями пользователя на защищаемом компьютере (причем как сознательными, так и нет) связана большая часть угроз компьютерной безопасности.

Отметим, что централизованная и распределенная схемы администрирования – это две диаметрально противоположные точки зрения на защиту, требующие совершенно различных подходов к построению моделей и механизмов защиты. При этом сколько-нибудь гарантированную защиту информации можно реализовать только при принятии концепции полностью централизованной схемы администрирования, что подтверждается известными угрозами ОС.

Возможности моделей, методов и средств защиты будем рассматривать применительно к реализации именно концепции централизованного администрирования. Одним из элементов данной концепции является рассмотрение пользователя в качестве потенциального злоумышленника, способного осуществить НСД к защищаемой информации.

9.2.2. ОСНОВНЫЕ ВСТРОЕННЫЕ МЕХАНИЗМЫ ЗАЩИТЫ ОС И ИХ НЕДОСТАТКИ

Кратко остановимся на основных механизмах защиты, встроенных в современные универсальные ОС. Сделаем это применительно к возможности реализации ими принятой нами для рассмотрения концепции защиты конфиденциальной информации.

9.2.2.1. ОСНОВНЫЕ ЗАЩИТНЫЕ МЕХАНИЗМЫ ОС СЕМЕЙСТВА UNIX

Защита ОС семейства Unix в общем случае базируется на трех основных механизмах:

1) идентификации и аутентификация пользователя при входе в систему;
2) разграничении прав доступа к файловой системе, в основе которого лежит реализация дискреционной модели доступа;

3) аудит, т.е. регистрация событий.

При этом отметим, что для различных клонов ОС семейства Unix возможности механизмов защиты могут незначительно различаться, однако будем рассматривать ОС Unix в общем случае, без учета некоторых незначительных особенностей отдельных ОС этого семейства.

Построение файловой системы и разграничение доступа к файловым объектам имеет особенности, присущие данному семейству ОС. Рассмотрим кратко эти особенности. Все дисковые накопители (тома) объединяются в единую ***виртуальную файловую систему*** путем операции монтирования тома. При этом содержимое тома проецируется на выбранный каталог файловой системы. Элементами файловой системы являются также все устройства, подключаемые к защищаемому компьютеру (монтируемые к файловой системе). Поэтому разграничение доступа к ним осуществляется через файловою систему.

Каждый файловый объект имеет индексный дескриптор, в котором среди прочего хранится информация о разграничении доступа к данному файловому объекту. Права доступа делятся на три категории: доступ для владельца, доступ для группы и доступ для остальных пользователей. В каждой категории определяются права на чтение, запись и исполнение (в случае каталога – просмотр).

Пользователь имеет уникальные символьный идентификатор (имя) и числовой идентификатор (UID). Символьный идентификатор предъявляется пользователем при входе в систему, числовой используется операционной системой для определения прав пользователя в системе (доступ к файлам и т.д.).

Принципиальные недостатки защитных механизмов ОС семейства Unix. Рассмотрим в общем случае недостатки реализации системы защиты ОС семейства Unix в части невыполнения требований к защите конфиденциальной информации, напрямую связанные с возможностью НСД к информации.

Для начала отметим, что в ОС семейства Unix, вследствие реализуемой ею концепции администрирования (не централизованная), невозможно обеспечить замкнутость (или целостность) программной среды. Это связано с невозможностью установки атрибута "исполнение" на каталог (для каталога данный атрибут ограничивает возможность "обзора" содержимого каталога). Поэтому при разграничении администратором доступа пользователей к каталогам, пользователь, как "владелец" создаваемого им файла, может занести в свой каталог исполняемый файл и, как его "владелец", установить на файл атрибут "исполнение", после чего запустить записанную им программу. Эта проблема непосредственно связана с реализуемой в ОС концепцией защиты информации.

Не в полном объеме реализуется дискреционная модель доступа, в частности, не могут разграничиваться права доступа для пользователя "root" (UID = 0), т.е. данный субъект доступа исключается из схемы управления доступом к ресурсам. Соответственно все запускаемые им процессы имеют неограниченный доступ к защищаемым ресурсам. С этим недостатком системы защиты связано множество атак, в частности:

- несанкционированное получение прав root;
- запуск с правами root собственного исполняемого файла (локально либо удаленно внедренного), при этом несанкционированная программа получает полный доступ к защищаемым ресурсам и т.д.

Кроме того, в ОС семейства Unix невозможно встроенными средствами гарантированно удалять остаточную информацию. Для этого в системе абсолютно отсутствуют соответствующие механизмы.

Необходимо также отметить, что большинство ОС данного семейства не обладают возможностью контроля целостности файловой системы, т.е. не содержат соответствующих встроенных средств. В лучшем случае дополнительными утилитами может быть реализован контроль конфигурационных файлов ОС по расписанию в то время, как важнейшей возможностью данного механизма можно считать контроль целостности программ (приложений) перед их запуском, контроль файлов данных пользователя и т.д.

Что касается регистрации (аудита), то в ОС семейства Unix не обеспечивается регистрация выдачи документов на "твердую копию", а также некоторые другие требования к регистрации событий.

Если же трактовать требования к управлению доступом в общем случае, то при защите компьютера в составе ЛВС необходимо управление доступом к узлам сети. Однако встроенными средствами зашиты некоторых ОС семейства Unix управление доступом к узлам не реализуется.

Из приведенного анализа видно, что многие механизмы, необходимые с точки зрения выполнения формализованных требований, большинством ОС семейства Unix не реализуется в принципе, либо реализуется лишь частично.

9.2.2.2. ОСНОВНЫЕ ЗАЩИТНЫЕ МЕХАНИЗМЫ ОС СЕМЕЙСТВА WINDOWS (NT/2000/XP)

Теперь кратко остановимся на основных механизмах защиты, реализованных в ОС семейства Windows, и проведем анализ защищенности ОС семейства Windows (NT/2000). Отметим, что здесь ряд объектов доступа (в частности, устройства, реестр ОС и т.д.) не являются объектами файловой системы. Поэтому возникает вопрос, как следует трактовать требование "Система защиты должна контролировать доступ наименованных субъектов (пользователей) к наименованным объектам (файлам, программам, томам и т.д.)". Не ясно, являются ли объектами доступа, к которым, следуя формальным требованиям, необходимо разграничивать доступ пользователей, например, реестр ОС и т.д.

В отличие от семейства ОС Unix, где все задачи разграничительной политики доступа к ресурсам решаются средствами управления доступом к объектам файловой системы, доступ в данных ОС разграничивается собственным механизмом для каждого ресурса. Другими словами, при рассмотрении механизмов защиты ОС Windows встает задача определения и задания требований к полноте разграничений (это определяется тем, что считать объектом доступа).

Также, как и для семейства ОС Unix, здесь основными механизмами защиты являются:

1) идентификация и аутентификация пользователя при входе в систему;

2) разграничение прав доступа к ресурсам, в основе которого лежит реализация дискреционной модели доступа (отдельно к объектам файловой системы, к устройствам, к реестру ОС, к принтерам и др.);

3) аудит, т.е. регистрация событий.

Здесь явно выделяются (в лучшую сторону) возможности разграничений прав доступа к файловым объектам (для NTFS) – существенно расширены атрибуты доступа, устанавливаемые на различные иерархические объекты файловой системы (логические диски, каталоги, файлы). В частности, атрибут "исполнение" может устанавливаться и на каталог, тогда он наследуется соответствующими файлами.

При этом существенно ограничены возможности управления доступом к другим защищаемым ресурсам, в частности, к устройствам ввода. Например, здесь отсутствует атрибут "исполнение", т.е. невозможно запретить запуск несанкционированной программы с устройств ввода.

***Принципиальные недостатки защитных механизмов ОС семейства Windows* (*NT/2000/XP*).** Прежде всего рассмотрим принципиальные недостатки защиты ОС семейства Windows, напрямую связанные с возможностью НСД к информации. При этом в отличие от ОС семейства Unix в ОС Windows невозможна в общем случае реализация централизованной схемы администрирования механизмов защиты или соответствующих формализованных требований. Вспомним, что в ОС Unix это распространялось лишь на запуск процессов. Связано это с тем, что в ОС Windows принята иная концепция реализации разграничительной политики доступа к ресурсам (для NTFS).

В рамках этой концепции разграничения для файла приоритетнее, чем для каталога, а в общем случае – разграничения для включаемого файлового объекта приоритетнее, чем для включающего. Это приводит к тому, что пользователь, создавая файл и являясь его "владельцем", может назначить любые атрибуты доступа к такому файлу (т.е. разрешить к нему доступ любому иному пользователю). Обратиться к этому файлу может пользователь (которому назначил права доступа "владелец") вне зависимости от установленных администратором атрибутов доступа на каталог, в котором пользователь создает файл. Данная проблема непосредственно связана с реализуемой в ОС Windows концепцией защиты информации.

Далее, в ОС семейства Windows (NT/2000/XP) не в полном объеме реализуется дискреционная модель доступа, в частности, не могут разграничиваться права доступа для пользователя "Система". В ОС присутствуют не только пользовательские, но и системные процессы, которые запускаются непосредственно системой. При этом доступ системных процессов не может быть разграничен. Соответственно, все запускаемые системные процессы имеют неограниченный доступ к защищаемым ресурсам. С этим недостатком системы защиты связано множество атак, в частности, несанкционированный запуск собственного процесса с правами системного. Кстати, это возможно и вследствие некорректной реализации механизма обеспечения замкнутости программной среды.

В ОС семейства Windows (NT/2000/XP) невозможно в общем случае обеспечить замкнутость (или целостность) программной среды. Это связано совершено с иными проблемами, чем в ОС семейства Unix, в которых невозможно установить атрибут "исполнение" на каталог. Для выяснения сложности данного вопроса рассмотрим два способа, которыми в общем случае можно реализовать данный механизм, причем оба способа несостоятельны. Итак, механизм замкнутости программной среды в общем случае может быть обеспечен:

– заданием списка разрешенных к запуску процессов с предоставлением возможности пользователям запускать процессы только из этого списка. При этом процессы задаются полнопутевыми именами, причем средствами разграничения доступа обеспечивается невозможность их модернизации пользователем. Данный подход просто не реализуется встроенными в ОС механизмами;

– разрешением запуска пользователями программ только из заданных каталогов при невозможности модернизации этих каталогов. Одним из условий корректной реализации данного подхода является запрет пользователям запуска программ иначе, чем из соответствующих каталогов. Некорректность реализации ОС Windows данного подхода связана с невозможностью установки атрибута "исполнение" на устройства ввода (дисковод или CD-ROM). В связи с этим при разграничении доступа пользователь может запустить несанкционированную программу с дискеты, либо с диска CD-ROM (очень распространенная атака на ОС данного семейства).

Здесь же стоит отметить, что с точки зрения обеспечения замкнутости программной среды [т.е. реализации механизма, обеспечивающего возможность пользователям запускать только санкционированные процессы (программы)] действия пользователя по запуску процесса могут быть как явными, так и скрытыми.

Явные действия предполагают запуск процессов (исполняемых файлов), которые однозначно идентифицируются своим именем. Скрытые действия позволяют осуществлять встроенные в приложения интерпретаторы команд. Примером таковых могут служить офисные приложения. При этом скрытыми действиями пользователя будет запуск макроса.

В данном случае идентификации подлежит лишь собственно приложение, например, процесс winword.exe. При этом он может помимо своих регламентированных действий выполнять те скрытые действия, которые задаются макросом (соответственно, те, которые допускаются интерпретатором), хранящимся в открываемом документе. То же относится и к любой виртуальной машине, содержащей встроенный интерпретатор команд. При этом отметим, что при использовании приложений, имеющих встроенные интерпретаторы команд (в том числе офисных приложений), не в полном объеме обеспечивается выполнение требования по идентификации программ.

Возвращаясь к обсуждению недостатков, отметим, что в ОС семейства Windows (NT/2000/XP) невозможно встроенными средствами гарантированно удалять остаточную информацию. В системе просто отсутствуют соответствующие механизмы.

Кроме того, ОС семейства Windows (NT/2000/XP) не обладают в полном объеме возможностью контроля целостности файловой системы. Встроенные механизмы системы позволяют контролировать только собственные системные файлы, не обеспечивая контроль целостности файлов пользователя. Кроме того, они не решают важнейшую задачу данных механизмов – контроль целостности программ (приложений) перед их запуском, контроль файлов данных пользователя и др.

Что касается регистрации (аудита), то в ОС семейства Windows (NT/2000/XP) не обеспечивается регистрация выдачи документов на "твердую копию", а также некоторые другие требования к регистрации событий.

Опять же, если трактовать требования к управлению доступом в общем случае, то при защите компьютера в составе ЛВС необходимо управление доступом к узлам сети (распределенный пакетный фильтр). В ОС семейства Windows (NT/2000/XP) механизм управления доступа к узлам в полном объеме не реализуется.

Что касается разделяемых сетевых ресурсов, то фильтрации подвергается только входящий доступ к разделяемому ресурсу, а запрос доступа на компьютере, с которого он осуществляется, фильтрации не подлежит. Это принципиально, так как не могут подлежать фильтрации приложения, которыми пользователь осуществляет доступ к разделяемым ресурсам. Благодаря этому, очень распространенными являются атаки на протокол NETBIOS.

Кроме того, в полном объеме управлять доступом к разделяемым ресурсам возможно только при установленной на всех компьютерах ЛВС файловой системы NTFS. В противном случае невозможно запретить запуск несанкционированной программы с удаленного компьютера, т.е. обеспечить замкнутость программной среды в этой части.

Из приведенного анализа можно видеть, что многие механизмы, необходимые с точки зрения выполнения формализованных требований, ОС семейства Windows не реализуют в принципе, либо реализуют лишь частично.

С учетом сказанного можем сделать важный вывод относительно того, что большинством современных универсальных ОС не выполняются в полном объеме требования к защите АС по классу 1Г. Это значит, что, учитывая требования нормативных документов, они не могут без использования добавочных средств защиты применяться для защиты даже конфиденциальной информации. При этом следует отметить, что основные проблемы защиты здесь вызваны не невыполнимостью ОС требований к отдельным механизмам защиты, а принципиальными причинами, обусловленными реализуемой в ОС концепцией защиты. Концепция эта основана на реализации распределенной схемы администрирования механизмов защиты, что само по себе является невыполнением формализованных требований к основным механизмам защиты.

9.2.3. АНАЛИЗ СУЩЕСТВУЮЩЕЙ СТАТИСТИКИ УГРОЗ ДЛЯ СОВРЕМЕННЫХ УНИВЕРСАЛЬНЫХ ОС. СЕМЕЙСТВА ОС И ОБЩАЯ СТАТИСТИКА УГРОЗ

На сегодняшний день существует достаточно большая статистика угроз ОС, направленных на преодоление встроенных в ОС механизмов защиты, позволяющих изменить настройки механизмов безопасности, обойти разграничения доступа и т.д. Таким образом, статистика фактов НСД к информации показывает, что большинство распространенных систем (универсального назначения) довольно уязвимы с точки зрения безопасности. И это несмотря на отчетливую тенденцию к повышению уровня защищенности этих систем.

Здесь необходимо отметить, что на практике современные информационные системы, предназначенные для обработки конфиденциальной информации, строятся уже с учетом дополнительных мер безопасности, что также косвенно подтверждает изначальную уязвимость современных ОС.

Рассмотрим операционные системы, фигурирующие в публикуемых списках системных и прикладных ошибок, позволяющих получить несанкционированный доступ к системе, понизить степень ее защищенности или добиться отказа в обслуживании (системного сбоя):

MS Windows 9X	BSD	AIX	BSD	Novell Netware
MS Windows NT	Solaris	SCO	HPUX	IOS (Cisco)
MS Windows 2000	Sun OS	Linux	IRIX	Digital Unix

Общее количество известных успешных атак для различных ОС, представлено в табл. 9.1, а их процентное соотношение – на диаграмме рис. 9.1.

Вследствие того, что большинство атак для операционных систем, построенных на базе Unix (BSD или AT&T), достаточно похожи, целесообразно объединить их в одну группу. Тоже самое можно сказать и об ОС семейства Windows. Таким образом, далее будем рассматривать только семейства ОС: Unix, MS Windows, Novell NetWare.

9.1. Количество известных успешных атак для различных ОС

Тип ОС	Количество атак	Тип ОС	Количество атак
MS Windows NT/2000	130	Linux	167
MS Windows 9X/ME	120	IRIX	84
BSD	64	HPUX	65
BSDI	10	AIX	42
Solaris	125	SCO	40
Sun OS	40	Novell NetWare	10
Digital Unix	25	IOS (Cisco)	7

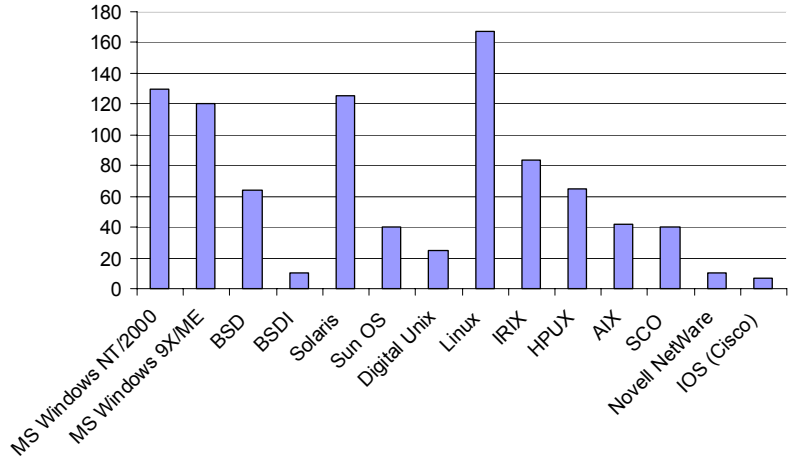

Рис. 9.1. Статистика соотношения угроз для различных ОС

Общее количество известных успешных атак для различных групп ОС представлено в табл. 9.2, а их процентное соотношение – на диаграмме рис. 9.2.

9.2. Общее количество успешных атак для различных групп ОС

Тип ОС	Количество атак
MS Windows	230
Unix	660
Novell Netware	10

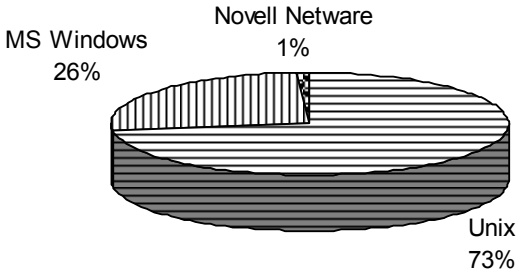

Рис. 9.2. Статистика соотношения угроз для семейств ОС

Относительно ОС Novell следует заметить, что данная ОС изначально создавалась как защищенная (не универсального назначения) ОС, основной функцией которой был защищенный файловый сервис. Это, с одной стороны, должно было обеспечить ее более высокий уровень защищенности, с другой стороны, налагало определенные ограничения по использованию. Однако, начиная с пятой версии, данная ОС начала приобретать свойства универсальности (с точки зрения применяемых протоколов и приложений), что в какой-то мере сказалось и на уровне ее защищенности.

9.2.4. ОБЗОР И СТАТИСТИКА МЕТОДОВ, ЛЕЖАЩИХ В ОСНОВЕ АТАК НА СОВРЕМЕННЫЕ ОС. КЛАССИФИКАЦИЯ МЕТОДОВ И ИХ СРАВНИТЕЛЬНАЯ СТАТИСТИКА

Анализируя рассматриваемые атаки, все методы, позволяющие несанкционированно вмешаться в работу системы, можно разделить на следующие группы:

1) позволяющие несанкционированно запустить исполняемый код;
2) позволяющие осуществить несанкционированные операции чтения/записи файловых или других объектов;
3) позволяющие обойти установленные разграничения прав доступа;
4) приводящие к отказу (Denial of Service) в обслуживании (системный сбой);
5) использующие встроенные недокументированные возможности (ошибки и закладки);
6) использующие недостатки системы хранения или выбора (недостаточная длина) данных об аутентификации (пароли) и позволяющие путем реверсирования, подбора или полного перебора всех вариантов получить эти данные;
7) троянские программы;
8) прочие.

Диаграмма, представляющая собой соотношение групп атак (для представленной выше их классификации) для ОС семейства Windows, представлена на рис. 9.3, для ОС семейства Unix – на рис. 9.4.

Из приведенного анализа можно сделать следующий важный вывод: угрозы, описанные в большинстве групп, напрямую используют различные недостатки ОС и системных приложений и позволяют при полностью сконфигурированных и работающих встроенных в ОС механизмах защиты осуществлять НСД, что подтверждает необходимость усиления встроенных механизмов защиты.

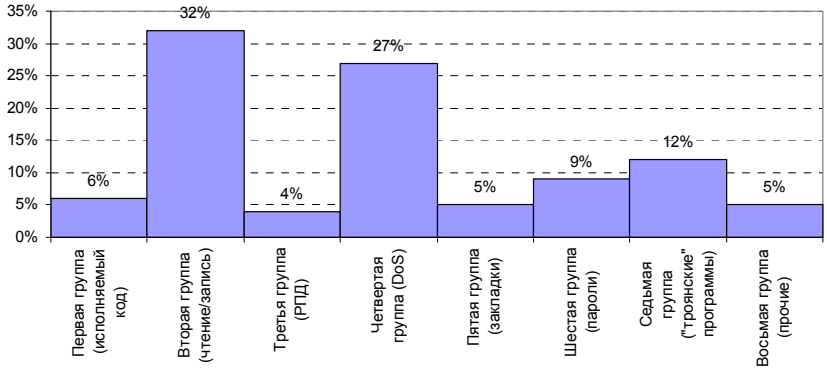

Рис. 9.3. Соотношение групп атак для ОС семейства Windows

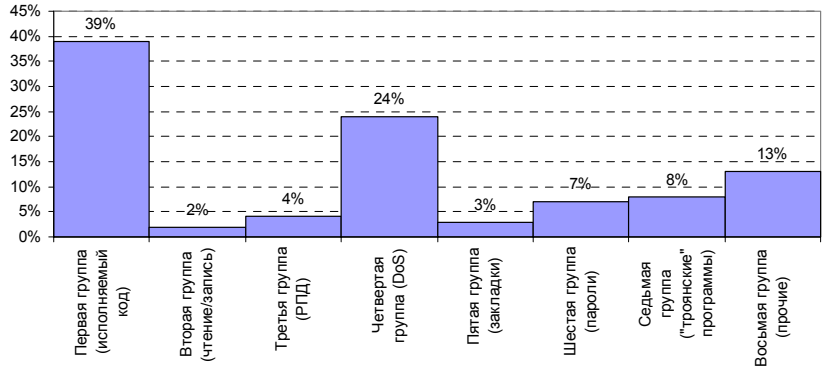

Рис. 9.4. Соотношение групп атак для ОС семейства UNIX

Анализируя представленную статистику угроз, можно сделать вывод, что большая их часть связана именно с недостатками средств защиты ОС, отмеченными выше, т.е. недостатками, связанными с невыполнением (полным, либо частичным) формализованных требований к защите, среди которых, в первую очередь, могут быть выделены:

1) некорректная реализация механизма управления доступом, прежде всего, при разграничении доступа к защищаемым объектам системных процессов и пользователей, имеющих права администратора;

2) отсутствие обеспечения замкнутости (целостности) программной среды.

Как видно, большинство атак осуществлялось либо с использованием некоторых прикладных программ, либо с применением встроенных в виртуальные машины средств программирования, т.е. возможность большинства атак напрямую связана с возможностью запуска злоумышленником соответствующей программы. При этом запуск может быть осуществлен как явно, так и скрыто, в рамках возможностей встроенных в приложения интерпретаторов команд.

Проведенный анализ известных угроз современным универсальным ОС полностью подтверждает, что большая их часть обусловлена именно реализуемым в ОС концептуальным подходом, состоящим в реализации схемы распределенного администрирования механизмов защиты. В рамках этой схемы пользователь рассматривается как доверенное лицо, являющееся элементом схемы администрирования и имеющее возможность назначать/изменять ПРД. При этом он не воспринимается как потенциальный злоумышленник, который может сознательно или несознательно осуществить НСД к информации, следовательно назначение механизмов добавочной защиты ОС состоит в реализации централизованной схемы администрирования механизмов защиты, в рамках которой будет осуществляться противодействие НСД пользователя к информации.

9.3. СИСТЕМА БЕЗОПАСНОСТИ ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS NT

Операционная система Windows NT всегда обладала прекрасными и широко применимыми на практике возможностями защиты. Однократная регистрация в домене Windows NT предоставляет пользователям доступ к ресурсам всей корпоративной сети.

Полноценный набор инструментов Windows NT Server облегчает администраторам управление системой защиты и ее поддержку. Например, администратор может контролировать круг пользователей, имеющих права доступа к сетевым ресурсам: файлам, каталогам, серверам, принтерам и приложениям. Учетными записями пользователей и правами для каждого ресурса можно управлять централизованно.

С помощью простых графических инструментов администратор задает принадлежность к группам, допустимое время работы, срок действия и другие параметры учетной записи. Администратор получает возможность аудита всех событий, связанных с защитой доступа пользователей к файлам, каталогам, принтерам и иным ресурсам. Система также способна блокировать учетную запись пользователя, если число неудачных попыток регистрации превышает заранее определенное. Адми-

нистраторы вправе устанавливать срок действия паролей, принуждать пользователей к периодической смене паролей и выбору паролей, затрудняющих несанкционированный доступ.

С точки зрения пользователя система защиты Windows NT Server полноценна и несложна в обращении. Простая процедура регистрации обеспечивает доступ к соответствующим ресурсам. Для пользователя невидимы такие процессы, как шифрование пароля на системном уровне. Пользователь сам определяет права доступа к тем ресурсам, которыми владеет. Например, чтобы разрешить совместное использование своего документа, он указывает, кто и как может с ним работать. Разумеется, доступ к ресурсам предприятия контролируется только администраторами с соответствующими полномочиями.

Более глубокий уровень безопасности – то, как Windows NT Server защищает данные, находящиеся в физической памяти компьютера. Доступ к ним предоставляется только имеющим на это право программам. Если данные больше не содержатся на диске, система предотвращает несанкционированный доступ к той области диска, где они содержались. При такой системе защиты никакая программа не "подсмотрит" в виртуальной памяти машины информацию, с которой оперирует в данный момент другое приложение.

Удаленный доступ через открытые сети и связь предприятий через Интернет стимулируют постоянное и быстрое развитие технологий безопасности. В качестве примера можно выделить сертификаты открытых ключей и динамические пароли. Архитектура безопасности Windows NT однозначно оценивается как превосходящая и эти, и многие будущие технологии. Перечислим функции безопасности Windows NT:

1. Информация о доменных правилах безопасности и учетная информация хранятся в каталоге Active Directory (служба каталогов Active Directory обеспечивает тиражирование и доступность учетной информации на многих контроллерах домена, а также позволяет удаленное администрирование).
2. В Active Directory поддерживается иерархичное пространство имен пользователей, групп и учетных записей машин (учетные записи могут быть сгруппированы по организационным единицам).
3. Административные права на создание и управление группами учетных записей пользователей могут быть делегированы на уровень организационных единиц (возможно установление дифференцированных прав доступа к отдельным свойствам пользовательских объектов).
4. Тиражирование Active Directory позволяет изменять учетную информацию на любом контроллере домена, а не только на первичном (копии Active Directory, хранящиеся на других контроллерах домена, обновляются и синхронизируются автоматически).
5. Доменная модель изменена и использует Active Directory для поддержки многоуровневого дерева доменов (управление доверительными отношениями между доменами упрощено в пределах всего дерева доменов).
6. В систему безопасности включены новые механизмы аутентификации, такие как Kerberos v5 и TLS (Transport Layer Security), базирующиеся на стандартах безопасности Интернета.
7. Протоколы защищенных каналов (SSL 3.0/TLS) обеспечивают поддержку надежной аутентификации клиента (осуществляется сопоставление мандатов пользователей в форме сертификатов открытых ключей с существующими учетными записями Windows NT).
8. Дополнительно к регистрации посредством ввода пароля может поддерживаться аутентификация с использованием смарт-карт.

В состав Windows NT входит Microsoft Certificate Server, позволяющий выдавать сотрудникам и партнерам сертификаты X.509 версии 3. Системные администраторы могут указывать, сертификаты каких уполномоченных являются доверяемыми в системе и, таким образом, контролировать аутентификацию доступа к ресурсам.

Внешние пользователи, не имеющие учетных записей Windows NT, могут быть аутентифицированы с помощью сертификатов открытых ключей и соотнесены с существующей учетной записью. Права доступа, назначенные для этой учетной записи, определяют права внешних пользователей на доступ к ресурсам.

В распоряжении пользователей находятся простые средства управления парами закрытых (открытых) ключей и сертификатами, используемые для доступа к ресурсам системы.

Технология шифрования встроена в операционную систему и позволяет использовать цифровые подписи для идентификации потоков.

9.3.1. СЕРВЕР АУТЕНТИФИКАЦИИ KERBEROS

Kerberos – это программный продукт, разработанный в середине 1980-х гг. в Массачусетском технологическом институте и претерпевший с тех пор ряд принципиальных изменений. Клиентские компоненты Kerberos присутствуют в большинстве современных операционных систем.

Kerberos предназначен для решения следующей задачи. Имеется открытая (незащищенная) сеть, в узлах которой сосредоточены ***субъекты*** – пользователи, а также клиентские и серверные программные системы. Каждый субъект обладает секретным ключом. Чтобы субъект C мог доказать свою подлинность субъекту S (без этого S не станет обслуживать C), он должен не только назвать себя, но и продемонстрировать знание секретного ключа. C не может просто послать S свой секретный ключ, во-первых, потому, что сеть открыта (доступна для пассивного и активного прослушивания), а, во-вторых, потому, что S не знает (и не должен знать) секретный ключ C. Требуется менее прямолинейный способ демонстрации знания секретного ключа.

Система Kerberos представляет собой ***доверенную третью сторону*** (т.е. сторону, которой доверяют все), владеющую секретными ключами обслуживаемых субъектов и помогающую им в попарной проверке подлинности.

Чтобы с помощью Kerberos получить доступ к S (обычно это сервер), C (как правило – клиент) посылает Kerberos запрос, содержащий сведения о нем (клиенте) и о запрашиваемой услуге. В ответ Kerberos возвращает так называемый ***билет***, зашифрованный секретным ключом сервера, и копию части информации из билета, зашифрованную секретным ключом клиента. Клиент должен расшифровать вторую порцию данных и переслать ее вместе с билетом серверу. Сервер, расшифровав билет, может сравнить его содержимое с дополнительной информацией, присланной клиентом. Совпадение свидетельствует

о том, что клиент смог расшифровать предназначенные ему данные (ведь содержимое билета никому, кроме сервера и Kerberos, недоступно), т.е. продемонстрировал знание секретного ключа. Значит, клиент – именно тот, за кого себя выдает. Подчеркнем, что секретные ключи в процессе проверки подлинности не передавались по сети (даже в зашифрованном виде) – они только использовались для шифрования. Как организован первоначальный обмен ключами между Kerberos и субъектами и как субъекты хранят свои секретные ключи – вопрос отдельный.

Проиллюстрируем описанную процедуру рис. 9.5, на котором обозначено: c и s – сведения (например, имя), соответственно, о клиенте и сервере; d1 и d2 – дополнительная (по отношению к билету) информация; Tc.s – билет для клиента C на обслуживание у сервера S; Kc и Ks – секретные ключи клиента и сервера; {info}K – информация info, зашифрованная ключом K.

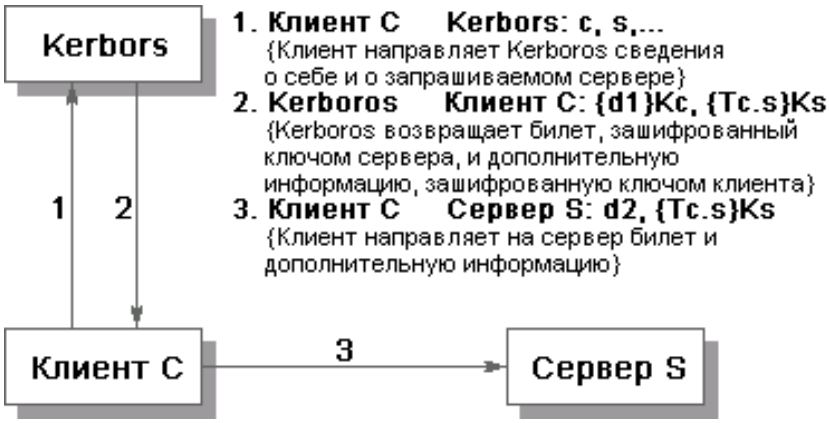

Рис. 9.5. Проверка сервером S подлинности клиента C

Протокол аутентификации Kerberos определяет взаимодействие между клиентом и сетевым сервисом аутентификации, известным как KDC (Key Distribution Center). В Windows NT KDC используется как сервис аутентификации на всех контроллерах домена. Домен Windows NT эквивалентен области Kerberos, но к ней обращаются как к домену. Реализация протокола Kerberos в Windows NT основана на определении Kerberos в RFC1510, клиент Kerberos реализован в виде ПФБ (поставщика функций безопасности) Windows NT, основанном на SSPI. Начальная аутентификация Kerberos интегрирована с процедурой WinLogon. Сервер Kerberos (KDC) интегрирован с существующими службами безопасности Windows NT, исполняемыми на контроллере домена. Для хранения информации о пользователях и группах он использует службу каталогов Active Directory.

Протокол Kerberos усиливает существующие функции безопасности Windows NT и добавляет новые:

1) повышенная скорость аутентификации при установлении начального соединения (сервер приложений не обращается к контроллеру домена для аутентификации клиента);

2) делегирование аутентификации в многоярусных архитектурах клиент-сервер (при подключении клиента к серверу, последний имперсонирует (олицетворяет) клиента в этой системе, но если серверу для завершения транзакции нужно выполнить сетевое подключение к другому серверу, протокол Kerberos позволяет делегировать аутентификацию первого сервера и подключиться ко второму от имени клиента);

3) транзитивные доверительные отношения для междудоменной аутентификации (т.е пользователь может быть аутентифицирован в любом месте дерева доменов) упрощают управление доменами в больших сетях с несколькими доменами.

Интеграция Kerberos. Протокол Kerberos полностью интегрирован с системой безопасности и контроля доступа Windows NT. Начальная регистрация в Windows NT обеспечивается процедурой WinLogon, использующей ПФБ Kerberos для получения начального билета TGT. Другие компоненты системы, например, Redirector, применяют интерфейс SSPI к ПФБ Kerberos для получения сеансового билета для удаленного доступа к файлам сервера SMB.

Взаимодействие Kerberos. Протокол Kerberos версии 5 реализован в различных системах и используется для единообразия аутентификации в распределенной сети.

Под взаимодействием Kerberos подразумевается общий протокол, позволяющий учетным записям аутентифицированных пользователей, хранящимся в одной базе, осуществлять доступ ко всем сервисам в гетерогенной среде. Взаимодействие Kerberos основывается на следующих характеристиках:

1) общий протокол аутентификации пользователя или сервиса по основному имени при сетевом подключении;

2) возможность определения доверительных отношений между областями Kerberos и создания ссылочных запросов билетов между областями;

3) поддержка определенных в RFC 1510 требований к взаимодействию, относящихся к алгоритмам шифрования и контрольных сумм, взаимной аутентификации и другим возможностям билетов;

4) поддержка форматов маркера безопасности Kerberos версии 5 для установления контекста и обмена сообщениями.

Поддержка Kerberos открытых ключей. В Windows NT также реализованы расширения протокола Kerberos, поддерживающие дополнительно к аутентификации с совместно используемым секретным ключом аутентификацию, основанную на парах открытого (закрытого) ключа. Поддержка открытых ключей позволяет клиентам запрашивать начальный ключ TGT с помощью закрытого ключа, в то время как KDC проверяет запрос с помощью открытого ключа, полученного из сертификата X.509 (хранится в пользовательском объекте в каталоге Active Directory), Сертификат пользователя может быть выдан

как сторонним уполномоченным сертификации (Certification Authority), так и Microsoft Certificate Server, входящим в Windows NT. После начальной аутентификации закрытым ключом используются стандартные протоколы Kerberos для получения сеансовых билетов на доступ к сетевым службам.

Модель безопасности Windows NT обеспечивает однородный и унифицированный механизм контроля за доступом к ресурсам домена на основе членства в группах. Компоненты безопасности Windows NT доверяют хранимой в каталоге информации о защите. Например, сервис аутентификации Windows NT хранит зашифрованные пароли пользователей в безопасной части каталога объектов пользователя. По умолчанию операционная система "считает", что правила безопасности защищены и не могут быть изменены кем-либо несанкционированно. Общая политика безопасности домена также хранится в каталоге Active Directory.

Делегирование административных полномочий – гибкий инструмент ограничения административной деятельности рамками части домена. Этот метод позволяет предоставить отдельным сотрудникам возможность управления пользователями или группами в заданных пределах и, в то же время, не дает им прав на управление учетными записями, относящимися к другим подразделениям.

Права на определение новых пользователей или создание групп пользователей делегируются на уровне OU или контейнера, в котором создана учетная запись.

Существует три способа делегирования административных полномочий:

1) на изменение свойств определенного контейнера, например, LocalDomainPolicies самого домена;
2) на создание и удаление дочерних объектов определенного типа (пользователи, группы, принтеры и пр.) внутри OU;
3) на обновление определенных свойств некоторых дочерних объектов внутри OU (например, право устанавливать пароль для объектов типа User).

Делегировать полномочия просто. Достаточно выбрать лицо, которому будут делегированы полномочия, и указать, какие именно полномочия передаются. Интерфейс программы администрирования Active Directory позволяет без затруднений просматривать информацию о делегировании, определенную для контейнеров.

Наследование прав доступа означает, что информация об управлении доступом, определенная в высших слоях контейнеров в каталоге, распространяется ниже – на вложенные контейнеры и объекты-листья. Существуют две модели наследования прав доступа: динамическая и статическая. При динамическом наследовании права определяются путем оценки разрешений на доступ, назначенных непосредственно для объекта, а также для всех родительских объектов в каталоге. Это позволяет эффективно управлять доступом к части дерева каталога, внося изменения в контейнер, влияющий на все вложенные контейнеры и объекты-листья. Обратная сторона такой гибкости – недостаточно высокая производительность из-за времени определения эффективных прав доступа при запросе пользователя.

В Windows NT реализована статическая форма наследования прав доступа, иногда также называемая наследованием в момент создания. Информация об управлении доступом к контейнеру распространяется на все вложенные объекты контейнера. При создании нового объекта наследуемые права сливаются с правами доступа, назначаемыми по умолчанию. Любые изменения наследуемых прав доступа, выполняемые в дальнейшем на высших уровнях дерева, должны распространяться на все дочерние объекты. Новые наследуемые права доступа распространяются на объекты Active Directory в соответствии с тем, как эти новые права определены. Статическая модель наследования позволяет увеличить производительность.

9.3.2. ЭЛЕМЕНТЫ БЕЗОПАСНОСТИ СИСТЕМЫ

Рассмотрим вопросы реализации политики безопасности: управление учетными записями пользователей и групп, исполнение и делегирование административных функций.

Учетные записи пользователей и групп. Любой пользователь Windows NT характеризуется определенной учетной записью. Под учетной записью понимается совокупность прав и дополнительных параметров, ассоциированных с определенным пользователем. Кроме того, пользователь принадлежит к одной или нескольким группам. Принадлежность к группе позволяет быстро назначать права доступа и полномочия.

К встроенным учетным записям пользователей относятся:

- Guest– учетная запись, фиксирующая минимальные привилегии гостя;
- Administrator – встроенная учетная запись для пользователей, наделенных максимальными привилегиями;
- Krbtgt – встроенная учетная запись, используемая при начальной аутентификации Kerberos.

Кроме них имеются две скрытые встроенные учетные записи:

- System – учетная запись, используемая операционной системой;
- Creator owner – создатель (файла или каталога).

Перечислим встроенные группы:

- локальные (Account operators; Administrators; Backup operators; Guests; Print operators; Replicator; Server operators; Users);
- глобальные (Domain guests – гости домена; Domain Users – пользователи домена; Domain Admins – администраторы домена).

Помимо этих встроенных групп имеется еще ряд специальных групп:

- Everyone – в эту группу по умолчанию включаются вообще все пользователи в системе;
- Authenticated users – в эту группу включаются только аутентифицированные пользователи домена;
- Self – сам объект.

Для просмотра и модификации свойств учетной записи достаточно щелкнуть имя пользователя или группы и на экране появится диалоговое окно User Properties:

General – общее описание пользователя;

Address – домашний и рабочий адрес пользователя;

Account – обязательные параметры учетной записи;

Telephone/notes – необязательные параметры;

Organization – дополнительные необязательные сведения;

Membership – обязательная информация о принадлежности пользователя к группам;

Dial-in – параметры удаленного доступа;

Object – идентификационные сведения о пользовательском объекте;

Security – информация о защите объекта.

Локальная политика безопасности регламентирует правила безопасности на локальном компьютере. С ее помощью можно распределить административные роли, конкретизировать привилегии пользователей, назначить правила аудита.

По умолчанию поддерживаются следующие области безопасности:

- политика безопасности – задание различных атрибутов безопасности на локальном и доменном уровнях; так же охватывает некоторые установки на машинном уровне;
- управление группами с ограничениями – позволяет управлять членством в группах, которые, по мнению администратора, "чувствительны" с точки зрения безопасности системы;
- управление правами и привилегиями – позволяет редактировать список пользователей и их специфических прав и привилегий;
- деревья объектов – включают три области защиты: объекты каталога Active Directory, ключи реестра, локальную файловую систему; для каждого объекта в дереве шаблоны безопасности позволяют конфигурировать и анализировать характеристики дескрипторов защиты, включая владельцев объекта, списки контроля доступа и параметры аудита;
- системные службы (сетевые или локальные) – построенные соответствующим образом дают возможность независимым производителям программного обеспечения расширять редактор конфигураций безопасности для устранения специфических проблем.

Конфигурирование безопасности. Для конфигурирования параметров безопасности системы используются шаблоны.

Управление доступом к реестру. *Реестр* – это дерево объектов. Доступ к каждому объекту в дереве должен быть регламентирован. Выбрав в окне обзорного просмотра ветвь, соответствующую шаблону Custom, щелкните папку Registry. В правой части окна появится список ветвей реестра, доступ к которым можно ограничивать. В шаблоне, поставляемом с редактором, приведена ветвь MACHINE\HARDWARE, которую надо истолковывать как HKEY_LOCAL_MACHINE\Hardware. Чтобы добавить к дереву новые ветви, их надо в явном виде прописать в шаблоне с помощью любого текстового редактора. Для разграничения доступа к выбранной ветви реестра дважды щелкните ее имя и укажите нужный тип доступа и имя соответствующей учетной записи. Изменения будут занесены в шаблон.

9.4. ЗАЩИТА В ОПЕРАЦИОННОЙ СИСТЕМЕ UNIX

Операционная система Unix относится к категории многопользовательских многопрограммных ОС, работающих в режиме разделения времени. Богатые возможности, заложенные в ОС Unix, сделали ее наиболее популярной в мире. ОС Unix поддерживается практически на всех типах ЭВМ.

Организация работ в ОС Unix основана на понятии последовательного процесса как единицы работы, управления и потребления ресурсов. Взаимодействие процессов внутри ядра (процесс вызывает ядро как подпрограмму) происходит по принципу сопрограмм. Последовательность вычислений внутри процесса строго выдерживается: процесс, в частности, не может активизировать ввод-вывод и продолжать вычисление параллельно с ним. В этом случае требуется создать параллельный процесс.

Ядро ОС Unix состоит из двух основных частей: управления процессами и управления устройствами. Управление процессами резервирует ресурсы, определяет последовательность выполнения процессов и принимает запросы на обслуживание. Управление устройствами контролирует передачу данных между ОП и периферийными устройствами.

В любой момент времени выполняется либо программа пользователя (процесс), либо команда ОС. В каждый момент времени лишь один пользовательский процесс активен, а все остальные приостановлены. Ядро ОС Unix служит для удовлетворения потребностей процессов.

Процесс – это программа на этапе выполнения. В некоторый момент времени программе могут соответствовать один или несколько процессов, или не соответствовать ни одного. Считается, что процесс является объектом, учтенным в специальной таблице ядра системы. Наиболее важная информация о процессе хранится в двух местах: в таблице процессов и в таблице пользователя, называемой также контекстом процесса. Таблица процессов всегда находится в памяти и содержит на каждый процесс по одному элементу, в котором отражается состояние процесса: адрес в памяти или адрес свопинга, размер, идентификаторы процесса и запустившего его пользователя. Таблица пользователя существует для каждого активного процесса и к ней могут непосредственно адресоваться только программы ядра (ядро резервирует по одному контексту на каждый активный процесс). В этой таблице содержится информация, требуемая во время выполнения процесса: идентификационные номера пользователя и группы, предназначенные для определения привилегий доступа к файлам, ссылки на системную таблицу файлов для всех открытых процессом файлов, указатель на индексный дескриптор текущего каталога в таблице индексных дескрипторов и список реакций на различные ситуации. Если процесс приостанавливается, контекст становится недоступным и немодифицируемым.

Каталоги файловой системы ОС Unix "спрятаны" от пользователей и защищены механизмами ОС. Скрытой частью файловой организации в ОС Unix является индексный дескриптор файла, который описывает расположение файла, его длину, метод доступа к файлу, даты, связанные с историей создания файла, идентификатор владельца и т.д.

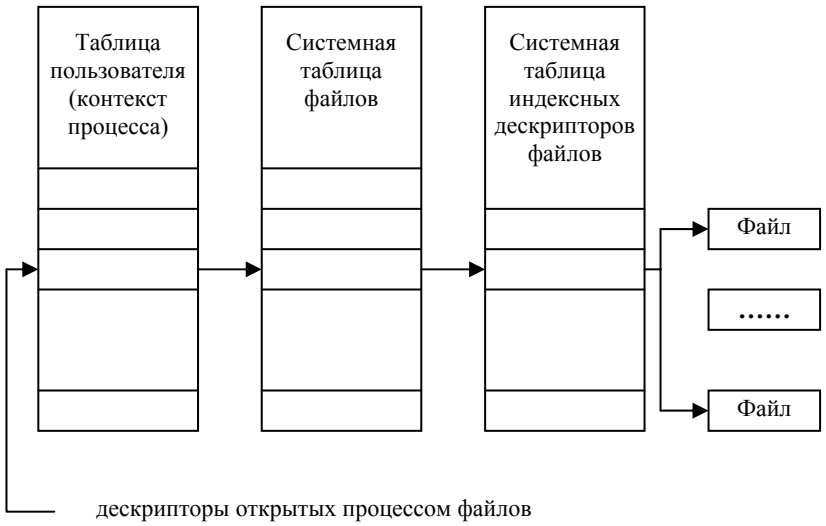

Рис. 9.6. Структура данных ядра ОС Unix

Работа с таблицами является привилегией ядра, что обеспечивает сохранность и безопасность системы. Структура данных ядра ОС, обеспечивающих доступ к файлам, приведена на рис. 9.6.

При взаимодействии с ОС Unix пользователь может обращаться к большому числу информационных объектов или файлов, объединенных в каталоги. Файловая система ОС Unix имеет иерархическую структуру.

В ОС Unix используется четыре типа файлов: обычные, специальные, каталоги, а в некоторых версиях ОС и FIFO-файлы (First In – First Out). Обычные файлы содержат данные пользователей. Специальные файлы предназначены для организации взаимодействия с устройствами ввода-вывода. Доступ к любому устройству реализуется как обслуживание запроса к специальному (дисковому) файлу. Каталоги используются системой для поддержания файловой структуры. Особенность каталогов состоит в том, что пользователь может читать их содержимое, но выполнять записи в каталоги (изменять структуру каталогов) может только ОС. В ОС Unix, организуются именованные программные каналы, являющиеся соединительным средством между стандартным выводом одной программы и стандартным вводом другой.

Схема типичной файловой системы ОС Unix приведена на рис. 9.7. Рассмотрим основные механизмы защиты данных, реализованные в ОС Unix.

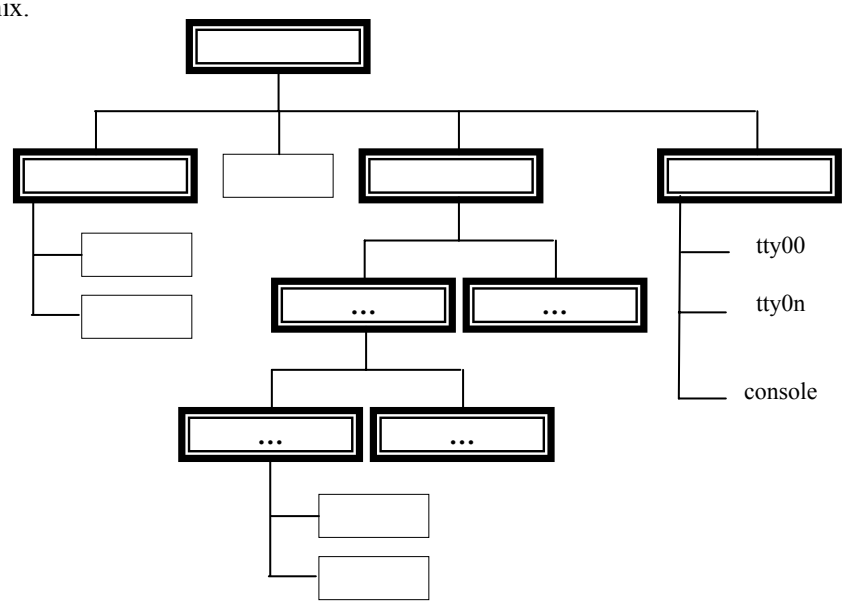

Рис. 9.7. Схема файловой системы ОС Unix

Управление доступом к системе. При включении пользователя в число абонентов ему выдается регистрационное имя (идентификатор) для входа в систему и пароль, который служит для подтверждения идентификатора пользователя. В отдельных версиях ОС Unix, помимо идентификатора и пароля, требуется ввод номера телефона, с которого выполняется подключение к системе. Администратор системы и пользователь могут изменить пароль командой passwd. При вводе этой команды ОС запрашивает ввод текущего пароля, а затем требует ввести новый пароль. Если предложенный пароль не удовлетворяет требования системы, то запрос на ввод пароля может быть повторен. Если предложенный пароль удовлетворителен, ОС просит ввести его снова, чтобы убедиться в корректности ввода пароля.

Пользователи, которым разрешен вход в систему, перечислены в учетном файле пользователей /etc/passwd. Этот текстовый файл содержит следующие данные: имя пользователя, зашифрованный пароль, идентификатор пользователя, иден-

тификатор группы, начальный текущий каталог и имя исполняемого файла, используемого в качестве интерпретатора команд. Пароль шифруется, как правило, с использованием DES-алгоритма.

Управление доступом к данным. Операционная система Unix поддерживает для любого файла комплекс характеристик, определяющих санкционированность доступа, тип файла, его размер и точное местоположение на диске. При каждом обращении к файлу система проверяет право пользоваться им. Операционная система Unix допускает выполнение трех типов операций над файлами: чтение, запись и выполнение. Чтение файла означает, что доступно его содержимое, а запись – что возможны изменения содержимого файла. Выполнение приводит либо к загрузке файла в ОП, либо к выполнению содержащихся в файле команд системного монитора Shell. Разрешение на выполнение каталога означает, что в нем допустим поиск с целью формирования полного имени на пути к файлу. Любой из файлов в ОС Unix имеет определенного владельца и привязан к некоторой группе. Файл наследует их от процесса, создавшего файл. Пользователь и группа, идентификаторы которых связаны с файлом, считаются его владельцами.

Идентификаторы пользователя и группы, связанные с процессом, определяют его права при доступе к файлам. По отношению к конкретному файлу все процессы делятся на три категории:

1) владелец файла (процессы, имевшие идентификатор пользователя, совпадающий с идентификатором владельца файла);

2) члены группы владельца файла (процессы, имеющие идентификатор группы, совпадающий с идентификатором группы, которой принадлежит файл);

3) прочие (процессы, не попавшие в первые две категории).

Владелец файла обладает одними привилегиями на доступ к нему, члены группы, в которую входит файл – другими, все остальные пользователи – третьими. Каждый файл содержит код защиты, который присваивается файлу при его создании. Код защиты располагается в индексном дескрипторе файла и содержит десять символов, причем первый символ определяет тип файла, а последующие девять – право на доступ к нему. Три вида операций (чтение, запись и выполнение) и три категории (уровни привилегий на доступ: владельцев, групп и прочих пользователей) дают в совокупности девять возможных вариантов разрешений или запретов на доступ к файлу. Первые три символа определяют возможности чтения (r), записи (w) и выполнения (e) на уровне владельца, следующие три – на уровне группы, в которую входит владелец, и последние три – на уровне остальных пользователей. Наличие символов r, w и e указывает на соответствующее разрешение.

Если процесс требует доступа к файлу, то сначала определяется категория, в которую по отношению к этому файлу он попадает. Затем из кода защиты выбираются те три символа, которые соответствуют данной категории, и выполняется проверка: разрешен ли процессу требуемый доступ. Если доступ не разрешен, системный вызов, посредством которого процесс сделал запрос на доступ, отвергается ядром ОС.

По соглашению, принятому в ОС Unix, привилегированный пользователь имеет идентификатор, равный нулю. Процесс, с которым связан нулевой идентификатор пользователя, считается привилегированным. Независимо от кода защиты файла привилегированный процесс имеет право доступа к файлу для чтения и записи. Если в коде защиты хотя бы одной категории пользователей (процессов) есть разрешение на выполнение файла, привилегированный процесс тоже имеет право выполнять этот файл.

С помощью специальных команд владелец файла (привилегированный пользователь) может изменять распределение привилегий. Команда Change mode позволяет изменить код защиты, команда Change owner меняет право на владение файлом, а команда Change group – принадлежность к той или иной группе. Пользователь может изменять режимы доступа только для файлов, которыми он владеет.

Защита хранимых данных. Для защиты хранимых данных в составе ОС Unix имеется утилита crypt, которая читает данные со стандартного ввода, шифрует их и направляет на стандартный вывод. Шифрование применяется при необходимости предоставления абсолютного права владения файлом.

Восстановление файловой системы. Операционная система Unix поддерживает три основных набора утилит копирования: программы volcopy/labelit, dump/restor и cpio. Программа volcopy целиком переписывает файловую систему, проверяя с помощью программы labelit соответствие меток требуемых томов. Программа dump обеспечивает копирование лишь тех файлов, которые были записаны позднее определенной даты (защита накоплением). Программа restor может анализировать данные, созданные программой dump, и восстанавливать отдельные файлы или всю файловую систему полностью. Программа cpio предназначена для создания одного большого файла, содержащего образ всей файловой системы или какой-либо ее части.

Для восстановления поврежденной, например, в результате сбоев в работе аппаратуры файловой системы используются программы fsck и fsdb.

За сохранность файловой системы, адаптацию программного обеспечения к конкретным условиям эксплуатации, периодическое копирование пользовательских файлов, восстановление потерянных данных и другие операции ответственность возложена на администратора системы.

Усложненное управление доступом. В составе утилит ОС Unix находится утилита cron, которая предоставляет возможность запускать пользовательские программы в определенные моменты (промежутки) времени и, соответственно, ввести временные параметры для ограничения доступа пользователей.

Для управления доступом в ОС Unix также применяется разрешение установки идентификатора владельца. Такое разрешение дает возможность получить привилегии владельца файла на время выполнения соответствующей программы. Владелец файлов может установить режим, в котором другие пользователи имеют возможность назначать собственные идентификаторы режима.

Доступ, основанный на полномочиях, использует соответствие меток. Для этого вводятся метки объектов (файлов) и субъектов (процессов), а также понятия доминанты и равенства меток (для выражения отношения между метками). Создаваемый файл наследует метку от создавшего его процесса. Вводятся соотношения, определяющие права процессов по отношению к файлам.

Интерфейс дискретного доступа существенно детализирует имеющиеся механизмы защиты ОС Unix. Вводимые средства можно разделить на следующие группы:

1) работа со списками доступа при дискретной защите;
2) проверка права доступа;
3) управление доступом на основе полномочий;
4) работа привилегированных пользователей.

В рамках проекта Posix создан интерфейс системного администратора. Указанный интерфейс определяет объекты и множества действий, которые можно выполнить над объектами. В качестве классов субъектов и объектов предложены пользователь, группа пользователей, устройство, файловая система, процесс, очередь, вход в очередь, машина, система, администратор, программное обеспечение и др. Определены атрибуты таких классов, операции над классами и события, которые могут с ними происходить.

9.5. ЗАЩИТА В ОПЕРАЦИОННОЙ СИСТЕМЕ NOVELL NETWARE

Авторизация доступа к данным сети. В NetWare реализованы три уровня защиты данных (рис. 9.8).

Здесь под аутентификацией понимается:

1) процесс подтверждения подлинности клиента при его подключении к сети,
2) процесс установления подлинности пакетов, передаваемых между сервером и рабочей станцией.

Права по отношению к файлу (каталогу) определяют, какие операции пользователь может выполнить с файлом (каталогом). Администратор может для каждого клиента сети определить права по отношению к любому сетевому файлу или каталогу.

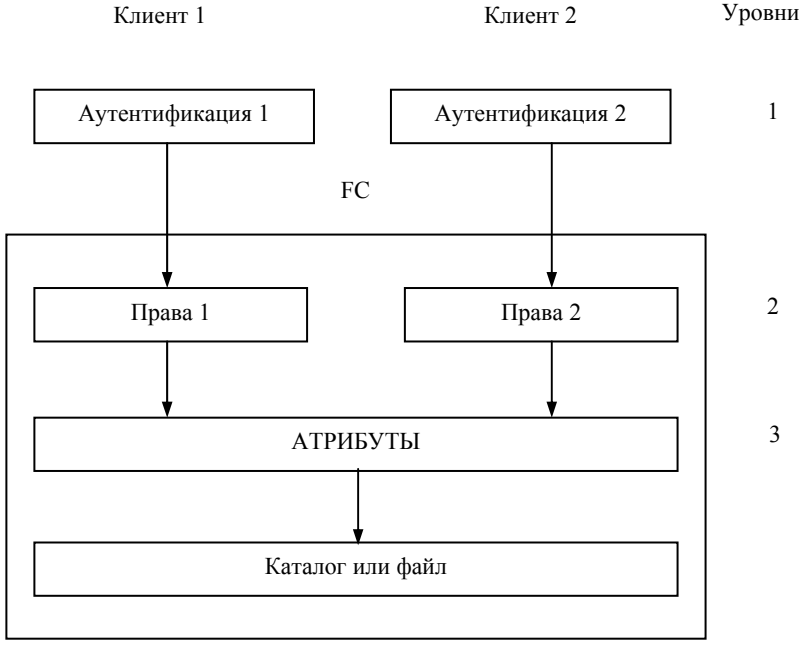

Рис. 9.8. Уровни защиты данных в Novell NetWare

Атрибуты определяют некоторые системные свойства файлов (каталогов). Они могут быть назначены администратором для любого сетевого файла или каталога.

Например, чтобы записать данные в файл, клиент должен:

1) знать свой идентификатор и пароль для подключения к сети,
2) иметь право записи данных в этот файл,
3) файл должен иметь атрибут, разрешающий запись данных.

Следует отметить, что атрибуты файла (каталога) имеют более высокий приоритет, чем права пользователей по отношению к этому файлу.

Аутентификация пользователей при подключении к сети. Подключение к сети выполняется с помощью утилиты LOGIN.EXE. Эта программа передает на сервер идентификатор, введенный пользователем.

По этому идентификатору NetWare выполняет поиск соответствующего объекта пользователя в системной базе данных сетевых ресурсов. Если в базе данных хранится значение пароля для этого клиента, то NetWare посылает на рабочую станцию зашифрованный с помощью пароля открытый ключ (симметричное шифрование). На рабочей станции этот ключ расшифровывается с помощью пароля, введенного пользователем, и используется для получения подписи запроса (пакета) к серверу о продолжении работы. Сервер расшифровывает эту подпись с помощью закрытого ключа (асимметричное шифрование), проверяет ее и посылает подтверждение на рабочую станцию. В дальнейшем каждый NCP-пакет снабжается подписью, получаемой в результате кодирования открытым ключом контрольной суммы содержимого пакета и случайного числа Nonce. Это число генерируется для каждого сеанса. Поэтому подписи пакетов не повторяются для разных сеансов, даже если пользователь выполняет те же самые действия

NCP-пакеты могут подписываться и рабочими станциями, и файловым сервером. Для инициирования включения подписи в NCP-пакеты администратор может задать один из следующих уровней:

0 – сервер не подписывает пакет;

1 – сервер подписывает пакет, если этого требует клиент (уровень на станции больше или равен 2);

2 – сервер подписывает пакет, если клиент также способен это сделать (уровень на станции больше или равен 1);

3 – сервер подписывает пакет и требует этого от всех клиентов (иначе подключение к сети невозможно).

Права пользователей по отношению к каталогам и файлам. Права, которые могут быть предоставлены пользователю (или группе пользователей) по отношению к каталогу или файлу, перечислены в табл. 9.3.

Права и фильтры (маски) наследуемых прав назначаются администратором сети с помощью утилит NetWare. Но назначение прав для каждого пользователя по отношению ко всем требуемым файлам и каталогам – это утомительная задача. В NetWare предлагается механизм наследования прав. Прежде всего введем некоторые определения.

Опекун (Trustees) – это пользователь (группа пользователей, другой объект), которому администратор с помощью утилиты (например, FILER) явно назначает права по отношению к какому-либо файлу или каталогу. Такие права называются опекунскими назначениями.

Фильтр наследуемых прав (IRF – Inherited Right Filter) – это свойство файла (каталога), определяющее, какие права данный файл (каталог) может унаследовать от родительского каталога. Фильтр назначается администратором с помощью утилиты (например, FILER).

Наследуемые права – права, передаваемые (распространяемые) от родительского каталога.

Эффективные права – права, которыми пользователь реально обладает по отношению к файлу или каталогу.

9.3. Список возможных прав по отношению к каталогу или файлу

Право	Обозначение	Описание
Supervisor	S	Предоставляет все права по отношению к каталогу или файлу, включая возможность назначения этого права другим пользователям. Не блокируется фильтром наследуемых прав IRF. Это право не может быть удалено ниже по дереву каталогов
Read	R	Чтение существующего файла (просмотр содержимого текстового файла, просмотр записей в файле базы данных и т.д.)
Write	W	Запись в существующий файл (добавление, удаление частей текста, редактирование записей базы данных)
Create	C	Создание в каталоге новых файлов (и запись в них) и подкаталогов. На уровне файла позволяет восстанавливать файл, если он был ошибочно удален
Erase	E	Удаление существующих файлов и каталогов
Modify	M	Изменение имен и атрибутов (файлов и каталогов), но не содержимого файлов
File Scan	F	Просмотр в каталоге имен файлов и подкаталогов. По отношению к файлу – возможность видеть структуру каталогов от корневого уровня до этого файла (путь доступа)
Access Control	A	Возможность предоставлять другим пользователям все права, кроме Supervisor. Возможность изменять фильтр наследуемых прав IRF

Права доступа к объектам NDS и их свойствам. Системная база данных сетевых ресурсов (СБДСР) представляет собой совокупность объектов, их свойств и значений этих свойств. В NetWare 4.x эта база данных называется NDS (NetWare Directory Services), а в NetWare 3.x – Bindery. Объекты NDS связаны между собой в иерархическую структуру, которую часто называют деревом NDS. На верхних уровнях дерева (ближе к корню [Root]) описываются логические ресурсы, которые принято называть *контейнерными* объектами. На самом нижнем (листьевом) уровне располагаются описания физических ресурсов, которые называют *оконечными* объектами.

В качестве контейнерных объектов используются объекты типа [Root] (корень), C (страна), O (организация), OU (организационная единица). Оконечные объекты – это User (пользователь), Group (группа), NetWare Server (сервер NetWare), Volume (том файлового сервера), Directories (директория тома) и т.д. Оконечные объекты имеют единое обозначение – CN.

В NetWare 4.x разработан механизм защиты дерева NDS. Этот механизм очень похож на механизм защиты файловой системы, который был рассмотрен ранее. Чтобы облегчить понимание этого механизма, оконечный объект можно интерпретировать как файл, а контейнерный объект – как каталог, в котором могут быть созданы другие контейнерные объекты (как бы подкаталоги) и оконечные объекты (как бы файлы). На рис. 9.9 представлена схема дерева NDS. Здесь символами [Root], C, O, OU обозначены контейнерные объекты, а символами CN – оконечные объекты.

В отличие от файловой системы здесь права по отношению к какому-либо объекту можно предоставить любому контейнерному или оконечному объекту дерева NDS. В частности допустимо рекурсивное назначение прав объекта по отношению к этому же объекту.

Права, которые могут быть предоставлены объекту по отношению к другому или тому же самому объекту, перечислены в табл. 9.4.

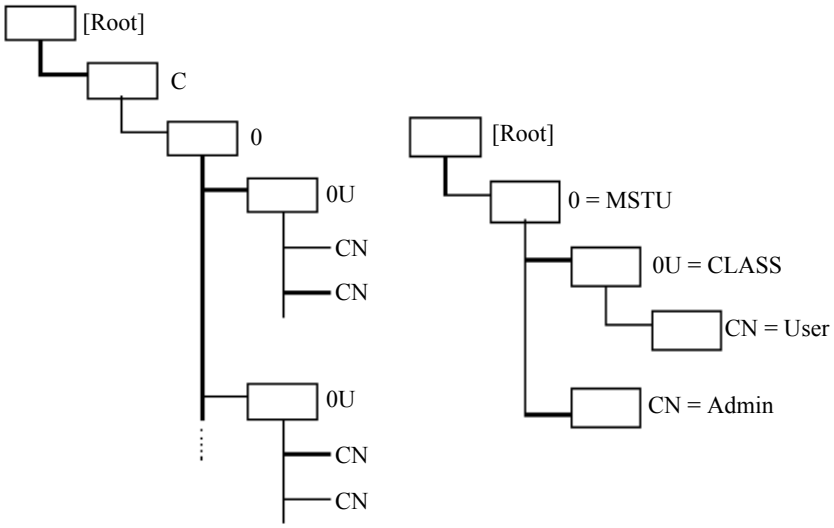

Рис. 9.9. Схема дерева NDS

9.4. Список возможных прав по отношению к объекту

Право	Обозначение	Описание
Supervisor	S	Гарантирует все привилегии по отношению к объекту и его свойствам. В отличие от файловой системы это право может быть блокировано фильтром наследуемых прав IRF, который может быть назначен для каждого объекта
Browse	B	Обеспечивает просмотр объекта в дереве NDS
Create	C	Это право может быть назначено только по отношению к контейнерному объекту (контейнеру). Позволяет создавать объекты в данном и во всех дочерних контейнерах
Delete	D	Позволяет удалять объект из дерева NDS
Rename	R	Позволяет изменять имя объекта

Администратор сети может для каждого объекта в дереве NDS определить значения свойств этого объекта. Для объекта User – это имя Login, требования к паролю, пароль пользователя, пользовательский сценарий подключения и т.д.

Контрольные вопросы к теме 9

1. Сформулируйте список функциональных дефектов с точки зрения защиты в используемой ОС.
2. Какие элементы безопасности содержит ОС Windows NT?
3. Назовите элементы безопасности ОС UNIX?
4. Охарактеризуйте элементы безопасности ОС Novell NetWare?

Т е м а 10. СИСТЕМЫ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

10.1. КЛАССИФИКАЦИЯ СИСТЕМ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Системы защиты ПО широко распространены и находятся в постоянном развитии, благодаря расширению рынка ПО и телекоммуникационных технологий. Необходимость использования систем защиты (СЗ) ПО обусловлена рядом проблем, среди которых следует выделить: незаконное использование алгоритмов, являющихся интеллектуальной собственностью автора, при написании аналогов продукта (промышленный шпионаж); несанкционированное использование ПО (кража и копирование); несанкционированная модификация ПО с целью внедрения программных злоупотреблений; незаконное распространение и сбыт ПО (пиратство).

Существующие системы защиты программного обеспечения можно классифицировать по ряду признаков, среди которых можно выделить:

1) метод установки;
2) используемые механизмы защиты;
3) принцип функционирования.

Системы защиты ПО по ***методу установки*** можно подразделить на:

1) системы, устанавливаемые на скомпилированные модули ПО;
2) системы, встраиваемые в исходный код ПО до компиляции;
3) комбинированные.

Системы первого типа наиболее удобны для производителя ПО, так как легко можно защитить уже полностью готовое и аттестированное ПО, а потому и наиболее популярны. В то же время стойкость этих систем достаточно низка (в зависимости от принципа действия СЗ), так как для обхода защиты достаточно определить точку завершения работы "конверта" защиты и передачи управления защищенной программе, а затем принудительно ее сохранить в незащищенном виде.

Системы второго типа неудобны для производителя ПО, так как возникает необходимость обучать персонал работе с программным интерфейсом (API) системы защиты с вытекающими отсюда денежными и временными затратами. Кроме того, усложняется процесс тестирования ПО и снижается его надежность, так как кроме самого ПО ошибки может содержать API системы защиты или процедуры, его использующие. Но такие системы являются более стойкими к атакам, потому что здесь исчезает четкая граница между системой защиты и как таковым ПО.

Наиболее живучими являются комбинированные системы защиты. Сохраняя достоинства и недостатки систем второго типа, они максимально затрудняют анализ и дезактивацию своих алгоритмов.

По ***используемым механизмам защиты*** СЗ можно классифицировать на:

1) системы, использующие сложные логические механизмы;
2) системы, использующие шифрование защищаемого ПО;
3) комбинированные системы.

Системы первого типа используют различные методы и приемы, ориентированные на затруднение дизассемблирования, отладки и анализа алгоритма СЗ и защищаемого ПО. Этот тип СЗ наименее стоек к атакам, так как для преодоления защиты достаточно проанализировать логику процедур проверки и должным образом их модифицировать.

Более стойкими являются системы второго типа. Для дезактивации таких защит необходимо определение ключа дешифрации ПО. Самыми стойкими к атакам являются комбинированные системы.

Для защиты ПО используется ряд методов, таких как:

1. *Алгоритмы запутывания* – используются хаотические переходы в разные части кода, внедрение ложных процедур – "пустышек", холостые циклы, искажение количества реальных параметров процедур ПО, разброс участков кода по разным областям ОЗУ и т.п.
2. *Алгоритмы мутации* – создаются таблицы соответствия операндов-синонимов и замена их друг на друга при каждом запуске программы по определенной схеме или случайным образом, случайные изменения структуры программы.
3. *Алгоритмы компрессии данных* – программа упаковывается, а затем распаковывается по мере выполнения.
4. *Алгоритмы шифрования данных* – программа шифруется, а затем расшифровывается по мере выполнения.
5. *Вычисление сложных математических выражений в процессе отработки механизма защиты* – элементы логики защиты зависят от результата вычисления значения какой-либо формулы или группы формул.
6. *Методы затруднения дизассемблирования* – используются различные приемы, направленные на предотвращение дизассемблирования в пакетном режиме.
7. *Методы затруднения отладки* – используются различные приемы, направленные на усложнение отладки программы.
8. *Эмуляция процессоров и операционных систем* – создается виртуальный процессор и/или операционная система (не обязательно существующие) и программа-переводчик из системы команд IBM в систему команд созданного процессора или ОС, после такого перевода ПО может выполняться только при помощи эмулятора, что резко затрудняет исследование алгоритма ПО.
9. *Нестандартные методы работы с аппаратным обеспечением* – модули системы защиты обращаются к аппаратуре ЭВМ, минуя процедуры ОС, и используют малоизвестные или недокументированные ее возможности.

В свою очередь, злоумышленники так же применяют ряд методов и средств для нарушения систем защиты. Ситуация противостояния разработчиков СЗПО и злоумышленников постоянно изменяется за счет комбинирования уже известных

методов защиты и нападения, а также за счет создания и использования новых методов. В целом это взаимодействие может быть описано схемой на рис. 10.1.

Рис. 10.1. Схема противостояния технических методов и средств защиты

По ***принципу функционирования СЗ*** можно подразделить на:

1) упаковщики/шифраторы;
2) СЗ от несанкционированного копирования;
3) СЗ от несанкционированного доступа (НСД).

10.2. ДОСТОИНСТВА И НЕДОСТАТКИ ОСНОВНЫХ СИСТЕМ ЗАЩИТЫ

Рассмотрим достоинства и недостатки основных систем защиты ПО, исходя из принципов их функционирования.

10.2.1. УПАКОВЩИКИ/ШИФРАТОРЫ

Первоначально основной целью упаковщиков/шифраторов являлось уменьшение объема исполняемого модуля на диске без ущерба для функциональности программы, но позднее на первый план вышла цель защиты ПО от анализа его алгоритмов и несанкционированной модификации. Для достижения этого используются алгоритмы компрессии данных; приемы, связанные с использованием недокументированных особенностей операционных систем и процессоров; шифрование данных, алгоритмы мутации, запутывание логики программы, приведение ОС в нестабильное состояние на время работы ПО и др.

Положительные стороны:

1. В рамках периода безопасного использования данные системы обеспечивают высокий уровень защиты ПО от анализа его алгоритмов.
2. Методы упаковки/шифрации намного увеличивают стойкость СЗ других типов.

Отрицательные стороны:

1. Практически все применяемые методы замедляют выполнение кода ПО.
2. Шифрование/упаковка кода ПО вызывает затруднения при обновлении (update) и исправлении ошибок (bugfix, servicepack).
3. Возможно повышение аппаратно-программных требований ПО.

4. В чистом виде данные системы не применимы для авторизации использования ПО.
5. Эти системы применимы лишь к продуктам небольшого объема.
6. Данный класс систем уязвим, так как программный код может быть, распакован или расшифрован для выполнения.
7. Обладают небольшим сроком безопасного использования ввиду п. 4.
8. Упаковка и шифрование исполняемого кода вступает в конфликт с запрещением самомодифицирующегося кода в современных ОС.

10.2.2. СИСТЕМЫ ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННОГО КОПИРОВАНИЯ

СЗ от несанкционированного копирования осуществляют "привязку" ПО к дистрибутивному носителю (гибкий диск, CD и др.). Данный тип защит основывается на глубоком изучении работы контроллеров накопителей, их физических показателей, нестандартных режимах разбивки, чтения/записи и т.п. При этом на физическом уровне создается дистрибутивный носитель, обладающий предположительно неповторимыми свойствами (нестандартная разметка носителя информации или/и запись на него дополнительной информации – пароля или метки), а на программном – создается модуль, настроенный на идентификацию и аутентификацию носителя по его уникальным свойствам. При этом возможно применение приемов, используемых упаковщиками/шифраторами.

Положительные факторы:

1. Затруднение нелегального копирования и распространения ПО.
2. Защита прав пользователя на приобретенное ПО.

Отрицательные факторы:

1. Большая трудоемкость реализации системы защиты.
2. Замедление продаж из-за необходимости физической передачи дистрибутивного носителя информации.
3. Повышение системных требований из-за защиты (наличие накопителя).
4. Снижение отказоустойчивости ПО.
5. Несовместимость защиты и аппаратуры пользователя (накопитель, контроллер).
6. На время работы ПО занимается накопитель.
7. Угроза кражи защищенного носителя.

10.2.3. СИСТЕМЫ ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

СЗ от НСД осуществляют предварительную или периодическую аутентификацию пользователя ПО или его компьютерной системы путем запроса дополнительной информации. К этому типу СЗ можно отнести системы парольной защиты ПО, системы "привязки" ПО к компьютеру пользователя, аппаратно-программные системы с электронными ключами и системы с "ключевыми дисками". В первом случае "ключевую" информацию вводит пользователь, во втором – она содержится в уникальных параметрах компьютерной системы пользователя, в третьем – "ключевая" информация считывается с микросхем электронного ключа, в четвертом – она хранится на диске.

10.2.3.1. Парольные защиты

Этот класс СЗПО, на сегодняшний день, является самым распространенным. Основной принцип работы данных систем заключается в идентификации и аутентификации пользователя ПО путем запроса дополнительных данных, это могут быть название фирмы и/или имя и фамилия пользователя и его пароль либо только пароль/регистрационный код. Эта информация может запрашиваться в различных ситуациях, например, при старте программы, по истечении срока бесплатного использования ПО, при вызове процедуры регистрации либо в процессе установки на ПК пользователя. Процедуры парольной защиты просты в реализации. Большинство парольных СЗПО используют логические механизмы, сводящиеся к проверке правильности пароля/кода и запуску или не запуску ПО, в зависимости от результатов проверки. Существуют также системы, шифрующие защищаемое ПО и использующие пароль или производную от него величину как ключ дешифрации. Обычно они реализованы в виде защитного модуля и вспомогательных библиотек и устанавливаются на уже скомпилированные модули ПО.

Слабым звеном парольных защит является блок проверки правильности введенного пароля/кода. Для такой проверки можно сравнивать введенный пароль с записанным в коде ПО правильным либо с правильно сгенерированным из введенных дополнительных данных паролем. Возможно также сравнение производных величин от введенного и правильного паролей, например их хэш-функций, в таком случае в коде можно сохранять только производную величину, что повышает стойкость защиты. Путем анализа процедур проверки можно найти реальный пароль, записанный в коде ПО, найти правильно сгенерированный пароль из введенных данных либо создать программу для перебора паролей для определения пароля с нужной хэш-суммой. Кроме того, если СЗПО не использует шифрования, достаточно лишь принудительно изменить логику проверки для получения беспрепятственного доступа к ПО.

Шифрующие системы более стойки к атакам, но при использовании простейших или некорректно реализованных криптоалгоритмов есть опасность дешифрации ПО.

Для всех парольных систем существует угроза перехвата пароля при его вводе авторизованным пользователем. Кроме того, в большинстве СЗПО данного типа процедура проверки используется лишь единожды, обычно при регистрации или установке ПО, затем система защиты просто отключается, что создает реальную угрозу для НСД при незаконном копировании ПО.

Положительные стороны:

1. Надежная защита от злоумышленника-непрофессионала.
2. Минимальные неудобства для пользователя.
3. Возможность передачи пароля/кода по сети.
4. Отсутствие конфликтов с системным и прикладным ПО и АО.
5. Простота реализации и применения.
6. Низкая стоимость.

Отрицательные стороны:

1. Низкая стойкость большинства систем защиты данного типа.
2. Пользователю необходимо запоминать пароль/код.

10.2.3.2. Системы "привязки" ПО

Системы этого типа при установке ПО на ПК пользователя осуществляют поиск уникальных признаков компьютерной системы либо они устанавливаются самой системой защиты. После этого модуль защиты в самом ПО настраивается на поиск и идентификацию данных признаков, по которым в дальнейшем определяется авторизованное или неавторизованное использование ПО. При этом возможно применение методик оценки скоростных и иных показателей процессора, материнской платы, дополнительных устройств, ОС, чтение/запись в микросхемы энергонезависимой памяти, запись скрытых файлов, настройка на наиболее часто встречаемую карту использования ОЗУ и т.п.

Слабым звеном таких защит является тот факт, что на ПК пользователя ПО всегда запускается на выполнение, что приводит к возможности принудительного сохранения ПО после отработки системы защиты, исследование самой защиты и выявление данных, используемым СЗПО для аутентификации ПК пользователя.

Положительные факторы:

1. Не требуется добавочных аппаратных средств для работы защиты.
2. Затруднение несанкционированного доступа к скопированному ПО.
3. Простота применения.
4. "Невидимость" СЗПО для пользователя.

Отрицательные факторы:

1. Ложные срабатывания СЗПО при любых изменениях в параметрах ПК.
2. Низкая стойкость при доступе злоумышленника к ПК пользователя.
3. Возможность конфликтов с системным ПО.

10.2.3.3. Программно-аппаратные средства защиты ПО с электронными ключами

Этот класс СЗПО в последнее время приобретает все большую популярность среди производителей программного обеспечения (ПО). Под программно-аппаратными средствами защиты в данном случае понимаются средства, основанные на использовании так называемых "аппаратных (электронных) ключей". ***Электронный ключ*** – это аппаратная часть системы защиты, представляющая собой плату с микросхемами памяти и, в некоторых случаях, микропроцессором, помещенную в корпус и предназначенную для установки в один из стандартных портов ПК (COM, LPT, PCMCIA, USB) или слот расширения материнской платы. Так же в качестве такого устройства могут использоваться смарт-карты(Smart-Card). По результатам проведенного анализа, программно-аппаратные средства защиты в настоящий момент являются одними из самых стойких систем защиты ПО от НСД.

Электронные ключи по архитектуре можно подразделить на *ключи с памятью* (без микропроцессора) и *ключи с микропроцессором* (и памятью).

Наименее стойкими являются системы с аппаратной частью первого типа. В таких системах критическая информация (ключ дешифрации, таблица переходов) хранится в памяти электронного ключа. Для дезактивации таких защит в большинстве случаев необходимо наличие у злоумышленника аппаратной части системы защиты (перехват диалога между программной и аппаратной частями для доступа к критической информации).

Наиболее стойкими являются системы с аппаратной частью второго типа. Такие комплексы содержат в аппаратной части не только ключ дешифрации, но и блоки шифрации/дешифрации данных, таким образом при работе защиты в электронный ключ передаются блоки зашифрованной информации, а принимаются оттуда расшифрованные данные. В системах этого типа достаточно сложно перехватить ключ дешифрации, так как все процедуры выполняются аппаратной частью, но остается возможность принудительного сохранения защищенной программы в открытом виде после отработки системы защиты. Кроме того, к ним применимы методы криптоанализа.

Положительные факторы:

1. Значительное затруднение нелегального распространения и использования ПО.
2. Избавление производителя ПО от разработки собственной системы защиты.
3. Высокая автоматизация процесса защиты ПО.
4. Наличие API системы для более глубокой защиты.
5. Возможность легкого создания демо-версий.
6. Достаточно большой выбор таких систем на рынке.

Отрицательные факторы:

1. Затруднение разработки и отладки ПО из-за ограничений со стороны СЗ.
2. Дополнительные затраты на приобретение СЗ и обучение персонала.
3. Замедление продаж из-за необходимости физической передачи аппаратной части.
4. Повышение системных требований из-за защиты (совместимость, драйверы).

5. Снижение отказоустойчивости ПО.
6. Несовместимость систем защиты и системного или прикладного ПО пользователя.
7. Несовместимость защиты и аппаратуры пользователя.
8. Ограничения из-за несовместимости электронных ключей различных фирм.
9. Снижение расширяемости компьютерной системы.
10. Затруднения или невозможность использования защищенного ПО в переносных и блокнотных ПК.
11. Наличие у аппаратной части размеров и веса (для COM/LPT = 5×3×2 см ~ 50 г).
12. Угроза кражи аппаратного ключа.

10.2.3.4. Средства защиты ПО с "ключевыми дисками"

В настоящий момент этот тип систем защиты мало распространен, ввиду его морального устаревания. СЗПО этого типа во многом аналогичны системам с электронными ключами, но здесь критическая информация хранится на специальном, ключевом, носителе. Основной угрозой для таких СЗПО является перехват считывания критической информации, а также незаконное копирование ключевого носителя.

Положительные и отрицательные стороны данного типа СЗПО практически полностью совпадают с таковыми у систем с электронными ключами.

10.3. ПОКАЗАТЕЛИ ЭФФЕКТИВНОСТИ СИСТЕМ ЗАЩИТЫ

Необходимо отметить, что пользователи явно ощущают лишь отрицательные стороны систем защит. А производители ПО рассматривают только относящиеся к ним "плюсы" и "минусы" систем защиты и практически не рассматривают факторы, относящиеся к конечному потребителю. По результатам исследований был разработан набор показателей применимости и критериев оценки СЗПО.

Показатели применимости:

Технические – соответствие СЗПО функциональным требованиям производителя ПО и требованиям по стойкости, системные требования ПО и системные требования СЗПО, объем ПО и объем СЗПО, функциональная направленность ПО, наличие и тип СЗ у аналогов ПО – конкурентов.

Экономические – соотношение потерь от пиратства и общего объема прибыли, соотношение потерь от пиратства и стоимости СЗПО и ее внедрения, соотношение стоимости ПО и стоимости СЗПО, соответствие стоимости СЗПО и ее внедрения поставленным целям.

Организационные – распространенность и популярность ПО, условия распространения и использования ПО, уникальность ПО, наличие угроз, вероятность превращения пользователя в злоумышленника, роль документации и поддержки при использовании ПО.

***Критерии оценки*:**

Защита как таковая – затруднение нелегального копирования и доступа, защита от мониторинга, отсутствие логических брешей и ошибок в реализации системы.

Стойкость к исследованию/взлому – применение стандартных механизмов, новые/нестандартные механизмы.

Отказоустойчивость (*надежность*) – вероятность отказа защиты (НСД), время наработки на отказ, вероятность отказа программы защиты (крах), время наработки на отказ, частота ложных срабатываний.

Независимость от конкретных реализаций ОС – использование недокументированных возможностей, "вирусных" технологий и "дыр" ОС.

Совместимость – отсутствие конфликтов с системным и прикладным ПО, отсутствие конфликтов с существующим АО, максимальная совместимость с разрабатываемым АО и ПО.

Неудобства для конечного пользователя ПО – необходимость и сложность дополнительной настройки системы защиты, доступность документации, доступность информации об обновлении модулей системы защиты из-за ошибок/несовместимости/нестойкости, доступность сервисных пакетов, безопасность сетевой передачи пароля/ключа, задержка из-за физической передачи пароля/ключа, нарушения прав потребителя.

Побочные эффекты – перегрузка трафика, отказ в обслуживании, замедление работы защищаемого ПО и ОС, захват системных ресурсов, перегрузка ОЗУ, нарушение стабильности ОС.

Стоимость – стоимость/эффективность, стоимость/цена защищаемого ПО, стоимость/ликвидированные убытки.

Доброкачественность – доступность результатов независимой экспертизы, доступность информации о побочных эффектах, полная информация о СЗ для конечного пользователя.

Общая картина взаимодействия агентов рынка программного обеспечения представлена на схеме рис. 10.2. Из четырех указанных выше видов среды взаимодействия защищающейся стороне подконтрольны три вида – организационная, техническая и экономическая среда. Важнейшей средой взаимодействия является несомненно экономическая среда, так как экономическое взаимодействие, является первопричиной и целью всего взаимодействия.

При разработке и анализе защиты программного обеспечения необходимо учитывать существующую законодательную базу, при этом нужно проводить подробный экономический анализ ситуации, применяя различные критерии оценки, а затем создавать стратегию защиты, включающую применение технических и организационных мер защиты программного обеспечения.

Рис. 10.2. Взаимодействие участников процесса создания и распространения ПО

Контрольные вопросы к теме 10

1. Приведите классификацию систем защиты программного обеспечения.
2. Сравните основные технические методы и средства защиты программного обеспечения.
3. Назовите отличия систем защиты от несанкционированного доступа от систем защиты от несанкционированного копирования.
4. Дайте характеристику показателей эффективности систем защиты.
5. Приведите примеры взаимодействия участников процесса создания и распространения ПО.

Т е м а 11. ПРОТОКОЛИРОВАНИЕ И АУДИТ

11.1. Основные понятия

Под ***протоколированием*** понимается сбор и накопление информации о событиях, происходящих в информационной системе. У каждого сервиса свой набор возможных событий, но в любом случае их можно разделить на внешние (вызванные действиями других сервисов), внутренние (вызванные действиями самого сервиса) и клиентские (вызванные действиями пользователей и администраторов).

Аудит – это анализ накопленной информации, проводимый оперативно, в реальном времени или периодически. Оперативный аудит с автоматическим реагированием на выявленные нештатные ситуации называется активным.

Реализация протоколирования и аудита решает следующие задачи:

1) обеспечение подотчетности пользователей и администраторов;
2) обеспечение возможности реконструкции последовательности событий;
3) обнаружение попыток нарушений информационной безопасности;
4) предоставление информации для выявления и анализа проблем.

Слишком обширное или подробное протоколирование не только снижает производительность сервисов, но и затрудняет аудит, т.е. не увеличивает, а уменьшает информационную безопасность.

Разумный подход к упомянутым вопросам применительно к операционным системам предлагается в "Оранжевой книге", где выделены следующие события: вход в систему (успешный или нет); выход из системы; обращение к удаленной системе; операции с файлами (открыть, закрыть, переименовать, удалить); смена привилегий или иных атрибутов безопасности (режима доступа, уровня благонадежности пользователя и т.п.).

При протоколировании события рекомендуется записывать, по крайней мере, следующую информацию: дата и время события; уникальный идентификатор пользователя – инициатора действия; тип события; результат действия (успех или неудача); источник запроса (например, имя терминала); имена затронутых объектов (например, открываемых или удаляемых файлов); описание изменений, внесенных в базы данных защиты (например, новая метка безопасности объекта).

Характерная особенность протоколирования и аудита – зависимость от других средств безопасности. Идентификация и аутентификация служат отправной точкой подотчетности пользователей, логическое управление доступом защищает конфиденциальность и целостность регистрационной информации. Возможно, для защиты привлекаются и криптографические методы.

Возвращаясь к целям протоколирования и аудита, отметим, что обеспечение подотчетности важно, в первую очередь, как сдерживающее средство. Если пользователи и администраторы знают, что все их действия фиксируются, они, возможно, воздержатся от незаконных операций. Очевидно, если есть основания подозревать какого-либо пользователя в нечестности, можно регистрировать все его действия, вплоть до каждого нажатия клавиши. При этом обеспечивается не только возможность расследования случаев нарушения режима безопасности, но и откат некорректных изменений (если в протоколе присутствуют данные до и после модификации). Тем самым защищается целостность информации.

Реконструкция последовательности событий позволяет выявить слабости в защите сервисов, найти виновника вторжения, оценить масштабы причиненного ущерба и вернуться к нормальной работе.

Выявление и анализ проблем могут помочь улучшить такой параметр безопасности, как доступность. Обнаружив узкие места, можно переконфигурировать или перенастроить систему, снова измерить производительность и т.д.

Непросто осуществить организацию согласованного протоколирования и аудита в распределенной разнородной системе. Во-первых, некоторые компоненты, важные для безопасности (например, маршрутизаторы), могут не обладать своими ресурсами протоколирования; в таком случае их нужно экранировать другими сервисами, которые возьмут протоколирование на себя. Во-вторых, необходимо увязывать между собой события в разных сервисах.

11.2. Активный аудит

Под подозрительной активностью понимается поведение пользователя или компонента информационной системы, являющееся злоумышленным (в соответствии с заранее определенной политикой безопасности) или нетипичным (согласно принятым критериям).

Задача активного аудита – оперативно выявлять подозрительную активность и предоставлять средства для автоматического реагирования на нее.

Активность, не соответствующую политике безопасности, целесообразно разделить на атаки, направленные на незаконное получение полномочий, и на действия, выполняемые в рамках имеющихся полномочий, но нарушающие политику безопасности.

Атаки нарушают любую осмысленную политику безопасности. Иными словами, активность атакующего является разрушительной независимо от политики. Следовательно, для описания и выявления атак можно применять универсальные методы, инвариантные относительно политики безопасности, такие как сигнатуры и их обнаружение во входном потоке событий с помощью аппарата экспертных систем.

Сигнатура атаки – это совокупность условий, при выполнении которых атака считается имеющей место, что вызывает заранее определенную реакцию. Простейший пример сигнатуры – "зафиксированы три последовательные неудачные попытки входа в систему с одного терминала", пример ассоциированной реакции – блокирование терминала до прояснения ситуации.

Действия, выполняемые в рамках имеющихся полномочий, но нарушающие политику безопасности, мы будем называть злоупотреблением полномочиями. Злоупотребления полномочиями возможны из-за неадекватности средств разграничения доступа выбранной политике безопасности. Простейшим примером злоупотреблений является неэтичное поведение суперпользователя, просматривающего личные файлы других пользователей. Анализируя регистрационную информацию, можно обнаружить подобные события и сообщить о них администратору безопасности, хотя для этого необходимы соответствующие средства выражения политики безопасности.

Выделение злоупотреблений полномочиями в отдельную группу неправомерных действий, выявляемых средствами активного аудита, не является общепринятым, но подобный подход имеет право на существование.

Нетипичное поведение выявляется статистическими методами. В простейшем случае применяют систему порогов, превышение которых является подозрительным. В более развитых системах производится сопоставление долговременных характеристик работы (называемых долгосрочным профилем) с краткосрочными профилями. (Здесь можно усмотреть аналогию биометрической аутентификации по поведенческим характеристикам.)

Применительно к средствам активного аудита различают ошибки первого и второго рода: пропуск атак и ложные тревоги. Нежелательность ошибок первого рода очевидна; ошибки второго рода – отвлекают администратора безопасности от действительно важных дел, косвенно способствуя пропуску атак.

Достоинства сигнатурного метода – высокая производительность, малое число ошибок второго рода, обоснованность решений. Основной недостаток – неумение обнаруживать неизвестные атаки и вариации известных атак.

Основные достоинства статистического подхода – универсальность и обоснованность решений, потенциальная способность обнаруживать неизвестные атаки, т.е. минимизация числа ошибок первого рода. Минусы заключаются в относительно высокой доле ошибок второго рода, плохой работе в случае, когда неправомерное поведение является типичным, когда типичное поведение плавно меняется от легального к неправомерному, а также в случаях, когда типичного поведения нет (как показывает статистика, примерно 5…10 %).

Средства активного аудита могут располагаться на всех линиях обороны информационной системы. На границе контролируемой зоны они могут обнаруживать подозрительную активность в точках подключения к внешним сетям (не только попытки нелегального проникновения, но и действия по "прощупыванию" сервисов безопасности). В корпоративной сети, в рамках информационных сервисов и сервисов безопасности, активный аудит в состоянии обнаружить и пресечь подозрительную активность внешних и внутренних пользователей, выявить проблемы в работе сервисов, вызванные как нарушениями безопасности, так и аппаратно-программными ошибками. Важно отметить, что активный аудит, в принципе, способен обеспечить защиту от атак на доступность.

Активный аудит развивается более десяти лет, и первые результаты казались весьма многообещающими. Довольно быстро удалось реализовать распознавание простых типовых атак, однако затем было выявлено множество проблем, связанных с обнаружением заранее неизвестных атак, атак распределенных, растянутых во времени и т.п. (Оперативное пополнение базы сигнатур атак таким решением, конечно, не является.) Тем не менее, и на нынешней стадии развития активный аудит полезен как один из рубежей (вернее, как набор прослоек) эшелонированной обороны.

11.3. Функциональные компоненты и архитектура

В составе средств активного аудита можно выделить следующие функциональные компоненты:

- компоненты генерации регистрационной информации. Они находятся на стыке между средствами активного аудита и контролируемыми объектами;
- компоненты хранения сгенерированной регистрационной информации;
- компоненты извлечения регистрационной информации (сенсоры).

Обычно различают сетевые и хостовые сенсоры, имея в виду под первыми выделенные компьютеры, сетевые карты которых установлены в режим прослушивания, а под вторыми – программы, читающие регистрационные журналы операционной системы. На наш взгляд, с развитием коммутационных технологий это различие постепенно стирается, так как сетевые сенсоры приходится устанавливать в активном сетевом оборудовании и, по сути, они становятся частью сетевой ОС;

- компоненты просмотра регистрационной информации. Могут помочь при принятии решения о реагировании на подозрительную активность;
- компоненты анализа информации, поступившей от сенсоров. В соответствии с данным выше определением средств активного аудита, выделяют пороговый анализатор, анализатор нарушений политики безопасности, экспертную систему, выявляющую сигнатуры атак, а также статистический анализатор, обнаруживающий нетипичное поведение;
- компоненты хранения информации, участвующей в анализе. Такое хранение необходимо, например, для выявления атак, протяженных во времени;
- компоненты принятия решений и реагирования ("решатели"). "Решатель" может получать информацию не только от локальных, но и от внешних анализаторов, проводя так называемый корреляционный анализ распределенных событий;
- компоненты хранения информации о контролируемых объектах. Здесь могут храниться пассивные данные и методы, необходимые, например, для извлечения из объекта регистрационной информации или для реагирования;
- компоненты, играющие роль организующей оболочки для менеджеров активного аудита, называемые мониторами и объединяющие анализаторы, "решатели", хранилище описаний объектов и интерфейсные компоненты. В число последних входят компоненты интерфейса с другими мониторами как равноправными, так и входящими в иерархию. Такие интерфейсы необходимы, например, для выявления распределенных, широкомасштабных атак;
- компоненты интерфейса с администратором безопасности.

Средства активного аудита строятся в архитектуре менеджер/агент. Основными агентскими компонентами являются сенсоры. Анализ, принятие решений – функции менеджеров. Очевидно, между менеджерами и агентами должны быть сформированы доверенные каналы.

Подчеркнем важность интерфейсных компонентов. Они полезны как с внутренней для средств активного аудита точки зрения (обеспечивают расширяемость, подключение компонентов различных производителей), так и с внешней точки зрения. Между менеджерами (между компонентами анализа и "решателями") могут существовать горизонтальные связи, необходимые для анализа распределенной активности. Возможно также формирование иерархий средств активного аудита с вынесением на верхние уровни информации о наиболее масштабной и опасной активности.

Обратим также внимание на архитектурную общность средств активного аудита и управления, являющуюся следствием общности выполняемых функций. Продуманные интерфейсные компоненты могут существенно облегчить совместную работу этих средств.

Контрольные вопросы к теме 11

1. Приведите определение понятий "протоколирование" и "аудит".
2. Назовите задачи, реализуемые протоколированием и аудитом.
3. Дайте характеристику задачи активного аудита.
4. Дайте характеристику сигнатурного метода активного аудита.
5. Охарактеризуйте функциональные компоненты активного аудита.

СПИСОК ЛИТЕРАТУРЫ

Основная

1. Дейтел, Х.М. Операционные системы. Ч. 2: Распределенные системы, сети, безопасность / Х.М. Дейтел, П.Дж. Дейтел, Д.Р. Чофнес. – М. : Бином, 2006.
2. Дейтел, Х.М. Операционные системы. Ч. 1: Основы и принципы / Х.М. Дейтел, П.Дж. Дейтел, Д.Р. Чофнес. – М. : Бином, 2006.
3. Гордеев, А.В. Операционные системы : учебник для вузов / А.В. Гордеев. – СПб. : Питер, 2004. – 416 с.
4. Олифер, В.Г. Сетевые операционные системы / В.Г. Олифер, Н.А. Олифер. – СПб. : Питер, 2001. – 544 с.
5. Танненбаум, Э. Современные операционные системы. 2-е изд. / Э. Танненбаум. – СПб. : Питер, 2002. – 1040 с.
6. Кастер, Х. Основы Windows NT и NTFS. Русская редакция / Х. Кастер. – М., 1996.
7. Проскурин, В.Г. Защита в операционных системах / В.Г. Проскурин, С.В. Крутов, И.В. Мацкевич. – М. : Радио и связь, 2000.

Дополнительная

1. Белкин, П.Ю. Защита программ и данных / П.Ю. Белкин, О.О. Михальский, А.С. Першаков [и др.]. – М. : Радио и связь, 1999.
2. Дунаев, С.Б. UNIX System V Release 4.2. Общее руководство / С.Б. Дунаев. – М. : Диалог-МИФИ, 1996.
3. Фролов, А.В. Библиотека системного программиста. Т. 11 – 13. Операционная система Microsoft Windows 3.1 для программиста / А.В. Фролов, Г.В. Фролов. – М. : Диалог-МИФИ, 1994.
4. Фролов, А.В. Библиотека системного программиста. Т. 14. Графический интерфейс GDI в Microsoft Windows / А.В. Фролов,
Г.В. Фролов. – М. : Диалог-МИФИ, 1994.
5. Фролов, А.В. Библиотека системного программиста. Т. 17. Microsoft Windows 3.1. Дополнительные главы / А.В. Фролов, Г.В. Фролов. – М. : Диалог-МИФИ, 1995.
6. Фролов, А.В. Библиотека системного программиста. Т. 26–27. Программирование для Windows NT / А.В. Фролов, Г.В. Фролов. – М. : Диалог-МИФИ, 1996.
7. Дейтел, Г. Введение в операционные системы. В 2 т. / Г. Дейтел ; пер с англ. – М. : Мир, 1987. – 359 с., 398 с.
8. Завгородний, В.И. Комплексная защита информации в компьютерных системах : учебное пособие для вузов / В.И. Завгородний. – М. : Логос, 2001. – 264 с.
9. Зегжда, Д.П. Основы безопасности информационных систем / Д.П. Зегжда, А.М. Ивашко. – М. : Горячая Линия–Телеком, 2000. – 452 с.
10. Малюк, А.А. Введение в защиту информации в автоматизированных системах / А.А. Малюк, С.В. Пазизин, Н.С. Погожин. – М. : Горячая Линия–Телеком, 2001. – 148 с.
11. Теоретические основы компьютерной безопасности : учебное пособие для вузов / П.Н. Девянин, О.О. Михальский, Д.И. Правиков, А.Ю. Щербаков. – М. : Радио и связь, 2000. – 192 с.
12. Щеглов, А.Ю. Защита компьютерной информации от несанкционированного доступа / А.Ю. Щеглов. – СПб. : Наука и техника, 2004. – 384 с.
13. UNIX: Руководство системного администратора. 3-е изд. / Эви Немет, Гарт Снайдер, Скотт Сибасс, Трент Р.Хейн ; пер. с англ. С.М. Тимачева; под ред. М.В. Коломыцева. – Киев : BHV, 1998. – 832 с.
14. Стандарты и рекомендации в области информационной безопасности // JetInfo. – 1996. – № 1 – 3. – 32 с.
15. Доступность как элемент информационной безопасности // JetInfo. – 1997. – № 1. – 28 с.
16. Программно-технологическая безопасность информационных систем // JetInfo. – 1997. – № 6–7. – 28 с.
17. Общие критерии оценки безопасности информационных технологий и перспективы их использования // JetInfo. – 1998. – № 1. – С. 12 – 17.
18. Современная трактовка сервисов безопасности // JetInfo. – 1999. – № 5. – С. 14 – 24.
19. Аудит безопасности информационных систем // JetInfo. – 1999. – № 9. – 24 с.