1. Характеристика операционных систем специального назначения

**1.1. Операционная система МСВС**

В настоящее время в ВС РФ принята единая концепция развития автоматизированных систем управления, обеспечивающая единое информационное пространство на базе распределенных сетей доступа.

Данная концепция включает в себя унифицированный комплекс общего программного обеспечения:

* операционная система;
* СУБД;
* технология проектирования АСУ;
* средства комплексной защиты программного обеспечения.

Основными документами, определяющими данную концепцию, являются:

1. ДГШ N 331/4/04 14 30.10.01 года «Концепция создания и оснащение базовыми информационными защищенными компьютерными технологиями ВС РФ», которая обеспечивает переход действующих и разрабатываемых автоматизированных систем ВС РФ на отечественные базовые защищенные информационные технологии на основе ОС МСВС, СУБД «Линтер В – 6.0», обеспечивающих техническую независимость, информационную безопасность и обеспечение построения единого защищенного информационного пространства ВС РФ».
2. Директива МО РФ 0414 19 августа 2002 год запрещает использование в ВС РФ несертифицированных программных продуктов и технологии рассылки. Принято на снабжение ГИС военно-топографическим управлением МО РФ

* «ИНТЕГРАЦИЯ», «СОГРИН», формат карт определен как *\*.SXF*.

1. Приказ МО РФ № 190 13.05.02 года. Согласно данного приказа на вооружение приняты следующие программные продукты:

* ОС МСВС;
* СУБД «Линтер В-6.0»;
* КП «Офис» 1.0 (комплект офисных приложений).

**Общие сведения, назначение и состав ОС МСВС**. ОС МСВС (мобильная система вооруженных сил) – это мобильная; многопользовательская, многозадачная операционная система, поддерживающая симметричные многопроцессорные архитектуры и работающая как в режиме командной строки, так и в режиме графического интерфейса. Разработана на основе ОС *Linux*.

Напомним, что ОС – это совокупность программных средств, обеспечивающих управление аппаратными ресурсами вычислительной системы и взаимодействие программных процессов с аппаратурой, другими процессами и пользователями.

Основное назначение ОС МСВС – управление ресурсами системы и процессами, использующими эти ресурсы при вычислениях.

ОС МСВС – это программное средство для управления техническими средствами и задачами в следующих областях:

* автоматизированные системы управления производством;
* автоматизированные системы управления технологическим процессом;
* информационные системы;
* системы массового обслуживания;
* системы сбора и анализа информации;
* многопользовательские системы общего назначения.

ОС МСВС – это программное изделие, поставляемое в виде загрузочного модуля и комплекта эксплуатационной документации. Загрузочный модуль поставляется на *CD-ROM*, а комплект эксплуатационной документации поставляется на бумажном носителе.

В состав ОС МСВС входят четыре комплекса:

* базовая конфигурация ОС;
* система графического интерфейса;
* система защиты от НСД;
* средства разработки.

Реализация ОС МСВС обеспечивает выполнение следующих функций:

* управление выполнением процессов;
* планирование очередности предоставления выполняющимся процессам времени центрального процессора;
* выделение выполняемому процессу оперативной памяти;
* выделение внешней памяти с целью обеспечения эффективного хранения информации и выборки данных пользователя;
* управление доступом процессов к периферийным устройствам (терминалы, дисководы, сетевое оборудование и др.).

ОС МСВС предоставляет различным категориям пользователей средства по реализации широкого спектра функциональных возможностей.

Для программистов и системных программистов – это средства разработки и отладки прикладных и системных программ, для администраторов систем и сетей – это высокоэффективные средства установки и управления системами и сетями, а для рядовых пользователей – это средства эффективной работы с различными прикладными программами.

Для автоматизации повседневной деятельности должностных лиц МО РФ предусмотрен комплекс программ КП «ОФИС», функционирующих под управлением ОС МСВС.

Организация пользовательского интерфейса и обслуживание файловой структуры. Комплекс «Базовая конфигурация ОС». Комплекс

«Базовая конфигурация ОС» предназначен для выполнения основных функций ОС и по существу является ОС, в которой отсутствуют система графического интерфейса, система защиты от несанкционированного доступа и средства разработки.

Комплекс «Базовая конфигурация ОС» может иметь самостоятельное применение в тех случаях, когда пользователь не нуждается в специальных средствах защиты информации, средствах разработки программ и вся работа пользователя осуществляется в режиме командной строки. Для функционирования этого комплекса требуется значительно меньше вычислительных ресурсов, чем для функционирования ОС МСВС.

В состав комплекса «Базовая конфигурация ОС» входят четыре компонента:

* ядро и окружение ФЛИР.10001-01;
* системные утилиты ФЛИР. 10002-01;
* системные библиотеки ФЛИ Р. 10003-01;
* средства установки ФЛИР. 10004-01. Системные утилиты содержат:
* набор утилит первичной инициализации системы;
* набор стандартных (*POSIX*) утилит для работы с файлами;
* набор утилит для работы с файловыми системами;
* набор утилит для архивирования и создания резервных копий;
* файловый менеджер с псевдографическим интерфейсом для консоли;
* набор утилит поддержки бюджетов пользователей и групп;
* программы – планировщики;
* набор программ для работы с печатающими устройствами;
* набор программ, работающих с сетевыми службами;
* программу поддержки русской раскладки клавиатуры;
* программы для работы с текстовыми файлами;
* командные интерпретаторы;
* систему выдачи пользовательских инструкций;
* сетевые службы.

Комплект «Системные библиотеки» обеспечивает взаимодействие ядра и

исполняемых модулей.

Компонент «Средства установки» обеспечивает начальную установку и дополнительную настройку комплекса «Базовая конфигурация ОС». Кроме того, компонент «Средства установки» обеспечивает возможность установки прикладных пакетов для их работы под управлением комплекса «Базовая конфигурация ОС».

**Комплекс «Система графического интерфейса»**. В состав комплекса

«Система графического интерфейса» входят четыре компонента:

* серверы графического интерфейса ФЛИР. 10005-01;
* библиотеки графического интерфейса ФЛИР. 10006-01;
* оконные менеджеры ФЛИР. 10007-01;
* утилиты графического интерфейса ФЛИР. 10008-01.

Комплекс «Система графического интерфейса» предназначен для организации выполнения прикладных задач в режиме графического интерфейса.

Для ОС типа *Unix* стандартом графического интерфейса стала система *X Window System* (далее по тексту *X-Window*).

*X Window System — оконная система, обеспечивающая стандартные инструменты и протоколы для построения графического интерфейса пользователя. Используется в UNIX-подобных ОС.*

*X Window System обеспечивает базовые функции графической среды: отрисовку и перемещение окон на экране, взаимодействие с устройствами ввода, такими как, например, мышь и клавиатура. X Window System не определяет деталей интерфейса пользователя — этим занимаются менеджеры окон, которых разработано множество. По этой причине внешний вид программ в среде X Window System может очень сильно различаться в зависимости от возможностей и настроек конкретного оконного менеджера.*

*В X Window System предусмотрена сетевая прозрачность: графические приложения могут выполняться на другой машине в сети, а их интерфейс при этом будет передаваться по сети и отображаться на локальной машине пользователя. В контексте X Window System термины «клиент» и «сервер» имеют непривычное для многих пользователей значение: «сервер» означает локальный дисплей пользователя (дисплейный сервер), а «клиент» — программу, которая этот дисплей использует (она может выполняться на удалённом компьютере).*

ОС МСВС является ОС семейства *Linix*, графическим окружением которой является *X Window*.

Система графического интерфейса (СГИ) обеспечивает многооконный графический режим работы в среде ОС МСВС в режиме разделения времени, который разрешает одновременное использование на одном экране нескольких прикладных программ, находящихся на одном или нескольких узлах локальной вычислительной сети.

СГИ предоставляет пользователю возможность одновременно взаимодействовать с несколькими прикладными программами, каждая из которых представлена на экране одним или несколькими окнами. В СГИ предусмотрены разнообразные возможности для графического вывода. СГИ обеспечивает также вывод текста с использованием большого количества шрифтов.

СГИ предлагает различные методы работы с цветовыми возможностями, что позволяет создавать программы, не требующие изменений при работе с дисплеями различных типов. Предусмотрена также возможность передачи данных между прикладными программами. Для взаимодействия с выполняющимися программами пользователь может использовать манипулятор «мышь» и клавиатуру. Движение «мыши» отображается на экране с помощью курсора, который может изменять свою форму и цвет при перемещении из одного окна в другое.

СГИ является мощным средством для разработки и создания современного графического интерфейса отдельных прикладных программ и при этом дает возможность отображать на одном экране графического дисплея результаты решения прикладных программ, выполняющихся на различных узлах ЛВС.

Типичные объекты интерфейса включают следующее:

* строки меню;
* строки прокрутки;
* всплывающие меню;
* спускающиеся меню;
* клавиши;
* «горячие» клавиши.

Пользовательская среда СГИ также экономит время путем выполнения различных задач одновременно. Например, можно запустить программу в одном окне, компиляцию в другом окне и выполнить командную процедуру в третьем окне.

В составе системы зашиты от несанкционированного доступа в ОС МСВС функционируют следующие механизмы, обеспечивающие разграничение доступа и аудит:

* мандатное управление доступом (*Mandatory Access Control*, *MAC*). Мандатная политика позволяет определять для информации уровни секретности и принадлежность к различным категориям;
* дискреционное управление доступом (*Discretionary Access Control*, *DAC*). Каждому файлу ОС МСВС сопоставляется *ACL*, позволяющий контролировать доступ к данному файлу с точностью до отдельного пользователя системы;
* аудит. В ОС МСВС функционирует система аудита, позволяющая протоколировать события отдельно для каждого пользователя системы. ОС МСВС позволяет осуществлять аудит открытия файлов, запуска программ, установку драйверов, вход и выход пользователей и других событий;
* привилегии. В ОС МСВС функционирует система привилегий, позволяющая отдельным пользователям выполнение различных административных действий.

**Мандатное управление доступом**. В ОС МСВС все процессы и файлы имеют мандатные метки. Мандатная метка представляет собой совокупность уровня секретности и набора категорий. Доступ процессов к файлам регулируется на основе сравнения их мандатных меток. Сетевое взаимодействие компьютеров, управляемых ОС МСВС, осуществляется также в соответствии с мандатной политикой.

В ОС МСВС поддерживается до восьми уровней секретности (от 0 до 7, 0 – самый низкий, 7 – самый высокий) и до 61 различной категории. Каждому уровню или категории назначается соответствующее имя и значение с помощью программы *macadmin*. Например, система может быть настроена на работу с тремя уровнями секретности: «Несекретно», «Секретно», «Совершенно секретно».

Единая система документации ОС МСВС позволяет получить информацию о самых разных аспектах функционирования системы. Доступ к документации осуществляется с помощью *Web*-технологии.

Система печати ОС МСВС поддерживает механизм мандатного управления доступом, осуществляет маркировку печатных листов и учет печати документов. Факт печати регистрируется в журнале учета размножения печатных документов. Для работы с этим журналом используется программа, позволяющая его просматривать, редактировать некоторые поля записей, и распечатывать.

Для генерации паролей пользователей ОС МСВС используется программа генерации паролей. Формирование паролей осуществляется на основе алгоритма генерации псевдослучайной последовательности символов.

Для осуществления удаленного мониторинга компьютеров с ОС МСВС применяется система контроля функционирования, состоящая из сервера и специальных агентов. Данная система позволяет получать информацию о состоянии процессов, дисковой подсистемы, подсистем ядра, а также контролировать работоспособность сетевых сервисов.

Для построения ЗАС на основе ОС МСВС с возможностью временной совместимости с *Windows NT* используется система терминального доступа к *Windows NT*.

Особенностью ОС МСВС являются встроенные средства защиты, удовлетворяющие требованиям РД ГТК РФ для 2-го класса защищенности СВТ от НСД. Средства защиты включают мандатное, полное дискреционное и ролевое управление доступом, а также развитые средства аудита.

Ролевая модель используется для децентрализации функций суперпользователя. Задача администрирования системы разделена на несколько частей, для выполнения которых существуют администраторы конфигурирования, безопасности и аудита.

Большое внимание уделено легкому и гибкому администрированию системы. ОС МСВС предоставляет администраторам графические интерфейсы для управления пользователями, файлами, политикой безопасности, политикой аудита, общесистемными и сетевыми настройками.

Развитые функциональные возможности и выполнение требований обеспечивают возможность использования защищенной ОС МСВС при построении отечественных ЗАС.

**Обслуживание файловой структуры**. ФС – это методы и структуры данных, которые используются операционной системой для хранения файлов на диске или его разделе.

ФС ОС МСВС соответствует типу *Ext*3 (*Ext*4) ФС обеспечивает поддержку длинных имен, символических связей, а также обеспечивает поддержку файловых систем *ISO9660*, *FAT* (*MS-DOS*), *NTFS* (*Windows NT*). В ФС ОС МСВС так же предусмотрена возможность представления имен файлов русскими буквами. Файловые системы являются основой для хранения всех данных в ОС МСВС. Программы, библиотеки, системные и пользовательские файлы – все они располагаются в файловых системах.

При работе с ОС МСВС видимое пользователем файловое пространство имеет древовидную структуру. На ее вершине находится корень. Каталоги и

файлы образуют ответвления от этого корня. Самый верхний каталог известен как корневой каталог (*root directory*).

Пользователи видят дерево каталогов и файлов как единую цельную структуру. На самом деле различные каталоги этого дерева могут соответствовать различным разделам диска, находиться на других дисках и даже на других компьютерах. Раздел диска присоединяется к дереву в каталоге, называемом точкой монтирования (*mount point*). Точка монтирования и все расположенные ниже каталоги образуют файловую систему.

Файловые системы, поддерживаемые ОС МСВС:

* *DOS FAT*;
* *FAT* (*Windows*);
* *ISO 9660 CD-ROM*;
* *NTFS* (чтение);
* *Second Extended FS* (*ext2*);
* *NFS*;
* *SMB-FS*.

**Взаимодействие с аппаратным обеспечением и условия применения**. Выбор конфигурации технических средств зависит от таких факторов как количество пользователей и типы запускаемых программ.

Рекомендуется следующая конфигурация ЭВМ для установки ОС МСВС:

* процессор *Pentium* или выше;
* *RAM 32* Мбайт или больше;
* жесткий диск 1000 Мбайт или больше;
* *CD-ROM*;
* видеокарта *SVGA*;
* звуковая карта, совместимая с *Sound Blaster*;
* сетевая карта;
* монитор;
* клавиатура и манипулятор типа «мышь».

Для разработки больших прикладных программ и сокращения времени компиляции рекомендуется ЭВМ с высокой тактовой частотой и большим объемом оперативной памяти. Разработанная программа затем может выполняться на ЭВМ, работающей под управлением ОС МСВС с меньшим быстродействием и меньшим объемом оперативной памяти.

Одним из существенных недостатков является проблема совместимости операционной системы МСВС с широким спектром различного периферийного оборудования (принтеры, сканеры, плоттеры, флешки и др.). Для обеспечения их работы необходимо написание драйверов под данную ОС.

В зависимости от выполняемых функций при эксплуатации ОС МСВС, могут быть выделены следующие категории пользователей:

* системные программисты;
* системные администраторы;
* сетевые администраторы;
* программисты;
* операторы;
* рядовые пользователи.

На этапе разработки ОС МСВС системные программисты выполняют функции по разработке и отладке системных программ, компоновке комплексов программ и в целом ОС МСВС, компоновке дистрибутивного диска и начальной установке ОС МСВС. На этапе эксплуатации ОС МСВС системные программисты выполняют только начальную установку комплекса, включая настройку звуковой карты и сетевой карты.

Системные администраторы выполняют функции по поддержке работоспособности и организации работы компьютерной системы, включая создание резервных копий некоторых файлов, установку и настройку новых программ, создание и удаление пользователей, проверку целостности файловой системы, поддержание на жестком диске необходимого свободного места, установку драйверов периферийных устройств и т. д. Системные администраторы (в одиночку или с группой поддержки) должны предоставлять каждому пользователю возможность безопасной, надежной и эффективной работы. Для этого им необходимо обладать соответствующими знаниями и уметь настраивать и поддерживать ОС МСВС таким образом, чтобы обеспечить эффективную работу системы с учетом многопользовательского режима, при котором должно корректно выполняться множество задач с различными приоритетами. ОС МСВС требует начального конфигурирования системы и внимания администраторов системы доля обеспечения корректной, бесперебойной и эффективной работы системы, и пользователей.

Сетевые администраторы выполняют функции по поддержке работоспособности компьютерной сети и сетевых служб, включая регулярную проверку функционирования сети, просмотр журнала регистрации сообщений об ошибках и нештатных ситуаций, защиту сетевых пользователей от злоумышленников и управление сетевыми приложениями.

Программисты выполняют функции по разработке и отладке прикладных программ.

Операторы выполняют функции по загрузке прикладных программ, проведению расчетов и выводу результатов на терминалы в соответствии с инструкциями пользователей, а также функции по завершению работы с прикладными программами. Кроме того, операторы должны выполнять некоторые функции, которые на время им могут передать системные и сетевые администраторы.

Рядовые пользователи (далее просто пользователи) – это пользователи, выполняющие какие-либо работы на ЭВМ (кроме работ, выполняемых пользователями первых пяти категорий, перечисленных выше) по проведению расчетов научного, экономического и инженерного характера или работ с офисными приложениями, такими как редактирование текстов, использование электронных таблиц и баз данных и др.

На объектах внедрения функции различных категорий пользователей выполняют должностные лица. У некоторых должностных лиц могут быть совмещены функции отдельных категорий пользователей. Например, могут быть совмещены функции системного программиста и администратора системы, или могут быть совмещены функции системного программиста, администратора системы и администратора сети. Как правило, в небольших системах функции системного программиста по начальной установке системы передаются администраторы системы. Чтобы стать пользователем ОС МСВС, необходимо получить имя, пароль и ряд других атрибутов.

**Основные возможности ОС МСВС**. ОС МСВС ФЛИР.80001-01 – это мобильная, многопользовательская, многозадачная операционная система, поддерживающая симметричные многопроцессорные архитектуры и работающая как в режиме командной строки, так и в режиме графического интерфейса.

ОС МСВС – это инструментальное средство для управления техническими средствами и задачами в следующих областях:

* автоматизированные системы управления производством;
* автоматизированные системы управления технологическим процессом;
* информационные системы;
* системы массового обслуживания;
* системы сбора и анализа информации;
* многопользовательские системы общего назначения.

**Сетевые возможности ОС МСВС**. ОС МСВС предоставляет различным категориям пользователей возможности по выполнению базовых сетевых функций, выполнению сетевых служб и приложений и по администрированию сетей.

Сетевые возможности ОС МСВС:

* выполнение базовых сетевых функций;
* работа в режиме клиент-сервер;
* работа в режиме распределенных вычислений;
* передача файлов;
* удаленный терминальный доступ;
* передача почтовых сообщений;
* доступ к файловым системам удаленных ЭВМ;
* доступ к сетевой информационной службе;
* доступ к службе доменных имен;
* администрирование сети.

ОС МСВС является сетевой ОС, в которой содержатся сетевые службы, реализующие функционально полный набор протоколов *TCP/IP* для организации работы в локальных (по протоколу *Ethernet*) и распределенных (по протоколам *SLIP*, *PLIP* и *РРР*) сетях.

Основные сервисы ОС МСВС. К ним относят наборы программ (сервисы) Samba, *SSH* и *DNS*.

**Сервис Samba**. Для *Unix-*подобных ОС разработано немало продуктов, реализующих сервис SMB (*Session Message Block*) − протокол разделения файлов и служб печати. *Samba* − сервис SMB, работающий в различных ОС, в том числе и в ОС МСВС, и по своим возможностям она сравнима с *Windows NT Server*.

Сервис *Samba*, установленный на сервер, предоставляет следующие основные услуги:

* сервис файлов и печати;
* аутентификацию и авторизацию;
* функции сервера имен *NetBIOS*;
* средства просмотра (*browsing*) *NetBIOS*;
* средства приема и посылки сообщений *WinPopup*.

Кроме того, сервер с ПО *Samba* может выполнять функции главного контроллера домена *NT* (*New Technology*) – домена новой технологии, члена домена *NT* или участника рабочей группы Windows, сервера имен NBNS (*NetBIOS Name Server*) – сервера имен в среде *NetBIOS*, браузеров, поддерживает шифрование трафика с помощью *SSL* (*Security Socket Level*) – алгоритма уровеня защищенных каналов и многое другое.

**Сервис *SSH***. Сервис *SSH* (*Secure SHell*) – сервис безопасности ядра *Unix*подобной операционной системы обеспечивает возможность удаленного выполнения команд и копирования файлов с аутентификацией клиента и сервера, шифрованием и сжатием передаваемых данных (пароли также шифруются).

Дополнительно обеспечивается шифрование данных *X Windows* и перенаправление любых TCP-соединений (а это уже ухудшает безопасность, т.к. позволяет делать туннели в обход *firewall*). Защищает от атак с подменой (*spoof*) IP-адресов (включая *source routing*), *DNS*-сервера и маршрутизации, от подслушивания паролей и аутентификации, от подслушивания и манипуляции данными на промежуточных машинах или локальной сети. Сервис SSH не защищает, если атакующий получил права root на вашем хосте или права к домашней директории.

Таким образом, сервис *SSH* – это набор программ для обеспечения безопасной работы в сети, который позволяет осуществлять:

* строгую аутентификацию. Исправлено большое число дыр в безопасности (например *IP*, *routing*, и *DNS spoofing*);
* усиленную безопасность. Все соединения автоматически и прозрачно шифруются;
* безопасную работу в рамках сессий. Программы автоматически устанавливают переменную *DISPLAY* на удаленном сервере, и перенаправляют любые соединения по защищенным шифрованным каналам.

Произвольные *TCP/IP* порты могут быть перенаправлены через шифрованные каналы в обоих направлениях (например, для *e-cash* транзакций).

Нет необходимости переподготовки обычных пользователей.

Позволяет никогда не доверять сети. Минимальное доверие соединению с удаленной стороны. Минимальное доверие серверам доменных имен. Чистая *RSA* аутентификация не доверяет ничему кроме личных ключей, где *RSA* (*Rivest, Shamir, Adleman*) – алгоритм двухключевого (несимметричного) шифрования сообщений с открытым ключом.

**Сервис DNS**. Сетевая распределенная база данных, известная как *Berkeley InterNet Domain server* (*BIND*), которая также известна как *Domain Name Service* (*DNS*) − служба (сервис) доменных имен предназначена для отображения имен узлов сети в их IP-адреса и обратно.

*DNS* – это распределенная база данных, которая содержит информацию о компьютерах, включенных в сеть *Internet*. Характер данных зависит от конкретной машины, но чаще всего информация включает имя машины, IP-адрес и данные для маршрутизации почты.

Как известно, помимо IP-адресов машины идентифицируются доменными именами, более легкими для запоминания и отражающими логическое структурирование сети и, часто, функциональное назначение той или иной машины. Доменное имя состоит из символьных полей, разделенных точками. Крайнее правое поле обозначает домен верхнего уровня, далее, справа налево, следуют поддомены в порядке иерархической вложенности, крайнее левое поле обозначает имя машины. Например, *os.cbit.ivan.ru* – машина os в домене cbit, который находится внутри домена ivan, который в свою очередь находится внутри домена ru.

Дерево доменных имен аналогично файловой системе *Unix*-подобных ОС, типа МСВС. Корнем дерева является домен «.» (точка). Полное – абсолютное или полностью определенное, *fully qualified domain name* – доменное имя закан-

чивается точкой, обозначающей корень доменного дерева, но часто эта завершающая точка опускается.

**Основные сетевые возможности ОС МСВС**. ОС МСВС предоставляет различным категориям пользователей возможности по выполнению базовых сетевых функций, выполнению сетевых служб и приложений, а также по администрированию сетей.

Сетевые возможности ОС МСВС:

* выполнение базовых сетевых функций;
* работа в режиме клиент/сервер;
* работа в режиме распределенных вычислений;
* передача файлов;
* удаленный терминальный доступ;
* передача почтовых сообщений;
* доступ к файловым системам удаленных ЭВМ;
* доступ к сетевой информационной службе;
* доступ к службе доменных имен;
* администрирование сети.

ОС МСВС является сетевой ОС, в которой содержатся сетевые службы, реализующие функционально полный набор протоколов *TCP/IP* для организации работы в ЛВС (по протоколу *Ethernet*) и распределенных сетях (по протоколам *SLIP*, *PLIP* и *РРР*).

Напоминаем, что в семействе протоколов *TCP/IP* существует четыре уровня логического взаимодействия и соответственно четыре уровня отдельных протоколов:

* уровень взаимодействующих процессов (приложений);
* транспортный уровень;
* сетевой уровень;
* уровень сетевого интерфейса.

Эти четыре уровня протоколов называются стеком протоколов *TCP/IP*. На уровне взаимодействующих ОС МСВС обеспечивает реализацию сетевых протоколов высокого уровня таких как: *File Transfer Protocol* (FTP) – передача и преобразование файлов, *TELNET* – удаленный терминальный доступ, *Simple Mail Transfer Protocol* (*SMTP*) – передача почтовых сообщений, *Network File System* (*NFS*) – сетевая файловая система и *Domain Name Service* (*DNS*) – преобразование имен в адреса.

Протоколы транспортного уровня обеспечивают передачу данных между процессами, выполняющимися на различных ЭВМ. К транспортным протоколам относятся *Transmission Control Protocol* (*TCP*) – протокол управления передачей и *User Datagram Protocol* (*UDP*) – протокол пользовательских дейтаграмм. Протокол *TCP* ориентирован на соединения, которые обеспечивают

дуплексную передачу данных между пользовательскими процессами. Протокол *UDP* работает без установления соединений, и для контроля за полнотой и порядком передачи данных необходимы дополнительные процедуры пользователя, такие как прикладное квитирование, тайм-ауты и счетчики номеров сообщений.

Сетевой уровень составляют протоколы, обеспечивающие передачу данных между ЭВМ, подключенными к различным сетям. Основной функцией сетевого уровня является маршрутизация пакетов данных. ЭВМ, реализующая функцию маршрутизации, называется шлюзом (*gateway*) или маршрутизатором (*router*). Шлюз имеет несколько сетевых интерфейсов, подключенных к различным физическим сетям Основное назначение шлюза – это выбор маршрута передачи данных из одного сетевого интерфейса в другой. К сетевому уровню относятся *Internet Protocol* (*IP*) -протокол *Internet* и *Internet Control Message Protocol* (*ICMP*) − протокол управления сообщениями.

Уровень сетевого интерфейса обеспечивается протоколами доступа к физической сети между ЭВМ, подключенными к одному сетевому сегменту, например к сегменту *Ethernet* или каналу точка-точка. К этому уровню относятся протоколы *Ethernet*, *SLIP*, *PLIP*, *PPP*.

**Выполнение базовых сетевых функций**. ОС МСВС имеет ряд сетевых возможностей и большинство базовых сетевых функций, таких как, работа с файлами, работа с принтерами, создание резервных копий и другие могут быть реализованы посредством компьютерной сети. Базовые сетевые функции выполняются пользователем режиме командной строки. При работе в сети возможно разделение файлов. Для этого все файлы коллективного доступа, с которыми работают пользователи, размещаются на файл-сервере. Пользователи могут копировать, переименовывать, перемещать и удалять файлы, а также выполнять некоторые другие операции. Для выполнения каких-либо действий необходимо войти на файл-сервер.

При работе в сети возможно разделение не только файлов, но и принтеров Пользователи могут иметь доступ к одному и тому же принтеру через сеть. Кроме того, пользователи могут управлять выводом результатов на принтер.

**Работа в режиме клиент/сервер**. ОС МСВС позволяет пользователям организовать работу в режиме клиент/сервер. Технология клиент/сервер дает реальные преимущества пользователям и становится преобладающим способом обработки данных.

Архитектура клиент/сервер по сравнению с традиционной однокомпьютерной средой позволяет получить систему обработки информации с улучшенным соотношением цена/производительность, допускающую простое расширение, наращивание и адаптацию к изменяющимся требованиям.

**Работа в режиме распределенных вычислений**. Распределенные вычисления – это более широкое понятие, чем понятие клиент/сервер. Технология распределенных вычислений предполагает, что задания, поступающие от клиентских ЭВМ, обрабатываются несколькими ЭВМ Распределенные вычисления рассматриваются как архитектура клиент/сервер с участием нескольких серверов. Применительно к распределенной обработке термин сервер означает просто программу, отвечающую на запросы и выполняющую необходимые действия по запросу клиента.

Поскольку распределенные вычисления – это один из видов систем клиент/сервер, то пользователи получают здесь в основном те же преимущества, что и в технологии клиент/сервер, например, увеличение общей пропускной способности. Кроме того, интеграция дискретных сетевых компонентов и обеспечение их функционирования как единого целого способствует увеличению эффективности и снижению издержек.

**Работа с сетевыми приложениями**. Сетевые прикладные программы коллективно используются пользователями и могут быть как встроенными в ОС МСВС (например, прикладные программы службы передачи файлов, службы удаленного терминального доступа и др.), так и поставляемыми в составе различных прикладных пакетов (например, СУБД «Линтер-ВС!» ФЛИР 80002-01). Кроме возможности выполнения базовых сетевых функций ОС МСВС предоставляет пользователям возможность работы с сетевыми службами, такими как передача файлов, удаленный терминальный доступ, передача почтовых сообщений, сетевая файловая система, сетевая информационная система и система доменных имен. Чтобы воспользоваться этими службами, необходимо произвести установку и конфигурирование этих служб. Работу сетевых служб обеспечивают протоколы *FTP*, *TELNET*, *SMTP*, *NFS* и др.

**Служба передачи файлов**. В ОС МСВС служба передачи файлов (служба *FTP*) обеспечивается с помощью интерактивной программы *ftp*, вызываемой на клиентской стороне, и сервера *ftpd*, который запускается на ЭВМ, выполняющей функцию сервера службы *FTP*. Обе программы реализуют протокол передачи файлов *FTP*. Для копирования файлов клиенту необходимо знание имени и пароля пользователя, которому принадлежат файлы на сервере службы *FTP*.

Вызов программы *ftp* осуществляется командой *ftp* имя сервера. Интерактивный доступ к серверу службы *FTP* обеспечивается внутренними командами программы *ftp*:

* + *open, user, close* − связь с удаленной ЭВМ;
  + *cd, dir, pwd* − работа с каталогами в *ftp* сервере;
  + *get, put* − получение и передача файлов;
  + *ascii, binary, status* − установка параметров передачи.

Выход из программы *ftp* осуществляется по команде *bye*.

Служба *FTP* удобна для организации сервера загрузки и сервера архивов в корпоративных сетях.

**Служба удаленного терминального доступа**. ОС МСВС предоставляет пользователям возможность работы в сети по протоколу *TCP/IP* посредством эмуляции терминала.

Служба удаленного терминального доступа (служба *TELNET*) обеспечивается с помощью интерактивной программы *telnet*, вызываемой на клиентской стороне, и сервера *telnetd* на удаленной ЭВМ, к которой необходимо иметь командный доступ. Обе программы реализуют протокол *TELNET*. Служба *TELNET* применяется, когда необходимо зарегистрироваться на удаленной ЭВМ и выполнить на ней некоторую работу, в частности, выполнить программу или выполнить администрирование. Для выполнения регистрации клиенту необходимо знание имени и пароля пользователя на удаленной ЭВМ.

Вызов службы *TELNET* осуществляется следующей командой *telnet* [система[порт]], где в качестве параметра «система» задается *IP*-адрес удаленной ЭВМ, а параметр «порт» задает номер порта *telnetd*.

После установления соединения все, что набирается на клавиатуре клиентской ЭВМ, передается на удаленную ЭВМ, а все то, что выводится на удаленной ЭВМ появляется на экране клиентской ЭВМ. Служба *TELNET* не предназначена для передачи файлов. Для передачи файлов необходимо выйти из службы *TELNET* и воспользоваться службами *FTP* или *NFS*.

Выход из службы *TELNET* осуществляется при закрытии сессии по команде *logout*.

**Служба передачи почтовых сообщений**. В ОС МСВС служба передачи почтовых сообщений обеспечивается с помощью трех составляющих: пользовательского агента *mail*, транспортного агента *postfix* и доставочного агента *mail*. Пользовательский агент mail позволяет пользователям читать и составлять почтовые сообщения. Транспортный агент *postfix* отвечает за пересылку почтовых сообщений с одной ЭВМ на другую. Доставочный агент *mail* помещает сообщения в почтовые ящики пользователей-получателей, что позволяет пользователям выводить список полученной почты и читать почту.

Транспортный агент *postfix* обеспечивает прием почты от пользовательского агента *mail*, интерпретирует адреса получателей и передает почту на соответствующие ЭВМ для последующей доставки. Помимо этого, транспортный агент принимает приходящую почту от других транспортных агентов и оставляет ее в общем почтовом каталоге, из которого почта выбирается доставочным агентом для передачи конечным пользователям. Транспортный агент *postfix* поддерживает протокол передачи почтовых сообщений *SMTP*.

**Сетевая файловая система *NFS***. Одна из наиболее полезных функций, которая может быть реализована с помощью компьютерной сети, это разделение файлов через сетевую файловую систему. В ОС МСВС используется сетевая файловая система *NFS*, называемая также службой *NFS*, которая разработана корпорацией *Sun Microsystems*.

Сетевая служба *NFS* позволяет клиентским ЭВМ совместно использовать файловые системы серверных ЭВМ. Сетевая служба *NFS* прозрачна для пользователей и даже при отказе сервера информация для клиентов не пропадает. Клиенты ожидают, когда сервер вновь начнет функционировать, а затем продолжают работу. Доступ к файловым системам удаленных ЭВМ обеспечивается с помощью нескольких программ на стороне сервера и на стороне клиента. На стороне сервера существуют три программы, используемые для обеспечения службы *NFS: rpc.portmapper, rpc.nfsd и rpc.mountd*.

Программа *rpc.portmapper* не обеспечивает напрямую службы *NFS*, a перенаправляет обращения, сделанные с других ЭВМ к службам *NFS*.

Программа *rpc.nfsd* переводит запросы к службе *NFS* в действительные запросы к локальной файловой системе.

Программа *rpc.mountd* запрашивается для монтирования и размонтирования файловых систем.

На стороне сервера выполняется экспортирование файловой системы. Это означает, что определенные поддеревья, задаваемые каталогами, объявляются доступными для клиентских ЭВМ. Информация об экспортированных файловых системах заносится в файл /*etc/export*, в котором указывается, какие каталоги доступны для указанных клиентских ЭВМ и какими правами доступа обладают клиентские ЭВМ при выполнении операций на сервере. Запросы монтирования поступают от клиентских ЭВМ к серверу монтирования *mountd*. Сервер *mountd* проверяет правильность клиентского запроса на монтирование и разрешает серверу службы *NFS* – *nfsd* обслуживать запросы клиента, выполнившего монтирование. Клиенту разрешается выполнять различные операции с экспортированной файловой системой в пределах полномочий клиента. Для получения хорошего качества обслуживания клиентов на сервере службы *NFS* одновременно запускается несколько копий процесса *nfsd*.

На стороне клиента команда mount модифицирована таким образом, чтобы она могла понимать запись «имя\_машины»: «каталог», где «имя\_машины» – имя сервера *NFS*, «каталог» – экспортированный каталог сервера службы *NFS.* Для удаленных файловых систем, которые являются частью постоянной конфигурации клиента, записи о монтируемых файловых системах службы *NFS* должны быть перечислены в файле /*etc/fstab* для автомонтирования во время начальной загрузки клиентской ЭВМ. На клиентской стороне предусмотрена сервисная программа *nfsiod*, которая выполняет базовые операции блочного кэширования файловых систем для получения хорошего времени доступа к серверу службы *NFS*.

При работе с сетевой файловой системой любые операции над файлами, производимые на локальной ЭВМ, передаются через сеть на удаленную ЭВМ.

Вызов и останов службы *NFS* осуществляется пользователем с полномочиями корневого пользователя командами /*etc/rc.d/init.d/nfs start* и /*etc/rc.d/init.d/nfs stop.*

**Настройка сервера *NFS***. После установки сервера *NFS* его необходимо настроить. Для этого отредактировать файл /*etc/exports*. Допустим если необходимо, чтобы файловая система /*mn/eris/local*, которая находится на машине eris была доступна для машины abc, то в файл /*etc/exports* на машине *eris* необходимо поместить следующие строки:

/*mn/eris/local abc(rw).*

Вышеуказанные строки дают машине abc право на чтение и запись в каталоге /*mn/eris/local*, вместо rw, можно написать ro, что будет означать право только на чтение.

Если пользователь отредактировал файл /*etc/exports*, то он должен быть уверен, что *nfsd* и *mountd* знают о том, что файл изменен. Традиционный способ проверить это – запустить программу *exports*. А для того чтобы проверить, что *nfsd* и *mountd* запущены правильно необходимо дать команду *rpcinfo -p* (в результате мы должны увидеть, что *nfsd* и *mountd* запущены).

**Настройка клиента *NFS***. Монтирование */mn/eris/local* с машины *eris* делается с помощью команды

*mount –o rsize=1024,wsize=1024, eris:/mn/eris/local /mnt/xxx,*

где *ххх* – имя фашего каталога в директории /mnt.

Если вместо монтирования файловой системы команда mount выдаст сообщение об ошибке, то файл */etc/exports* является неправильным или пользователь забыл запустить программу exports после редактирования.

Чтобы прекратить пользоваться файловой системой выполните:

*mount /mnt/xxx.*

Для того чтобы выполнялось автоматическое монтирование файловой системы *nfs* при загрузке, необходимо отредактировать файл */etc/fstab*, для нашего примера:

…

*eris:/mn/eris/local /mnt/xxx nfs rsize=1024,wsize=1024 0 0.*

…

Существуют дополнительные опции, которые управляют способом, которым клиент NFS отрабатывает прекращение работы сервера или отключение сети. Так программа *hard* при разрыве соединения с сервером приостанавливает свое выполнение, когда сервер будет запущен заново, то программа продолжит работу с прерванного места. Процесс не может быть прерван или убит до тех пор, пока пользователь явно не укажет опцию *intr*.

Тогда запись в файле /*etc/fstab* будет выглядеть следующим образом:

*eris:/mn/eris/local /mnt/xxx nfs rsize=1024,wsize=1024,hard,intr 0 0*.

**Оптимизация *NFS***. Обычно, если не заданы опции *rsize* и *wsize*, то *NFS* будет читать и писать блоками по 4096 или 8192 байт. Некоторые драйверы сетевых карт не могут оптимально обрабатывать такие большие блоки. В этом случае надо поэкспериментировать и найти значения *rsize* и *wsize*, которые работают так быстро на сколько это возможно. Эти значения должны быть кратными 1024 и не больше чем 16384 байт, поскольку это максимальный размер блока данных в NFS. Опция noexec запрещает выполнение файлов на смонтированной файловой системе.

**Сетевая информационная служба**. *Network Information Service* (*NIS*) − сетевая информационная служба – это система на основе базы данных клиент/сервер. Чаще всего она применяется для совместного использования паролей и групп файлов в сети.

Служба *NIS* разработана корпорацией *Sun Microsystems* как часть ОС SunOS Служба *NIS* была модернизирована и названа *NIS+*. Служба *NIS+* совершеннее, чем *NIS*, особенно в области безопасности.

ОС МСВС может работать либо как сервер службы *NIS+*, либо как клиент службы *NIS+*.

Когда пользователь конфигурирует свою сеть, он обнаруживает, что некоторые из его конфигурационных файлов не являются специальными для хоста, но требуют частого обновления. Служба *NIS+* позволяет пользователю установить мастер-сервер, где хранятся эти файлы, например, файлы */etc/passwd* и */etc/group*, и далее конфигурировать каждую ЭВМ в своей сети как клиента этого сервера. Всякий раз, когда клиенту нужно извлечь элемент из файла

*/etc/passwd*, он вместо этого взаимодействует с сервером службы *NIS+.*

Чтобы файл при помощи службы *NIS+* стал общедоступным, необходимо учесть два предварительных условия. Первое: файл должен быть организован в виде таблицы, по крайней мере, с одним элементом, который является единственным в своем роде во всем файле. Второе: файл в своей необработанной форме должен быть обычным текстовым файлом.

При соответствии приведенным критериям эти файлы конвертируются в файлы формата DBM (обычный формат базы данных, допускающий быстрый поиск).

Оригинальные текстовые файлы вместе с файлами *DBM*, созданными из них, хранятся на мастер-сервере службы *NIS+*.

Если серверы и клиенты службы *NIS+* хотят установить связь между собой, то они должны находиться в одном домене. Необходимо заметить, что домен службы *NIS*+ – это не то же самое, что домен службы *DNS*, хотя для них можно использовать одинаковое имя.

Клиенты и серверы объединяются в домен, следовательно, в данный момент клиент может принадлежать только одному домену. Как только клиент присоединяется к домену, он посылает широковещательное послание, чтобы найти сервер службы *NIS*+ данного домена.

Существует два вида серверов службы *NIS*+: мастер-серверы (*master servers*) и подчиненные серверы (*slave servers*). Мастер-серверы службы *NIS*+ содержат текстовые файлы, используемые для генерации файлов *DBM*, и любые изменения в базе данных должны быть проведены через эти файлы.

Подчиненные серверы службы *NIS*+ разработаны для поддержки мастерсерверов путем передачи части нагрузки на них с мастер-серверов. Когда на мастер-сервере обновляется файл, инициируется программа рассылки, и подчиненный сервер службы *NIS*+ получает обновленные копии *DBM*-файлов.

**Служба доменных имен**. Сетевая распределенная база данных, известная как *Berkeley InterNet Domain server* (*BIND*) или *DNS* служба доменных имен. Служба *DNS* предоставляет эффективный и относительно простой механизм отображения имен узлов в IP-адреса и обратно.

Службу *DNS* достаточно сложно настроить, но когда она настроена, поддерживать ее в таком состоянии довольно просто.

Служба *DNS* предоставляет механизм преобразования имен узлов, сетей и почтовых псевдонимов в IP-адреса и наоборот. Это делается путем разбиения IP-адресов и пространства имен на различные логические группы. Каждая логическая группа содержит достоверную информацию о своих ЭВМ.

Систему *DNS* можно разделить на три части:

* пространство доменных имен;
* серверы имен;
* библиотека разрешения имен.

Пространство доменных имен – это спецификация структуры дерева, определяющая набор узлов и предоставляющая информацию о них. Каждый узел в дереве имеет базу данных, хранящую информацию обо всех подчиненных ему узлах Запросы извлекают информацию из базы данных.

Серверы имен – это программы, которые поддерживают актуальность данных, хранящихся в пространстве доменных имен. Каждый сервер имен хранит полную информацию о подчиненных пространствах имен, а также кэширует информацию о других пространствах.

Сервер имен содержит полную информацию о подчиненной области. Эта информация делится на зоны, каждая из которых для обеспечения надежности может храниться на нескольких серверах имен. Каждый сервер имен знает о других серверах, отвечающих за свои зоны. Если приходит запрос по зоне, за которую отвечает данный сервер имен, он возвращает необходимую информацию. Если же запрос приходит по другой зоне, сервер имен обращается к тому серверу имен, который отвечает за соответствующую зону.

Библиотека разрешения имен – это программы или библиотеки функций, которые извлекают информацию из серверов имен в соответствии с запросом.

Чтобы использовать службу *DNS*, следует настроить библиотеку разрешения имен на отдельной ЭВМ. Это необходимо сделать для применения службы DNS даже тогда, когда не предполагается использовать локальный сервер имен.

**Служба *Samba***. *Служба Session Message Block* (*SMB*), известная также как Samba, предоставляет следующие возможности:

* разделение файловых систем ОС МСВС ОС семейства *Windows*;
* совместное использование принтеров, подключенных к системе ОС МСВС, операционными системами семейства *Windows*.

*Samba* представляет собой протокол, используемый *Microsoft* для разделения файлов и служб печати. Протокол *Samba* реализуется комплексом программ, который состоит из нескольких компонентов:

* сервисная служба *smbd*;
* сервисная служба *nmbd*;
* сервисная служба *smbclient*;
* утилита *testparm*;
* утилита *smbstatus*.

Сервисная служба *smbd* обеспечивает работу службы печати и разделения файлов для клиентов *Samba*, таких как *Windows for Workgroups*, *Windows NT* и *LanManager*. Конфигурационные параметры сервисной службы *smbd* описываются в файле *smb.conf*.

Сервисная служба *nmbd* обеспечивает работу службы имен NetBIOS, а также может использоваться для запроса других сервисных служб имен.

Сервисная служба *smbclient* реализует простейший *FTP*-подобный клиент, который может использоваться для доступа к другим серверам Samba и для печати на принтерах, подключенных к серверам *Samba*.

Утилита *testparm* позволяет протестировать конфигурационный файл *smb.conf*.

Утилита smbstatus сообщает, кто в настоящее время пользуется сервером smbd.

Главный конфигурационный файл службы Samba называется /*etc/smb.conf*. Он состоит из нескольких основных разделов:

[*global*] – в нем описаны параметры, управляющие сервером в целом, а также значения параметров по умолчанию для других разделов;

[*homes*] – позволяет пользователям подключаться к рабочим каталогам пользователей без их явного описания;

[*printers*] – используется для предоставления доступа к принтерам, определенным в файле /*etc/printcap*;

[*xxx*] – в этом разделе создается разделяемый каталог с именем *ххх.*

После создания конфигурационного файла необходимо протестировать его корректность при помощи утилиты testparm. Программа *testparm* проверяет наличие в файле /*etc/smb.conf* внутренних противоречий и несоответствий. Если программа не обнаружила ошибок, то можно надеяться, что при запуске *smbd* конфигурационный файл будет загружен без проблем.

Замечание . Применение *testparm* не дает гарантии, что все сервисы и ресурсы, описанные, в конфигурации *Samba* доступны и будут корректно работать.

В случае обнаружения ошибок о них будет предоставлена полная информация. Если внутренних противоречий и несоответствий обнаружено не будет, то после нажатия еще раз <*Enter*> testparm протестирует каждый раздел, определенный в конфигурационном файле.

Пример:

[*global*] [*homes*]

*Workgroup = QQQ comment = Home Directories server string = Samba Server path = /tmp*

*host allow = 192.168.25.1 192.168.25.127 public = yes log file = /var/log/samba/log.%1 read only = no max log size = 50*

*security = share*

*socket options* = *TCP\_NODELAY*.

Важно знать, что означают следующие строки (с остальными опциями можно ознакомиться в справочном файле по *smb.conf* объемом 219 страниц):

*quest ok = yes* – ресурс доступен всем пользователям без пароля;

*host allow* = 192.168.25.0/255.255.255.0 – список *IP* адресов, кому разрешен доступ к ресурсам *Samba*;

*security* = (*domain* (*user* (*share*) – безопасность ресурса обеспечивается на уровне домена, пользователя или отсутствует;

*socket options* = *TCP\_NODELAY* – опция для повышения производительности доступа к ресурсу.

**Запуск сервера Samba**. Сервер *Samba* состоит из двух сервисных программ – *smbd* и *nmbd*. *Smbd* обеспечивает работу службы разделения файлов и принтеров, a *nmbd* поддерживает имена *NetBIOS*. Сервер запускается либо из инициализирующих сценариев, либо из *inetd* в качестве системного сервиса. Если сервер *Samba* запускается из сценариев инициализации, то можно воспользоваться для запуска и остановки работы сервера следующей командой

/*etc/rc.d/init.d/smb start или stop*.

2. Операционные системы реального времени

**Основные понятия, назначение ОС реального времени**. Система реального времени (СРВ) – это система, правильность функционирования которой зависит не только от логической корректности вычислений, но и от времени, за которое эти вычисления производятся.

Система работает в реальном времени, если ее быстродействие адекватно скорости протекания физических процессов на объектах контроля или управления. Широкое применение системы реального времени получили в различных автоматизированных системах управления. Примерами систем реального времени являются системы управления атомными электростанциями или какими-нибудь технологическими процессами, медицинского мониторинга, управления вооружением, космической навигации, разведки, управления лабораторными экспериментами, робототехника, телеметрические системы управления, системы сигнализации. Быстродействие этих систем должно быть адекватно скорости работы объекта, система должна «успевать» управлять объектом.

Основные требования к СРВ являются:

* возможность параллельного выполнения нескольких задач;
* предсказуемость;
* важно максимальное (не среднее) время отклика на событие;
* особые требования в вопросах безопасности;
* возможность безотказной работы в течении длительного времени.
* Различают два класса СРВ:
* специализированная – система, где конкретные временные требования заранее определены;
* универсальная – система должна выполнять произвольные (заранее не определенные) временные задачи, без применения специальных средств (техники и программ).

Рассмотрим операционную систему, работающую в режиме реального времени на примере операционной системы *QNX*.

**Сетевая ОС реального времени *QNX***. Операционная система *QNX* является мощной операционной системой, позволяющей проектировать сложные программные системы, работающие в реальном времени как на одном единственном компьютере, так и в локальной вычислительной сети. Встроенные средства операционной системы *QNX* обеспечивают поддержку многозадачного режима на одном компьютере и взаимодействие параллельно выполняемых задач на разных компьютерах, работающих в среде локальной вычислительной сети. Основным языком программирования в системе является язык С. Основная операционная среда соответствует стандартам *POSIX*-интерфейса. Это позволяет с небольшими доработками перенести необходимое накопленное программное обеспечение в среду операционной системы *QNX* для организации их работы в среде распределенной обработки.

ОС *QNX* является сетевой, мультизадачной, многопользовательской (многотерминальной) и масштабируемой. С точки зрения пользовательского интерфейса и API она очень похожа на *Unix*. Она была разработана канадской фирмой *QNX Software Systems Limited* в 1989 году по заказу Министерства обороны США. Причем эта система построена на совершенно других архитектурных принципах, отличных от принципов, использованных при создании ОС *Unix*.

*QNX* была первой коммерческой ОС, построенной на принципах микроядра и обмена сообщениями. Система реализована в виде совокупности независимых (но взаимодействующих через обмен сообщениями) процессов различного уровня (менеджеры и драйверы), каждый из которых реализует определенный вид сервиса.

**Архитектура системы *QNX***. *QNX* – это ОС реального времени, позволяющая эффективно организовать распределенные вычисления. В системе реализована концепция связи между задачами на основе сообщений, посылаемых от одной задачи к другой, причем задачи эти могут находиться как на одном и том же компьютере, так и на удаленных, но связанных локальной вычислительной сетью. Реальное время и концепция связи между процессами в виде сообщений оказывают решающее влияние на разрабатываемое для *QNX* программное обеспечение и на программиста, стремящегося с максимальной выгодой использовать преимущества системы.

Микроядро имеет объем в несколько десятков килобайт (в одной из версий

–10 Кбайт, в другой – менее 32 Кбайт), то есть это одно из самых маленьких ядер среди всех существующих операционных систем.

В этом объеме помещаются:

* механизм передачи сообщений между процессами (*IPC*);
* редиректор3 прерываний;
* блок планирования выполнения задач;
* сетевой интерфейс для перенаправления сообщений (менеджер *Net*). Механизм передачи межпроцессных сообщений(*IPC*) занимается пересыл-

кой сообщений между процессами и является одной из важнейших частей операционной системы, так как все общение между процессами, в том числе и системными, происходит через сообщения.

Сообщение в *QNX* – это последовательность байтов произвольной длины (0–65535 байтов) произвольного формата. Протокол обмена сообщениями выглядит таким образом. Например, задача блокируется для ожидания сообщения. Другая же задача посылает первой сообщение и при этом блокируется сама, ожидая ответа. Первая задача разблокируется, обрабатывает сообщение и отвечает, разблокируя при этом вторую задачу. Сообщения и ответы, пересылаемые между процессами при их взаимодействии, находятся в теле отправляющего их процесса до того момента, когда они могут быть приняты. Это значит, что, с одной стороны, уменьшается вероятности повреждения сообщения в процессе передачи, а с другой – уменьшается объем оперативной памяти, необходимый для работы ядра.

Кроме того, уменьшается число пересылок из памяти в память, что разгружает процессор. Особенностью процесса передачи сообщений является то, что в сети, состоящей из нескольких компьютеров под управлением *QNX*, сообщения могут прозрачно передаваться процессам, выполняющимся на любом из узлов.

Редиректор прерываний является частью ядра и занимается перенаправлением аппаратных прерываний в связанные с ними процессы. Благодаря такому подходу возникает один побочный эффект – с аппаратной частью компьютера работает ядро, оно перенаправляет прерывания процессам – обработчикам прерываний. Обработчики прерываний обычно встроены в процессы, хотя каждый из них исполняется асинхронно с процессом, в который он встроен. Обработчик исполняется в контексте процесса и имеет доступ ко всем глобальным переменным процесса. При работе обработчика прерываний прерывания разрешены, но обработчик приостанавливается только в том случае, если произошло более высокоприоритетное прерывание. Если это позволяется аппаратной частью, к одному прерыванию может быть подключено несколько обработчиков, и каждый из них получит управление при возникновении прерывания.

Этот механизм позволяет пользователю избежать работы с аппаратным обеспечением напрямую и тем самым избегать конфликтов между различными процессами, работающими с одним и тем же устройством. Для обработки сигналов от внешних устройств чрезвычайно важно минимизировать время между возникновением события и началом непосредственной его обработки. Этот фактор существен в любой области применения, от работы терминальных устройств до возможности обработки высокочастотных сигналов.

Блок планирования выполнения задач (диспетчер задач) занимается обеспечением многозадачности. В этой части ОС *QNX* предоставляет разработчику огромный простор для выбора той методики выделения ресурсов процессора задаче, которая обеспечит наиболее подходящие условия для критических приложений или обеспечит такие условия для некритических приложений, что они выполнятся за разумное время, не мешая работе критических приложений.

К выполнению своих функций, как диспетчера, ядро приступает в следующих случаях:

* какой-либо процесс вышел из блокированного состояния;
* истек квант времени для процесса, владеющего *CPU*;
* работающий процесс прерван каким-либо событием.

Диспетчер выбирает процесс для запуска среди неблокированных процессов в порядке значений их приоритетов, которые располагаются в диапазоне от 0 (наименьший) до 31 (наибольший). Обслуживание каждого из процессов зависит от метода диспетчеризации, с которым он работает (уровень приоритета и метод диспетчеризации могут динамически меняться во время работы).

В *QNX* существуют три метода диспетчеризации: *FIFO* (первым пришел – первым обслужен), *round–robin* (процессу выделяется определенный квант времени для работы) и адаптивный, который является наиболее используемым.

*Первый* наиболее близок к кооперативной многозадачности. То есть процесс выполняется до тех пор, пока он не перейдет в состояние ожидания сообщения, состояние ожидания ответа на сообщение или не отдаст управление ядру. При переходе в одно из таких состояний процесс помещается последним в очередь процессов с таким же уровнем приоритета, а управление передается процессу с наибольшим приоритетом.

Во *втором* варианте все происходит так же, как и в предыдущем, с той разницей, что период, в течение которого процесс может работать без перерыва, ограничивается неким квантом времени. Процесс, работающий с адаптивным методом, в *QNX* ведет себя следующим образом:

* когда процесс полностью использовал выделенный ему квант времени, его приоритет снижается на 1, если в системе есть процессы с тем же уровнем приоритета, готовые к исполнению;
* если процесс с пониженным приоритетом остается не обслуженным в течение секунды, его приоритет увеличивается на 1;
* если процесс блокируется, ему возвращается оригинальное значение приоритета.

По умолчанию процессы запускаются в режиме адаптивной многозадачности. В этом же режиме работают все системные утилиты *QNX*. Процессы, работающие в разных режимах многозадачности, могут одновременно находиться в памяти и исполняться. Важный элемент реализации многозадачности – это приоритет процесса. Обычно приоритет процесса устанавливается при его запуске. Но есть дополнительная возможность, называемая «вызываемый клиентом приоритет». Как правило, она реализуется для серверных процессов (исполняющих запросы на какое-либо обслуживание). При этом приоритет процесса-сервера устанавливается только на время обработки запроса и становится равным приоритету процесса клиента.

Сетевой интерфейс в системе *QNX* является неотъемлемой частью ядра. Он, конечно, взаимодействует с сетевым адаптером через сетевой драйвер, но базовые сетевые сервисы реализованы на уровне ядра. При этом передача сообщения процессу, находящемуся на другом компьютере, ничем не отличается с точки зрения приложения от передачи сообщения процессу, выполняющемуся на том же компьютере. Благодаря такой организации сеть превращается в однородную вычислительную среду. При этом для большинства приложений не имеет значения, с какого компьютера они были запущены, на каком исполняются и куда поступают результаты их работы. Такое решение принципиально отличает *QNX* от остальных ОС, которые тоже имеют все необходимые средства для работы в сети, и делает системы, работающие под ее управлением, по-настоящему распределенными.

Все сервисы *QNX*, не реализованные непосредственно в ядре, работают как стандартные процессы в полном соответствии с основными концепциями микроядерной архитектуры. С точки зрения операционной системы системные процессы ничем не отличаются от всех остальных. Как, впрочем, и драйверы устройств.

Единственное, что нужно сделать, написав новый драйвер устройства в *QNX*, чтобы он стал частью операционной системы, – это изменить конфигурационный файл системы так, чтобы драйвер запускался при загрузке.

*QNX* является сетевой операционной системой и позволяет организовать эффективные распределенные вычисления. Для организации сети на каждой машине, называемой узлом, помимо ядра и менеджера процессов должен быть запущен уже упомянутый выше менеджер *Net*. Менеджер *Net* не зависит от аппаратной реализации сети. Данная аппаратная независимость обеспечивается за счет использования сетевых драйверов. В *QNX* имеются драйверы для различных сетей, например *Ethernet, Arcnet, Token Ring*. Кроме этого, имеется возможность организации сети через последовательный канал или модем.

В *QNX* реализовано встроенное сетевое взаимодействие «точка-точка». Например, находясь на машине *А*, можно скопировать файл с гибкого диска, подключенного к машине В, на жесткий диск, подключенный к машине *С*.

По существу, сеть из машин *QNX* действует как один мощный компьютер. Любые ресурсы (модемы, диски, принтеры) могут быть добавлены к системе простым их подключением к любой машине в сети. *QNX* поддерживает одновременную работу в сетях *Ethernet*, *Arcnet*, *Serial* и *Token Ring* и обеспечивает более чем один-единственный путь для коммуникации, а также балансировку нагрузки в сетях. Если кабель или сетевая плата выходит из строя таким образом, что связь через эту сеть прекращается, то система будет автоматически перенаправлять данные через другую сеть. Это происходит в режиме «on-line», предоставляя пользователю автоматическую сетевую избыточность и увеличивая скорость коммуникаций во всей системе.

Каждому узлу в сети соответствует уникальный целочисленный идентификатор – логический номер узла. Любой поток в сети *QNX* имеет прозрачный доступ (при наличии достаточных привилегий) ко всем ресурсам сети, то же самое относится и к взаимодействию потоков. Для взаимодействия потоков, находящихся на разных узлах сети, используются те же самые вызовы ядра, что и для потоков, выполняемых на одном узле. В том случае, если потоки находятся на разных узлах сети, ядро переадресует запрос менеджеру сети.

Для организации обмена в сети используется надежный и эффективный протокол транспортного уровня *FLEET*. Каждый из узлов может принадлежать одновременно нескольким *QNX*-сетям. В том случае если сетевое взаимодействие может быть реализовано несколькими путями, для передачи выбирается незагруженная и более скоростная сеть.

Сетевое взаимодействие является узким местом в большинстве операционных систем и обычно создает значительные проблемы для систем реального времени. Для того чтобы обойти это препятствие, разработчики *QNX* создали собственную специальную сетевую технологию *FLEET* и соответствующий протокол транспортного уровня *FTL* (*FLEET transport layer*). Этот протокол не базируется ни на одном из распространенных сетевых протоколов типа IPX или *NetBios* и обладает рядом качеств, которые делают его уникальным. Эта уникальная технология *FLEET* [*Fault-tolerance* (отказоустойчивая), *Load-balancing* (регулирующая нагрузку), *Efficient* (эффективная), *Extensible* (расширяемая), *Transparent* (прозрачная)] дает новое качество сетевой обработки данных.

Благодаря этой технологии сеть компьютеров с *QNX* фактически можно представлять, как один виртуальный суперкомпьютер. Все ресурсы любого из узлов сети автоматически доступны другим. Это значит, что любая программа может быть запущена на любом узле, при этом ее входные и выходные потоки могут быть направлены на любое устройство на любых других узлах.

Например, утилита *make* в *QNX* автоматически распараллеливает компиляцию пакетов из нескольких модулей на все доступные узлы сети, а затем собирает исполняемый модуль по мере завершения компиляции на узлах. Специальный драйвер, входящий в комплект поставки, позволяет использовать для сетевого взаимодействия любое устройство, с которым может быть ассоциирован файловый дескриптор, например последовательный порт, что открывает возможности для создания глобальных сетей.

Достигаются все эти удобства за счет того, что поддержка сети частично обеспечивается и микроядром (специальный код в его составе позволяет *QNX* фактически объединять все микроядра в сети в одно ядро). Разумеется, за такие возможности приходится платить тем, что мы не можем получить драйвер для какой–либо сетевой платы от кого-либо еще, кроме фирмы *QSSL*, то есть использоваться может только то оборудование, которое уже поддерживается. Однако ассортимент такого оборудования достаточно широк и периодически пополняется новейшими устройствами.

Когда ядро получает запрос на передачу данных процессу, находящемуся на удаленном узле, он переадресовывает этот запрос менеджеру *Net*, в подчинении которого находятся драйверы всех сетевых карт. Имея перед собой полную картину состояния всего сетевого оборудования, менеджер *Net* может отслеживать состояние каждой сети и динамически перераспределять нагрузку между ними. В случае, когда одна из сетей выходит из строя, информационный поток автоматически перенаправляется в другую доступную сеть, что очень важно при построении высоконадежных систем. Кроме поддержки своего собственного протокола, *Net* обеспечивает передачу пакетов *TCP/IP*, *SMB* и многих других, используя то же сетевое оборудование. Производительность *QNX* в сети приближается к производительности аппаратного обеспечения.

При проектировании системы реального времени, как правило, необходимо обеспечить одновременное выполнение нескольких приложений. В *QNX/Neutrmo* параллельность выполнения достигается за счет использования потоковой модели POSIX, в которой процессы в системе представляются в виде совокупности потоков. Поток является минимальной единицей выполнения и диспетчеризации для ядра *Neutrino*, процесс определяет адресное пространство для потоков. Каждый процесс состоит минимум из одного потока. *QNX* представляет богатый набор функций для синхронизации потоков. В отличие от потоков само ядро не подлежит диспетчеризации. Код ядра исполняется только в том случае, когда какой-нибудь поток вызывает функцию ядра или при обработке аппаратного прерывания.

Как уже говорилось, *QNX* базируется на концепции передачи сообщений. Передачу сообщений, а также их диспетчеризацию осуществляет ядро системы. Кроме того, ядро управляет временными прерываниями. Выполнение остальных функций обеспечивается задачами-администраторами. Программа, желающая создать задачу, посылает сообщение администратору задач (модуль *task*) и блокируется для ожидания ответа. Если новая задача должна выполняться одновременно с порождающей ее задачей, администратор задач *task* создает ее и, отвечая, выдает порождающей задаче идентификатор (*id*) созданной задачи. В противном случае никакого сообщения не посылается до тех пор, пока новая задача не закончится сама по себе. В этом случае в ответе администратора задач будут содержаться конечные характеристики закончившейся задачи.

Сообщения отличаются количеством данных, которые передаются от одной задачи точно к другой задаче. Данные копируются из адресного пространства первой задачи в адресное пространство второй, и выполнение первой задачи приостанавливается до тех пор, пока вторая задача не вернет ответное сообщение. В действительности обе задачи кратковременно взаимодействуют во время выполнения передачи. Ничто, кроме длины сообщения (максимальная длина 65 535 байт), не заботит *QNX* при передаче сообщения. Существует несколько протоколов, которые могут быть использованы для сообщений.

Основные операции над сообщениями – это послать, получить и ответить, а также несколько их вариантов для обработки специальных ситуаций. Получатель всегда идентифицируется своим идентификатором задачи, хотя существуют способы ассоциировать имена с идентификатором задачи. Наиболее интересные варианты операций включают в себя возможность получать (копировать) только первую часть сообщения, а затем получать оставшуюся часть такими кусками, какие потребуются. Это может быть использовано для того, чтобы сначала узнать длину сообщения, а затем динамически распределить принимающий буфер. Если необходимо задержать ответное сообщение до тех пор, пока не будет получено и обработано другое сообщение, то чтение первых нескольких байт дает вам компактный «обработчик», через который позже можно получить доступ ко всему сообщению. Таким образом, ваша задача предохраняется от того, чтобы хранить в себе большое количество буферов.

Другие функции позволяют программе получать сообщения только тогда, когда она уже ожидает их приема, а не блокироваться до тех пор, пока не прибудет сообщение, и транслировать сообщение к другой задаче без изменения идентификатора передатчика. Задача, которая транслировала сообщение, в транзакции невидима.

Кроме этого, *QNX* обеспечивает объединение сообщений в структуру данных, называемую очередью. Очередь – это просто область данных в третьей, отдельной задаче, которая временно принимает передаваемое сообщение и немедленно отвечает передатчику. В отличие от стандартной передачи сообщений, передатчик немедленно освобождается для того, чтобы продолжить свою работу. Задача администратора очереди хранит в себе сообщение до тех пор, пока приемник не будет готов прочитать его; это он делает путем запроса сообщения у администратора очереди. Любое количество сообщений (ограничено только возможностью памяти) может храниться в очереди. Они хранятся и передаются в том порядке, в котором они были приняты. Может быть создано любое количество очередей. Каждая очередь идентифицируется своим именем.

Помимо сообщений и очередей в *QNX* для взаимодействия задач и организации распределенных вычислений имеются так называемые порты, которые позволяют формировать сигнал одного конкретного условия, и механизм исключений.

Порт подобен флагу, известному всем задачам на одном и том же узле (но не на различных узлах). Он имеет точно два состояния, которые могут трактоваться как «присоединить» и «освободить», хотя пользователь может использовать свою интерпретацию; например, «занят» и «доступен». Порты используются для быстрой простой синхронизации между задачей и обработчиком прерываний устройства. Они нумеруются от нуля до максимум 32 (на некоторых типах узлов возможно и больше). Первые 20 номеров зарезервированы для использования операционной системой.

С портом могут быть выполнены три операции:

* присоединить порт;
* отсоединить порт;
* послать сигнал в порт.

Одновременно к порту может быть присоединена только одна задача. Если другая задача попытается «отсоединиться» от того же самого порта, то произойдет отказ при вызове функции, и управление вернется к задаче, которая в настоящий момент присоединена к этому порту. Это самый быстрый способ обнаружить идентификатор другой задачи, подразумевая, что задачи могут договориться использовать один номер порта. Напомним, что все рассматриваемые задачи должны находиться на одном и том же узле. При работе нескольких узлов специальные функции обеспечивают большую гибкость и эффективность.

Любая задача может посылать сигнал в любой порт независимо от того, была ли она присоединена к нему или нет (предпочтительно, чтобы не была присоединена). Сигнал подобен не блокирующей передаче пустого сообщения, т. е. передатчик не приостанавливается, а приемник не получает какие-либо данные; он только отмечает, что конкретный порт изменил свое состояние.

Задача, присоединенная к порту, может ожидать прибытия сигнала или может периодически читать порт. *QNX* хранит информацию о сигналах, передаваемых в каждый порт, и уменьшает счетчик после каждой операции «приема» сигнала («чтение» возвращает счетчик и устанавливает его в нуль). Сигналы всегда принимаются перед сообщениями, давая им тем самым больший приоритет над сообщениями. В этом смысле сигналы часто используются обработчиками прерываний для того, чтобы оповестить задачу о внешних (аппаратных) событиях (действительно, обработчики прерываний не имеют возможности посылать сообщения и должны использовать сигналы).

В отличие от описанных выше методов, которые строго синхронизируются, «исключения» обеспечивают асинхронное взаимодействие. То есть исключение может прервать нормальное выполнение потока задачи. Они, таким образом, являются аварийными событиями. *QNX* резервирует для себя 16 исключений для того, чтобы оповещать задачи о прерывании с клавиатуры, нарушении памяти и подобных необычных ситуациях. Остальные 16 исключений могут быть определены и использованы прикладными задачами.

Системная функция может быть вызвана для того, чтобы позволить задаче управлять своей собственной обработкой исключений, выполняя свою собственную внутреннюю функцию во время возникновения исключения. Заметим, что функция исключения задачи вызывается асинхронно операционной системой, а не самой задачей. Вследствие этого исключения могут иметь сильнодействующее побочное влияние на операции (например, передачу сообщений), которые выполняются в это же время. Обработчики исключений должны быть написаны очень аккуратно.

Одна задача может установить одно или несколько исключений на другой задаче. Они могут быть комбинацией системных исключений (определенных выше) и исключений, определяемых приложениями, что обеспечивает другие возможности для межзадачного взаимодействия.

Благодаря такому свойству *QNX*, как возможность обмена посланиями между задачами и узлами сети, программы не заботятся о конкретном размещении ресурсов в сети. Это свойство придает системе необычную гибкость. Так, узлы могут произвольно добавляться и изыматься из системы, не затрагивая системные программы. *QNX* приобретает эту конфигурационную независимость благодаря применению концепции «виртуальных» задач.

У виртуальных задач непосредственный код и данные, будучи на одном из удаленных узлов, возникают и ведут себя так, как если бы они были локальными задачами какого-то узла со всеми атрибутами и привилегиями. Программа, посылающая сообщение в сети, никогда не посылает его точно. Сначала она открывает «виртуальную цепочку». Виртуальная цепочка включает все виртуальные задачи, связанные между собой. На обоих концах такой связи имеются буферы, которые позволяют хранить самое большое послание из тех, которые цепочка может нести в данном сеансе связи. Сетевой администратор помещает в эти буферы все сообщения для соединенных задач.

Виртуальная задача, таким образом , занимает всего лишь пространство, необходимое для буфера и входа в таблице задач. Чтобы открыть связь, необходимо знать идентификатор узла и задачи, с которой устанавливается связь. Для этого необходимо знать идентификатор задачи администратора, ответственного за данную функцию, или глобальное имя сервера. Наша задача может вообще выполняться на другом узле, где, допустим, имеется более совершенный процессор.

**Особенности мультипрограммирования в системах реального времени**. В системах реального времени, в которых главным критерием эффективности является обеспечение временных характеристик вычислительного процесса, планирование имеет особое значение. Любая система реального времени должна реагировать на сигналы управляемого объекта в течение заданных временных ограничений. Это время называется временем реакции системы, а соответствующее свойство системы − реактивностью. Необходимость тщательного планирования работ облегчается тем, что в системах реального времени весь набор выполняемых задач известен заранее.

Кроме того, часто в системе имеется информация о временах выполнения задач, моментах активизации, предельных допустимых сроках ожидания ответа и т. д. Эти данные могут быть использованы планировщиком для создания статического расписания или для построения адекватного алгоритма динамического планирования.

При разработке алгоритмов планирования для систем реального времени необходимо учитывать, какие последствия в этих системах возникают при несоблюдении временных ограничений. Если эти последствия катастрофичны, как, например, для системы управления полетами или атомной электростанцией, то операционная система реального времени, на основе которой строится управление объектом, называется жесткой (*hard*). Если же последствия нарушения временных ограничений не столь серьезны, то есть сравнимы с той пользой, которую приносит система управления объектом, то система является гибкой или мягкой (*soft*) системой реального времени. Примером мягкой системы реального времени является система резервирования билетов. Если из-за временных нарушений оператору не удается зарезервировать билет, это не очень страшно − можно просто послать запрос на резервирование заново.

В системах реального времени не стремятся максимально загружать все устройства, наоборот, при проектировании программного управляющего комплекса обычно закладывается некоторый «запас» вычислительной мощности на случай пиковой нагрузки. Статистические аргументы о низкой вероятности возникновения пиковой нагрузки, основанные на том, что вероятность одновременного возникновения большого количества независимых событий очень мала, не применимы ко многим ситуациям в системах управления. Например, в системе управления атомной электростанцией в случае возникновения крупной аварии атомного реактора многие аварийные датчики сработают одновременно и создадут коррелированную нагрузку. Если система реального времени не спроектирована для поддержки пиковой нагрузки, то может случиться так, что система не справится с работой именно тогда, когда она нужна в наибольшей степени.

В жестких системах реального времени время завершения выполнения каждой из критических задач должно быть гарантировано для всех возможных сценариев работы системы. Такие гарантии могут быть даны, либо в результате исчерпывающего тестирования всех возможных сценариев поведения управляемого объекта и управляющих программ, либо в результате построения статического расписания, либо в результате выбора математически обоснованного динамического алгоритма планирования.

При построении расписания надо иметь в виду, что для некоторых наборов задач в принципе невозможно найти расписания, при котором бы удовлетворялись заданные временные характеристики. С целью определения возможности существования расписания могут быть использованы различные критерии. Например, в качестве простейшего критерия может служить условие, что разность между предельным сроком выполнения задачи (после появления запроса на ее выполнение) и временем ее вычисления (при условии непрерывного выполнения) всегда должна быть положительной. Очевидно, что такой критерий является необходимым, но недостаточным. Точные критерии, гарантирующие наличие расписания, являются очень сложными в вычислительном отношении.

В мягких системах реального времени предполагается, что заданные временные ограничения могут иногда нарушаться, поэтому здесь обычно применяются менее затратные способы планирования.

В зависимости от характера возникновения запросов на выполнение задач полезно разделять их на два типа: периодические и спорадические. Начиная с момента первоначального запроса все будущие моменты запроса периодической задачи можно определить заранее путем прибавления к моменту начального запроса величины, кратной известному периоду. Времена запросов на выполнение спорадических задач заранее не известны.

При выборе алгоритма планирования следует учитывать данные о возможной зависимости задач. Эта зависимость может выступать, например, в виде ограничений на последовательность выполнения задач или их синхронизации, вызванной взаимными исключениями (запрете выполнения некоторых задач в течение определенных периодов времени).

С практической точки зрения алгоритмы планирования зависимых задач более важны, чем алгоритмы планирования независимых задач. При наличии дешевых микроконтроллеров нет смысла организовывать мультипрограммное выполнение большого количества независимых задач на одном компьютере, так как при этом значительно возрастает сложность программного обеспечения. Обычно одновременно выполняющиеся задачи должны обмениваться информацией и получать доступ к общим данным для достижения общей цели системы, то есть являются зависимыми задачами. Поэтому существование некоторого предпочтения последовательности выполнения задач или взаимного исключения − это скорее норма для систем управления реального времени, чем исключение.

Проблема планирования зависимых задач очень сложна, нахождение ее оптимального решения требует больших вычислительных ресурсов, сравнимых с теми, которые требуются для собственно выполнения задач управления.