Лекция 7. Операционные системы

1.1. Основные понятия, назначения и функции ОС

Основные понятия, назначения и функции ОС

Для того, чтобы ответить на вопрос, что представляет собой операционная система, необходимо сначала рассмотреть вопрос, из чего состоит вычислительная система (ВС) в целом. Обобщенно структура вычислительной системы представлена на рис. 1.

Рисунок 1 – Пользователь и обобщенная структура вычислительной системы

Во-первых, ВС состоит из того, что называют *аппаратным* или *техническим обеспечением* (англ. *hardware*): процессоры, память, мониторы, таймеры, дисковые устройства, накопители на магнитных лентах, сетевая коммуникационная аппаратура, принтеры и т.д., объединенные магистральным соединением (шиной).

Во-вторых, ВС состоит из *программного обеспечения* (ПО), в котором выделяют две части – *системное* и *прикладное*. Системное ПО – это набор программ, которые управляют компонентами ВС, такими как процессор, коммуникационные и периферийные устройства, и предназначены для обеспечения функционирования и работоспособности системы в целом. Большинство из них отвечают непосредственно за контроль и объединение в единое целое различных компонентов аппаратного оборудования ВС, обеспечение работы компьютера самого по себе и выполнение различных прикладных программ. Системное ПО противопоставляется прикладному ПО, которое напрямую решает проблемы пользователя и предназначено для выполнения определенных пользовательских задач и рассчитано на непосредственное взаимодействие с пользователем. К прикладному ПО, как правило, относят разнообразные вспомогательные программы (игры, текстовые процессоры и т.п.).

Следует отметить, что деление на прикладное и системное ПО является отчасти условным и зависит от того, кто осуществляет такое деление. Компилятор языка *C* для обычного программиста – системная программа, а для системного – прикладная.

Принимая во внимание вышеизложенное, следует отметить, что операционная система является фундаментальным компонентом системного программного обеспечения.

Очевидно, что операционная система является основным компонентом любой вычислительной системы и во многом определяет эффективность ее функционирования в целом. При этом, дать однозначное определение операционной системе затруднительно. Главным образом это связано с тем, что операционная система выполняет целый ряд разнородных функций, начиная от обеспечения пользователю-программисту удобств посредством предоставления удобного интерфейса к аппаратной части вычислительной системы и заканчивая обеспечением рационального управления ресурсами вычислительной системы. В связи с этим целесообразно дать несколько различных определений и сделать акцент на цели создания операционных систем, их функции и предназначение.

Главными целями разработчиков операционных систем являются следующие:

1. Эффективное использование всех компьютерных ресурсов.
2. Повышение производительности труда программистов.
3. Простота, гибкость, эффективность и надежность организации вычислительного процесса.
4. Обеспечение независимости прикладного ПО от аппаратного ПО.

Операционная система (ОС) – это программа, которая обеспечивает возможность рационального использования оборудования компьютера удобным для пользователя образом.

ОС – базовый комплекс компьютерных программ, обеспечивающий управление аппаратными средствами компьютера, работу с файлами, ввод и вывод данных, а также выполнение прикладных программ и утилит.

Кроме различных определений ОС, два из которых приведены выше, пользователи выделяют ряд различных «точек зрения» на ОС:

- ОС как виртуальная машина;
- ОС как система управления ресурсами;
- ОС как защитник пользователей и программ;
- ОС как постоянно функционирующее ядро.

Для более полного представления об ОС рассмотрим основные «точки зрения» пользователей более подробно.

ОС как виртуальная машина. Использование архитектуры персонального компьютера на уровне машинных команд является крайне неудобным для использования прикладными программами. Так, работа с диском предполагает знание внутреннего устройства его электронного компонента – контроллера для ввода команд вращения диска, поиска и форматирования дорожек, чтения и записи секторов и т.д. Работа по организации прерываний, работы таймера, управления памятью и т. д. также может требовать при программировании знания и учета большого количества деталей.

В связи с этим необходимо обеспечить интерфейс между пользователем и компьютером, скрывая лишние подробности за счет использования относительно простых и высокоуровневых абстракций. Например, представлять информационное пространство диска как набор файлов, которые можно открывать для чтения или записи, использовать для получения или сброса информации, а затем закрывать, создавать иллюзию неограниченного размера операционной памяти, числа процессоров и прочее. Обеспечением такого высокоуровневого абстрагирования занимается ОС, что позволяет представлять ее пользователю в виде *виртуальной машины*, с которой проще иметь дело, чем непосредственно с оборудованием компьютера.

ОС как система управления ресурсами. В случае, если несколько программ, работающих на одном компьютере, будут пытаться одновременно осуществлять вывод на принтер, то можно получить «мешанину» строчек и страниц. ОС должна предотвращать такого рода хаос за счет

буферизации подобной информации и организации очереди на печать. Не менее актуальная проблема – проблема управления ресурсами для многопользовательских компьютеров.

Таким образом, ОС как *менеджер ресурсов* осуществляет упорядоченное и контролируемое распределение процессоров, памяти и других ресурсов между различными программами.

ОС как защитник пользователей и программ. Если в вычислительной системе требуется обеспечение совместной работы нескольких пользователей, то возникает проблема организации их безопасной деятельности. Так, необходимо обеспечить:

- сохранность информации на диске, защиту от повреждения или удаления файлов;
- разрешение программам одних пользователей произвольно вмешиваться в работу программ других пользователей;
- пресечение попыток несанкционированного использования вычислительной системы.

Эти задачи, как правило, возложены на ОС как организатора безопасной работы пользователей и их программ.

ОС как постоянно функционирующее ядро. Можно говорить об ОС, как о программе (программах), постоянно работающей на компьютере и взаимодействующей с множеством прикладных программ. Очевидно, что такое определение верно лишь отчасти, т.к. во многих современных ОС постоянно работает на компьютере лишь часть ОС, которую принято называть ее ядром.

Учитывая рассмотренное многообразие точек зрения на ОС, целесообразно выполнить обзор предназначений и функций ОС, для чего, в свою очередь, стоит рассмотреть эволюцию развития вычислительных систем в целом и операционных систем, в частности.

1.2.1. История развития ОС

История развития ОС

Поколения ОС, так же как и аппаратные средства связаны с достижениями в области создания электронных компонентов: ламп (1-е поколение), транзисторов (2-е поколение), интегральных микросхем (ИС, 3-е поколение), больших и сверхбольших интегральных схем (БИС и СБИС, 4-е и 5-е поколения). Рассмотрим эволюцию ОС более подробно.

Первое поколение (1940-е – 50-е гг.). В эти годы ОС отсутствуют. Первые шаги в области разработки электронных вычислительных машин были предприняты в конце Второй мировой войны. В середине 40-х гг. были созданы первые ламповые вычислительные устройства и появился принцип программы, хранящейся в памяти машины (*John Von Neumann*, июнь 1945 г.). В то время одна и та же группа людей участвовала и в проектировании, эксплуатации и в программировании вычислительной машины. Это была скорее научно-исследовательская работа в области вычислительной техники, а не регулярное использование компьютеров в качестве инструмента решения каких-либо практических задач из других прикладных областей. Программирование осуществлялось исключительно на машинном языке. Об ОС не было и речи, все задачи организации вычислительного процесса решались вручную каждым программистом с пульта управления. За пультом мог находиться только один пользователь. Программа загружалась в память машины в лучшем случае с колоды перфокарт, а обычно с помощью панели переключателей.

Вычислительная система выполняла одновременно только одну операцию (ввод-вывод или собственно вычисления). Отладка программ велась с пульта управления с помощью изучения состояния памяти и регистров машины. В конце этого периода появляется первое системное программное обеспечение: в 1951–1952 гг. возникают прообразы первых компиляторов с символических языков (*Fortran* и др.), а в 1954 г. *Nat Rochester* разрабатывает Ассемблер для *IBM*-701.

Второе поколение (1950-е – 60-е гг.). С середины 50-х гг. начался следующий период в эволюции вычислительной техники, связанный с появлением новой технической базы – полупроводниковых элементов. Применение транзисторов вместо часто перегоравших электронных ламп привело к повышению надежности компьютеров. Теперь машины непрерывно могут работать достаточно долго, чтобы на них можно было возложить выполнение практически важных задач.

Снижается потребление вычислительными машинами электроэнергии, совершенствуются системы охлаждения. Размеры компьютеров уменьшились. Снизилась стоимость эксплуатации и обслуживания вычислительной техники. Началось использование ЭВМ коммерческими фирмами. Одновременно наблюдается бурное развитие алгоритмических языков (*LISP*, *COBOL*, *ALGOL*-60, *PL*-1 и т.д.). Появляются первые настоящие компиляторы, редакторы связей, библиотеки математических и служебных подпрограмм. Упрощается процесс программирования. Пропадает необходимость взваливать на одних и тех же людей весь процесс разработки и использования компьютеров. Именно в этот период происходит разделение персонала на программистов и операторов, специалистов по эксплуатации и разработчиков вычислительных машин.

Изменяется процесс отладки программ. Теперь пользователь приносит программу с входными данными в виде колоды перфокарт и указывает необходимые ресурсы. Такая колода получает название *задания*. Оператор загружает задание в память машины и запускает его на исполнение. Полученные выходные данные печатаются на принтере, и пользователь получает их обратно через некоторое (довольно продолжительное) время.

Появляются первые системы *пакетной обработки*, которые просто автоматизируют запуск одной программы из пакета за другой и тем самым увеличивают коэффициент загрузки процессора. При реализации систем пакетной обработки был разработан формализованный язык управления заданиями, с помощью которого программист сообщал системе и оператору, какую работу он хочет выполнить на вычислительной машине. Системы пакетной обработки стали прообразом современных ОС, они были первыми системными программами, предназначенными для управления вычислительным процессом.

Следует отметить основные недостатки, присущие вычислительным системам второго поколения:

1. Использование части машинного времени (времени процессора) на выполнение системной управляющей программы.
2. Программа, получившая доступ к процессору, обслуживается до ее завершения. При этом, если возникает потребность в передаче данных между внешними устройствами и памятью, то процессор простаивает, ожидая завершения операции обмена. С другой стороны, при работе процессора простаивают внешние устройства. Для персонального компьютера проявление фактора простоя процессора не столь существенно, так как стоимость его не велика, чего не скажешь о больших и дорогих ЭВМ.

Третий период развития вычислительных машин относится к началу 1960 – 70 гг. В это время в технической базе произошел переход от отдельных полупроводниковых элементов типа транзисторов к интегральным микросхемам. Вычислительная техника становится более надежной и дешевой. Растет сложность и количество задач, решаемых компьютерами. Повышается производительность процессоров.

Повышению эффективности использования процессорного времени мешает низкая скорость работы механических устройств ввода-вывода. Вместо непосредственного чтения пакета заданий с перфокарт в память начинают использовать его предварительную запись, сначала на магнитную ленту, а затем и на диск. Когда в процессе выполнения задания требуется ввод данных, они читаются с диска. Точно так же выходная информация сначала копируется в системный буфер и записывается на ленту или диск, а печатается только после завершения задания. Вначале действительные операции ввода-вывода осуществлялись в режиме *off-line*, т.е. с использованием других, более простых, отдельно стоящих компьютеров. В дальнейшем они начинают выполняться на том же компьютере, который производит вычисления (уже в режиме *on-line*). Такой прием получает название *spooling* (сокращение от *Simultaneous Peripheral Operation On Line*) или подкачки-откачки данных. Введение техники подкачки-откачки в пакетные системы позволило совместить реальные операции ввода-вывода (в основном – печати) одного задания с выполнением другого задания, но, в то же время потребовало разработки аппарата прерываний для извещения процессора об окончании этих операций.

Магнитные ленты были устройствами *последовательного доступа* (информация считывалась с них в том порядке, в каком была записана). Появление магнитного диска, для которого не важен порядок чтения информации (устройства *прямого доступа*), привело к дальнейшему развитию

вычислительных систем. При обработке пакета заданий на магнитной ленте очередность запуска заданий определялась порядком их ввода. При обработке пакета заданий на магнитном диске появилась возможность выбора очередного выполняемого задания. Пакетные системы начинают заниматься планированием заданий: в зависимости от наличия запрошенных ресурсов, срочности вычислений и т.д. на выполнение выбирается то или иное задание.

Дальнейшее повышение эффективности использования процессора было достигнуто с помощью *мультипрограммирования*. Идея мультипрограммирования заключается в следующем: пока одна программа выполняет операцию ввода-вывода, процессор не простаивает, как это происходило при однопрограммном режиме, а выполняет другую программу. Когда операция ввода-вывода заканчивается, процессор возвращается к выполнению первой программы. При этом каждая программа загружается в свой участок оперативной памяти, называемый разделом, и не должна влиять на выполнение другой программы.

Появление мультипрограммирования требует настоящей революции в строении вычислительной системы. Особую роль здесь играет аппаратная поддержка (многие аппаратные новшества появились еще на предыдущем этапе эволюции), наиболее существенные особенности которой перечислены ниже.

- Реализация защитных механизмов. Программы не должны иметь самостоятельного доступа к распределению ресурсов, что приводит к появлению привилегированных и непривилегированных команд. Привилегированные команды, например команды ввода-вывода, могут исполняться только операционной системой. Говорят, что она работает в привилегированном режиме. Переход управления от прикладной программы к ОС сопровождается контролируемой сменой режима. Кроме того, это защита памяти, позволяющая изолировать конкурирующие пользовательские программы друг от друга, а ОС – от программ пользователей.
- Наличие прерываний. Внешние прерывания оповещают ОС о том, что произошло асинхронное событие, например завершилась операция ввода-вывода. Внутренние прерывания (сейчас их принято называть исключительными ситуациями) возникают, когда выполнение программы привело к ситуации, требующей вмешательства ОС, например деление на ноль или попытка нарушения защиты.
- Развитие параллелизма в архитектуре. Прямой доступ к памяти и организация каналов ввода-вывода позволили освободить центральный процессор от рутинных операций.

Не менее важна в организации мультипрограммирования роль собственно ОС. Она отвечает за следующие операции:

- Организация интерфейса между прикладной программой и ОС при помощи системных вызовов
- Организация очереди из заданий в памяти и выделение процессора одному из заданий потребовало планирования использования процессора.
- Переключение с одного задания на другое требует сохранения содержимого регистров и структур данных, необходимых для выполнения задания, иначе говоря, контекста для обеспечения правильного продолжения вычислений.
- Поскольку память является ограниченным ресурсом, нужны стратегии управления памятью, то есть требуется упорядочить процессы размещения, замещения и выборки информации из памяти.
- Организация хранения информации на внешних носителях в виде файлов и обеспечение доступа к конкретному файлу только определенным категориям пользователей.
- Поскольку программам может потребоваться произвести санкционированный обмен данными, необходимо их обеспечить средствами коммуникации.
- Для корректного обмена данными необходимо разрешать конфликтные ситуации, возникающие при работе с различными ресурсами и предусмотреть координацию программами своих действий, т.е. снабдить систему средствами синхронизации.

Мультипрограммные системы обеспечили возможность более эффективного использования системных ресурсов (например, процессора, памяти, периферийных устройств), но они еще долго оставались пакетными. Пользователь не мог непосредственно взаимодействовать с заданием и должен был предусмотреть с помощью управляющих карт все возможные ситуации. Отладка программ по-прежнему занимала много времени и требовала изучения многостраничных распечаток содержимого памяти и регистров или использования отладочной печати.

Параллельно внутренней эволюции вычислительных систем происходила и внешняя их эволюция. До начала этого периода вычислительные комплексы были, как правило, несовместимы. Каждый имел собственную ОС, свою систему команд и т.д. В результате программу, успешно работающую на одном типе машин, необходимо было полностью переписывать и заново отлаживать для выполнения на компьютерах другого типа. В начале третьего периода появилась идея создания семейств программно совместимых машин, работающих под управлением одной и той же ОС. Первым семейством программно совместимых компьютеров, построенных на интегральных микросхемах, стала серия машин IBM/360. Разработанное в начале 60-х годов, это семейство значительно превосходило машины второго поколения по критерию цена/производительность. За ним последовала линия компьютеров PDP, несовместимых с линией IBM, и лучшей моделью в ней стала PDP-11.

Сила «одной семьи» была одновременно и ее слабостью. Широкие возможности этой концепции (наличие всех моделей: от мини-компьютеров до гигантских машин; обилие разнообразной периферии; различное окружение; различные пользователи) порождали сложную и громоздкую ОС. Миллионы строчек Ассемблера, написанные тысячами программистов, содержали множество ошибок, что вызывало непрерывный поток публикаций о них и попыток исправления. Только в операционной системе OS/360 содержалось более 1000 известных ошибок. Тем не менее, идея стандартизации ОС была широко внедрена в сознание пользователей и в дальнейшем получила активное развитие.

К этому же периоду относится появление первых *операционных систем реального времени* (ОСРВ), в которых ЭВМ применяется для управления 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 сойдет с орбиты, экспериментальные данные могут быть потеряны, толщина гальванического покрытия не будет соответствовать норме и т.п. Характерным для ОСРВ является обеспечение заранее заданных интервалов времени реакции на предусмотренные события для получения управляющего воздействия. Поскольку в технологических процессах промедление может привести к нежелательным и даже опасным последствиям, ОСРВ работают со значительной недогрузкой, так как важнейшей характеристикой является постоянная готовность системы – ее *реактивность*.

Системное программное обеспечение этого периода решало множество проблем, связанных с защитой результатов работы различных программ, защитой данных в оперативной памяти и других данных. Кроме того, ОС должна управлять новыми устройствами, входящими в состав аппаратного обеспечения. Для решения этих задач системное программное обеспечение сформировалось в сложную систему, требующую для реализации своих возможностей значительных вычислительных ресурсов.

ОС четвертого поколения (1970-80-е гг.) были многорежимными системами, обеспечивающими пакетную обработку, разделение времени, режим реального времени и мультипроцессорный режим. Они были громоздкими, дорогостоящими («монстры» операционных систем). Например, стоимость разработки ОС OS/360 фирмой IBM соизмерима с затратами на реализацию американским национальным космическим агентством программы высадки человека на луну. Такие ОС, будучи прослойкой, между пользователем и аппаратурой ЭВМ, привели к значительному усложнению вычислительной обстановки. Для выполнения простейшей программы необходимо было изучать сложные языки управления заданием (*Job Control Language – JCL*). К этому периоду относится появление вытесняющей многозадачности (*Preemptive scheduling*) и использование концепции баз данных для хранения больших объемов информации для организации распределенной обработки. Программисты перестали использовать перфокарты и магнитные ленты для хранения своих данных. Вводится приоритетное планирование (*Prioritized scheduling*) и

выделение квот на использование ограниченных ресурсов компьютеров (процессорного времени, дисковой памяти, физической (оперативной) памяти).

Появление электронно-лучевых дисплеев и переосмысление возможностей применения клавиатур поставили на очередь решение этой проблемы. Логическим расширением систем мультипрограммирования стали системы *разделения времени* (*time-sharing* системы). В них процессор переключается между задачами не только на время операций ввода-вывода, но и через определенные интервалы времени. Эти переключения происходят так часто, что пользователи могут взаимодействовать со своими программами во время их выполнения, то есть интерактивно. В результате появляется возможность одновременной работы нескольких пользователей на одной компьютерной системе. У каждого пользователя для этого должна быть хотя бы одна программа в памяти. Чтобы уменьшить ограничения на количество работающих пользователей была внедрена идея неполного нахождения исполняемой программы в оперативной памяти. Основная часть программы находится на диске, и фрагмент, который необходимо в данный момент выполнять, может быть загружен в оперативную память, а ненужный – выкачан обратно на диск. Это реализуется с помощью механизма виртуальной памяти. Основным достоинством такого механизма является создание иллюзии неограниченной оперативной памяти ЭВМ.

В системах разделения времени пользователь получил возможность эффективно производить отладку программы в интерактивном режиме и записывать информацию на диск, не используя перфокарты, а непосредственно с клавиатуры. Появление *on-line*-файлов привело к необходимости разработки развитых файловых систем.

Пятое поколение (с середины 1980-х гг. по н.в.). Период характеризуется уменьшением стоимости компьютеров и увеличением стоимости труда программиста. Появление персональных компьютеров позволило установить компьютер практически каждому пользователю на рабочем столе. Благодаря широкому распространению вычислительных сетей и средств оперативной обработки (работающих в режиме *on-line*), пользователи получают доступ к территориально распределенным компьютерам. Появились микропроцессоры, на основе которых создаются все новые и новые персональные компьютеры, которые могут быть использованы как автономно, так и в качестве терминалов более мощных вычислительных систем. При передаче информации по линиям связи усложняются проблемы защиты информации, шифрования данных. Возникло понятие сетевого компьютера (*Network computer*), способного получать все ресурсы через компьютерную сеть. Понятие файловой системы распространяется на данные, доступные по различным сетевым протоколам.

Число людей, пользующихся компьютером, значительно возросло, что выдвигает требование наличия дружественного интерфейса пользователя, ориентации на неподготовленного пользователя. Появились системы с управлением с помощью меню и элементов графического интерфейса. Начала широко распространяться концепция виртуальных машин. Пользователь более не заботится о физических деталях построения ЭВМ или сетей. Он имеет дело с функциональным эквивалентом компьютера, создаваемым для него операционной системой, представляющим виртуальную машину. Таким образом, возникла концепция *виртуализации ресурсов* ЭВМ – создание функциональных программно моделируемых эквивалентов реального монопольного ресурса, допускающих их совместное использование многими процессами. Например, мультипрограммирование – виртуализация центрального процессора, буферный ввод-вывод – виртуализация устройств ввода и вывода.

Широкое внедрение получила концепция распределенной обработки данных. Развитием распределенной обработки данных стала технология *«клиент – сервер»*, в которой серверный процесс предоставляет возможность использовать свои ресурсы клиентскому процессу по соответствующему протоколу взаимодействия. Название сервера отображает вид ресурса, который предоставляется клиентским системам (сервер печати, сервер вычислений, сервер баз данных, сервер новостей, сервер FTP, сервер WWW и т.д.).

1.2.2. Основные функции ОС

Основные функции ОС

Обзор этапов развития вычислительных и операционных систем позволяет все функции ОС условно разделить на две различные группы – *интерфейсные* и *внутренние*. К интерфейсным функциям ОС относят:

- управление аппаратными средствами;
- управление устройствами ввода- вывода;
- поддержку файловой системы;
- поддержку многозадачности (разделение использования памяти, времени выполнения);
- ограничение доступа, многопользовательский режим работы, планирование доступа пользователей к общим ресурсам;
- интерфейс пользователя (команды в MS DOS, Unix; графический интерфейс в ОС Windows);
- поддержка работы с общими данными в режиме коллективного пользования;
- поддержка работы в локальных и глобальных сетях.

К внутренним функциям ОС, которые выделились в процессе эволюции вычислительных и операционных систем, следует отнести:

- реализацию обработки прерываний;
- управление виртуальной памятью;
- планирование использования процессора;
- обслуживание драйверов устройств.

1.2.3. Особенности современного этапа развития ОС

Особенности современного этапа развития ОС

В 90-е годы практически все ОС, занимающие заметное место на рынке, стали сетевыми. Сетевые функции сегодня встраиваются в ядро ОС, являясь ее неотъемлемой частью. Операционные системы получили средства для работы со всеми основными технологиями локальных (*Ethernet, Fast Ethernet, Gigabit Ethernet, Token Ring, FDDI, ATM*) и глобальных (*X.25, frame relay, ISDN, ATM*) сетей, а также средства для создания составных сетей (*IP, IPX, AppleTalk, RIP, OSPF, NLSP*). В ОС используются средства мультиплексирования нескольких стеков протоколов, за счет которого компьютеры могут поддерживать одновременную сетевую работу с разнородными клиентами и серверами. Появились специализированные ОС, которые предназначены исключительно для выполнения коммуникационных задач. Например, сетевая ОС *IOS* компании *Cisco Systems*, работающая в маршрутизаторах, организует в мультипрограммном режиме выполнение набора программ, каждая из которых реализует один из коммуникационных протоколов.

Во второй половине 90-х годов все производители ОС резко усилили поддержку средств работы с *Internet* (кроме производителей *Unix*-систем, в которых эта поддержка всегда была существенной). Кроме самого стека *TCP/IP* в комплект поставки начали включать утилиты, реализующие такие популярные сервисы *Internet* как *telnet, ftp, DNS* и *Web*. Влияние *Internet* проявилось и в том, что компьютер превратился из чисто вычислительного устройства в средство коммуникаций с развитыми вычислительными возможностями.

Особое внимание в течение всего последнего десятилетия уделялось корпоративным сетевым ОС. Очевидно, что и в обозримом будущем они продолжат свое развитие. Корпоративная ОС характеризуется способностью устойчиво работать в крупных сетях, которые характерны для больших предприятий, имеющих отделения в десятках городов и, возможно, в разных странах. Таким сетям органически присуща высокая степень гетерогенности программных и аппаратных средств, поэтому корпоративная ОС должна успешно взаимодействовать с ОС разных типов и работать на различных аппаратных платформах.

На современном этапе развития операционных систем на передний план вышли средства обеспечения безопасности. Это связано с возросшей ценностью информации, обрабатываемой компьютерами, а также с повышенным уровнем угроз, существующих при передаче данных по

сетям, особенно по публичным, таким как Интернет. Многие операционные системы обладают сегодня развитыми средствами защиты информации, основанными на шифрации данных, аутентификации и авторизации.

Современным операционным системам присуща многоплатформенностъ, то есть способность работать на совершенно различных типах компьютеров. Многие операционные системы имеют специальные версии для поддержки кластерных архитектур, обеспечивающих высокую производительность и отказоустойчивость. Исключением пока является ОС *NetWare*, все версии которой разработаны для платформы *Intel*, а реализации функций *NetWare* в виде оболочки для других ОС, например *NetWare for AIX*, успеха на имели.

2.1. Архитектурные особенности ОС

Архитектурные особенности ОС

Рассмотрев эволюцию развития вычислительных и операционных систем, функции ОС «извне», рассмотрим, что представляют собой ОС «изнутри» и какие подходы существуют к их построению.

2.1.1. Монолитное ядро

Монолитное ядро

По сути, ОС – это программа, которую можно реализовать с использованием процедур и функций. Если при этом ОС 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 режим*, и наоборот, то такая архитектура построения ОС называется *монолитным ядром* (англ. *monolithic kernel*).

Архитектура «монолитное ядро» характеризуется тем, что:

- каждая процедура может вызвать каждую;
- все процедуры работают в привилегированном режиме;
- все части монолитного ядра работают в одном адресном пространстве;
- ядро «совпадает» со всей ОС;
- сборка (компиляция) ядра осуществляется отдельно для каждого компьютера, при установке, добавлении или исключении отдельных компонент требуется перекомпиляция;
- старейший способ организации ОС.

Архитектура «монолитное ядро» имеют долгую историю развития и усовершенствования и, на данный момент, являются наиболее архитектурно зрелыми и пригодными к эксплуатации. Вместе с тем, монолитность ядер усложняет их отладку, понимание кода ядра, добавление новых функций и возможностей, удаление кода, унаследованного от предыдущих версий. «Разбухание» кода монолитных ядер также повышает требования к объёму оперативной памяти, требуемому для функционирования ядра ОС. Это делает монолитные ядерные архитектуры мало пригодными к эксплуатации в системах, сильно ограниченных по объёму ОЗУ, например, встраиваемых системах, производственных микроконтроллерах и т.д.

Старые монолитные ядра требовали перекомпиляции при любом изменении состава оборудования. Следует отметить, что большинство современных ядер позволяют во время работы динамически подгружать модули, выполняющие части функции ядра. Такие ядра называются модульными ядрами. Возможность динамической подгрузки модулей не нарушает монолитности архитектуры ядра, так как динамически подгружаемые модули загружаются в адресное пространство

ядра и в дальнейшем работают как интегральная часть ядра. Не следует путать модульность ядра с *гибридной* или *микроядерной* архитектурой (см. ниже).

Примером систем с монолитным ядром служит большинство *Unix*–подобных систем, таких как *BSD* или *Linux.*

2.1.2. Микроядерная архитектура

Микроядерная архитектура

При разработке ОС используют подход, при котором значительную часть системного кода переносят на уровень пользователя с одновременной минимизацией ядра. Системы, разработанные с использованием такого подхода, называют реализованными в микроядерной архитектуре (англ. *microkernel architecture*). В этом случае построение ядра ОС осуществляется так, что большинство составляющих ОС являются самостоятельными программами, а взаимодействие между ними обеспечивает специальный модуль ядра – *микроядро*, работающее в привилегированном режиме и обеспечивающее взаимодействие между программами, планирование использования процессора, первичную обработку прерываний, операции ввода-вывода и базовое управление памятью (рис. 2).

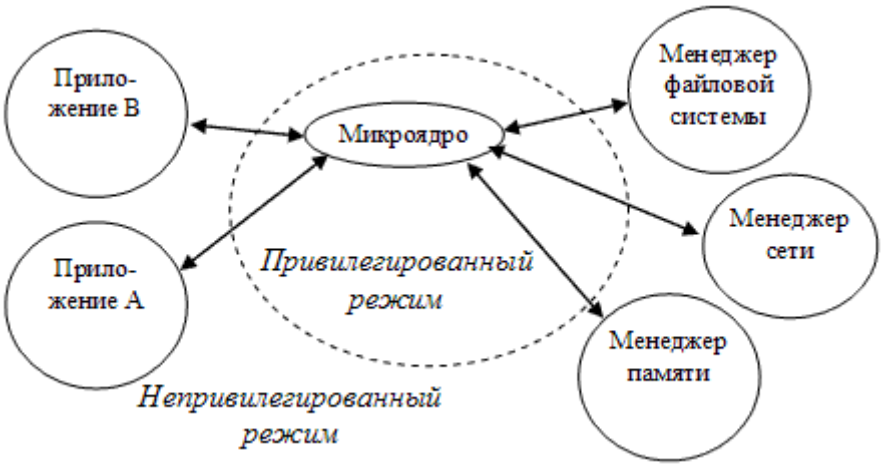

Рисунок 2 – Микроядерная архитектура операционной системы

В микроядерных ОС выделяют центральный компактный модуль, относящийся к супервизорной части системы. Этот модуль имеет очень небольшие размеры и выполняет относительно небольшое количество управляющих функций, но позволяет передать управление на другие управляющие модули, которые и выполнят затребованную функцию. Микроядро – это минимальная главная (стержневая) часть ОС, служащая основой модульных и переносимых расширений. Микроядро само является модулем системного ПО, работающим в наиболее приоритетном состоянии компьютера и поддерживающим связи с остальной частью операционной системы, которая рассматривается как набор серверных приложений (служб).

Основная идея, заложенная в технологию микроядра, заключается в том, чтобы создать необходимую среду верхнего уровня иерархии, из которой можно легко получить доступ ко всем функциональным возможностям уровня аппаратного обеспечения. При этом микроядро является стартовой точкой для создания всех остальных модулей системы. Остальные модули, реализующие необходимые системе функции, вызываются из микроядра и выполняют сервисную роль, получая при этом статус обычного процесса.

Важнейшая задача при разработке микроядра заключается в выборе базовых примитивов, которые должны находиться в микроядре для обеспечения необходимого и достаточного сервиса. В микроядре содержится и исполняется минимальное количество кода, необходимое для реализации основных системных вызовов. В число этих вызовов входят передача сообщений и организация другого общения между внешними по отношению к микроядру процессами, поддержка управления прерываниями, а также ряд других весьма немногочисленных функций. Остальные системные функции, характерные для «обычных» (не микроядерных) операционных систем, обеспечиваются

как модульные дополнения-процессы, взаимодействующие главным образом между собой посредством передачи сообщений.

К преимуществам построения ОС в данной архитектуре относят:

- упрощенное добавление и отладка компонентов ОС без необходимости перезапуска системы за счет высокой степени модульности ядра;
- возможность без прерывания работы системы, загружать и выгружать новые драйверы, файловые системы и т.д.
- возможность отладки компонентов ядра с помощью обычных программных средств;
- повышенная надежность системы (ошибка на уровне непривилегированной программы менее опасна, чем отказ на уровне режима ядра).

К недостаткам построения ОС в данной архитектуре относят:

- дополнительные накладные расходы, связанные с передачей сообщений;
- усложнение процесса проектирования при попытке снижения возможных накладных расходов (требуется «аккуратное» проектирование, разбиение системы на компоненты, минимизация взаимодействия между ними).

2.1.3. Многоуровневые системы

Многоуровневые системы

Обеспечивая строгую структуризацию, можно представить всю вычислительную систему в виде ряда уровней с хорошо определенными связями между ними. При этом объекты уровня N могут вызывать только объекты уровня N-1. Чем ниже уровень, тем более привилегированные команды и действия может выполнять модуль, находящийся на этом уровне. Впервые такой подход был применен при создании системы *THE* (*Technishe Hogeschool Eindhoven*) в 1968 г. Дейкстрой (*Dijkstra*) и его студентами (рис. 3).

5	Интерфейс пользователя
4	Управление вводом-выводом
3	Драйвер устройства связи оператора и консоли
2	Планирование задач и процессов
1	Управление памятью
0	Аппаратное обеспечение

Рисунок 3 – Структура системы THE

Вычислительные системы, реализованные в подобной архитектуре, называют *многоуровневыми системами* (англ. *layered systems*).

В качестве достоинства многоуровневых систем отмечают:

- простоту реализации (за счет того, что при использовании операций нижнего слоя не нужно знать, как они реализованы, нужно лишь понимать, что они делают);
- простоту тестирования (отладка осуществляется послойно и при возникновении ошибки всегда легко локализовать ошибку);
- простоту модификации (при необходимости можно заменить лишь один слой, не трогая остальные).

К недостаткам относят:

- сложность разработки (непросто верно определить порядок и состав каждого из слоев);
- меньшая по сравнению с монолитными системами эффективность за счет необходимости прохождения целого ряда слоев (например, для выполнения операций ввода-вывода программе пользователя придется последовательно проходить все слои от верхнего до нижнего).

2.1.4. Виртуальные машины

Виртуальные машины

Виртуальной машиной (англ. *virtual machine*) называют программную или аппаратную среду, исполняющую некоторый код (например, байт-код или машинный код реального процессора). Зачастую виртуальная машина эмулирует работу реального компьютера. На виртуальную машину, так же как и на реальный компьютер можно инсталлировать ОС, у виртуальной машины может быть BIOS, оперативная память, жёсткий диск (выделенное место на жёстком диске реального компьютера), могут эмулироваться периферийные устройства. На одном компьютере может функционировать несколько виртуальных машин. На рис. 4 представлена обобщенная структура некоторой виртуальной машины с тремя различными ОС.

Виртуальная машина реализует для пользователя имитацию *hardware* в вычислительной системе (процессор, привилегированные и непривилегированные команды, устройства ввода-вывода, прерывания и т.д.). При обращении к «виртуальному hardware» на уровне привилегированных команд в действительности происходит системный вызов реальной ОС, которая и производит все необходимые действия.

<table>
<tr><td>Программа пользователя</td><td>Программа пользователя</td><td>Программа пользователя</td></tr>
<tr><td>MS-DOS</td><td>Linux</td><td>Windows NT</td></tr>
<tr><td>Виртуальное hardware</td><td>Виртуальное hardware</td><td>Виртуальное hardware</td></tr>
<tr><td colspan="3">Реальная операционная система</td></tr>
<tr><td colspan="3">Реальное аппаратное обеспечение</td></tr>
</table>

Рисунок 4 – Обобщенная структура некоторой виртуальной машины

Недостатками реализации ОС в подобных архитектурах является снижение эффективности виртуальных машин по сравнению с реальным компьютером, и, как правило, их громоздкость. Преимуществом является использование в рамках одной вычислительной системы программ, созданных для разных ОС. Примерами ОС, реализованных в подобной архитектуре, являются *CP/CMS* (*VM/370*) для семейства машин *IBM/370*, *VMWare Workstation* компании *VMWare*.

2.1.5. Смешанные системы

Смешанные системы

В большинстве случаев современные ОС используют различные комбинации подходов, рассмотренных в пп. 2.1.1-2.1.4, реализуя смешанные (гибридные) ОС. Например, ядро ОС *Linux* представляет собой монолитную систему с элементами микроядерной архитектуры. Системы 4.4*BSD* и *MkLinux* – монолитные ОС, работающие на микроядре *Mach* (микроядро обеспечивает управление

виртуальной памятью и работу низкоуровневых драйверов; остальные функции, в том числе взаимодействие с прикладными программами, осуществляется монолитным ядром). Совместно элементы микроядерной архитектуры и элементы монолитного ядра используются в ядре *Windows NT*:

- компоненты ядра *Windows NT* располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных ОС;
- все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных.

2.2. Классификация ОС

Классификация ОС

В зависимости от выбранного признака, по которому один объект отличают от другого, вариантов классификации может быть множество. Что касается ОС, здесь уже давно сформировалось относительно небольшое количество классификаций: по назначению, по режиму обработки задач, по способу взаимодействия с системой и, наконец, по способам построения (архитектурным особенностям системы).

Прежде всего, традиционно различают ОС *общего* и *специального назначения*. Системы специального назначения, в свою очередь, подразделяются на ОС для носимых микрокомпьютеров и различных встроенных систем, организации и ведения баз данных, решения задач реального времени и т.п. Еще недавно ОС для персональных компьютеров относили к ОС специального назначения. Сегодня современные мультизадачные ОС для персональных компьютеров уже многими относятся к ОС общего назначения, поскольку их можно использовать для самых разнообразных целей.

По *режиму обработки* задач различают ОС, обеспечивающие *однопрограммный* и *мультипрограммный* (мультизадачный, многозадачный) режимы. Любая задержка в решении программы (например, для осуществления операций ввода-вывода данных) используется для выполнения других программ. Однозадачные ОС (например, *MS-DOS, MSX*) выполняют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером, а также включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем.

Следует различать понятия *«мультипрограммный режим»* и *«мультизадачный режим»*. Принципиальное отличие этих понятий заключается в том, что мультипрограммный режим обеспечивает параллельное выполнение нескольких приложений, и при этом программисты, создающие эти программы, не должны заботиться о механизмах организации их параллельной работы (эти функции берет на себя сама ОС). Мультизадачный режим, наоборот, предполагает, что забота о параллельном выполнении и взаимодействии приложений ложится как раз на прикладных программистов. Современные ОС для персональных компьютеров реализуют мультипрограммный и мультизадачный режимы работы.

Среди множества существующих вариантов реализации многозадачности можно выделить две группы:

1. *Невытесняющая многозадачность* (*NetWare, Windows 3.x*) – механизм планирования процессов целиком сосредоточен в ОС. В этом случае активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление ОС для того, чтобы та выбрала из очереди другой готовый к выполнению процесс.
2. *Вытесняющая многозадачность* (*Windows NT, OS/2, Unix*) – механизм планирования процессов распределен между системой и прикладными программами. При вытесняющей многозадачности решение о переключении процессора с одного процесса на другой принимается ОС, а не самим активным процессом.

Также многозадачные ОС подразделяют на различные типы в соответствии с использованными при их разработке критериями эффективности:

- *системы пакетной обработки* (например, *ЕС*, критерий – коэффициент загрузки процессора);
- *системы разделения времени* (*Unix, VMS*, критерий – удобство и эффективность работы пользователей при одновременном выполнении нескольких пользовательских приложений);
- *системы реального времени* (*QNX, RT/11*, критерий – реактивность).

Информация о системах пакетной обработки и разделения времени приведена выше в п. 1.2. Как отмечено выше, основной особенностью ОСРВ является обеспечение обработки поступающих заданий в течение заданных интервалов времени, которые нельзя превышать. Поток заданий в общем случае не является планомерным и не может регулироваться оператором (характер следования событий можно предсказать лишь в редких случаях), то есть задания поступают в непредсказуемые моменты времени и без всякой очередности. Лучшие характеристики по производительности для систем реального времени обеспечиваются однотерминальными ОСРВ. Средства организации мультитерминального режима всегда замедляют работу системы в целом, но расширяют функциональные возможности системы. Одной из наиболее известных ОСРВ для персональных компьютеров является ОС QNX [18].

Если принимать во внимание способ взаимодействия с компьютером, то можно говорить о *диалоговых* системах и системах *пакетной обработки*. Доля последних хоть и не убывает в абсолютном исчислении, но в процентном отношении она существенно сократилась по сравнению с диалоговыми системами.

При организации работы с вычислительной системой в диалоговом режиме можно говорить об *однопользовательских* (однотерминальных) и многопользовательских (*мультитерминальных*) ОС. В мультитерминальных ОС с одной вычислительной системой одновременно могут работать несколько пользователей, каждый со своего терминала. При этом у пользователей возникает иллюзия, что у каждого из них имеется собственная вычислительная система. Очевидно, что для организации мультитерминального доступа к вычислительной системе необходимо обеспечить мультипрограммный режим работы. В качестве одного из примеров мультитерминальных ОС для персональных компьютеров можно назвать *Linux*. Некая имитация мультитерминальных возможностей имеется и в системе *Windows XP*. В этой ОС каждый пользователь после регистрации (входа в систему) получает свою виртуальную машину. Если необходимо временно предоставить компьютер другому пользователю, вычислительные процессы первого можно не завершать, а просто для этого другого пользователя система создает новую виртуальную машину. В результате компьютер будет выполнять задачи и первого, и второго пользователя. Количество параллельно работающих виртуальных машин определяется имеющимися ресурсами. Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей.

Кроме того, если в ОС отсутствуют или присутствуют средства поддержки многопроцессорной обработки, они могут быть разделены на *многопроцессорные* и *однопроцессорные*. Как правило, функции мультипроцессирования 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, NetWare 4.1* фирмы *Novell*, однако, очевидно, их наличие усложняет алгоритмы управления ресурсами. В свою очередь, 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.

Следует отметить еще один признак, по которому разделяют ОС – организация работы с вычислительной сетью. По этому признаку выделяют *сетевые* ОС и *распределенные* ОС (следует отметить, что иногда в литературе такое разделение отсутствует). Сетевая ОС характеризуется тем, что наделена развитыми функциями работы с сетью, а также контроля доступа к файлам (систему прав доступа). К сетевым ОС относят как системы для рабочих мест (*Novell for DOS, MS Windows, GNU/Linux*), так серверные ОС (*GNU/Linux*, семейство *BSD*-систем, серверные версии *MS Windows*), а также специализированные ОС сетевого оборудования (*Cisco IOS*) [19].

При использовании распределенной ОС пользователь не знает, на локальной или удаленной машине хранятся его файлы и выполняются его программы, он может не знать, подключен ли его компьютер к сети. Внешне распределенная ОС выглядит как обычная автономная система, а ее внутреннее строение имеет существенные отличия от автономных систем.

Также ОС классифицируют по архитектуре, в которой они реализованы. Виды архитектур, в которых реализуются ОС, достаточно подробно изложены выше в п. 2.1.