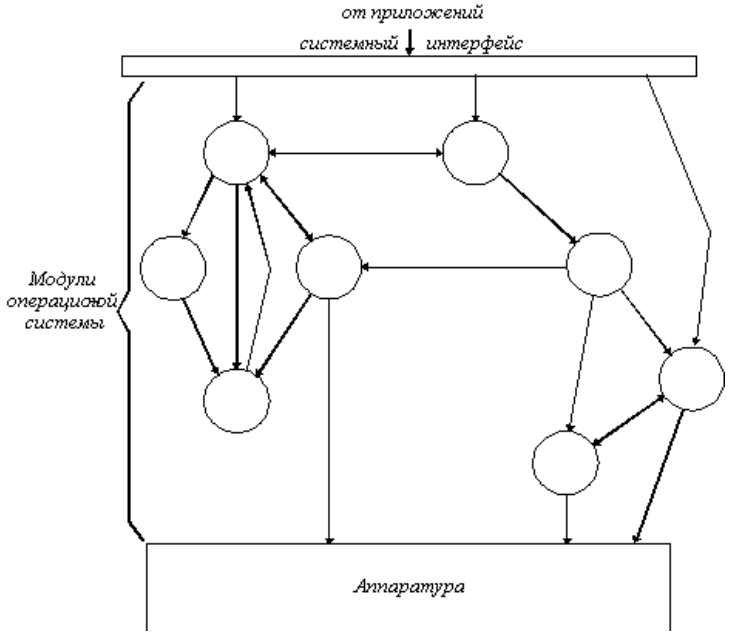
# 1. Монолитные системы

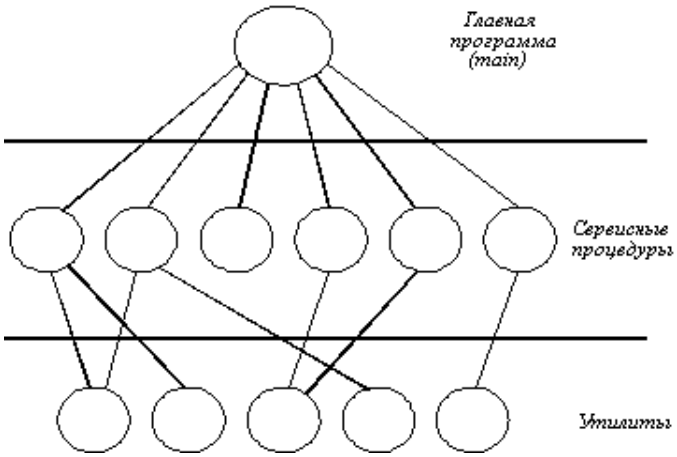



В общем случае "структура" монолитной системы представляет собой отсутствие структуры. ОС написана как набор процедур, каждая из которых может вызывать другие, когда ей это нужно. При использовании этой техники каждая процедура системы имеет хорошо определенный интерфейс в терминах параметров и результатов, и каждая вольна вызвать любую другую для выполнения некоторой нужной для нее полезной работы.

Однако даже монолитные системы могут быть немного структурированными. При обращении к системным вызовам, поддерживаемым ОС, параметры помещаются в строго определенные места, такие, как регистры или стек, а затем выполняется специальная команда прерывания, известная как вызов ядра или вызов супервизора.

Такая организация ОС предполагает следующую структуру:

1. Главная программа, которая вызывает требуемые сервисные процедуры.
2. Набор сервисных процедур, реализующих системные вызовы.
3. Набор утилит, обслуживающих сервисные процедуры.




# 2. Многоуровневые системы

Обобщением предыдущего подхода является организация ОС как иерархии уровней. Уровни образуются группами функций операционной системы - файловая система, управление процессами и устройствами и т.п. Каждый уровень может взаимодействовать только со своим непосредственным соседом - выше- или нижележащим уровнем. Прикладные программы или модули самой операционной системы передают запросы вверх и вниз по этим уровням.

Первой системой, построенной таким образом была простая пакетная система THE, которую построил Дейкстра и его студенты в 1968 году.

Система имела 6 уровней. Уровень 0 занимался распределением времени процессора, переключая процессы по прерыванию или по истечении времени. Уровень 1 управлял памятью - распределял оперативную память и пространство на магнитном барабане для тех частей процессов (страниц), для которых не было места в ОП, то есть слой 1 выполнял функции виртуальной памяти. Слой 2 управлял связью между консолью оператора и процессами. С помощью этого уровня каждый процесс имел свою собственную консоль оператора. Уровень 3 управлял устройствами ввода-вывода и буферизовал потоки информации к ним и от них. С помощью уровня 3 каждый процесс вместо того, чтобы работать с конкретными устройствами, с их разнообразными особенностями, обращался к абстрактным устройствам ввода-вывода, обладающим удобными для пользователя характеристиками.

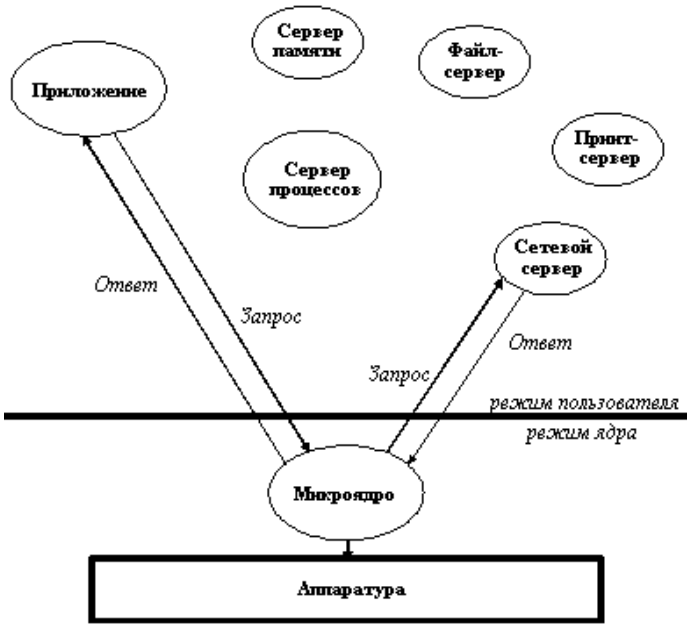
В системе THE многоуровневая схема служила, в основном, целям разработки, так как все части системы компоновались затем в общий объектный модуль.

Дальнейшее обобщение многоуровневой концепции было сделано в ОС MULTICS. В системе MULTICS каждый уровень (называемый кольцом) является более привилегированным, чем

вышележащий. Когда процедура верхнего уровня хочет вызвать процедуру нижележащего, она должна выполнить соответствующий системный вызов, то есть команду TRAP (прерывание), параметры которой тщательно проверяются перед тем, как выполняется вызов.

## 3. Модель клиент-сервер и микроядра

Модель клиент-сервер - это еще один подход к структурированию ОС. В широком смысле модель клиент-сервер предполагает наличие программного компонента - потребителя какого-либо сервиса - клиента, и программного компонента - поставщика этого сервиса - сервера. Взаимодействие между клиентом и сервером стандартизуется, так что сервер может обслуживать клиентов, реализованных различными способами и, может быть, разными производителями. При этом главным требованием является то, чтобы они запрашивали услуги сервера понятным ему способом. Инициатором обмена обычно является клиент, который посылает запрос на обслуживание серверу, находящемуся в состоянии ожидания запроса. Один и тот же программный компонент может быть клиентом по отношению к одному виду услуг, и сервером для другого вида услуг. Модель клиент-сервер является удобным концептуальным средством ясного представления функций того или иного программного элемента в той или иной ситуации. Эта модель успешно применяется не только при построении ОС, но и на всех уровнях программного обеспечения, и имеет в некоторых случаях более узкий, специфический смысл, сохраняя, естественно, при этом все свои общие черты.




Каждый сервер выполняется в пользовательском режиме. Клиент, которым может быть либо другой компонент ОС, либо прикладная программа, запрашивает сервис, посылая сообщение на сервер. Ядро ОС (называемое здесь микроядром), работая в привилегированном режиме, доставляет сообщение нужному серверу, сервер выполняет операцию, после чего ядро возвращает результаты клиенту с помощью другого сообщения.

Данная теоретическая модель является идеализированным описанием системы клиент-сервер, в которой ядро состоит только из средств передачи сообщений. В действительности различные варианты реализации модели клиент-сервер в структуре ОС могут существенно различаться по объему работ, выполняемых в режиме ядра.

Микроядро реализует жизненно важные функции, лежащие в основе операционной системы. Это базис для менее существенных системных служб и приложений. Именно вопрос о том, какие из системных функций считать несущественными, и, соответственно, не включать их в состав ядра, является предметом спора среди соперничающих сторонников идеи микроядра.

Есть два пути реализации данной модели:

1. разместить несколько чувствительных к режиму работы процессора, серверов, в пространстве ядра, что обеспечит им полный доступ к аппаратуре и , в то же время, связь с другими процессами с помощью обычного механизма сообщений. Такой подход был использован, например, при разработке Windows NT: кроме микроядра, в привилегированном режиме работает часть Windows NT, называемая executive управляющей программой. Она включает ряд компонентов, которые управляют виртуальной памятью, объектами, вводом-выводом и файловой системой (включая сетевые драйверы), взаимодействием процессов, и частично системой безопасности;
2. оставить в ядре только небольшую часть сервера, представляющую собой механизм реализации решения, а часть, отвечающую за принятие решения, переместить в пользовательскую область. В соответствии с этим подходом, например, в микроядре размещается только часть системы управления процессами (и нитями), реализующая диспетчеризацию, а все функции, связанные с анализом приоритетов, выбором очередного процесса для активизации, принятием решения о переключении на новый процесс и другие аналогичные функции выполняются вне микроядра.

В настоящее время именно операционные системы, построенные с использованием модели клиент-сервер и концепции микроядра, в наибольшей степени удовлетворяют требованиям, предъявляемым к современным ОС.

Микроядро не обязательно подразумевает небольшую систему. Использование модели клиент-сервер повышает надежность. Эта модель хорошо подходит для распределенных вычислений.

# 4. Гибридное ядро

Модифицированные микроядра (минимальная реализация основных функций ядра операционной системы компьютера), позволяющие для ускорения работы запускать «несущественные» части в пространстве ядра.

На практике концепция смешанного ядра часто подчёркивает не только достоинства, но и недостатки обоих типов ядер.

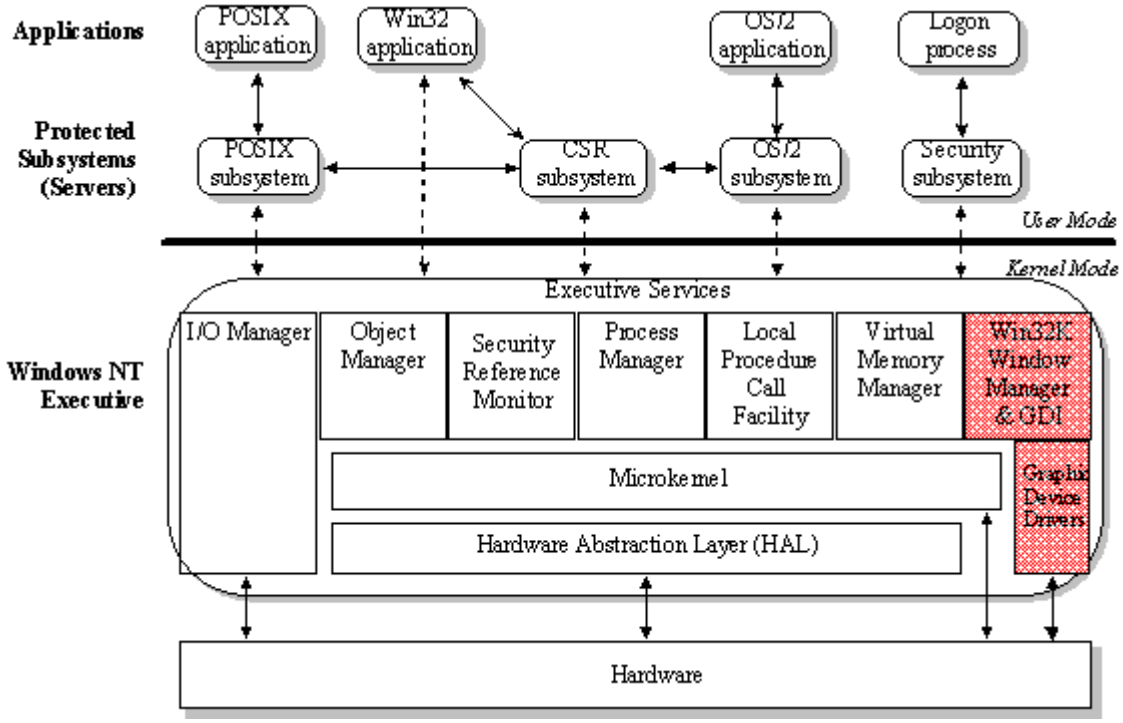



Рис – гибридное ядро Windows 4.0