Монолитные ОС

Вся ОС работает как единая программа в режиме ядра, написана в виде набора процедур, связанных вместе в одну большую исполняемую программу. Каждая процедура может свободно вызвать любую другую процедуру.

Системные вызовы, предоставляемые ОС, запрашиваются путем помещения параметров в четко определенное место (прим.: в стек), а затем выполняется инструкция *trap*. Эта инструкция переключает машину из пользовательского режима в режим ядра и передает управление операционной системе. Затем ОС извлекает параметры и определяет, какой сист. вызов должен быть выполнен.

Такая организация предполагает следующую базовую структуру операционной системы:

- Основная программа, которая вызывает требуемую служебную процедуру.
- Набор служебных процедур, выполняющих системные вызовы.
- Набор вспомогательных процедур, содействующих работе служебных процедур.

В этой модели для каждого системного вызова имеется одна ответственная за него служебная процедура, которая его и выполняет.

Многоуровневые системы

Обобщением подхода построения монолитных ядер является организация операционной системы в виде иерархии уровней, каждый из которых является надстройкой над нижележащим уровнем. Примеры ОС: ***THE*** и ***MULTICS***

ОС THE (Э. Дейкстра)

Уровень 0 - распределением процессорного времени, переключением между процессам. Уровень 0 обеспечивал основу многозадачности центрального процессора.

Уровень 1 управлял памятью.

Уровень 2 управлял связью каждого процесса с консолью пользователя. Над этим уровнем каждый процесс фактически имел свою собственную консоль оператора.

Уровень 3 управлял устройствами ввода-вывода и буферизацией информационных потоков в обоих направлениях. Над третьим уровнем каждый процесс мог работать с абстрактными устройствами IO, имеющими определенные свойства.

На **уровне 4** работали пользовательские программы, которым не надо было заботиться о процессах, памяти, консоли или управлении вводом-выводом.

Процесс системного оператора размещался на **уровне 5**.

MULTICS

Дальнейшее обобщение многоуровневой концепции было сделано в системе MULTICS. Вместо уровней для описания MULTICS использовались серии концентрических колец, где внутренние кольца обладали более высокими привилегиями по отношению к внешним. Когда процедуре из внешнего кольца требовалось вызвать процедуру внутреннего кольца, ей нужно было создать эквивалент системного вызова.

Микроядра

Замысел, положенный в основу конструкции микроядра, направлен на достижение высокой надежности за счет разбиения операционной системы на небольшие модули. *Микроядро* — запускается в режиме ядра, а все остальные запускаются в виде относительно слабо наделенных полномочиями обычных пользовательских процессов. Примеры ОС: ***Integrity***, ***K42***, ***L4***, ***PikeOS***, ***QNX***, ***Symbian*** и ***MINIX 3***.

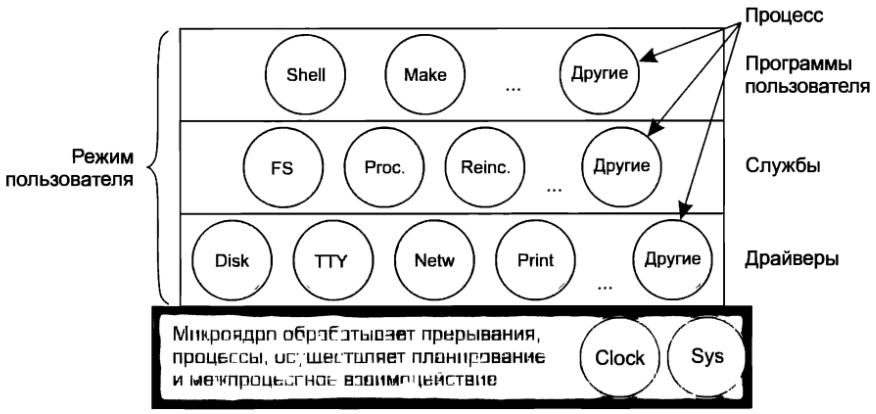

Рис. 1.22. Структура системы MINIX 3

Клиент-серверная архитектура

Вариация идеи микроядер выражается в обособлении двух классов процессов: серверов и клиентов. Достаточно часто самый нижний уровень представлен микроядром, но это не обязательно. Связь между клиентами и серверами часто организуется с помощью передачи сообщений.

Виртуальная машина

Идея изоляции пользовательского уровня от деталей вычислительной системы. Вместо прямого или косвенного обращения к ним все программы взаимодействуют только с виртуальной машиной – функциональным эквивалентом реальной, но реализуемой различными аппаратными и программными средствами. Представляет из себя гипервизор (бывает 2 типов: запускающийся на голом железе, или работающий поверх основной ОС). Гипервизор позволяет запускать гостевые ОС.

Экзоядро

Экзоядро – программа распределяющая физические ресурсы между несколькими виртуальными машинами.

Смешанная архитектура

Попытка совместить достоинства различных архитектур и/или компенсировать их недостатки. Встречается часто, т.к. чистые архитектуры на практике адаптированы к конкретным реальным условиям.