Введение в операционные системы

Наумов Д.А., доц. каф. КТ

Операционные системы и системное программное обеспечение, 2019

Содержание лекции

1. Структура вычислительной системы
2. История развития операционных систем
3. Основные понятия, концепции ОС
4. Архитектурные особенности ОС

Операционная система

это программа, которая обеспечивает возможность рационального использования оборудования компьютера удобным для пользователя образом.

Уровни программного обеспечения компьютерной системы

История развития операционных систем

- 1945-1955 - компьютеры работали без операционных систем
- 1955-1965 - первые системы пакетной обработки (автоматизировали запуск одной программ за другой и тем самым увеличивали коэффициент загрузки процессора)
- 1965-1980 - многозадачность (способ организации вычислительного процесса, при котором на одном процессоре попеременно выполняются несколько задач) и система разделения памяти
- Системы разделения времени - вариант многозадачности, при котором у каждого пользователя есть свой диалоговый терминал (фактически - многопользовательская система)
- мини-компьютеры (первый - в 1961г.), система MULTICS (предшественник UNIX)
- разновидности несовместимых UNIX (System V, BSD). Разработка стандарта POSIX (минимальный интерфейс системного вызова, который должны поддерживать системы UNUX).

История развития операционных систем

- В 1974г. был выпущен центральный процессор Intel 8080, для него была создана операционная система CP/M. В 1977г. она была переработана для других процессоров, например Zilog Z80.
- В начале 80-х была разработана система MS-DOS, и стала основной системой для микрокомпьютеров.
- 80-е годы - графический интерфейс пользователя (GUI, Graphical User Interface).
- С 1985 года стала выпускаться Windows, в то время она была графической оболочкой к MS-DOS вплоть до 1995г., когда вышла Windows 95.
- IBM и Microsoft начинали совместно разрабатывать операционную систему OS/2. Она должна была поддерживать вытесняющую многозадачность, виртуальную память, графический пользовательский интерфейс, виртуальную машину для выполнения DOS-приложений. Первая версия вышла 1987г.
- отказ от OS/2 в пользу Windows NT. Первая версия вышла в 1993г.

В середине 80-х стали бурно развиваться сети персональных компьютеров, работающие под управлением сетевых или распределенных операционных систем.

1. **Сетевая операционная система** не имеет отличий от операционной системы однопроцессорного компьютера. Она обязательно содержит программную поддержку для сетевых интерфейсных устройств (драйвер сетевого адаптера), а также средства для удаленного входа в другие компьютеры сети и средства доступа к удаленным файлам.
2. **Распределенная операционная система**, напротив, представляется пользователям простой системой, в которой пользователь не должен беспокоиться о том, где работают его программы или где расположены файлы, все это должно автоматически обрабатываться самой операционной системой.

- В 1987г. была выпущена операционная система MINIX (прототип LINUX), она была построена на схеме микро ядра.
- В 1991г. была выпущена LINUX, в отличии от микроядерной MINIX она стала монолитной. Чуть позже вышла FreeBSD

Операционная система как виртуальная машина

ОС предоставляет пользователю виртуальную машину, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальную машину.

Операционная система как менеджер ресурсов

Чтобы несколько программ могло работать с одним ресурсом (процессор, память), необходима система управления ресурсами.

Способы распределения ресурса:

- Временной - когда программы используют его по очереди, например, так система управляет процессором.
- Пространственный - программа получает часть ресурса, например, так система управляет оперативной памятью и жестким диском.

Системные вызовы (system calls)

это интерфейс между операционной системой и пользовательской программой. Они создают, удаляют и используют различные объекты, главные из которых – процессы и файлы. Пользовательская программа запрашивает сервис у операционной системы, осуществляя системный вызов.

При системном вызове задача переходит в привилегированный режим или режим ядра (kernel mode). Поэтому системные вызовы иногда еще называют программными прерываниями, в отличие от аппаратных прерываний, которые чаще называют просто прерываниями.
В большинстве операционных систем системный вызов осуществляется командой программного прерывания (INT).
Программное прерывание – это синхронное событие, которое может быть повторено при выполнении одного и того же программного кода.

Интерфейс прикладного программирования, API (Application Programming Interface)

интерфейс между операционной системой и программами, определяемый набором системных вызовов.

Некоторые системные вызовы в POSIX:

- fork - создание нового процесса
- exit - завершение процесса
- open - открывает файл
- close - закрывает файл
- read - читает данные из файла в буфер
- write - пишет данные из буфера в файл

В UNIX вызовы почти один к одному идентичны библиотечным процедурам, которые используются для обращения к системным вызовам.

Win32 API отделен от системных вызовов. Это позволяет в разных версиях менять системные вызовы, не переписывая программы. Поэтому непонятно является ли вызов системным (выполняется ядром), или он обрабатывается в пространстве пользователя. Некоторые системные вызовы в Windows (Win32 API):

- CreatProcess (fork) - создание нового процесса
- ExitProcess (exit) - завершение процесса
- CreatFile (open) - открывает файл
- CloseHandle (close) - закрывает файл

В Win32 API существует более 1000 вызовов. Такое количество связано и с тем, что графический интерфейс пользователя UNIX запускается в пользовательском режиме, а в Windows встроен в ядро. Поэтому Win32 API имеет много вызовов для управления окнами, текстом, шрифтами т.д.

Прерывание (hardware interrupt)

это событие, генерируемое внешним (по отношению к процессору) устройством.

- важный тип аппаратных прерываний – прерывания таймера, которые генерируются периодически через фиксированный промежуток времени. Прерывания таймера используются операционной системой при планировании процессов.
- Каждый тип аппаратных прерываний имеет собственный номер, однозначно определяющий источник прерывания.
- Аппаратное прерывание – это асинхронное событие, то есть оно возникает вне зависимости от того, какой код исполняется процессором в данный момент.
- Обработка аппаратного прерывания не должна учитывать, какой процесс является текущим.

Исключительная ситуация (exception)

событие, возникающее в результате попытки выполнения программой команды, которая по каким-то причинам не может быть выполнена до конца.

Исключительные ситуации, как и системные вызовы, являются синхронными событиями, возникающими в контексте текущей задачи.

- исправимые (например, отсутствие информации в памяти)
- неисправимые (деление на ноль)

Файл

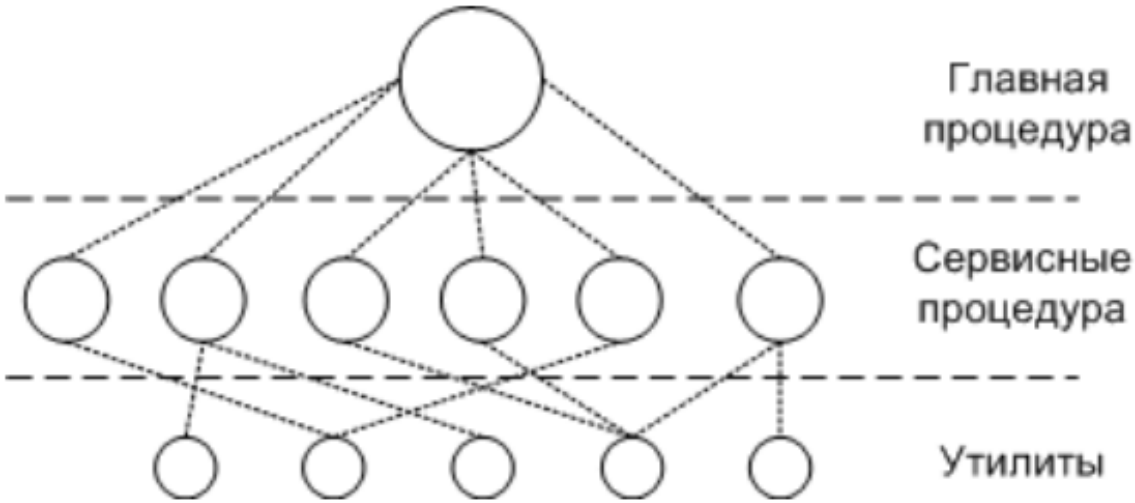
(обычно) именованная часть пространства на носителе информации.

Главная задача файловой системы (file system) – скрыть особенности ввода-вывода и дать программисту простую абстрактную модель файлов, независимых от устройств.

Процессы, нити

наиболее фундаментальные понятия, будут рассмотрены далее

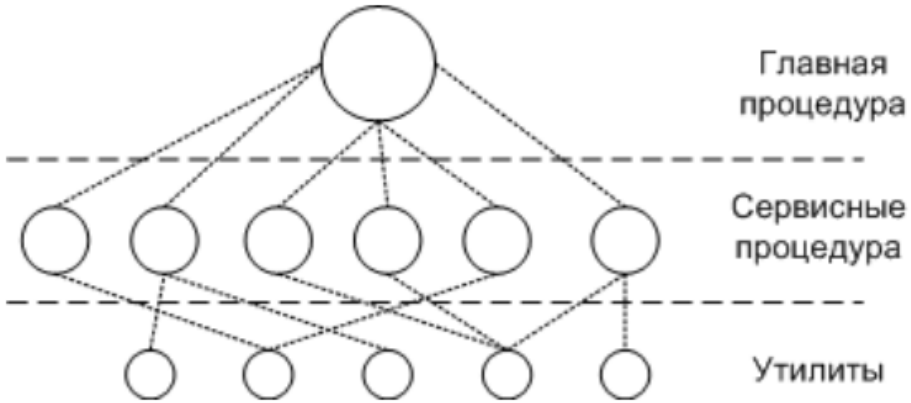
Система с монолитным ядром



Простая модель монолитной системы

- главная программа, которая вызывает требуемые сервисные процедуры.
- набор сервисных процедур, реализующих системные вызовы
- набор утилит, обслуживающих сервисные процедуры

Для каждого системного вызова имеется одна сервисная процедура. Утилиты выполняют функции, которые нужны нескольким сервисным процедурам.

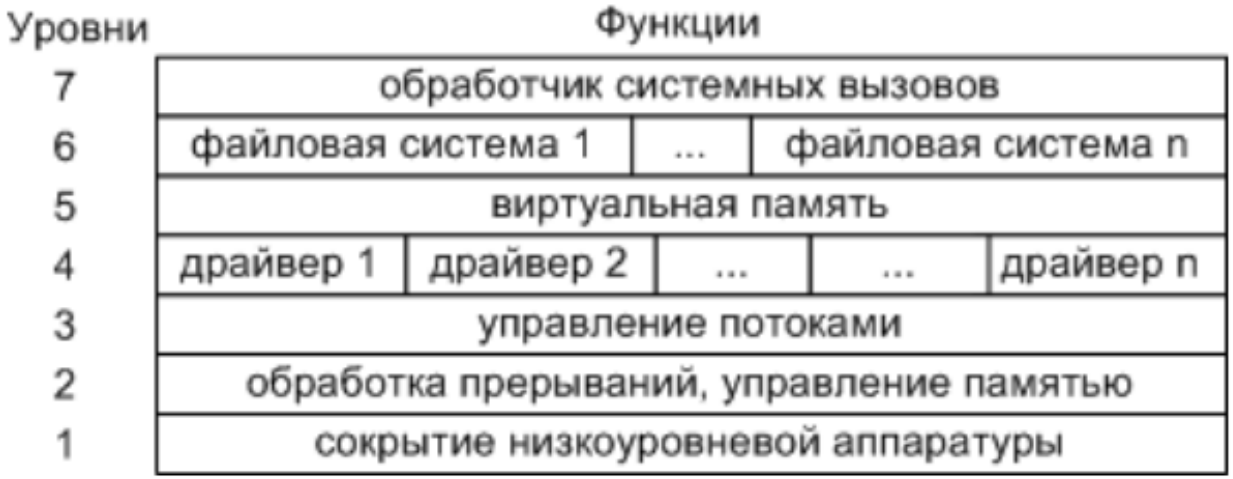


Простая модель монолитной системы

Этапы обработки вызова:

- Принимается вызов
- Выполняется переход из режима пользователя в режим ядра
- ОС проверяет параметры вызова для того, чтобы определить, какой системный вызов должен быть выполнен
- После этого ОС обращается к таблице, содержащей ссылки на процедуры, и вызывает соответствующую процедуру

Многоуровневые системы



Пример структуры многоуровневой 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.

Многоуровневые системы

Уровни	Функции				
7	обработчик системных вызовов				
6	файловая система 1	...	файловая система n		
5	виртуальная память				
4	драйвер 1	драйвер 2	...	...	драйвер n
3	управление потоками				
2	обработка прерываний, управление памятью				
1	сокрытие низкоуровневой аппаратуры				

Пример структуры многоуровневой системы

Преимущества

- Высокая производительность

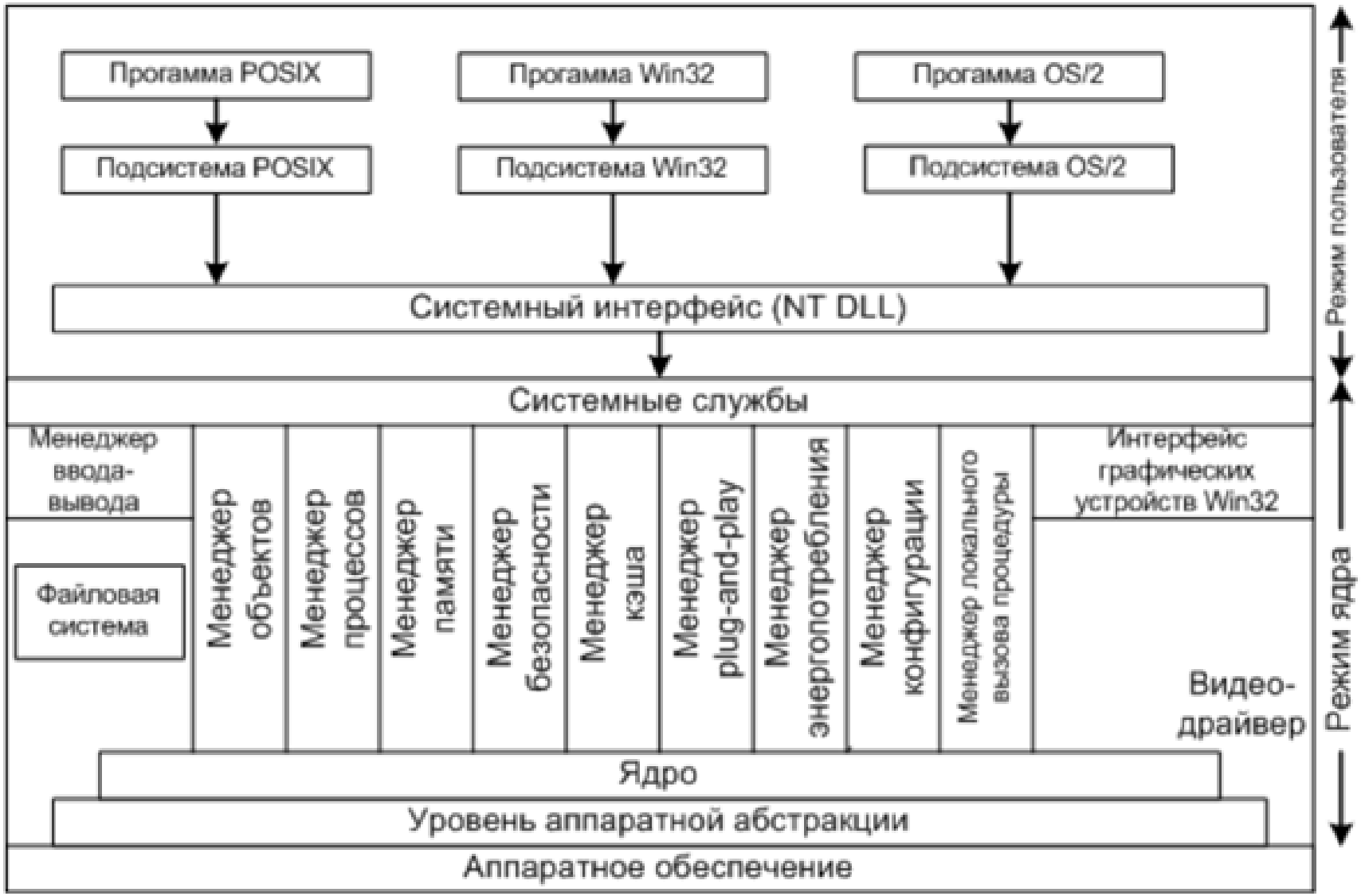
Недостатки

- Большой код ядра, и как следствие большое содержание ошибок
- Ядро плохо защищено от вспомогательных процессов

Пример многоуровневой структуры OS UNIX

Структура ОС UNIX

Пример структуры OS Windows NT



Структура Windows 2000

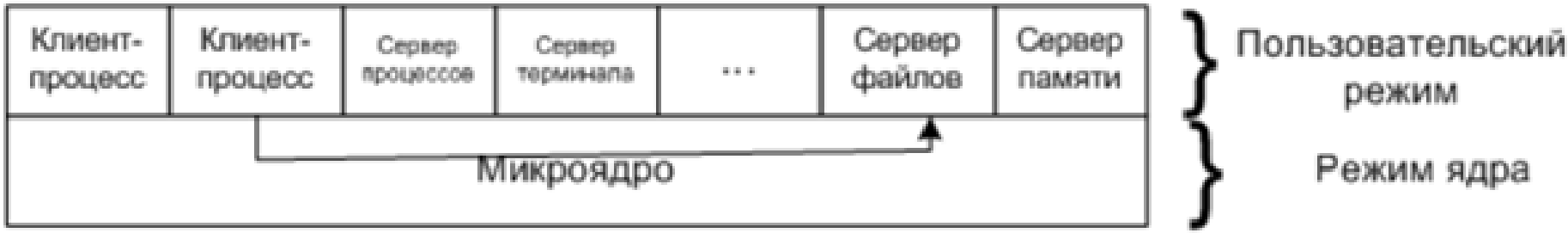
Модель экзоядра

Принцип экзоядра: все отдать пользовательским программам.

Например, зачем нужна файловая система?

- каждая пользовательская программа сможет иметь свою файловую систему.
- операционная система должна обеспечить безопасное распределение ресурсов среди соревнующихся за них пользователей.

Микроядерная архитектура (Эта модель является средним между двумя предыдущими моделями)



Модель клиент-сервер

В этой модели вводятся два понятия:

- Серверный процесс (который обрабатывает запросы)
- Клиентский процесс (который посылает запросы)

Недостатки В задачу ядра входит только управление связью между клиентами и серверами.

Микроядерная архитектура (Эта модель является средним между двумя предыдущими моделями)

Модель клиент-сервер

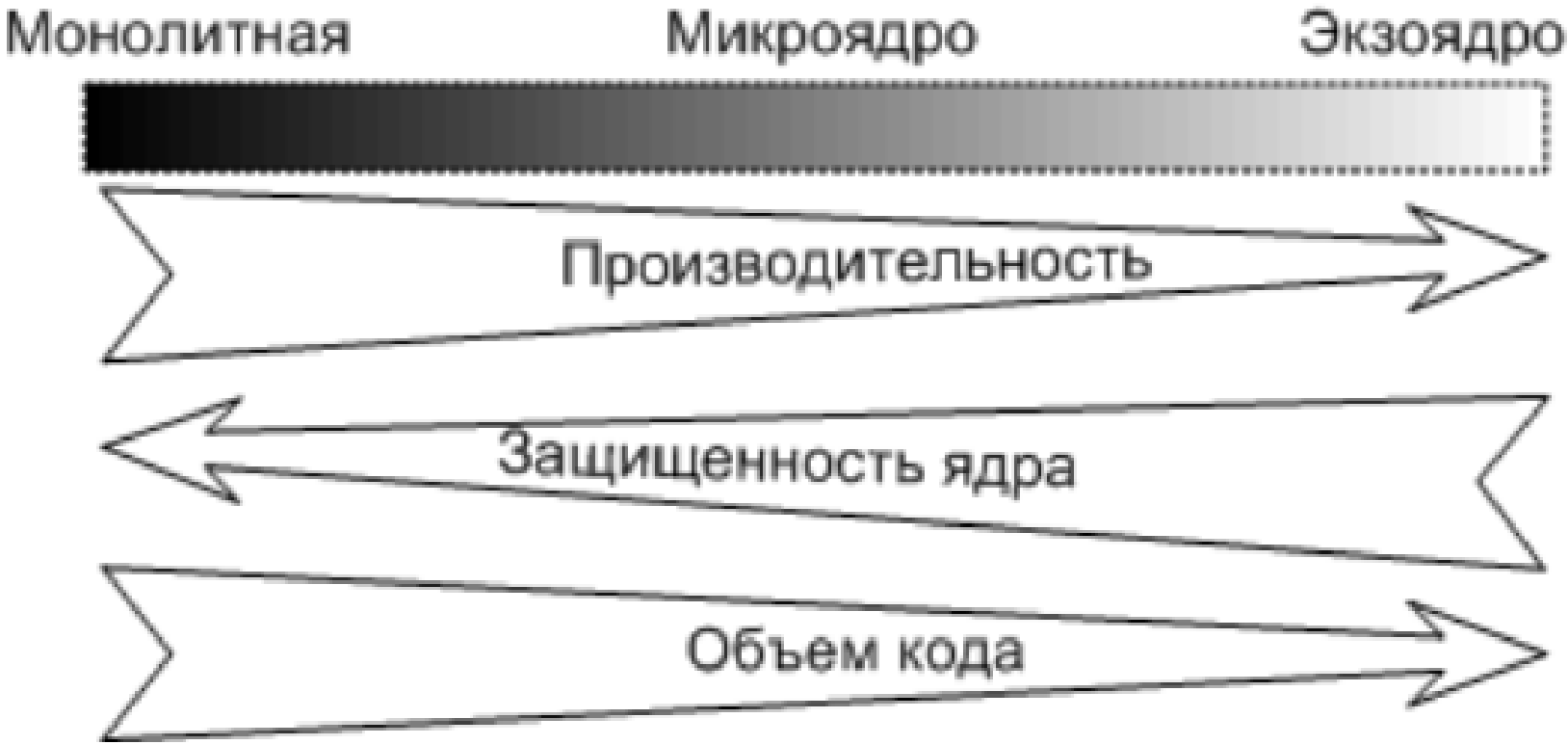
Преимущества

- Малый код ядра и отдельных подсистем, и как следствие меньшее содержание ошибок.
- Ядро лучше защищено от вспомогательных процессов.
- Легко адаптируется к использованию в распределенной системе

Недостатки

- Уменьшение производительности.

Сравнение моделей



Сравнения моделей.