

# Управление вводом - выводом

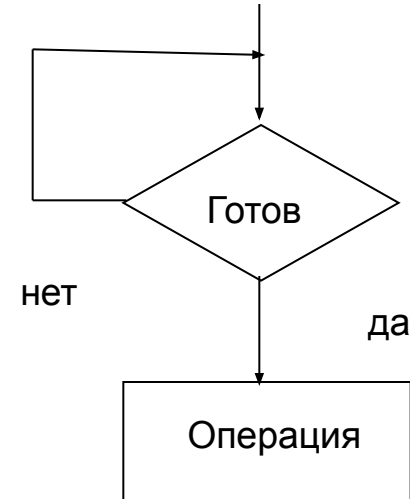
- Функции подсистемы ввода-вывода
- Принципы организации ввода-вывода
- Буферизация
- Ввод и вывод в ОС UNIX
- Ввод и вывод Windows 2000

# Функции подсистемы ввода-вывода

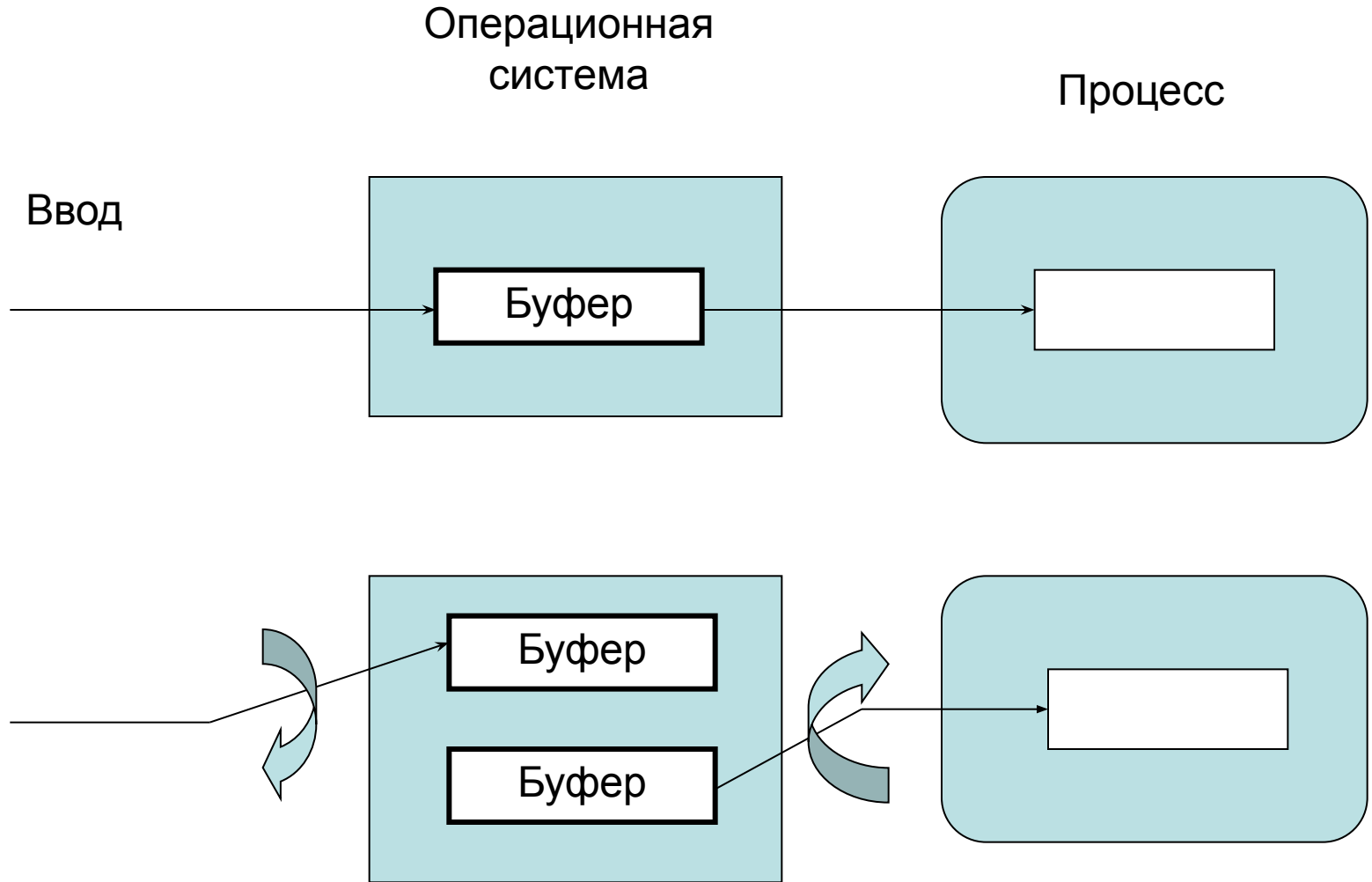
- Организация параллельной работы устройств ввода-вывода и процессора
- Согласование скоростей обмена и кэширование данных
- Разделение устройств между процессами
- Обеспечение удобного логического интерфейса между устройствами и остальной частью системы
- Поддержка широкого спектра драйверов с возможностью простого включения в систему нового драйвера
- Динамическая загрузка и выгрузка драйверов
- Поддержка нескольких файловых систем
- Поддержка синхронных и асинхронных операций ввода-вывода

# Принципы организации ввода-вывода

- Синхронный обмен
  - Обмен по запросу
  - Обмен с ожиданием готовности
- Асинхронный обмен
- Прямой доступ к памяти

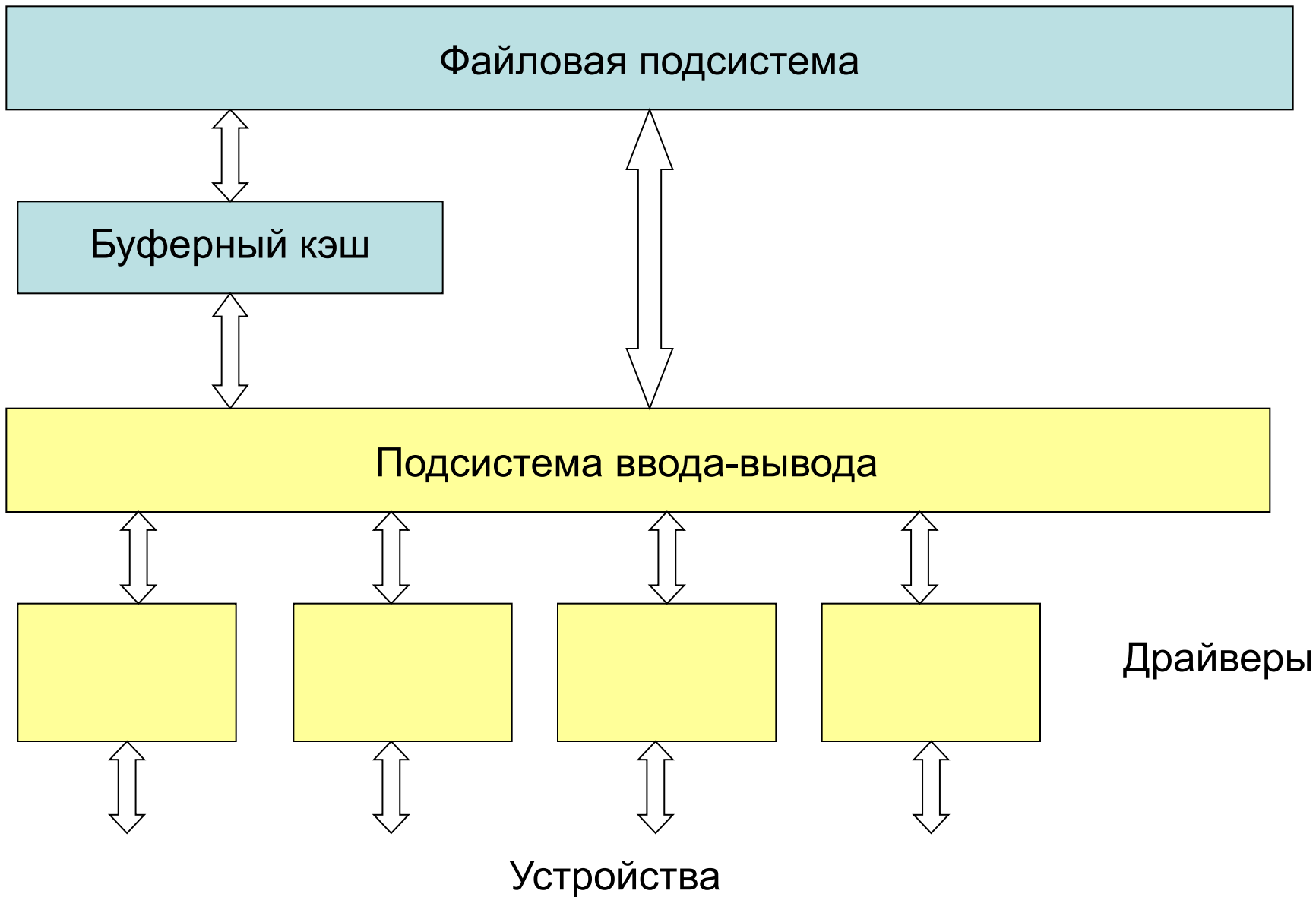


# Буферизация



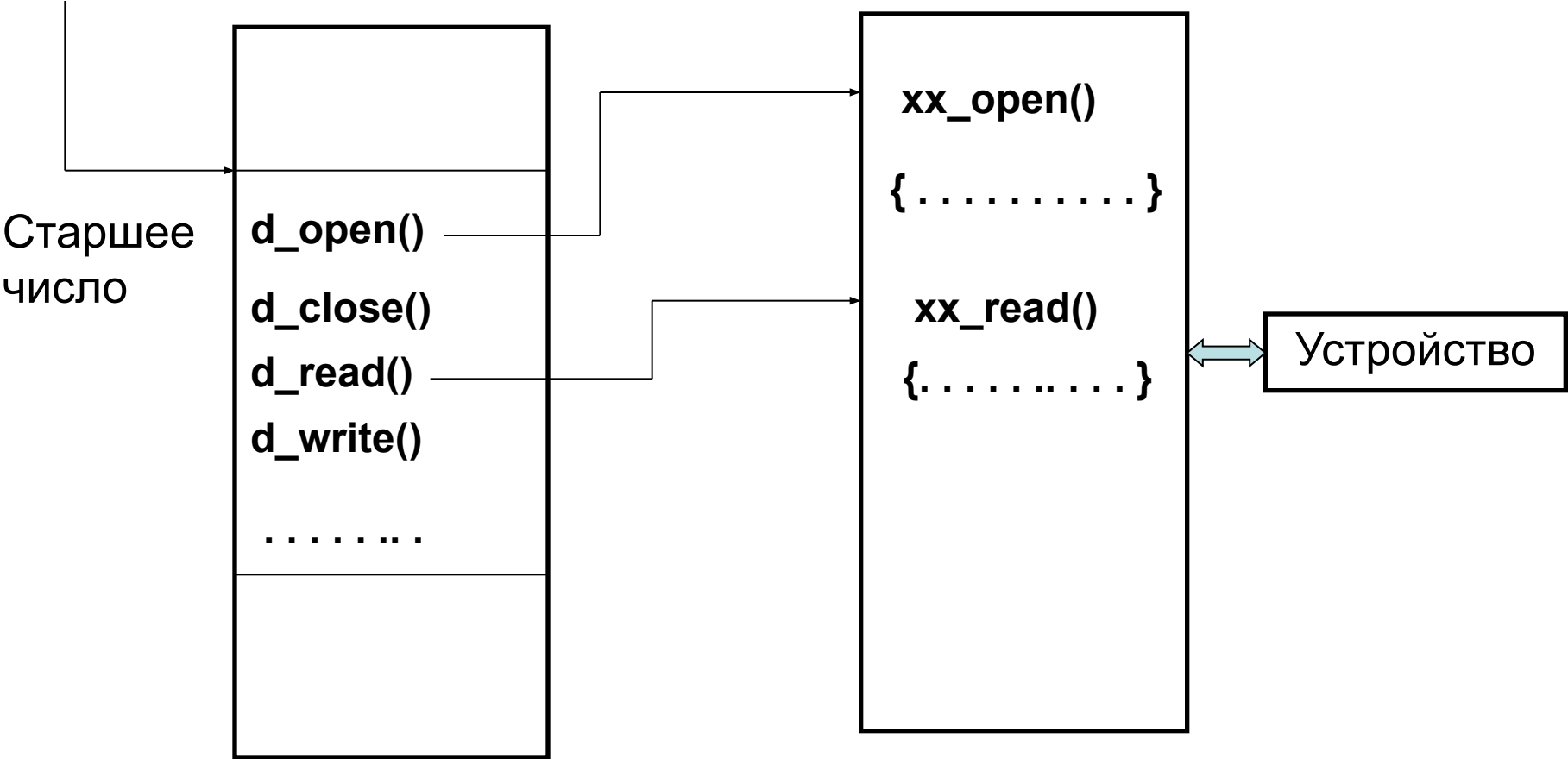
# Ввод и вывод в ОС UNIX

- Типы драйверов
  - Символьные драйверы
  - Блочные драйверы
  - Драйверы низкого уровня
  - Драйверы псевдоустройств (/dev/kmem, /dev/ksyms, /dev/mem, /dev/null, /dev/zero)
- Адресация драйвера
  - Старший номер – major number
  - Младший номер – minor number
- Коммутаторы устройств – bdevsw и cdevsw



Коммутатор устройств

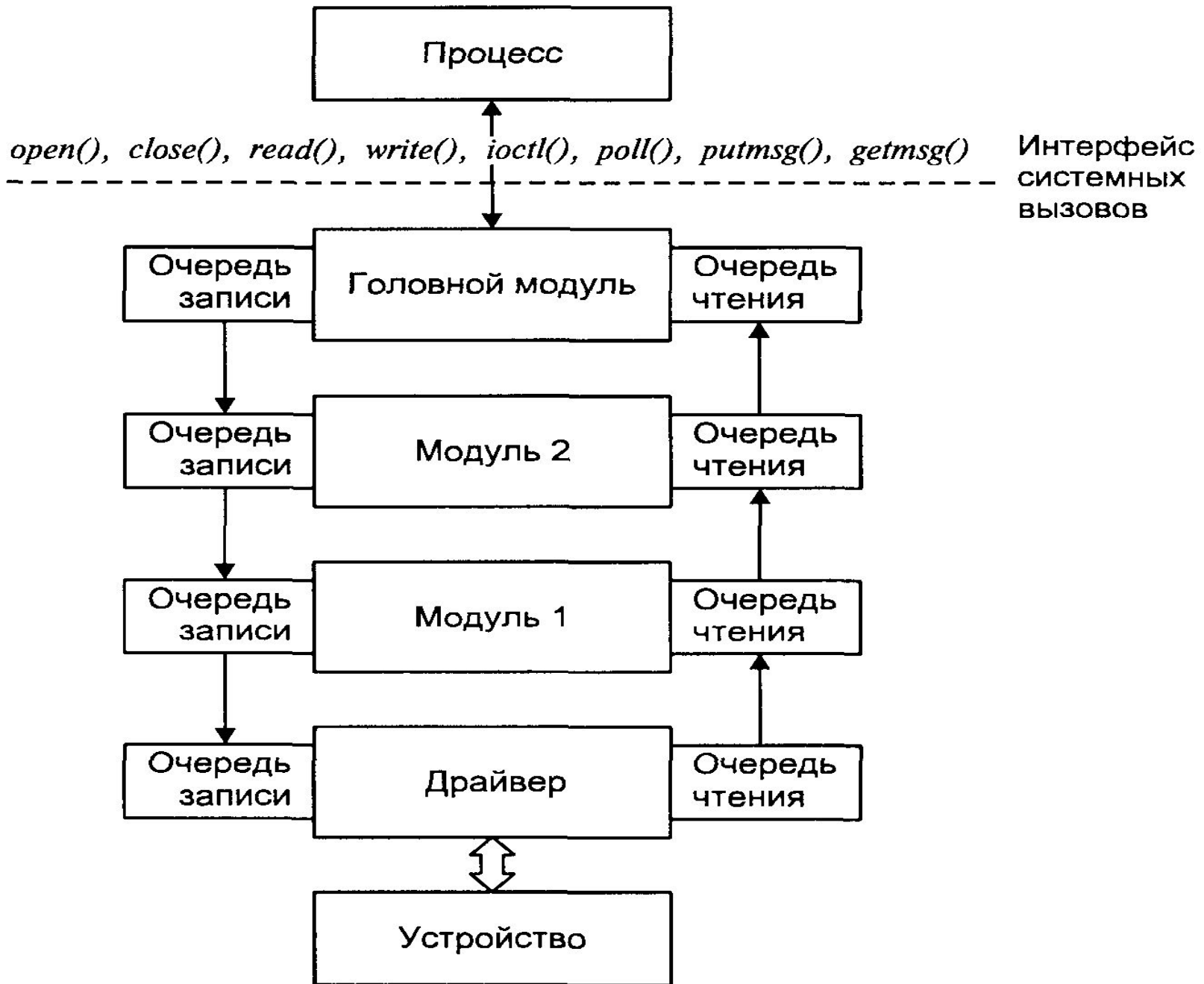
Драйвер

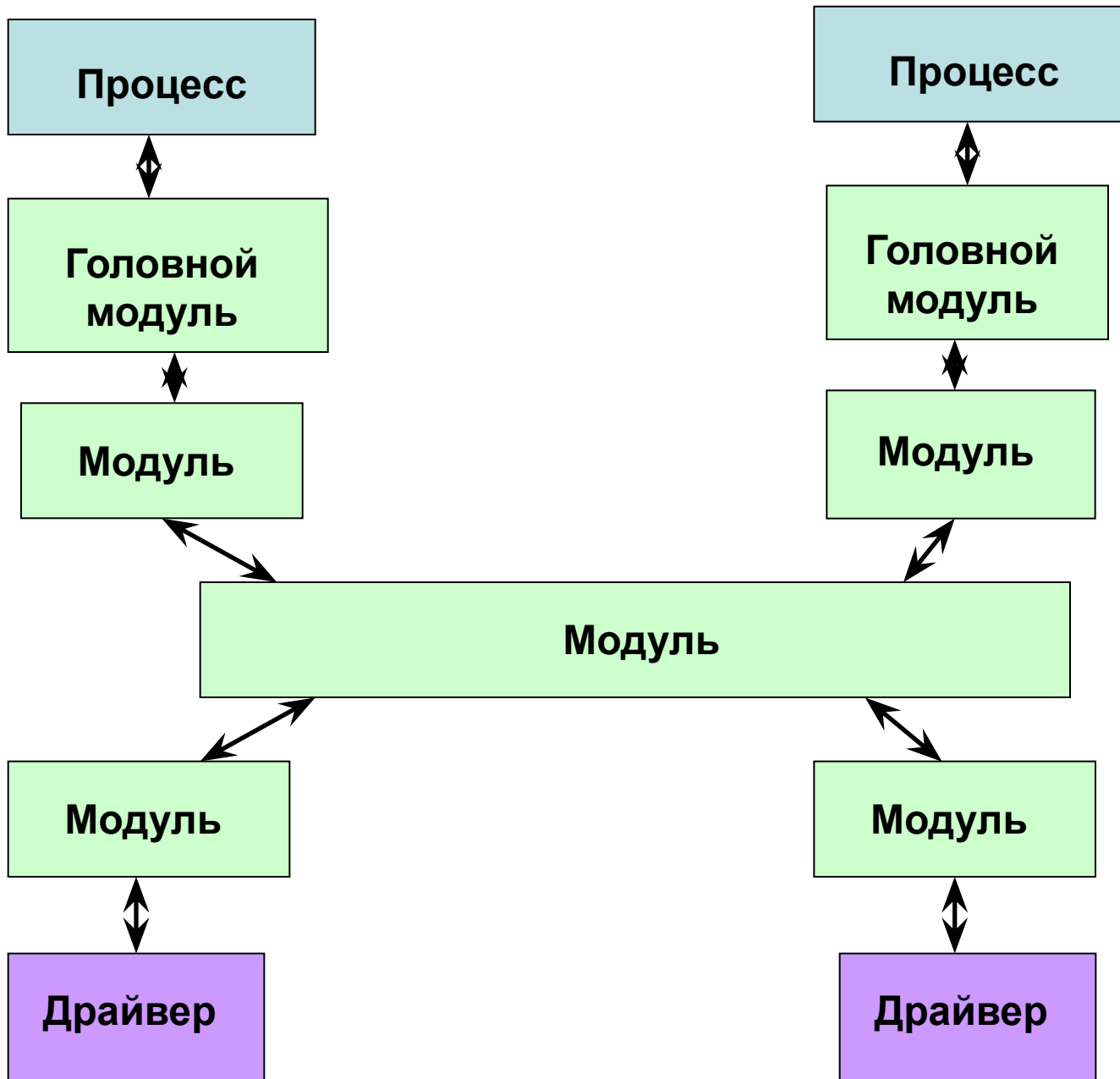




# Точки входа

- `xxopen()` – открытие устройства
- `xxclose()` – закрытие устройства
- `xxread()` – чтение данных для символьного устройства
- `xxwrite()` – запись данных для символьного устройства
- `xxioctl()` – управление символьным устройством
- `xxintr()` – обработка прерываний
- `xxstrategy()` – общая точка входа для операций блочного ввода-вывода, ввод-вывод инициируются прерываниями





# Ввод и вывод Windows 2000

**Диспетчер ввода-вывода**

**Диспетчер кэша**

**Драйверы файловой  
системы**

**Драйверы сети**

**Драйверы аппаратуры**

# Основные модули

- Диспетчер кэша. Управляет кэшированием всей подсистемы ввода-вывода.
  - Отложенная запись. Записи обновляются только в кэше. На диск записывается только последняя версия обновления.
  - Отложенное подтверждение. Работа с транзакциями.
- Драйверы файловой системы. Работа с томами.
- Драйверы сети. Интегрированные сетевые возможности и поддержка распределенных приложений.
- Драйверы аппаратуры. Работа с регистрами периферийных устройств.

# Режимы ввода-вывода

- Асинхронный режим – приложение инициирует операцию и продолжает работу. Способы оповещения о завершении.
  - Сигнал объекту устройства ядра.
  - Сигнал объекту события ядра.
  - Оповещение о вводе-выводе.
  - Порты завершения ввода-вывода
- Синхронный режим – приложение блокируется до завершения операции.

# Типы драйверов (WDM)

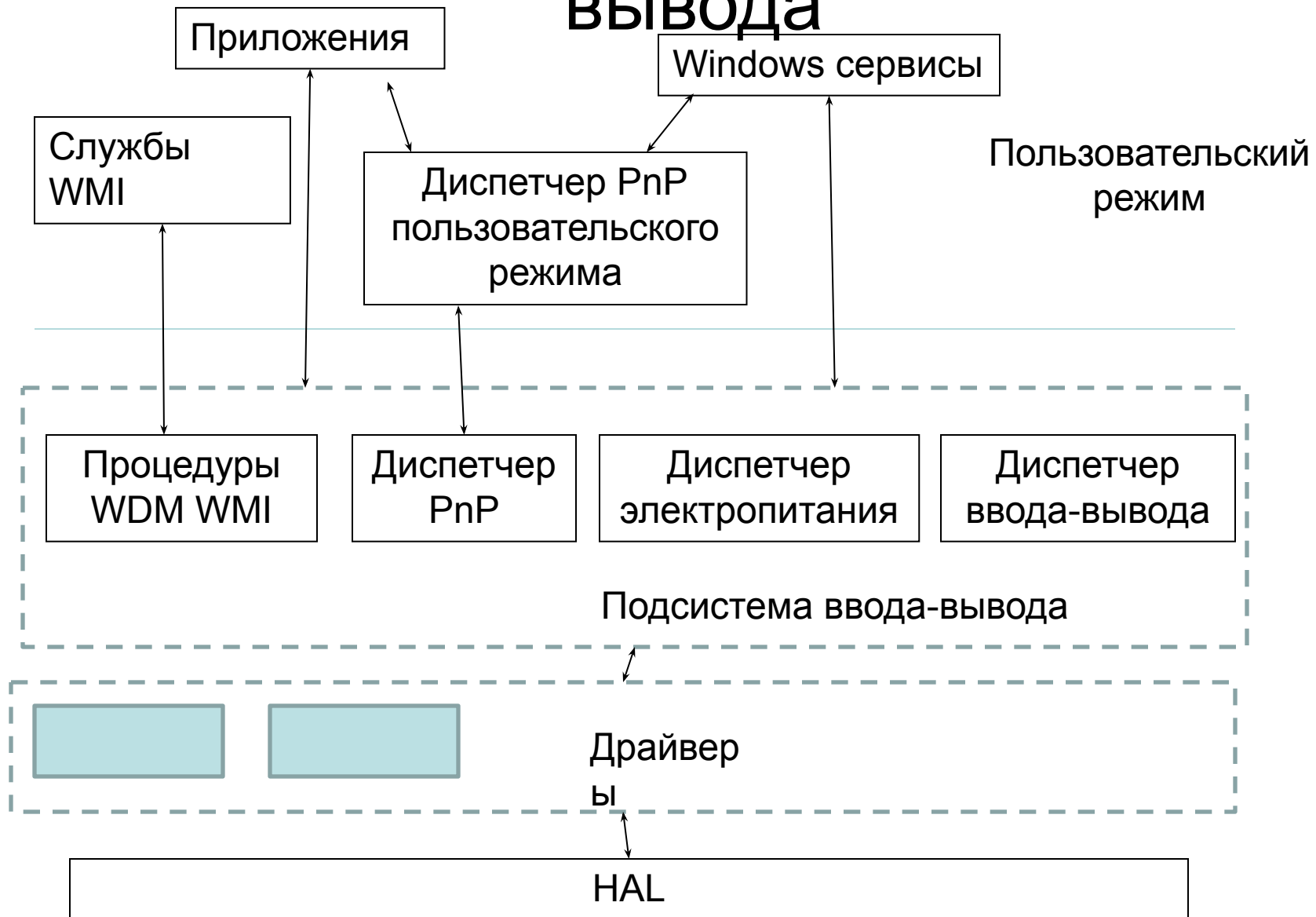
- Драйверы пользовательского режима (UMD)
  - Драйверы виртуальных устройств (VDD)
  - Драйверы принтеров
- Драйверы режима ядра (KMD)
  - Драйверы файловой системы
  - Унаследованные драйверы
  - Драйверы видеоадаптеров
  - Драйверы потоковых устройств
  - WDM - драйверы

# Уровни драйверов

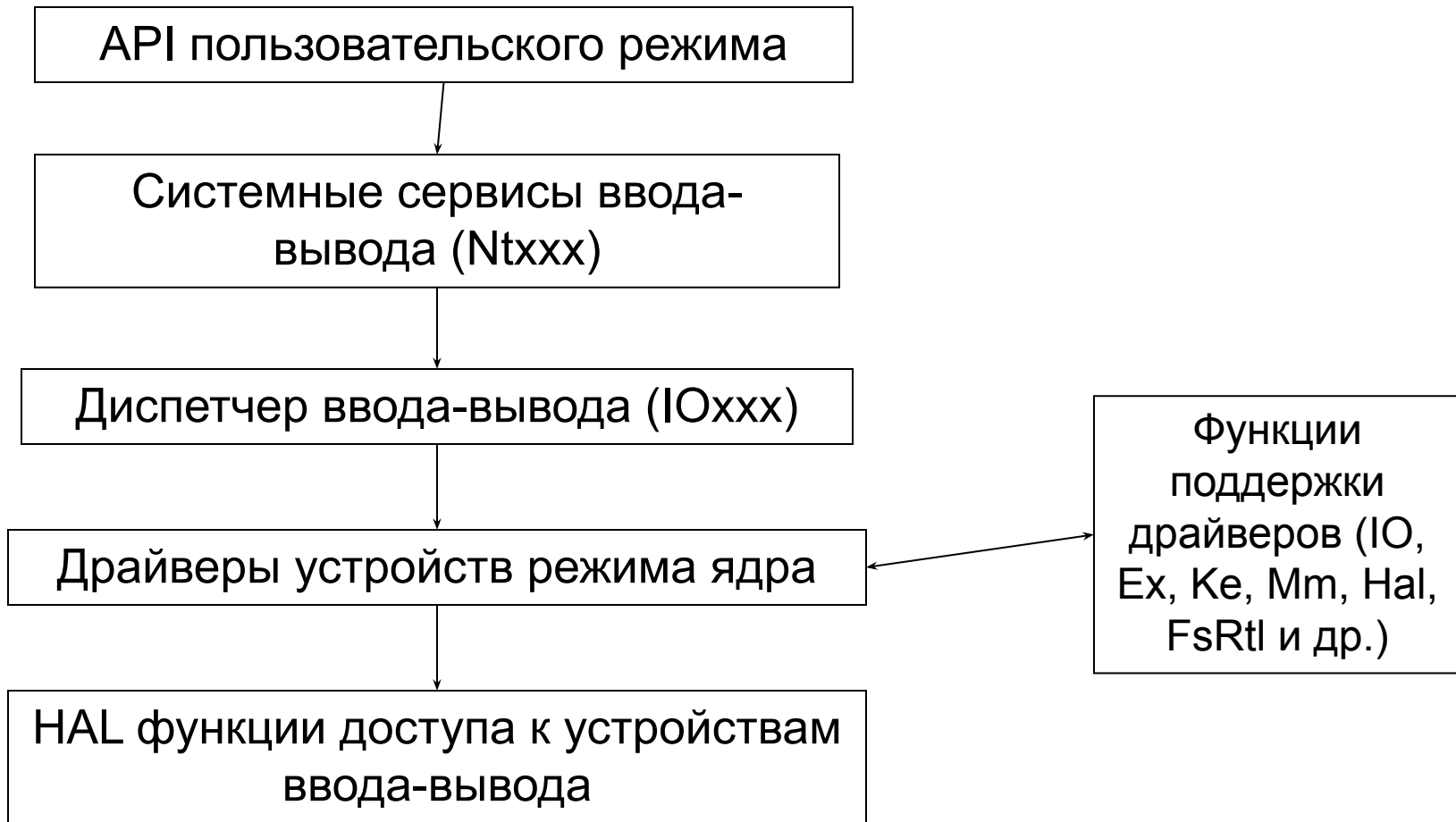
- Уровни типа драйверов
  - Шинные драйверы
  - Фильтр – драйверы
  - Функциональные драйверы
- Фильтр – драйверы
  - Фильтр-драйверы шины
  - Фильтр-драйверы устройства и классовый фильтр-драйвер
  - Функциональный драйвер
  - Вышестоящие фильтр-драйверы устройства и классовый фильтр-драйвер



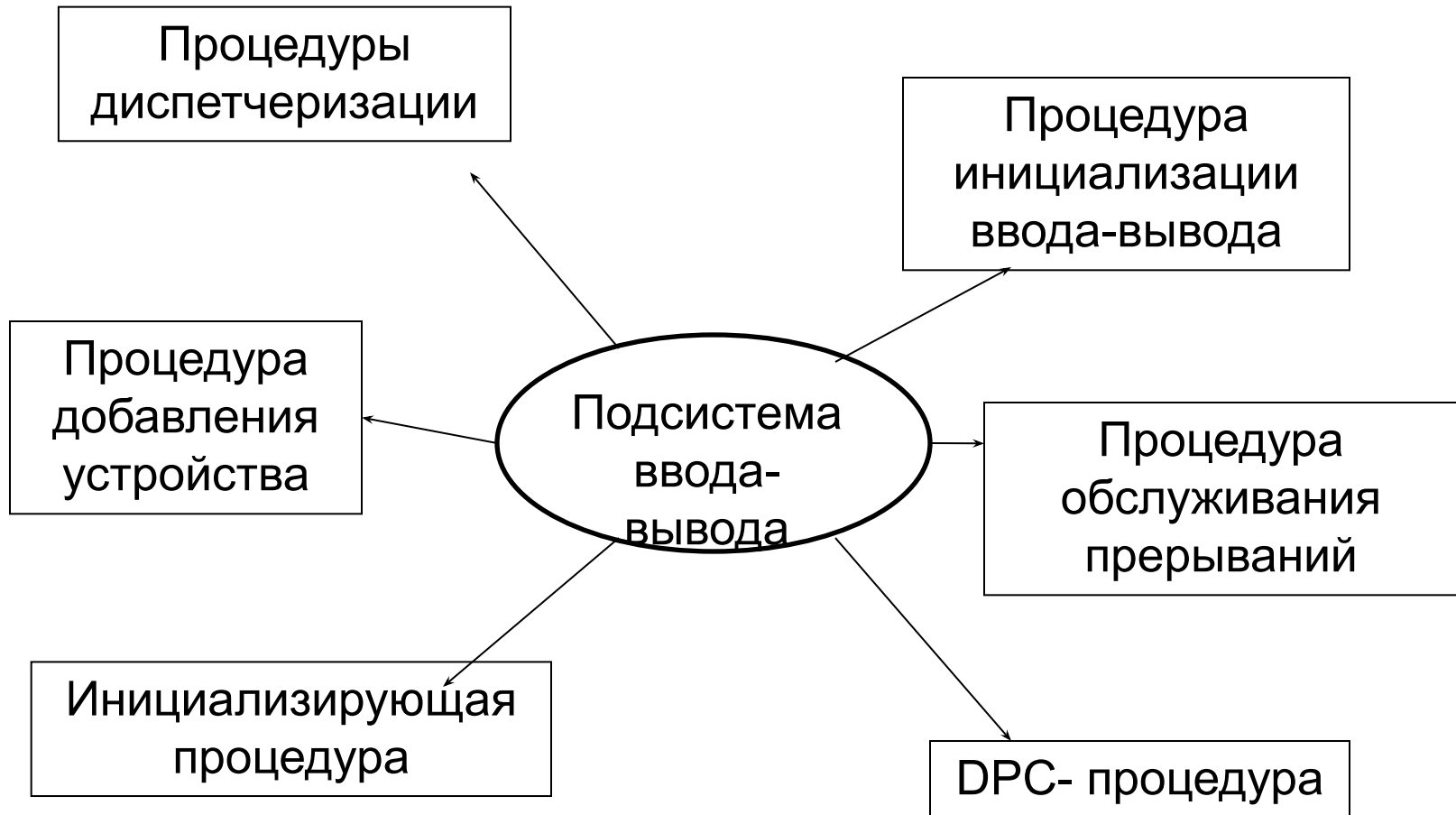
# Компоненты подсистемы ввода-вывода

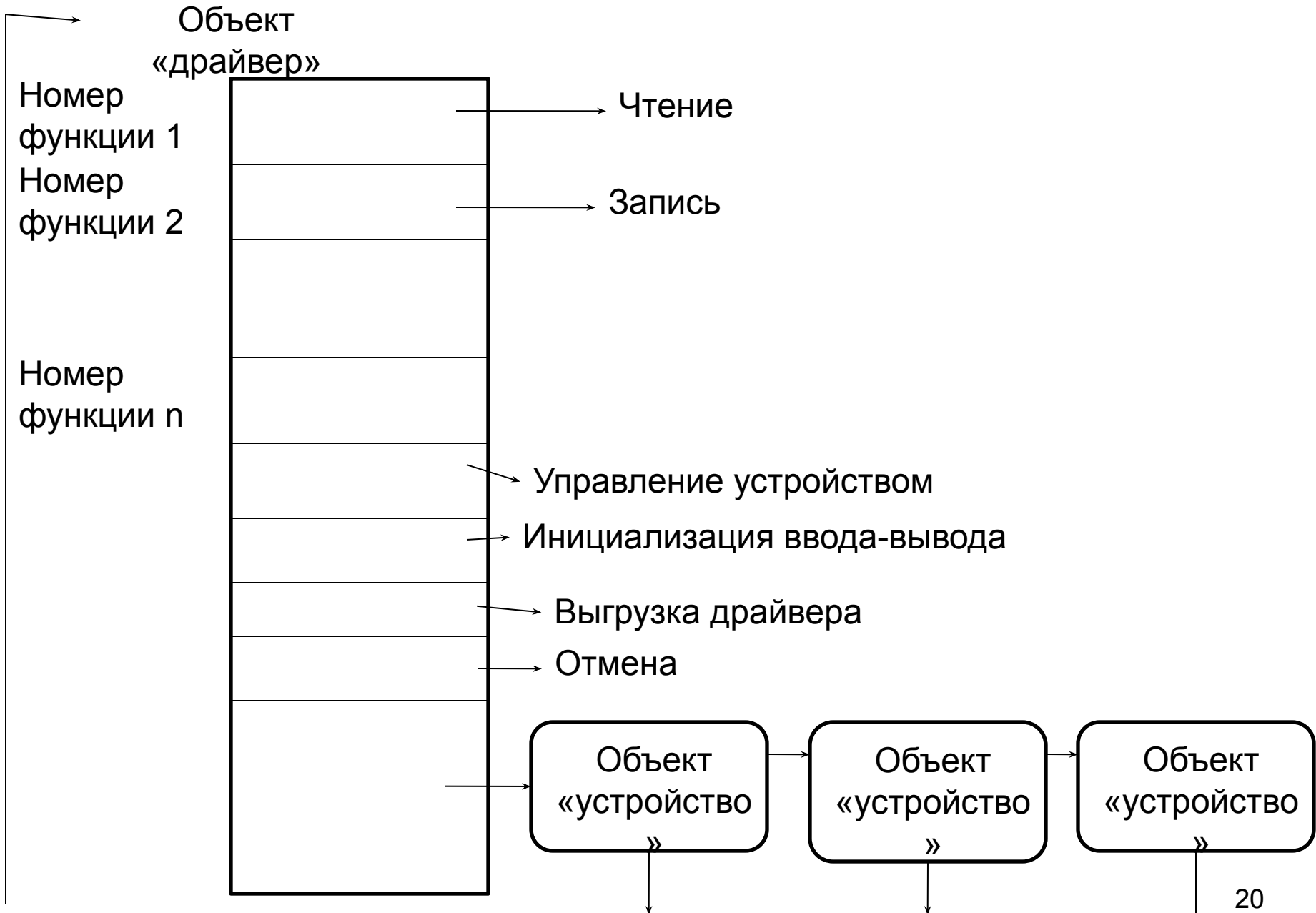


# Базовая схема обработки запроса ввода-вывода



# Основные процедуры драйвера





```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject,
                   IN PUNICODE_STRING
                       RegistryPath);
{ DriverObject->DriverUnload = AddDevice;
  DriverObject->DriverExtension->AddDevice = AddDevice;
  DriverObject->MajorFunction[IRP_MJ_PNP] = DispatchPnP;
  DriverObject-> MajorFunction[IRP_MJ_POWER] =
    DispatchPower;
  . . . . .
  return STATUS_SUCCESS;
}
```

```
void XxxUnload(IN PDRIVER_OBJECT DriverObject);
```

```
NTSTATUS XxxAddDevice(IN PDRIVER_OBJECT DriverObject,  
                    IN PDEVICE_OBJECT  
PhysicalDeviceObject);
```

## Основные задачи функции

- Вызывается функция IoCreateDevice
- Регистрируются один или несколько интерфейсов устройств
- Иницируется объект расширения устройства и поля Flags объекта устройства
- Вызывается IoAttachDeviceToDeviceStack для включения нового устройства

```
PDEVICE_OBJECT IoAttachDeviceToDeviceStack(  
    IN PDEVICE_OBJECT SourceDevice,  
    IN PDEVICE_OBJECT TargetDevice);
```

# Недостатки WDM

- Сложность написания драйверов
- Большое количество разных моделей минипортов
- Большинство драйверов могут выполняться только в режиме ядра
- Обилие различных драйверных моделей приводит к трудности при тестировании и верификации кода драйверов

# Цели WDF

- Простота написания драйверов и гибкость для быстрой адаптации к новым возможностям системы
- Драйверная модель не должна зависеть от основных компонентов ОС
- Драйверная модель должна обеспечивать, чтобы один исполняемый файл драйвера работал на разных версиях ОС
- Драйверная модель должна быть расширяемой
- Драйверная модель должна позволять большинству драйверов работать в пользовательском режиме
- Драйверная модель должна поддерживать написание драйверов на языке высокого уровня

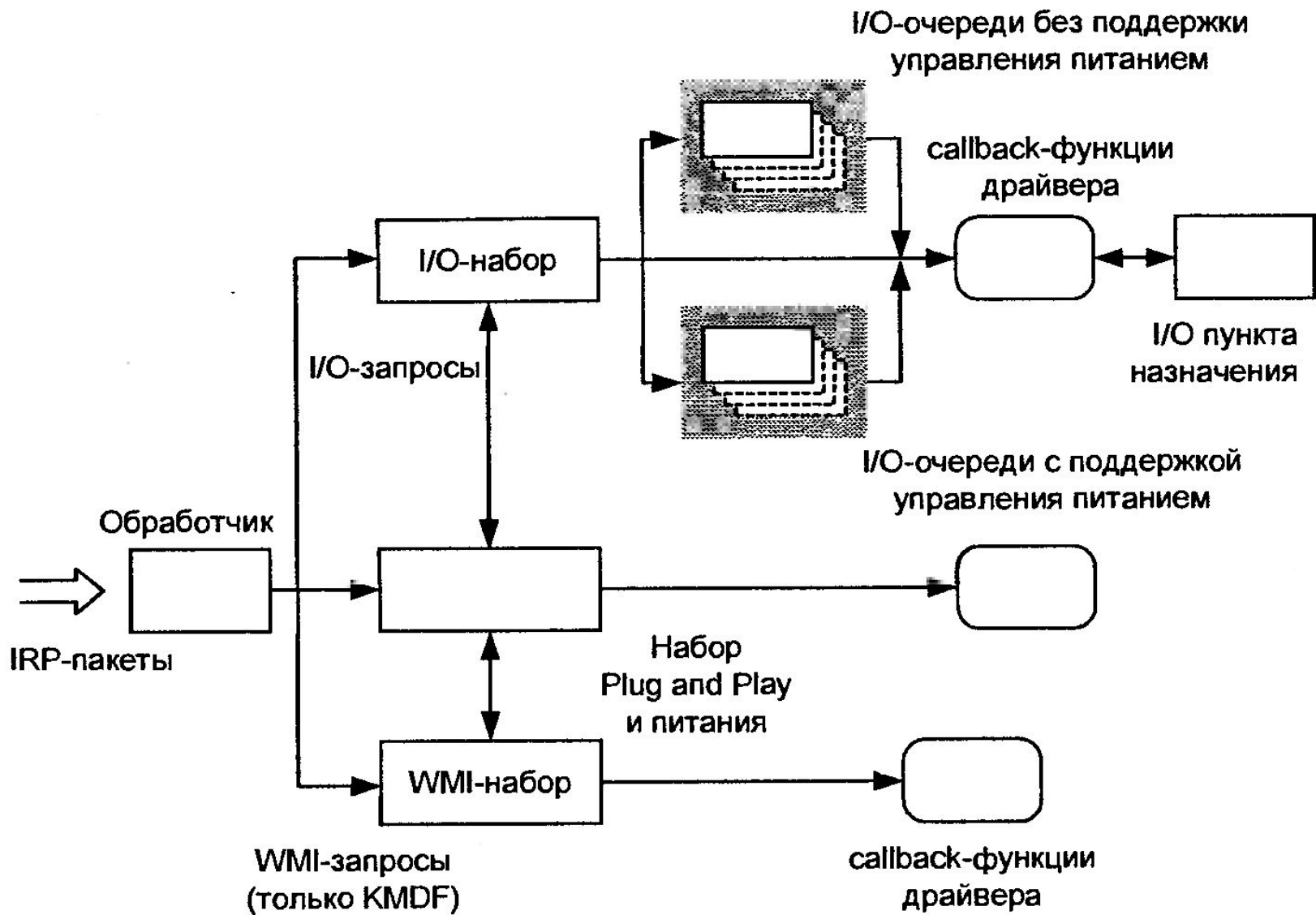


# Объекты KMDF

- WDFDRIVER – объект драйвера
- WDFDEVICE – объект устройства
- WDFQUEUE – очередь запросов ввода-вывода
- WDFINTERRUPT – представляет ресурсы прерывания
- WDFREQUEST – запрос ввода-вывода
- WDFMEMORY – буфер для запроса ввода-вывода

# UMDF объекты

- IWDFObject – базовый тип WDF-объекта
- IWDFDriver – объект драйвера
- IWDFDevice – объект устройства
- IWDFFile – объект файла
- IWDFIoQueue – очередь запросов ввода-вывода
- IWDFIoTarget – целевой драйвер запроса ввода-вывода
- IWDFMemory – предоставляет доступ к области памяти



# KMDF поддерживает следующие типы драйверов

- Шинные драйверы для стека устройств PnP
- Фильтр – драйверы для устройств PnP
- Legacy – драйверы для устройств, не включенных в стек PnP
- Функциональные драйверы для устройств PnP