Ввод - Вывод

Системное и прикладное программное обеспечение

Принципы работы аппаратуры ввода-вывода

Аппаратуру ввода-вывода можно рассматривать с разных точек зрения:

- Инженерной
  - микросхемы,
  - провода,
  - источники питания,
  - двигатели и прочие физические компоненты;
- Программной
  - интерфейс, предоставляемый программному обеспечению,
  - команды, принимаемые аппаратурой,
  - функции, выполняемые аппаратурой,
  - ошибки, о которых аппаратура может сообщить.

Устройства ввода-вывода

Устройства ввода-вывода можно грубо разделить на две категории:

- блочные устройства,
- символьные устройства.

Блочными называются устройства, хранящие информацию в виде блоков фиксированного размера, причем у каждого блока имеется адрес.

Важное свойство блочного устройства состоит в том, что каждый его блок может быть прочитан независимо от остальных блоков.

Устройства ввода-вывода

Устройства ввода-вывода можно грубо разделить на две категории:

- блочные устройства,
- символьные устройства.

Символьное устройство принимает или предоставляет поток символов без какой-либо блочной структуры.

Оно не является адресуемым и не выполняет операцию поиска. Принтеры, сетевые интерфейсные карты, мыши.

Скорости передачи данных типичных устройств

Устройство	Скорость данных
Клавиатура	10 байт/с
Мышь	100 байт/с
Модем 56 К	7 Кбайт/с
Телефонная линия	8 Кбайт/с
Двойная линия ISDN	16 Кбайт/с
Лазерный принтер	100 Кбайт/с
Сканер	400 Кбайт/с
Классическая сеть Ethernet	1,25 Мбайт/с
Шина USB (Universal Serial Bus)	1,5 Мбайт/с
Цифровая видеокамера	4 Мбайт/с
IDE-диск	5 Мбайт/с
40x CD-ROM	6 Мбайт/с
Быстрая сеть Ethernet	12,5 Мбайт/с
Шина ISA	16,7 Мбайт/с
IDE-диск (ATA-2)	16,7 Мбайт/с
FireWire (IEEE 1394)	50 Мбайт/с
XGA-монитор	60 Мбайт/с
Сеть SONET OC-12	78 Мбайт/с
Диск SCSI Ultra 2	80 Мбайт/с
Гигабитная сеть Ethernet	125 Мбайт/с
Лента Ultrium	320 Мбайт/с
Шина PCI	528 Мбайт/с
Объединительная плата Sun Gigaplane XB	20 Гбайт/с

Контроллеры устройств

Устройства ввода-вывода обычно состоят из **механической части** и **электронной части**.

Часто эти части можно разделить для придания модели более модульного и общего вида.

Электронный компонент устройства называется **контроллером устройства** или **адаптером**. В персональных компьютерах он часто принимает форму печатной платы, вставляемой в слот расширения.

Механический компонент находится в самом устройстве. Плата контроллера обычно снабжается разъемом, к которому может быть подключен кабель, ведущий к самому устройству. Многие контроллеры способны управлять несколькими идентичными устройствами.

Контроллеры устройств (2)

Если интерфейс между контроллером и устройством является стандартным (ANSI, IEEE или ISO), тогда различные компании могут выпускать отдельно контроллеры и устройства, удовлетворяющие данному интерфейсу.

Так, многие компании производят жесткие диски, соответствующие интерфейсу IDE или SCSI.

Интерфейс между устройством и контроллером часто является интерфейсом очень низкого уровня.

Контроллеры устройств (3)

Работа контроллера заключается в конвертировании последовательного потока битов в блок байтов и выполнение коррекции ошибок, если это необходимо.

Обычно байтовый блок собирается бит за битом в буфере контроллера. Затем проверяется контрольная сумма блока, и если она совпадает с указанной в заголовке сектора, блок объявляется считанным без ошибок, после чего он копируется в оперативную память.

Отображение на адресное пространство памяти ввода-вывода

У каждого контроллера есть **несколько регистров**, с помощью которых с ним может общаться центральный процессор.

При помощи записи в эти регистры операционная система велит устройству предоставить или принять данные. Читая из этих регистров, операционная система может узнать состояние устройства, например, готово ли оно к приему новой команды.

Помимо управляющих регистров, у многих устройств есть **буфер данных**, из которого операционная система может читать данные, а также писать данные в него.

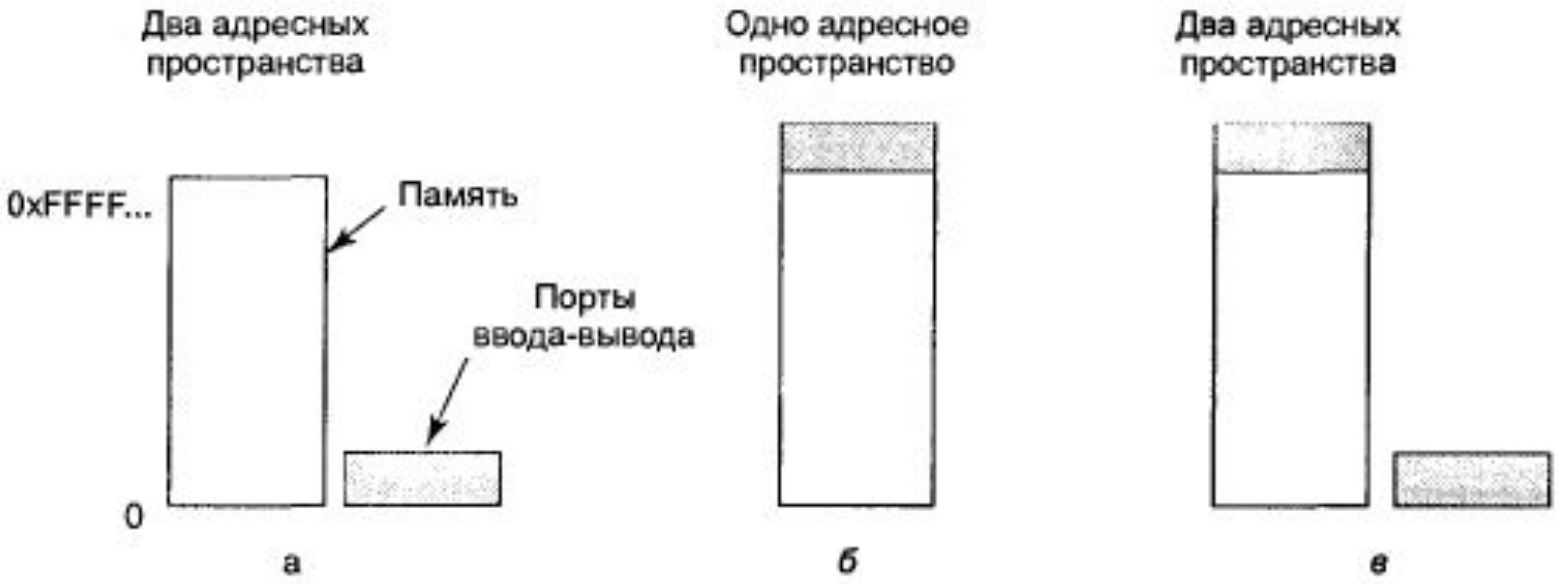
Отображение на адресное пространство памяти ввода-вывода (2)

Существует два альтернативных способа реализации доступа к управляющим регистрам и буферам данных устройств ввода-вывода.

Первый вариант заключается в том, что каждому управляющему регистру назначается **номер порта ввода-вывода**, 8- или 16-разрядное целое число. При помощи такой специальной команды процессора, центральный процессор может прочитать или записать содержимое управляющего регистра устройства из порта PORT в регистр процессора REG.

Второй вариант заключается в отображении всех управляющих регистров периферийных устройств на адресное пространство памяти. Каждому управляющему регистру назначался уникальный адрес в памяти. Такая система называется **отображаемым на адресное пространство памяти вводом-выводом**.

Отображение на адресное пространство памяти ввода-вывода (3)



Реализации доступа к управляющим регистрам

Обычно для регистров устройств отводятся адреса на вершине адресного пространства. Также существуют различные гибридные схемы, с отображаемыми на адресное пространство памяти буферами данных и отдельными портами ввода-вывода. Эта схема довольно широко применяется, например, в совместимых с IBM PC компьютерах на базе процессоров x86 и Pentium, в которых, помимо портов ввода-вывода с номерами от 0 до 64 К, адресное пространство оперативной памяти от 640 К до 1 М зарезервировано под буферы данных устройств ввода-вывода.

Как работают эти схемы?

Реализации доступа к управляющим регистрам (2)

1. Центральный процессор хочет прочитать слово данных либо из памяти, либо из порта ввода-вывода;
2. Он выставляет нужный адрес на адресную шину, после чего выставляет сигнал READ на управляющую шину;
    1. Вторая сигнальная линия позволяет отличить обращение к памяти от обращения к порту.
    2. В зависимости от состояния этой линии шины управления на запрос процессора реагирует устройство (контроллер) ввода-вывода или память.
3. Если пространство адресов общее, то каждый модуль памяти и каждое устройство ввода-вывода сравнивает выставленный на шину адрес с обслуживаемым им диапазоном адресов.
4. Если выставленный на шину адрес попадает в этот диапазон, то соответствующе устройство реагирует на запрос процессора. Поскольку выделенные внешним устройствам адреса удаляются из памяти, память не реагирует на них и конфликта адресов не происходит.

Достоинства отображения на адресное пространство памяти ввода-вывода

1. При такой схеме для обращения к устройствам ввода-вывода не требуются специальные команды процессора, такие как IN и OUT.
2. При отображении регистров ввода-вывода на память не требуется специального механизма защиты от пользовательских процессов, пытающихся обращаться к внешним устройствам.
3. При отображении регистров ввода-вывода на память каждая команда процессора, обращающаяся к памяти, может с тем же успехом обращаться к управляющим регистрам устройства.

Недостатки отображения на адресное пространство памяти ввода-вывода

1. В большинстве современных компьютеров применяется кэширование памяти. Кэширование управляющих регистров привело бы просто к катастрофе. Отображение регистров ввода-вывода на память увеличивает сложность аппаратуры и операционной системы, которой приходится управлять избирательным кэшированием.
2. При едином адресном пространстве все модули памяти и все устройства ввода-вывода должны изучать все обращения процессора к памяти, чтобы определить, на которые им следует реагировать. Однако в конструкции современных персональных компьютеров наблюдается тенденция в сторону использования выделенной высокоскоростной шины.

Недостатки отображения на адресное пространство памяти ввода-вывода

Сложность применения выделенной шины памяти на машинах с отображением регистров ввода-вывода на память состоит в том, что у устройств ввода-вывода нет способа увидеть адреса памяти, выставляемые процессором на эту шину, следовательно, они не могут реагировать на такие адреса.

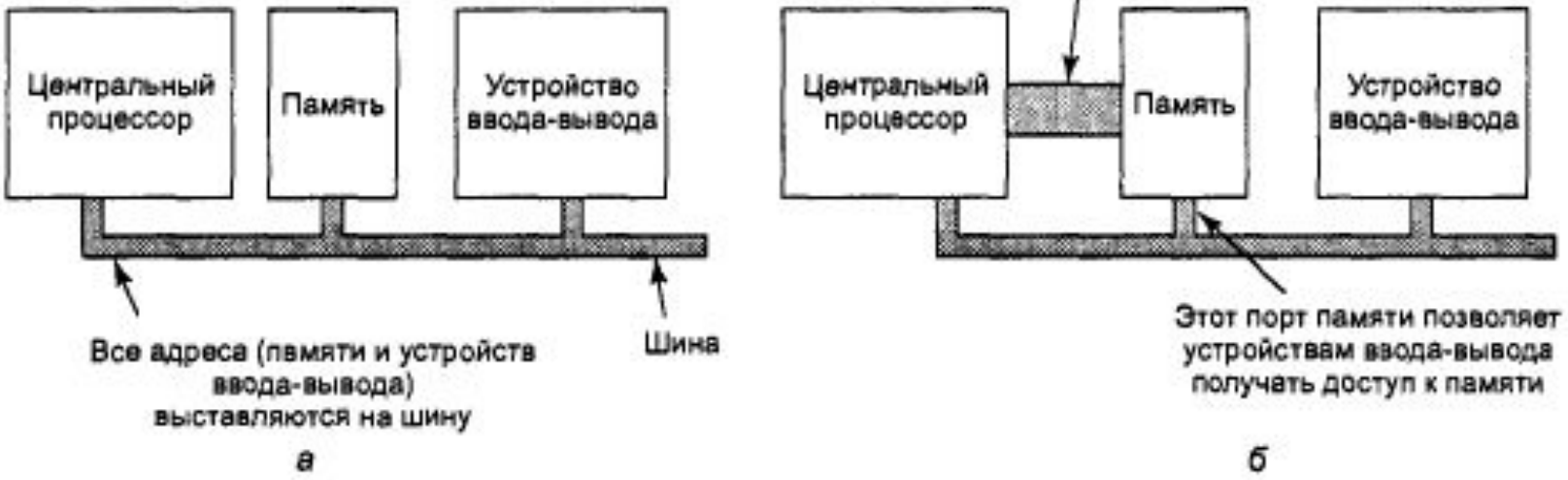
Чтобы отображение регистров ввода-вывода могло работать на системах с несколькими шинами, необходимы специальные меры.

Конструкторские решения

1. Сначала все обращения к памяти посылаются процессором по выделенной быстрой шине напрямую памяти. Если память не может ответить на эти запросы, процессор пытается сделать это еще раз по другим шинам.
2. Установка на шину памяти специального следящего устройства, передающего все адреса потенциально заинтересованным устройствам ввода-вывода.
3. Третье решение, используемое в компьютерах на базе процессора Pentium, состоит в фильтрации адресов микросхемой моста PCI. Эта микросхема содержит регистры диапазона, заполняемые во время загрузки компьютера.

Например, диапазон адресов от 640 К до 1 М может быть помечен как не относящийся к памяти. Все адреса, попадающие в подобный диапазон, передаются не памяти, а на шину PCI.

Конструкторские решения



Прямой доступ к памяти (DMA)

Независимо от того, отображаются ли регистры или буферы ввода-вывода на память или нет, центральному процессору необходимо как-то адресоваться к контроллерам устройств для обмена данными с ними.

Центральный процессор может запрашивать данные от контроллера ввода-вывода по одному байту, но подобная организация обмена данными крайне неэффективна, так как расходует огромное количество процессорного времени. Поэтому на практике часто применяется схема, называемая **прямым доступом к памяти** (DMA, direct memory access).

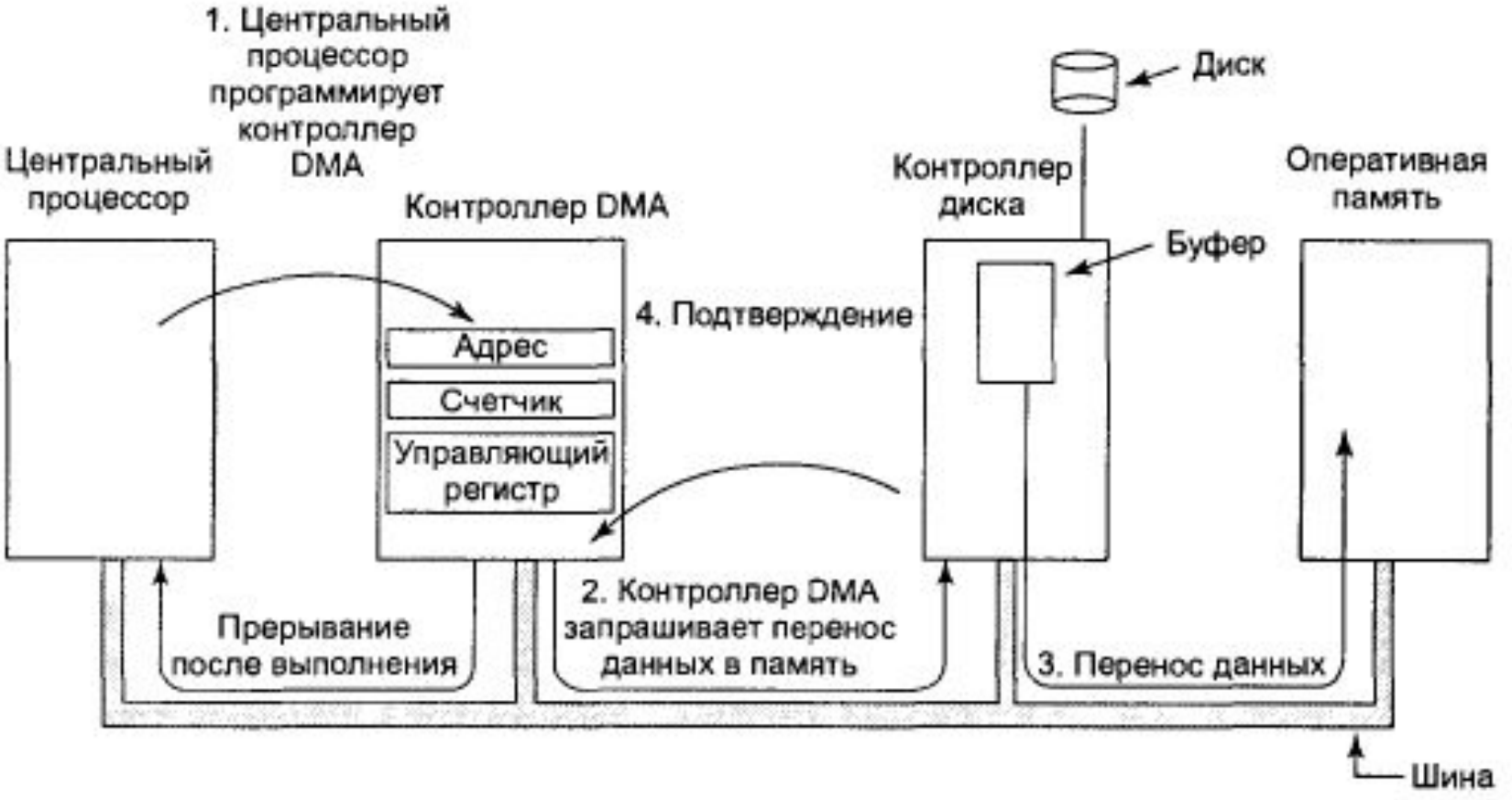
Прямой доступ к памяти (DMA) (2)

Операционная система может воспользоваться прямым доступом к памяти только при наличии аппаратного DMA-контроллера, который есть у большинства систем.

Иногда DMA-контроллер интегрируется в другие контроллеры, например в дисковый контроллер, но такой дизайн требует оснащения DMA-контроллерами каждого периферийного устройства. Как правило, DMA-контроллер, устанавливаемый на материнской плате, обслуживает запросы по передаче данных.

DMA-контроллер может получать доступ к системной шине независимо от центрального процессора. Он содержит несколько регистров, доступных центральному процессору для чтения и записи. К ним относятся регистр адреса памяти, счетчик байтов и один или более управляющих регистров.

Прямой доступ к памяти (DMA) (3)



Прямой доступ к памяти (DMA) (4)

1. Контроллер считывает с диска блок последовательно, бит за битом, пока весь блок не окажется во внутреннем буфере контроллера.
2. Затем контроллер проверяет контрольную сумму, чтобы убедиться, что при чтении не произошло ошибки.
3. После этого контроллер инициирует прерывание.
4. Когда операционная система начинает работу, она может прочитать блок диска побайтно или пословно, в цикле сохраняя считанное слово или байт в оперативной памяти.

Прямой доступ к памяти (DMA). Ход работы DMA контроллера

1. Сначала центральный процессор программирует DMA-котроллер, устанавливая его регистры и указывая таким образом, какие данные и куда следует переместить.
2. Затем процессор дает команду дисковому контроллеру прочитать данные во внутренний буфер и проверить контрольную сумму.
3. Данные получены и проверены контроллером диска, DMA может начинать работу.
4. DMA-контроллер начинает перенос данных, посылая дисковому контроллеру по шине запрос чтения (шаг 2).
5. Запись в память является еще одним стандартным циклом шины (шаг 3).
6. Когда запись закончена, контроллер диска также по шине посылает сигнал подтверждения контроллеру DMA (шаг 4).
7. Затем контроллер DMA увеличивает используемый адрес памяти и уменьшает значение счетчика байтов.
8. После этого шаги со 2-го по 4-й повторяются, пока значение счетчика не станет равно нулю. По завершении цикла копирования контроллер DMA инициирует прерывание процессора, сообщая ему таким образом, что перенос данных завершен.

Прямой доступ к памяти (DMA) (5)

Контроллеры DMA значительно различаются по степени своей сложности.

Самые простые из них за один раз выполняют одну операцию переноса данных, как описывалось выше. Более сложные контроллеры могут выполнять сразу несколько подобных операций. У таких контроллеров несколько каналов, каждый из которых управляется своим набором внутренних регистров. Центральный процессор начинает с того, что загружает в эти регистры соответствующие параметры. Все операции переноса данных должны выполняться с различными устройствами ввода-вывода. После переноса каждого слова данных контроллер DMA решает, какое устройство будет им обслужено следующим. Этот выбор может производиться циклически или при помощи приоритетной схемы, предоставляющей одним устройствам преимущество по сравнению с другими.

Одновременно несколько запросов могут дожидаться исполнения, при условии, что существует способ однозначно отличить подтверждения различных устройств. Часто с этой целью для каждого канала DMA используются различные линии подтверждения.

Прямой доступ к памяти (DMA) (5)

Многие шины могут работать в двух режимах: в пословном и поблочном. Некоторые контроллеры DMA также могут функционировать в обоих режимах.

В пословном режиме процедура выглядит так, как описывалось выше: контроллер DMA выставляет запрос на перенос одного слова и получает его. Если центральному процессору также нужна эта шина, ему приходится подождать. Этот механизм называется **захватом цикла** (cycle stealing), потому что контроллер устройства периодически «подкрадывается» и забирает случайный цикл шины у центрального процессора, слегка его тормозя.

В блочном режиме контроллер DMA велит устройству занять шину, сделать серию пересылок и отпустить шину. Такой способ действий называется **пакетным режимом**. Он более эффективен, чем захват цикла, поскольку занятие шины требует времени, а в пакетном режиме эта процедура выполняется всего один раз для передачи целого блока данных. Недостатком этого метода является то, что при переносе большого блока данных он может заблокировать центральный процессор и другие устройства на существенный промежуток времени.

Прямой доступ к памяти (DMA) (5)

В обсуждавшейся нами модели, иногда называемой сквозным режимом, контроллер DMA велит контроллеру устройства переслать данные напрямую в оперативную память. В некоторых DMA-контроллерах используется также режим, при котором контроллер устройства посылает слово данных контроллеру DMA, который затем выставляет на шину еще один запрос для передачи этого слова туда, куда его нужно передать. При такой схеме требуется лишний цикл шины на передачу каждого слова, зато такая схема обладает большей гибкостью, так как также позволяет выполнять копирование с устройства на устройство, минуя память, и даже из памяти в память.

Большинство контроллеров DMA используют для передачи данных физические адреса памяти. Чтобы использовать физические адреса памяти, операционная система должна преобразовать виртуальный адрес буфера памяти в физический и записать этот физический адрес в адресный регистр контроллера DMA. В некоторых контроллерах DMA применяется альтернативная схема, при которой в контроллер DMA записывается сразу виртуальный адрес. В этом случае контроллер DMA должен использовать менеджер памяти MMU для преобразования адреса.

Виртуальный адрес может быть выставлен на адресную шину только в том случае, когда MMU является частью памяти, а не частью центрального процессора.

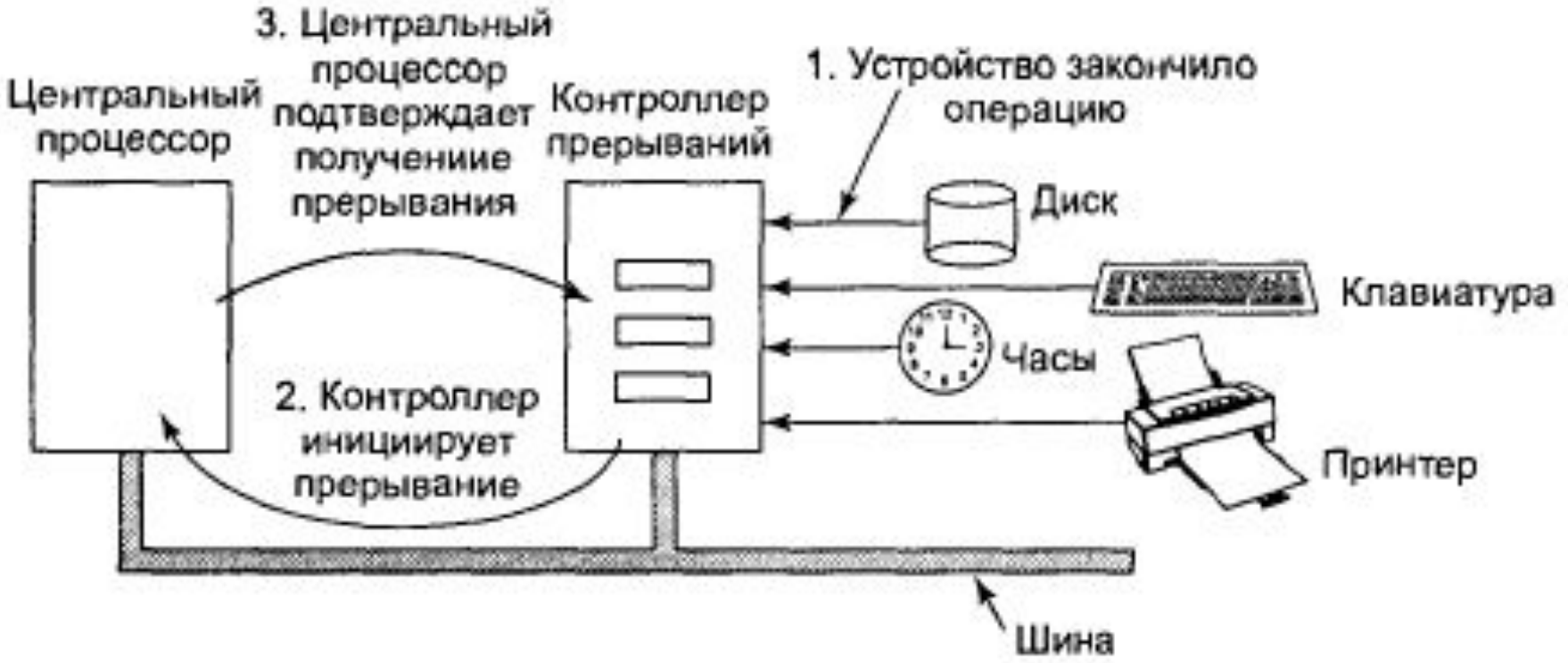
Прямой доступ к памяти (DMA) (5)

Как мы уже упоминали, до начала операции DMA диск сначала считывает данные в свой внутренний буфер. Возможно, вы задаетесь вопросом, почему контроллер не помещает данные прямо в оперативную память, по мере получения их с диска. Другими словами, зачем ему нужен внутренний буфер? Тому есть две причины. Во-первых, при помощи внутренней буферизации контроллер диска может проверить контрольную сумму до начала переноса данных в память. Если контрольные суммы не совпадают, формируется сигнал об ошибке и перенос данных не производится.

Во-вторых, дело в том, что как только началась операция чтения с диска, биты начинают поступать с постоянной скоростью, независимо от того, готов контроллер диска их принимать или нет. Если контроллер диска попытается писать эти данные напрямую в память, ему придется делать это по системной шине. Если при передаче очередного слова шина окажется занятой каким-либо другим устройством, контроллеру диска придется ждать. Если следующее слово с диска прибудет раньше, чем контроллер успеет сохранить предыдущее, контроллер либо потеряет предыдущее слово, либо ему придется сохранять его где-либо еще. Таким образом, необходимость внутреннего буферирования становится очевидной. При наличии внутреннего буфера контроллеру диска шина не нужна до тех пор, пока не начнется операция DMA. В результате устройство контроллера диска оказывается проще, так как при операции DMA пересылки данных параметр времени не является критичным.

DMA используется не во всех компьютерах. Главный аргумент против использования DMA состоит в том, что центральный процессор обычно значительно превосходит DMA-контроллер по скорости и может выполнить ту же работу значительно быстрее. При отсутствии другой работы у центрального процессора заставлять быстрый центральный процессор ждать, пока медленный контроллер DMA выполнит свою работу, бессмысленно. Кроме того, компьютер без контроллера DMA, с центральным процессором, выполняющим всю работу программно, оказывается дешевле, что крайне важно в производстве компьютеров нижней ценовой категории.

Прерывания


3. Центральный процессор подтверждает получение прерывания
Центральный процессор
Контроллер прерываний
1. Устройство закончило операцию
Диск
Клавиатура
Часы
Принтер
2. Контроллер инициирует прерывание
Шина

Прерывания. Аппаратный уровень.

Когда устройство ввода-вывода заканчивает свою работу, оно инициирует прерывание. Для этого устройство выставляет сигнал на выделенную устройству специальную линию шины. Этот сигнал распознается микросхемой контроллера прерываний, расположенной на материнской плате. Контроллер прерываний принимает решение о дальнейших действиях.

При отсутствии других необработанных запросов прерывания контроллер прерываний обрабатывает прерывание немедленно. Если прерывание уже обрабатывается, и в это время приходит запрос от другого устройства по линии с более низким приоритетом, то новый запрос просто игнорируется.

Для обработки прерывания контроллер выставляет на адресную шину номер устройства, требующего к себе внимания, и устанавливает сигнал прерывания на соответствующий контакт процессора. Этот сигнал заставляет процессор приостановить текущую работу и начать выполнять обработку прерывания.

Принципы программного обеспечения ввода-вывода

Задачи программного обеспечения ввода-вывода

Ключевая концепция разработки программного обеспечения ввода-вывода известна как **независимость от устройств**. Эта концепция означает возможность написания программ, способных получать доступ к любому устройству ввода-вывода, без предварительного указания конкретного устройства. Например, программа, читающая данные из входного файла, должна с одинаковым успехом работать с файлом на дискете, жестком диске или компакт-диске. При этом не должны требоваться какие-либо изменения в программе.

Например, должна быть возможность дать команду вроде

```
sort < input > output
```

и эта команда должна работать, независимо от того, что указано в качестве входного устройства - гибкий диск, IDE-диск, SCSI-диск или клавиатура.

Задачи программного обеспечения ввода-вывода

Все проблемы, связанные с отличиями этих устройств, должна решать операционная система.

Тесно связан с концепцией независимости от устройств **принцип единообразного именования**. Имя файла или устройства должно быть просто текстовой строкой или целым числом и никоим образом не зависеть от физического устройства.

В системе UNIX все диски могут быть произвольным образом интегрированы в иерархию файловой системы, так что пользователю не обязательно знать, какое имя какому устройству соответствует.

Задачи программного обеспечения ввода-вывода

Важным аспектом программного обеспечения ввода-вывода является:

- **обработка ошибок**.
- **способ переноса данных**.
- **буферизация**.
- **понятие выделенных устройств** и **устройств коллективного использования**.

Задачи программного обеспечения ввода-вывода

- **обработка ошибок**. Ошибки должны обрабатываться как можно ближе к аппаратуре. Если контроллер обнаружил ошибку чтения, он должен попытаться по возможности исправить эту ошибку сам. Если он не может это сделать, тогда эту ошибку должен обработать драйвер устройства, возможно, попытавшись прочитать этот блок еще раз.
- **способ переноса данных**: **синхронный** (**блокирующий**) против **асинхронного** (**управляемого прерываниями**). Большинство операций ввода-вывода на физическом уровне являются асинхронными - центральный процессор запускает перенос данных и отправляется заниматься чем-либо другим, пока не придет прерывание. Программы пользователя значительно легче написать, используя блокирующие операции ввода-вывода.

Задачи программного обеспечения ввода-вывода

- **буферизация**. Часто данные, поступающие с устройства, не могут быть сохранены сразу там, куда они в конечном итоге направляются. Например, когда пакет приходит по сети, операционная система не знает, куда его поместить, пока не будет изучено его содержимое, для чего этот пакет нужно где-то временно сохранить.
- **понятие выделенных устройств** и **устройств коллективного использования**. С некоторыми устройствами ввода-вывода, такими как диски, может одновременно работать большое количество пользователей.

Программный ввод-вывод

Существует три фундаментально различных способа осуществления операций ввода-вывода.

1. программный ввод-вывод
2. вводом-выводом с управляемым прерываниями
3. вводом-выводом с использованием DMA

Простейший вид ввода-вывода состоит в том, что всю работу выполняет центральный процессор. Этот метод называется **программным вводом-выводом**.

Программный ввод-вывод

Программный ввод-вывод

Существенный аспект программного ввода-вывода состоит в том, что после печати каждого символа процессор в цикле опрашивает готовность устройства.

Такое поведение процессора называется **опросом** или **ожиданием готовности**, а также **активным ожиданием**.

```
copy_from_user(buffer. p. count); /* p - буфер ядра */
for (i=0: i<count: i++){ /* цикл символов */
while (*printer_status_reg != READY): /*цикл ожидания*/
*printer_data_reg = p[i]: /* печать символа */
}
return_to_user();
```

Управляемый прерываниями ввод-вывод. Пример. Печать принтера.

Если принтер может печатать, скажем, 100 символов в секунду, то на печать каждого символа уходит 10 мс.

Это значит, что после записи каждого символа в регистр данных принтера центральный процессор должен ждать в цикле целых 10 мс, пока ему не позволят записать в регистр следующий символ.

Этого времени более чем достаточно для переключения контекста и запуска другого процесса на 10 мс, которые в противном случае просто будут потеряны.

Управляемый прерываниями ввод-вывод

Предоставить центральному процессору возможность делать что-нибудь в то время, когда принтер переходит в состояние готовности, можно при помощи прерываний.

Когда выполняется системный вызов печати строки, как мы уже показывали, буфер копируется в пространство ядра и первый символ строки копируется на принтер, как только принтер выставит бит готовности.

После этого центральный процессор вызывает планировщик, который запускает какой-либо другой процесс. Процесс, попросивший распечатать строку, оказывается заблокирован на весь период печати строки.

Управляемый прерываниями ввод-вывод

Когда принтер напечатал символ и готов принять следующий, он инициирует прерывание.

Это прерывание вызывает остановку текущего процесса и сохранение его состояния. Затем запускается процедура обработки прерывания от принтера. Если напечатаны все символы, обработчик прерывания предпринимает необходимые меры для разблокировки процесса пользователя.

В противном случае он печатает следующий символ, подтверждает прерывание и возвращается к процессу, выполнение которого было приостановлено прерыванием от принтера.

Управляемый прерываниями ввод-вывод

Печать строки при помощи ввода-вывода, управляемого прерываниями:

- программа, выполняемая при обращении к системному вызову (а);
- процедура обработки прерываний (б).

```
copy_from_user(buffer,p,count);
enable_interrupts();
while(*printer_status_reg!=READY);
*printer_data_register=p[0];
scheduler();
```

а

```
if (count==0) {
    unblock_user();
} else {
        *printer_data_register=p[i];
        count=count-1;
        i=i+1;
}
acknowledge_interrupt();
return_from_interrupt();
```

б

Ввод-вывод с использованием DMA

Идея в использовании DMA состоит в том, чтобы позволить контроллеру DMA поставлять принтеру символы по одному, не беспокоя при этом центральный процессор.

По существу, этот метод почти не отличается от программного ввода-вывода, с той лишь разницей, что всю работу вместо центрального процессора выполняет контроллер DMA.

Наибольший выигрыш от использования DMA состоит в уменьшении количества прерываний с одного на печатаемый символ до одного на печатаемый буфер.

Если символов много, а прерывания обрабатываются медленно, то этот выигрыш весьма существенен.

С другой стороны, контроллер DMA обычно значительно уступает центральному процессору в скорости.

Ввод-вывод с использованием DMA

Печать строки при помощи DMA:

- программа, выполняемая при обращении к системному вызову (а);
- процедура обработки прерываний (б)

```
copy_from_user(buffer,p,count);
set_up_DMA_controller();
scheduler();
```

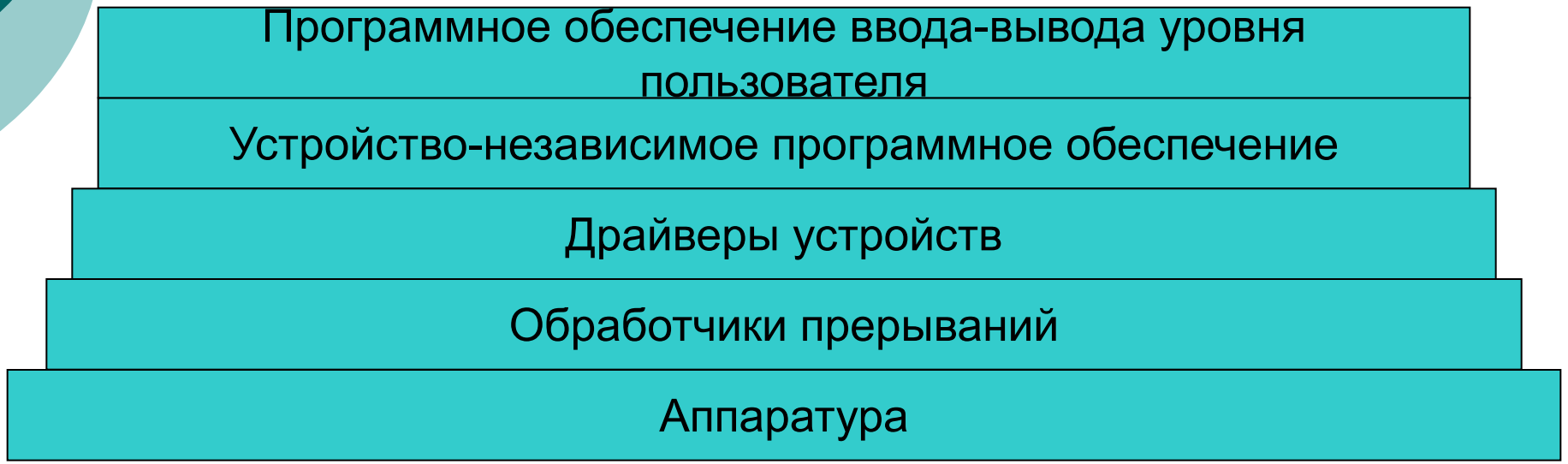
а

```
acknowledge_interrupt();
unblock_user();
return_from_interrupt();
```

б

Программные уровни ввода-вывода

Программные уровни ввода-вывода



Обработчики прерываний

Прерывания должны быть скрыты внутри операционной системы.

Пример реализации: блокировка драйвера.

Драйвер блокирует себя сам:

- выполнив на семафоре процедуру **down**,
- процедуру **wait** на переменной состояния,
- процедуру **receive** на сообщении.

Далее начинает работу обработчик прерываний.

По окончании он может разблокировать драйвер.

Порядок обработки прерываний

1. Сохранить все регистры (включая PSW), не сохраненные аппаратурой.
2. Установить контекст для процедуры обработки прерываний.
3. Установить указатель стека для процедуры обработки прерываний.
4. Выдать подтверждение контроллеру прерываний.
5. Скопировать содержимое регистров с того места, где они были сохранены, в таблицу процессов.
6. Запустить процедуру обработки прерываний.
7. Выбрать процесс, которому передать управление.
8. Установить контекст MMU для следующего работающего процесса.
9. Загрузить регистры нового процесса, включая его PSW.
10. Начать выполнение нового процесса.

Драйверы устройств

Каждый контроллер имеет набор регистров управления и регистров состояния.

Например, драйвер мыши.

Для управления каждым устройством ввода-вывода требуется специальная программа, называемая **драйвером устройства.**

Обычно пишется производителем устройства и распространяется вместе с устройством.

Драйверы устройств (2)

- Каждый драйвер устройства обычно поддерживает один тип устройств или класс близких устройств.
- Чтобы получить доступ к аппаратной части устройства (к регистрам контроллера), драйвер устройства должен быть частью ядра операционной системы.

- Драйвер, работающий в пространстве пользователя.

Возможно создать драйвер с системными вызовами для чтения и записи регистров устройств.

Недостатки: основная причина крушения операционной системы - драйверы, содержащие ошибки.

Драйверы устройств (3)

Драйверы устройств. Классификация.

классификация драйверов:

- блочные устройства (диски)
- символьные устройства (клавиатуры, принтеры)

В большинстве операционных систем определены:

- стандартный интерфейс, который должны поддерживать все блочные драйверы,
- стандартный интерфейс, поддерживаемый всеми символьными драйверами.

Эти интерфейсы включают наборы процедур, которые могут вызываться остальной операционной системой для обращения к драйверу.

Независимое от устройств программное обеспечение ввода-вывода

Некоторая часть программного обеспечения ввода-вывода предназначена для работы с конкретными устройствами, другая часть является независимой от устройств.

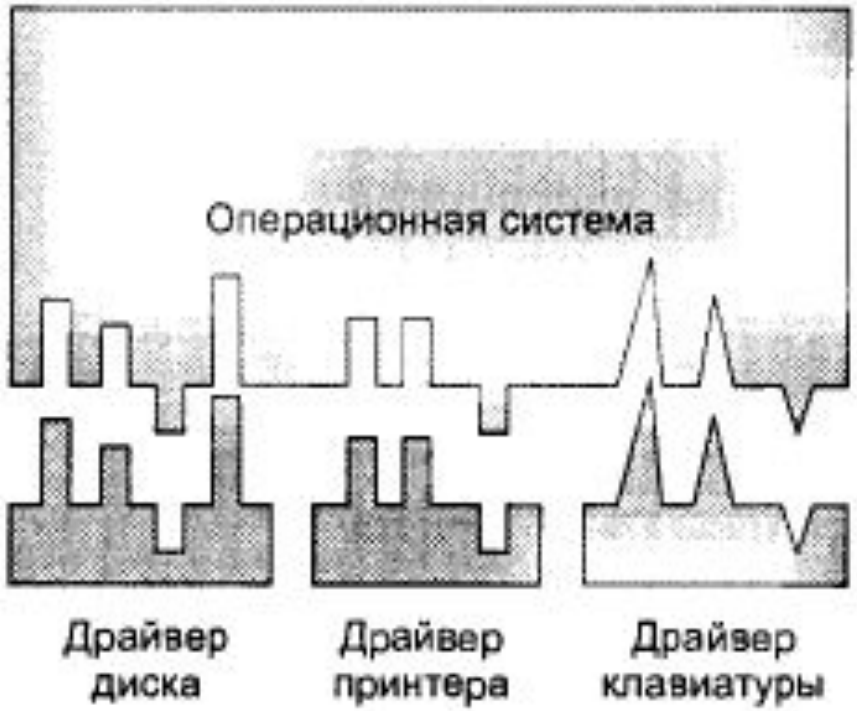
Расположение точной границы между драйверами и независимым от устройств программным обеспечением зависит от системы.

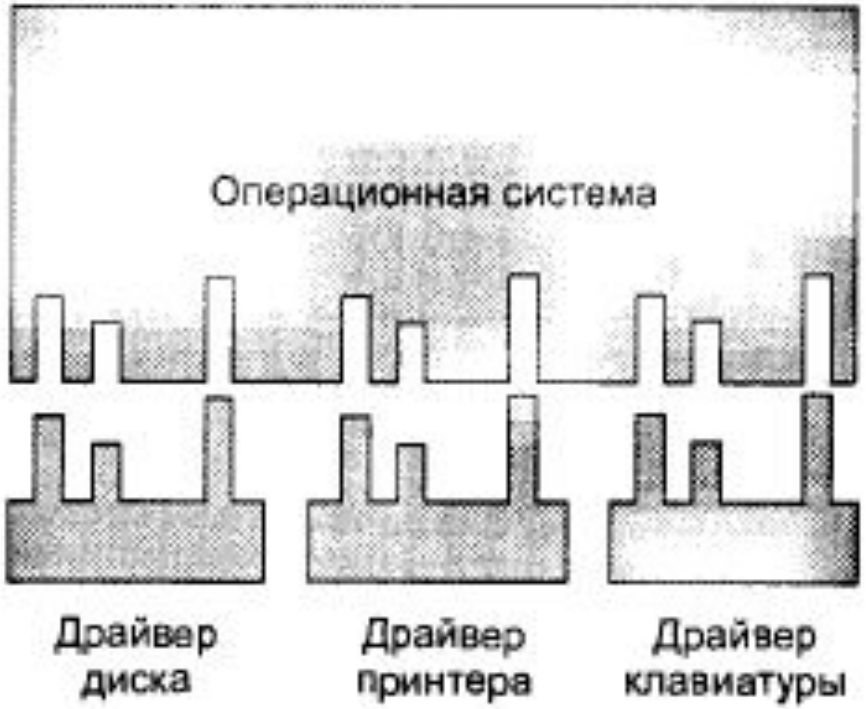
Основная задача независимого от устройств программного обеспечения состоит в выполнении функций ввода-вывода, общих для всех устройств, и **предоставлении единообразного интерфейса** для программ уровня пользователя.

Функции независимого от устройств программного обеспечения

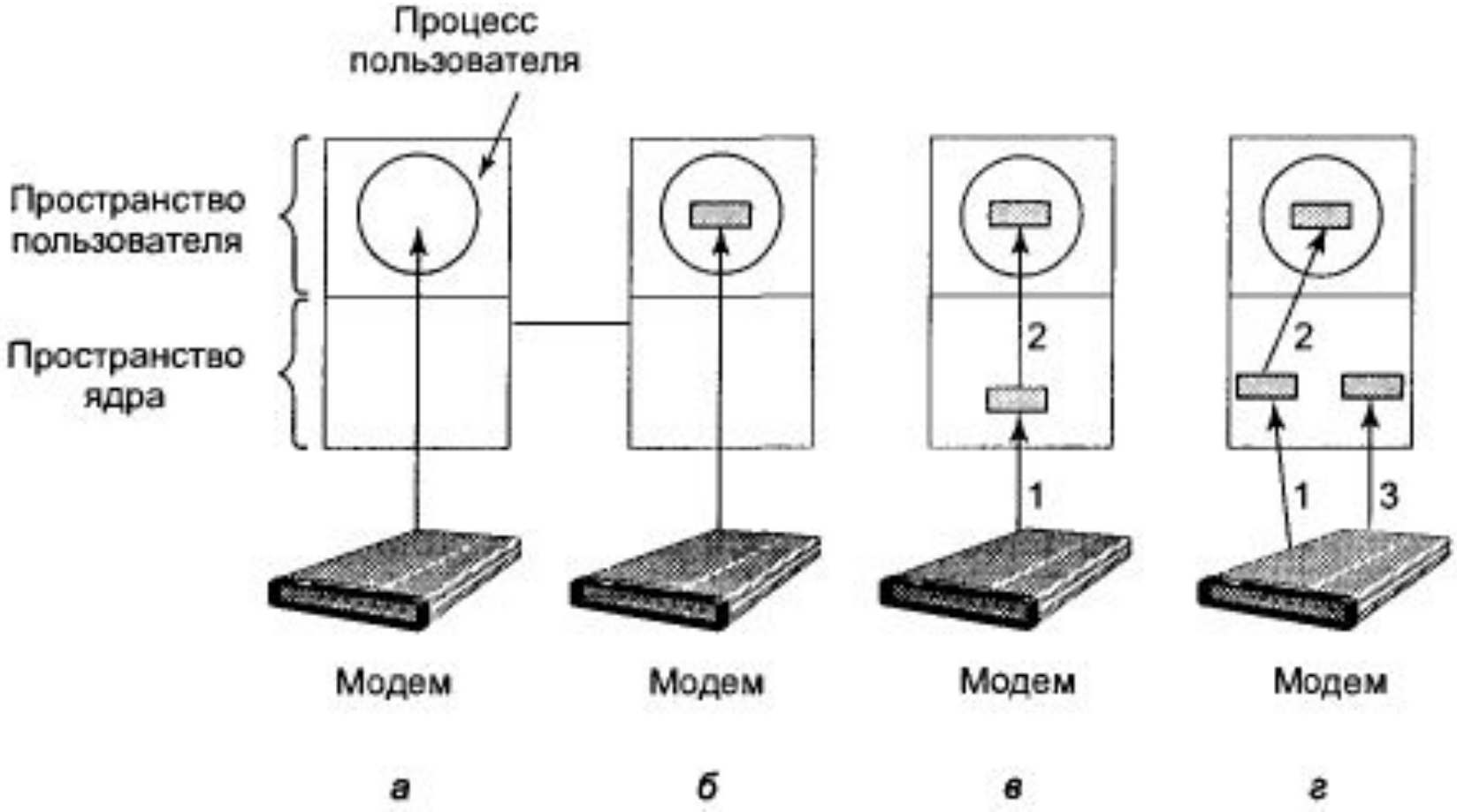
- Единообразный интерфейс для драйверов устройств
- Буферизация
- Сообщение об ошибках
- Захват и освобождение выделенных устройств
- Размер блока, не зависящий от устройства

Единообразный интерфейс для драйверов устройств

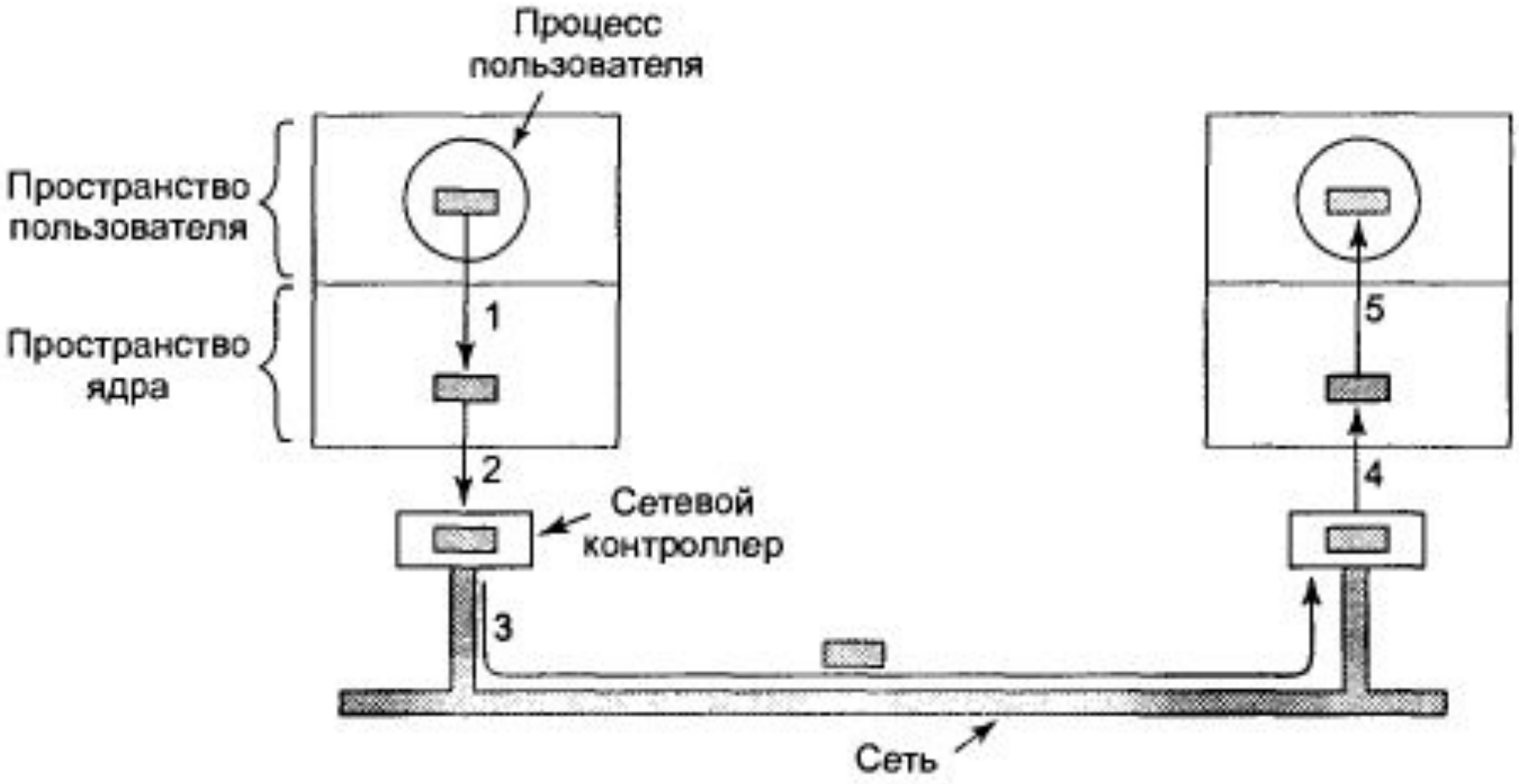




Буферизация



Буферизация (2)



Сообщения об ошибках. Ошибки программирования.

Ошибки значительно чаще случаются в контексте ввода-вывода, нежели в других контекстах.

Высокая скорость обработки ошибок операционной системой.

Структура обработки ошибок является независимой от устройств.

Один из классов ошибок ввода-вывода составляют **ошибки программирования**.

Например:

- записать данные на устройство ввода;
- предоставление неверного адреса буфера;
- обращение к несуществующему устройству.

Сообщения об ошибках. Ошибки ввода-вывода.

Например:

- попытка записать в поврежденный блок диска;
- попытка прочитать данные с выключенной видеокамеры.

В таких случаях драйвер сам решает, что ему делать. Если драйвер не может сам принять решение, он передает сведения о возникшей проблеме в вышестоящие инстанции.

Действия этого программного обеспечения зависят от окружения и природы ошибки. Если это простая ошибка чтения, а программа интерактивная, можно отобразить диалоговое окно с вопросом к пользователю о дальнейших действиях. Если программа не интерактивная, единственная возможность состоит в аварийном завершении системного вызова с соответствующим кодом ошибки.

Захват и освобождение выделенных устройств

Некоторые устройства, например устройство записи компакт-дисков.

Операционная система должна рассмотреть запросы на использование этого устройства и либо принять их, либо отказать в выполнении запроса, в зависимости от доступности запрашиваемого устройства.

Альтернативный подход состоит в предоставлении специального механизма для запроса и освобождения выделенных устройств. Попытка захватить недоступное устройство вызовет блокировку вызывающего процесса вместо возврата с ошибкой. Блокированные процессы устанавливаются в очередь.

Независимый от устройств размер блока

У различных дисков могут быть разные размеры сектора.

Независимое от устройств программное обеспечение должно скрывать этот факт от верхних уровней и предоставлять им единообразный размер блока, например, объединяя несколько физических сегментов в один логический блок.

При этом более высокие уровни имеют дело только с абстрактными устройствами, с одним и тем же размером логического блока, не зависящим от размера физического сектора.

Некоторые символьные устройства предоставляют свои данные побайтно (например, модемы), тогда как другие выдают их большими порциями (например, сетевые интерфейсы). Эти различия также могут быть скрыты.

Функции независимого от устройств программного обеспечения

- Единообразный интерфейс для драйверов устройств
- Буферизация
- Сообщение об ошибках
- Захват и освобождение выделенных устройств
- Размер блока, не зависящий от устройства

Программное обеспечение ввода-вывода пространства пользователя

Хотя большая часть программного обеспечения ввода-вывода находится в операционной системе, небольшие его порции состоят из библиотек, присоединенных к программам пользователя, или даже целых программ, работающих вне ядра. Системные вызовы, включая системные вызовы ввода-вывода, обычно состоят из библиотечных процедур. Если программа на С содержит вызов

```
count = write (fd, buffer, nbytes);
```

библиотечная процедура write будет скомпонована с программой и, таким образом, будет содержаться в двоичной программе, загружаемой в память во время выполнения программы.

Набор всех этих библиотечных процедур, несомненно, является частью системы ввода-вывода.

Спулинг

Не все программное обеспечение ввода-вывода пространства пользователя состоит из библиотечных процедур. Другую важную категорию составляет система спулинга.

Спулинг (spooling - подкачка, предварительное накопление данных) представляет собой способ работы с выделенными устройствами в многозадачной системе.

Рассмотрим типичное устройство, на котором используется спулинг: *принтер*.

В принципе можно разрешить каждому пользователю открывать специальный символьный файл принтера, однако представьте себе, что процесс открыл его, а затем не обращался к принтеру в течение нескольких часов. Ни один другой процесс в это время не сможет ничего напечатать.

Вместо этого создается специальный процесс, называемый **демоном**, и специальный каталог, называемый **каталогом спулинга** или **каталогом спулера**.

Уровни и основные функции системы ввода-вывода