Операционные системы

Управление вводом/выводом [1]

Соловьев А. В.

ПетрГУ – КИИСиФЭ

(Rev. 2018 06 07)

[1] По материалам «Таненбаум Э., Бос Х. Современные операционные системы. СПб.: Питер, 2015.»

Принципы организации ввода/вывода

Блочные и символьные устройства

Блочные устройства хранят информацию в блоках фиксированной длины (512 байт...64 Кбайт), у каждого блока есть собственный адрес. Передача данных ведётся пакетами из одного или нескольких целых (последоват.) блоков. Каждый блок можно читать независимо от других. Примеры: накопители – жёсткие диски, CD/DVD/Blueray, флешки.

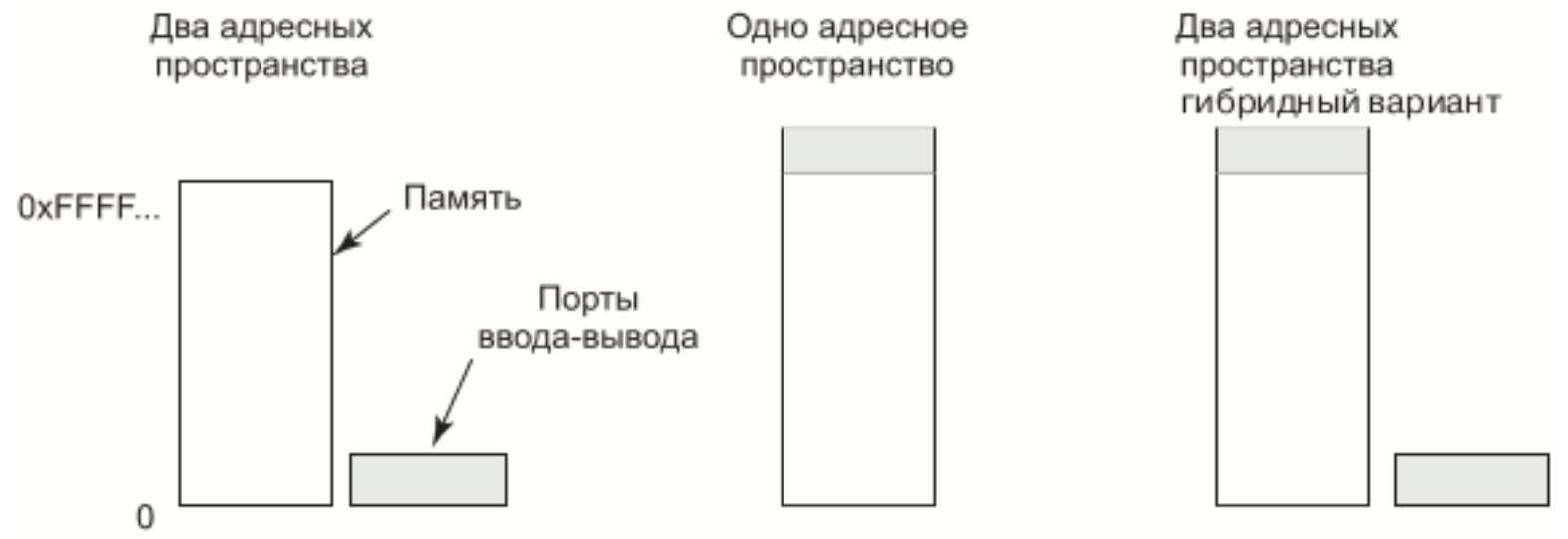
Символьные устройства выдают или воспринимают поток символов, не относящийся ни к какой блочной структуре. Не являются адресуемыми и не имеют операции позиционирования. Примеры: принтер, мышь, NIC.
К символьным зачастую относят все устройства, не попадающие в класс блочных: часы, экран, звуковая карта и проч.

При работе с у-вами I/O приходится иметь дело с широким диапазоном скоростей:

- клавиатура – 10 байт/с
- мышь – 100 байт/с
- модем – 10 Кбайт/с
- сканер – 1 Мбайт/с
- адаптер WiFi – 7 Мбайт/с
- gigabit ethernet – 125 Мбайт/с
- SATA-3 / USB 3.0 – 600 Мбайт/с
- сеть SONET OC-768 – 5 Гбайт/с

Пространство памяти для ввода/вывода (1)

Как ЦП обменивается данными с регистрами управления и буферами данных устройств?



- отдельное *пространство портов ввода-вывода*
- *отображаемый на адресное пространство памяти ввод-вывод*
- гибридный вариант

```
IN REG,PORT
OUT PORT,REG
```

```
MOV REG,MEM
MOV MEM,REG
```

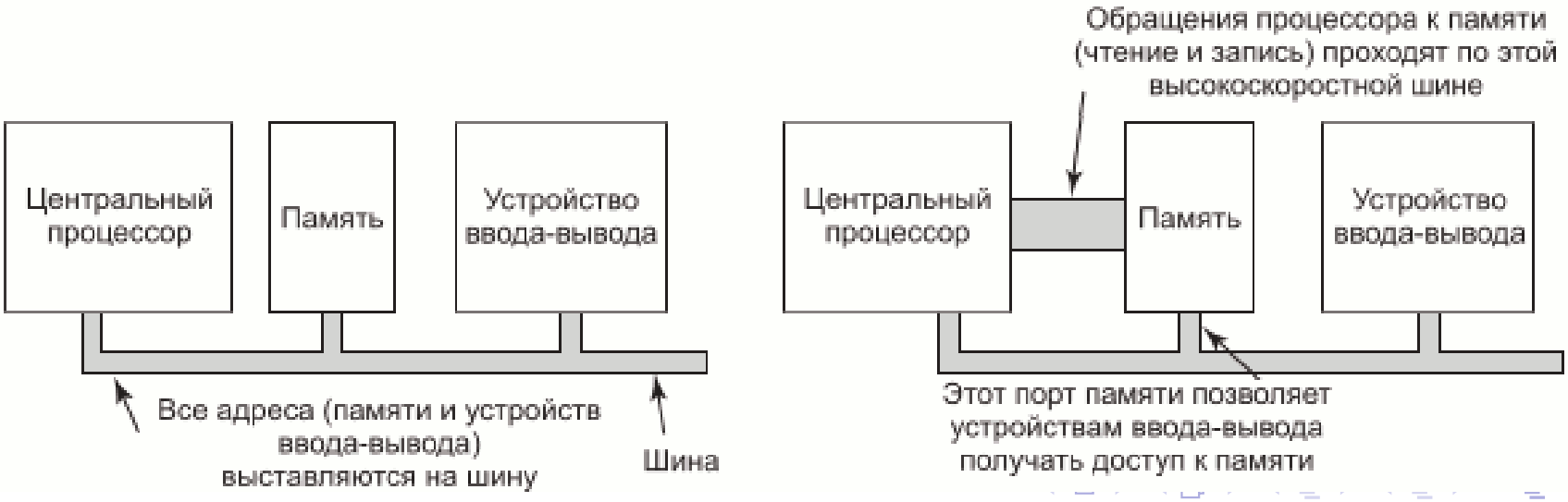
Пространство памяти для ввода/вывода (2)

Достоинства отображаемого I/O:

- не нужны ассемблерные вставки с инструкциями IN/OUT
- не нужен дополнительный механизм защиты для пространства I/O
- упрощение ассемблерного кода

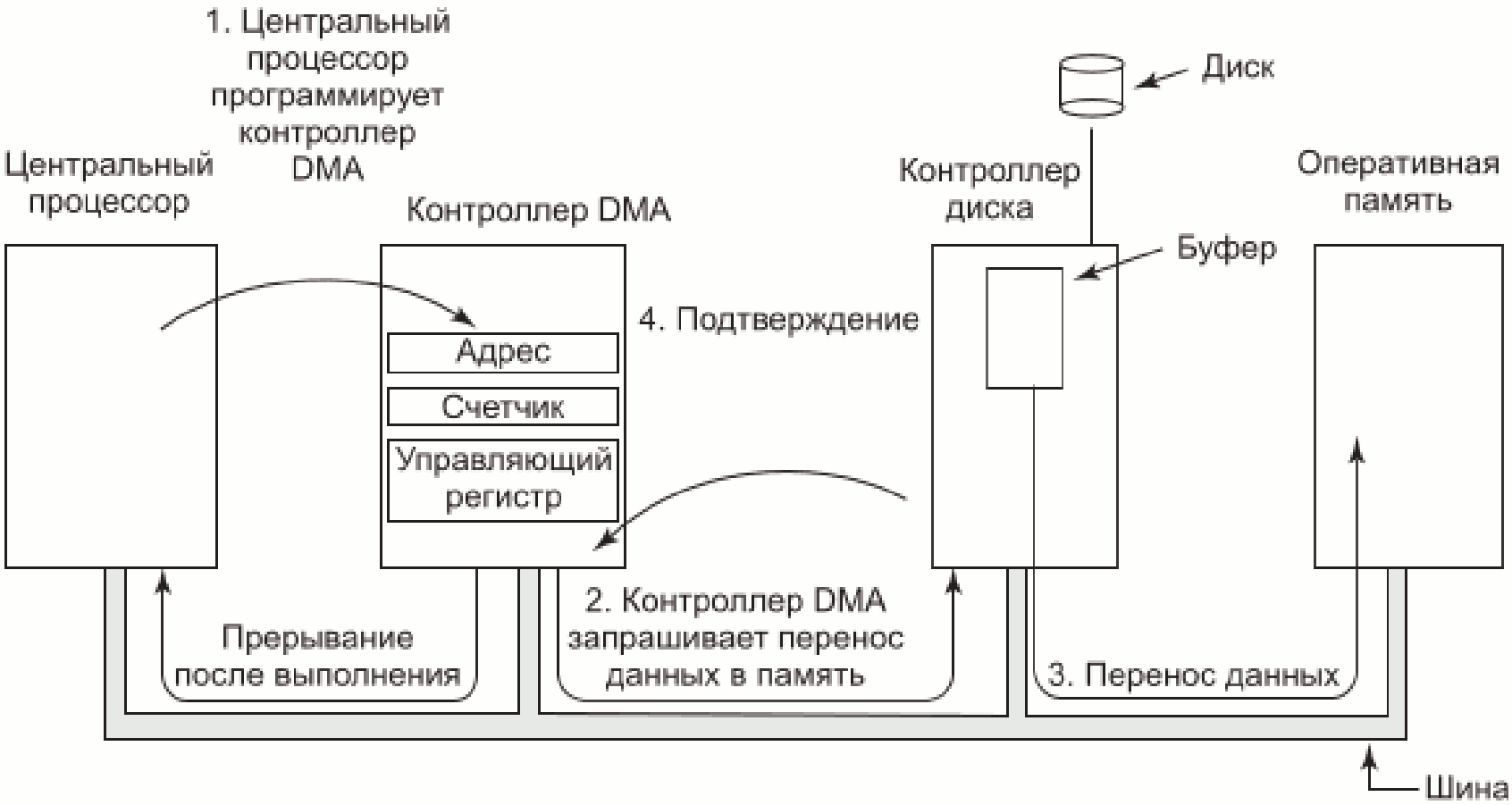
Недостатки отображаемого I/O:

- необходимо выборочное отключение кэширования
- проблема нескольких шин



Прямой доступ к памяти (1)

Задача: освободить ЦП от побайтного чтения внутреннего буфера у-ва I/O.



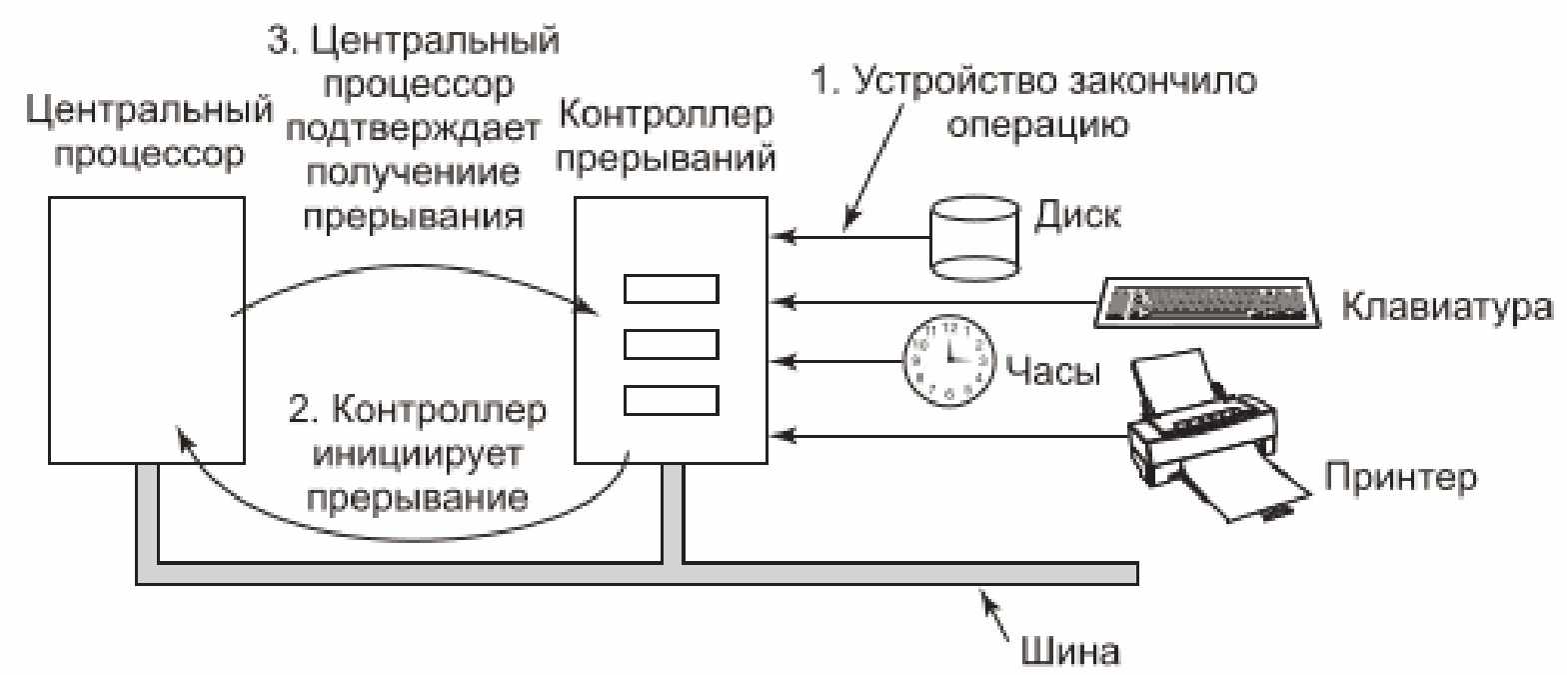
Прямой доступ к памяти (2)

Нюансы реализаций:

- DMA-контроллер выделен (единственный) или встроен в каждое устройство?
- Передача данных напрямую у-во $\Leftrightarrow$ память (один цикл шины, сквозной режим) или через буфер DMA-контроллера (два цикла шины)?
- DMA-контроллер многоканальный?
- Передача данных пословная или поблочная (пакетная)?
- Адреса на шине: физические или виртуальные (внешнее MMU)?
- Буфер в памяти непрерывный (bounce buffer) или фрагментирован виртуализацией (scatter-gather list)?

Прерывания

Когда у-во I/O завершает порученную ему работу (требует обслуживания), оно инициирует прерывание.



Перед запуском обработчика прерывания необходимо сохранить контекст: где? (спец. регистры или стек; стек пользовательский или стек ядра)

Точные или неточные прерывания?

Принципы создания ПО ввода/вывода

Задачи

- Избегать зависимости от конкретных устройств (чтение файла с флешки, CD или жёсткого диска должно происходить одинаково).
- Единообразие именования (устройств, файлов).
- Обрабатывать ошибки как можно ближе к аппаратуре.
- Интерфейсы синхронного и асинхронного ввода/вывода (на уровне ОС – как правило асинхронно, но на уровне приложений – предпочтительно синхронно, однако асинхронные интерфейсы тоже необходимы).
- Буферизация (синхронизация скоростей получения данных из буфера и скорости наполнения буфера).
- Реализовать совместное или монопольное использование устройств (диск – совместно, принтер – монопольно, ...)

Программный опрос (активное ожидание)

На примере вывода строки на принтер:

- syscall "открыть (принтер)" (блокировка? ошибка?)
- syscall "запись (в принтер)"
  1. скопировать выводимый текст в буфер ядра (или изменить карту памяти, чтобы ядро имело доступ к пользовательскому буферу)
  2. прочитать регистр статуса принтер: принтер готов?
  3. если не готов, п. 2
  4. если готов, берём очередной символ из буфера и записываем в регистр данных принтера
  5. если строка не исчерпана, п. 2
- syscall "закрыть (принтер)"

ЦП занят всё время, пока выполняется операция I/O (неэффективно для сложных систем).

Ввод/вывод по прерыванию

Контроллер принтера переводится в режим сигнализирования прерываниями о готовности. Первый символ копируется в регистр данных принтера, как только он пожелает его принять. В этот момент ЦП обращается к планировщику и запускается какой-нибудь другой процесс. Процесс, запросивший распечатку строки, блокируется до тех пор, пока не будет распечатана вся строка.

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 распечатаны все символы, обработчик прерывания предпринимает действие по разблокированию процесса пользователя. В противном случае он печатает следующий символ, подтверждает прерывание и возвращается к процессу, выполнение которого было приостановлено прерыванием от принтера.

Недостаток: прерывания выдаются на каждый символ.

Ввод/вывод с использованием DMA

Контроллер DMA используется для посимвольной передачи строки принтеру без участия ЦП. Вместо прерывания от принтера на каждый символ, генерируется одно прерывание от контроллера DMA на весь буфер.

Выгодно при большом количестве символов и медленной обработке прерываний.

Однако контроллер DMA обычно работает намного медленнее, чем ЦП (скорости контроллера DMA может не хватать для управления устройством на полной скорости).

Уровни ПО ввода/вывода

ПО I/O уровня пользователя	Обращение к вызовам I/O, форматирование, спулинг
Устройство-независимое ПО ОС (HAL)	Именование, защита, блокировки, буферизация, ошибки
Драйверы устройств	Установка регистров устройств, завершение операций
Обработчики прерываний	Активация соответствующего драйвера
Аппаратура	

Уровни ПО ввода/вывода: обработчики прерываний

1. Сохранить все регистры.
2. Установить контекст для процедуры обработки прерывания (TLB, MMU и таблицы страниц), стек для процедуры обработки прерывания.
3. Послать подтверждение контроллеру прерываний (разрешить прерывания).
4. Запустить процедуру обработки прерывания, которая извлечет информацию из регистров контроллера устройства, вызвавшего прерывание.
5. Выбрать следующий запускаемый процесс. Если прерывание привело к готовности какого-то ранее заблокированного процесса, имеющего высокий уровень приоритета, то теперь может быть выбран запуск именно этого процесса.
6. Установить контекст (MMU, TLB, ...) для следующего процесса.
7. Загрузить регистры нового процесса.
8. Запустить выполнение нового процесса.

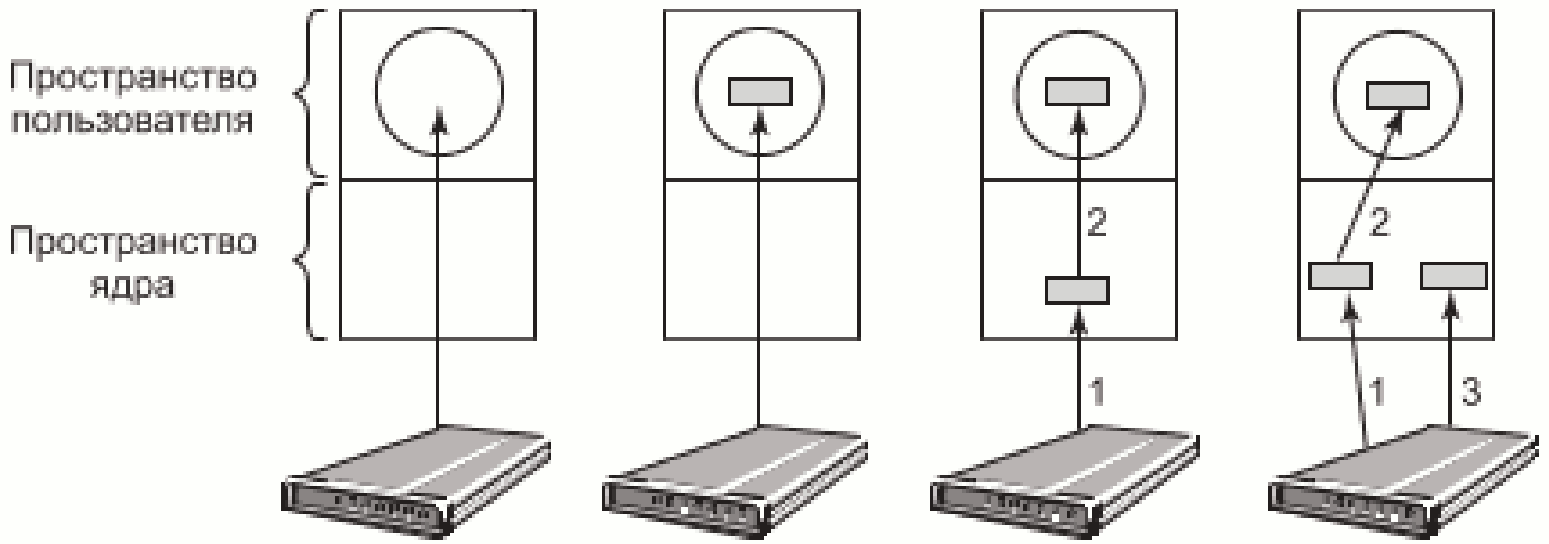
Уровни ПО ввода/вывода: драйверы устройств

Задачи:

- Как инициализировать устройство?
- Как управлять энергопотреблением устройства?
- Проверить корректность входных параметров, перевести абстрактные понятия (номер блока) в конкретные (номер дорожки, головки, сектора).
- Используется ли устройство в данный момент? (Формирование очереди запросов; проверка готовности устройства; запуск после простоя).
- Восприятие абстрактных запросов на чтение и запись от HAL, перевод их в последовательность команд, записываемых в регистры контроллера устройства, и отслеживание порядка их выполнения.
- Драйвер должен быть реентерабельным.
- Возможно ли горячее подключение?

Уровни ПО ввода/вывода: HAL

- Предоставление унифицированного и-фейса для драйверов устройств.
- Буферизация.



- Сообщения об ошибках.
- Распределение и высвобождение выделенных устройств.
- Предоставление размера блока, не зависящего от конкретных устройств.

Уровни ПО ввода/вывода: уровень пользователя

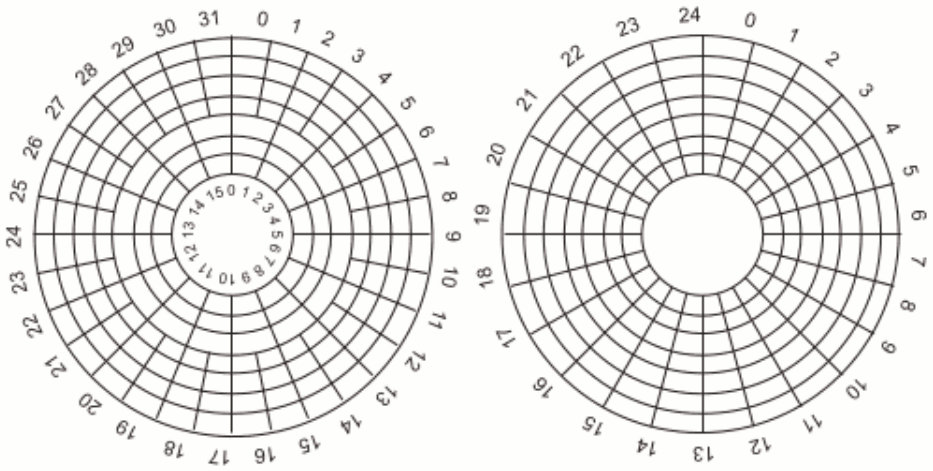
(Например: библиотеки, прикомпонованные к пользовательским программам; демоны спулинга.)

- «Оборачивание» системных вызовов.
- Форматирование.
- Спулинг (формирование очередей запросов).

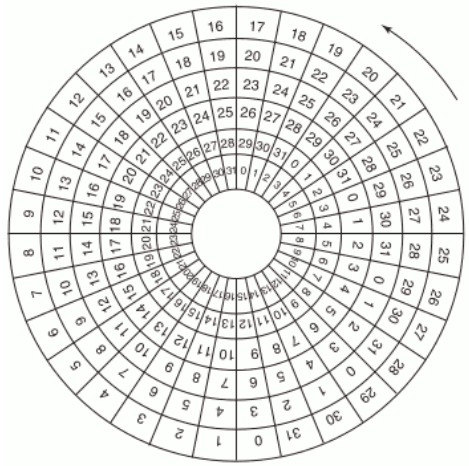
Особенности работы с некоторыми устройствами ввода/вывода

Диски: особенности геометрии

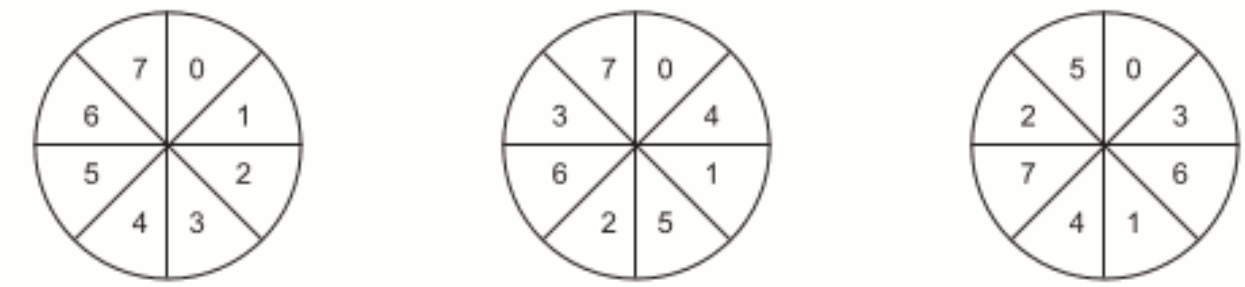
Реальная и виртуальная геометрия диска:
($\rightarrow$ *LBA*)



Отклонение («перекос») цилидров (cylinder skew):

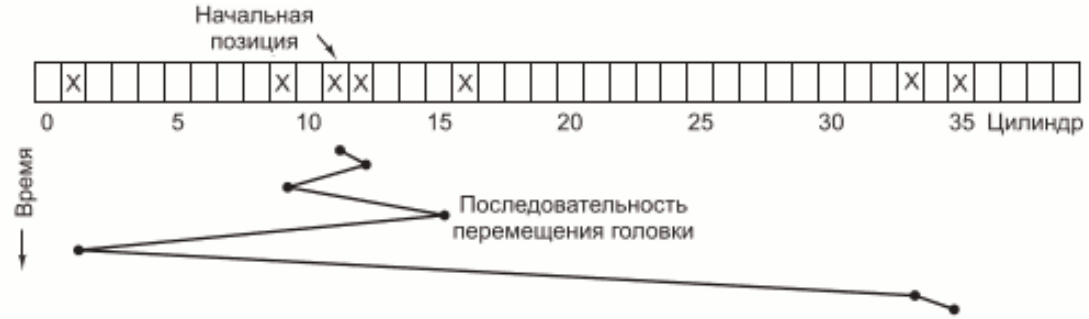


Чередование секторов (interleaving):

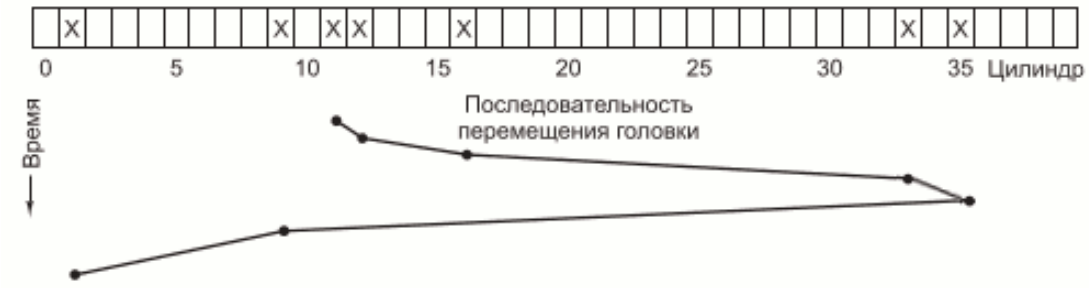


Диски: алгоритмы планирования перемещения головок

- «Первым пришёл – первым обслужен» – First-Come, First-Served (FCFS).
- Позиционирование на ближайший цилиндр – Shortest Seek First (SSF).



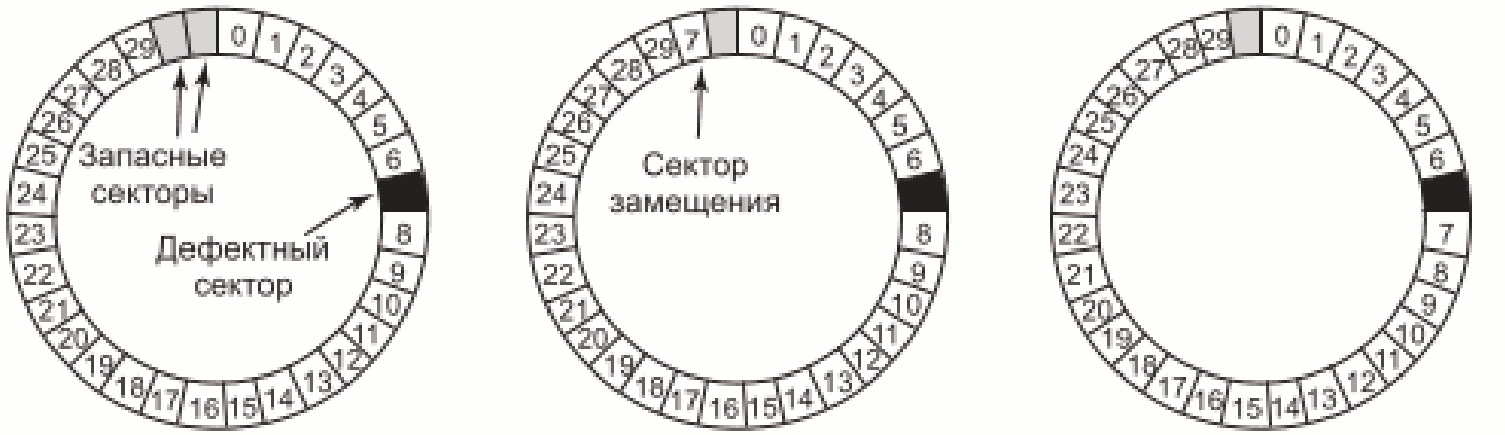
- Алгоритм лифта – Elevator algorithm.



Не имеют смысла для виртуальной геометрии

Диски: обработка ошибок

- Замещение сбойных секторов на уровне контроллера:



(без сдвига или со сдвигом)

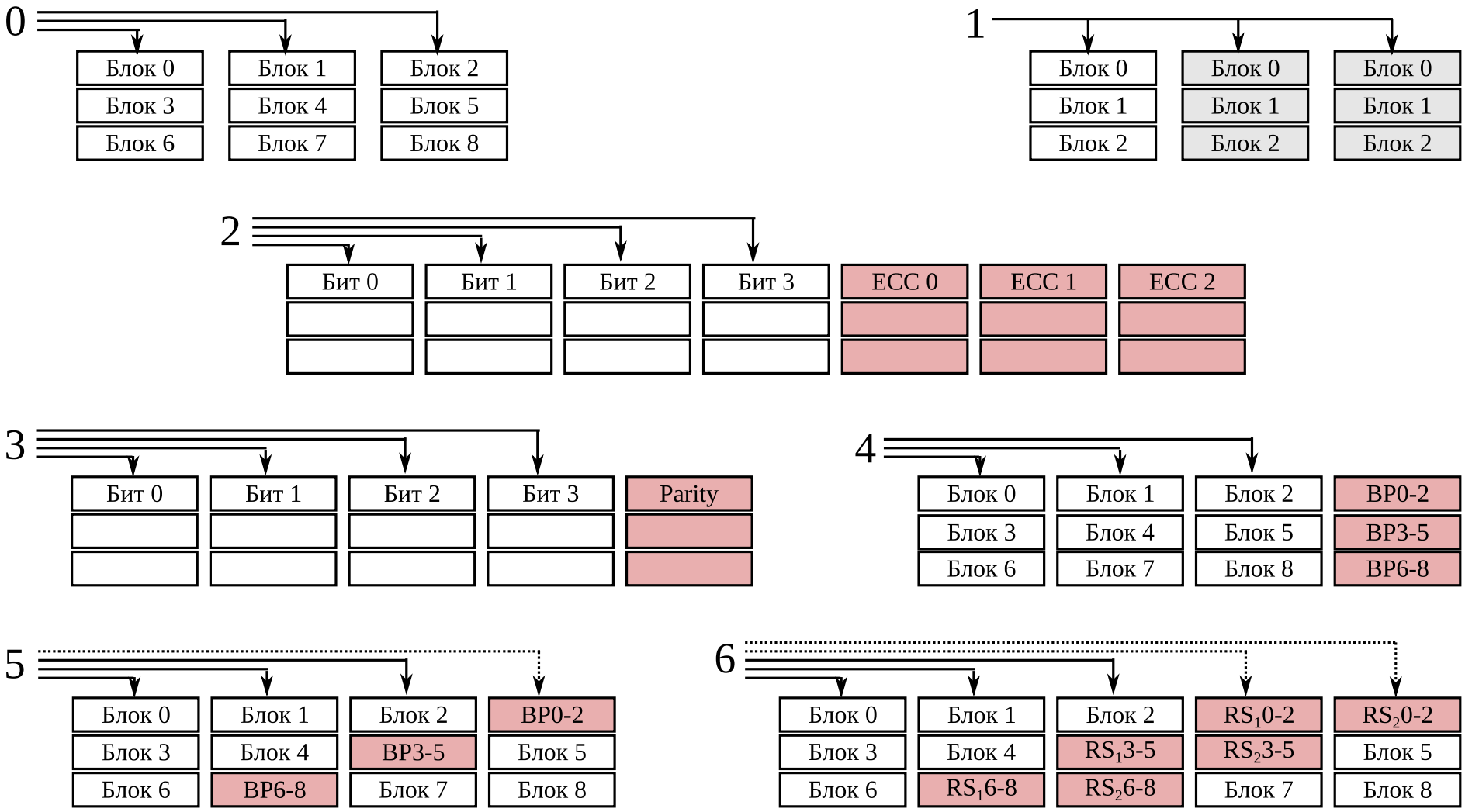
- Обработка сбойных секторов на уровне файловой системы.

Диски: RAID (характеристики)

RAID = Redundant Array of Inexpensive/Independent Disks
SLED = Single Large Expensive Disk

N дисков	Ёмкость	Доп.сбойных	Надёжность	R/W-скорость
RAID 0 (Stripping): без избыточности				
от 2	$S \times N$	нет	очень низкая	высокая
RAID 1 (Mirroring): зеркалирование				
от 2	S	$N-1$	высокая	высокая/средняя
RAID 2 (Byte Stripping with ECC): код Хемминга, $k = \log_2(N+1)$				
от 5/7	$S \times (N-k)$	1	средняя	средняя
RAID 3 (Byte Stripping with Parity Drive): контроль чётности				
от 3	$S \times (N-1)$	1	средняя	средняя
RAID 4 (Block Stripping with Parity Drive): контроль чётности				
от 3	$S \times (N-1)$	1	средняя	средняя
RAID 5 (Block Stripping with Stripped Parity): контроль чётности				
от 3	$S \times (N-1)$	1	средняя	средняя
RAID 6 (Block Stripping with Double Distributed Parity): коды Рида–Соломона				
от 4	$S \times (N-2)$	2	высокая	высокая/низкая

Диски: RAID (схемы)



Часы (1)

Аппаратура: кварцевый генератор периодических импульсов + счётчик + регистр. Переполнение/обнуление счётчика $\longrightarrow$ прерывание ЦП.
Режимы работы: однократный запуск, циклический запуск.
Циклические прерывания = такт системных часов (clock ticks).
Кроме того, есть резервные часы (энергонезависимые, питающиеся от батарейки) – *часы реального времени* (realtime clock). Системные часы считывают с них время при старте.

Задачи драйвера часов:

- Ведение показаний времени суток.
- Предотвращение работы процессов дольше позволенного.
- Обработка системного вызова alarm (пользовательские таймеры).
- Предоставление сторожевых программируемых таймеров для компонентов самой ОС.
- Ведение аналитического, мониторингового и статистического сбора информации.

Часы (2)

Проблема разрядности счётчика времени:

- подсчёт в тактах (50 Гц) – 32 бита хватает на 2 года,
- подсчет в секундах – 32 бита хватает на 136 лет.

Варианты решения:

- 64-битный счётчик тактов,
- 32-битный счётчик секунд + счётчик тактов (с начала секунды/суток).

Проблема нескольких таймеров при наличии одних часов.

Запросы на прерывание от таймера выстраиваются в связанный список. Пример для 4203, 4207, 4213, 4215 и 4216.

Альтернатива: программные таймеры (время проверяется по любому системному вызову).

Интерфейсные устройства: клавиатура

Перевод скан-кодов нажатых клавиш в ASCII-коды символов или управляющие действия. Прерывания генерируются по нажатию/отпусканию клавиш.

Режим без обработки (*неканонический*):

dste ←←← *ate*⏎

Режим с обработкой (*канонический*, требуется буфер):

date

Режим с *эхопечатью* (*echoing*, с отображением) или без отображения.

Обработка специальных символов: TAB, ENTER (CTRL+M), CTRL+H (Backspace), CTRL+Q, CTRL+S, CTRL+D, CTRL+\, ...

Интерфейсные устройства: мышь

Минимальное фиксируемое перемещение = 1 микки (примерно 0.1 мм).

Сообщение от мыши содержит: Δx, Δy, кнопки. Частота сообщений ≈ 40 Гц.

Учёт и различение кликов: отличить двойной от одинарного (два клика достаточно близки в пространстве (в микки) и также близки по времени (в миллисекундах)).

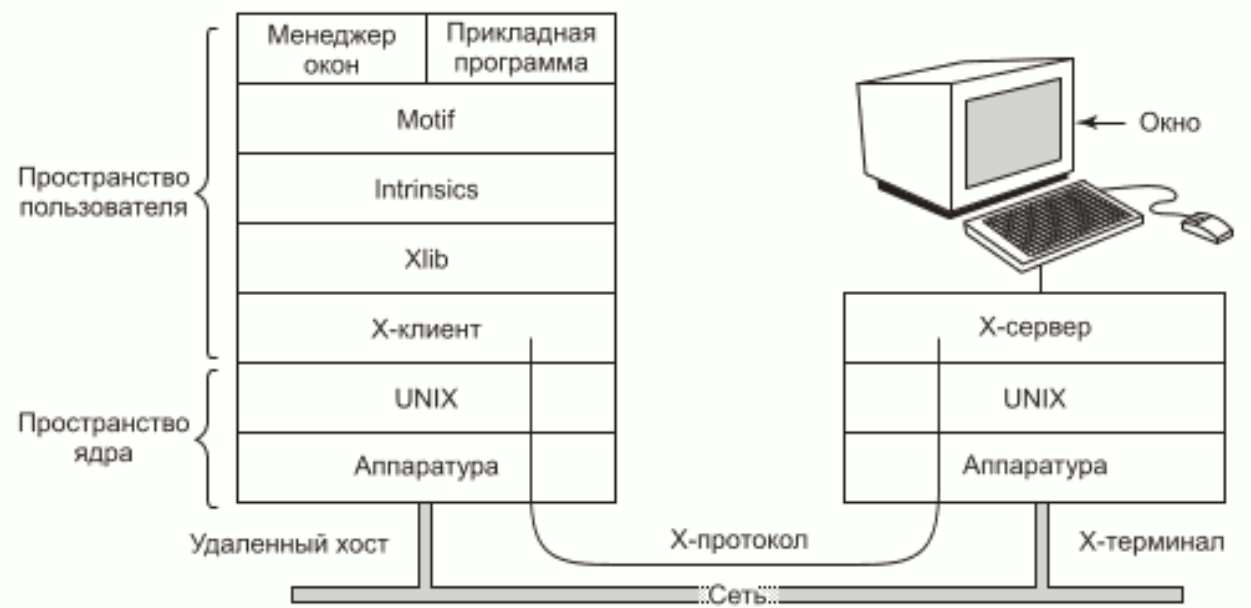
Интерфейсные устройства: текстовый дисплей

Кроме посимвольного вывода иногда требуется обновлять экран сложным способом, например: переместить курсор, вставить/удалить символы/строки с позиции курсора и т. д. $\longrightarrow$ *управляющие (escape) последовательности*.
Для разных терминалов – разные последовательности.
termcap – база терминалов (введена в BSD), впоследствии – terminfo.
Стандарт ANSI на escape-последовательности:

ESC [*n* A	Перемещение вверх на *n* строк
ESC [*n* B	Перемещение вниз на *n* строк
ESC [*n* C	Перемещение вправо на *n* позиций
ESC [*n* D	Перемещение влево на *n* позиций
ESC [*m*; *n* H	Перемещение курсора в позицию (*m*, *n*)
ESC [*s* J	Очистка экрана от позиции курсора (0 – до конца, 1 – от начала, 2 – всего экрана)
ESC [*s* K	Очистка строки от позиции курсора (0 – до конца, 1 – от начала, 2 – всей строки)
ESC [*n* L	Вставка *n* строк в позицию курсора
ESC [*n* M	Удаление *n* строк с позиции курсора
ESC [*s* m	Включение отображения (0 – нормального, 4 – полужирного, 5 – мигающего, 7 – инвертированного)

Интерфейсные устройства: графический дисплей

GUI (Graphical User Interface), Xerox PARC Alto (1973)
Четыре наиболее важных абстракции (WIMP): окна – Windows, значки – Icons, меню – Menus и указывающие устройства – Pointing device.
ПО GUI может быть в пространстве пользователя (UNIX – X Window System) либо в пространстве ядра (Windows).
Графический адаптер: видеопамять + спецпроцессор. 1920x1800 x 24 бита x 25 кадров/сек $\approx$ 260 Мбайт/с.



Сообщения протокола X11:

- команды вывода графики от программы к PC;
- ответы PC на программные запросы;
- извещения о событиях клавиатуры, мыши и других устройств;
- сообщения об ошибках.

Управление энергопотреблением

Наиболее энергопотребляющие у-ва: дисплей (подсветка), ЦП, НЖМД.
Режимы работы: включенный, спящий, ждущий (stand-by), выключенный.

Реализовать алгоритмы и эвристические правила, позволяющие принимать правильные решения о том, что и когда следует отключать («правильные» носит сугубо субъективный характер).
Дисплей: регулирование подсветки (в разных частях дисплея разная?)
НЖМД: раскрутка диска достаточно громоздкий и длительный процесс, поэтому алгоритмы как правило консервативные. Ёмкий кэш не помешает.
ЦП: регулирование таковой частоты и/или напряжения питания (понижение производительности).
Память: сброс на диск и отключение.
Беспроводная связь: в WiFi БС рассылает beacon frame периодически, если нечего передавать, до следующего beacon frame можно спать.
Управление температурным режимом: вкл./выкл. вентиляторов.
Управление/сбор статистики с «умного» аккумулятора.
Прикладное ПО регулирует свою производительность.