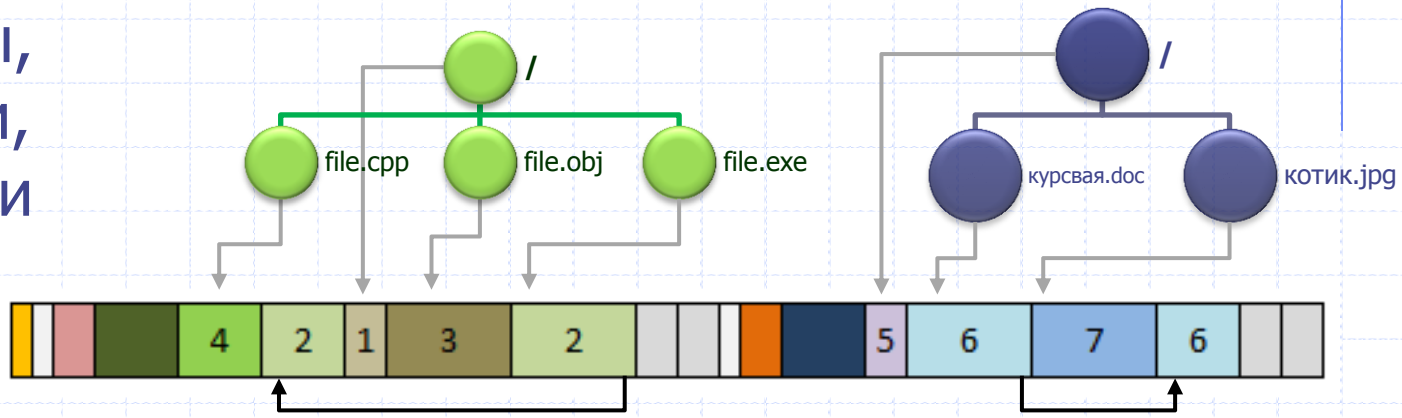
Организация файловых систем

Уровни организации дискового пространства

Файлы, директории, ссылки



Системные файлы

суперблок (заголовок ФС)

структуры данных ФС

блоки данных

Блоки ФС

Разделы

Сектора диска

Размер блока

- Маленький размер
  + уменьшает внутреннюю фрагментацию
  - увеличивает размеры системных структур данных
  - ограничивает размер ФС
  - снижает скорость работы с диском
- Большой размер

Учёт свободного пространства

- При помощи битового вектора
  + простота
  + эффективность
  - может иметь большой размер
- При помощи связанного списка
  - долго просматривать
- Как файл
  + занимаемое служебными структурами место пропорционально объему свободного пространства

Выделение непрерывной последовательности блоков

Файл характеризуется адресом первого блока и длиной (в блоках)

\+ Легко реализовать

\+ Хорошее быстродействие

\- Внешняя фрагментация

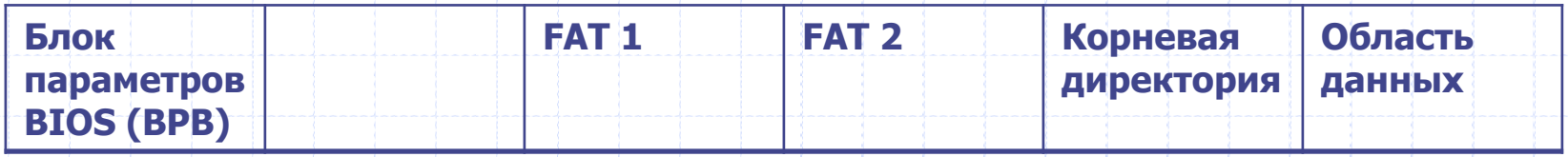
\- Сложности при изменении размера файла

Пригоден для стационарных ФС

File Allocation Table (FAT)

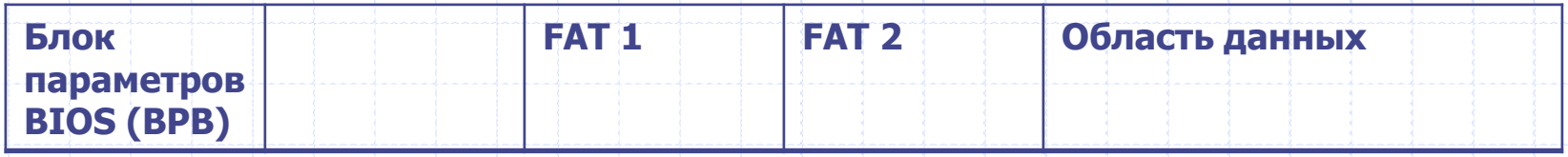
FAT 12, FAT 16

зарезервированные сектора



информация о размере ФС и расположении системных структур
загрзучик ОС

FAT 32



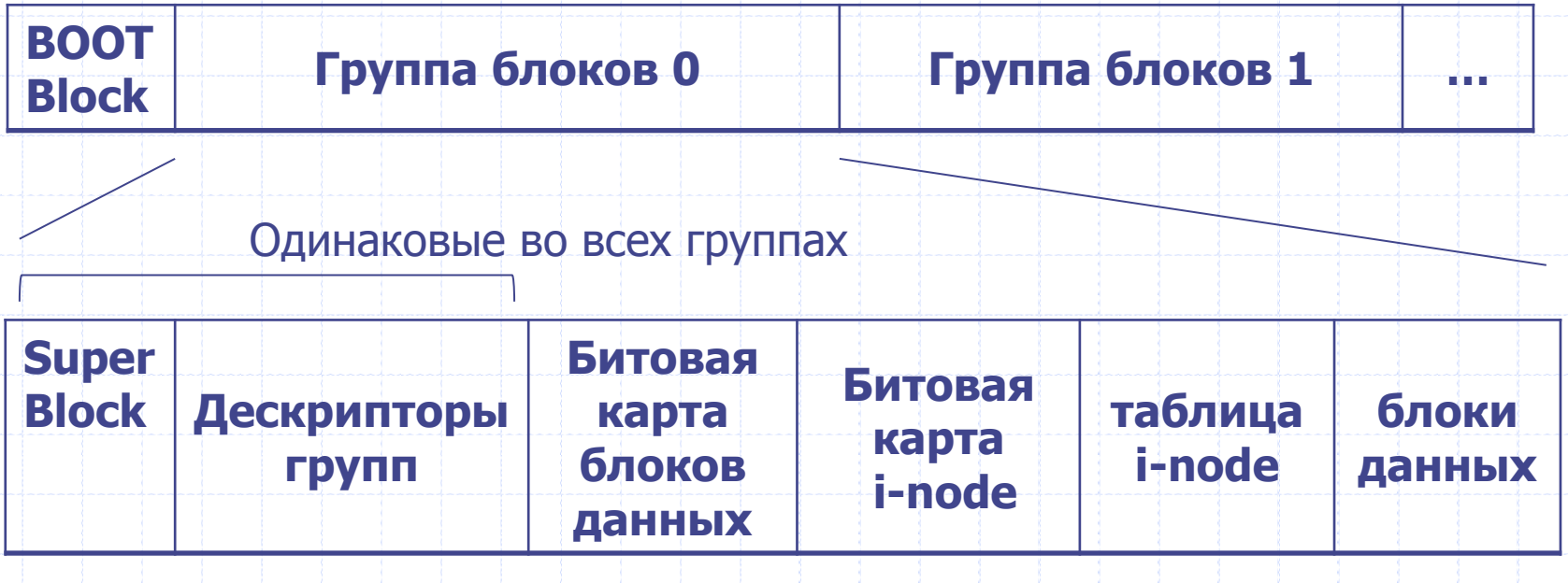
FS Info
Копия загрузочного сектора
Продолжение загрузчика ОС

File Allocation Table (FAT)

- Создаётся таблица, в которой для каждого блока хранится номер следующего за ним.
- Запись в директории указывает на первый блок файла.

№	1	2	3	4	5	6	7	8	9	10	11
FAT		10	11		EOF	2	EOF			7	5

Файловые системы UNIX EXT2



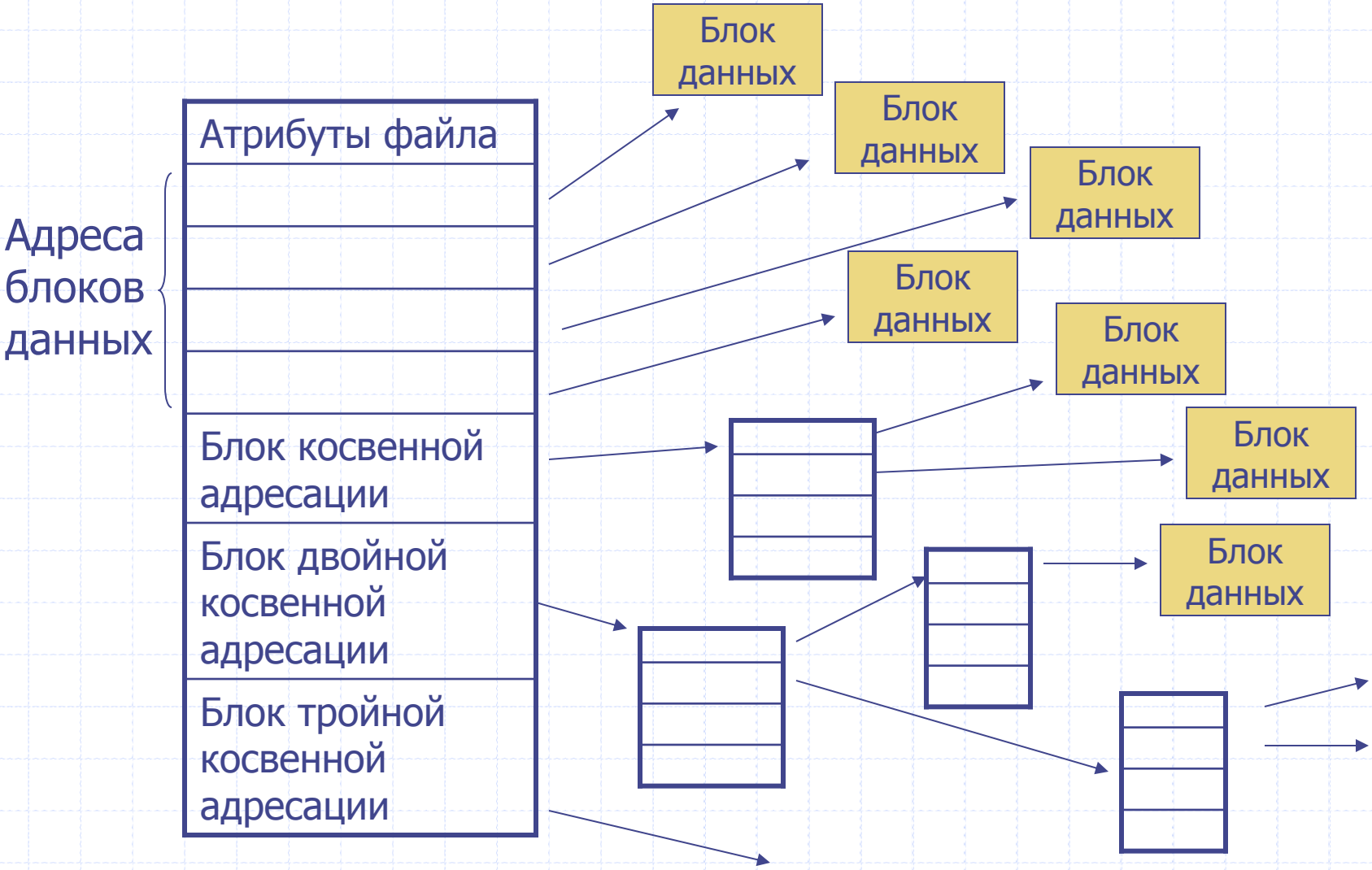
Для каждой группы:

- Номера блоков битовых карт блоков данных и i-node
- Начало таблицы i-node
- Число свободных блоков данных и i-node
- Число директорий

Индексные узлы (i-node)

- С каждым файлом связывается специальная структура, называемая i-node (индексный узел)
- I-node содержит метаинформацию файла, включая расположение блоков данных
- Указатели из таблицы ссылаются на блоки файла
- Запись в директории ссылается на i-node
- Для больших файлов применяется косвенная адресация (многоуровневые таблицы)

Индексные узлы (i-node)



Производительность файловой системы

- Кэширование
- Оптимальное размещение на диске
- Планирование дисковых операций

Кэширование

Размещение в оперативной памяти буфера, содержащего ряд блоков диска

- Замещение блоков
  - С учётом важности (пользовательские, индексных узлов, директорий, …)
- Отложенная запись
- Взаимодействие с виртуальной памятью

Оптимальное размещение

- Желательно размещать данные одного файла (индексного узла) последовательно
- Желательно размещать индексные узлы рядом с данными
- Устранение фрагментации
  - Предотвращение
    - Предварительное резервирование (ОС или пользователем)
    - Отложенное выделение
  - Дефрагментация
    - Во время работы
    - Отдельными утилитами

Планирование операций ввода-вывода

Цель планирования – сокращение времени выполнения операций.

- Время обмена информацией между магнитной головкой и ОЗУ
- Время позиционирования
  - Время перемещения головок (seek time)
  - Задержка на вращение (latency time)

Алогритм FCFS

Запросы организуются в очередь и обслуживаются в порядке поступления

+ простота

- низкая эффективность

Алгоритм SSTF

Short Seek Time First – в начале обрабатываем запросы с минимальным отклонением от текущего положения головок.

- может приводить к длительным задержкам отдельных запросов
- не является оптимальным

Алгоритмы сканирования

Scan: Перемещаемся с одного края диска на другой и выполняем подходящие запросы.

Look: После обработки последнего запроса в текущем направлении поворачиваем обратно.

C-Scan: Запросы обрабатываются только при проходе в одну сторону.

Алгоритмы сканирования

N-Step Scan:

Очередь запросов разбивается на блоки из N элементов, каждый блок обрабатывается согласно алгоритму SCAN

+ Не отдаёт предпочтение одному краю диска если запросы постоянно поступают

N=0: FCFS

N→∞: SCAN

Алгоритмы сканирования сравнение

Диск с 200 цилиндрами (0…199)

Последовательность обращений к дорожкам:

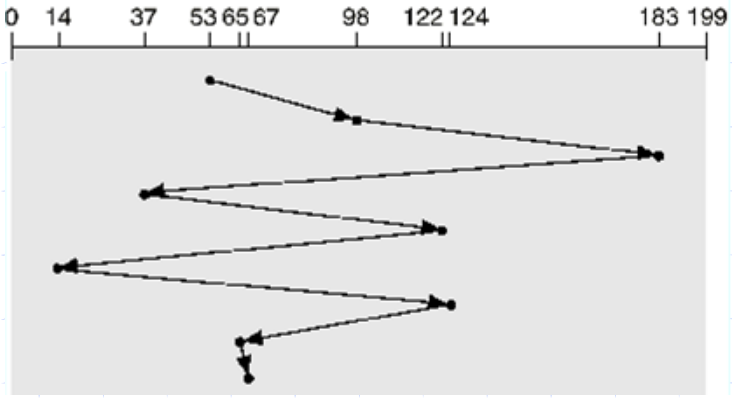
98 183 37 122 14 124 65 67

Начальное положение: 53

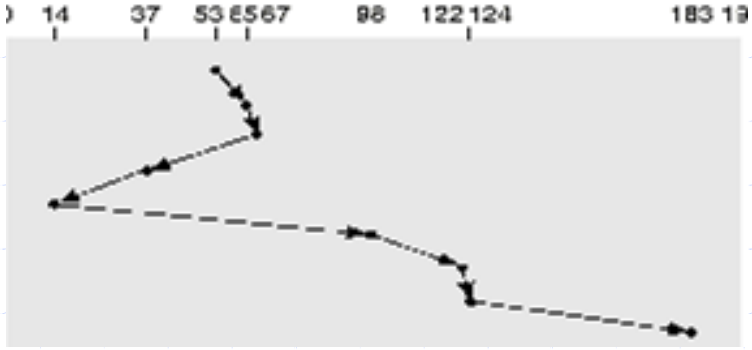
Алгоритм	Число перемещений
FCFS	640
SSTF	236
SCAN	236
Оптимальное	208

Алгоритмы сканирования сравнение

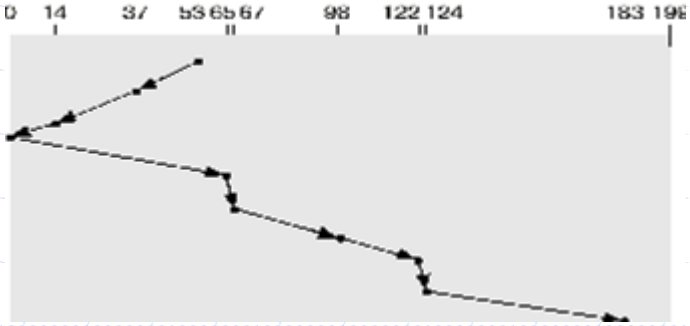
FCFS



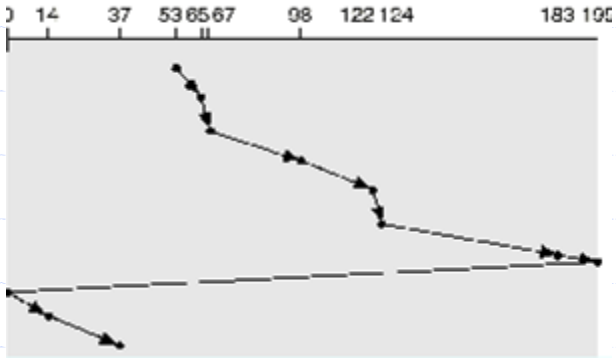
SSTF



SCAN



C-SCAN



Ошибка в GZIP

Задача: сжать файл, оригинал удалить.

```
ifd = open("foo", O_RDONLY)
ofd = open("foo.gz", O_WRONLY | O_CREAT | O_EXCL, 0600)
if (ofd < 0)
        error()
Много чтений и записи:
   read(ifd, buf, bufsize) // каждый раз проверяются ошибки
   write(ofd, buf, amount_to_write) // каждый раз проверяются ошибки
if (close(ofd) != 0)
    error();
if (close(ifd) != 0)
    error();
 if (unlink("foo") != 0)
    error();
exit(0);
```

Записывает один блок

Записывает другой блок

Могут быть переупорядочены

Если в этот момент вырубить питание, данные могут быть потеряны

Ошибка в GZIP - исправление

Задача: сжать файл, оригинал удалить.

```
ifd = open("foo", O_RDONLY)
ofd = open("foo.gz", O_WRONLY | O_CREAT | O_EXCL, 0600)
if (ofd < 0)
        error()
Много чтений и записи:
   read(ifd, buf, bufsize) // каждый раз проверяются ошибки
   write(ofd, buf, amount_to_write) // каждый раз проверяются ошибки
if (close(ofd) != 0)
    error();
if (close(ifd) != 0)
    error();
 if (unlink("foo") != 0)
    error();
exit(0);
```

```
if (fsync(ofd) != 0)
    error();
```

- Данные не могут быть потеряны.
- Работает медленней.

Особенности FS для носителей на Flash памяти

- Стирание блоков
  - Блок не может быть перезаписан без предварительного стирания. Целесообразно стирать неиспользуемые блоки во время простоев.
  - Операция стирания может быть выполнена только над большими фрагментами памяти.
- Произвольный доступ
  - Нет необходимости избегать операций на не последовательных участках.
- Распределение износа
  - Желательно равномерно нагружать все участки носителя.

Файловая система NTFS

- Журналирование
  - Используется для системных структур ($LogFile)
- Запись всех изменений на томе / Change Journal
- Сжатие
- Шифрование / Encrypting File System (EFS)
- Квоты

Файловая система NTFS

- Альтернативные потоки данных / Alternate Data Streams (ADS)
- Разряженные файлы
- Хранение одной копии / Single Instance Storage (SIS)
- Жесткие ссылки
- Теневая копия / Volume Shadow Copy Service (VSS)
- Транзакции / Transactional NTFS

Файловая система NTFS

- Точки повторной обработки / Reparse points
- Точки монтирования / Volume Mount Points
- Переходы между директориями / Directory Junction
- Символические ссылки
- Иерархическое управление носителями / Hierarchical Storage Management (HSM)

Строение NTFS

- Файл в NTFS представляется набором атрибутов

  Потоки данных, имя, информация о расположении, дескриптор безопасности,…

- Используется 16-битная кодировка имен файлов
- Для индексации данных используются B+ деревья
- Атрибуты всех файлов хранятся в Master File Table (MFT)
- Запись директории содержит имя файла, индекс записи в MFT и индекс повторного использования записи

Некоторые системные файлы NTFS

Индекс	Имя	Назначение
0	$MFT	Master File Table
1	$MFTMirr	Копия критических записей MFT (0…3)
2	$LogFile	Журнал транзакций метаданных файловой системы
3	$Volume	Описание тома (флаги, метка тома)
4	$AttrDef	Таблица соотвествий атрибутов именам для MFT
5	.	Корневая директория
6	$Bitmap	Битовая маска занятых/свободных кластеров
7	$Boot	Загрузочный сектор (загрузчик, расположение MFT)
8	$BadClus	Файл со списком кластеров с поврежденными секторами
9	$Secure	База данных ACL
11…26		Расширения
27		Первый индекс для обычных файлов

Атрибуты и потоки

- Для каждого файла создаётся запись MFT, содержащая список атрибутов (потоков) переменной длины, описывающий все потоки, связанные с файлом
  - Размер записи 1 кб, дополняется если меньше
  - Стандартные атрибуты файлов хранятся в записи фиксированного размера
- Атрибуты имеют
  - Тип
  - Имя (опционально)
  - Данные (опционально)
- Некоторые потоки не могут иметь имени
  - Имя файла
  - Короткое имя файла
- Для обычных файлов
  - Данные лежат в безымянном потоке типа $DATA
  - Альтернативные потоки имеют имя и тип $DATA

Резидентные и нерезидентные потоки

NTFS предпочитает хранить данные потоков внутри дескриптора в MFT. Такие потоки называют резидентными.

- Может целиком уместиться файл размером 700…800 байт
- Некоторые потоки должны быть резидентными
  - Имя, обязательные атрибуты, таблицы размещения для нерезидентных потоков
- Некоторые потоки не могут быть резидентными
  - Сжатые, зашифрованные и разряженные
- Если таблица размещения данных потока не помещается в MFT то используется косвенная адресация
- Если дескрипторы потоков не помещаются в записи MFT то для дополнительных дескрипторов используется нерезидентный поток

Хранение нерезидентных потоков

- Нерезидентные данные хранятся в последовательностях кластеров, называемых run (полоса)
- Каждая полоса представляется начальным кластером и длиной
- Начальные кластеры кодируются смещением от начального кластера предыдущей полосы
- Длина и смещение хранятся в полях переменного размера (1…8 байт), число байт для каждого записано в первом байте записи, описывающей полосу
- Для пустых фрагментов разряженных файлов смещение равно 0.

Пример описания нерезидентных потоков

- Описание: 31 38 73 25 34 32 14 01 E5 11 02 31 42 AA 00 03 00
- Полоса 1: 31 38 73 25 34
  - Заголовок 0x31 – смещение 3 байта, размер 1 байт
  - Размер 0x38
  - Смещение 0x342573
- Полоса 2: 32 14 01 E5 11 02
  - Заголовок 0x32 – смещение 3 байта, размер 2 байт
  - Размер: 0x114
  - Смещение: 0x211E5 (кластер 0x363758=0x211E5+0x342573)
- Полоса 3: 31 42 AA 00 03
  - Заголовок 0x31 – смещение 3 байта, размер 1 байт
  - Размер 0x42
  - Смещение 0x300AA (кластер 0x393802=0x363758+0x300AA)
- Полоса 4: 00
  - Заголовок 00 – конец
- Файл состоит из полос:
  - 56 (0x38) кластеров начиная с кластера 0x342573
  - 276 (0x114) кластеров начиная с кластера 0x363758
  - 66 (0x42) кластера начиная с кластера 0x393802

Пример описания разряженного потока

- Описание 11 30 20 01 60 11 10 30 00
- Полоса 1: 11 30 20
  - Заголовок 0x11 – смещение 1 байт, размер 1 байт
  - Размер 0x30
  - Смещение 0x20
- Полоса 2: 01 60
  - Заголовок 0x01 – смещение 0 байт, размер 1 байт
  - Размер: 0x60
  - Смещение: 0 – разряженный (пустой) блок
- Полоса 3: 11 10 30
  - Заголовок 0x11 – смещение 1 байт, размер 1 байт
  - Размер 0x10
  - Смещение 0x30 (кластер 0x50=0x20+0x30)
- Полоса 4: 00
  - Заголовок 00 – конец
- Файл состоит из полос:
  - 48 (0x30) кластеров начиная с кластера 0x20
  - 96 (0x60) пустых кластеров
  - 16 (0x10) кластера начиная с кластера 0x50

Представление сжатых данных

- Для сжатия данные разбиваются на блоки, по 16 кластеров
- Если блок удалось сжать, то он представляется одной или несколькими полосами общей длиной меньше 16 и затем пустой (смещение=0) полосой, дополняющей до 16 кластеров
- Если есть полосы длиной более 16 кластеров то данные не удалось сжать
- Если полос с данными нет совсем то это было 16 кластеров нулей

Пример описания сжатого потока

- Описание 11 08 40 01 08 11 10 08 00
- Полоса 1: 11 08 40
  - Заголовок 0x11 – смещение 1 байт, размер 1 байт
  - Размер 0x08
  - Смещение 0x40
- Полоса 2: 01 08
  - Заголовок 0x01 – смещение 0 байт, размер 1 байт
  - Размер: 0x08
  - Смещение: 0 – пустой (сжато) блок
- Полоса 3: 11 10 08
  - Заголовок 0x11 – смещение 1 байт, размер 1 байт
  - Размер 0x10
  - Смещение 0x08 (кластер 0x48=0x40+0x08)
- Полоса 4: 00
  - Заголовок 00 – конец
- Файл состоит из полос:
  - 8 (0x08) кластеров сжатых данных (16 распакованных) начиная с кластера 0x40
  - 16 (0x10) кластеров обычных данных начиная с кластера 0x48