Что намерено: полный свод

Содержание

- 1. Кадр: сколько его в куске
- 2. Частоту не надо угадывать
- 3. Пакет — не кадр
- 3.1. Ключевые кадры: пакетами, а не декодированием
- 3.2. Час материала и отмена на живом FFmpeg
- 4. Синхрон звука: считается в миллисекундах
- 5. Проверка «без потерь»
- 6. Контейнер и кодек: что во что кладётся
- 7. Метаданные: что переживает рез
- 8. Предел инструмента: камерный RAW
- 9. Растяжение времени: фильтр съедает 15 мс
- 10. Грабли FFmpeg, найденные измерением
- 11. Случаи, когда измерение соврало
- 12. Чем это меряется
- 4.1. Дрейф: его нет, и вот чем это показано
- 12.1. Вещательный MXF: 4K, десять потоков, служебные данные
- 12.2. Разделение: части складываются в исходник
- 12.3. Звуковые файлы: рез по сэмплу
- 13. Чего мы не проверяли

> **Про интерфейс в этом документе.** Точки реза находят в плеере mpv, который открывается кнопкой из раздела НАРЕЗКА, и переносят в поля кнопками — либо вводят таймкодом с клавиатуры. Измерения ниже — о движке и FFmpeg, от окна они не зависят.

Рабочая тетрадь всего, что меряли по точности резки, синхрону и пределам FFmpeg. Отсюда берутся числа для таблиц отказов в коде и для решений; началась эта работа с разбора Shutter Encoder.

Правило одно: **ни одно утверждение здесь не взято из документации.** Каждая ячейка таблицы — реальная попытка, каждое число — замер. Где измерение оказалось неверным, это тоже записано: в разделе 11 собраны случаи, когда правдоподобный вывод пришлось отменить.

Версия FFmpeg: **9.0.1**; ключевые числа переснимались на 8.0.1 и совпадают, расхождения отмечены отдельно. В поставку версия попадает не отсюда — её выбирает установщик по версии драйвера NVIDIA, и для большинства машин это ветка 8.x. Поэтому проверять приходится обе.

Порядок величин, за которые идёт разговор: **кадр** (33 мс на 29,97), **звуковой пакет** (8 мс), **сэмпл** (20 мкс). Смещение на один кадр в начале смены — это рассинхрон, который монтажёр будет искать руками.

---

1. Кадр: сколько его в куске

Просили отрезок — сколько кадров вышло. Считается точной дробью, а не десятичной частотой.

Частота	Дробь	Запрос	Правильный ответ	Через `round(fps)`
23,976	24000/1001	50 с	**1199**	1200
29,97	30000/1001	4 мин	**7193**	7200
59,94	60000/1001	4 мин	**14386**	14400
59,94	60000/1001	50 с	**2998**	3000
25	25/1	4 мин	6000	6000

Разница между колонками и есть ×1.001: **3,9 с на час**.

Две детали, на которых легко ошибиться повторно:

- **Точка во времени и таймкод — разные величины.** Non-drop `00:04:00:00` на 29,97 — это кадр 7200, то есть 240,24 с реального времени.
- **Конец отрезка округляется вверх.** Кадр 899 на 29,97 стоит на 29,9966 с — внутри тридцатой секунды, значит он в куске. Округление к ближайшему теряет его, и рез выходит короче на

кадр. Так же и с 2998 кадрами выше: кадр 2997 стоит на 49,99983 с, то есть внутри пятидесятой секунды.

Проверено на материале

Источник	Запрос	Кадров	Расхождение
ProRes 25p	3 → 9	150	0
HEVC long-GOP 29,97	12 → 30	600	0 (после добора, см. раздел 3)
H.264 all-intra 23,976	3 → 9	144	0
29,97 drop-frame	5 → 15	330	0
59,94 drop-frame	3 → 9	360	0
MPEG-TS (старт 1,44 с)	2 → 8	300	0
4 канала PCM	2 → 8	150	0
Canon HEVC 4:2:2 10 бит	3 → 10	166	0
ARRI ProRes 4444 XQ 24,000	2 → 8	144	0
VFR (30 → 17,6)	4 → 16	по времени	+1,07 с, объявлено

2. Частоту не надо угадывать

Соблазн, которому мы поддались и от которого отказались: «камеры пишут круглое там, где идёт дробное, значит 24 надо читать как 23,976». Проверка показала, что `ffprobe` **не округляет**:

Файл	`r_frame_rate`	Измерено (кадры / длительность)
Canon MVI_0757	`24000/1001`	23,97679
Тест-клип 23,976	`24000/1001`	23,97614
ARRI Alexa Mini	`24/1`	**24,00000** (260 кадров на 10,8333 с)

Alexa снимает любую частоту, и снятое на честные 24,000 объявляет `24/1`. Читать такой файл как 23,976 — это создать ошибку ×1.001, а не устранить её: пять кадров за четыре минуты, ровно

то же, за что критикуется чужая нарезка.

Вывод: частота берётся у файла как есть. Ту, что не удалось прочитать, честнее отклонить, чем заменить догадкой.

---

3. Пакет — не кадр

`-count_packets` быстр именно потому, что не декодирует: 0,14 с против 5,7 с на четырёх минутах 59,94. Обычно одно равно другому — но не на хвосте long-GOP.

Что считали	**HEVC long-GOP, запрос 600 кадров**
Пакетов записано	**600**
Кадров декодируется	**597**
Длительность потока	19,920 с вместо 20,020

Последние три B-кадра ссылаются на картинку за границей реза, декодер их отбрасывает. Файл короче на **0,1 с**, при этом все проверки рапортуют успех. То же самое на 8.0.1 — значит это не регрессия версии, а слепое пятно метода.

Недобор устойчив и линеен:

Просим пакетов	**600**	**601**	**602**	**603**	**604**
Декодируется кадров	597	598	599	**600**	601

Выдаёт проблему `duration` потока — она уже считает по-честному и достаётся даром. Поэтому декодировать стоит только тогда, когда длительность не сошлась с планом больше чем на полкадра.

3.1. Ключевые кадры: пакетами, а не декодированием

Рез и сведения о файле спрашивают, где лежат ключевые кадры.

Поиск ключевых кадров декодировал картинки. `-skip_frame nokey` пропускает не-ключевые кадры — а на all-intra материале пропускать нечего, и ffprobe честно декодирует всё

окно. На ARRI ProRes 4444 XQ (2944×2160, 12 бит) это **54 секунды** на один вопрос, при том что сам кадр строится за 0,85 с. Пакеты несут флаг `K` и декодера не требуют:

Способ	ARRI ProRes 15 с окна	ProRes 1080p	HEVC long-GOP
`-skip_frame nokey` (декодирует)	55,2 с	0,31 с	0,05 с
`-show_packets` + флаг K	**0,76 с**	**0,05 с**	**0,04 с**

Списки при этом **совпадают полностью**: проверено на десяти источниках — long-GOP, all-intra, VFR, MPEG-TS, drop-frame, ProRes, ARRI, — худшее расхождение по времени 0,0. Это существенно: от того же списка зависит точка реза.

Запуск процесса ничего не стоит; стоит декодирование. Запуск ffmpeg — 0,02 с, один кадр 4K H.264 из файла на 825 МБ — 1,2 с, и уменьшение кадра до 480×270 эту цифру не меняет: платится за то, чтобы кадр добыть — открыть файл, прыгнуть в место и развернуть 3840×2160.

3.2. Час материала и отмена на живом FFmpeg

Всё предыдущее меряно на клипах до минуты. Часовой файл собран склейкой копий **без перекодирования**: 66 копий 55-секундного 4K дали **1 час 25 с, 50,7 ГБ**, H.264 3840×2160 на 23,976.

Что	На часовом файле
Прочитать факты о файле	0,43 с (повторно 0,28 с)
Ключевые кадры на 58-й минуте	0,47 с, повторно из кэша **0,00 с**
Рез 10 с на 58-й минуте	**254 кадра, +0,000 с**, 2,3 с работы
Рез последних 10 с файла	**358 кадров, +0,000 с**, 2,5 с работы

Отмена посреди записи. Механизм был написан, но никогда не проверялся на живом FFmpeg. Проверка: рез получаса из часового файла, отмена через 6 секунд.

- дочерний процесс гасится корректно (`terminate`, пять секунд, затем `kill`);
- операция возвращается как несостоявшаяся, а не как успех;
- **но на диске оставалось 6 ГБ нечитаемого файла** — под именем готового куска. Мультиплексор не успел записать индекс, `moov atom not found`, ни один плеер такое не

откроет, а в папке оно выглядит результатом.

Прежнее правило удаляло только файлы нулевой длины, рассуждая, что частичный результат может быть нужен. Это верно ровно до тех пор, пока частичный файл открывается. Теперь проверяется читаемость: что открывается — остаётся (и в журнале сказано, до какой секунды), что не открывается — удаляется с объяснением. После правки отмена не оставляет ничего.

---

4. Синхрон звука: считается в миллисекундах

Меряется не «есть ли рассинхрон» вообще, а **изменил ли его рез**: у любого источника есть своя задержка, и абсолютное значение ничего не говорит.

Материал — клип, где метка есть в обеих дорожках одновременно: белый кадр в начале каждой секунды и щелчок 20 мс в тот же момент.

Контейнер	Изменение против исходника
MP4	0…2 мс
MOV	0…5 мс
MKV	1…3 мс
MXF	0…5 мс (после перезаписи PCM; до неё было 9…18 мс)

Про MXF отдельно. При простом копировании PCM контейнер приносит с собой сэмплы от границы чанка до точки реза — **384 сэмпла, 8 мс звука**, который относится к моменту **до** реза. Перезапись того же PCM той же разрядности кладёт звук на сэмпл и совпадает с эталонным резом байт в байт. Поколение не теряется: PCM переписывается в PCM.

Ещё одна мелочь того же порядка: MXF принимает PCM **только** little-endian. Canon пишет `pcm_s16be`, поэтому при выводе в MXF байты переставляются — те же сэмплы, порядок, которого требует контейнер.

---

5. Проверка «без потерь»

Заявление «поток скопирован, не пережат» проверяется, а не декларируется.

Картинка. Хэши декодированных кадров куска и того же диапазона источника (`-f framemd5`) — совпадение до байта.

Звук. Жёстче: вытащить звук куска и звук всего источника в raw PCM и поискать первый дословно во втором. На камерном файле нашёлся — и позиция вхождения дала точное смещение реза: **48 сэмплов, 1 мс, сороковая доля кадра**.

Картинка на входе куска (open-GOP). Число кадров ничего не говорит о том, что видно. При open-GOP кадры после ключевого могут ссылаться на картинки до него, и кусок, начатый оттуда, показывает их битыми — обычная претензия к копированию long-GOP. Проверка прямая: сравнить хэши **декодированных** кадров куска с теми же кадрами источника, считая от ключевого кадра, на который лёг вход.

Источник	Вход куска	Сравнено	Совпало
HEVC long-GOP 29,97 (`has_b_frames=2`)	ключевой на 10,010 с	40 кадров	**40**
Камерный 4K H.264 23,976	ключевой на 18,852 с	40 кадров	**40**

Бит в бит, без единого расхождения: на входе куска видно ровно то же, что в источнике.

---

6. Контейнер и кодек: что во что кладётся

Каждая ячейка — запись одной секунды с `-c copy` и **чтение результата обратно** со сверкой кодека. Проверять по коду возврата нельзя: файл может записаться и не читаться.

Видео	MP4	MOV	MKV	MXF	TS	M2TS
H.264	да	да	да	да	да	да
HEVC	да	да	да	**нет**	да	да
ProRes	**нет**	да	да	да	**нет**	**нет**
DNxHD/DNxHR	**нет**	да	да	да	**нет**	**нет**
MPEG-2	да	да	да	да	да	да
AV1	да	**нет**	да	**нет**	**нет**	**нет**
VP9	да	**нет**	да	**нет**	**нет**	**нет**

Звук	MP4	MOV	MKV	MXF	TS	M2TS
PCM 16 LE	да	да	да	да	**нет**	**нет**
PCM 16 **BE** (`twos`, Canon)	да	да	да	**нет**	**нет**	**нет**
PCM 24 LE	да	да	да	да	**нет**	**нет**
PCM 32 float	да	да	да	**нет**	**нет**	**нет**
AAC	да	да	да	**нет**	да	да
AC-3 / E-AC-3	да	да	да	**нет**	да	да
MP3	да	да	да	**нет**	да	да
FLAC	да	**нет**	да	**нет**	**нет**	**нет**
ALAC	да	да	да	**нет**	**нет**	**нет**
Opus	да	**нет**	да	**нет**	да	да

Таблицы пересняты на 9.0.1 целиком и совпали с 8.0.1 до ячейки.

---

7. Метаданные: что переживает рез

Что	MP4	MOV	MKV
Поворот (`rotation=90`)	да	да	да
Язык дорожек	да	да	да
Теги проекта (title, artist)	да	да	да
Камерные теги (make, model, brands)	да	да	частично
Таймкод дорожкой `tmcd`	да	да	**нет** (текстовый тег)

`-map_metadata 0` **не переносит** камерные теги в MP4 — ни `make`, ни `model`, ни бренд вроде `mp42hvc1CAEP`: мультиплексор пишет свои `isomiso2mp41`. Нужен `-movflags use_metadata_tags`.

Таймкод при копировании MOV → MP4 переносится дорожкой, но **значение остаётся прежним** — сдвиг на длину отрезанного надо писать самому.

---

8. Предел инструмента: камерный RAW

Два формата, которые FFmpeg не открывает ни в какой версии:

Файл	Внутри	Что видит ffprobe
Sony BURANO X-OCN LT	X-OCN в MXF OP1a	поток без имени кодека, `width=0`; звук `pcm_s24le` читается
ARRI Alexa Mini `.mxf`	ARRIRAW в MXF	то же самое

Простое копирование даёт «dimensions not set» и файл нулевой длины. Проверено на 8.0.1 и на 9.0.1 — одинаково. Такой материал режется в родном ПО производителя или после транскода, и честнее сказать это до запуска, чем показать нулевой файл.

Тот же ARRI в **MOV** (ProRes 4444 XQ, 2944×2160, 12 бит, 5 дорожек PCM) режется кадр в кадр вместе с таймкодом.

Отдельная мелочь оттуда же: ARRI пишет цвет как GBR с зарезервированными primaries, и `swscale` отказывается делать из этого миниатюру — «Unsupported input». Кадр из такого файла приходится просить, назвав цветовое пространство явно.

---

9. Растяжение времени: фильтр съедает 15 мс

Проверялось для пересчёта частоты, где звук тянется вслед за картинкой.

Отношение	Источник 10 с	Источник 0,25 с
60 → 24	−0,10 %	−3,28 %
120 → 24	−0,13 %	−5,11 %
1000 → 24	−0,16 %	−6,04 %
8000 → 24	−0,16 %	−6,14 %

Потеря зависит **не от коэффициента, а от длины входа**: и `atempo`, и `rubberband` съедают около **15 мс** исходного звука — это их внутренний буфер. На реальном материале это доли процента; на четвертьсекундном обрывке при растяжении в сотни раз — уже секунды. Дописывание тишины на вход (`apad`) возвращает часть, но вносит собственный перелёт, поэтому фильтр оставлен как есть, а граница записана здесь.

Заодно проверено, что источник может быть любым: 8000 кадров в секунду → 24 прошли без потери единого кадра (2000 кадров, ровно 83,3333 с).

---

10. Грабли FFmpeg, найденные измерением

Ни одна не очевидна из документации, и каждая стоила отдельного разбора.

№	Грабля	Проявление
1	`-ss` до `-i` сбрасывает время	`-to` после него считает от нового нуля
2	`round(fps)` на дробных частотах	4 минуты на 29,97 превращаются в 240,27 с
3	Округление конца «к ближайшему»	теряется кадр, стоящий на 29,9966
4	`-t` при копировании	перебор на 1–3 кадра на потоке с B-кадрами
5	`-frames:v` точен только с точным `-ss`	при seek мимо ключевого кадра кусок короче
6	HEVC с B-пирамидой	муксер пишет на 2–4 кадра меньше запрошенного
7	`-avoid_negative_ts make_zero`	видео уезжает на 0,1 с от звука
8	`start_time` ≠ 0 (MPEG-TS 1,44 с)	все точки смещаются на эту величину
9	`avg_frame_rate` на VFR	12 запрошенных секунд превращаются в 16
10	`-n` на существующий файл	«already exists» и **код возврата 0**
11	PCM при копировании в MXF	384 лишних сэмпла (8 мс) в начале
12	`-map_metadata 0` в MP4	камерные теги и бренды теряются
13	`pan=mono|c0=c1` на моно	пишет тишину, код возврата 0
14	MXF + сжатый звук	«Could not write header», код 4294967274

№	Грабля	Проявление
15	Пакет ≠ кадр	600 записано, 597 декодируется
16	Камерный RAW в MXF	«dimensions not set», файл нулевой длины
17	Предупреждение в `stderr` ломает разбор JSON	`json.loads` падает на файле, который читается прекрасно

11. Случаи, когда измерение соврало

Здесь стоит быть особенно честным: каждый из этих выводов выглядел убедительно, и каждый оказался неверным.

«MKV уводит звук на 107 мс». Сравнивались «первая найденная вспышка» и «первый найденный щелчок». Один поток открывался меткой, другой нет — детекторы описывали **разные события**. Оба потока были сдвинуты вместе, рассинхрона не было. Правильно — сопоставлять каждую вспышку со своим щелчком и смотреть на весь набор. Предупреждение, добавленное в продукт по этому «открытию», пришлось снимать.

«ProRes нельзя положить в MXF». Проверка шла с партнёрским звуком AAC, который MXF не принимает вовсе. Ячейка показывала отказ партнёра, а не проверяемого кодека. Партнёрский поток надо подбирать под контейнер.

«HEVC режется точно». Считались пакеты. Кадров на три меньше — см. раздел 3.

«Звук скопирован дословно» — дважды подряд ни о чём. Побитовость проверяется поиском сэмплов куска внутри источника, и оба раза поиск находил совпадение, которое ничего не доказывало. Сначала сигналом была синусоида: она периодична, любой её кусок совпадает с началом, и «найдено на 0,000 с» получается всегда. Потом клип имел единственный ключевой кадр — в нуле, так что рез «с третьей секунды» физически начинался с начала файла, и совпадение снова выходило честным по форме и пустым по смыслу. Проверять надо на неповторяющемся сигнале (розовый шум) и на клипе с частыми ключевыми кадрами (`-g 25`): тогда совпадение находится ровно на запрошенной секунде, и это уже что-то значит.

Отдельно: `blackdetect` работает целыми кадрами, `silencedetect` — окнами; вместе они дают около 40 мс погрешности, то есть **больше**, чем величина, которую обычно меряют. На них легко увидеть сдвиг, которого нет.

12. Чем это меряется

```
# кадры: пакеты (быстро) и декодированные (честно)
ffprobe -v error -select_streams v:0 -count_packets -show_entries
stream=nb_read_packets -of csv=p=0 FILE
ffprobe -v error -select_streams v:0 -count_frames  -show_entries
stream=nb_read_frames  -of csv=p=0 FILE

# ключевые кадры вокруг точки, без скана всего файла
ffprobe -v error -select_streams v:0 -skip_frame nokey -read_intervals
"START%+WINDOW" -show_entries frame=pts_time -of csv=p=0 FILE

# положение вспышки: обязательно с passthrough, иначе фильтр дублирует кадры под свои
часы
ffmpeg -v info -i FILE -map 0:v:0 -vf
"signalstats,metadata=print:key=lavfi.signalstats.YAVG" -fps_mode passthrough -f null
-

# звук по сэмплам, а не детектором: точность 20 мкс
ffmpeg -v error -i FILE -map 0:a:0 -ac 1 -ar 48000 -f s16le -

# побитовая проверка копирования
ffmpeg -v error -i OUT -map 0:v:0 -frames:v 100 -f framemd5 -
```

Пробы читать **с разделёнными потоками**: Sony MXF встречает каждое чтение строкой «could not resolve file descriptor strong ref», и если `stderr` подмешан в `stdout`, разбор JSON падает на файле, который читается прекрасно.

4.1. Дрейф: его нет, и вот чем это показано

Синхрон в разделе 4 меряли на клипах. Вопрос был поставлен иначе: **уезжает ли звук от картинки по мере длины файла** — то, что в монтаже всплывает через полчаса и портит всё.

Час материала (120 копий тридцатисекундного дубля, склеенных без перекодирования, 696 МБ), и из него вырезано по 10 секунд в трёх местах:

Откуда режем	Картинка	Звук	Разница
10-я секунда	10,000000	9,990000	−10 мс

Откуда режем	Картинка	Звук	Разница
30-я минута	10,000000	9,984000	−16 мс
59-я минута	10,000000	10,000000	**0 мс**

Величина не растёт. В конце часа она ровно ноль, в середине — шестнадцать миллисекунд; это не накопление, а то, где пришлась граница аудиопакета (21 мс на этом файле). Дрейф выглядел бы иначе: 10 мс в начале, 300 в конце.

Начало обоих потоков во всех трёх кусках — **0,000000**: рез не сдвигает звук относительно картинки, он их только вместе укорачивает.

Так и должно быть, и причина простая: потоки **копируются**. Пакеты переносятся со своими исходными метками времени, часы остаются часами источника, и умножать их не на что. Дрейф появляется там, где время пересчитывают — при перекодировании или растяжении, — и там он у нас измерен отдельно (раздел 9: фильтры съедают 15 мс независимо от коэффициента).

Недобор в конце куска убирается там, где он мешает: при перезаписи PCM звук режется по сэмплу, и на вещательном MXF картинка и звук вышли одинаковыми до микросекунды — 3,003000 и 3,003000 (раздел 12.1).

Ловушка этого измерения

Первый замер сравнивал последний **пакет** звука с последним пакетом картинки и дал бессмыслицу: «звук на 40 мс длиннее». У видео с B-кадрами порядок хранения не совпадает с порядком показа — последние три пакета шли 9,84 / 9,96 / 9,92, — и «последний пакет» оказался не последним кадром. Сравнивать надо длительности потоков, а не хвосты пакетов.

12.1. Вещательный MXF: 4K, десять потоков, служебные данные

Два клипа с реальной съёмки, `A004C012_201031NW.mxf` и `A006C010_201031JD.mxf` (98 и 281 МБ на 3,4 и 6,2 секунды — 243 и 378 Мбит/с). Материал плотнее всего, что мерили до сих пор:

Что	Значение
Видео	H.264 **all-intra**, 4096×2160, `yuv422p10le`, 23,976
Звук	**восемь** отдельных моно-дорожек PCM 24 бита / 48 кГц

Что	Значение
Данные	`smpte_436m_anc` — служебные данные SDI
Таймкод	06:49:06:03 и 04:13:14:11 — настоящие съёмочные

Каждый кадр — ключевой. Прилипания начала здесь не бывает вовсе: запрос на 00:00:01,000 лёг на кадр 00:00:01,001, то есть на ближайший кадр, а не на далёкий опорный.

Рез `MXF → MXF`, оба файла:

Файл	Запрос	Кадров	Расхождение	Таймкод
A006C010	1,000 → 4,000	72	**+0,000 с**	06:49:06:03 → 06:49:07:03
A004C012	0,500 → 2,500	48	**+0,000 с**	04:13:14:11 → 04:13:14:23

В результате **все десять потоков**: картинка, восемь моно-дорожек и служебные данные. Хэш первого кадра результата совпал с хэшем кадра источника в точке реза — `1105dab6b3f5c81761e95784e58a1bd8` в обоих. Звук переписан по сэмплу, как для MXF и положено.

Что здесь выяснилось про поток данных

Правило «какие контейнеры держат поток данных» было написано по предположению и оказалось неверным дважды:

1. **MXF держит.** Считалось, что нет. Вещательный оригинал несёт дорожку SMPTE 436M и принимает её обратно без единого возражения.
2. **Контейнера мало — важно происхождение.** MOV держит потоки данных вообще, но эту дорожку не принимает: рез `MXF → MOV` **упал** с кодом 4294967274 и без внятного сообщения. Поток данных пишется под тот формат, в котором живёт, и переезжает только в такой же.

Теперь при смене контейнера раздел говорит об этом **до** запуска и режет дальше, а не падает:

```
[WARNING] The data stream (smpte_436m_anc) was written for MXF and MOV will not
          take it, so it stays behind. Keeping the container as MXF carries it
          over.
```

Отдельная мелочь, стоившая невнятного сообщения: у MXF-потока пустой четырёхсимвольный тег, и `ffprobe` отдаёт его как `[0][0][0][0]`. В сообщении теперь стоит имя кодека —

`smpte_436m_anc`, — а тег используется там, где он осмысленный: `gpmd`, `tmcd`.

12.2. Разделение: части складываются в исходник

Старое «Разделить надвое» собиралось из двух независимых резов — «убрать конец» до точки и «убрать начало» от неё. Начало второй части прыгало к ключевому кадру, конец первой — нет, и встречались они, только если точка сама стояла на ключевом кадре.

Материал — 30 секунд H.264 640×360 с ключевым кадром каждую секунду (750 кадров, 30 ключевых), PCM 16 бит 48 кГц стерео, таймкод 01:00:00:00. Проверка самая жёсткая из возможных: хэш каждого декодированного кадра частей подряд против списка исходника и сырые сэмплы частей подряд против сэмплов исходника.

Точки	Было	Стало
10,0 с — на ключевом кадре	750 кадров	750, хэши совпали
10,2 с — после ключевого	755: **5 кадров в обеих частях**	750, хэши совпали
10,7 с — перед ключевым	743: **7 кадров ни в одной**	750, хэши совпали
10,7 и 20,3 с — три части	—	750, хэши совпали
те же точки в обратном порядке	—	750, хэши совпали

Звук во всех пяти случаях — 1 440 000 сэмплов частей подряд, **побайтово** равные исходнику.

Как устроено теперь: каждая точка один раз ложится на **ближайший** ключевой кадр, и этот кадр — общий стык: одна часть кончается перед ним, следующая с него начинается. Длина части считается через `round`, а не `ceil`: метка ключевого кадра уже стоит на границе кадра, а на 23,976 кадр 240 читается как 10,010010 с — 240,00024 кадра, и `ceil` положил бы его в обе части.

12.3. Звуковые файлы: рез по сэмплу

Материал — 30 секунд стереошума 48 кГц, 1 440 000 сэмплов, в WAV 24 бита, FLAC, MP3 320k, AAC 256k и Vorbis; точки 10,2345 и 20,0001 с — сэмплы 491 256 и 960 005. Метод один на всё: декодировать части и исходник в сырые сэмплы и сравнить побайтово.

Точная резка

Кусок декодируется от начала файла и режется `atrim` по номерам сэмплов.

Что	Результат
WAV, три части тремя проходами	491 256 + 468 749 + 479 995; подряд — исходник **бит в бит**
WAV, три части одним проходом (`asplit`)	то же, бит в бит
FLAC, три части	то же, 24 бита сохранились
середина вырезана одним проходом (`concat`), WAV и FLAC	971 251 сэмпл — исходник без куска, бит в бит
MP3, AAC, Vorbis → float WAV	кусок бит в бит равен тому же отрезку декодированного исходника

Кодирование куска в 468 749 сэмплов:

Кодер	Декодируется в	Бит в бит
MP3 320k	468 749 (+0)	нет, сжатие
Opus 256k	468 749 (+0)	нет
AAC 256k	468 992 (**+243**)	нет
FLAC, ALAC, WAV 24	468 749 (+0)	**да**

FLAC и ALAC сами держат 16 или 24 бита по исходнику; из float и 32 бит пишут 24. Opus из 44,1 и 96 кГц и MP3 из 96 кГц через soxr — длина точна до сэмпла.

Заголовок против декодера

Кусок в 468 749 сэмплов, записанный разными кодерами:

Формат	Заголовок	Декодер
WAV, FLAC, ALAC, WavPack, TTA, MP3, Vorbis	точно	точно
AAC в M4A	−29	+243
AAC в ADTS	+5 342	+1 267

Формат	Заголовок	Декодер
Opus	+312	точно
AC-3, E-AC-3	+1 267	+1 267

Поэтому раздел меряет результат декодером, а у сжатого исходника и конец файла считает декодером: AAC-файл «на 30,000 с» декодировался на 768 сэмплов длиннее, и кусок до конца обещал бы длину, которой у файла нет.

Broadcast WAV и iXML

FFmpeg читает чанк `bext` в теги под своими именами (`comment`, `encoded_by`, `date`, `creation_time`), а пишет его, только если попросить `-write_bext 1`, — из тегов с другими именами (`description`, `originator`, `origination_date`, `origination_time`). Разобрано по байтам чанка:

Как резали	Что в `bext` результата
`-map_metadata 0`	чанка нет вовсе
и `-write_bext 1`	`time_reference`, UMID, история кодирования; описание, источник, дата, время — **пустые**
и поля под именами муксера	всё, и `time_reference` сдвинут на 491 256

iXML FFmpeg не видит ни перед чанком `fmt`, ни после чанка `data` и в результат не пишет: байтов `<BWFXML>` в результате нет.

Копирование пакетов

Точка уходит к ближайшей границе пакета. Разделение на три части, кусок из середины сам по себе и вырезание середины со склейкой, в сэмплах:

Кодек	Разделение	Кусок из середины	Склейка
MP3, как делает FFmpeg	−4 625, звук сдвинут после первого стыка	−529, сдвинут	длина верна, 1 632 у стыка
MP3, заголовок только у части с начала	+47, по 1 632 у каждого стыка	длина верна, первые 1 632	—

Кодек	Разделение	Кусок из середины	Склейка
AAC	длина верна, по 1 024 у стыков	первые 1 024	+1 024, сдвинута целиком
AAC, часы склейки назад на праймер	—	—	длина верна, 1 024 у стыка
Vorbis	−2 048, сдвиг	−1 024, сдвиг	длина верна, 1 024 у стыка
Vorbis, часть с пакета раньше	—	с границы пакета, **бит в бит**	—
Opus	−624, сдвиг	−312, первые ~0,5 с другие	длина верна, 20 590 у стыка

Старт на один или четыре пакета раньше ничего не даёт MP3, AAC и Opus: искажение в начале куска то же. Склейка прямо из исходника через `inpoint` и `outpoint` демультиплексора `concat` у MP3, AAC и Opus такая же, как склейка частей, у Vorbis — сломана (+45 056 сэмплов).

Штатным путём

Весь путь — `trim_media`, как его зовёт кнопка, — прогнан на WAV с `bext` и iXML, FLAC 24 бита, MP3 с обложкой, M4A, OGG Vorbis и Opus: **34 проверки из 34**, от сэмплов частей и полей `bext` до обложки, отказа при пустой средней части, пробного прогона и отказа перезаписать.

---

13. Чего мы не проверяли

Список открытый — чтобы никто, включая автора, не решил, что тема закрыта:

- камеры: **вещательный MXF 4K all-intra с восемью дорожками и служебными данными проверен** (раздел 12.1), Canon HEVC и ARRI ProRes проверены раньше; остаются Panasonic, BMPCC, GoPro с телеметрией, AVCHD прямо с карты;
- файлы **длиннее** часа (час проверен, см. раздел 3.2);
- нехватка места на диске в середине записи;
- источник на сетевом пути или занятый другой программой;
- две операции одновременно;

- звук 96 кГц (в видео редкость; 32 бита float на 48 кГц — обычный случай с рекордеров — проверен: `pcm_f32le` переживает рез, звук куска найден в источнике дословно на 3,0000 с, смещение **+0,00 мс**);
- звук, уходящий за последний кадр на границу пакета (0,1 с после добора хвоста): об этом сообщается, но на слух не проверялось;
- звуковые файлы с настоящего рекордера: `bext` и iXML проверены на файле, собранном по спецификации, многоканальный poly-WAV — нет;
- копирование пакетов AC-3, E-AC-3, DTS и AAC вне M4A — не мерили, и поэтому этот режим для них не предлагается;
- DSD, APE и TAK в точной резке с выбором FLAC или WAV.

Устройство раздела НАРЕЗКА и его отказы — в TRIM_MATRIX_RU.md, подробно — в TRIM_RU.md, решения — в DECISIONS_RU.md.