

Анализ изменений между двумя версиями ядра Linux

Горбацевич Анна, Горбатюк Олег, Сюй София

15 февраля 2026 г.

Постановка задачи

Целью работы являлась разработка инструмента для анализа изменений кода между двумя версиями ядра Linux.

Требовалось:

- взять репозиторий ядра Linux;
- выбрать две произвольные версии (теги);
- собрать статистику изменений;
- сгруппировать изменения по директориям;
- вывести агрегированную информацию в удобном табличном виде.

Дополнительно необходимо было учесть масштаб проекта Linux Kernel (миллионы строк кода, десятки тысяч файлов).

Описание решения

Решение состоит из двух частей:

1. Скрипт `kernel-fetch.sh` для эффективной загрузки репозитория.
2. Программа `diff-analyzer` на C++ для анализа изменений.

1. Загрузка репозитория

Репозиторий ядра Linux является одним из крупнейших open-source проектов. Полный clone занимает несколько гигабайт и содержит всю историю коммитов.

Изначально рассматривался вариант использования GitHub API для получения diff удалённо. Однако был обнаружен критический недостаток:

- GitHub API возвращает diff максимум для 300 файлов;
- при сравнении версий ядра изменения затрагивают тысячи файлов;
- это делает API непригодным для анализа крупных релизов.

В результате было принято решение работать напрямую с Git.

Чтобы избежать загрузки полной истории репозитория, был реализован **shallow clone** с использованием фильтрации:

- `-filter=blob:none`
- `-filter=tree:0`
- загрузка только двух необходимых тегов

Таким образом:

- скачивается минимально необходимый набор объектов;
- не загружается вся история коммитов;
- значительно уменьшается размер репозитория на диске;
- ускоряется подготовительный этап анализа.

2. Анализ изменений

Программа `diff-analyzer.cpp` выполняет:

1. запуск команды:

```
git diff --numstat <ref1>..<ref2>
```

2. парсинг вывода построчно;

3. извлечение:

- количества добавленных строк,
- количества удалённых строк,
- пути к файлу;

4. агрегацию статистики по директориям.

Для каждой директории собираются:

- число изменённых файлов;
- добавленные строки;
- удалённые строки;
- общее количество изменений;
- процент от общего объёма изменений.

Поддерживаются параметры:

- глубина группировки по директориям;
- вывод только top-N директорий.

Сложности реализации

1. Масштаб репозитория

Ядро Linux содержит:

- десятки тысяч файлов;
- миллионы строк кода;
- изменения между версиями могут превышать сотни тысяч строк.

Например, при сравнении двух версий были получены:

- 883К строк изменений;
- 13080 файлов;

Обработка такого объёма данных требует:

- потокового чтения вывода `git diff`;
- аккуратного управления памятью;
- эффективной агрегации статистики.

2. Ограничения GitHub API

Попытка использовать GitHub API показала, что:

- существует лимит в 300 файлов на diff;
- большие релизы ядра существенно превышают этот лимит;
- данные возвращаются частично, что делает анализ некорректным.

Это потребовало перехода к работе напрямую с Git.

3. Оптимизация загрузки

Полный `clone` репозитория Linux:

- занимает значительное место на диске;
- требует длительного времени скачивания.

Использование `shallow clone` позволило:

- ограничиться только нужными тегами;
- минимизировать объём скачиваемых данных;
- ускорить работу инструмента.

Результат работы

В результате был реализован инструмент, который:

- автоматически загружает необходимые версии ядра;
- анализирует изменения между ними;
- агрегирует данные по директориям;
- выводит удобную таблицу с процентным распределением изменений;
- масштабируется на крупные diff (десятки тысяч файлов).

Полученный инструмент позволяет быстро оценить:

- какие подсистемы ядра изменились больше всего;
- распределение изменений по структуре проекта;
- относительный “вес” изменений.

Пример результата работы инструмента

Git Diff Analysis 883K lines changed across 13080 files (++565K --318K)						
Directory	Files	Added	Removed	Total	%	Distribution
drivers/	5472	286K	88.4K	375K	42.5%	
fs/	857	19.2K	128K	147K	16.7%	#####
arch/	2170	81.4K	29.1K	110K	12.5%	####
tools/	1034	53.4K	15.4K	68.9K	7.8%	##
Documentation/	1113	30.8K	9597	40.4K	4.6%	#
sound/	567	23.2K	15.9K	39.2K	4.4%	#
include/	677	18.2K	4679	22.9K	2.6%	#
net/	395	10.0K	7964	17.9K	2.0%	#
kernel/	193	8883	4613	13.4K	1.5%	
mm/	101	8378	4883	13.2K	1.5%	
rust/	122	10.8K	839	11.7K	1.3%	
lib/	142	5290	3116	8406	1.0%	
scripts/	74	3866	1742	5608	0.6%	
crypto/	23	306	1669	1975	0.2%	
block/	35	768	622	1390	0.2%	
io_uring/	29	886	471	1357	0.2%	
(root)	8	762	211	973	0.1%	
security/	22	475	493	968	0.1%	
samples/	18	485	55	540	0.1%	
virt/	7	280	130	410	0.0%	
usr/	4	172	86	258	0.0%	
init/	7	122	17	139	0.0%	
{drivers/	2	2	45	47	0.0%	
ipc/	4	15	16	31	0.0%	
{arch/	4	0	0	0	0.0%	

На изображении приведён пример вывода разработанного инструмента при сравнении двух версий ядра Linux. В рассматриваемом примере:

- всего изменено 883 000 строк;
- изменения затронули 13 080 файлов;
- добавлено 565 000 строк;
- удалено 318 000 строк.

Таблица демонстрирует распределение изменений по директориям верхнего уровня. Наибольший вклад в объём изменений внесла директория `drivers/` — 42.5% от общего количества строк. Это ожидаемо, поскольку подсистема драйверов является одной из самых крупных частей ядра.

Также значительные изменения наблюдаются в:

- `fs/` — файловые системы;
- `arch/` — архитектурно-зависимый код;
- `tools/` — вспомогательные утилиты.

Гистограмма распределения (столбец *Distribution*) визуально отражает относительный «вес» изменений каждой директории. Это позволяет быстро определить, какие подсистемы подверглись наиболее существенной модификации в рамках выбранного релиза.

Таким образом, инструмент обеспечивает:

- масштабируемость при анализе крупных diff;
- удобную агрегацию данных;
- наглядное представление результатов.

Заключение

Задача анализа изменений между версиями ядра Linux оказалась нетривиальной из-за:

- огромного объёма репозитория;
- ограничений внешних API;
- необходимости оптимизации загрузки данных.

Реализованное решение сочетает:

- эффективную загрузку данных (shallow clone),
- потоковый анализ diff,
- удобное представление агрегированной статистики.

Инструмент может быть использован для анализа любых крупных Git-репозиториев.