Анализ проекта на языке Си

Существует ряд инструментов для анализа исходного кода на Си : ncc, Coverity prevent, pmccabe, klocwork, Rhapsody.

И вспомогательные пакеты : dia, autodia, zgrviewer, codeviz и др.

Информационное обеспечение ИО

Важная часть поддержки проекта это информационное программное обеспечение. Такое как база знаний.

На сайте

http://106.109.9.71/wiki/index.php/Main_Page

в группе SWL-DTV собраны рецепты использования различных инструментах. Использован движёк МедиаВики.

В эту базу знаний была добавлена необходимая информация.

http://106.109.9.71/wiki/index.php/Ncc

Желательно дополнить эту информацию хранилищем архивов типа FTP или каталог на сервере ПО - \\aardvark

Статический анализ кода на Си

- Анализ кода
- И создание моделей

Можно делать на первом этапе.

- На следующем этапе мы помещаем всю информацию в Базу Данных.

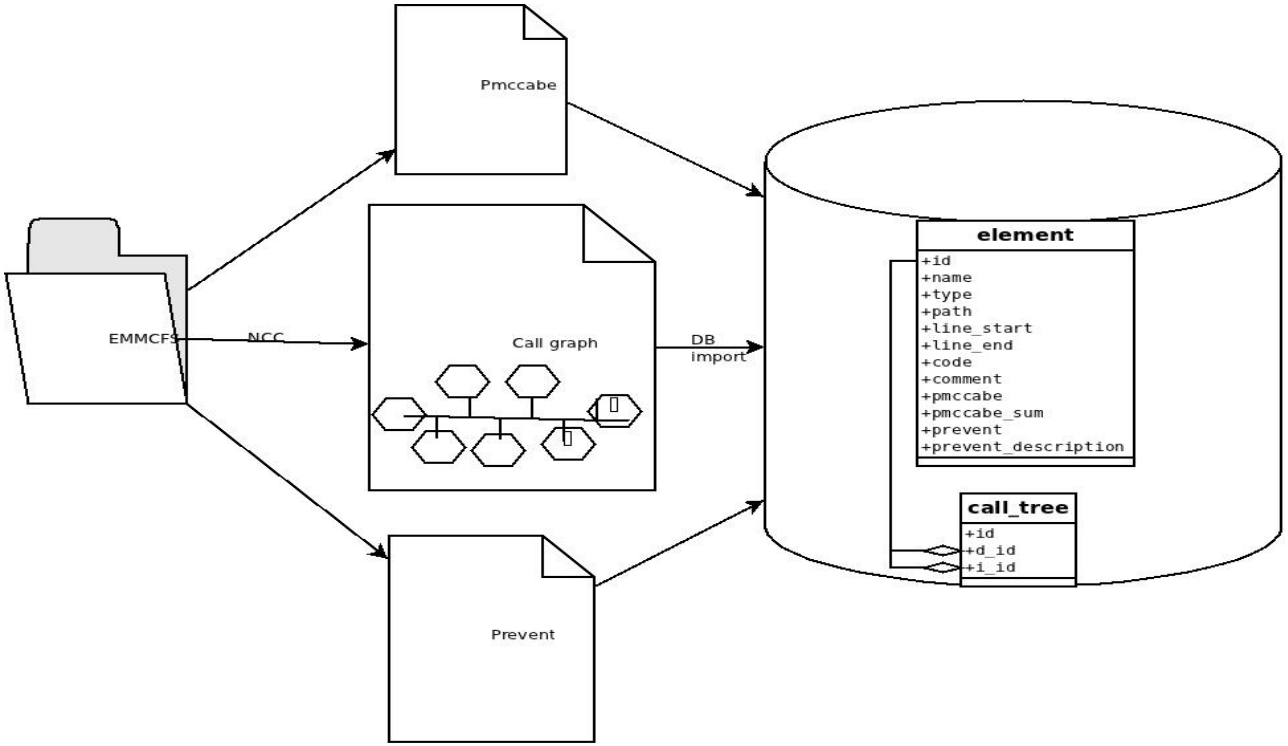
БД позволяет делать гибкие запросы и собрать статистику по коду. Например — количество функций, вызовов из функций, сегментацию кода и т.д.

Статический анализ элементов ядра Linux

- Чтобы использовать Coverity prevent для анализа кода части ядра Линукс необходимо:
- Настроить минимальную конфигурацию ядра Linux без модулей.
- Опционально : убрать флаги оптимизации и включить отладочную информацию в код ядра.
- Предварительно собрать ядро.
- Удалить объектные файлы в местах анализа кода.
- Отредактировать файл настройки Coverity prevent. В частности отменить команду make clean.

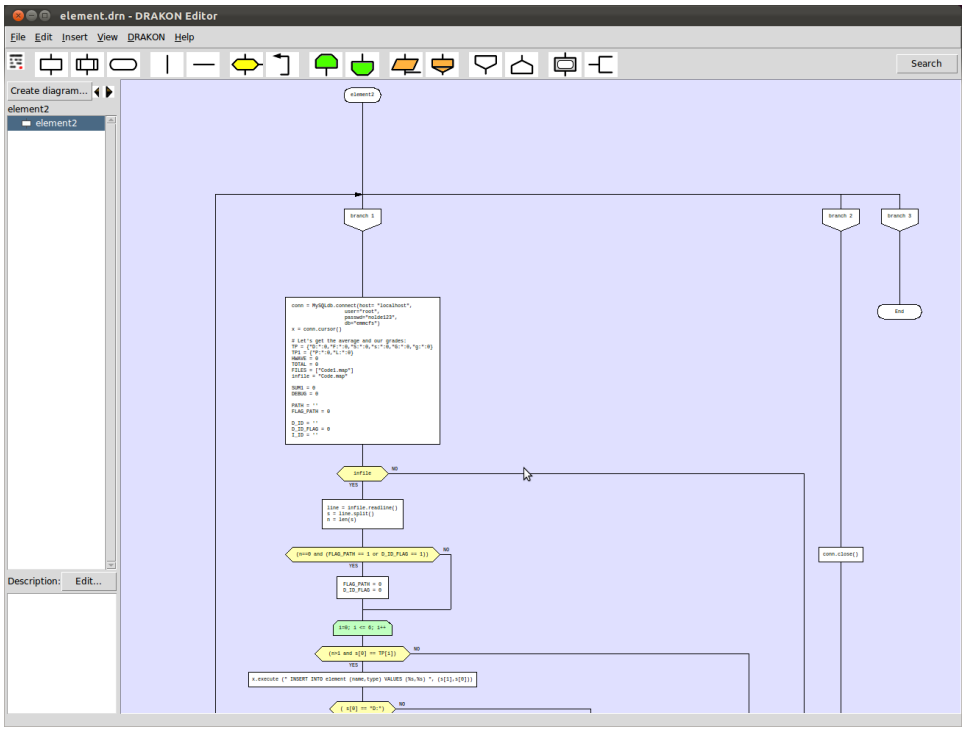
Заполнение БД.

- Всё заполнение БД делается регулярными скриптами на языке Python (который является стандартом в отрасли S\W)
- База данных конвертируется в модель и обратно автоматически.



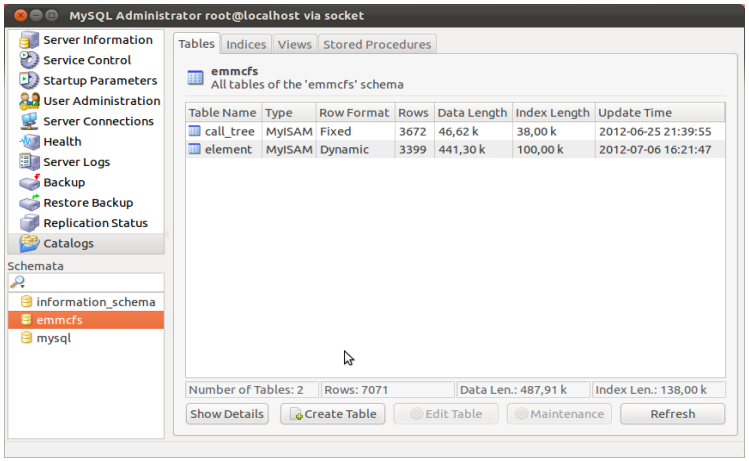
Модели для скриптов БД

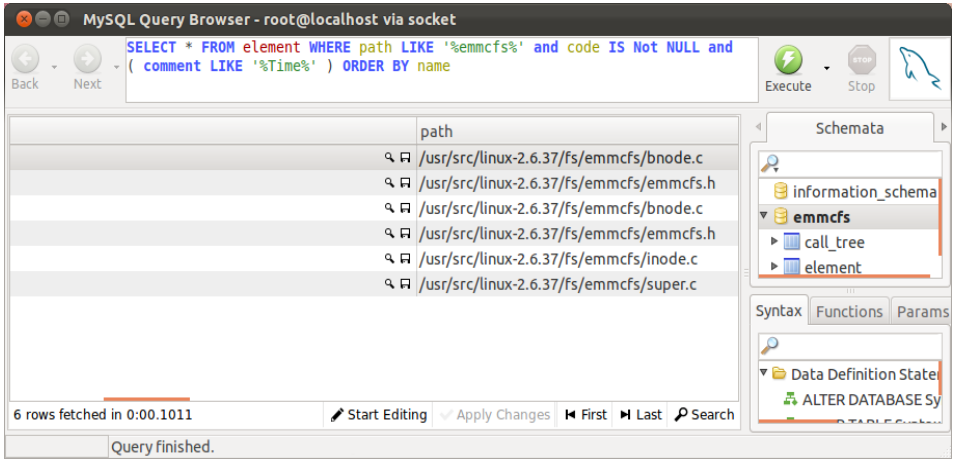
Был использован мета-графический язык Drakon для создания алгоритма скрипта и последующей генерации кода на Python или нескольких других языков C, C++, Java, Tcl.



Анализ БД

- Используются прикладные программы: MySQL Admin, Query Browser





Для фильтрации данных используются SQL запросы:

SELECT * FROM element WHERE path LIKE '%emmcfs%' and code IS Not NULL and (comment LIKE '%Time%') ORDER BY name

Анализ архитектуры проекта

- Мною разработан алгоритм, выделения типа функций.

Я выделяю 7 уровней:

Classification functions by the following features:

- 1. The **object**.
- 2. Upgrade to the states. Finite State Machine (**FSM**).
- 3. To implement the **switching** functions / equalization / branching.
- 4. To implement the function **parser** (parsing) /

Два дополнительных уровня:

- 8. Along the route objectives. Target way.
- 9. By the user and planned processes. The sensitivity analysis.

Благодаря разбиению всех функций на уровни можно проанализировать не достаточное разбиение функций на под- функции. **И вплотную подойти к автоматической генерации тестов для всех функций.**

Результаты анализа

В результате анализа каждой функции в отдельности в контексте Call graph удалось выделить 12 наиболее усложнённых функций.

Именно в этих функциях были впоследствии найдены ошибки программой Coverity prevent и высокий индекс cyclomatic complexity by pmccabe.

Дальнейшая работа

Предлагаю помещать в БД результаты динамического анализа кода, фиксировать ошибки по утечке памяти (duma-dml).

Так же сохранять в БД модели отдельных функций и целых модулей.

Создать в перспективе автоматизированный интегрирующий инструмент, который автоматизирует работу с общепринятыми программами для тестирования и