Содержание

1 О проекте.

1.1 О модуле.

1.2 О GUI.

1.3 О взаимодействии.

2 Интерфейс модуля.

3 GUI.

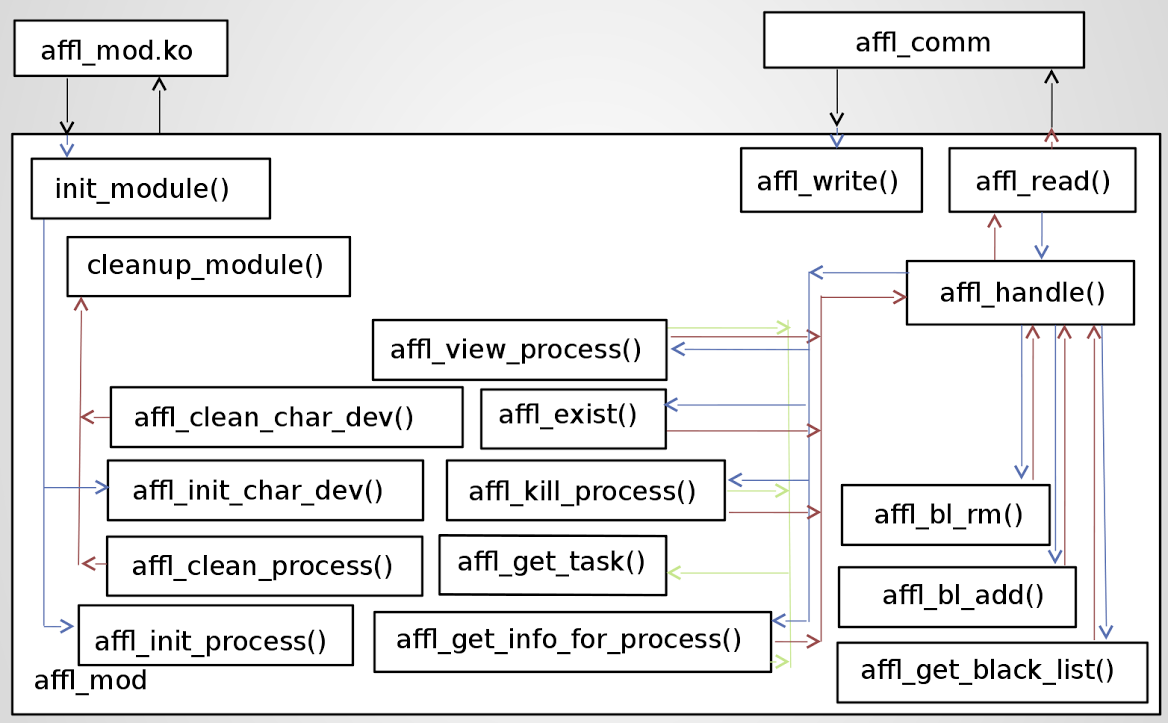
4.Баги.

5 Тесты.

**1 О проекте.**

**1.1 О модуле.**

Структура драйвера



Драйвер состоит из:

* 4-х логических частей:
* Символьное устройство;
* Работа с процессами;
* Обработчик команд;
* Инициализация самого драйвера;
* 6-и файлов:
* affl\_char\_dev.h;
* affl\_char\_dev.c;
* affl\_process.h;
* affl\_procecc.c;
* affl\_main.h;
* afffl\_main.c;
* 26 функций
* **int** **affl\_open**(**struct** inode \*i, **struct** file \*f);
* **int** **affl\_close**(**struct** inode \*i, **struct** file \*f);
* ssize\_t **affl\_read**(**struct** file \*f, **char** \_\_user \*buf, size\_t len, loff\_t \*off);
* ssize\_t **affl\_write**(**struct** file \*f, **const** **char** \_\_user \*buf, size\_t len, loff\_t \*off);
* **int** **affl\_init\_char\_dev**(**const** **char**\* file\_name, **const** **char**\* device\_name);
* **void** **affl\_clean\_char\_dev**(**void**);
* **int** **affl\_init\_process**(**void**);
* **void** **affl\_clean\_process**(**void**);
* **unsigned** **int** **affl\_handle**(**const** **char**\* input, **char**\* user\_buf);
* **void**\* **find\_sym**(**const** **char** \*sym);
* **int** **affl\_get\_proc\_name**(**const** **char**\* input, **char**\*\* proc\_name);
* **int** **affl\_get\_proc\_PID**(**const** **char**\* input, **int**\* PID);
* **int** **affl\_get\_task**(**void**);
* **int** **affl\_get\_quantity\_tasks**(**void**);
* **int** **affl\_get\_black\_list**(**char**\* user\_buf);
* **int** **affl\_get\_info\_for\_process**(**int** pid, **char**\* user\_buf);
* **unsigned** **int** **affl\_view\_process**(**char**\* user\_buf);
* **int** **affl\_kill\_process**(**const** **char**\* name, **int** PID);
* **int** **affl\_from\_name\_to\_pid**(**char**\* name);
* **int** **affl\_exist**(**char**\* user\_buf);
* **void** **affl\_bl\_print**(**void**);
* **int** **affl\_bl\_add**(**char**\* arg);
* **int** **affl\_bl\_rm**(**char**\* arg);
* **int** **affl\_bl\_cmp**(**const** **char**\* arg);
* **void** **affl\_add\_list\_process\_mass**(**const** **char**\* proc\_name, **int** PID);

Описание функций

**int** **affl\_open**(**struct** inode \*i, **struct** file \*f):

Определение лежит в файле affl\_char\_dev.c, объявление лежит в файле affl\_char\_dev.h.

Параметры i и f передаются системой.

Это перегрузка системной функции для файла символьного устройства, при открытии файла устройства будет вызываться эта функция и инкрементировать счетчик открытий.

**int** **affl\_close**(**struct** inode \*i, **struct** file \*f);

Определение лежит в файле affl\_char\_dev.c, объявление лежит в файле affl\_char\_dev.h.

Параметры i и f передаются системой.

Это перегрузка системной функции для файла символьного устройства, при закрытии файла устройства будет вызываться эта функция и декркментировать счетчик открытий.

ssize\_t **affl\_read**(**struct** file \*f, **char** \_\_user \*buf, size\_t len, loff\_t \*off).

Определение лежит в файле affl\_char\_dev.c, объявление лежит в файле affl\_char\_dev.h.

Все входные параметры передаются системой, возвращает количество считанных символов.

Это перегрузка системной функции для файла символьного устройства, копирует содержимое buf в глобальную переменную ядра **char** affl\_kernel\_buf[255] и передает указатель на копию buf, функции affl\_handle(**const** **char**\* input, **char**\* user\_buf), также в эту функцию передается сам buf для возможности записи в файл из affl\_handle(**const** **char**\* input, **char**\* user\_buf).

ssize\_t **affl\_write**(**struct** file \*f, **const** **char** \_\_user \*buf, size\_t len, loff\_t \*off)

Определение лежит в файле affl\_char\_dev.c, объявление лежит в файле affl\_char\_dev.h.

Все входные параметры передаются системой, возвращает количество считанных символов.

Это перегрузка системной функции для файла символьного устройства, копирует содержимое переменной buf пользовательского пространства в переменную ядерного пространства, также копирует количество записанных символов в глобальную переменную **int** affl\_size.

**int** **affl\_init\_char\_dev**(**const** **char**\* file\_name, **const** **char**\* device\_name)

Определение лежит в файле affl\_char\_dev.c, объявление лежит в файле affl\_char\_dev.h.

В параметр file\_name передается имя файла смивольного устройства, файл будет находиться в директории /dev/, в параметр device\_name передаеться имя регестрируемого символьного устройства, при успешном завершении функция возвращает 0, при не удачном -1.

**void** **affl\_clean\_char\_dev**(**void**)

Определение лежит в файле affl\_char\_dev.c, объявление лежит в файле affl\_char\_dev.h.

Функция разрушает символьное устройство и удаляет файл устройства.

**int** **affl\_init\_process**(**void**)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция инициализирует указатели на функции системных вызовов, адресами системных вызовов, также, функция производит подмену адреса системного вызова sys\_execve в таблице системных вызовов.

**void** **affl\_clean\_process**(**void**)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция записывает в таблицу системных вызовов старый адрес sys\_execve.

**unsigned** **int** **affl\_handle**(**const** **char**\* input, **char**\* user\_buf)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция обрабатывает команды, поступившие в файл устройства. Вызов этой функции производится из функции **affl\_read**. В input передается содержимое файла символьного устройства, а в user\_buf передается адрес буфера из пользовательского пространства, для возможности записи в файл из пространства ядра. Возвращаемое значение возвращает количество передаваемых символов в user\_buf.

**void**\* **find\_sym**(**const** **char** \*sym)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает название системного вызова и в случае усеха — возвращает адрес системного вызова, иначе -озващает NULL.

**int** **affl\_get\_proc\_name**(**const** **char**\* input, **char**\*\* proc\_name)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает содержимое файла символьного устройства и адрес строки proc\_name. После отработки функция пишет в proc\_name имя процесса(функция не проверяет существования такого процесса), в случае успеха возвращает 0, в противном случае -1.

**int** **affl\_get\_proc\_PID**(**const** **char**\* input, **int**\* PID)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает содержимое файла символьного устройства и адрес переменной PID. В случае, если полученный pid из строки input существует, функция записывает полученный pid в переменную PID и функция возвращает 0, если полученный pid равен -1, в переменную PID записываеться -1 и функция возвращает 0, в противном случае в PID записываеться 0 и функция возвращает -1.

**int** **affl\_get\_task**(**void**)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция заполняет глобальную переменную массива структур, которая содержит имя процесса и pid процесса.

**int** **affl\_get\_quantity\_tasks**(**void**)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция возвращает количество процессов в системе.

**int** **affl\_get\_black\_list**(**char**\* user\_buf)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает указатель на буффер пользовательского пространства и возвращает количество записанных символов в user\_buf. Функция записывает в содержимое черного списа.

**int** **affl\_get\_info\_for\_process**(**int** pid, **char**\* user\_buf)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает pid, и указатель на буффер пользовательского пространства, возвращает количество записанных символов в user\_buf. Если pid существует функция пишет в user\_buf имя процесса, pid, полную командную строку, количество используемых ресурсов и сами ресурсы (файлы, pipes, socets). Если pid не существует, функция пишет в user\_buf «-1»;

**unsigned** **int** **affl\_view\_process**(**char**\* user\_buf)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция пишет в пользовательский буффер pid, имя всех процессов, в первой строке указываеться количество процессов. Возвращает количество записанных символов в user\_buf.

**int** **affl\_kill\_process**(**const** **char**\* name, **int** PID)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает имя процесса и PID, причем PID может быть как валидным так и иметь значение «-1». В первом случае убиваеться процесс имеющий такой PID и не имеет значения пфраметр name, во втором случае убиваються все процессы имеющие имя name. В успешном случае функция возвращает 0, в противном случае -1;

**int** **affl\_from\_name\_to\_pid**(**char**\* name)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает строку и если есть процесс с таким именем — функция возвращает валидный pid, иначе возвращает 0.

**int** **affl\_exist**(**char**\* user\_buf)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Если файл существует и привязан к символьному устройству, то при вызове этой функции туда будет записан «0». Возвращает количество записанных символов в буффер пользовательского пространства.

**void** **affl\_bl\_print**(**void**)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

При изменении черного списка функция печатает в лог ядра черный список.

**int** **affl\_bl\_add**(**char**\* arg)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает строку и добавляет её в черный список. В успешном случае возвращает «0», если список переполнен, функция возвращает «-2», если такая строка уже существует черном списке, функция верне «-1».

**int** **affl\_bl\_rm**(**char**\* arg)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает строку и если есть такая строка в черном списке — строка удаляеться из черного списка и функция возвращает «0», в противном случае функция возвращает «-1».

**int** **affl\_bl\_cmp**(**const** **char**\* arg)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция принимает строку и сравнивает её с черным списком. В успешном случае возвращает «0» , в противном «-1».

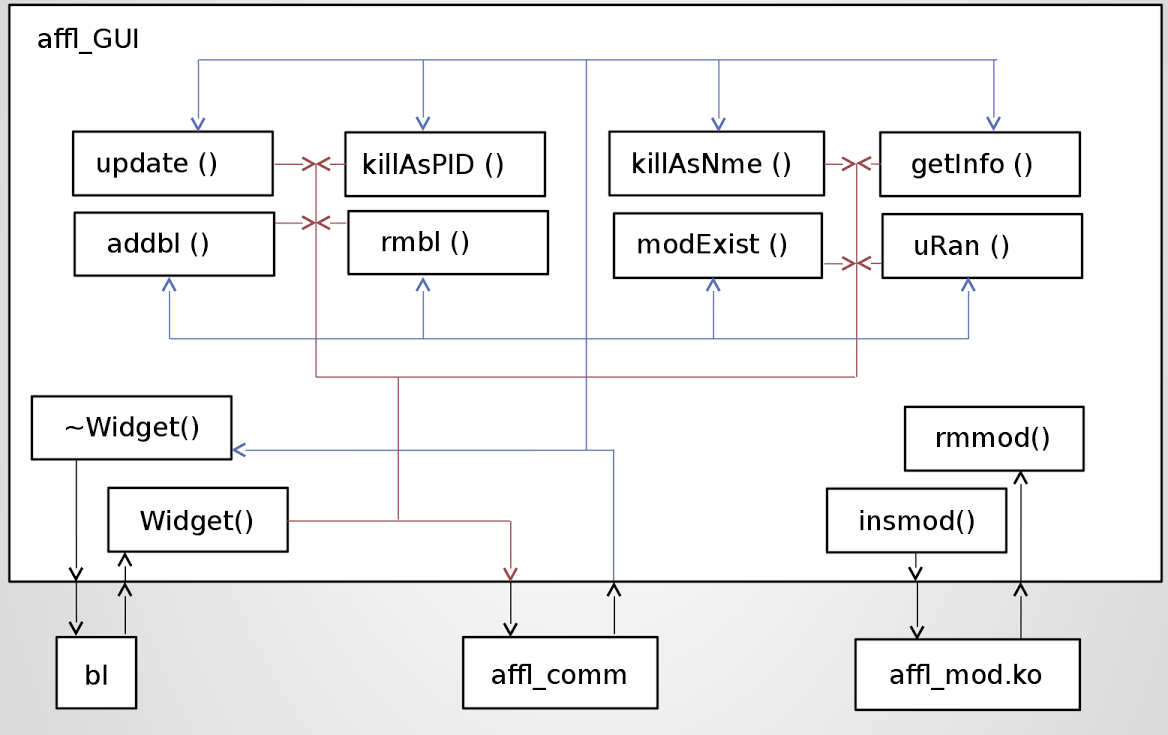
**void** **affl\_add\_list\_process\_mass**(**const** **char**\* proc\_name, **int** PID)

Определение лежит в файле affl\_process.c, объявление лежит в файле affl\_process.h.

Функция добавляет в массив структур имя процесса и PID.

**1.2 О GUI**

Структура GUI



Форма имеет:

4 файла:

* main.cpp
* widget.cpp
* widget.h
* widget.ui

12 функций:

* **void** **resiveMesage**(**QString** arg);
* **void** **resiveStatMod**(**QString** arg);
* **void** **insmod**();
* **void** **rmmod**();
* **void** **update**();
* **void** **killAsPID**();
* **void** **killAsName**();
* **void** **getInfo**(**int** row, **int** collom);
* **void** **addbl**();
* **void** **rmbl**();
* **void** **modExist**();
* **void** **uRun**();

Описание функций

**void** **sendMesage**(**QString**)

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция принимает строку и печатает её в в поле состояния драйвера: загружен он или не загружен.

**void** **sendStatMod**(**QString**)

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция принимает строку и печатает её в в поле состояния (операция вполнена успешно или нет).

**void** **insmod**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция загружает драйвер.

**void** **rmmod**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция выгружает драйвер.

**void** **update**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция обновляет список процессов.

**void** **killAsPID**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция убивает проуесс по его PID.

**void** **killAsName**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция убивает проуесс по его имени.

**void** **getInfo**(**int** row, **int** collom)

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция выводит информацию о выбранном процессе.

**void** **addbl**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция добавляет имя процесса в черный список

**void** **rmbl**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция удаляет строку из черного списка.

**void** **modExist**()

Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция проверяет, подключен ли файл к символьному устройству.

**void** **uRun**()

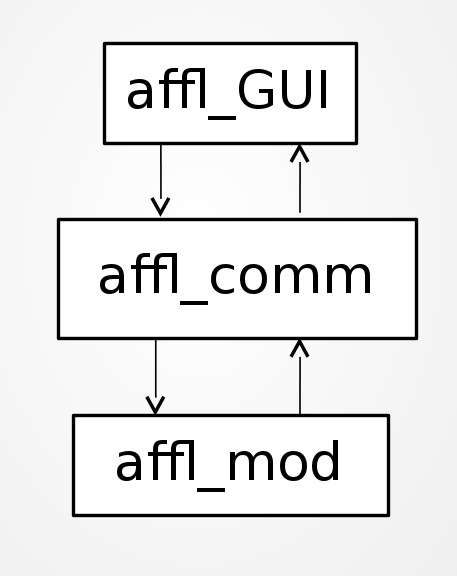
Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.

Функция добавляет строку в черный список.

**1.3 О взаимодействии**

Пользовательская программа пишет команды в файл affl\_comm, после чего драйвер affl\_mod считывает команду и пишет ответ в файл.

Это изображено на рисунке ниже:



**2 Интерфейс модуля:**

Просмотреть список процессов:

sudo sh -c "echo "view@" > affl\_comm" && sudo cat affl\_comm

Убить процесс по PID:

sudo sh -c "echo "kill@%PID%" > affl\_comm" && sudo cat affl\_comm

Убить все процессы с таким именем:

sudo sh -c "echo "kill@programm\_name#%-1%" > affl\_comm" && sudo cat affl\_comm

Добавить имя процесса в черный список:

sudo sh -c "echo "addProc@programm\_name#" > affl\_comm" && sudo cat affl\_comm

Убрать имя процесса из черного списка:

sudo sh -c "echo "rmProc@programm\_name#" > affl\_comm" && sudo cat affl\_comm

Показать информацию о процессе:

sudo sh -c "echo "getInfo@%PID%" > affl\_comm" && sudo cat affl\_comm

Проверка на подключение файла к символьному устройству:

sudo sh -c "echo "exist@" > affl\_comm" && sudo cat

**3. GUI**

Содержит:

* Одна форма.

Содержит 6 кнопок:

* Кнопка «Update» обновляет список процессов.
* Кнопка «killAsName» убивает все процессы с выбранным именем.
* Кнопка «killAsPid» цбивает проуес по его PID.
* Кнопка «=>» добавляет процесс в черный список.
* Кнопка «<=» убирает процесс из черного списка.
* Кнопка «UnRun» добавляет в черный список вписанное имя, которое вписываеться в текстовое поле выше.

Левая таблица содержит список всех процессов.

Правая таблица содержит черный список.

Нижнее текстовое поле выводит информацию о выбранном процессе.

**4.Баги**

* Не работает функция добавления в черный список на х64 архитектуре, из-за того, что на этих системах вместо системного вызова sys\_execve применяеться stub\_execve.
* Если при загрузке драйвера, в /dev содержится файл с таким же именем, то драйвер не будет читать и писать в этот файл.

**5.Тесты**

