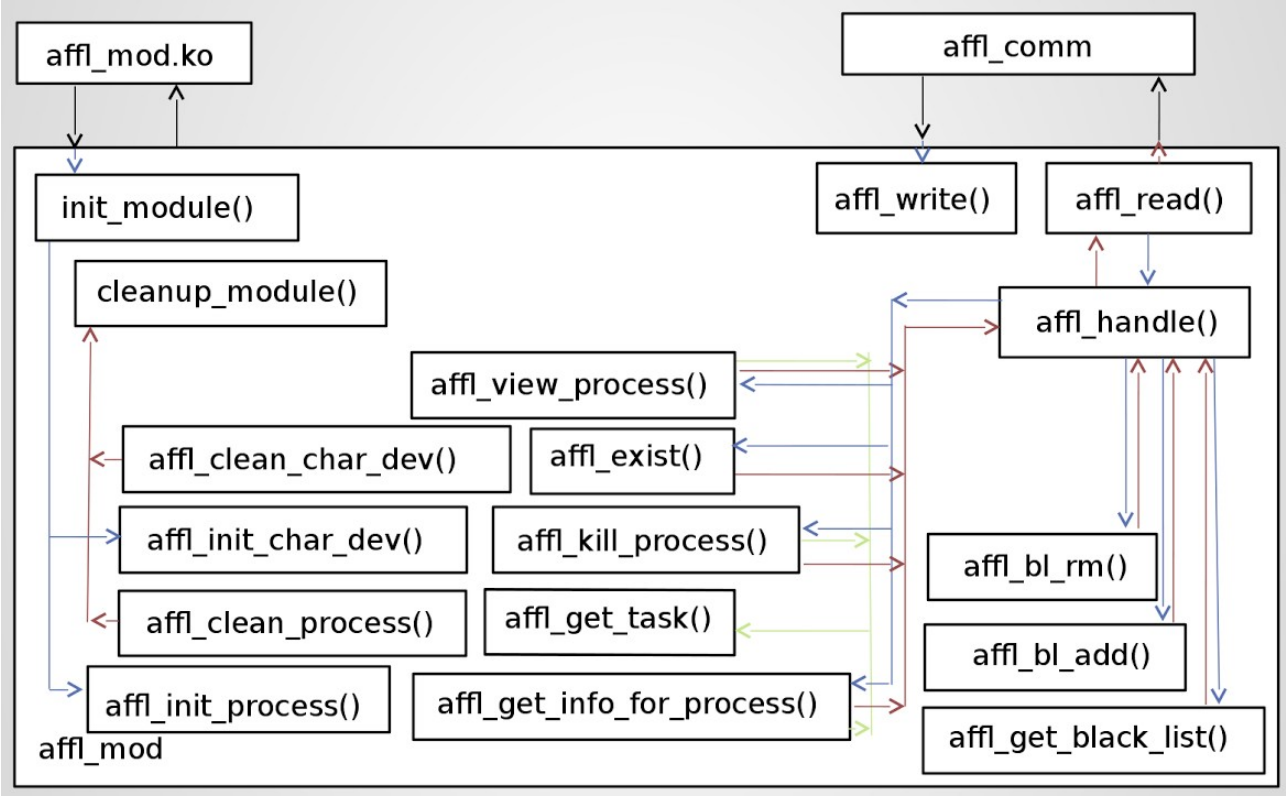
Содержание

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 списка.

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 убивается процесс имеющий такой PID и не имеет значения пфраметр name, во втором случае убиваются 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)

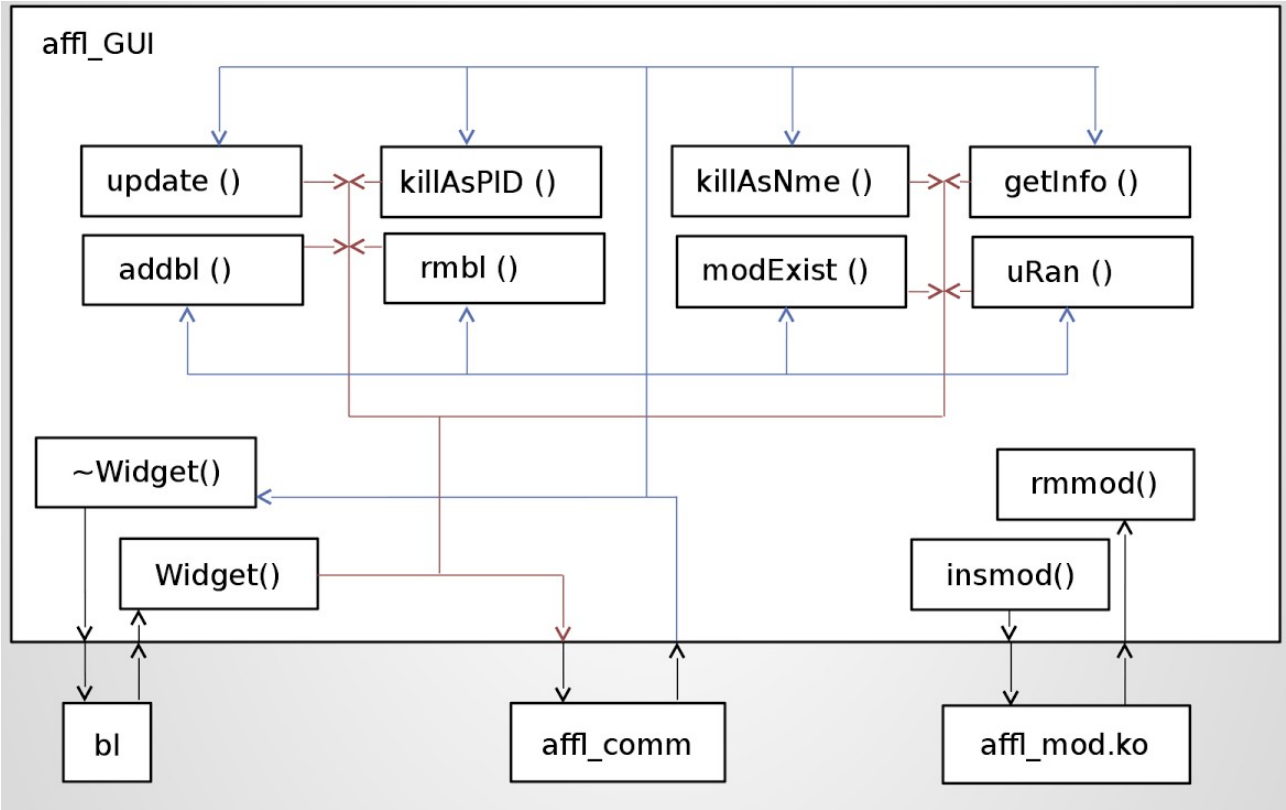
Определение лежит в файле affl_process.c, объявление лежит в файле affl_process.h.
Функция принимает строку и сравнивает её с черным списком. В успешном случае возвращает «0» , в противном «-1».

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

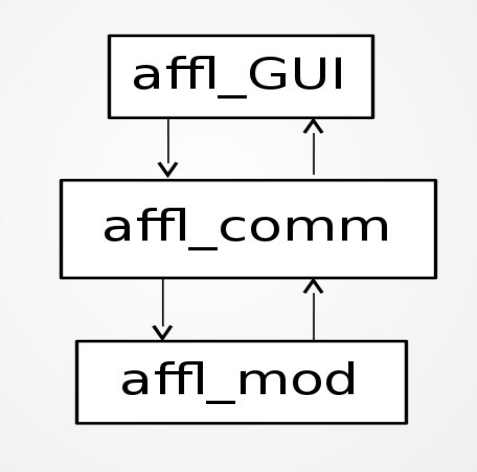
void rmbl()
Определение лежит в файле widget.cpp, объявление лежит в файле widget.h.
Функция удаляет строку из черного списка.

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 x64 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

Тесты	Kubuntu 3.x x86	Kubuntu 3.x x64
Существование affl_comm в /dev	-	-
Большое количество процессов	+	+
Функция kill as PID	+	+
Функция kill as Name	+	+
Функция View	+	+
Функция add to black list	+	-
Функция rm process from black list	+	-
Функция пасивной защиты	+	-
Функция getInfo	+	+