2013, Калуга

LinuxTesting.org

Генерация модели окружения для статической верификации драйверов, состоящих из нескольких модулей ядра Linux

Илья Захаров
ilja.zakharov@ispras.ru

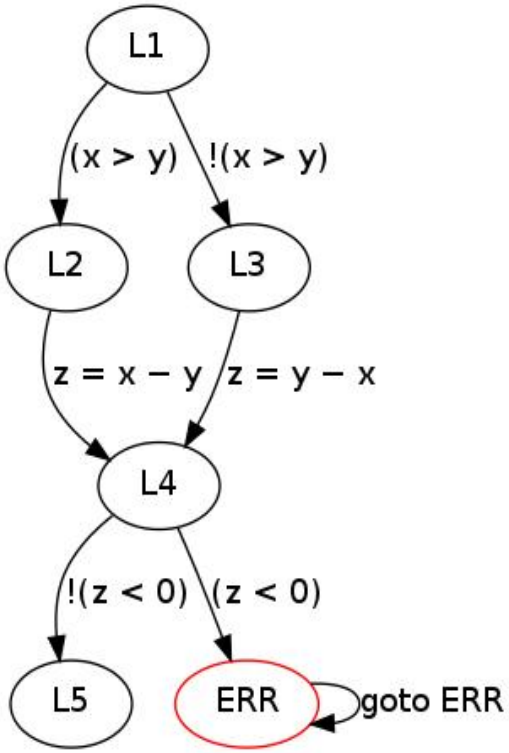
http://linuxtesting.org/project/ldv,
gsoc2013/glaurung/5001

ISP RAS
Institute for System Programming of the Russian Academy of Sciences

Статическая верификация

Анализ исходного кода программы без ее реального выполнения

```
        void main()
        {
          int x;
          int y;
          int z;
L1:       if (x > y) {
L2:         z = x - y;
          } else {
L3:         z = y - x;
          }
L4:       if (z < 0)
ERR:        goto ERR;
L5:     }
```



Инструменты статической верификации Си программ

CPAchecker

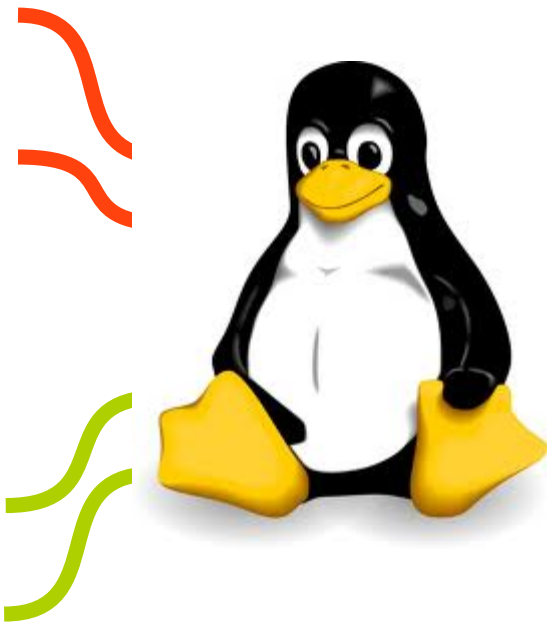
UFO

CBMC

ИСП РАН BLAST

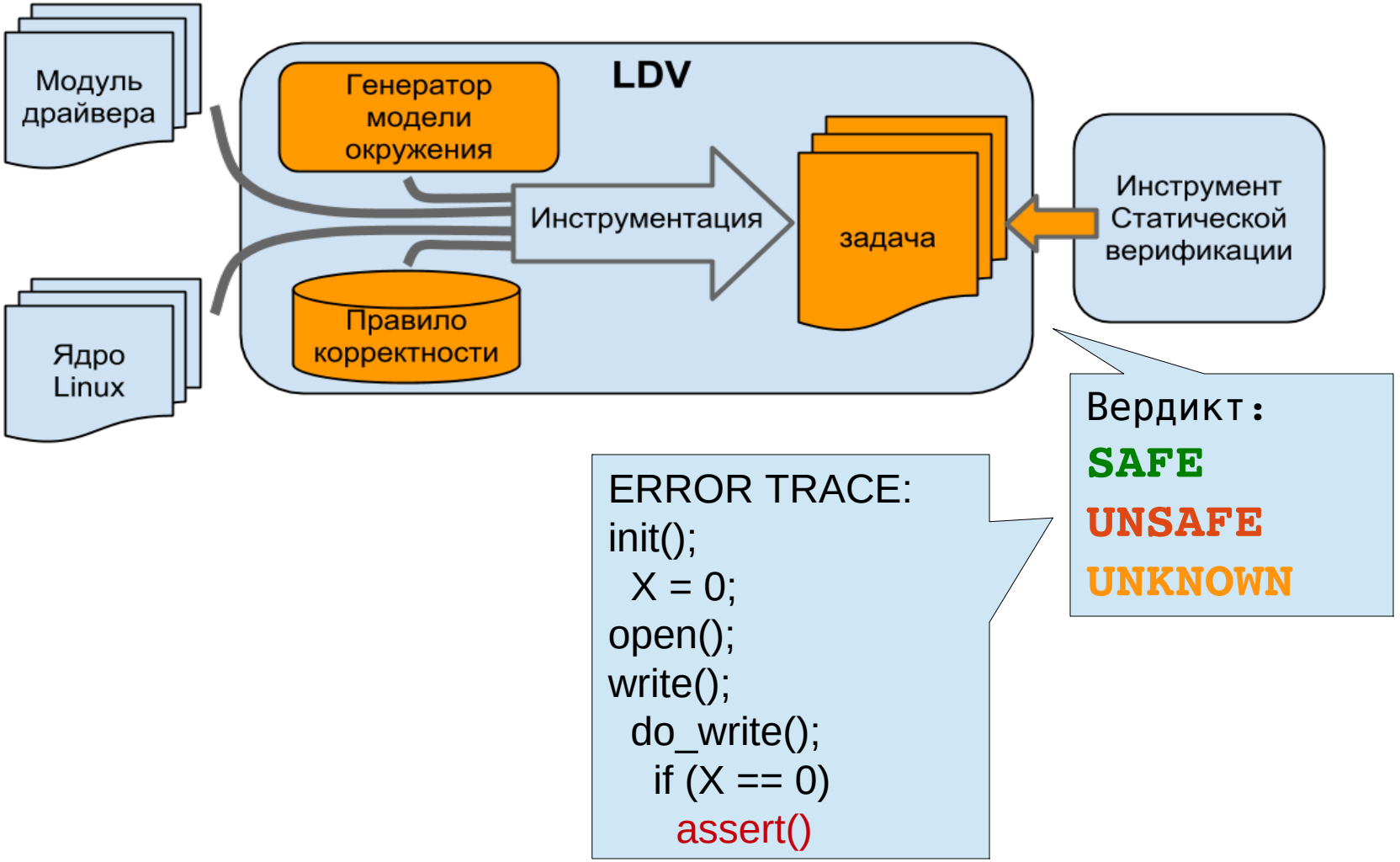
Статический анализ ядра Linux

- Высокая сложность и объем исходного кода ядра
- Высокая степень взаимозависимости компонентов

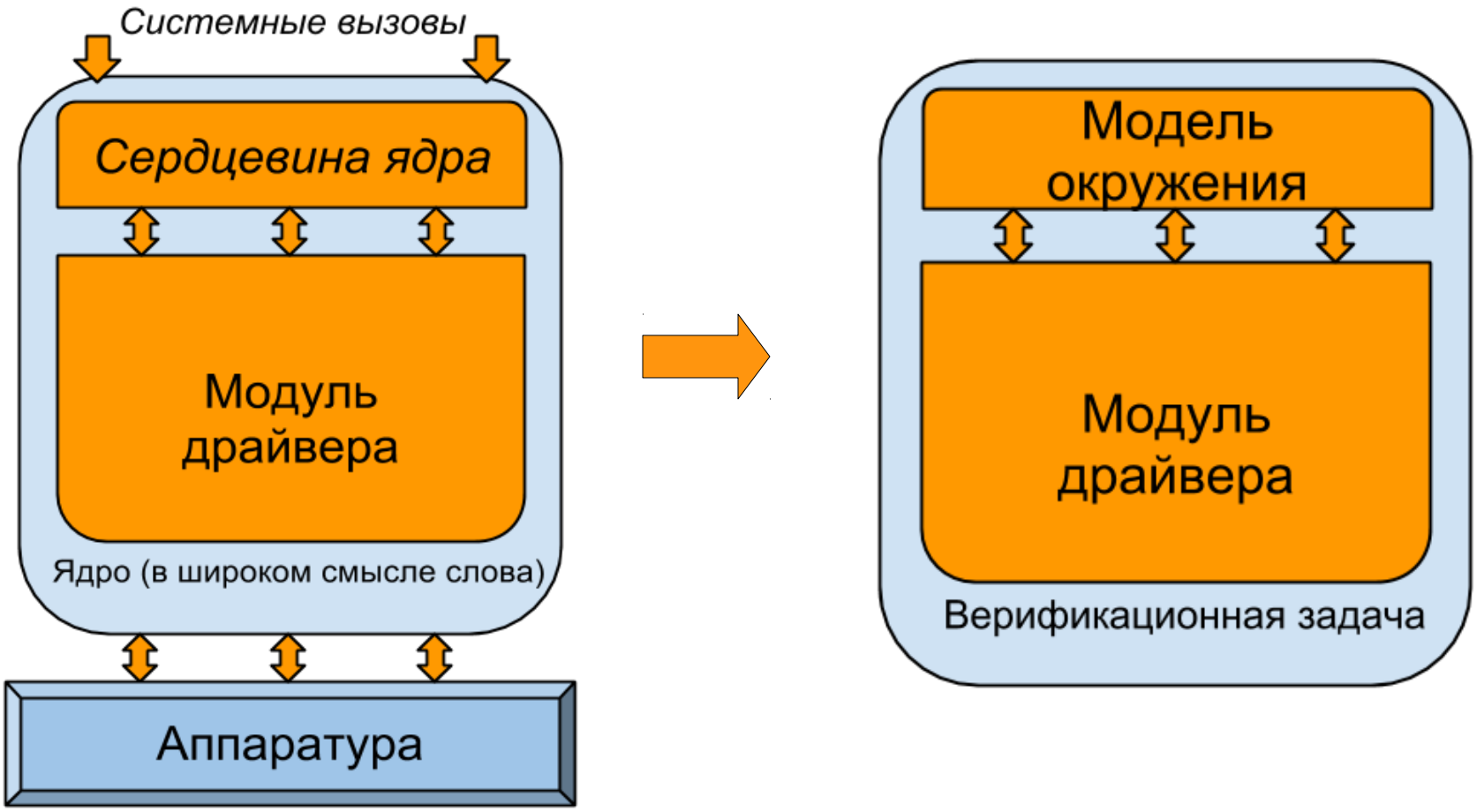


- Модули небольшого размера
- Редко используется рекурсия и арифметика с плавающей точкой

Система верификации LDV



Модель окружения драйвера



Пример модуля драйвера

```
int usbpn_probe(struct usb_interface *intf, const struct usb_device_id *id){
    ...
}
static void usbpn_disconnect(struct usb_interface *intf){
    ...
}

static struct usb_driver usbpn_struct = {
    .name =         "ldv-test",
    .probe =        usbpn_probe,
    .disconnect =   usbpn_disconnect,
};

int __init usbpn_init(void){
    return usb_register(&usbpn_struct);
}

void __exit usbpn_exit(void){
    usb_deregister(&usbpn_struct );
}
```

Обработчики

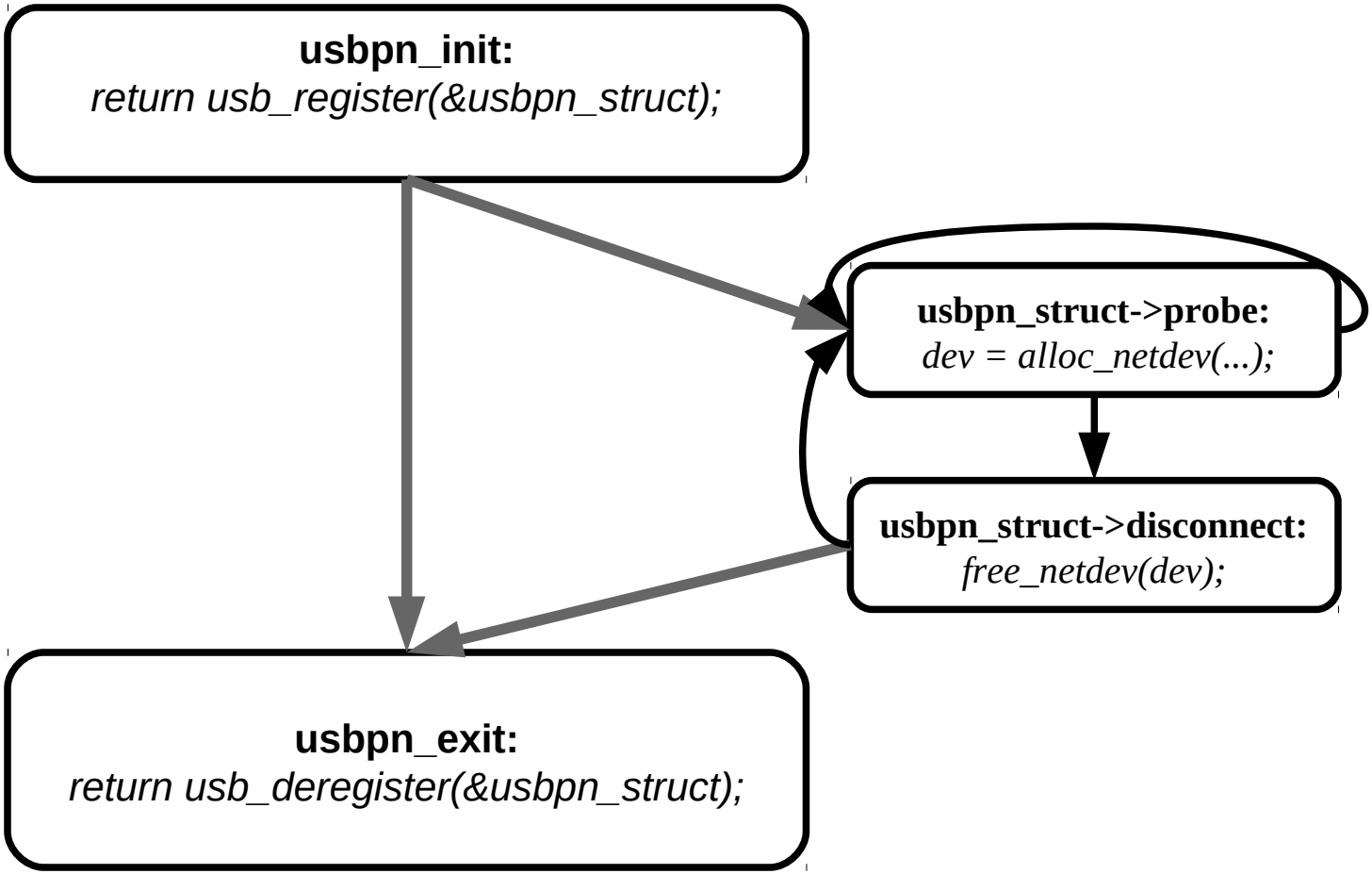
Функции инициализации и выхода

Регистрация и дерегистрация обработчиков

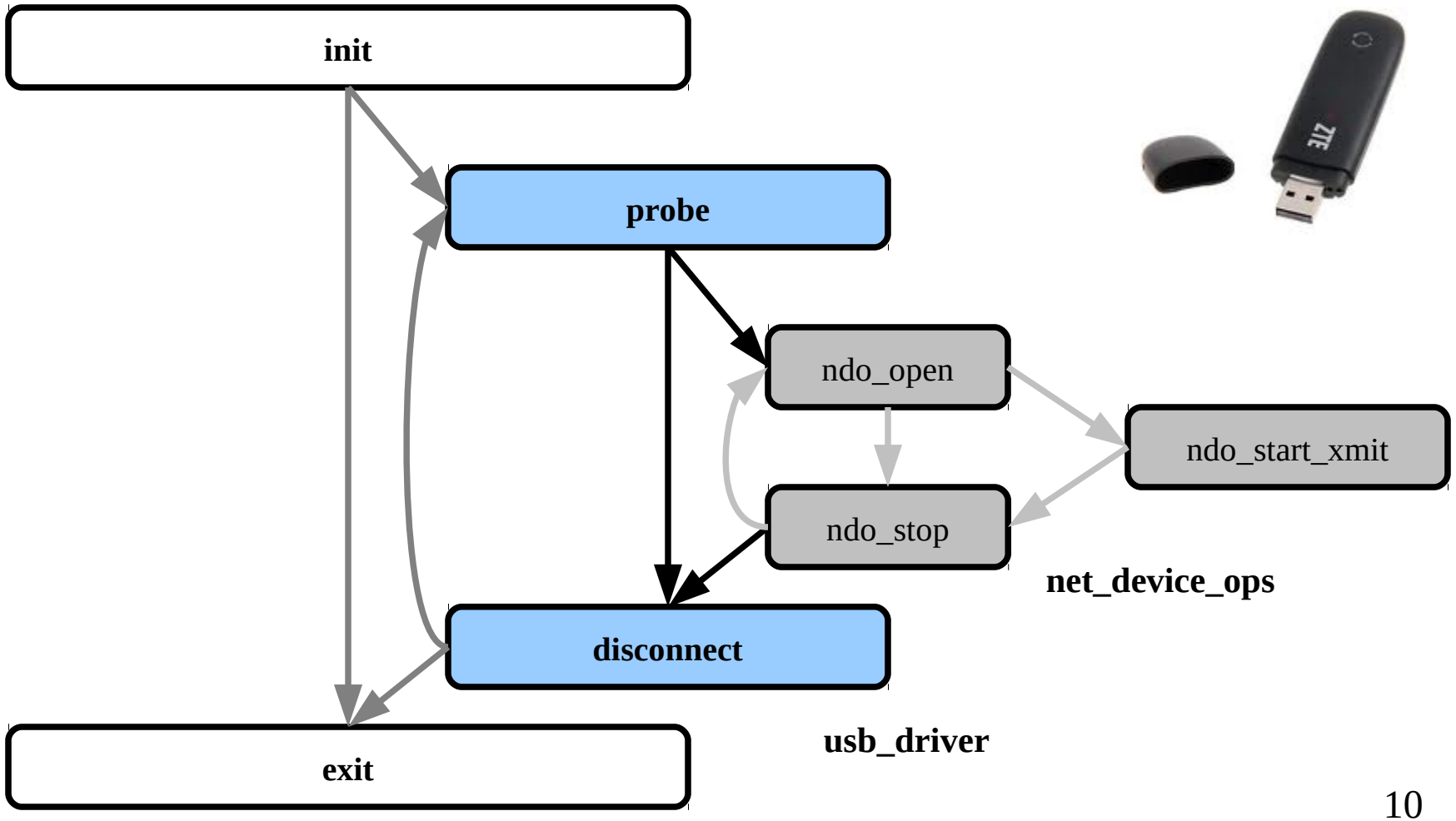
Требования к модели окружения

- Вызов функций инициализации и выхода модуля
- Вызов обработчиков из каждой группы с заданным порядком и параметрами вызова
- Вызов обработчиков из разных групп с учетом их регистрации
- Моделирование прерываний, таймеров, тасклетов и др.

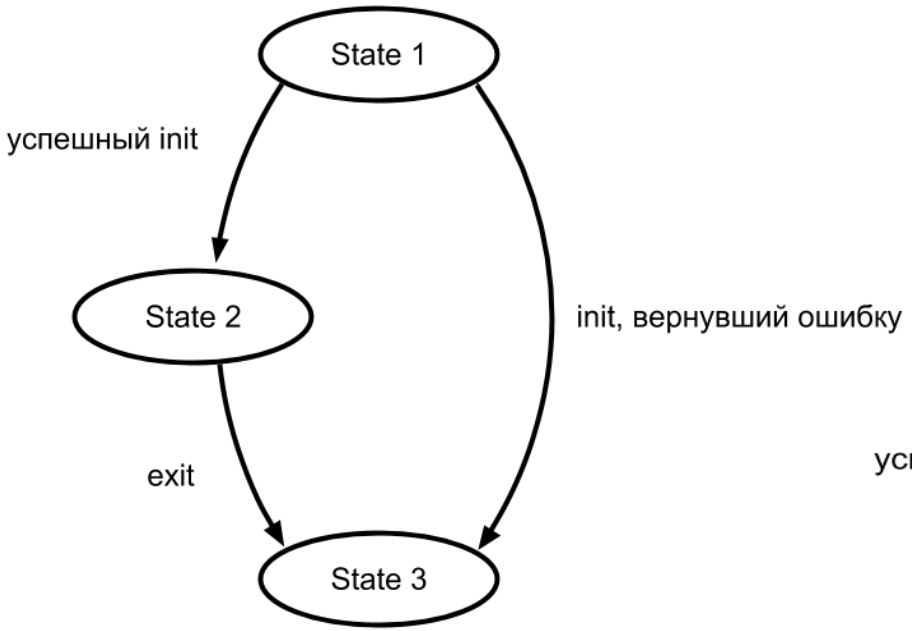
Моделирование окружения для одной группы обработчиков



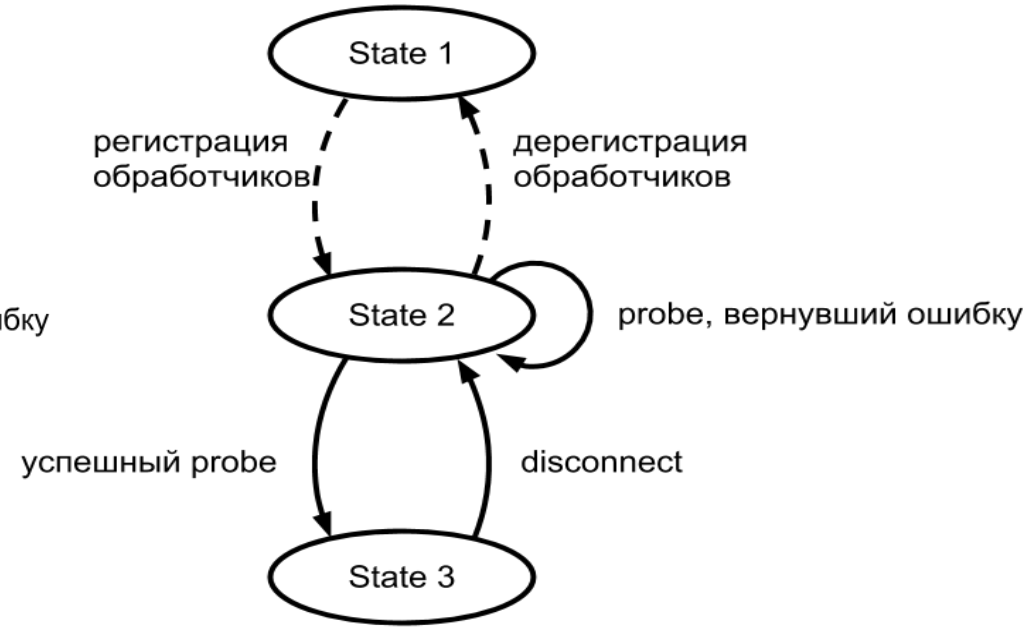
Моделирование окружения для нескольких групп обработчиков



Представление модели в виде параллельной композиции Пи-процессов

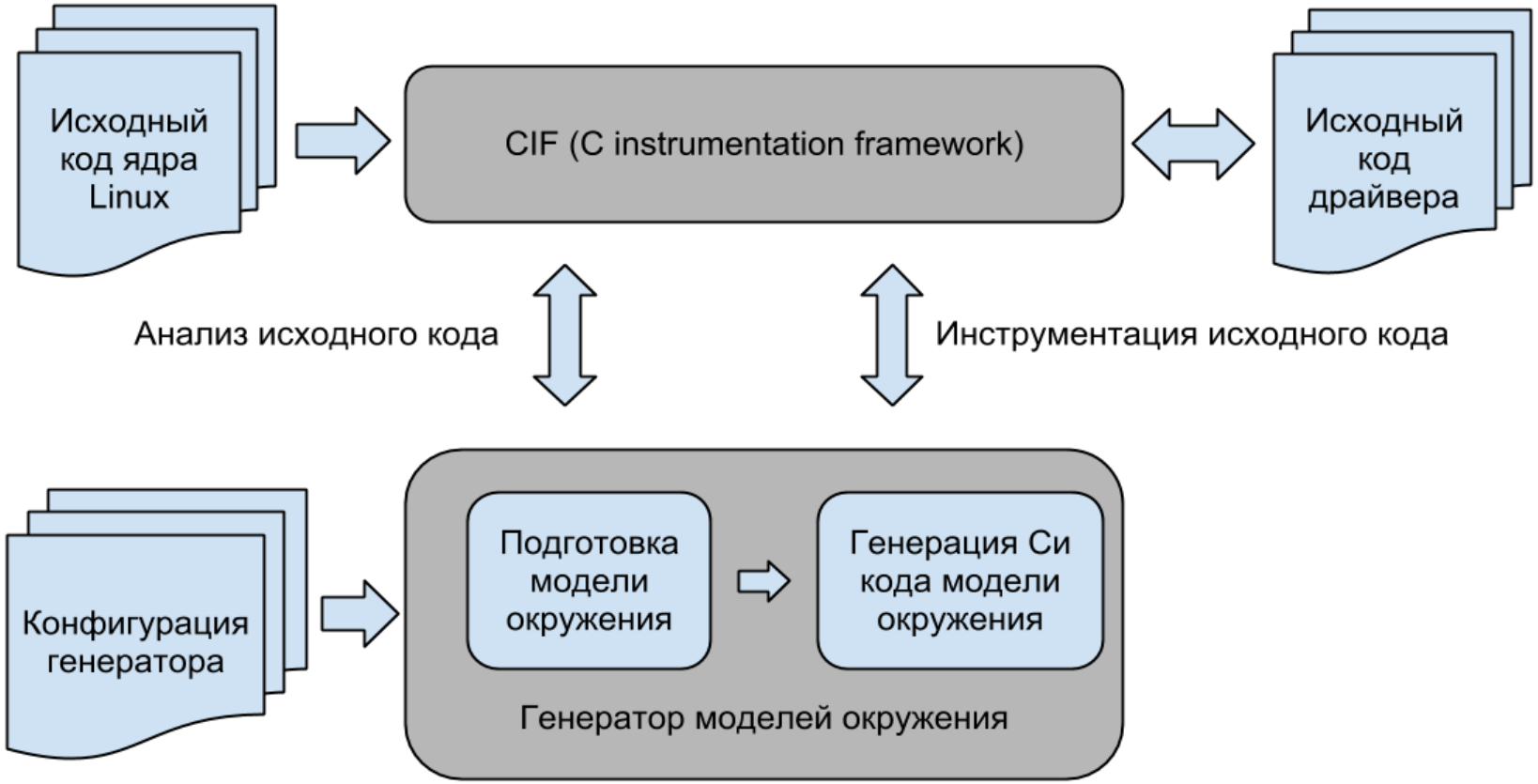


Пи-процесс для функций init и exit



Пи-процесс для usb_driver

Генерация модели окружения в системе LDV



Достоинства метода генерации модели окружения

- Модель окружения расширяема и позволяет добавлять новые элементы в модель
- Конфигурация позволяет задать практически любое необходимое поведение окружения
- Автоматическая генерация моделей окружения
  - Анализ исходного кода при помощи CIF
  - Инструментация исходного кода при помощи CIF

Недостатки метода генерации модели окружения

- Необходимость привлечения квалифицированного специалиста для подготовки конфигурации генератора
- Трудно поддержать большое число разных групп обработчиков
- Ограниченные возможности анализа исходного кода

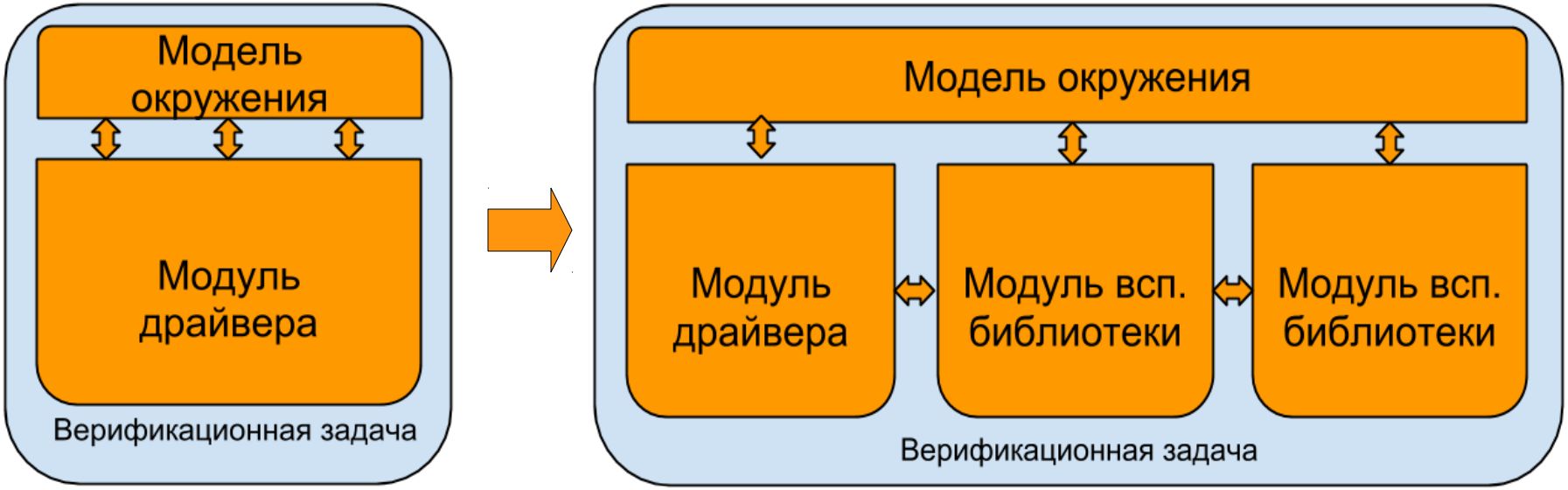
Как улучшить метод генерации модели окружения ?

Замена части модели окружения на соответствующую часть реального окружения позволит снизить уровень требований к полноте модели окружения

Вспомогательные библиотеки драйвера

- Драйвер – код, предназначенный для взаимодействия с конкретным устройством
- Вспомогательные библиотеки - часть ядра, с которой взаимодействует данный драйвер
  - Регистрация обработчиков
  - Экспортируемые функции
  - Инициализация и выделение памяти

Модель окружения для драйвера и вспомогательных библиотек



Преимущества и недостатки совместной верификации группы модулей

- Верифицируются новые функции, пропущенные из-за неполноты модели окружения
- При верификации анализируются функции библиотек, а не их модели
- Не требуется генерировать модель для ряда групп обработчиков
- Возрастет сложность модели окружения
- Увеличиваются потребности верификаторов в ресурсах (память, время)

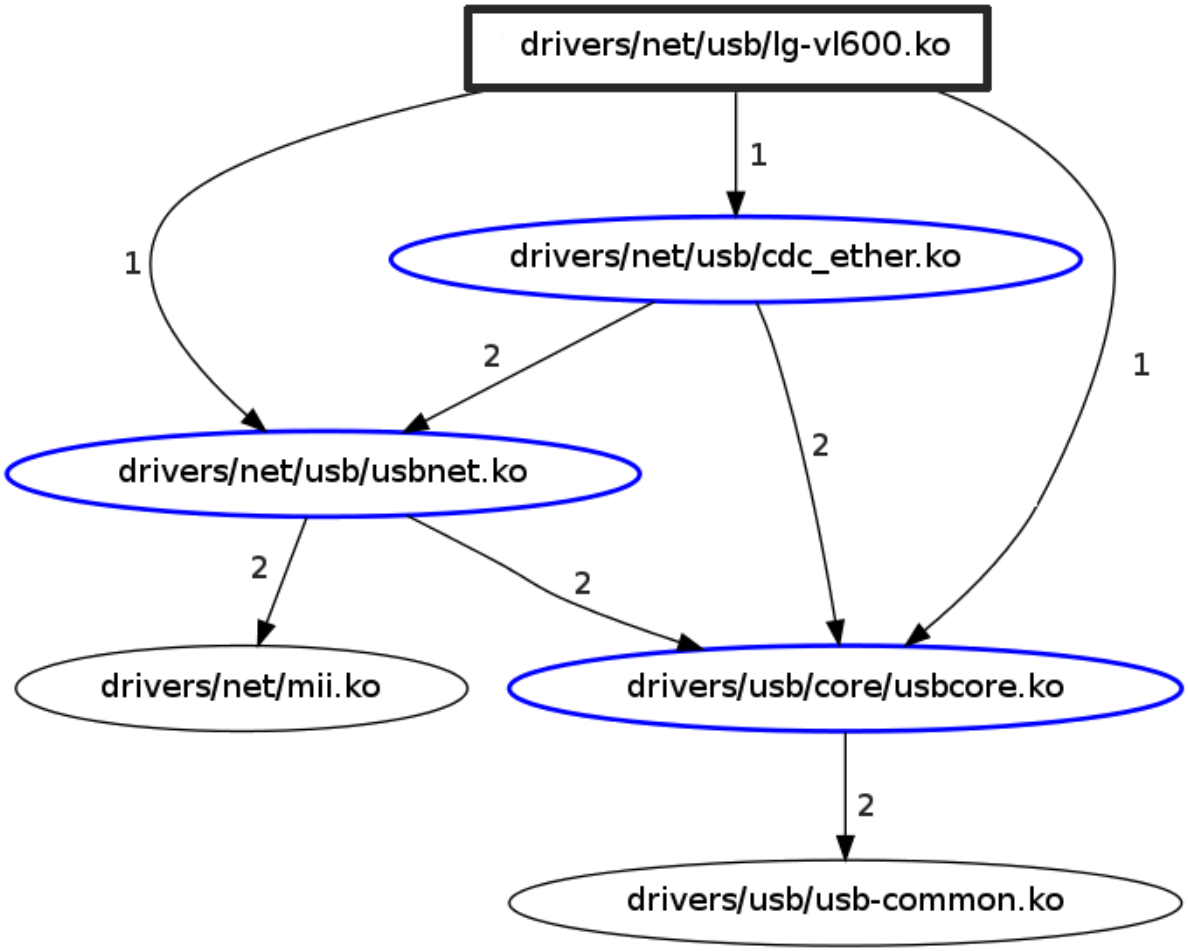
Постановка задачи

- Автоматически определять вспомогательные библиотеки для каждого модуля драйвера
- Генерировать модель окружения для группы модулей

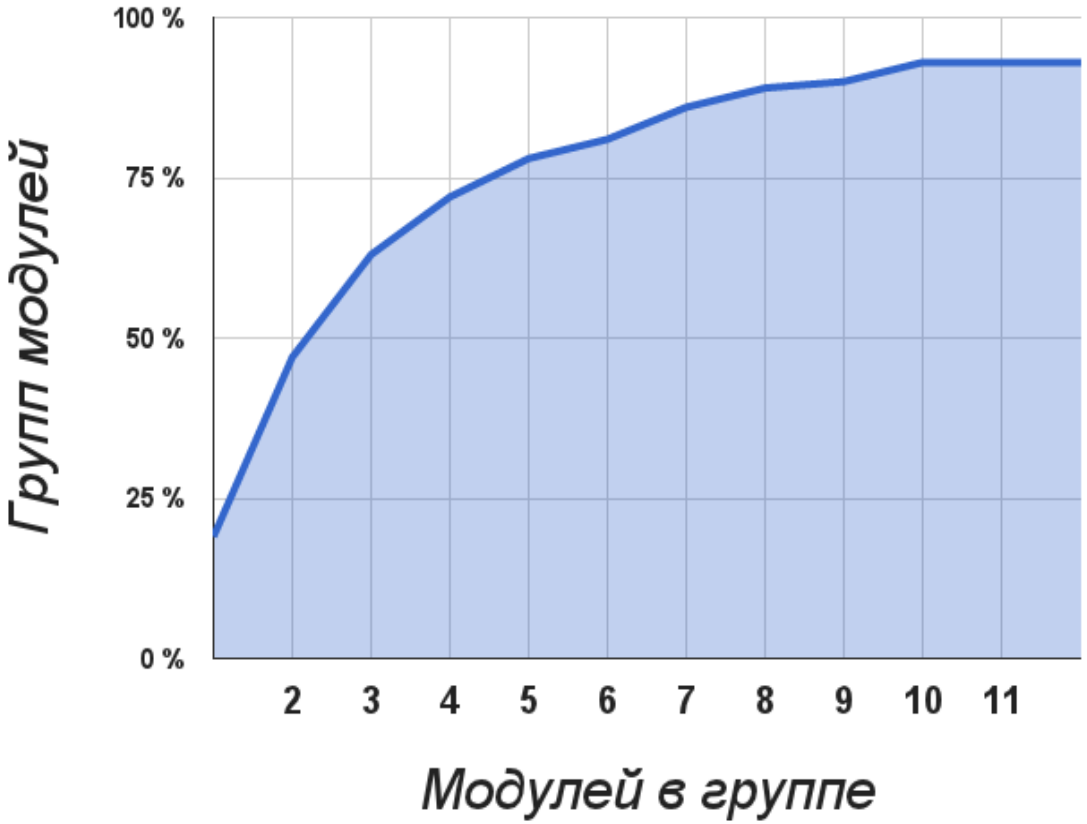
Выбор группы модулей

- Критерий - экспортируемые и импортируемые модулем ядра символы
- Извлечение зависимостей выполняется при помощи depmod
- Для каждого модуля рекурсивно определяются модули, от которых он зависит

Пример ориентированного графа зависимостей модуля



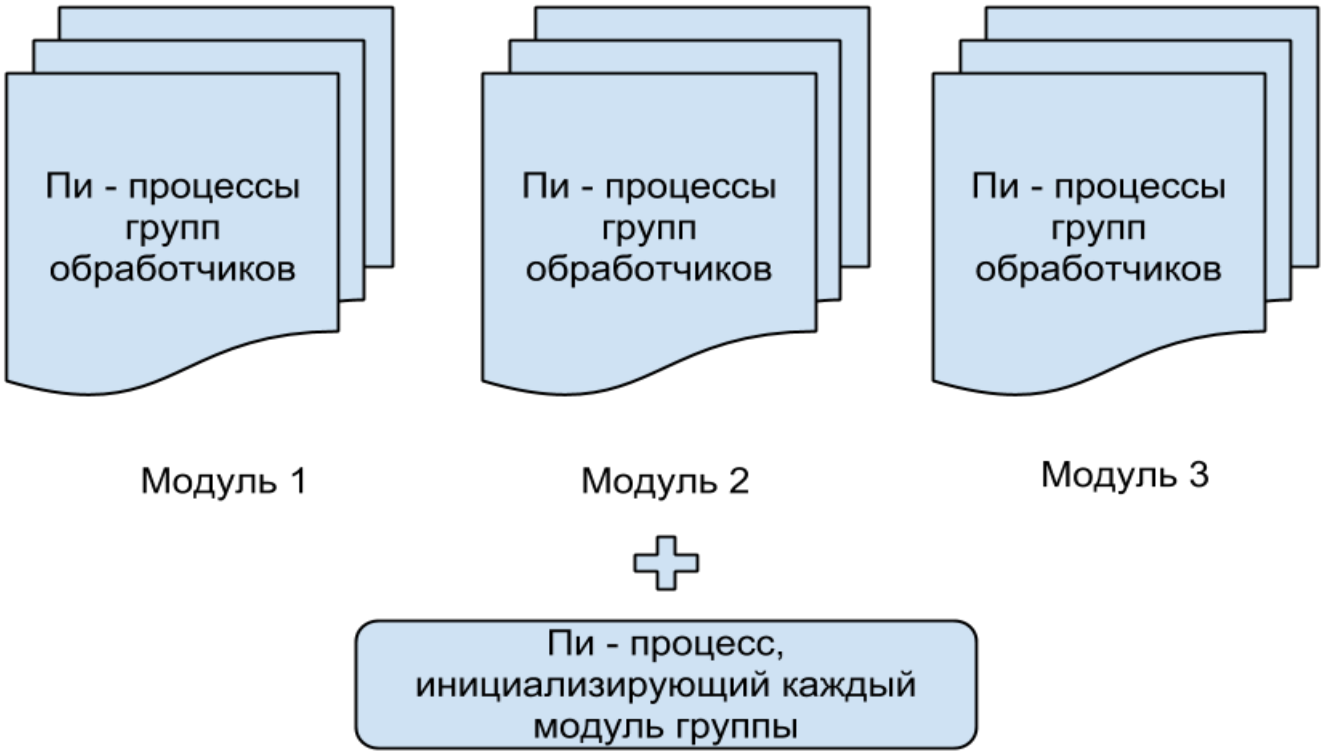
Размер групп модулей



Модель окружения для одного модуля

- Опирается на анализ исходного кода
- Представляет собой параллельную композицию Пи-процессов соответствующих группам обработчиков

Модель окружения для группы модулей



Дополнительные требования к модели окружения

- Инициализация всех модулей группы перед вызовом обработчиков
- Учитывать межмодульное взаимодействие при моделировании каждой группы обработчиков

Результаты

- Разработан и реализован метод автоматического выделения вспомогательных библиотек модуля ядра
- Начата разработка генератора моделей окружения для группы модулей

Дальнейшие направления работы

- Завершить разработку генератора моделей окружения для группы модулей
- Эксперименты с другими алгоритмами выделения групп модулей
- Конфигурирование полноты модели
- Получение и анализ результатов верификации ядра

LinuxTesting.org

Спасибо за внимание!

Ilja Zakharov
Ilja.zakharov@ispras.ru

http://linuxtesting.org/project/ldv,
gsoc2013/glaurung/5001

ISP RAS
Institute for System Programming of the Russian Academy of Sciences