23 Ноября 2012, Москва

LinuxTesting.org

Моделирование окружения драйверов операционной системы Linux для поддержки процесса статической верификации

Захаров Илья Сергеевич
Ilja.zakharov@ispras.ru

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

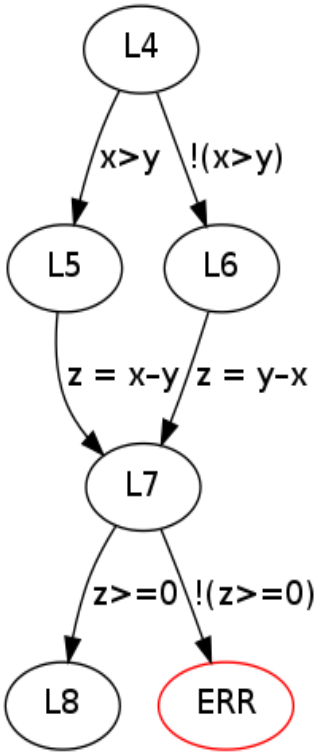
Введение

- Большое число разработчиков
  - Более 1000
  - С 2005 года участвовало более 7800
- Высокая скорость разработки
  - Релиз каждые 2-3 месяца
  - Включает около 10 тыс. патчей
- Высокие требования к корректности
  - Работа в привилегированном режиме
  - 85 % падений системы из-за ошибок в драйверах
- Высокая сложность кода

Инструменты статической верификации.

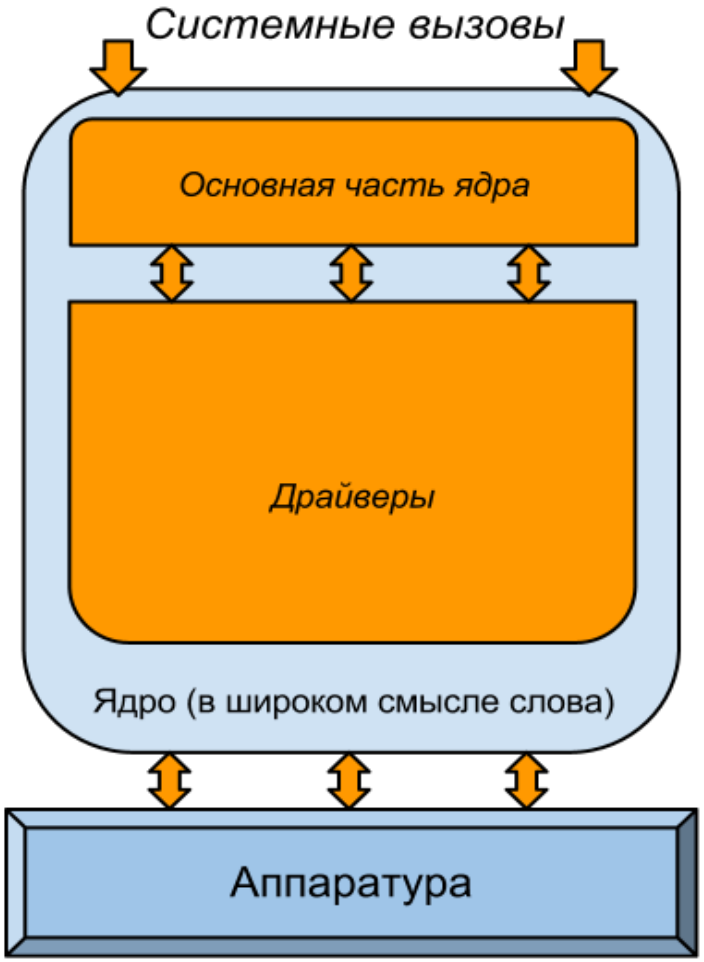
```
int x,y,z;

L1: void main() {
L2:         x = nondet_int();
L3:         y = nondet_int();
L4:         if(x>y) {
L5:                 z=x-y;
            } else {
L6:                 z=y-x;
            }
L7:         assert(z>=0);
L8: }
```



Необходима точка входа

Драйверы в ядре Linux.



Пример драйвера

```
int ldv_probe(struct usb_interface *intf, const struct usb_device_id *id){
    ...
}
static void ldv_disconnect(struct usb_interface *intf){
    ...
}
static struct usb_driver ldv_driver = {
    .name =         "ldv-test",
    .probe =    ldv_probe,
    .disconnect =    ldv_disconnect,
};
int __init ldv_init(void){
    return usb_register(&ldv_driver);
}
void __exit ldv_exit(void){
    usb_deregister(&ldv_driver);
}
module_init(ldv_init);
module_exit(ldv_exit);
```

Обработчики (точки входа)

Структура драйвера

Функция загрузки (точка входа)

Функция регистрации (точка выхода)

Функция выгрузки (точка входа)

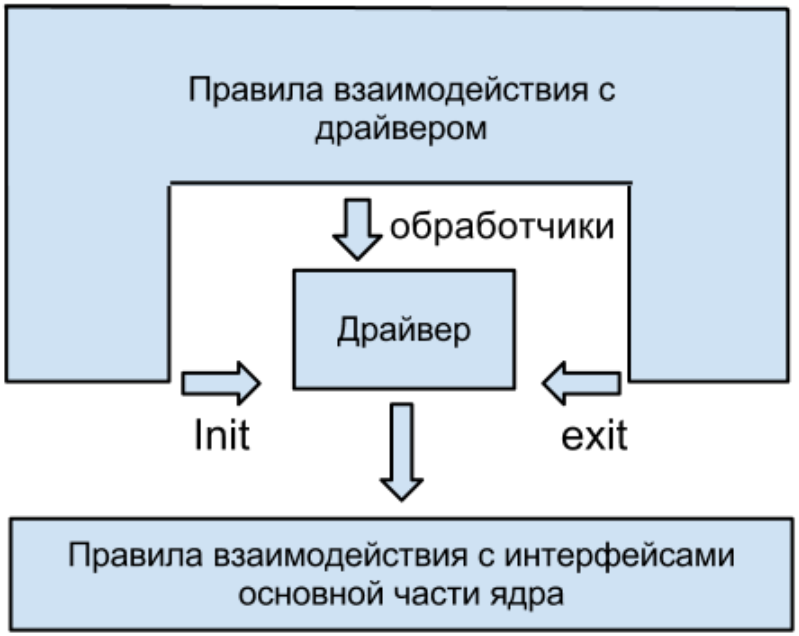
Функция дерегистрации (точка выхода)

Макросы обозначающие функции загрузки и выгрузки

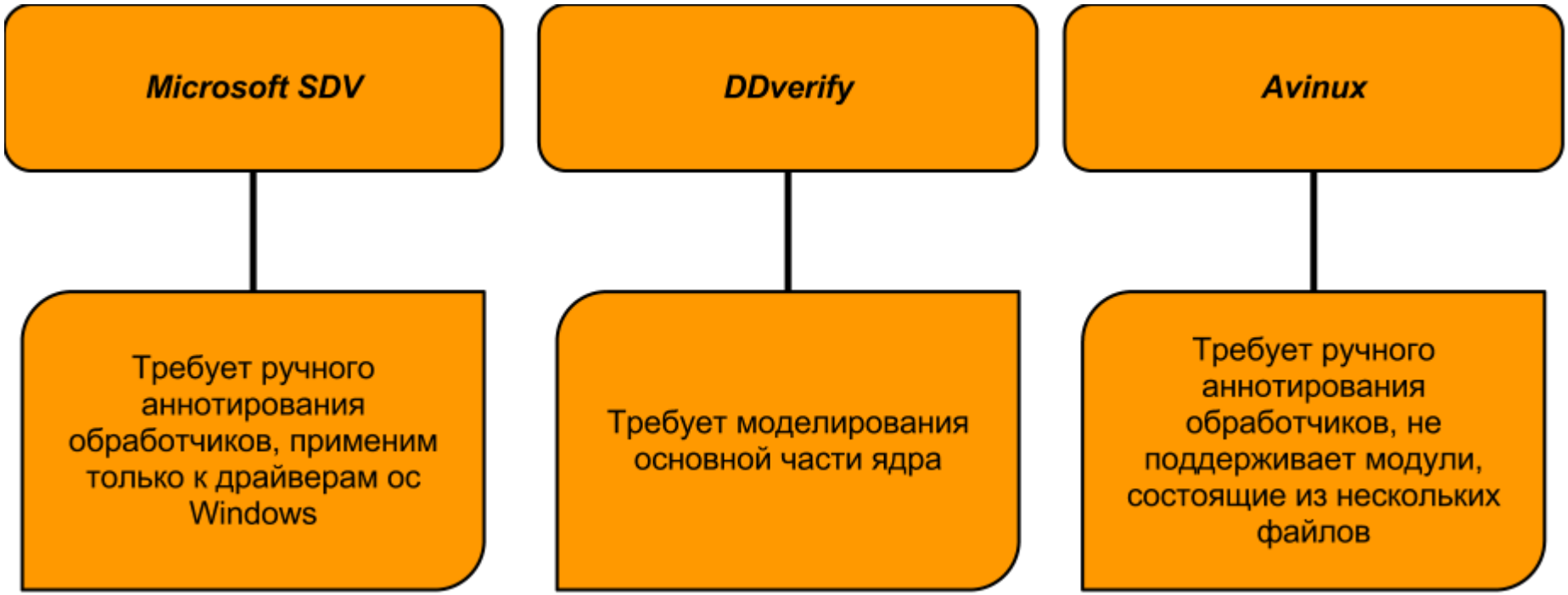
Модель окружения

Обращения из основной части ядра:

- Вызов *init* и *exit*
- *Вызов обработчиков из одной структуры драйвера*
- *Взаимодействие обработчиков из разных структур драйвера*



Инструменты статической верификации драйверов



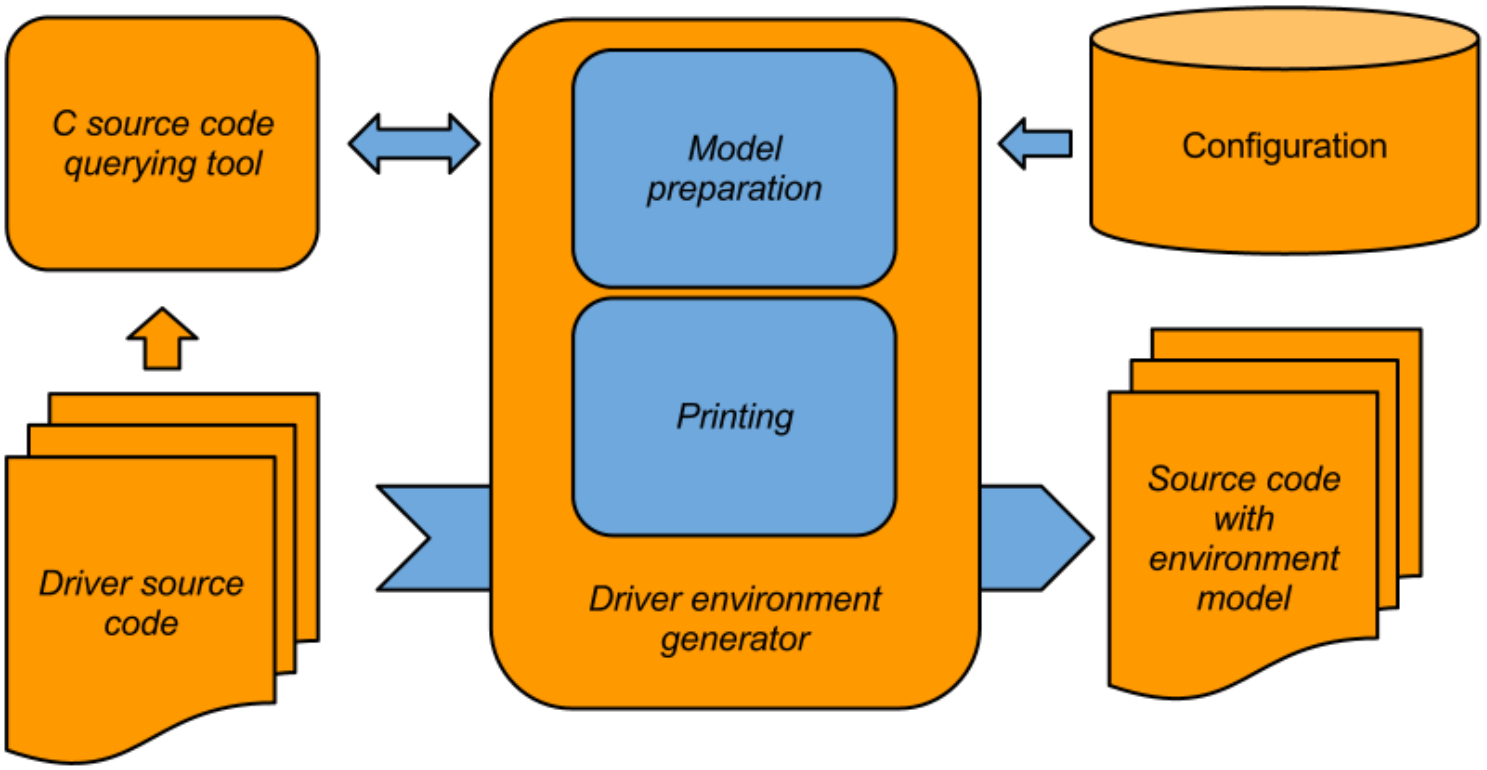
Ограничения генератора окружений LDV

Недостаточные возможности по заданию конфигурации:

- Нельзя описать все возможные последовательности вызовов обработчиков из одной структуры.
- Нельзя описать Взаимодействие обработчиков из разных структур драйвера

Сбор информации по исходному коду драйверов для моделирования окружения на основе регулярных выражений.

Генерация окружения



Особенности нового генератора окружений

- Улучшенные возможности по заданию конфигурации:
  - Можно описать все возможные последовательности вызовов обработчиков из одной структуры.
  - Можно описать Взаимодействие обработчиков из разных структур драйвера
- Сбор информации по исходному коду драйверов при помощи внешнего инструмента.
- Повышение удобства конфигурирования и анализа модели

Результаты

В среднем 30 % лучшие показатели по сравнению со старым генератором моделей окружения LDV.

- Использовался инструмент статической верификации BLAST
- ядра Linux 3.0.1 и 3.5.3
- Проверялось несколько видов ошибок: инициализация спинлоков, освобождение памяти с флагом ATOMIC, парное использование module_put и module_get и т.д.

Дальнейшее направление работы

- Расширение существующей конфигурации
- Добавление в модель прерываний, таймеров, очередей и т. д.
- Оптимизация сгенерированной модели для верификатора

LinuxTesting.org

Спасибо!

Илья Захаров
Ilja.zakharov@ispras.ru
http://linuxtesting.org/project/ldv

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