Московский физико-технический институт (государственный университет)
Факультет радиотехники и кибернетики
Кафедра информатики и вычислительной техники

Поддержка сигналов POSIX в ОС “Эльбрус”

Студент: Родионов Дмитрий, группа 213

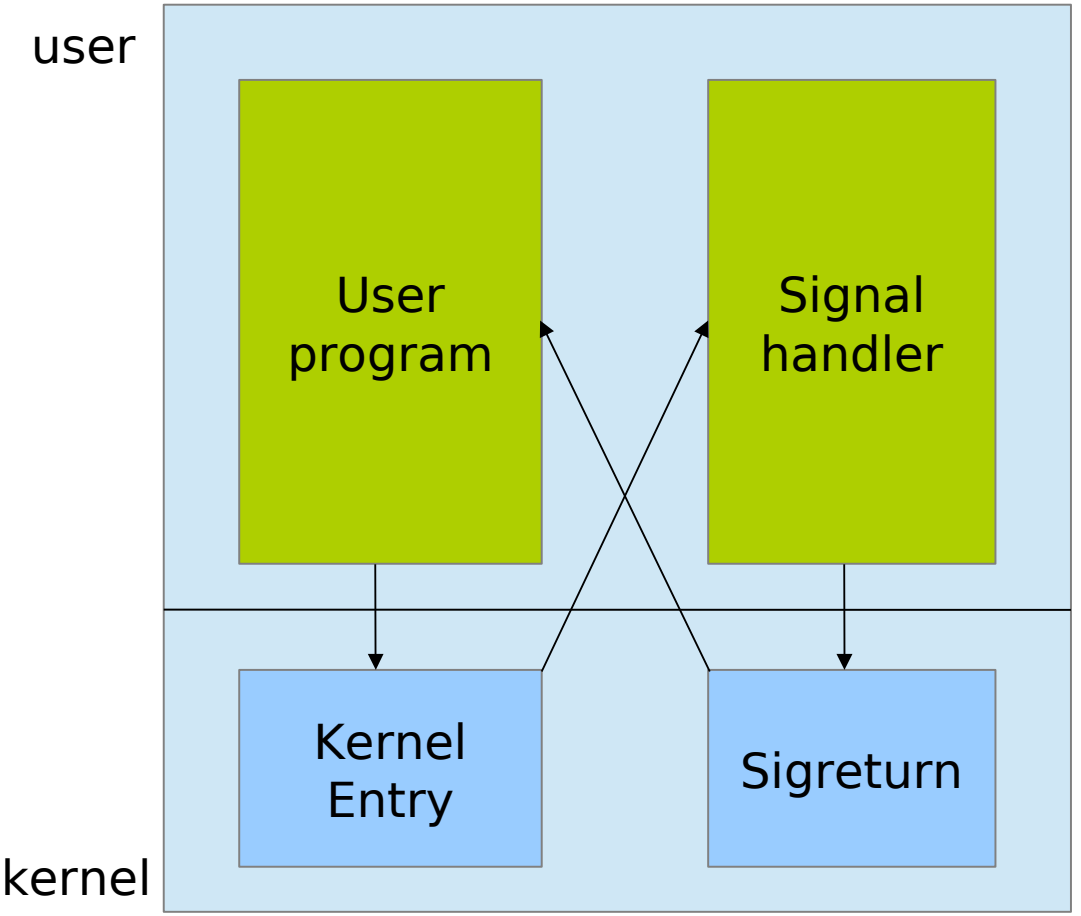
Научный руководитель: Федоров А.В.

Сигналы Unix

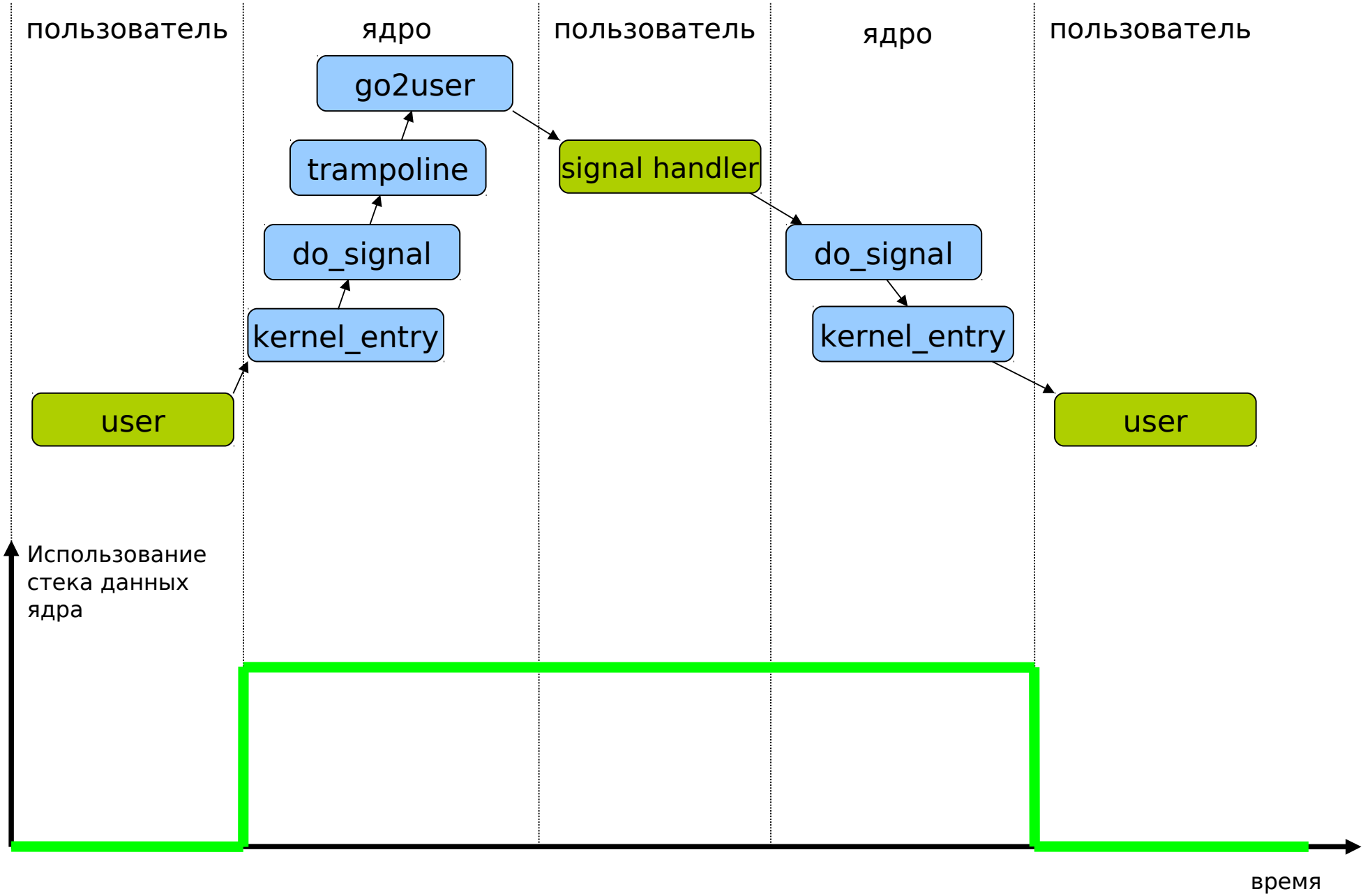
Сигнал – асинхронное уведомление процесса о каком-либо событии.

Сигналы в Linux являются одним из способов взаимодействия между процессами (**IPC** – inter-process communication).

Доставка сигнала – процесс передачи управления пользовательскому обработчику.



Существующая схема доставки сигналов



Постановка задачи

Недостаток существующей схемы доставки сигналов: во время исполнения обработчика стек данных ядра хранит контекст пользователя. Итог – невозможна большая глубина вложенности обработчиков, т.к. стек данных ядра ограничен.

Задача: разработать и протестировать схему доставки сигналов, при которой стек данных ядра будет свободен к моменту вызова обработчика.

Требования к реализации:

- Не занимать стек данных ядра
- Обеспечить перезапуски системных вызовов

Аппаратные стеки

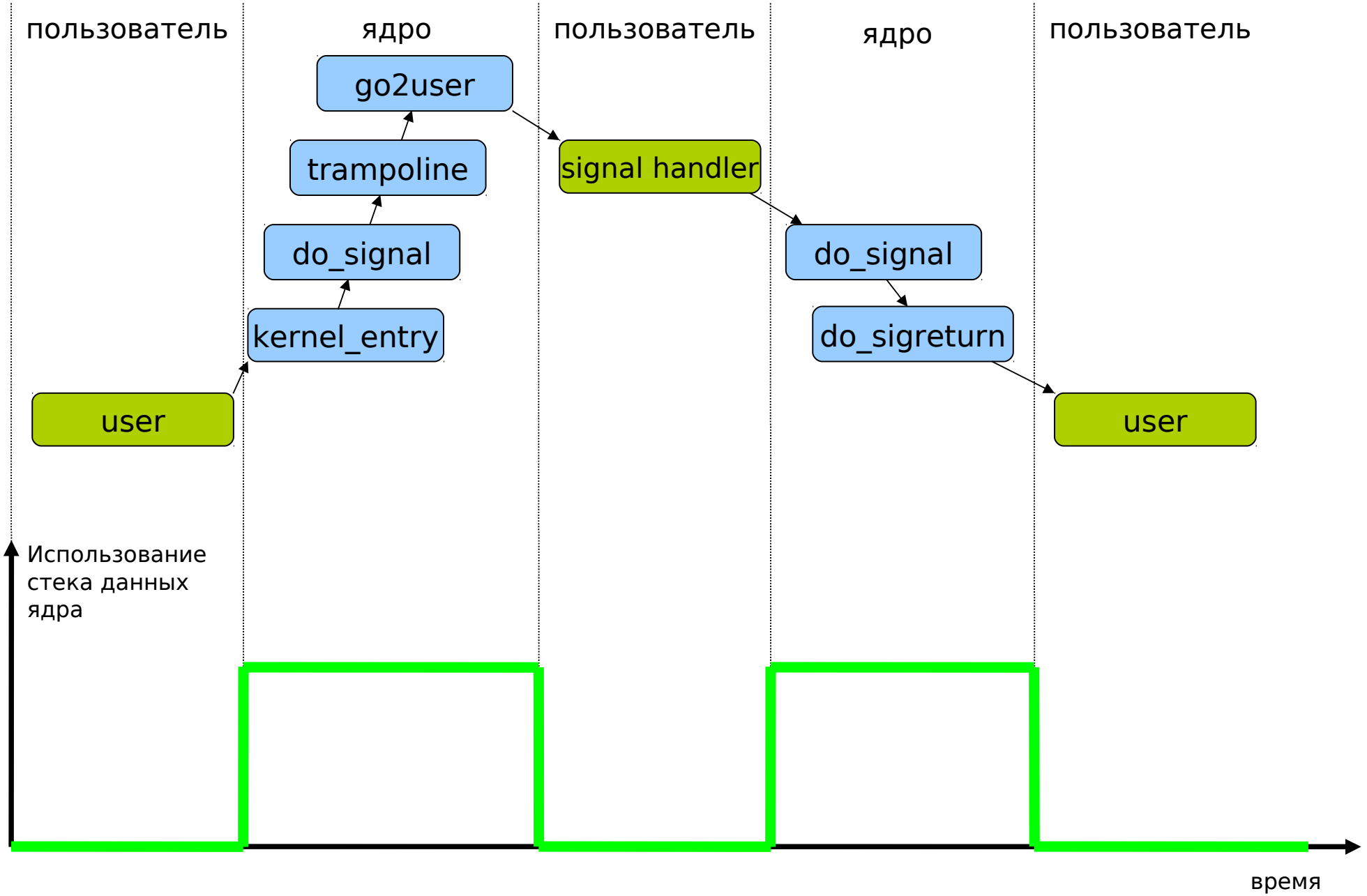
Помимо стека данных, доступного пользователю, на архитектуре Эльбрус существует два **привилегированных** стека:

Процедурный стек — стек, через который происходит передача параметров и в котором хранятся локальные переменные.

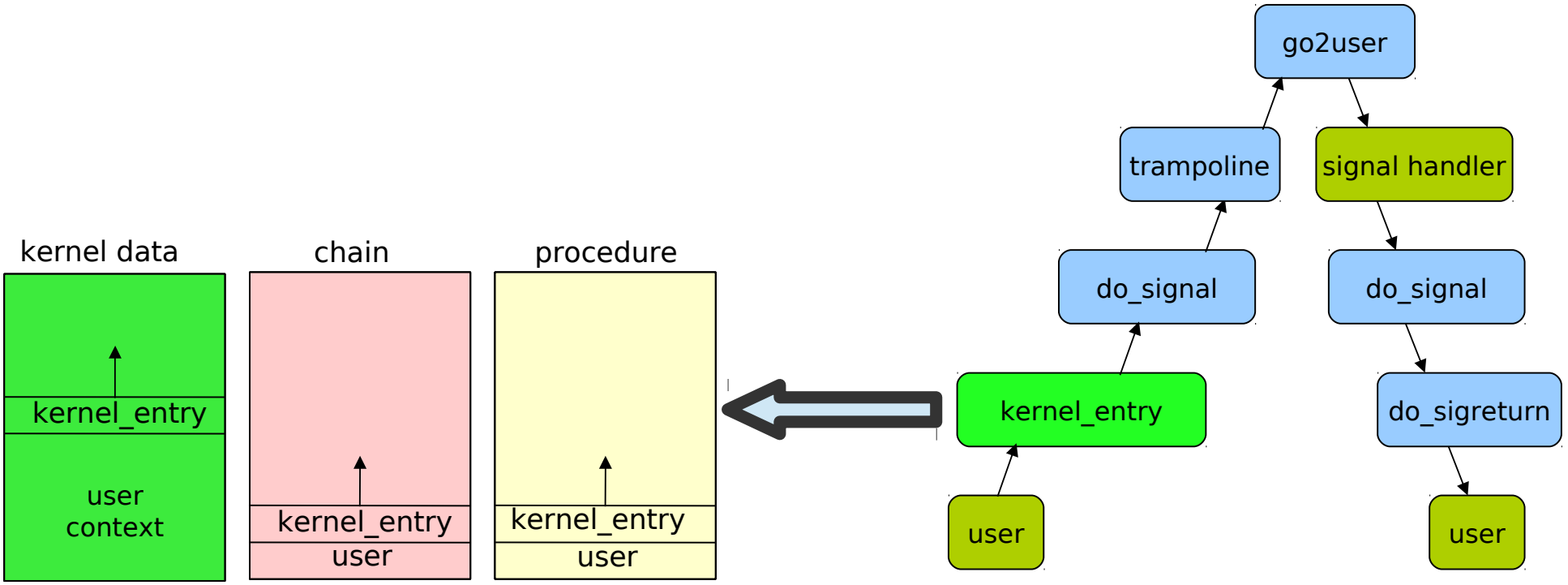
Стек связующей инфомации — стек, хранящий в себе информацию об изменениях, необходимых при возврате из процедуры.

Подход к проблеме: использовать для хранения пользовательского контекста защищенный от пользователя процедурный стек, в отличие от других архитектур, использующих стек данных пользователя.

Предлагаемая схема доставки сигналов

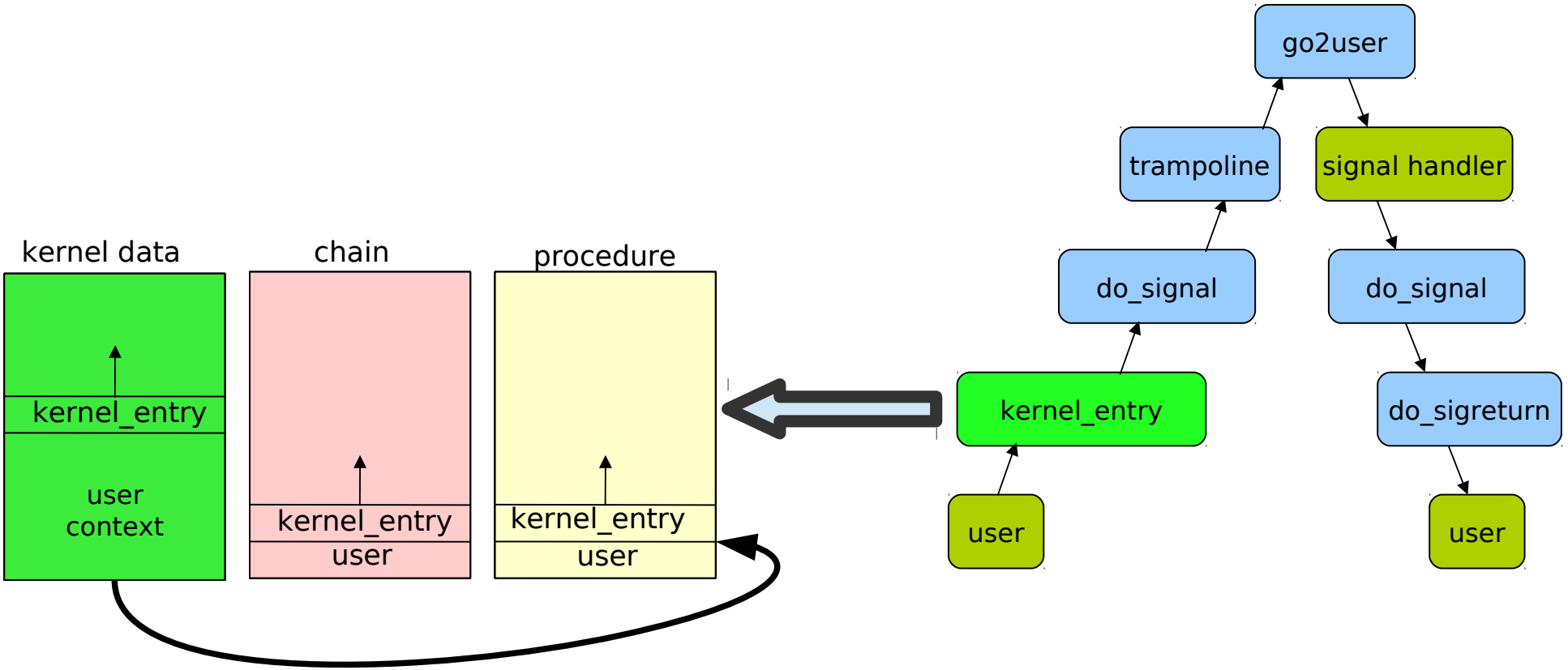


Состояние аппаратных стеков



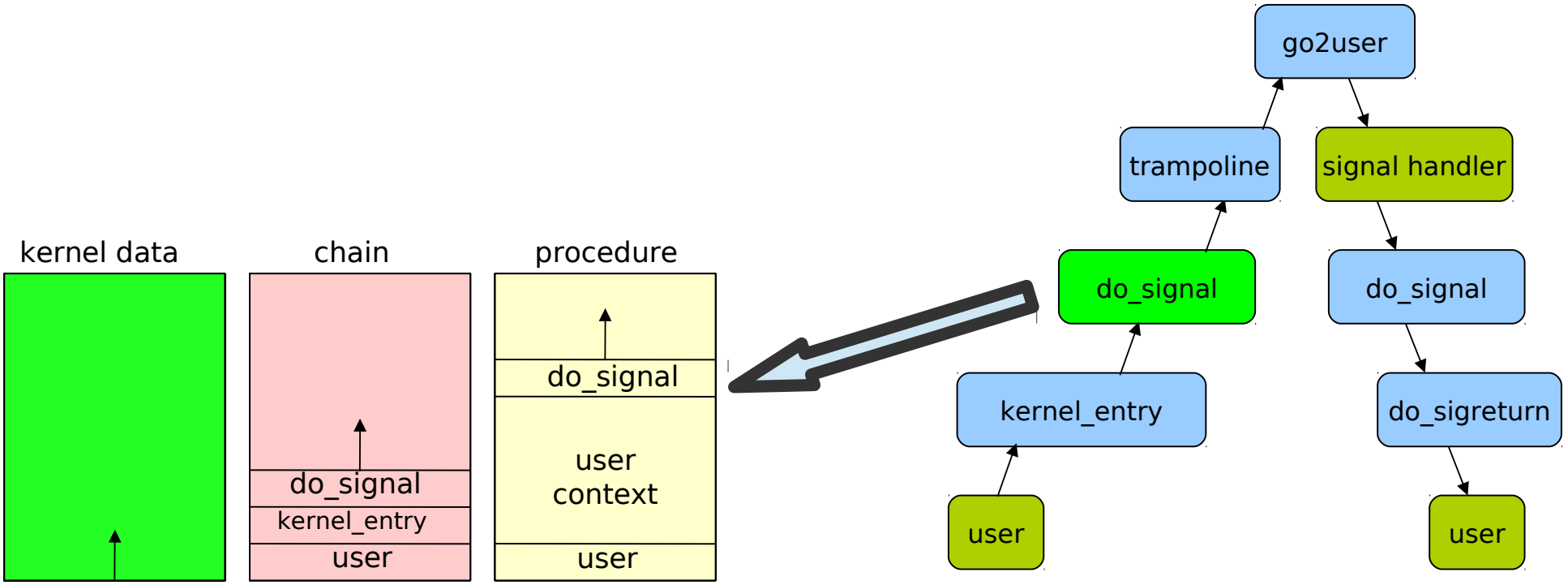
Процедура **do_signal()** сохраняет контекст пользователя в процедурное окно предыдущей функции

Состояние аппаратных стеков



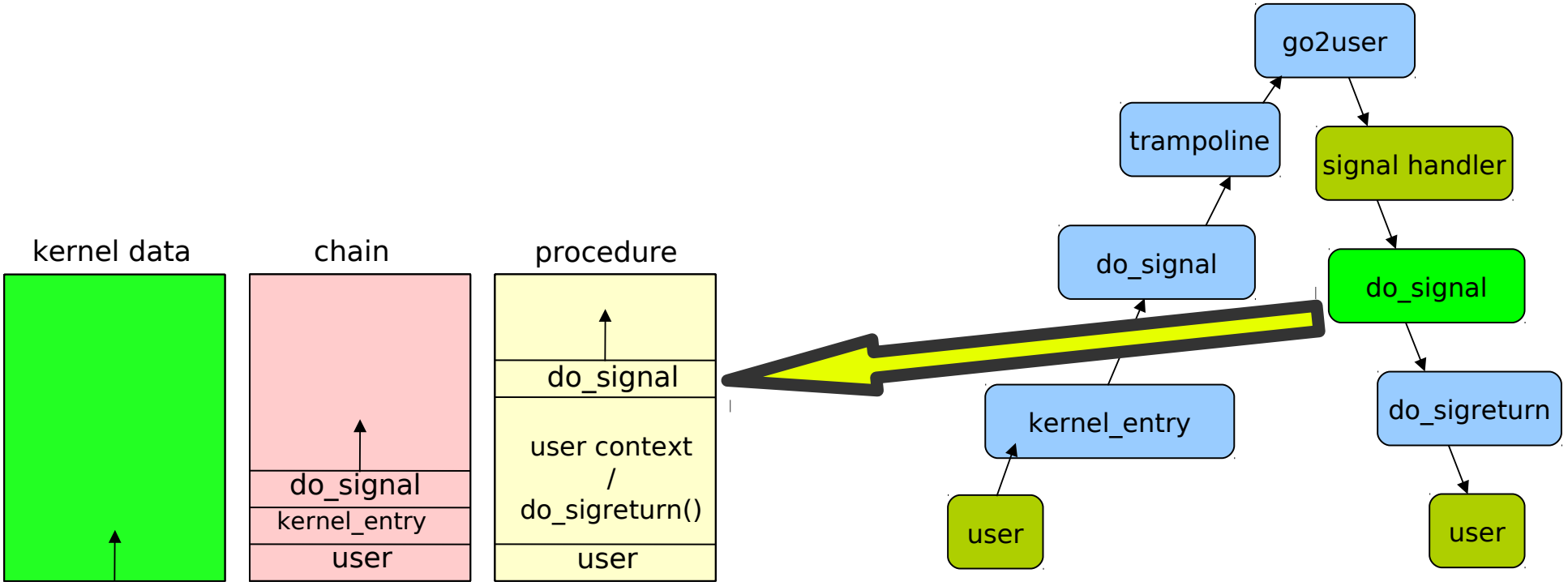
Процедура **do_signal()** сохраняет контекст пользователя в процедурное окно предыдущей функции

Состояние аппаратных стеков



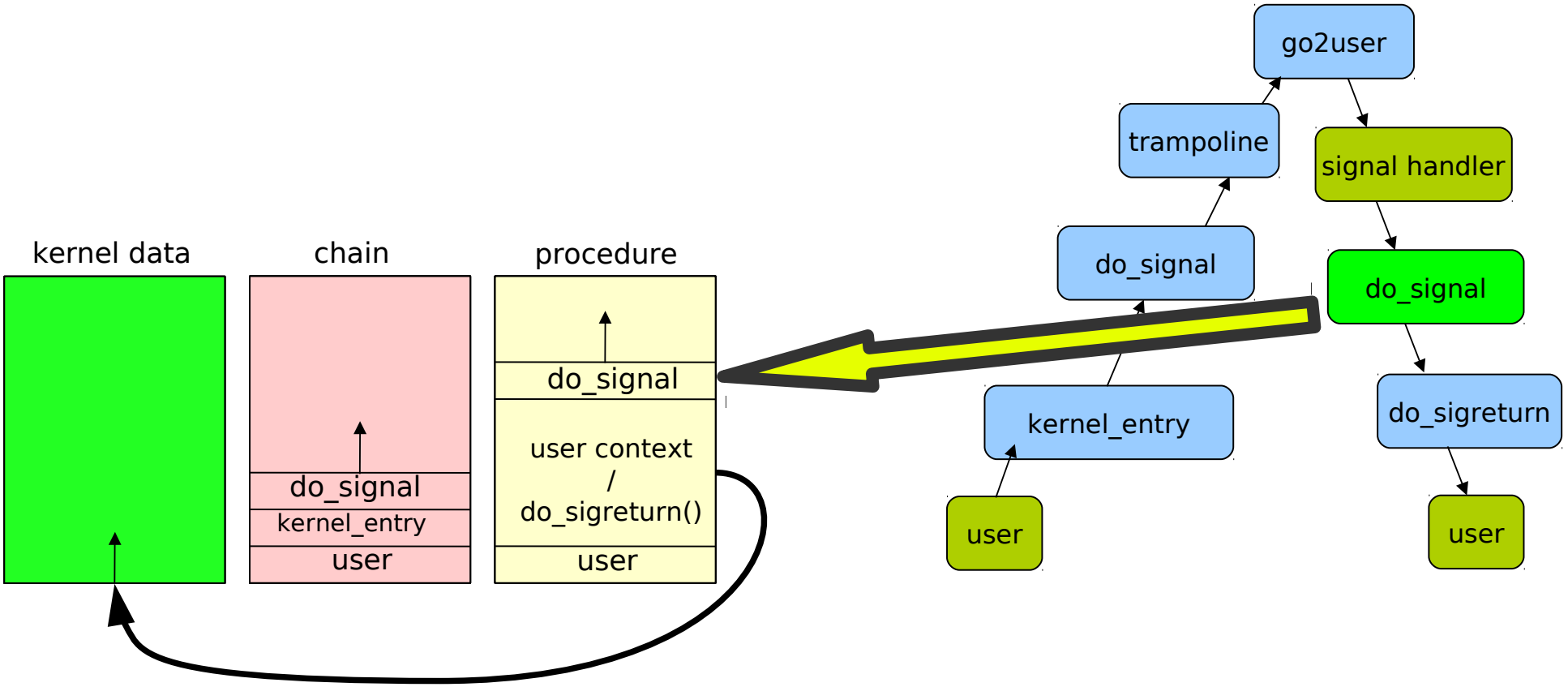
Процедура **do_signal()** сохраняет контекст пользователя в процедурное окно предыдущей функции

Восстановление контекста



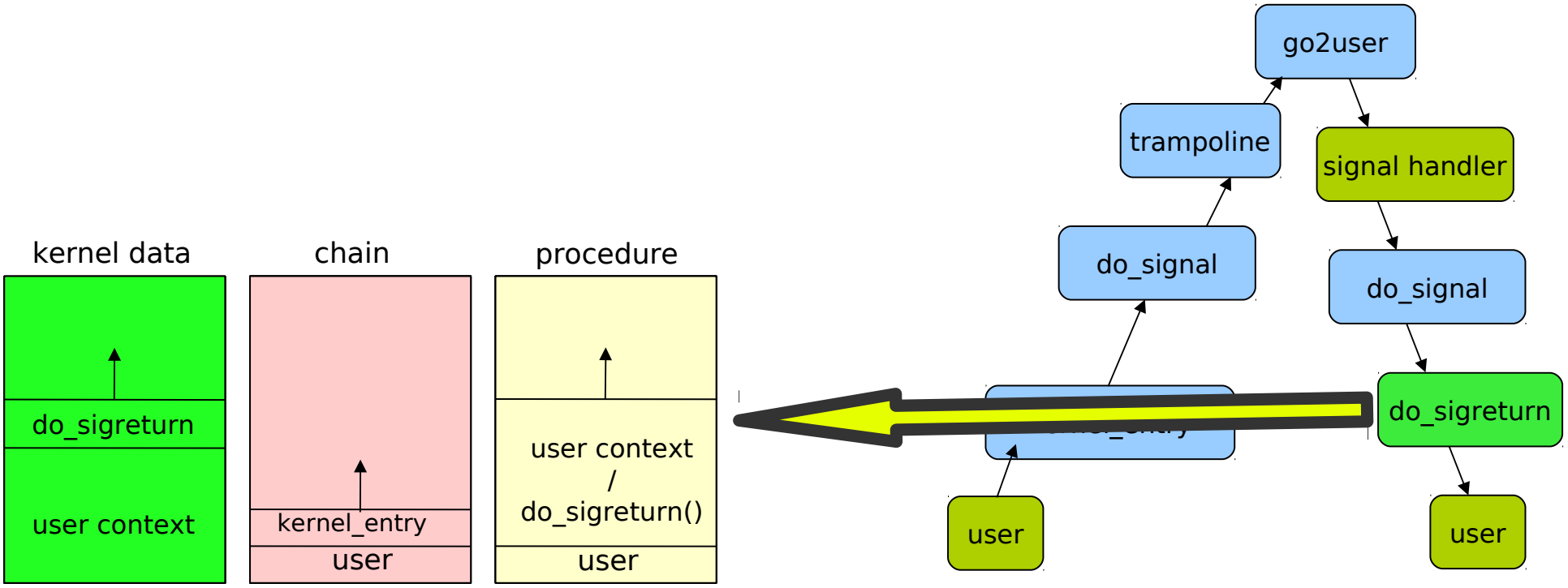
После возврата из обработчика функция **do_signal()** копирует контекст пользователя в стек данных ядра

Восстановление контекста



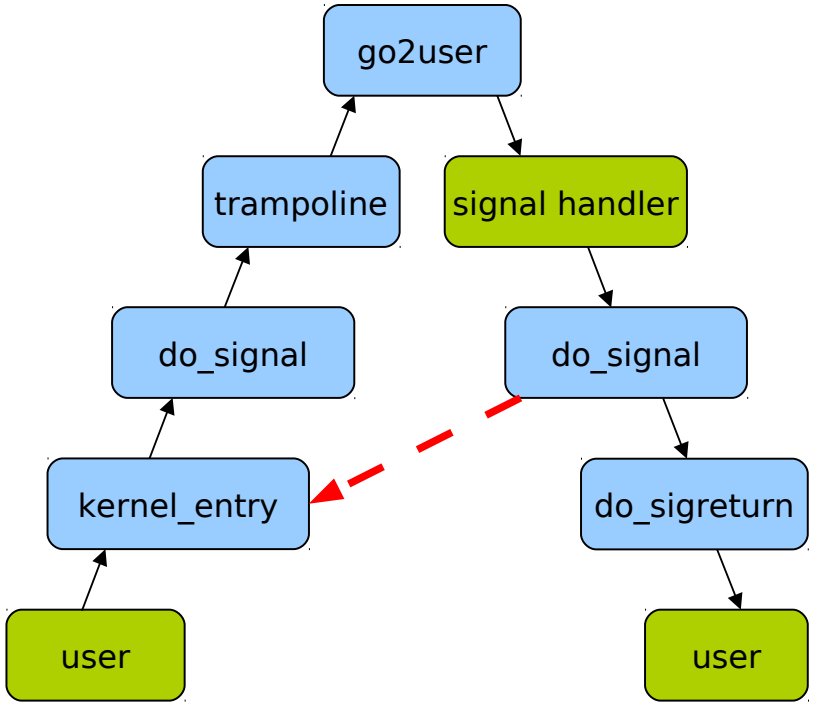
После возврата из обработчика функция **do_signal()** копирует контекст пользователя в стек данных ядра

Восстановление контекста



После возврата из обработчика функция **do_signal()** копирует контекст пользователя в стек данных ядра

Поддержка перезапусков системных вызовов



- существующая схема

В существующей схеме из do_signal осуществлялся возврат в kernel_entry. В новой схеме в этот момент в стеке данных ядра уже уничтожен кадр функции kernel_entry.

Проблема: нельзя осуществить перезапуск системного вызова средствами языка Си в функции kernel_entry.

Решение: восстановить все регистры в соответствии с состоянием на момент входа в ядро и сделать туда **jump**.

Тестирование

- было произведено тестирование пакетами posixtestsuite и LTP
- была проверена возможность доставки большого количества вложенных сигналов (100 000)
- время доставки сигнала увеличилось на ~20% из-за копирования контекста

Результаты

- Разработана и отлажена новая схема доставки сигналов
- Поддержаны перезапуски системных вызовов
- Найдена и исправлена ошибка с рестартами системных вызовов в защищенном режиме.
- Проверка произведена пакетами LTP и posixtestsuite.

Дальнейшая работа

- Стало возможным привести структуру *struct pt_regs* к виду, сходному с другими архитектурами.
- Переход на архитектурно-независимый интерфейс с signal_setup_done().