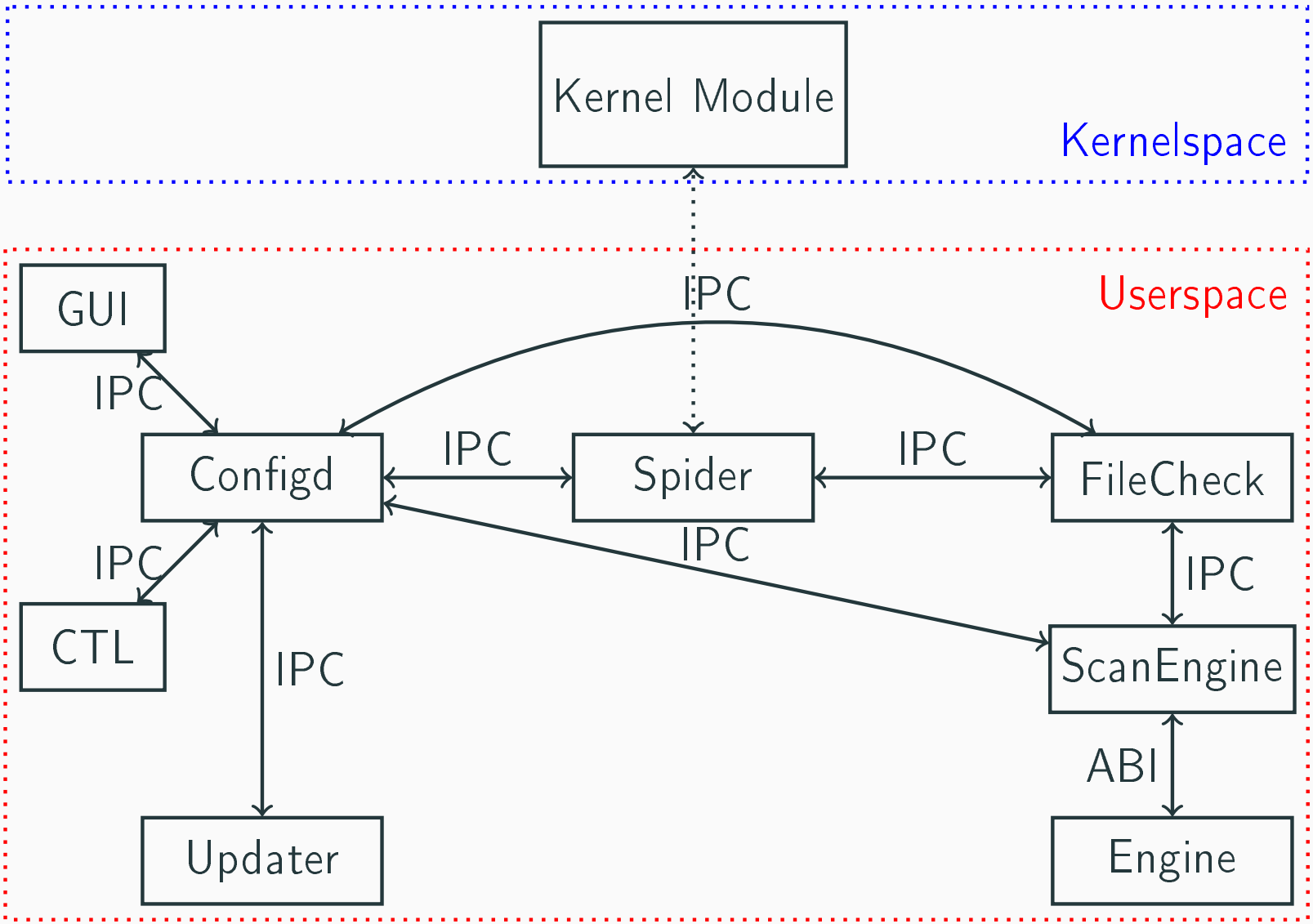
Разделяемые библиотеки без внешних зависимостей

Дмитрий Банщиков
me@ubique.spb.ru

17 апреля 2018 г.

Архитектура антивируса



Что нужно сделать для портирования?

- Собрать подходящим компилятором портабельные компоненты

Что нужно сделать для портирования?

- Собрать подходящим компилятором портабельные компоненты
- Написать специфичный для архитектуры/платформы код

Что нужно сделать для портирования?

- Собрать подходящим компилятором портабельные компоненты
- Написать специфичный для архитектуры/платформы код
- Портировать антивирусное ядро

Почему антивирусное ядро тяжело портировать?

- Связка C и assembler

Почему антивирусное ядро тяжело портировать?

- Связка C и assembler
- Жесткая привязка к ILP32

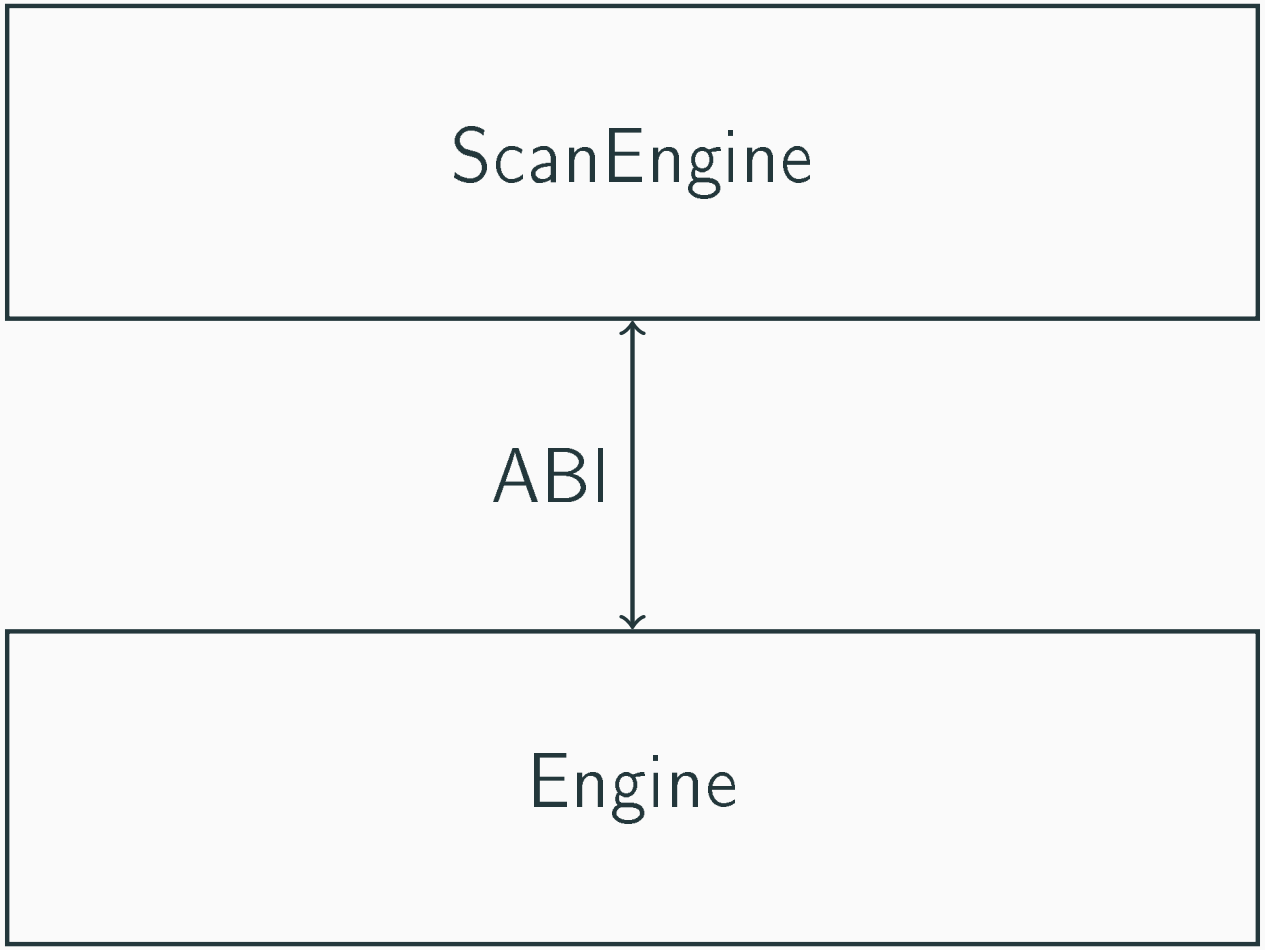
Почему антивирусное ядро тяжело портировать?

- Связка C и assembler
- Жесткая привязка к ILP32
- Misaligned access

Почему антивирусное ядро тяжело портировать?

- Связка C и assembler
- Жесткая привязка к ILP32
- Misaligned access
- Большая кодовая база

Взаимодействие сканера и антивирусного ядра



Engine ABI

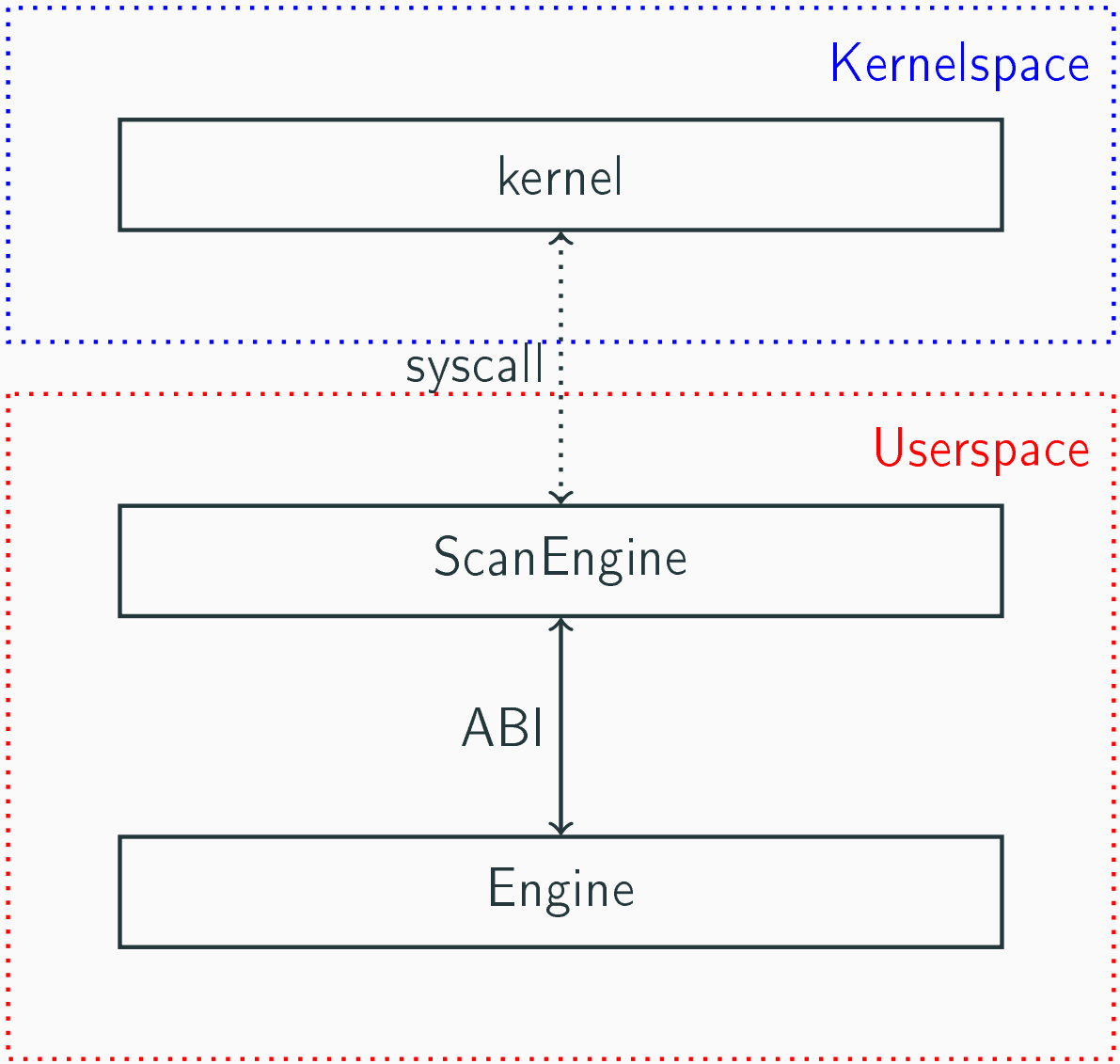
```
struct shell_t {
  uintptr_t (*dw_malloc)(uintptr_t size);
  uintptr_t (*dw_free)(uintptr_t addr);
  uintptr_t (*dw_open)(uintptr_t path, uintptr_t mode);
  uintptr_t (*dw_close)(uintptr_t fd);
  [...]
};

uinptr_t engine_init(shell_t* shell, ...);
```

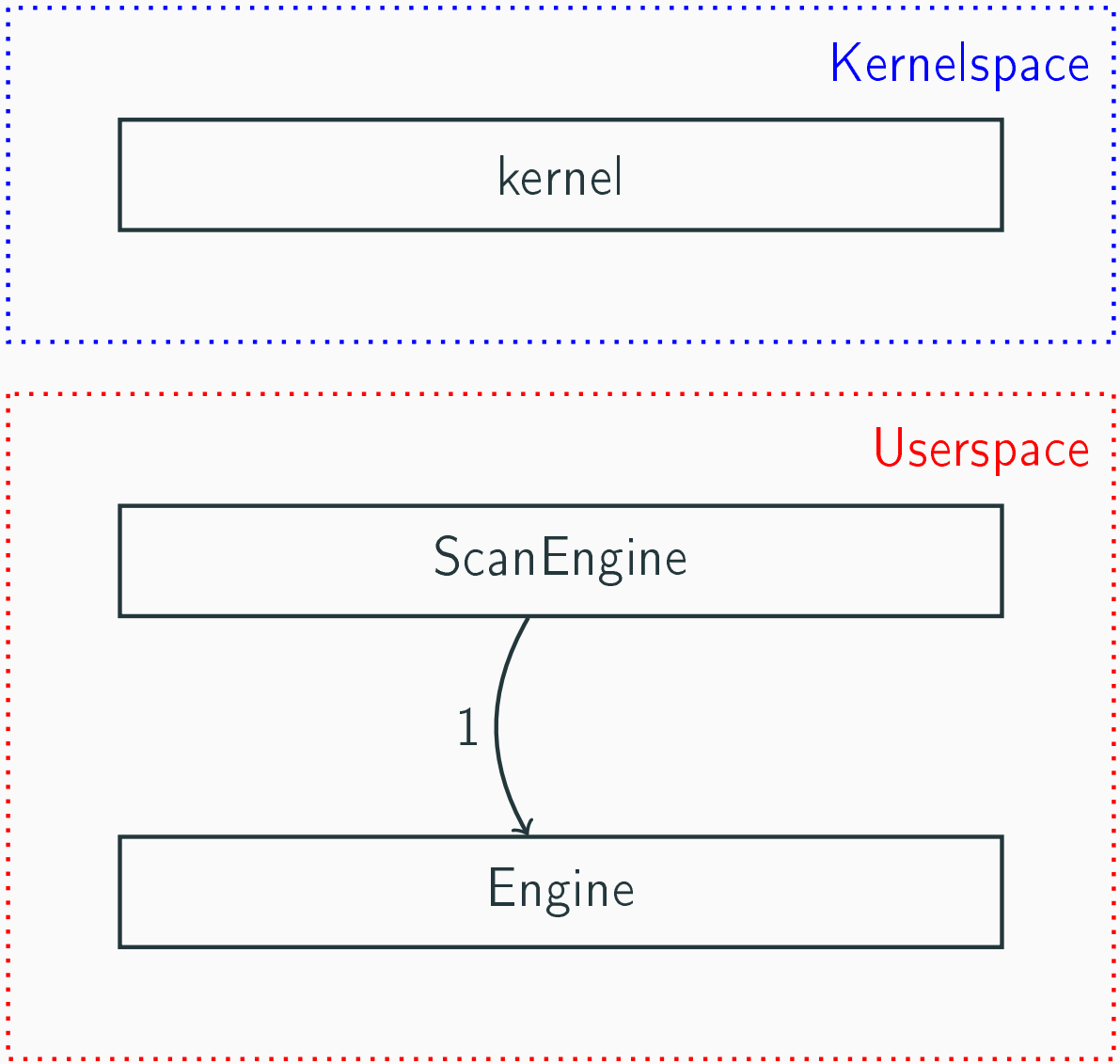
Engine ABI

```
using entry point_t = uintptr_t(uintptr_t call,
                                uintptr_t arg1,
                                uintptr_t arg2,
                                ...);
entry_point_t* entry_point = engine_init(shell, ...);
```

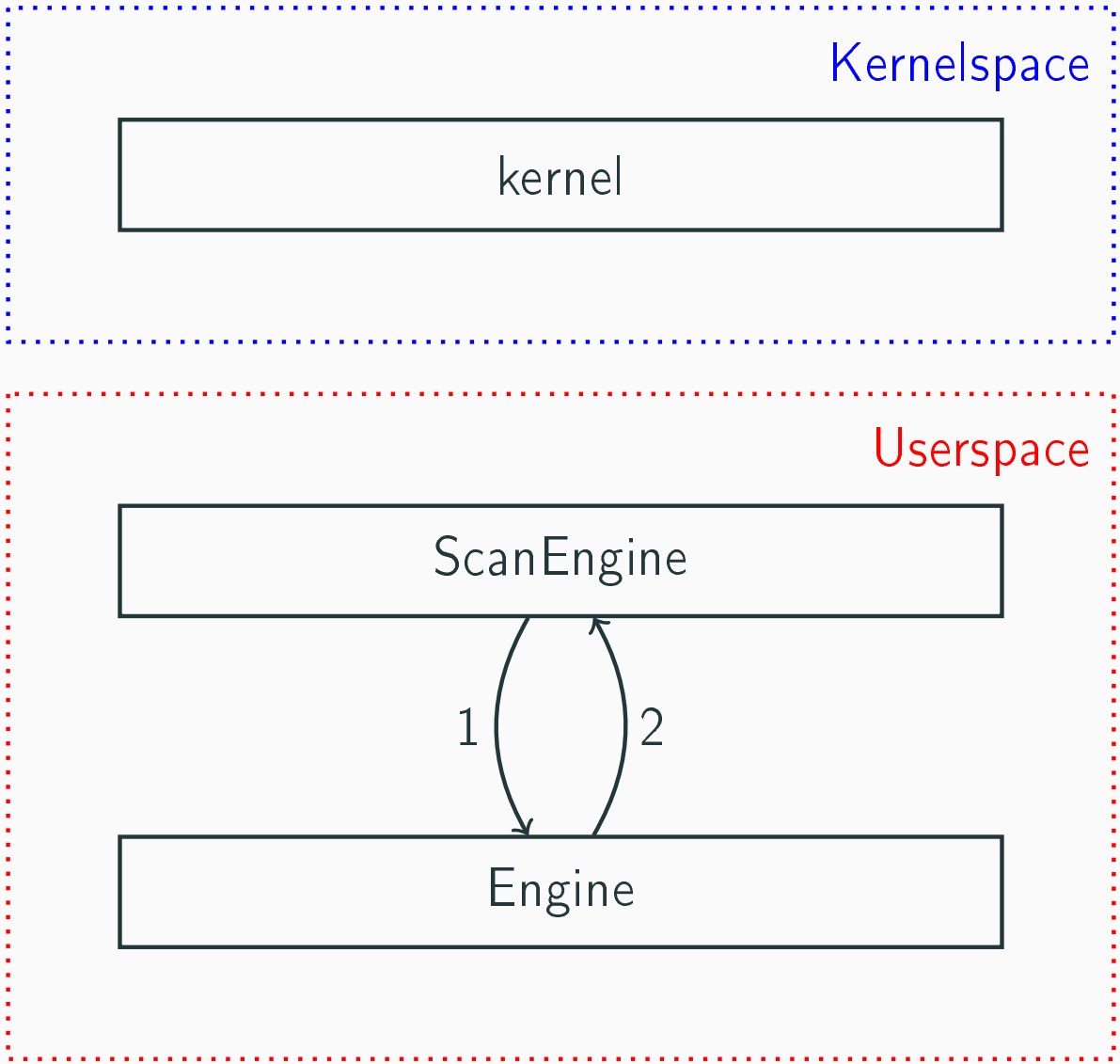
Взаимодействие сканера и антивирусного ядра



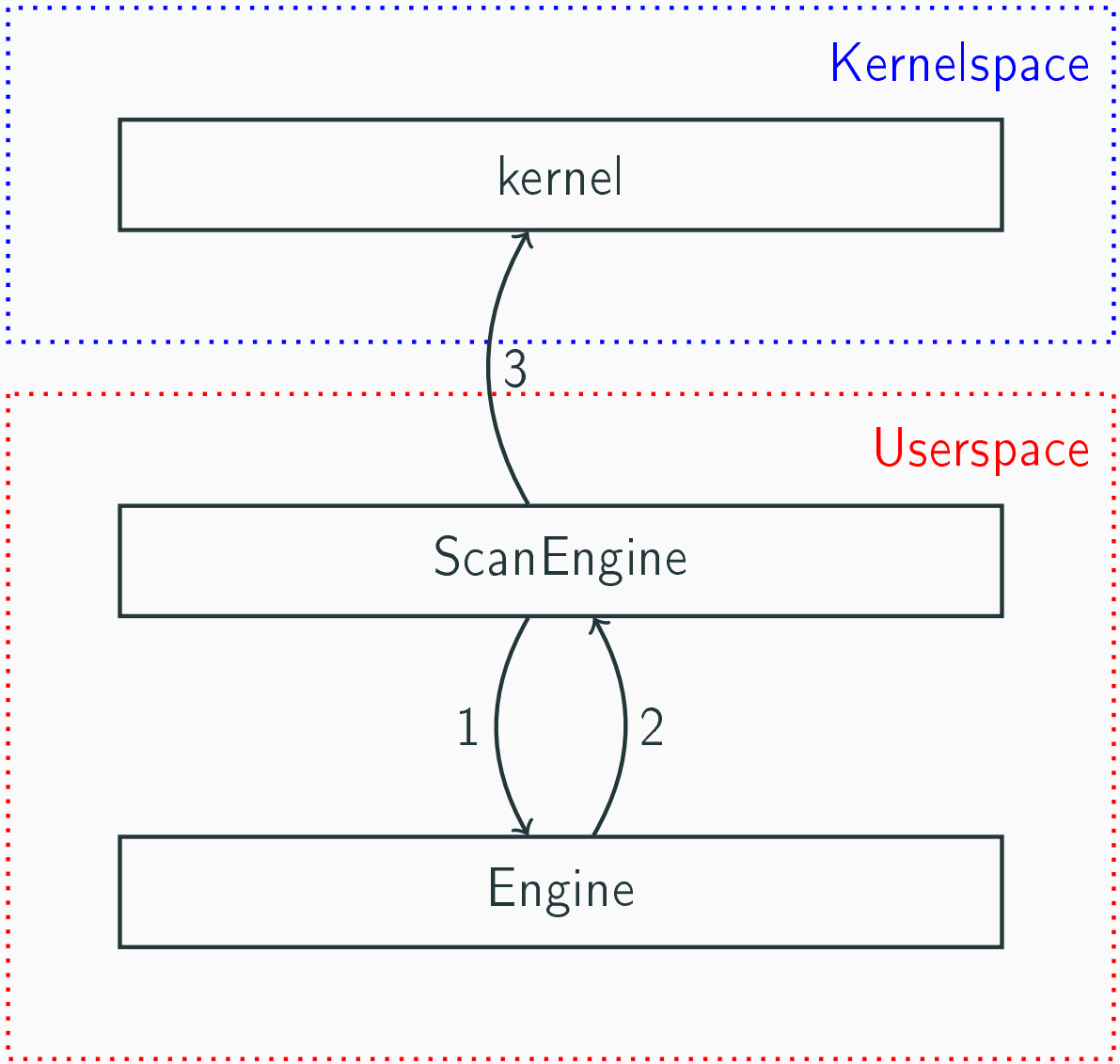
Взаимодействие сканера и антивирусного ядра



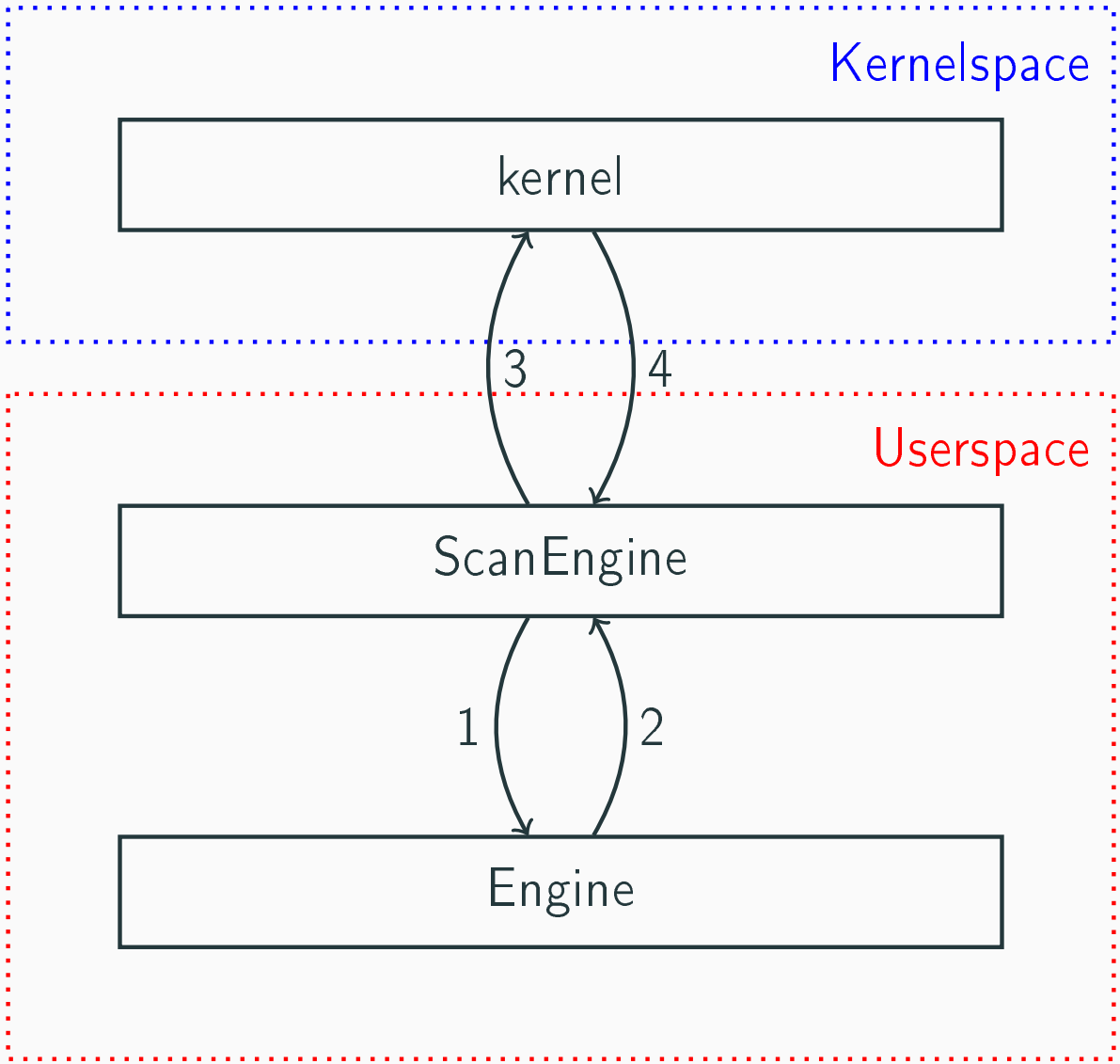
Взаимодействие сканера и антивирусного ядра



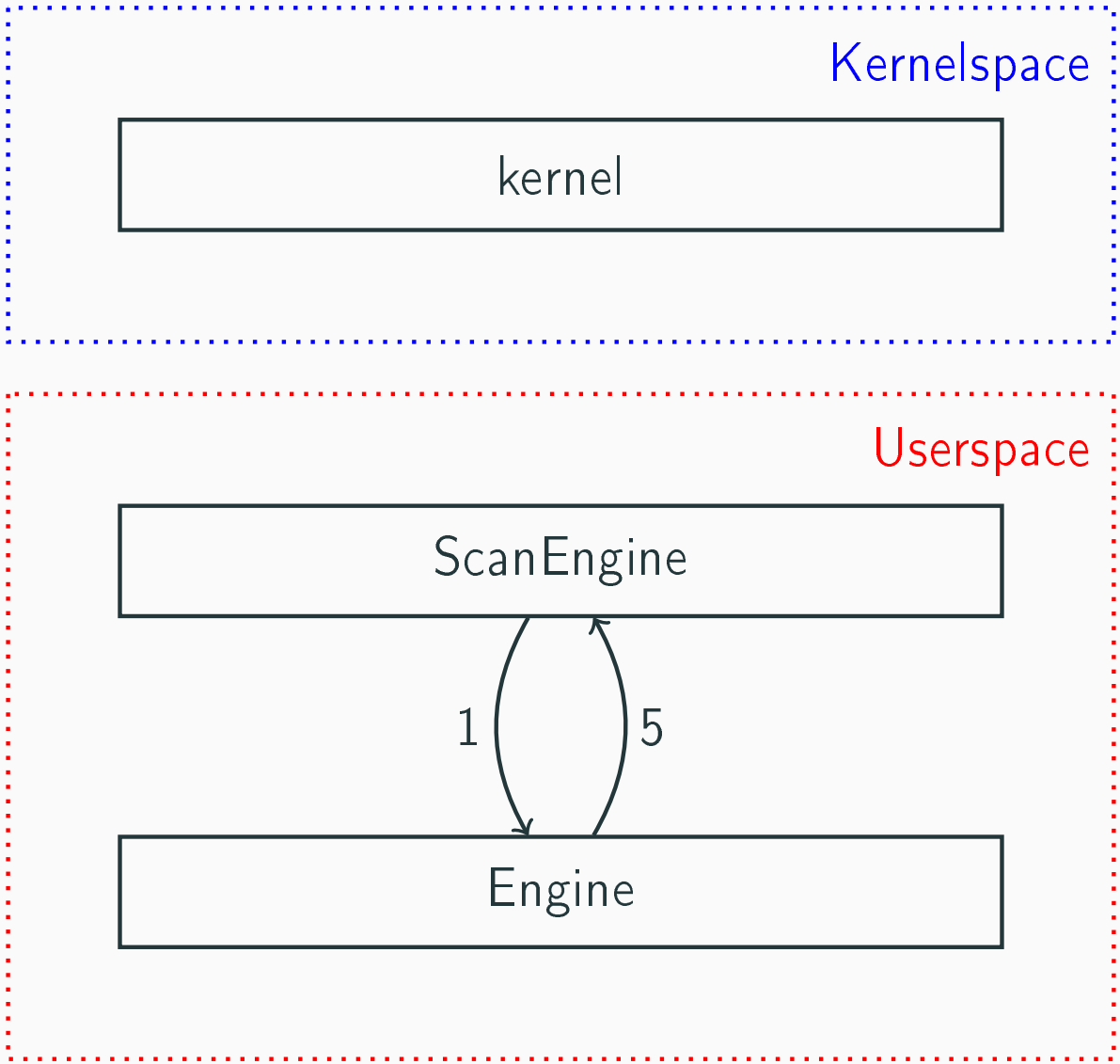
Взаимодействие сканера и антивирусного ядра



Взаимодействие сканера и антивирусного ядра



Взаимодействие сканера и антивирусного ядра



Прочие проблемы

- Прорастание SHELL в код ядра

Прочие проблемы

- Прорастание SHELL в код ядра
- Невозможность смены SHELL

Прочие проблемы

- Прорастание SHELL в код ядра
- Невозможность смены SHELL
- Сложность использования сторонних библиотек

Прочие проблемы

- Прорастание SHELL в код ядра
- Невозможность смены SHELL
- Сложность использования сторонних библиотек
- Неудобное UNIT тестирование

Чего мы хотим?

- Решить проблемы с портированием

Чего мы хотим?

- Решить проблемы с портированием
- Решить проблемы с прорастанием SHELL в ядро

Чего мы хотим?

- Решить проблемы с портированием
- Решить проблемы с прорастанием SHELL в ядро
- Иметь возможность смены SHELL

Чего мы хотим?

- Решить проблемы с портированием
- Решить проблемы с прорастанием SHELL в ядро
- Иметь возможность смены SHELL
- Использовать полноценный C++

Чего мы хотим?

- Решить проблемы с портированием
- Решить проблемы с прорастанием SHELL в ядро
- Иметь возможность смены SHELL
- Использовать полноценный C++
- Возможность использования сторонних библиотек

Чего мы хотим?

- Решить проблемы с портированием
- Решить проблемы с прорастанием SHELL в ядро
- Иметь возможность смены SHELL
- Использовать полноценный C++
- Возможность использования сторонних библиотек
- Удобное UNIT тестирование

Что нужно сохранить?

ABI

NewLib

- Библиотека C

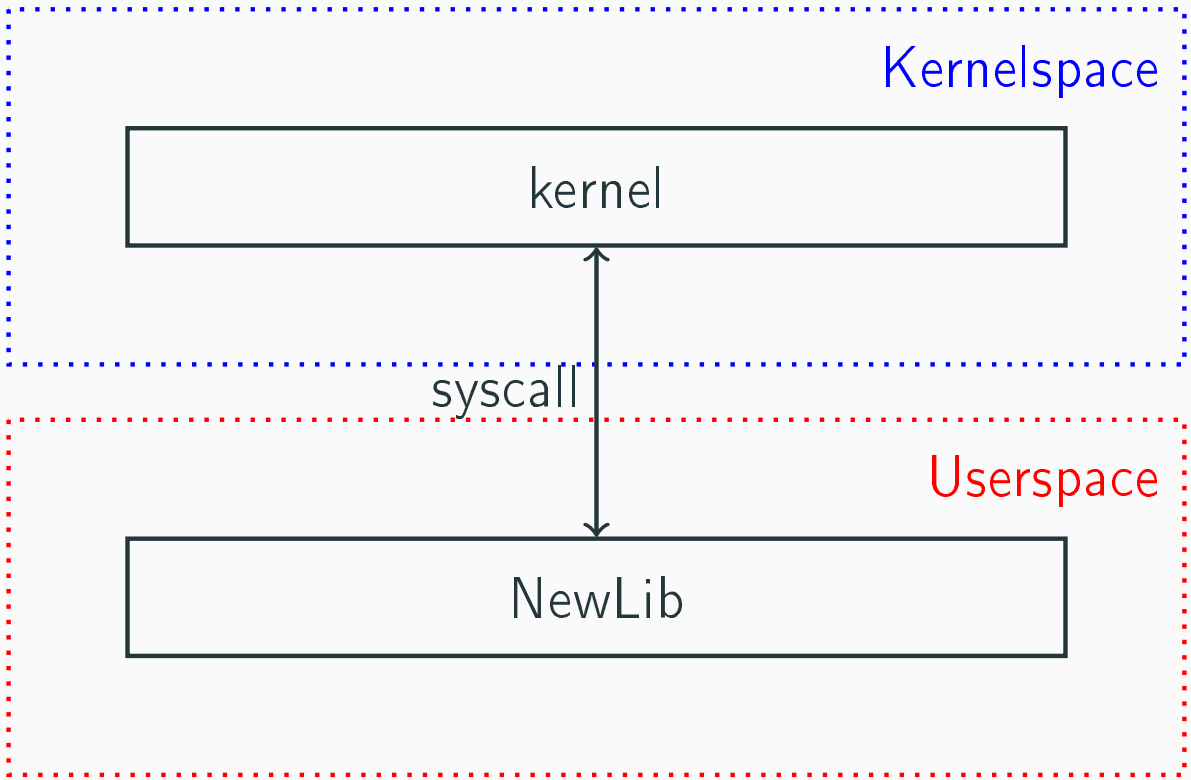
NewLib

- Библиотека C
- Ориентация на embedded

NewLib

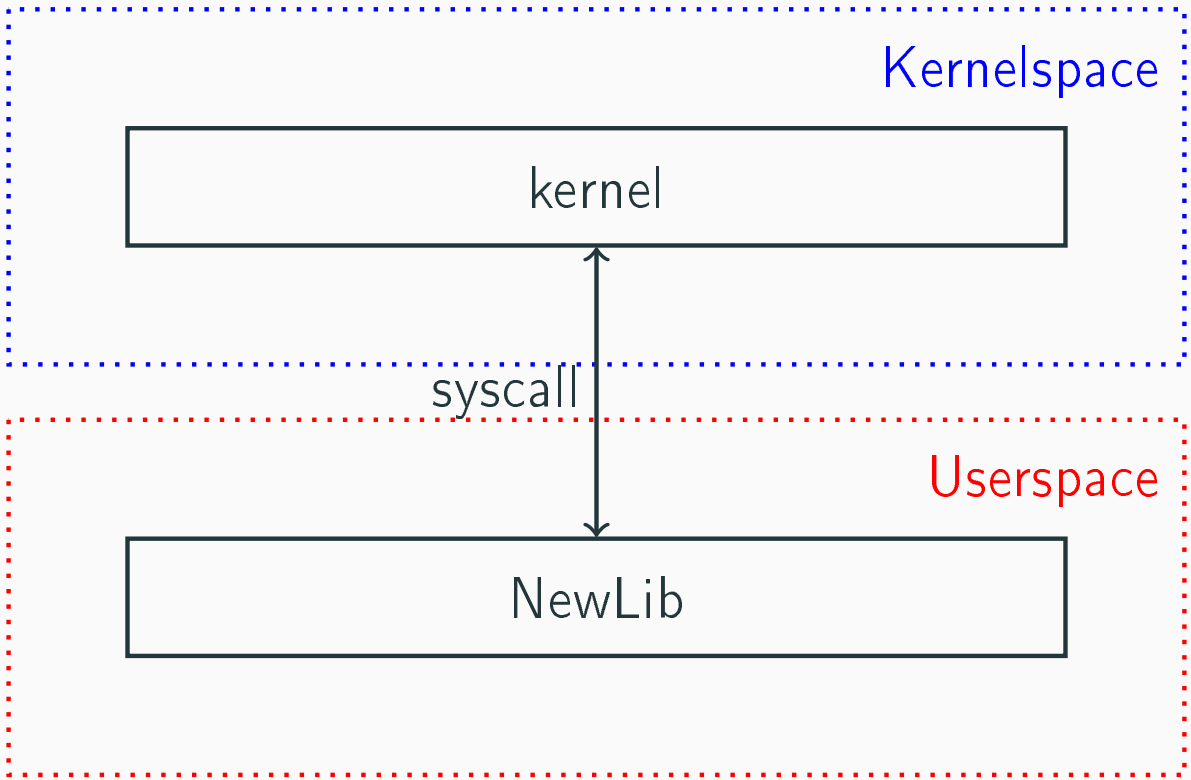
- Библиотека C
- Ориентация на embedded
- Production ready

NewLib



- Портируемость

NewLib



- Портируемость
- Необходимые системные вызовы: open, close, read, write, fork, etc.

NewLib

```
int fstat(int fd, struct stat* st)
{
    if(shell) {
        const off_t size = shell->dw_get_file_size(0, fd);
        if(size == -1) {
            errno = EINVAL;
            return -1;
        }

        st->st_mode = S_IFCHR;
        st->st_size = size;
    }

    errno = EAGAIN;

    return -1;
}
```

NewLib

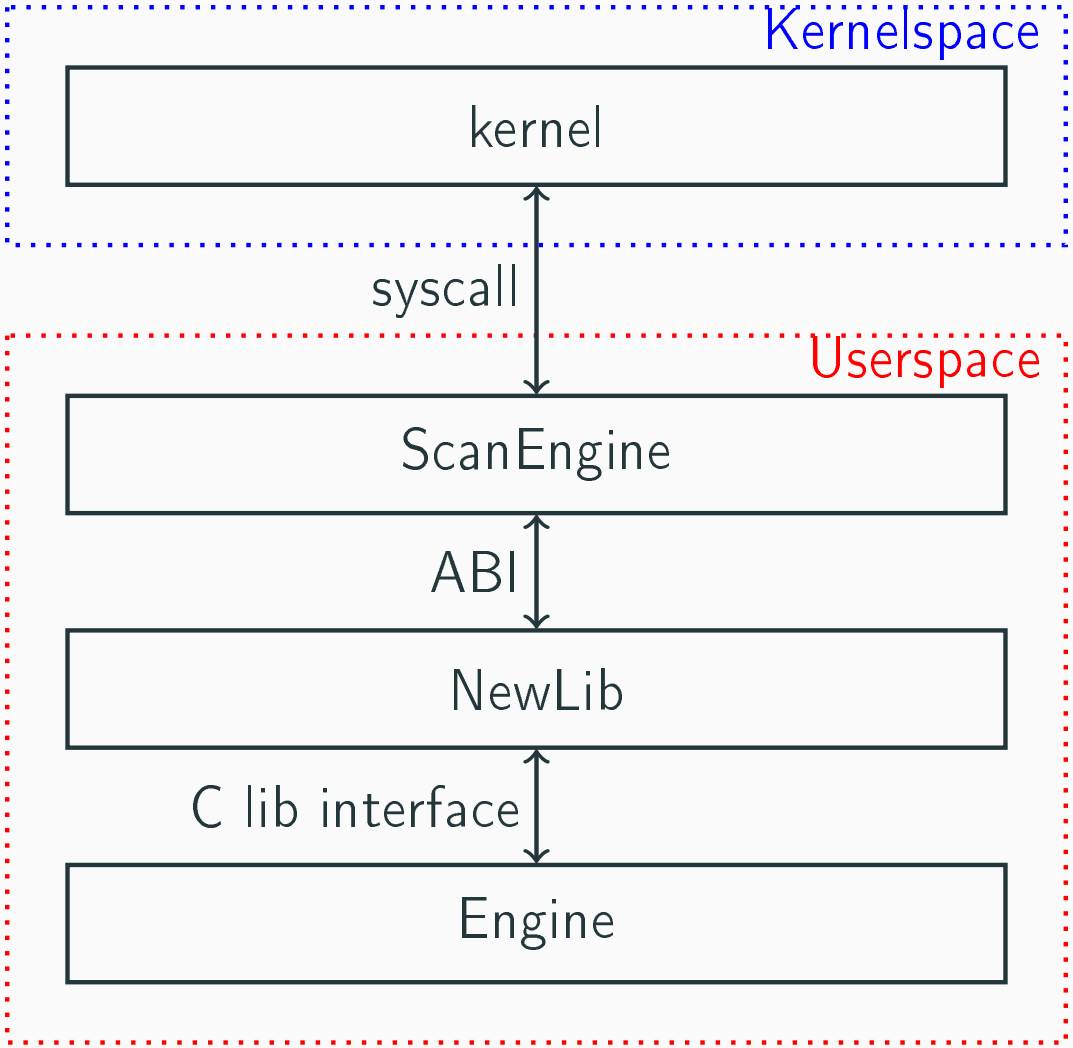
```
pid_t fork(void)
{
    errno = ENOSYS;

    return -1;
}
```

NewLib

```
void func() {
  const size_t memory_size = 42;
  char* memory = malloc(memory_size);
  if(memory) {
    snprintf(memory, memory_size, "Don't␣panic!");
    free(memory);
  }
}
```

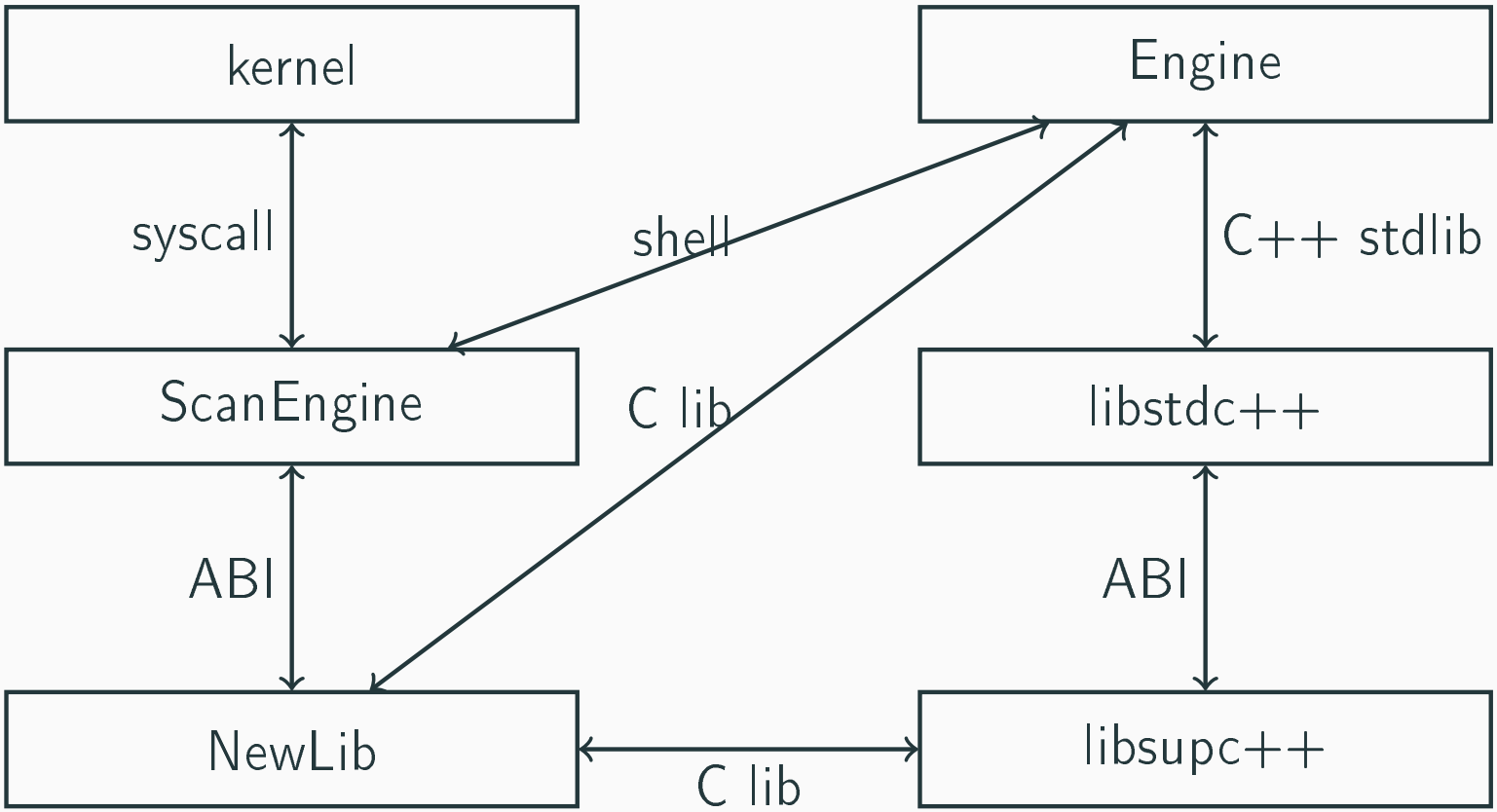
ScanEngine + NewLib + Engine



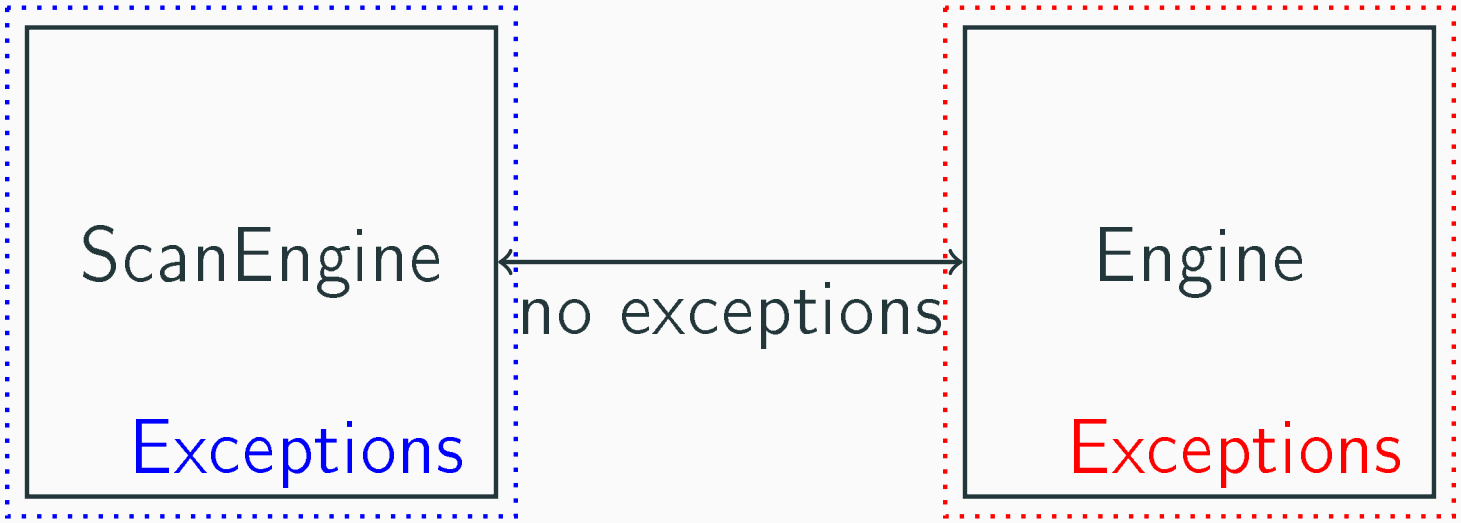
ScanEngine + NewLib + libsupc++ + libstdc++ + Engine

```
void func() {
  std::vector<int> vector(42);
  std::iota(vector.begin(), vector.end(), 0);
}
```

ScanEngine + NewLib + libsupc++ + libstdc++ Engine



Exceptions



Статистика

- Оверхед по размеру - 1 MB

Статистика

- Оверхед по размеру - 1 MB
- Оверхед по размеру с оптимизациями - 650KB

Выводы

- Сохранение ABI

Выводы

- Сохранение ABI
- Портируемость - нужен только подходящий компилятор

Выводы

- Сохранение ABI
- Портируемость - нужен только подходящий компилятор
- Возможность использовать C++ без ограничений

Выводы

- Сохранение ABI
- Портируемость - нужен только подходящий компилятор
- Возможность использовать C++ без ограничений
- Возможность смены SHELL

Выводы

- Сохранение ABI
- Портируемость - нужен только подходящий компилятор
- Возможность использовать C++ без ограничений
- Возможность смены SHELL
- Возможность использования сторонних библиотек без модификаций

Выводы

- Сохранение ABI
- Портируемость - нужен только подходящий компилятор
- Возможность использовать C++ без ограничений
- Возможность смены SHELL
- Возможность использования сторонних библиотек без модификаций
- Удобное UNIT тестирование

Вопросы?