C++ CoreHard Autumn 2017

Actors for fun and profit

Евгений Охотников

stiffstream

Предистория

C++ CoreHard Autumn 2016:

Модель акторов и C++: что, зачем и как?

C++ CoreHard Winter 2017:

Шишки, набитые за 15 лет использования акторов в C++.

Модель Акторов и C++:
миф или реальность?

Реальность!

- промышленная автоматизация (АСУ ТП);
- телеком;
- электронная и мобильная коммерция;
- имитационное моделирование;
- САПР;
- игростроение;
- ПО промежуточного слоя (СУБД, ...)
- ...

Есть готовые инструменты:

Наиболее известные:

- QP/C++ (двойная лицензия);
- Just::Thread Pro: Actors Edition (коммерческая лицензия);
- C++ Actor Framework (BSD-3-Clause лицензия);
- SObjectizer (BSD-3-Clause лицензия);

Кроме того:

- OOSMOS (двойная лицензия, язык C, но может использоваться и в C++);
- Asyncronous Agents Library (входит в Visual Studio);

Вопрос на миллион:

А нужна ли вообще Модель Акторов?

Нужен ли самосвал на трассе Формулы-1?

Архиважное уточнение:

Для чего?

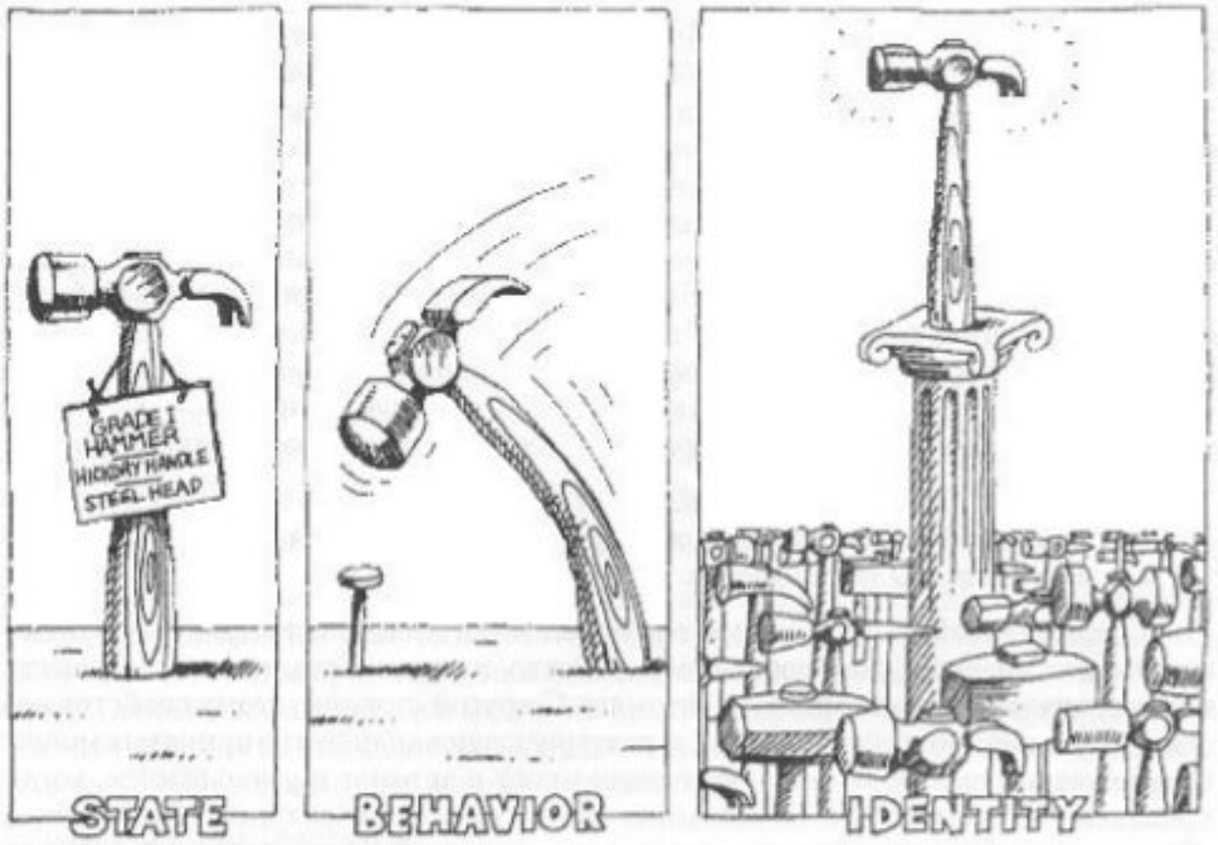
Однако!

"Эксперименты новичков должны проводиться в лабораториях для новичков"

Из книги "Вы, конечно, шутите, мистер Фейман!"

Модель Акторов как...

...способ смотреть на мир. Другими словами: если у вас в руках молоток...



Основные принципы Модели Акторов

- актор – это некая сущность, обладающая поведением;
- акторы реагируют на входящие сообщения;
- получив сообщение актор может:
  - отослать некоторое количество сообщений другим акторам;
  - создать некоторое количество новых акторов;
  - определить для себя новое поведение для обработки последующих сообщений.

Молоток и гвозди – это понятно, но...

...далеко не всегда они в принципе нужны.

Яркий пример: расчетные задачи из вычислительной математики.

Лакмусовые бумажки

Есть несколько маркеров, которые могут подсказать, что выбор Модели Акторов будет оправданным.

Могут подсказать. А могут и не подсказать.

Маркер №1

Fire-and-Forget

Fire-and-Forget

1. Тотальный контроль не нужен.
2. Результаты начатых операций вот прямо здесь и сейчас не нужны.
3. Если что-то не случилось, то ничего страшного.

Мы постоянно используем этот принцип в повседневной жизни

Fire-and-Forget: примеры "за" и "против"

Пример №1: две рабочие нити, одна только читает данные, вторая только разбирает их.

Пример №2: коммит в БД.

Fire-and-Forget. Резюме.

Если fire-and-forget для вашей задачи естественен и повсеместен, то в сторону Модели Акторов можно смотреть.

Если же превалируют блокирующие синхронные операции, то с применением Модели Акторов могут быть сложности.

Маркер №2

Конечные автоматы

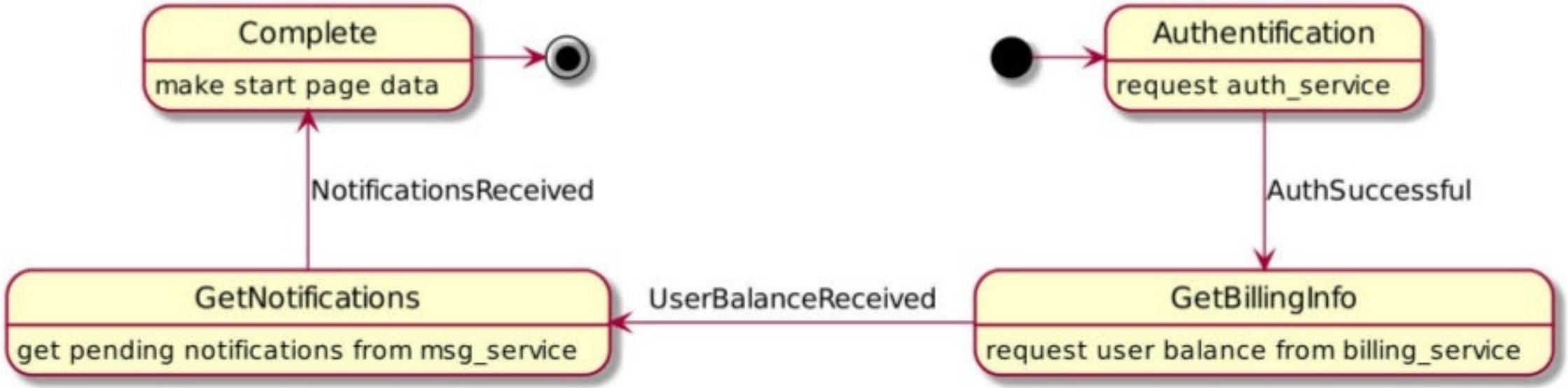
Принципы Модели Акторов еще раз

- **актор – это некая сущность, обладающая поведением**;
- акторы реагируют на входящие сообщения;
- **получив сообщение актор может**:
  - отослать некоторое количество сообщений другим акторам;
  - создать некоторое количество новых акторов;
  - **определить для себя новое поведение для обработки последующих сообщений**.

Конечные автоматы

Не все ~~йогурты~~ конечные автоматы одинаково полезны

Пример: вход в он-лайн кинотеатр



Простой синхронный код лучше

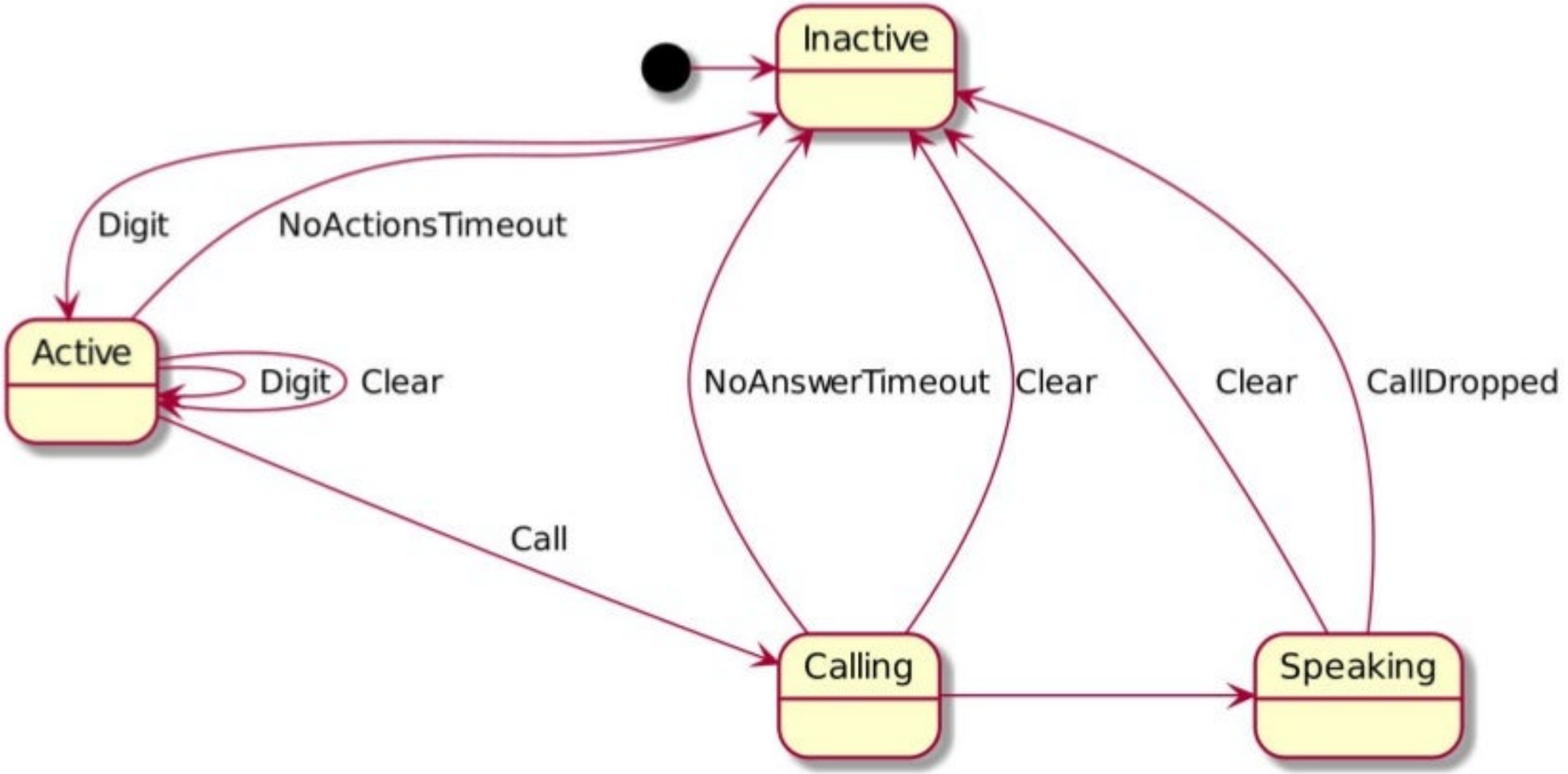
```
auto process_login(const login_params & params) -> start_page_data
{
  const auto auth_result = request_auth_service(params);
  if(auth_result.valid_user())
  {
    const auto balance = request_balance(auth_result.user_token());
    const auto pending_messages = request_pending_messages(auth_result.user_token());
    return make_start_page_data(auth_result, balance, pending_messages);
  }
  else
    return make_unknown_user_page_data();
}
```

Обычные потоки (или сопрограммы) + CSP или task based parallelism.

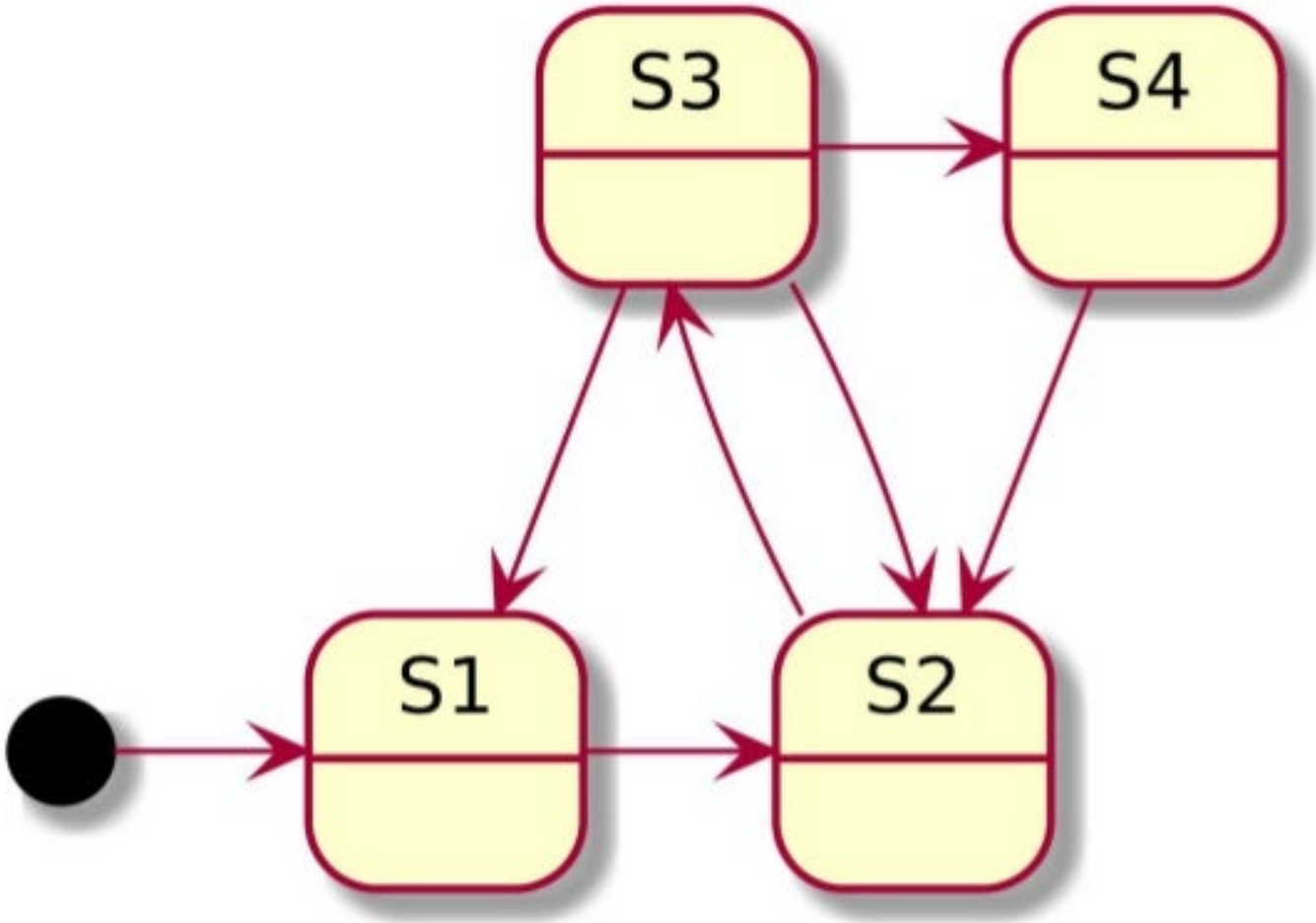
Конечные автоматы

Есть и полезные конечные автоматы

Полезные конечные автоматы (1)



Полезные конечные автоматы (2)



Полезные конечные автоматы (3)

Продвинутые возможности конечных автоматов:

- реакция на вход/выход в/из состояния;
- иерархия состояний (вложенные состояния, наследование событий);
- история для состояний;
- ограничения на время пребывания в состоянии.

Для тех, кто хочет приобщиться:

David Harel

Statecharts: A Visual Formalism For Complex Systems

(1987)

Конечные автоматы. Резюме.

Если ваша предметная область кишит конечными автоматами, то Модель Акторов может вам помочь.

Маркер №3

Архитектура Shared Nothing

Shared Nothing: контрольный в голову

Могут ли сущности в программе работать без каких либо разделяемых данных?

Может ли каждая сущность быть представлена в виде отдельного процесса?

Это таки экстремизм.
Но он отрезвляет.

Очевидно, что...

...архитектура Shared Nothing не всегда применима

Кроме того...

...все зависит от высоты, с которой мы смотрим вниз
(уровень абстракции имеет значение)

Shared Nothing. Резюме.

Если использование архитектуры Shared Nothing затруднительно и/или ведет к дополнительным накладным расходам, то в сторону Модели Акторов можно не смотреть.

Но вообще Shared Nothing – это отличная вещь. Очень сильно упрощает жизнь. Особенно в многопоточном программировании.

Модель Акторов способствует внедрению архитектуры Shared Nothing.

Не маркер, но тем не менее

Таймеры

Напрямую не связано с Моделью Акторов, но...

...таймеры в виде отложенных сообщений – это очень удобно на практике.

Следствие того, что акторы общаются с внешним миром только посредством сообщений.

Таймеры активно используются акторами из-за принципа Fire-and-Forget.

Пример: ожидание с ограничением по времени

Нужно накапливать запросы до тех пор, пока:

- поступит 100 штук (обрабатываем 100 запросов разом) или
- истечет тайм-аут в 250ms (обрабатываем сколько пришло).

Пример: ожидание с ограничением по времени

```
class bunch_processor {
    ...
public:
    void on_request(request & req) {
        requests_.push_back(move(req));
        if(1 == requests_.size())
            timeout_timer_ = send_delayed<timeout>(this, 250ms);
        else if(100 == requests_.size()) {
            timeout_timer_.reset();
            handle_collected_requests();
        }
    }
    void on_timer(timeout&) {
        handle_collected_requests();
    }
  ...
};
```

Таймеры. Резюме.

Если у вас много таймеров и с ними нужно активно работать, то Модель Акторов может упростить вашу работу.

Хватит сферических коней в вакууме!

Где это применяется?

Где это применяется?

Управление оборудованием

Где это применяется?

Имитационное моделирование

Где это применяется?

Тестовые стенды

Где это применяется?

Конвейерная обработка данных/транзакций

Ваш К.О.

1. Здравый смысл никто не заменит, здравый смысл незаменим!
2. Модель Акторов не является серебряной пулей.
3. В ряде случаев Модель Акторов упрощает жизнь.
4. Но нужно уметь ее использовать.
5. Учиться лучше "на кошках".
6. Есть готовые инструменты, которые позволяют учиться без велосипедостроения.

Хочешь акторов? Спроси меня как!

Абсолютно бесплатно и без SMS ;)

Все подробности здесь: http://eao197.blogspot.com/2017/03/progactors.html