Проектирование больших систем на C++

Коноводов В. А.

кафедра математической кибернетики ВМК
vkonovodov@gmail.com

Лекция 11
28.11.2020

Readability, Correctness, Efficiency

- Хорошо ли написан код?
- Корректно ли написан код?
- Эффективно ли работает код?
- Тестирование
- Формальная верификация (автоматическая/полу-автоматическая)
- ...

Тестирование

- Цель тестирования кода?
- Как искать баги в коде и упростить себе жизнь?
  - Хороший дизайн кода и code review
  - unit-тестирование
  - Интеграционное тестирование
  - UI-тесты
  - Мониторинги f
  - Acceptance тестирование

Тестирование

Unit-тесты (модульное тестирование):

- Быстрые
- Полные для своего компонента

Интеграционные тесты:

- Медленные
- Проверить всё невозможно
- Могут чаще флапать

Unit-тестирование

Задача: добавляем private-метод в класс с 100500 методами и полями. Нужно протестировать этот метод.

Behavior and State Based Testing

- Тесты, проверяющие что метод или функция отработали корректно, проверяют состояния объекта после вызова.
- Тесты на корректность самого взаимодействия объектов с другими объектами (а не результата).

gtest/gmock

Google C++ Testing Framework.

- Открытая библиотека для unit-тестирования.
- Ключевое понятие — assert
- Результат выполнения:
  - success
  - nonfatal failure
  - fatal failure
- Тест — набор assert'ов
- Набор тестов — тестовая программа или testing suite

gtest

```
ASSERT_TRUE(condition);
ASSERT_FALSE(condition);
ASSERT_EQ(val1, val2);
ASSERT_NE(val1, val2);
ASSERT_LT(val1, val2);
ASSERT_LE(val1, val2);
ASSERT_GT(val1, val2);
ASSERT_GE(val1, val2);
ASSERT_FLOAT_EQ(val1, val2);
ASSERT_DOUBLE_EQ(val1, val2);
ASSERT_NEAR(val1, val2, abs_error);

// strings
ASSERT_STREQ(str1, str2);
ASSERT_STRNE(str1, str2);
ASSERT_STRCASEEQ(str1, str2);
ASSERT_STRCASENE(str1, str2);
```

gtest

```
// exceptions
ASSERT_THROW(statement, exception_type);
ASSERT_ANY_THROW(statement);
ASSERT_NO_THROW(statement);
```

Можно добавлять message:

```
ASSERT_EQ(1, 2) << "i dont know why, but 1 != 2";
```

- `TEST` — макрос для определения теста
- `RUN_ALL_TESTS()` — запуск всех тестов

Тестирование и зависимости

- Зависимости классов
- Как обеспечить изоляцию?

Основные подходы:

- Dummy. Передаются в тестируемые классы/методы/функции в виде параметра без поведения, внутри как правило с ним ничего не происходит
- Stub. Подменяется внешняя зависимость, игнорируются все данные, входящие в stub из тестируемого объекта.
- Fake. Замена легковесной реализацией.
- Mock. Объекты, которые имитируют поведение реальных. Полезно, когда настоящие объекты непрактично/невозможно вставлять в юнит-тест, но поведение нужно сохранить.

EXPECT_CALL: пример

```
EXPECT_CALL(db, func("string", _))
  .WillOnce(DoAll(SetArgReferee<1>("abc"), Return(false)));
```

- Метод `func` должен быть вызван 1 раз
- Ему будет передан аргумент `"string"`
- Что передадут в качестве второго аргумента — не важно
- Метод сделает второй аргумент равным `"abc"`
- Метод вернет `false`

Тестирование

1. Не стоит использовать `EXPECT_TRUE`
2. Добавляйте больше контекста
3. Код тестов — это тоже код

Тестирование

Тест упал, что делать дальше?

Основные подходы:

1. tracing и logging

```
cerr << "loading data: started" << endl;
```

2. strace и похожие утилиты
3. gdb и похожие отладчики
4. asan и другие санитайзеры

Debug

Пусть мы хотим залогировать все создания объекта класса A.

```
class A {
    A() { std::cerr << "A::A()" << std::endl;}
};
```

Класс B.

```
class B {
    B() { std::cerr << "B::B()" << std::endl;}
};
```

Некрасиво.

Debug: еще варианты

```
std::cerr << __FILE__ << " " << __LINE__ << std::endl;
```

Макросы, которые в момент компиляции раскрываются в имя файла и номер строчки.

strace

ОС — средство взаимодействия программы с внешним миром (файлы, соединения, ...)
strace is a diagnostic, debugging and instructional userspace utility for Linux. It is used to monitor and tamper with interactions between processes and the Linux kernel, which include system calls, signal deliveries, and changes of process state.
`strace ./you_binary`
Получаем все системные вызовы, предоставленные операционной системой