Проектирование больших систем на C++

Коноводов В. А.

кафедра математической кибернетики ВМК

Лекция 12
30.11.2018

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

Fixtures

```
class A {
    int field;
  public:
    A(int k = 0) : field(k) { std::cout << "A()" << std::endl;}
    ~A() { std::cout << "~A()" << std::endl;}
    int GetField() const { return field;}
};

struct TestingWF : public ::testing::Test {
    A a;
};

TEST_F(TestingWF, TestA) {
    EXPECT_EQ(0, a.GetField());
}

int main(int argc, char** argv) {
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

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

GDB

- **r**un – start the debugged program
- **l**ist – list specified function or line
- **b**reak – set breakpoint
- **cat**ch – set catchpoint (exception breakpoint)
- **i**nfo – show information about the debugged program
- **s**tep – step program, steps into functions
- **n**ext – step program, steps over function calls
- **s**tep**i**, **n**ext**i** – step by instructions, not lines of code
- **p**rint – evaluate expression
- e**x**amine – display contents of memory address
- **c**ontinue – continue running (after breakpoint)
- **k**ill – stop execution of the program
- **b**acktrace – print backtrace of stack frames
- **up**, **do**wn, **f**rame, **sel**ect-frame – select stack frame
- ...