

# *Задание №1 в рамках вычислительного практикума*

## **Автоматизация функционального тестирования. Аргументы командной строки приложения**

Кострицкий А. С., Ломовской И. В.

Москва — 2023 — TS2305131050

### **Содержание**

<b>1</b>	<b>Цель работы</b>	<b>1</b>
<b>2</b>	<b>Задание</b>	<b>1</b>
<b>3</b>	<b>Описание скриптов</b>	<b>2</b>
3.1	Правила взаимодействия . . . . .	2
3.2	Примечания . . . . .	4
<b>4</b>	<b>Формат защиты</b>	<b>4</b>

## **1 Цель работы**

*Целью* данной работы является автоматизация процессов сборки и тестирования.

## **2 Задание**

Требуется:

1. Реализовать скрипты отладочной и релизной сборки.
2. Реализовать скрипты отладочной сборки с санитайзерами.
3. Реализовать скрипт очистки побочных файлов.
4. Реализовать компаратор для сравнения содержимого двух текстовых файлов.
5. Реализовать скрипт `pos_case.sh` для проверки позитивного тестового случая по определённым далее правилам.
6. Реализовать скрипт `neg_case.sh` для проверки негативного тестового случая по определённым далее правилам.
7. Обеспечить автоматизацию функционального тестирования.

## 3 Описание скриптов

### 3.1 Правила взаимодействия

Пример организации проекта:

```
/lab_00_00_00/  
  app.exe  
  main.c  
  main.o  
  build_debug.sh  
  build_debug_asan.sh  
  build_debug_msan.sh  
  build_debug_ubsan.sh  
  build_release.sh  
  collect_coverage.sh  
  check_scripts.sh  
  clean.sh  
  /func_tests/  
    readme.md  
    /scripts/  
      func_tests.sh  
      neg_case.sh  
      pos_case.sh  
      comparator.sh  
    /data/  
      pos_01_in.txt  
      pos_01_out.txt  
      pos_01_args.txt  
      pos_02_in.txt  
      ...  
      pos_05_in.txt  
      pos_05_out.txt  
      pos_05_args.txt  
      neg_01_in.txt  
      neg_01_args.txt  
      ...
```

В каталоге однофайлового проекта располагается исходных код программы в файле `main.c`. В этом же каталоге располагаются скрипты `build_debug.sh`, `build_release.sh`, с помощью которых автоматизируется создание отладочной и релизной сборок проекта. Внутри необходимо явно прописать команды для двух этапов сборки — компиляции и компоновки.

В каталоге проекта располагаются также любые другие скрипты, автоматизирующие какое-либо действие, например, `collect_coverage.sh` для автоматизации получения статистики полноты данных при тестировании. Разрешается создание дополнительных скриптов в каталоге проекта, являющихся надстройками над описанными скриптами.

Исполняемый файл приложения `app.exe`<sup>1</sup> и любые другие побочные файлы сборки создаются в каталоге проекта. Скрипт `clean.sh` очищает результаты сборки проекта и

---

<sup>1</sup>Да, с расширением `exe`!

любые другие побочные файлы.

В каталоге `func_tests/data` располагаются данные для функционального тестирования программы.

Позитивные входные данные следует располагать в файлах вида `pos_TT_in.txt`, выходные — в файлах вида `pos_TT_out.txt`, аргументы командной строки для приложения — в файлах вида `pos_TT_args.txt`, где `TT` — номер тестового случая.

Негативные входные данные следует располагать в файлах вида `neg_TT_in.txt`, выходные — в файлах вида `neg_TT_out.txt`, аргументы командной строки для приложения — в файлах вида `neg_TT_args.txt`, где `TT` — номер тестового случая.

В каталоге `func_tests` располагается файл `readme.md` с описанием тестовых случаев.

В каталоге `func_tests/scripts` располагаются скрипты для автоматизации выполнения функциональных тестов.

1. `comparator.sh` принимает два текстовых файла и проводит сравнение по определённому признаку. Как пример, это может быть компаратор, сравнивающий только целые числа в двух файлах, сравнивающий число букв «а», сравнивающий содержимое файлов только после подстроки «subst:», etc. Какой именно это компаратор — зависит от сути тестируемого проекта.
2. `pos_case.sh` принимает в качестве аргументов файл для подмены входного потока, файл эталонных выходных данных и файл аргументов командной строки, передаваемых в приложение.

```
pos_case.sh file_stream_in file_stream_out_expect file_args
```

- (a) Рабочей папкой скрипт считает папку, откуда был вызван.
  - (b) Скрипт ожидает от приложения нулевой код возврата в случае успеха.
  - (c) Скрипт по умолчанию работает в «молчаливом режиме», возвращает нуль при успешном тестировании, иначе — не нуль.
  - (d) Мусор после работы скрипта не очищается.
3. `neg_case.sh` принимает в качестве аргументов файл для подмены входного потока и файл аргументов командной строки, передаваемых в приложение.

```
neg_case.sh file_stream_in file_args
```

- (a) Рабочей папкой скрипт считает папку, откуда был вызван.
  - (b) Скрипт ожидает от приложения ненулевой код возврата в случае ошибки.
  - (c) Скрипт по умолчанию работает в «молчаливом режиме», возвращает нуль при успешном тестировании, иначе — не нуль.
  - (d) Мусор после работы скрипта не очищается.
4. `func_tests.sh` проводит функциональное тестирование с помощью `pos_case.sh` и `neg_case.sh`.
- (a) Рабочей папкой скрипт считает папку, откуда был вызван.
  - (b) Скрипт возвращает не нуль, если не был пройден хотя бы один тест. Советуем возвращать число проваленных тестов.
  - (c) Скрипт может работать не в «молчаливом режиме». Допускается любое читабельное оформление результатов тестирования при выводе в терминал.
  - (d) Скрипт проводит только функциональное тестирование, никакой автоматизации сборки и сбора статистики.

## 3.2 Примечания

1. Кодировкой всех файлов, включая файлы тестовых данных, считается UTF-8 с символом LF (Unix) в качестве символа новой строки.
2. Все скрипты должны быть написаны для работы в одной оболочке, например, bash или sh.
3. Все скрипты должны проходить проверку с помощью ShellCheck. Можно воспользоваться веб-версией или утилитой из репозитория. Разрешается написать скрипт для проверки других скриптов.
4. Если в программе используется в качестве входных данных бинарный файл, он не помещается под версионный контроль. Данные для тестирования должны быть преобразованы отдельно перед каждым тестом или перед проведением всех тестов. Рекомендуется добавить в тестируемую программу возможность запуска с ключами `t2b` и `b2t`.

## 4 Формат защиты

Студентом к дате основной защиты готовится страница отчёта в формате .docx (.odt), два файла .docx и .pdf в виде zip-архива прикрепляются к кафедральному moodle.

В отчёт следует поместить исходные коды всех скриптов с описанием назначения скриптов.