# **Ильченко Ева ИУ7-24Б**

# **Отчет по Заданию №1. Автоматизация функционального тестирования**

Цель: написать скрипты для автоматизации процессов тестирования и сборки

### 

### **Задание**

### Скрипты отладочной и релизной сборки

##### **build\_debug.sh**

Скрипт для отладочной сборки программы

#!/bin/bash

gcc-13 -c ./main.c -Werror -Wall -std=c99 -Wpedantic -O0

-Wextra -Wfloat-conversion -Wfloat-equal --coverage

gcc-13 --coverage ./main.o -o app.exe -lm

##### 

##### **build\_release.sh**

Скрипт для релизной сборки программы

#!/bin/bash

gcc-13 -c ./main.c -Werror -Wall -std=c99 -Wpedantic -O2 -Wextra -Wfloat-conversion -Wfloat-equal

gcc-13 ./main.o -o app.exe -lm

### Скрипты отладочной сборки с санитайзером

##### **build\_debug\_asan.sh**

Скрипт для реализации address sanitizer

#!/bin/bash

clang -c main.c -Werror -Wall -std=c99 -Wpedantic -O0 -Wextra -Wfloat-conversion -Wfloat-equal -fsanitize=address -fno-omit-frame-pointer -g

clang main.o -o app.exe -lm -fsanitize=address -static-libsan

##### 

##### **build\_debug\_msan.sh**

Скрипт для реализации memory sanitizer

#!/bin/bash

clang -c main.c -Werror -Wall -std=c99 -Wpedantic -O0 -Wextra -Wfloat-conversion -Wfloat-equal -fsanitize=memory -fno-omit-frame-pointer -g

clang main.o -o app.exe -lm -fsanitize=memory -fPIE -pie

##### 

##### **build\_debug\_ubsan.sh**

Скрипт для реализации undefined behavior sanitizer

#!/bin/bash

clang -c main.c -Werror -Wall -std=c99 -Wpedantic -O0 -Wextra -Wfloat-conversion -Wfloat-equal -fsanitize=undefined -fno-omit-frame-pointer -g

clang main.o -o app.exe -lm -fsanitize=undefined

### 

### Скрипт очистки побочных файлов

### **clean.sh**

#!/bin/bash

rm -f -- \*.o \*.exe ./func\_tests/data/temp.out \*.gcno \*.gcov \*.gcda "main"

### Компаратор для сравнения последовательностей действительных чисел, располагающихся в двух текстовых файлах, с игнорированием остального содержимого

### **comparator.sh**

На вход подаются 2 файла: файл с ожидаемым результатом работы программы и файл с действительным результатом работы программы. Фукнция get\_numbers() вытаскивает последовательность действительных чисел из файлов и далее они сравниваются. Если содержимое файлов одинаковое, то возвращается 0, иначе 1

#!/bin/bash

if [ $# -ne 2 ]; then

echo "Использование: comparator.sh файл1 файл2"

exit 1

fi

file1=$1

file2=$2

get\_numbers() {

grep -oE "[-+]?([0-9]+[.])?[0-9]" "$1"

}

f1=$(get\_numbers "$file1")

f2=$(get\_numbers "$file2")

if [ "$f1" != "$f2" ]; then

exit 1

fi

exit 0

1. Компаратор для сравнения содержимого двух текстовых файлов, располагающегося после первого вхождения подстроки «Result:␣».

**comparator.sh**

На вход подаются 2 файла: файл с ожидаемым результатом работы программы и файл с действительным результатом работы программы. Фукнция get\_numbers() вытаскивает строку из файлов после подстроки «Result:␣» и далее они сравниваются. Если содержимое файлов одинаковое, то возвращается 0, иначе 1

#!/bin/bash

if [ $# -ne 2 ]; then

echo "Использование: comparator.sh файл1 файл2"

exit 1

fi

file1=$1

file2=$2

get\_numbers() {

sed -n 's/. \* Result:\(. \* \)/\1/p' "$1"

}

f1=$(get\_numbers "$file1")

f2=$(get\_numbers "$file2")

if [ "$f1" != "$f2" ]; then

exit 1

fi

exit 0

### Скрипт для проверки позитивного тестового случая

##### **pos\_case.sh**

На вход подаются 2 файла: файл входных данных и файл ожидаемого результата работы программы. Входные данные подаются в программу и результат ее работы записывается в файл temp.out. Делается проверка, что код ошибки равен нулю. Далее файл с ожидаемым результатом работы программы и действительным результатом работы программы подаются в компаратор. Если содержимое файлов одинаковое, то возвращается 0, иначе 1

#!/bin/bash

if [ $# -ne 2 ]; then

echo "Использование: $0 файл\_входных\_данных файл\_ожидаемых\_выходных\_данных"

exit 1

fi

input\_file="$1"

expected\_output\_file="$2"

result\_file="./func\_tests/data/temp.out"

./app.exe < "$input\_file" > "$result\_file"

result=$?

if [ "$result" -ne 0 ]; then

exit 1

fi

if ./func\_tests/scripts/comparator.sh "$result\_file" "$expected\_output\_file"; then

exit 0

else

exit 1

fi

### Скрипт для проверки негативного тестового случая

##### **neg\_case.sh**

На вход подается файл с входными данными. Входные данные подаются в программу и результат ее работы записывается в файл temp.out. Если код выхода не равен нулю, то возвращается 1, иначе 0

#!/bin/bash

if [ $# -ne 1 ]; then

echo "Использование: $0 файл\_входных\_данных"

exit 1

fi

input\_file="$1"

result\_file="./func\_tests/data/temp.out"

./app.exe < "$input\_file" > "$result\_file"

result=$?

if [ "$result" -eq 0 ]; then

exit 1

else

exit 0

fi

### Скрипт для автоматизации функционального тестирования

##### **func\_tests.sh**

Скрипт запускает вначале все позитивные тесты из папки data и проверяет код выхода. Если он равен 0, то тест прошел и выводится “PASS”, иначе выводится “ERROR”, что означает, что программа не прошла позитивный тест. Далее скрипт запускает все негативные тесты из папки data и проверяте код выхода. Если он не равен 0, то негативный тест пройден и выводится “PASS”, иначе выводится “ERROR”, что означает, что программа не прошла негативный тест.

#!/bin/bash

n=1

err=0

for test in ./func\_tests/data/pos\_??\_in.txt; do

./func\_tests/scripts/pos\_case.sh "$test" "${test//in/out}"

value=$?

echo "Pos test $n: "

if [ $value -eq 0 ] ; then

printf "\033[92mPASS\033[0m\n"

else

printf "\033[31mERROR\033[0m\n"

err=$((err+1))

fi

n=$((n+1))

done

n=1

for test in ./func\_tests/data/neg\_??\_in.txt; do

./func\_tests/scripts/neg\_case.sh "$test"

value=$?

echo "Neg test $n: "

if [ $value -eq 0 ] ; then

printf "\033[92mPASS\033[0m\n"

else

printf "\033[31mERROR\033[0m\n"

err=$((err+1))

fi

n=$((n+1))

done

exit $err

### 

### Дополнительные скрипты для автоматизации тестирования

##### **collect\_coverage.sh**

Скрипт для сбора анализа покрытия тестами программы

#!/bin/bash

gcov-13 main.c

##### **check\_scripts.sh**

Скрипт для проверки других скриптов с помощью Shellcheck

#!/bin/bash

find . -name '\*.sh' -print0 | xargs -0 shellcheck

##### **run.sh**

Скрипт для автоматизации процесса запусков скриптов

#!/bin/bash

./clean.sh

./build\_debug.sh

echo -e "\033[35m---TESTS---\033[0m"

./func\_tests/scripts/func\_tests.sh

echo -e "\n\033[35m--COVEARGE--\033[0m"

./collect\_coverage.sh

./clean.sh

echo -e "\n\033[35m--ADDRESS SANITIZER--\033[0m"

./build\_debug\_asan.sh

./func\_tests/scripts/func\_tests.sh

./clean.sh

echo -e "\n\033[35m--UNDEFINED BEHAVIOR SANITIZER--\033[0m"

./build\_debug\_ubsan.sh

./func\_tests/scripts/func\_tests.sh

./clean.sh

**Заключение**

Были успешно написаны скрипты для автоматизации процессов тестирования и сборки