Ильченко Ева ИУ7-24Б

Отчет по Заданию №1. Автоматизация функционального тестирования

Цель: написать скрипты для автоматизации процессов тестирования и сборки

Задание

1) 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
```

2) 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
```

3) Скрипт очистки побочных файлов

clean.sh

```
#!/bin/bash

rm -f -- *.o *.exe ./func_tests/data/temp.out *.gcno *.gcov *.gcda
"main"
```

4) 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
```

5) 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
```

6) 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
```

7) 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
```

8) 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
```

9) 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