Паламарчук Андрей ИУ7-23Б

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

## **Цель работы**

Целью данной работы является автоматизация процессов сборки и тестирования.

## **Задачи**

1. Реализовать скрипты отладочной и релизной сборок.
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.

## **Реализация**

### **Скрипт релизной сборки**

#!/bin/bash

# Компиляция

gcc -std=c99 -c main.c func.c add\_entry.c sort.c print\_substr.c

# Комановка

gcc -o main\_release.exe main.o func.o add\_entry.o sort.o print\_substr.o -lm

### **Скрипт отладочной сборки**

#!/bin/bash

# Компиляция

gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -Wvla -c -g3 main.c func.c add\_entry.c sort.c print\_substr.c

# Комановка(-o название исполняемого файла)

gcc -o main\_debug.exe main.o func.o add\_entry.o sort.o print\_substr.o –lm

### **Скрипт отладочной сборки с адрес санитайзером**

#!/bin/bash

clang -std=c99 -Wall -fsanitize=address -fno-omit-frame-pointer -g main.c func.c add\_entry.c sort.c print\_substr.c -o main\_asan.exe

### **Скрипт отладочной сборки с санитайзером памяти**

#!/bin/bash

clang -std=c99 -Wall -fsanitize=memory -fno-omit-frame-pointer -g main.c func.c add\_entry.c sort.c print\_substr.c -o main\_msan.exe

### **Скрипт отладочной сборки с санитайзером UB**

#!/bin/bash

clang -std=c99 -Wall -fsanitize=undefined -fno-omit-frame-pointer -g main.c func.c add\_entry.c sort.c print\_substr.c -o main\_udsan.exe

### **Скрипт очистки побочных файлов**

#!/bin/bash

# \*.txt \*.exe \*.o \*.out \*.gcno \*.gcda \*.gcov

junk\_files1="./func\_tests/\*/\*.out"

junk\_files2="./func\_tests/\*/\*.exe"

junk\_files3="./\*.exe ./\*.o ./\*.out ./\*.gcno ./\*.gcda ./\*.gcov"

# Проверить, существует ли файл

for el in $junk\_files1 $junk\_files2 $junk\_files3;

do

if [[ -f $el ]]; then

rm "$el"

fi

done

### **Компаратор для сравнения двух файлов**

#!/bin/bash

# Проверка количества аргументов

if [ $# -ne 2 ]; then

exit 1

fi

out\_prog=$1

out\_test=$2

# сравниваем их

rc=$(cmp -s "$out\_prog" "$out\_test")

if [ ! "$rc" ]; then

exit 1

fi

exit 0

### **Скрипт pos\_case.sh**

#!/bin/bash

# Все делаем из папки lab\_!!!

# Проверка количества аргументов

if [ $# -ne 3 ]; then

exit 1

fi

file\_in=$1

file\_out=$2

file\_args=$3

# Коды ошибок

test\_pass="0"

test\_failed="1"

flag=$(head -n 1 "$file\_args")

file\_substr=$(sed '2q;d' "$file\_args")

# Если передали входной и выходной файлы,

# то передаём их исполняемому файлу \*.exe

touch ./func\_tests/scripts/prog\_pos.out

prog="./func\_tests/scripts/prog\_pos.out"

# Переводим текстовые файлы в бинарные

sh ./t2b.sh "$file\_in" "$file\_in"

# Если не флаг печати переводим file\_out

if [[ "$flag" != "fb" ]]; then

sh ./t2b.sh "$file\_out" "$file\_out"

fi

if [[ "$flag" == "ab" ]]; then

touch ./func\_tests/data/add.out

add\_entry="./func\_tests/data/add.out"

# Вытаскиваем запись, которую нужно добавить из file\_args

sed -n '3,7p' "$file\_args" > "$add\_entry"

command="./\*.exe ${flag} ${file\_in}"

$command < "$add\_entry" > "$prog"

else

command="./\*.exe ${flag} ${file\_in} ${file\_substr}"

$command > "$prog"

fi

return\_code="$?"

# Проверка завершения программы

if [ "$return\_code" -ne 0 ]; then # не нулевой код ошибки

sh ./b2t.sh "$file\_in" "$file\_in"

# Если не флаг печати переводим file\_out

if [[ "$flag" != "fb" ]]; then

sh ./b2t.sh "$file\_out" "$file\_out"

fi

exit "$test\_failed"

fi

if [[ "$flag" == "ab" ]]; then

# сравниваем выходные данные программы и данные в тесте

if sh ./func\_tests/scripts/comparator.sh "$file\_in" "$file\_out"; then # неверный тест

# Переводим бинарные файлы в текстовые

sh ./b2t.sh "$file\_in" "$file\_in"

sh ./b2t.sh "$file\_out" "$file\_out"

exit "$test\_failed"

else # верный тест

# Переводим бинарные файлы в текстовые

sh ./b2t.sh "$file\_in" "$file\_in"

sh ./b2t.sh "$file\_out" "$file\_out"

exit "$test\_pass"

fi

else

# сравниваем выходные данные программы и данные в тесте

if sh ./func\_tests/scripts/comparator.sh "$prog" "$file\_out"; then # неверный тест

# Переводим бинарные файлы в текстовые

sh ./b2t.sh "$file\_in" "$file\_in"

# Если не флаг печати переводим file\_out

if [[ "$flag" != "fb" ]]; then

sh ./b2t.sh "$file\_out" "$file\_out"

fi

exit "$test\_failed"

else # верный тест

# Переводим бинарные файлы в текстовые

sh ./b2t.sh "$file\_in" "$file\_in"

# Если не флаг печати переводим file\_out

if [[ "$flag" != "fb" ]]; then

sh ./b2t.sh "$file\_out" "$file\_out"

fi

exit "$test\_pass"

fi

fi

### **Скрипт neg\_case.sh**

#!/bin/bash

# Все делаем из папки lab\_!!!

# Проверка количества переданных файлов

if [ $# -ne 2 ]; then

exit 1

fi

file\_in=$1

file\_args=$2

# Коды ошибок

test\_pass="0"

test\_failed="1"

flag=$(head -n 1 "$file\_args")

file\_substr=$(sed '2q;d' "$file\_args")

# Если передали входной и выходной файлы,

# то передаём их исполняемому файлу \*.exe

touch ./func\_tests/scripts/prog\_neg.out

prog="./func\_tests/scripts/prog\_neg.out"

# Переводим текстовые файлы в бинарные

sh ./t2b.sh "$file\_in" "$file\_in"

if [[ "$flag" == "ab" ]]; then

touch ./func\_tests/data/add.out

add\_entry="./func\_tests/data/add.out"

# Вытаскиваем запись, которую нужно добавить из file\_args

sed -n '3,7p' "$file\_args" > "$add\_entry"

command="./\*.exe ${flag} ${file\_in}"

$command < "$add\_entry" > "$prog"

else

command="./\*.exe ${flag} ${file\_in} ${file\_substr}"

$command > "$prog"

fi

return\_code="$?"

# Переводим бинарные файлы в текстовые

sh ./b2t.sh "$file\_in" "$file\_in"

# Проверка завершения программы

if [ "$return\_code" -ne 0 ]; then # не нулевой код ошибки возврата

exit "$test\_pass" # верный тест

else

exit "$test\_failed" # неверный тест

fi

### **Скрипт func\_tests.sh**

Скрипт последовательно вызывает pos\_case.sh, neg\_case.sh и передает им все входные и выходные тестовые файлы (позитивные и негативные соответственно). А также выводит дополнительную информацию о пройденных/проваленных тестах.

#!/bin/bash

# Количество ошибочных тестов

count\_err=0

pos=0

neg=0

# Коды ошибок

test\_pass="0"

# Позитивные тесты

files="./func\_tests/data/pos\_??\_in.txt"

for file\_in in $files; do

# Вытаскиваем номер теста

number=$(echo "$file\_in" | grep -o "[0-9]\*")

# Проверка на наличие тестов(-z длина строки = 0)

if [ -z "$number" ]; then

break

fi

# Флаг наличия поз. тестов

pos=1

# Название выходного тестового файла

file\_out="./func\_tests/data/pos\_""$number""\_out.txt"

# Название файла с аргументами

file\_args="./func\_tests/data/pos\_""$number""\_args.txt"

tmp="./tmp.out"

cp -r "$file\_in" "$tmp"

# Выходной файл существует => передаем входной и выходной файлы в pos\_case.sh

# Не существует, то тест провален, переходим к следующему тесту

if [ -f "$file\_out" ]; then

command="sh ./func\_tests/scripts/pos\_case.sh ""$file\_in ""$file\_out ""$file\_args"

else

echo "POS\_""$number"": FAILED"

count\_err=$((count\_err + 1))

continue

fi

$command

return\_code="$?"

# Результат в соответствии с кодом возврата ./pos\_case.sh

if [ "$return\_code" = "$test\_pass" ]; then

echo "POS\_""$number"": PASSED"

fi

if [ "$return\_code" != "$test\_pass" ]; then

echo "POS\_""$number"": FAILED"

count\_err=$((count\_err + 1))

pos=$((pos + 1))

fi

cp -r "$tmp" "$file\_in"

done

# Негативные тесты

files="./func\_tests/data/neg\_??\_in.txt"

for file\_in in $files; do

# находим номер теста

number=$(echo "$file\_in" | grep -o "[0-9]\*")

# проверка на наличие тестов(-z длина строки = 0)

if [ -z "$number" ]; then

break

fi

# Флаг наличия нег. тестов

neg=1

# Название файла с аргументами

file\_args="./func\_tests/data/neg\_""$number""\_args.txt"

# Передаем входной тестовый файл в ./neg\_case.sh

command="sh ./func\_tests/scripts/neg\_case.sh ""$file\_in ""$file\_args"

$command

return\_code="$?"

# Результат в соответствии с кодом возврата ./neg\_case.sh

if [ "$return\_code" = "$test\_pass" ]; then

echo "NEG\_""$number"": PASSED"

fi

if [ "$return\_code" != "$test\_pass" ]; then

echo "NEG\_""$number"": FAILED"

count\_err=$((count\_err + 1))

fi

done

# Дополнительная информация

if [ "$count\_err" = 0 ]; then

echo "All tests passed."

else

echo "Failed $count\_err tests."

fi

if [ "$pos" = 0 ]; then

echo "No positive tests."

fi

if [ "$neg" = 0 ]; then

echo "No negative tests."

fi

exit "$count\_err"

### **Скрипт сборки с утилитой gcov**

#!/bin/bash

# Компиляция

gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -Wvla -c -g3 --coverage main.c func.c add\_entry.c sort.c print\_substr.c

# Комановка(-o название исполняемого файла)

gcc -o main\_gcov.exe main.o func.o add\_entry.o sort.o print\_substr.o --coverage –lm

### **Скрипт - результат покрытия кода**

#!/bin/bash

gcov main.c func.c add\_entry.c sort.c print\_substr.c

### **Скрипт проверки shellcheck-ом**

#!/bin/bash

this\_path="./\*.sh"

scripts\_path="./func\_tests/scripts/\*.sh"

for file in $this\_path $scripts\_path ; do

shellcheck "$file"

done

### **Скрипт tests.sh**

#!/bin/bash

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

### **Скрипт go.sh**

Скрипт последовательно запускает:

1. Скрипт, который формирует исполняемый файл
2. Скрипт, который проводит все позитивные и негативные тесты программы
3. Скрипт, который очищает ненужные файлы

Так проходит по сборкам со всеми санитайзерами, релизную и отладочную сборки, а также сборку с утилитой gcov.

#!/bin/bash

echo Result testing build\_debug:

sh ./build\_debug.sh

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

sh ./clean.sh

echo ""

echo Result testing build\_release:

sh ./build\_release.sh

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

sh ./clean.sh

echo ""

echo Result testing build\_debug\_asan:

sh ./build\_debug\_asan.sh

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

sh ./clean.sh

echo ""

echo Result testing build\_debug\_msan:

sh ./build\_debug\_msan.sh

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

sh ./clean.sh

echo ""

echo Result testing build\_debug\_ubsan:

sh ./build\_debug\_ubsan.sh

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

sh ./clean.sh

echo ""

echo Result gcov:

sh ./build\_gcov.sh

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

sh ./collect\_coverage.sh

sh ./clean.sh

echo ""

echo Temporary files removed

echo ""

### **Скрипт b2t.sh**

Скрипт формирует исполняемый файл, для перевода бинарного файла в текстовый, и запускает его с переданными аргументами (исходный файл и конечный файл).

#!/bin/bash

# Проверка количества аргументов

if [ $# -ne 2 ]; then

exit 1

fi

sh ./func\_tests/b2t/build\_b2t.sh

touch ./func\_tests/b2t/tmp1.out

tmp\_1="./func\_tests/b2t/tmp1.out"

echo -n > "$tmp\_1"

touch ./func\_tests/b2t/tmp2.out

tmp\_2="./func\_tests/b2t/tmp2.out"

echo -n > "$tmp\_2"

cp -r "$1" "$tmp\_1"

./func\_tests/b2t/b2t.exe "$tmp\_1" "$tmp\_2"

cp -r "$tmp\_2" "$2"

if [[ -f "$tmp\_1" ]]; then

rm "$tmp\_1"

fi

if [[ -f "$tmp\_2" ]]; then

rm "$tmp\_2"

fi

### **Скрипт сборки b2t.exe**

#!/bin/bash

# Компиляция

gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -Wvla -c -g3 ./func\_tests/b2t/b2t.c

# Комановка(-o название исполняемого файла)

gcc -o ./func\_tests/b2t/b2t.exe b2t.o -lm

### **Скрипт t2b.sh**

Скрипт формирует исполняемый файл, для перевода текстового файла в бинарный, и запускает его с переданными аргументами (исходный файл и конечный файл).

#!/bin/bash

# Проверка количества аргументов

if [ $# -ne 2 ]; then

exit 1

fi

sh ./func\_tests/t2b/build\_t2b.sh

touch ./func\_tests/t2b/tmp1.out

tmp\_1="./func\_tests/t2b/tmp1.out"

touch ./func\_tests/t2b/tmp2.out

tmp\_2="./func\_tests/t2b/tmp2.out"

cp -r "$1" "$tmp\_1"

./func\_tests/t2b/t2b.exe "$tmp\_1" "$tmp\_2"

cp -r "$tmp\_2" "$2"

if [[ -f "$tmp\_1" ]]; then

rm "$tmp\_1"

fi

if [[ -f "$tmp\_2" ]]; then

rm "$tmp\_2"

fi

### **Скрипт сборки t2b.exe**

#!/bin/bash

# Компиляция

gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -Wvla -c -g3 ./func\_tests/t2b/t2b.c

# Комановка(-o название исполняемого файла)

gcc -o ./func\_tests/t2b/t2b.exe t2b.o -lm