Паламарчук Андрей ИУ7-23Б

# **Автоматизация функционального тестирования**

## **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 последовательностей действительных чисел, располагающихся в двух текстовых файлах, с игнорированием остального содержимого.
5. Реализовать компаратор для сравнения содержимого двух текстовых файлов, располагающегося после первого вхождения подстроки «Result:␣».
6. Реализовать скрипт pos\_case.sh для проверки позитивного тестового случая по определённым далее правилам.
7. Реализовать скрипт neg\_case.sh для проверки негативного тестового случая по определённым далее правилам.
8. Обеспечить автоматизацию функционального тестирования.

## **Реализация**

### **Скрипт релизной сборки**

build\_release.sh

#!/bin/bash

if [ $# != 1 ]; then

echo Usage: ./build\_release.sh file\_name

exit 1

fi

a=$1

out=${a%%.c}

out\_f=$out"\_release.exe"

out\_s=$out".o"

# Компиляция

gcc -std=c99 -c "$1"

# Комановка

gcc -o "$out\_f" "$out\_s" –lm

### **Скрипт отладочной сборки**

build\_debug.sh

#!/bin/bash

if [ $# != 1 ]; then

echo Usage: ./build\_debug.sh file\_name

exit 1

fi

a=$1

out=${a%%.c}

out\_f=$out"\_debug.exe"

out\_s=$out".o"

# Компиляция

gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -Wvla -c -g3 "$1"

# Комановка(-o название исполняемого файла)

gcc -o "$out\_f" "$out\_s" -lm

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

build\_debug\_asan.sh

#!/bin/bash

if [ $# != 1 ]; then

echo Usage: ./build\_debug\_asan.sh file\_name

exit 1

fi

a=$1

out=${a%%.c}

out=$out"\_asan.exe"

clang -std=c99 -Wall -fsanitize=address -fno-omit-frame-pointer -g "$1" -o "$out"

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

build\_debug\_msan.sh

#!/bin/bash

if [ $# != 1 ]; then

echo Usage: ./build\_debug\_msan.sh file\_name

exit 1

fi

a=$1

out=${a%%.c}

out=$out"\_msan.exe"

clang -std=c99 -Wall -fsanitize=memory -fno-omit-frame-pointer -g "$1" -o "$out"

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

build\_debug\_ubsan.sh

#!/bin/bash

if [ $# != 1 ]; then

echo Usage: ./build\_debug\_udsan.sh file\_name

exit 1

fi

a=$1

out=${a%%.c}

out=$out"\_udsan.exe"

clang -std=c99 -Wall -fsanitize=undefined -fno-omit-frame-pointer -g "$1" -o "$out"

### **Скрипт очистки побочных файлов**

clean.sh

#!/bin/bash

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

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

junk\_files2="./\*.exe \*.o \*.gcno \*.gcda \*.gcov"

# Проверить, существует ли файл

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

do

if [[ -f $el ]]; then

rm "$el"

fi

done

### **Компаратор для последовательностей действительных чисел**

comparator.sh

#!/bin/bash

# Проверка количества аргументов

if [ $# -ne 2 ]; then

exit 1

fi

# Маска для данной задачи

mask="[+-]?[0-9][0-9]\*\.?[0-9]\*"

out\_prog=$1

out\_test=$2

# grep-ом собираем выхлоп из файлов

prog=$(grep -Eo "$mask" "$out\_prog")

test=$(grep -Eo "$mask" "$out\_test")

# сравниваем их

if [ "$prog" != "$test" ]; then

exit 1

fi

exit 0

### **Компаратор с вхождением подстроки «Result:␣»**

comparator.sh

#!/bin/bash  
  
# Проверка количества аргументов  
if [ $# -ne 2 ]; then  
exit 1  
fi  
# Маска для данной задачи  
mask="Result: .\*"  
  
out\_prog=$1  
out\_test=$2  
  
# grep-ом собираем выхлоп из файлов  
prog=$(grep -Eo "$mask" "$out\_prog")  
test=$(grep -Eo "$mask" "$out\_test")  
# сравниваем их  
if [ "$prog" != "$test" ]; then  
exit 1  
fi  
exit 0

### **Скрипт pos\_case.sh**

#!/bin/bash

# Все делаем из папки lab\_!!!

# Проверка количества переданных файлов

if [ $# -ne 2 ]; then

exit 1

fi

test\_in=$1

test\_out=$2

# Коды ошибок

test\_pass="0"

test\_failed="1"

# Если передали входной и выходной файлы,

# то передаём их исполняемому файлу \*.exe

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

command="./\*.exe"

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

$command < "$test\_in" > "$prog"

return\_code="$?"

# Проверка завершения программы

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

exit "$test\_failed"

fi

# сравниваем выходные данные программы и данные в тесте

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

exit "$test\_failed"

else # верный тест

exit "$test\_pass"

fi

### **Скрипт neg\_case.sh**

#!/bin/bash

# Все делаем из папки lab\_!!!

# Проверка количества переданных файлов

if [ $# -ne 1 ]; then

exit 1

fi

test\_in=$1

# Коды ошибок

test\_pass="0"

test\_failed="1"

# Если передали входной и выходной файлы,

# то передаём их исполняемому файлу \*.exe

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

command="./\*.exe"

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

$command < "$test\_in" > "$prog"

return\_code="$?"

# Проверка завершения программы

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

exit "$test\_pass" # верный тест

else # неверный тест

exit "$test\_failed"

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"

test\_failed="1"

# 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"

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

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\_failed" ]; then

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

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

pos=$((pos + 1))

fi

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

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

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

$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\_failed" ]; 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**

build\_gcov.sh

#!/bin/bash  
  
if [ $# != 1 ]; then  
echo Usage: ./build\_gcov.sh file\_name  
exit 1  
fi  
  
a=$1  
out=${a%%.c}  
out\_exe=$out"\_gcov.exe"  
  
gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal  
-Wfloat-conversion -Wvla -c -O0 -g3 --coverage "$1"  
gcc "$out".o -o "$out\_exe" --coverage –lm

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

collect\_coverage.sh

#!/bin/bash  
if [ $# != 1 ]; then  
echo Usage: ./collect\_coverage.sh file\_name  
exit 1  
fi  
gcov "$1"

### **Скрипт проверки shellcheck-ом**

check\_scripts.sh

#!/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**

Запускает скрипт func\_tests.sh

#!/bin/bash  
  
./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

if [ $# != 1 ]; then

echo Usage: ./go.sh file\_name

exit 1

fi

echo Result testing build\_debug:

./build\_debug.sh "$1"

./func\_tests/scripts/func\_tests.sh "$1"

./clean.sh

echo ""

echo Result testing build\_release:

./build\_release.sh "$1"

./func\_tests/scripts/func\_tests.sh "$1"

./clean.sh

echo ""

echo Result testing build\_debug\_asan:

./build\_debug\_asan.sh "$1"

./func\_tests/scripts/func\_tests.sh "$1"

./clean.sh

echo ""

echo Result testing build\_debug\_msan:

./build\_debug\_msan.sh "$1"

./func\_tests/scripts/func\_tests.sh "$1"

./clean.sh

echo ""

echo Result testing build\_debug\_ubsan:

./build\_debug\_ubsan.sh "$1"

./func\_tests/scripts/func\_tests.sh "$1"

./clean.sh

echo ""

echo Result gcov:

./build\_gcov.sh "$1"

./func\_tests/scripts/func\_tests.sh "$1"

./collect\_coverage.sh "$1"

./clean.sh

echo ""

echo Temporary files removed

echo ""

### **Скрипт go\_all.sh**

Последовательно запускает скрипт go.sh для каждой задачи

#!/bin/bash  
  
if [ $# != 1 ]; then  
echo Usage: ./go\_all.sh file\_name  
exit 1  
fi  
  
tasks="./lab[\_??]\*"  
for file in $tasks; do  
echo "$file"  
cd "$file" || exit 1  
./go.sh "$1"  
cd .. || exit 1  
done