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 "$1"
# Командовка
gcc -o "$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 "$1"
# Командовка(-o название исполняемого файла)
gcc -o "$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
```