Отчет по заданию №1 в рамках вычислительного практикума
Автоматизация функционального тестирования
Автор: Краснов Леонид ИУ7-21Б

Цель работы

Автоматизация процессов сборки и тестирования

Инструкция по эксплуатации

Скрипт предназначен для тестирования консольных программ, написанных на Си.
Для проведения тестирования программы нужно поместить программу в рабочую папку, где находятся скрипты build_debug.sh и build_release.sh.
После поместить в папку /func_tests/data/ файлы с эталонными данными, называя pos_in_[номер_теста].txt (файл с тем, что программа должна принять), pos_out_[номер_теста].txt (файл с тем, что программа должна вывести), и neg_in[номер_теста].txt (файл с некорректным вводом, при котором программа должна завершиться с ошибкой).
Запустить скрипт func_tests.sh, который производит сравнение с тем, что выдала программа и эталонными данными. Результатом работы скрипта является информация о том, какие тесты программа завалила и какие прошла.

Скрипты

Скрипт для отладочной сборки build_debug.sh

```
#!/bin/bash
gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion
-g -c main.c
gcc -o app.exe main.o -lm
./app.exe
```

Скрипт для релизной сборки buld_release.sh

```
#!/bin/bash
gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wvla -Wfloat-con-
version -o app.exe main.c -lm
./"app.exe"
```

Скрипт для очистки побочных файлов clean.sh

```
#!/bin/bash

for i in ./*; do
    if [[ $i != *.sh ]] && [[ $i != *.c ]]; then
        rm "$i" 2> proga.error
    fi
done
rm proga.error
```

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

```
#!/bin/bash
# Реализовать компаратор для сравнения последовательностей действительных чи-
сел, располагающихся в двух текстовых
# файлах, с игнорированием остального содержимого.
re="[0-9-]-"
file1="$1"
file2="$2"
flag=0

<"$file1" grep -E -o "$re*" > "out1_temp.txt"

<"$file2" grep -E -o "$re*" > "out2_temp.txt"
if cmp -s "out1_temp.txt" "out2_temp.txt" ; then
    flag=0
else
    flag=1
fi
rm "out1_temp.txt" "out2_temp.txt"
exit "$flag"
```

Компаратор для сравнения содержимого двух текстовых файлов, расположающегося после первого вхождения подстроки «Result: ␣».
comparator2.sh

```
#!/bin/bash
# Реализовать компаратор для сравнения содержимого двух текстовых файлов, рас-
полагающегося после первого
# вхождения подстроки «Result:␣».

file1=$1
file2=$2
echo "x" > "new1.txt"
echo "x" > "new2.txt"
cat "$file1" >> "new1.txt"
cat "$file2" >> "new2.txt"
file1="new1.txt"
file2="new2.txt"
cat "$file1" >> "out1.txt"
cat "$file2" >> "out2.txt"

sed '1,/Result:/ d' < "$file1" > "out3_1.txt"
cat "out3_1.txt" > "out3.txt"
comm -13 <(sort -u "out3.txt") <(sort -u "out1.txt") > "out4.txt"
< "out4.txt"  tail -n1 > "out3.txt"
cat "out3_1.txt" > "out1.txt"
rm out3.txt out3_1.txt out4.txt

sed '1,/Result:/ d' < "$file2" > "out3_1.txt"
cat "out3_1.txt" > "out3.txt"
comm -13 <(sort -u "out3.txt") <(sort -u "out2.txt") > "out4.txt"
< "out4.txt"  tail -n1 > "out3.txt"
cat "out3_1.txt" > "out2.txt"
rm out3.txt out3_1.txt out4.txt new1.txt new2.txt

if cmp -s "out1.txt" "out2.txt" ; then
    exit 0
else exit 1
fi
rm "out1.txt" "out2.txt"
```

Скрипт pos_case.sh для проверки позитивного тестового случая.

```
#!/bin/bash
# pos_case.sh принимает в качестве аргументов файл для подмены входного по-
тока, файл эталонных выходных данных и, при
# наличии, файл ключей, с которыми вызывается приложение.
# pos_case.sh file_stream_in file_stream_out_expect [file_app_args]
# (a) Рабочей папкой скрипт считает свою папку.
# (b) Скрипт ожидает от приложения нулевой код возврата в случае успеха.
# (c) Скрипт по умолчанию работает в «молчаливом режиме», возвращает нуль при
успешном тестировании, иначе — не нуль.
# (d) Мусор после работы скрипта не очищается.

file_in="$1"
file_out="$2"
#file_app_args="$3"
if [[ "$USE_VALGRIND" == "" ]]; then
    ../../app.exe < "$file_in" > "program_out.txt"
    rez="$?"
    ./comparator2.sh "program_out.txt" "$file_out"
    cmp_exit="$?"
    rm "program_out.txt"
    if [[ $cmp_exit -eq 0 ]]; then
        exit "$rez"
    else exit 1
    fi
else
    cmp_exit=0
    valgrind --log-file="../../trash.txt" ../../app.exe < "$file_in" > "pro-
gram_out.txt"
    if [[ -s "../../trash.txt" ]]; then
        cmp_exit=0
        rez="$?"
        ./comparator2.sh "program_out.txt" "$file_out"
        cmp_exit="$?"
        rm "program_out.txt"
        if [[ $cmp_exit -eq 0 ]]; then
            exit "$rez"
        else exit 1
        fi
    else
        rm "program_out.txt"
        cmp_exit=1
    fi
fi
exit "$cmp_exit"
```

Скрипт neg_case.sh для проверки негативного тестового случая

```
#!/bin/bash
# neg_case.sh принимает в качестве аргументов файл для подмены входного потока
# и, при наличии, файл ключей, с которыми вызывается приложение. neg_case.sh
file_stream_in [file_app_args]
# (a) Рабочей папкой скрипт считает свою папку.
# (b) Скрипт ожидает от приложения ненулевой код возврата в случае ошибки.
# (c) Скрипт по умолчанию работает в «молчаливом режиме», возвращает нуль при
успешном тестировании, иначе — не нуль.
# (d) Мусор после работы скрипта не очищается.
file_in="$1"
rez="$?"
out_code=0
if [[ "$USE_VALGRIND" == "" ]]; then
    "../../app.exe" < "$file_in" > "program_out.txt"
```

```
    rez="$?"
    out_code=0
    if [[ ! $rez -eq 0 ]]; then
        out_code=0
    else
        out_code=1
    fi
    rm "program_out.txt"
    exit $out_code
else
    valgrind --log-file="../../trash.txt" ../../app.exe < "$file_in" > "pro-
gram_out.txt"
    rez="?"
    if [[ -s "../../trash.txt" ]]; then
        rm "program_out.txt"
        exit 0
    else
        echo "Memory_fail"
        rm "program_out.txt"
        exit 1
    fi
fi
exit "$out_code"
```

Скрипт для обеспечения автоматизации функционального тестирования func_tests.sh

```
#!/bin/bash
# Автотесты
ls ../data > "tests.txt"
< "tests.txt" grep "neg" > "neg.txt"
< "tests.txt" grep "pos.*in" > "in.txt"
< "tests.txt" grep "pos.*out" > "out.txt"
i=0
while IFS= read -r line
do
    i=$((i + 1))
    arr_neg[$i]="../data/$line"
done < "neg.txt"
i=0
while IFS= read -r line
do
    i=$((i + 1))
    arr_pos_in[$i]="../data/$line"
done < "in.txt"
i=0
while IFS= read -r line
do
    i=$((i + 1))
    arr_pos_out[$i]="../data/$line"
done < "out.txt"
i=0
k=0
e=0
while [[ $i -lt ${#arr_neg[*]} ]]; do
    i=$((i + 1))
    ./neg_case.sh "${arr_neg[$i]}"
    ex_code="$?"
    if [[ "$ex_code" -eq 0 ]]; then
        k=$((k + 1))
        e=1
    fi
```

```
    if [[ $e -eq 1 ]]; then
            e=0
            echo "NEG_TEST $i passed"
        else
            echo "NEG_TEST $i failed"
        fi
done
echo "$k/$i neg tests passed"
i=0
k=0
e=0
while [[ $i -lt ${#arr_pos_in[*]} ]]; do
    i=$((i + 1))
    ./pos_case.sh "${arr_pos_in[$i]}" "${arr_pos_out[$i]}"
    ex_code="$?"
    if [[ "$ex_code" -eq 0 ]]; then
        e=1
        k=$((k + 1))
    fi
    if [[ $e -eq 1 ]]; then
        e=0
        echo "POS_TEST $i passed"
    else
        echo "POS_TEST $i failed"
    fi
done
echo "$k/$i pos tests passed"
rm "neg.txt" "in.txt" "out.txt" "tests.txt"
```