**Задание 1 часть 2**

**Скрипт build\_debug.sh**

#!/bin/bash

# Build program with debugging information

# Set default compile flags

FLAGS="-std=c99 -Wall -Werror -Wextra -Wpedantic -Wfloat-equal -Wfloat-conversion -Wvla -g -fprofile-arcs -ftest-coverage -lm"

# Compile all C source files in the current directory

gcc $FLAGS -c ./\*.c

# Link object files into the executable

gcc $FLAGS -o app.exe ./\*.o

Предназначен для компилирование программ с отладочной информации

**Скрипт build\_debug\_asan.sh**

#!/bin/bash

clang -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -fsanitize=address -fno-omit-frame-pointer -g \*.c

clang \*.c -o app.exe -lm

Предназначен для компилирование программ с отладочной информации, с адрес санитайзер.

**Скрипт build\_debug\_msan.sh**

#!/bin/bash

clang -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -fsanitize=memory -fPIE -pie -fno-omit-frame-pointer -g main.c

clang \*.c -o app.exe -lm

Предназначен для компилирование программ с отладочной информации, с память санитайзер.

**Скрипт build\_debug\_ubsan.sh**

#!/bin/bash

clang -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -fsanitize=undefined -fno-omit-frame-pointer -g main.c

clang \*.c -o app.exe -lm

Предназначен для компилирование программ с отладочной информации, с неопределенное поведение санитайзер

**Скрипт build\_relase.sh**

#!/bin/bash

gcc -c -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -Wvla main.c -03

clang \*.c -o app.exe -lm

Предназначен для компилирование программ при выпуске.

**Скрипт clean.sh**

#!/bin/bash

rm ./\*.exe ./\*.out ./\*.o ./\*.gcno ./\*.gcda ./\*.gcov ./\*.prof\* >/dev/null 2>&1

**Скрипт check\_script.sh**

#!/bin/bash

shellcheck ./\*sh ./func\_tests/scripts/\*.sh

Предназначен для shellcheck bash скрипты.

**Скрипт collect\_coverage.sh**

#!/bin/bash

find ./func\_tests/data -name '\*\_in.txt' | while read f; do

    ./app.exe <"$f" >/dev/null 2>&1

done

gcov ./\*.c

**Скрипт neg\_case.sh**

#!/bin/bash

usage="usage: neg\_case file\_stream\_in file\_stream\_out\_expect file\_args [-v]"

print\_error()

{

    cat <<-EOF

        file\_args: $file\_args

        file\_in: $file\_in

        return\_code: $ret\_code

        file\_out: $file\_out != $file\_expect

EOF

}

# error: wrong parameter count

if (( $# < 3 || $# > 4 )); then

    echo "neg\_case: Wrong parameter count"

    echo "$usage"

    exit 2

elif [[ $# -eq 4 && ($4 != "" && $4 != "-v") ]]; then

    echo "neg\_case: Wrong option"

    echo "$usage"

    exit 3

fi

# filename

file\_in=$1

file\_expect=$2

file\_args=$3

args=$(cat "$file\_args")

file\_exe=./app.exe

file\_out=/tmp/"$(basename "$file\_expect")"

# not file

if [[ -z $file\_in || \

      -z $file\_expect || \

      -z $file\_exe || \

      -z $file\_args ]]; then

    echo "neg\_case: Missing file"

    echo "$usage"

    exit 4

fi

# run app

if ! app="$file\_exe $args" "$app" <"$file\_in" >"$file\_out"; then

    ret\_code=$?

    if ((ret\_code == 0)); then

        print\_error

        exit 5

    fi

fi

# compare error message

if ! ../common/src/cmp\_after\_str.sh "$file\_out" "$file\_expect" "Error:" "$4"; then

    ret\_code=$?

    if ((ret\_code != 0)); then

        print\_error

    fi

fi

exit $ret\_code

Предназначен для тестирование незитивных входных данных.

**Скрипт pos\_case.sh**

#!/bin/bash

usage="usage: pos\_case file\_stream\_in file\_stream\_out\_expect file\_args [-v]"

print\_error() {

    cat <<-EOF

        file\_in: $file\_in

        return\_code: $ret\_code

        file\_out: $file\_out != $file\_expect

EOF

}

# error: wrong parameter count

if [[ $# -lt 3 || $# -gt 4 ]]; then

    echo "pos\_case: Wrong parameter count"

    echo "$usage"

    exit 2

fi

# error: wrong option

if [[ $# -eq 4 && ! -z $4 && $4 != "-v" ]]; then

    echo "pos\_case: Wrong option"

    echo "$usage"

    exit 3

fi

# Check if files are missing

missing\_files=false

for file in "$file\_in" "$file\_expect" "$file\_exe" "$file\_args"; do

    if [[ ! -f "$file" ]]; then

        missing\_files=true

        break

    fi

done

if [[ $missing\_files == true ]]; then

    echo "pos\_case: Missing file"

    echo "$usage"

    exit 4

fi

# Run app

$file\_exe < "$file\_in" > "$file\_out" ||

{

    ret\_code=$?

    print\_error

    exit 5

}

# compare result

if ! ./func\_tests/scripts/comparator.sh "$file\_out" "$file\_expect" "$4"; then

    print\_error

fi

exit $ret\_code

Предназначен для тестирование позитивных входных данных.

**Скрипт func\_tests.sh**

#!/bin/bash

usage="usage: func\_tests [-v]"

# error: wrong parameter count

[[ $# -gt 1 ]] && { echo "func\_tests: Wrong parameter count"; echo "$usage"; exit 2; }

# error: wrong option

[[ $# -eq 1 && $1 != "" && $1 != "-v" ]] && { echo "func\_tests: Wrong option"; echo "$usage"; exit 3; }

# positive tests

pos\_total=0

pos\_pass=0

file\_in="./func\_tests/data/pos\_\*\_in.txt"

while IFS= read -r file\_in; do

    if [[ -f $file\_in ]]; then

        pos\_total=$(("$pos\_total" + 1))

        ./func\_tests/scripts/pos\_case.sh "$file\_in" "${file\_in/\_in.txt/\_out.txt}" "$1"

        ret\_code=$?

        if [[ $ret\_code -eq 0 ]]; then

            pos\_pass=$(("$pos\_pass" + 1))

            result="Pass"

        else

            result="Not Pass"

        fi

        echo "Positive test: $file\_in. Return code: $ret\_code. Result: $result"

    fi

done < <(find ./func\_tests/data/ -name "pos\_\*\_in.txt")

echo "Total positive test: $pos\_total. Pass: $pos\_pass."

# negative tests

neg\_total=0

neg\_pass=0

file\_in="./func\_tests/data/neg\_\*\_in.txt"

while IFS= read -r file\_in; do

    if [[ -f $file\_in ]]; then

        neg\_total=$(("$neg\_total" + 1))

        ./func\_tests/scripts/neg\_case.sh "$file\_in" "${file\_in/\_in.txt/\_out.txt}" "$1"

        ret\_code=$?

        if [[ $ret\_code -eq 0 ]]; then

            neg\_pass=$(("$neg\_pass" + 1))

            result="Pass"

        else

            result="Not Pass"

        fi

        echo "Negative test: $file\_in. Return code: $ret\_code. Result: $result"

    fi

done < <(find ./func\_tests/data/ -name "neg\_\*\_in.txt")

echo "Total negative test: $neg\_total. Pass: $neg\_pass."

[[ $pos\_pass -ne $pos\_total || $neg\_pass -ne $neg\_total ]] && exit 1

**Скрипт comparator\_int.sh**

#!/bin/bash

usage="usage: comparator file1 file2 [-v]"

# error: wrong parameter count

if (( $# < 2 || $# > 3 )); then

    echo "comparator: Wrong parameter count"

    echo "$usage"

    exit 2

fi

# error: wrong option

if [[ $# -eq 3 && $3 != "" && $3 != "-v" ]]; then

    echo "comparator: Wrong option"

    echo "$usage"

    exit 3

fi

# filename

file1=$1

file2=$2

# regex

regex="^[+-]?[0-9]+$"

space="[[:space:]]"

# function to print verbose message

print\_verbose() {

    if [[ "$3" = "-v" ]]; then

        echo "$1"

    fi

}

# loop for working with 2 files simultaneously

while IFS= read -r -N 1 c1 <&3 && IFS= read -r -N 1 c2 <&4; do

    # number

    num1=""

    num2=""

    # find next number in file1

    for (( ; ; )); do

        # has eof

        if [[ -z $c1 ]]; then

            break

        fi

        # has space

        if [[ "$c1" =~ $space ]]; then

            if [[ "$num1" =~ $regex ]]; then

                break

            fi

            num1=""

        else

            # forming number

            num1=$num1$c1

        fi

        IFS= read -r -N 1 c1 <&3

    done

    # find next number in file2

    for (( ; ; )); do

        # has eof

        if [[ -z $c2 ]]; then

            break

        fi

        # has space

        if [[ "$c2" =~ $space ]]; then

            if [[ "$num2" =~ $regex ]]; then

                break

            fi

            num2=""

        else

            # forming number

            num2=$num2$c2

        fi

        IFS= read -r -N 1 c2 <&4

    done

    # compare numbers

    if [[ "$num1" =~ $regex && "$num2" =~ $regex ]]; then

        print\_verbose "Comparing $num1 & $num2" "$3"

        if [[ "$num1" != "$num2" ]]; then

            print\_verbose "file: $file1 != file: $file2" "$3"

            exit 1

        fi

    elif [[ "$num1" =~ $regex ]]; then

        print\_verbose "file1 has more numbers than file2" "$3"

        print\_verbose "file: $file1 != file: $file2" "$3"

        exit 4

    elif [[ "$num2" =~ $regex ]]; then

        print\_verbose "file2 has more numbers than file1" "$3"

        print\_verbose "file: $file1 != file: $file2" "$3"

        exit 5

    fi

done 3< "$file1" 4< "$file2"

**Скрипт comparator\_float.sh**

#!/bin/bash

usage="usage: comparator file1 file2 [-v]"

# error: wrong parameter count

if (( $# < 2 || $# > 3 )); then

    echo "comparator: Wrong parameter count"

    echo "$usage"

    exit 2

fi

# error: wrong option

if [[ $# -eq 3 && $3 != "" && $3 != "-v" ]]; then

    echo "comparator: Wrong option"

    echo "$usage"

    exit 3

fi

# filename

file1=$1

file2=$2

# regex

regex="^[+-]?[0-9]\*\.[0-9]+([eE][+-]?[0-9]+)?$"

space="[[:space:]]"

# function to print verbose message

print\_verbose() {

    if [[ "$3" = "-v" ]]; then

        echo "$1"

    fi

}

# read file1 and file2 simultaneously, find numbers

exec 3< "$file1"

exec 4< "$file2"

# loop for working with 2 files simultaneously

while read -r -N 1 c1 <&3 && read -r -N 1 c2 <&4; do

    # number

    num1=""

    num2=""

    # find next number in file1

    while [[ "$c1" =~ $space ]]; do

        if [[ "$num1" =~ $regex ]]; then

            break

        fi

        read -r -N 1 c1 <&3

    done

    # forming number in file1

    while [[ ! "$c1" =~ $space && "$c1" != "" ]]; do

        num1=$num1$c1

        read -r -N 1 c1 <&3

    done

    # find next number in file2

    while [[ "$c2" =~ $space ]]; do

        if [[ "$num2" =~ $regex ]]; then

            break

        fi

        read -r -N 1 c2 <&4

    done

    # forming number in file2

    while [[ ! "$c2" =~ $space && "$c2" != "" ]]; do

        num2=$num2$c2

        read -r -N 1 c2 <&4

    done

    # compare numbers

    if [[ "$num1" =~ $regex && "$num2" =~ $regex ]]; then

        print\_verbose "Comparing $num1 & $num2" "$3"

        if [[ "$num1" != "$num2" ]]; then

            print\_verbose "file: $file1 != file: $file2" "$3"

            exit 1

        fi

    elif [[ "$num1" =~ $regex ]]; then

        print\_verbose "file1 has more numbers than file2" "$3"

        print\_verbose "file: $file1 != file: $file2" "$3"

        exit 4

    elif [[ "$num2" =~ $regex ]]; then

        print\_verbose "file2 has more numbers than file1" "$3"

        print\_verbose "file: $file1 != file: $file2" "$3"

        exit 5

    fi

done

# files are equal

print\_verbose "file: $file1 == file: $file2" "$3"

exit 0

**Скрипт comparator\_after\_str.sh**

#!/bin/bash

usage="usage: comparator file1 file2 str [-v]"

# error: wrong parameter count

if (( $# < 3 || $# > 4 )); then

    echo "comparator: Wrong parameter count"

    echo "$usage"

    exit 2

fi

# error: wrong option

if [[ $# -eq 4 && $4 != "" && $4 != "-v" ]]; then

    echo "comparator: Wrong option"

    echo "$usage"

    exit 3

fi

# filename

file1=$1

file2=$2

# regex

regex=$3

len=${#regex}

# function to print verbose message

print\_verbose() {

    if [[ "$4" = "-v" ]]; then

        echo "$1"

    fi

}

# loop for working with 2 files simultaneously

while IFS= read -r -N 1 c1 <&3 && IFS= read -r -N 1 c2 <&4; do

    # find 'string:' in file1

    str1=""

    byte1=0

    for (( ; ; )); do

        byte1=$((byte1 + 1))

        # found 'string:'

        if [[ "$str1" == "$regex" ]]; then

            print\_verbose "file1: found $regex at $byte1 byte" "$4"

            break

        else

            # forming string

            str1=$str1$c1

            # max length > $len, cut it

            if (( ${#str1} > len )); then

                str1=${str1:1:len}

            fi

        fi

        # has eof

        if [[ -z $c1 ]]; then

            break

        fi

        IFS= read -r -N 1 c1 <&3

    done

    # find 'string:' in file2

    str2=""

    byte2=0

    for (( ; ; )); do

        byte2=$((byte2 + 1))

        # found 'string:'

        if [[ "$str2" == "$regex" ]]; then

            print\_verbose "file2: found $regex at $byte2 byte" "$4"

            break

        else

            # forming string

            str2=$str2$c2

            # max length > $len, cut it

            if (( ${#str2} > len )); then

                str2=${str2:1:len}

            fi

        fi

        # has eof

        if [[ -z $c2 ]]; then

            break

        fi

        IFS= read -r -N 1 c2 <&4

    done

    # found start position, now just compare

    for (( ; ; )); do

        # if two files reach the end

        if [[ -z $c1 && -z $c2 ]]; then

            # eof at the same time, equal

            if [[ -z $c1 && -z $c2 ]]; then

                print\_verbose "file: $file1 == file: $file2" "$4"

                exit 0

            # otherwise, not equal

            else

                print\_verbose "file: $file1 != file: $file2" "$4"

                exit 1

            fi

        fi

        IFS= read -r -N 1 c1 <&3 || c1=""

        IFS= read -r -N 1 c2 <&4 || c2=""

        # if char in file1 != char in file2

        if [[ "$c1" != "$c2" ]]; then

            print\_verbose "file: $file1 != file: $file2" "$4"

            exit 1

        fi

    done

done 3< "$file1" 4< "$file2"

**Скрипт comparator\_null.sh**

#!/bin/bash

# Compare nothing.

exit 0

**Скрипт comparator.sh**

#!/bin/bash

# Compare two files with diff util

# usage="usage: comparator file1 file2"

diff "$1" "$2"

Предназначен для сравнения содержимого двух текстовых файлов.

**Заключение**

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.

**Задание выполнено**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. William E Shotts - The Linux Command Line\_ A Complete Introduction (2019, No Starch Press)

2. Chris F. A. Johnson, Jayant Varma (auth.) - Pro Bash Programming\_ Scripting the GNU\_Linux Shell (2015, Apress)

3. https://www.man7.org/linux/man-pages/man1/bash.1.htm