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