

Лабораторная работа 13

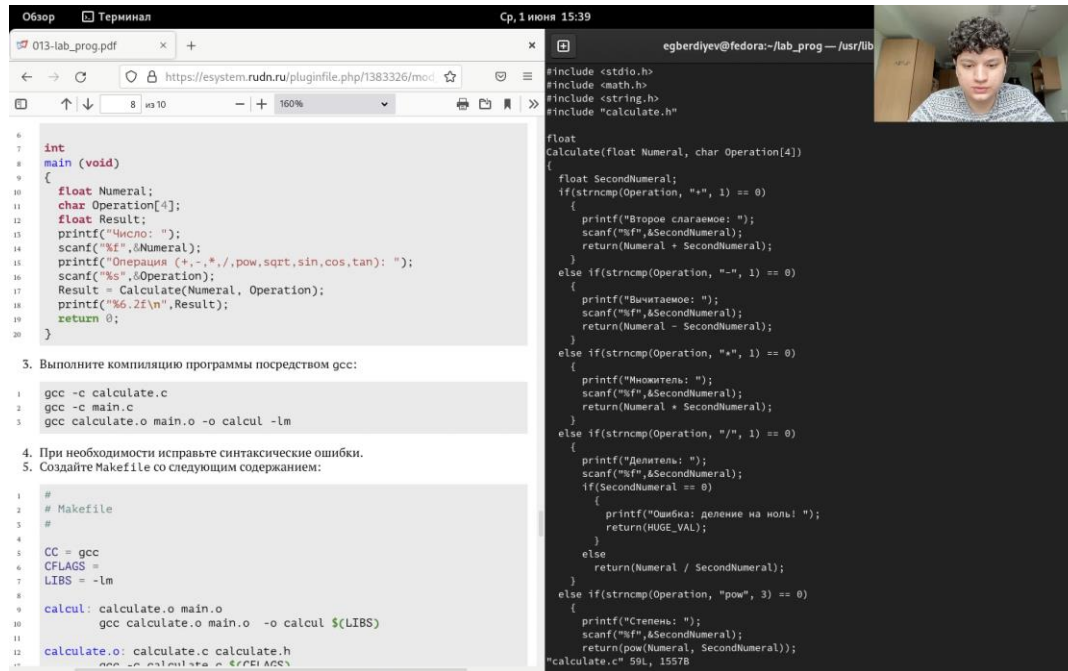
Бердыев Эзиз Группа НФИбд-01-21

Цель работы

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.

Выполнение

1. Создание 3 файла: calculate.c calculate.h main.c



```
6 int
7 main (void)
8 {
9     float Numeral;
10    char Operation[4];
11    float Result;
12    printf("Число: ");
13    scanf("%f",&Numeral);
14    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15    scanf("%s",&Operation);
16    Result = Calculate(Numeral, Operation);
17    printf("%g.2f\n",Result);
18    return 0;
19 }
20
```

3. Выполните компиляцию программы посредством gcc:

```
1 gcc -c calculate.c
2 gcc -c main.c
3 gcc calculate.o main.o -o calcul -lm
```

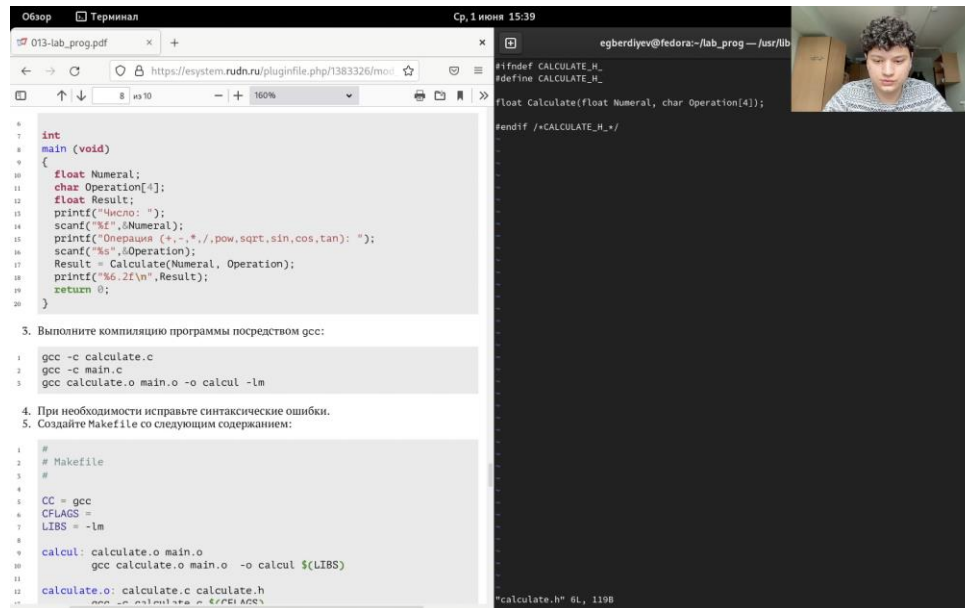
4. При необходимости исправьте синтаксические ошибки.
5. Создайте Makefile со следующим содержанием:

```
1 #
2 # Makefile
3 #
4 CC = gcc
5 CFLAGS =
6 LIBS = -lm
7
8 calcul: calculate.o main.o
9     gcc calculate.o main.o -o calcul $(LIBS)
10
11 calculate.o: calculate.c calculate.h
12     gcc -c calculate.c $(CFLAGS)
```

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "calculate.h"

float
Calculate(float Numeral, char Operation[4])
{
    float SecondNumeral;
    if(strncmp(Operation, "+", 1) == 0)
    {
        printf("Сложение: ");
        scanf("%f",&SecondNumeral);
        return(Numeral + SecondNumeral);
    }
    else if(strncmp(Operation, "-", 1) == 0)
    {
        printf("Вычитание: ");
        scanf("%f",&SecondNumeral);
        return(Numeral - SecondNumeral);
    }
    else if(strncmp(Operation, "*", 1) == 0)
    {
        printf("Умножение: ");
        scanf("%f",&SecondNumeral);
        return(Numeral * SecondNumeral);
    }
    else if(strncmp(Operation, "/", 1) == 0)
    {
        printf("Деление: ");
        scanf("%f",&SecondNumeral);
        if(SecondNumeral == 0)
        {
            printf("Ошибка: деление на ноль! ");
            return(HUGE_VAL);
        }
        else
            return(Numeral / SecondNumeral);
    }
    else if(strncmp(Operation, "pow", 3) == 0)
    {
        printf("Степень: ");
        scanf("%f",&SecondNumeral);
        return(pow(Numeral, SecondNumeral));
    }
    "calculate.c" 59L, 1557B
}
```

calculate.c.png



calculate.h.png

Обзор Терминал Ср, 1 июня 15:39

013-lab_prog.pdf x +

https://esystem.rudn.ru/pluginfile.php/1383326/mod/

```
int
main (void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Число: ");
    scanf("%f",&Numeral);
    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
    scanf("%s",&Operation);
    Result = Calculate(Numeral, Operation);
    printf("%g.2f\n",Result);
    return 0;
}
```

3. Выполните компиляцию программы посредством gcc:

```
gcc -c calculate.c
gcc -c main.c
gcc calculate.o main.o -o calcul -lm
```

4. При необходимости исправьте синтаксические ошибки.

5. Создайте Makefile со следующим содержанием:

```
#
# Makefile
#
CC = gcc
CFLAGS =
LIBS = -lm


calcul: calculate.o main.o
gcc calculate.o main.o -o calcul ${LIBS}

calculate.o: calculate.c calculate.h
gcc -c calculate.c ${CFLAGS}
```

```
#include <stdio.h>
#include "calculate.h"

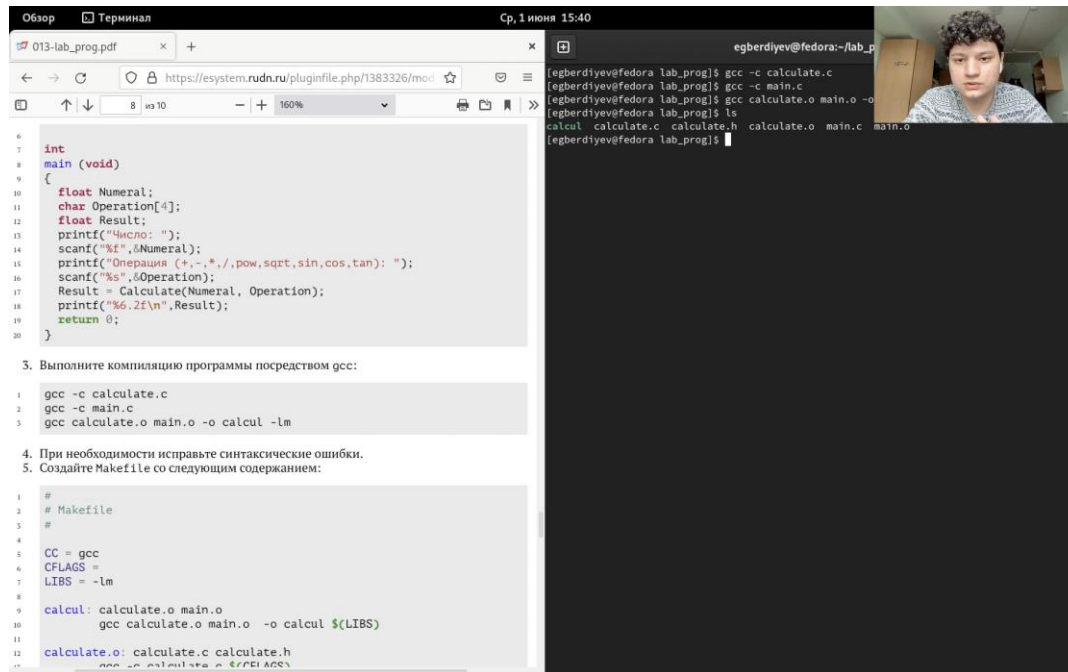
int
main (void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Число: ");
    scanf("%f",&Numeral);
    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
    scanf("%s",&Operation);
    Result = Calculate(Numeral, Operation);
    printf("%g.2f\n",Result);
    return 0;
}
```

"main.c" 171, 3388



main.c.png

2. Компиляция этих файлов



The screenshot shows a web browser window on the left and a terminal window on the right. The browser window displays a C program source code for a calculator. The terminal window shows the execution of gcc commands to compile the program into an executable.

```
6 int
7 main (void)
8 {
9     float Numeral;
10    char Operation[4];
11    float Result;
12    printf("Число: ");
13    scanf("%f",&Numeral);
14    printf("Операция (+, -, *, /, pow, sqrt, sin, cos, tan): ");
15    scanf("%s",&Operation);
16    Result = Calculate(Numeral, Operation);
17    printf("%g.2f\n",Result);
18    return 0;
19 }
20
```

3. Выполните компиляцию программы посредством gcc:

```
1 gcc -c calculate.c
2 gcc -c main.c
3 gcc calculate.o main.o -o calcul -lm
```

4. При необходимости исправьте синтаксические ошибки.
5. Создайте Makefile со следующим содержанием:

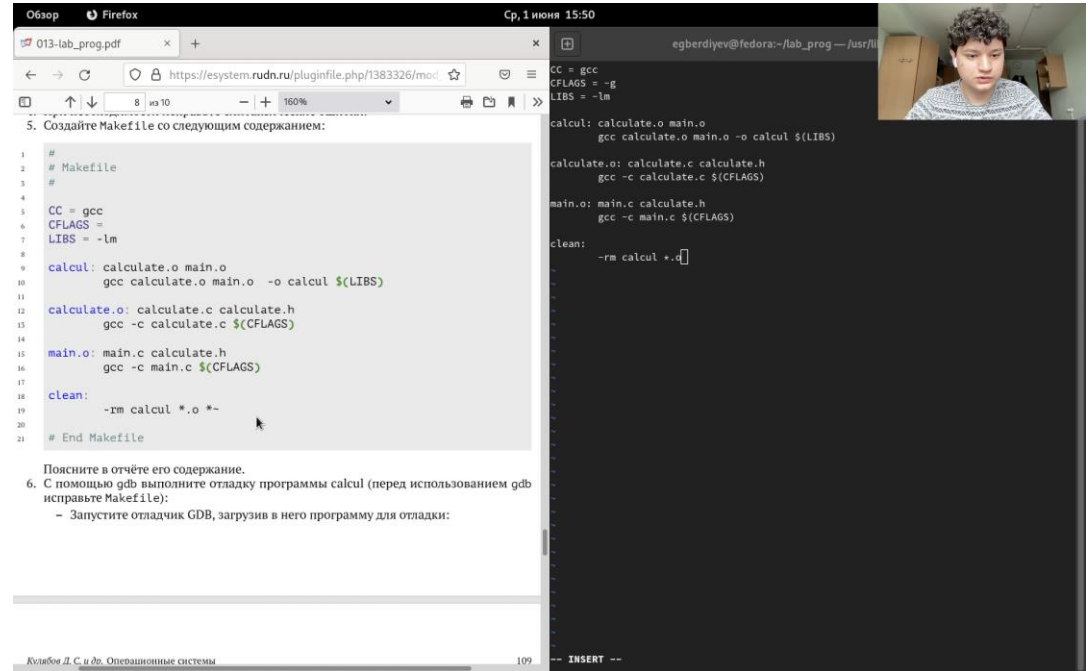
```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS =
7 LIBS = -lm
8
9 calcul: calculate.o main.o
10     gcc calculate.o main.o -o calcul $(LIBS)
11
12 calculate.o: calculate.c calculate.h
13     gcc -c calculate.c $(CFLAGS)
```

The terminal window on the right shows the following commands and output:

```
egberdiyev@fedora lab_prog]$ gcc -c calculate.c
egberdiyev@fedora lab_prog]$ gcc -c main.c
egberdiyev@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
egberdiyev@fedora lab_prog]$ ls
calcul calculate.c calculate.h calculate.o main.c main.o
egberdiyev@fedora lab_prog]$
```

Компиляция этих файлов.png

3. Создание Makefile и его исправление



The screenshot shows a video lecture interface. On the left, a Firefox browser window displays a PDF document titled '013-lab_prog.pdf' from 'esystem.rudn.ru'. The document content includes a task instruction and a Makefile template. On the right, a terminal window shows the execution of the Makefile commands. A small video inset in the top right corner shows the lecturer.

5. Создайте Makefile со следующим содержанием:

```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS =
7 LIBS = -lm
8
9 calcul: calculate.o main.o
10 gcc calculate.o main.o -o calcul $(LIBS)
11
12 calculate.o: calculate.c calculate.h
13 gcc -c calculate.c $(CFLAGS)
14
15 main.o: main.c calculate.h
16 gcc -c main.c $(CFLAGS)
17
18 clean:
19 -rm calcul *.o *-
20
21 # End Makefile
```

Поясните в отчёте его содержание.

6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

- Запустите отладчик GDB, загрузив в него программу для отладки:

```
CC = gcc
CFLAGS = -g
LIBS = -lm

calcul: calculate.o main.o
gcc calculate.o main.o -o calcul $(LIBS)

calculate.o: calculate.c calculate.h
gcc -c calculate.c $(CFLAGS)

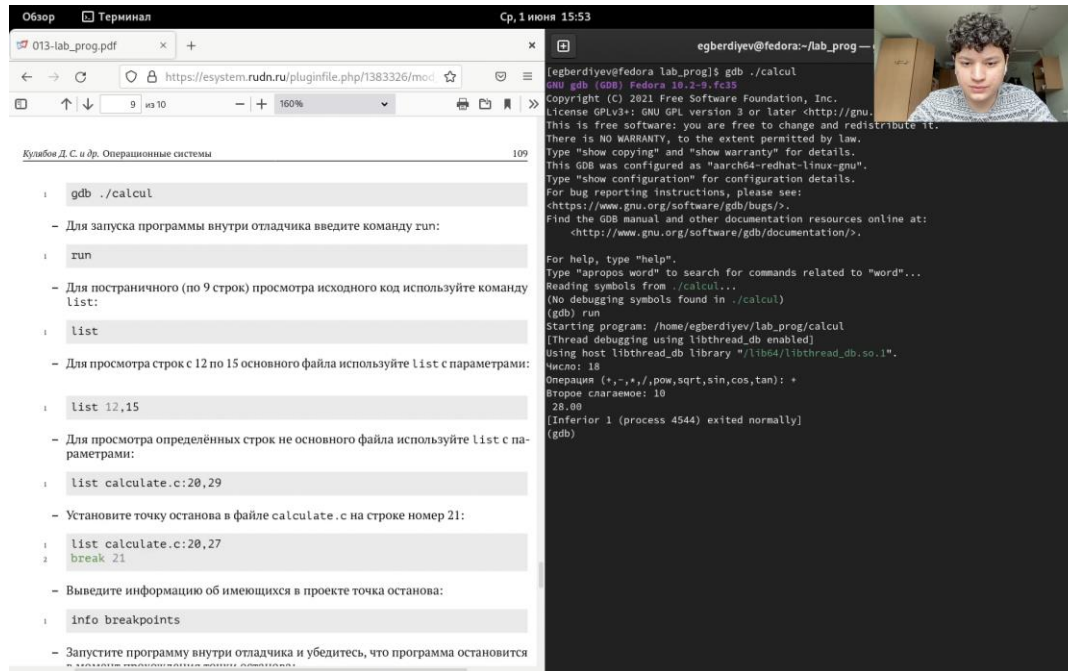
main.o: main.c calculate.h
gcc -c main.c $(CFLAGS)

clean:
-rm calcul *.o
```

Кудряков Д. С. и др. Операционные системы 109 -- INSERT --

Makefile.png

4. Запуск канкулятора



Запуск канкулятора.png

5. Анализирование файлов calculate.c main.c при помощи команды splint

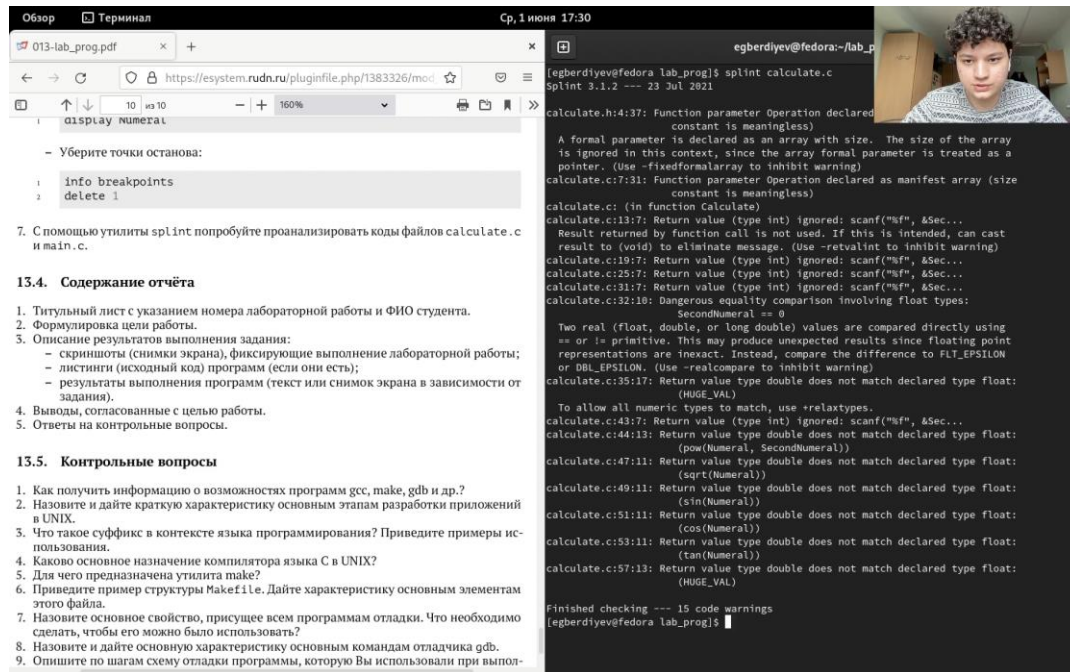
7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

13.4. Содержание отчёта

1. Титульный лист с указанием номера лабораторной работы и ФИО студента.
2. Формулировка цели работы.
3. Описание результатов выполнения задания:
 - скриншоты (снимки экрана), фиксирующие выполнение лабораторной работы;
 - листинги (исходный код) программ (если они есть);
 - результаты выполнения программ (текст или снимок экрана в зависимости от задания).
4. Выводы, согласованные с целью работы.
5. Ответы на контрольные вопросы.

13.5. Контрольные вопросы

1. Как получить информацию о возможностях программ gcc, make, gdb и др.?
2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.
3. Что такое суффикс в контексте языка программирования? Приведите примеры использования.
4. Каково основное назначение компилятора языка C в UNIX?
5. Для чего предназначена утилита make?
6. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла.
7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать?
8. Назовите и дайте основную характеристику основным командам отладчика gdb.
9. Опишите по шагам схему отладки программы, которую Вы использовали при выпол-



calculate.c.png

Обзор

Терминал

Ср, 1 июня 17:30

013-lab_prog.pdf

←

→

↺

↻

10

10

100%

⌵

⌶

⌵

⌶

⌵

⌶

⌵

⌶

display numeral

– Уберите точки останова:

1 info breakpoints

2 delete 1

7. С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c`.

13.4. Содержание отчёта

- Титульный лист с указанием номера лабораторной работы и ФИО студента.
- Формулировка цели работы.
- Описание результатов выполнения задания:
 - скриншоты (снимки экрана), фиксирующие выполнение лабораторной работы;
 - листинги (исходный код) программ (если они есть);
 - результаты выполнения программ (текст или снимок экрана в зависимости от задания).
- Выводы, согласованные с целью работы.
- Ответы на контрольные вопросы.

13.5. Контрольные вопросы

- Как получить информацию о возможностях программ `gcc`, `make`, `gdb` и др.?
- Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.
- Что такое суффикс в контексте языка программирования? Приведите примеры использования.
- Каково основное назначение компилятора языка C в UNIX?
- Для чего предназначена утилита `make`?
- Приведите пример структуры `Makefile`. Дайте характеристику основным элементам этого файла.
- Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать?
- Назовите и дайте основную характеристику основным командам отладчика `gdb`.
- Опишите по шагам схему отладки программы, которую Вы использовали при выпол-

egberdiyev@fedora:~/lab_p

```

calculate.c:32:10: Dangerous equality comparison involving
                  SecondNumeral == 0
Two real (float, double, or long double) values are c
== or != primitive. This may produce unexpected resul
representations are inexact. Instead, compare the difference to FLT_EPSILON
or DBL_EPSILON. (Use -realcompare to inhibit warning)
calculate.c:35:17: Return value type double does not match declared type float:
                  (HUGE_VAL)
To allow all numeric types to match, use -relaxtypes.
calculate.c:43:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:44:13: Return value type double does not match declared type float:
                  (pow(Numeral, SecondNumeral))
calculate.c:47:11: Return value type double does not match declared type float:
                  (sqrt(Numeral))
calculate.c:49:11: Return value type double does not match declared type float:
                  (sin(Numeral))
calculate.c:51:11: Return value type double does not match declared type float:
                  (cos(Numeral))
calculate.c:53:11: Return value type double does not match declared type float:
                  (tan(Numeral))
calculate.c:57:13: Return value type double does not match declared type float:
                  (HUGE_VAL)
Finished checking --- 15 code warnings
[egberdiyev@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2021
calculate.h:4:37: Function parameter Operation declared as manifest array (size
constant is meaningless)
A formal parameter is declared as an array with size. The size of the array
is ignored in this context, since the array formal parameter is treated as a
pointer. (Use -fixedformalarray to inhibit warning)
main.c: (in function main)
main.c:11:3: Return value (type int) ignored: scanf("%f", &Num...
Result returned by function call is not used. If this is intended, can cast
result to (void) to eliminate message. (Use -retvalint to inhibit warning)
main.c:13:14: Format argument 1 to scanf (%s) expects char * gets char (4) *:
                  &Operation
Type of parameter is not consistent with corresponding code in format string.
(Use -formattype to inhibit warning)
main.c:13:11: Corresponding format code
main.c:13:3: Return value (type int) ignored: scanf("%s", &Ope...
Finished checking --- 4 code warnings
[egberdiyev@fedora lab_prog]$

```

main.c.png

Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:
`while [$1 != "exit"] $1`. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое 2:
Как объединить (конкатенация) несколько строк в одну?
`cat file.txt | xargs | sed -e 's/\./.\n/g'`

3. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash? seq - выдает последовательность чисел. Реализовать ее функционал можно командой `for n in {1..5} do done`

4. Какой результат даст вычисление выражения $\$(10/3)$?

5. Укажите кратко основные отличия командной оболочки zsh от bash. Zsh очень сильно упрощает работу. Но существуют различия. Например, в zsh после for обязательно вставлять пробел, нумерация массивов в zsh начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендую Bash. Если скрипты вам не нужны - Zsh (более простая работа с файлами, например)

6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))` Верен

7. Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки? `Bash` позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от обычного языка программирования). Но относительно обычных языков программирования `bash` очень сжат. Тот же `C` имеет гораздо более широкие возможности для разработчика.

Вывод

Приобрел простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.