

# **РУДН. Операционные системы**

**Отчёт по лабораторной работе №12**

Косинов Никита Андреевич, НПИМбв-02-20

# Содержание

1	Цель работы	5
2	Ход работы	6
3	Написание программы “Простейший калькулятор”	7
4	Выводы	12

## Список иллюстраций

3.1	Создание файлов . . . . .	7
3.2	Код основного файла . . . . .	7
3.3	Компиляция файлов . . . . .	8
3.4	Исполняемые файлы . . . . .	8
3.5	Создание файла make . . . . .	8
3.6	Код файла make . . . . .	9
3.7	Отладка приложения . . . . .	9
3.8	Запуск программы внутри отладчика . . . . .	10
3.9	Проверка двух операций . . . . .	10
3.10	Установка точки останова . . . . .	10
3.11	Информация о точках останова . . . . .	10
3.12	Удаление точки останова . . . . .	11
3.13	Выход из отладчика gdb . . . . .	11
3.14	Анализ кода программы утилитой splint . . . . .	11

## **Список таблиц**

# 1 Цель работы

Познакомиться с основными принципами написания, отладки, тестирования приложения в **ОС Linux** на примере разработки приложения “Простейший калькулятор” на языке **C**.

## 2 Ход работы

Лабораторная работа выполнена в терминале **ОС Linux**, командной оболочке **bash**, отладчике **gdb** и хостинге хранения проектов **Github**.

Действия по лабораторной работе представлены в виде последовательных шагов.

По завершении отчёта, вся рабочая папка отправляется на репозиторий на *github*.

## 3 Написание программы “Простейший калькулятор”

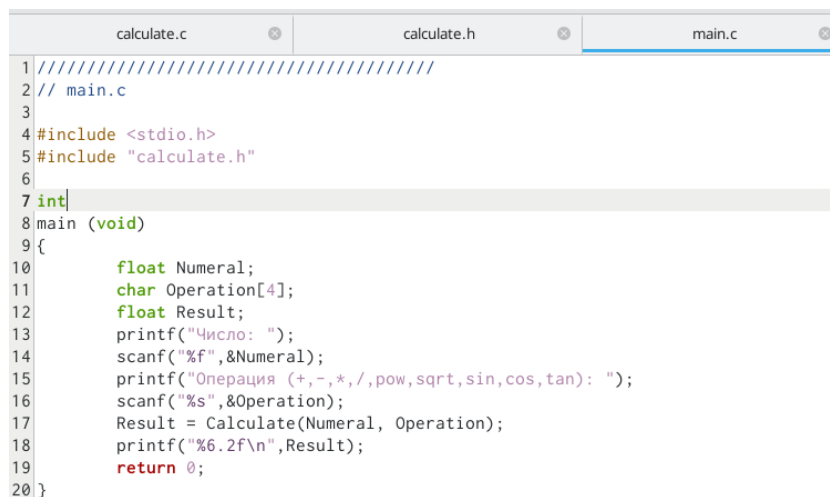
Перед началом работы создадим новый рабочий каталог **lab12** и перейдём внутрь. Также не забываем синхронизироваться с нашим **Git**.

1. В первую очередь перейдём в рабочую директорию и создадим необходимые файлы.

```
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro $ cd labs/lab12
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $ touch
h calculate.h calculate.c main.c
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $
```

Рис. 3.1: Создание файлов

2. Напишем необходимый код для нашего приложения.



```
1 //////////////////////////////////////////////////
2 // main.c
3
4 #include <stdio.h>
5 #include "calculate.h"
6
7 int
8 main (void)
9 {
10     float Numeral;
11     char Operation[4];
12     float Result;
13     printf("Число: ");
14     scanf("%f",&Numeral);
15     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
16     scanf("%s",&Operation);
17     Result = Calculate(Numeral, Operation);
18     printf("%.2f\n",Result);
19     return 0;
20 }
```

Рис. 3.2: Код основного файла

3. Сконфигурируем написанные программы с помощью компилятора **gcc**.

```
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $ gcc
-c calculate.c
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $ gcc
-c main.c
main.c: В функции «main»:
main.c:16:17: предупреждение: формат «%s» ожидает аргумент типа «char *», но аргумент 2
имеет тип «char (*)[4]» [-Wformat=]
   16 |         scanf("%s",&Operation);
      |         ~^ ~~~~~
      |         | |
      |         | | char (*)[4]
      |         | | char *
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $ gcc
calculate.o main.o -o calcul -lm
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $
```

Рис. 3.3: Компиляция файлов

4. Увидим, что после компиляции появились соответствующие исполняемые файлы.

```
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $ ls
calcul calculate.c calculate.h calculate.o main.c main.o presentation report
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $
```

Рис. 3.4: Исполняемые файлы

5. Создадим конфигурирующий *Makefile*.

```
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $ touch
h Makefile
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $
```

Рис. 3.5: Создание файла make

6. Пропишем в *Makefile* следующий код:



```

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

```

Рис. 3.6: Код файла make

Здесь синим подсвечены цели, после двоеточия указаны зависимости для этих целей, далее идут команды.

## 7. Запустим отладчик **gdb**

```

nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab12 $ gdb
./calcul
GNU gdb (Gentoo 13.2 vanilla) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(No debugging symbols found in ./calcul)
(gdb) █

```

Рис. 3.7: Отладка приложения

## 8. Запустим программу внутри отладчика для проверки корректности её работы.

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/n/a/nakosinov/work/study/2022-2023/Опера-
ционные системы/os-intro/labs/lab12/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib64/libthread_db.so.1".
Число: █
```

Рис. 3.8: Запуск программы внутри отладчика

9. Запускаем программу несколько раз, чтобы проверить работоспособность каждой функции нашего калькулятора. Запустить пришлось 9 раз, по одному на каждую команду, причём для операции деления дважды, чтобы проверить обработку исключения “деление на ноль”.

```
Число: 9
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): pow
Степень: 2
81.00
[Inferior 1 (process 23551) exited normally]
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/n/a/nakosinov/work/study/2022-2023/Опера-
ционные системы/os-intro/labs/lab12/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib64/libthread_db.so.1".
Число: 10
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): /
Делитель: 2
5.00
[Inferior 1 (process 23578) exited normally]
(gdb) █
```

Рис. 3.9: Проверка двух операций

10. Поставим точку останова командой *break* на 21-ю строку файла *calculate.c*, соответствующую моменту ввода вычитаемого. Выбираем ответ у (yes).

```
(gdb) break calculate.c:21
No symbol table is loaded. Use the "file" command.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (calculate.c:21) pending.
(gdb) █
```

Рис. 3.10: Установка точки останова

11. Проверим наличие точки останова, выведя информацию о всех таких ко-мандой *info breakpoints*.

Информация о точках останова

Рис. 3.11: Информация о точках останова

12. Удалим созданную точку останова командой *delete*. Снова выведем информацию о точках, чтобы убедиться в корректности выполненного удаления.

Удаление точки останова

Рис. 3.12: Удаление точки останова

13. Выход из отладчика производится командой *quit(q)*.

Выход из отладчика gdb

Рис. 3.13: Выход из отладчика gdb

14. Проанализируем код программы, используя утилиту *splint*.

Анализ кода программы утилитой splint

Рис. 3.14: Анализ кода программы утилитой splint

15. Видим, что эта утилита нашла 15 ошибок. Координаты ошибки даются после названия файла, далее идёт описание ошибки. Например, та, что идёт первой относится к связанному файлу *calculate.h*: 7 строка, 37 символ, ошибка связана с переменной *Operation[4]*. **Splint** утверждает, что бессмысленно указывать размер константы.

## 4 Выводы

Мы познакомились с отладчиком приложений **gdb** и более требовательным **splint**, протестировали написанное приложение “Простейший калькулятор”, научились ставить точки останова.