

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных
наук**

**Кафедра прикладной информатики и теории
вероятностей**

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14

**Средства, применяемые при
разработке программного обеспечения в ОС типа UNIX/Linux**

Студент: Тасыбаева Наталья Сергеевна

Группа: НПИбд-02-20

МОСКВА 2021г.

Цель работы:

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

Ход работы:

1. В домашнем каталоге я создала подкаталог ~/work/os/lab_prog (рис.1)

```
nstasihbaeva@dk3n51 ~ $ cd work
nstasihbaeva@dk3n51 ~/work $ cd os
nstasihbaeva@dk3n51 ~/work/os $ ls
lab09
nstasihbaeva@dk3n51 ~/work/os $ mkdir lab_prog
nstasihbaeva@dk3n51 ~/work/os $ dc lab_prog
dc: Will not attempt to process directory lab_prog
nstasihbaeva@dk3n51 ~/work/os $ cd lab_prog
nstasihbaeva@dk3n51 ~/work/os/lab_prog $
```

Рис.1

2. Создала в нём файлы: calculate.h, calculate.c, main.c (рис.2)

```
nstasihbaeva@dk3n51 ~/work/os/lab_prog $ touch calculate.h calculate.c main.c
nstasihbaeva@dk3n51 ~/work/os/lab_prog $ ls
calculate.c calculate.h main.c
nstasihbaeva@dk3n51 ~/work/os/lab_prog $
```

Рис.2

Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять sin, cos, tan. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится.

3. Код файла calculate.c: (рис.3)

```

4 #include "calculate.h"
5
6 float Calculate(float Numeral, char Operation[4]) {
7     float SecondNumeral;
8     if(strncmp(Operation, "+", 1)==0)
9     {
10         printf("Vtoroe slagaemoe: ");
11         scanf("%f", &SecondNumeral);
12         return (Numeral + SecondNumeral);
13     }
14     else if (strncmp(Operation, "-", 1) == 0)
15     {
16         printf("Vychytaemoe: ");
17         scanf("%f", &SecondNumeral);
18         return(Numeral - SecondNumeral);
19     }
20     else if(strncmp(Operation, "*", 1) == 0)
21     {
22         printf("Delitel: ");
23         scanf("%f",&SecondNumeral);
24         if(SecondNumeral == 0)
25         {
26             printf("Oshibka: delenie na null! ");
27             return(HUGE_VAL);
28         }
29         else
30             return (Numeral / SecondNumeral);
31     }
32     else if(strncmp(Operation, "pow", 3) == 0)
33     {
34         printf("Stepen: ");
35         scanf("%f", &SecondNumeral);
36         return(pow(Numeral, SecondNumeral));
37     }
38     else if(strncmp(Operation, "sqrt", 4) == 0) return (sqrt(Numeral));
39     else if(strncmp(Operation, "sin", 3) == 0) return(sin(Numeral));
40     else if(strncmp(Operation, "cos", 3) == 0) return(cos(Numeral));
41     else if(strncmp(Operation, "tan", 3) == 0) return (tan(Numeral));
42     else
43     {
44         printf("Nepravilno vvedeno deystvie ");
45         return (HUGE_VAL);}
46     }
47
48

```

Рис.3

4. Код файла calculate.h: (рис.4)

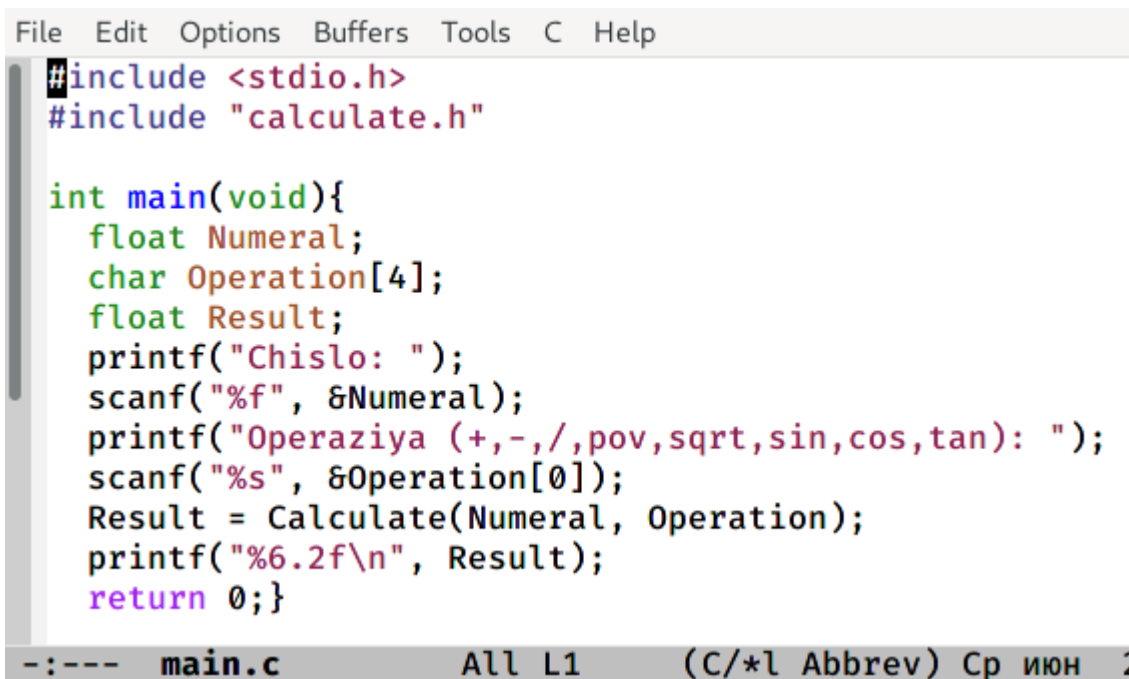
```

1 #ifndef CALCULATE_H_
2 #define CALCULATE_H_
3 float Calculate(float Numeral, char Operation[4]);
4 #endif/*CALCULATE_H_*/

```

Рис.4

5. Код файла main.c: (рис.5)



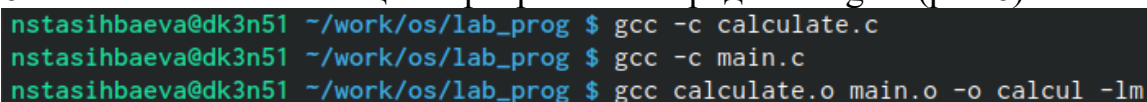
```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include "calculate.h"

int main(void){
    float Numeral;
    char Operation[4];
    float Result;
    printf("Chislo: ");
    scanf("%f", &Numeral);
    printf("Operaziya (+,-,/,pov,sqrt,sin,cos,tan): ");
    scanf("%s", &Operation[0]);
    Result = Calculate(Numeral, Operation);
    printf("%6.2f\n", Result);
    return 0;}

-:--- main.c All L1 (C/*l Abbrev) Ср июн 2
```

Рис.5

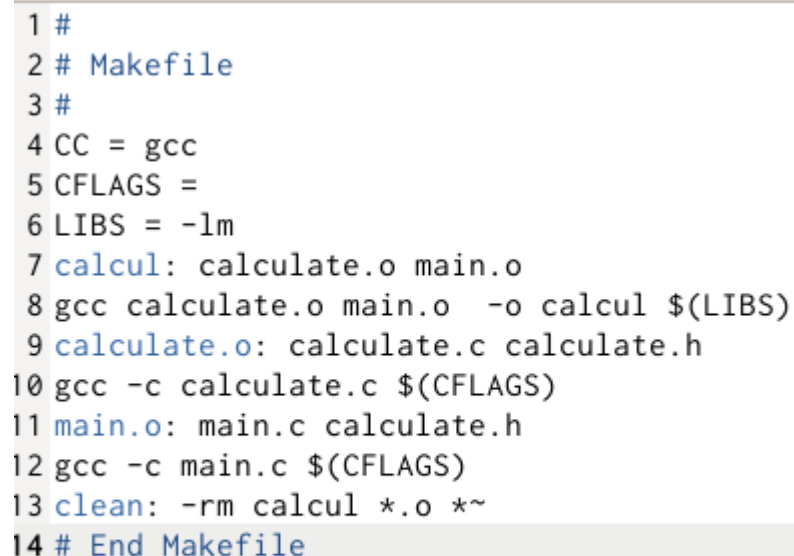
6. Выполнила компиляцию программы посредством gcc: (рис.6)



```
nstasihbaeva@dk3n51 ~/work/os/lab_prog $ gcc -c calculate.c
nstasihbaeva@dk3n51 ~/work/os/lab_prog $ gcc -c main.c
nstasihbaeva@dk3n51 ~/work/os/lab_prog $ gcc calculate.o main.o -o calcul -lm
```

Рис.6

7. Создала Makefile со следующим содержанием: (рис.7)



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

Рис.7

8. Запустила отладчик GDB, загрузив в него программу для отладки:
gdb ./calcul (рис.8)

```

nstasihbaeva@dk3n51 ~/work/os/lab_prog $ emacs Makefile
#nstasihbaeva@dk3n51 ~/work/os/lab_prog $ gdb ./calcul
GNU gdb (Gentoo 10.1 vanilla) 10.1
Copyright (C) 2020 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.8

9. Для запуска программы внутри отладчика ввела команду run: (рис.9)

```

(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/n/s/nstasihbaeva/work/os/lab_prog/calcul
Chislo: 2
Operaziya (+,-,/,pov,sqrt,sin,cos,tan): pov
Nepravilno vvedeno deystvie    inf
[Inferior 1 (process 39257) exited normally]

```

Рис.9

10. Далее все команды, которые с оператором list в моём терминале не работает. (рис.10)

```

(gdb) list
No symbol table is loaded.  Use the "file" command.
(gdb) list 12,15
No symbol table is loaded.  Use the "file" command.
(gdb) █

```

Рис.10

Вывод:

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

Ответы на контрольные вопросы:

1. Как получить информацию о возможностях программ gcc, make, gdb и др.?

Воспользоваться интернетом, воспользоваться командой man, info

2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.

Unix поддерживает следующие основные этапы разработки приложений:

- создание исходного кода программы;
- сохранение различных вариантов исходного текста;
- анализ исходного текста; Необходимо отслеживать изменения исходного кода, а также при работе более двух программистов над проектом программы нужно, чтобы они не делали изменений кода в одно время.
- компиляция исходного текста и построение исполняемого модуля
- тестирование и отладка;
- сохранение всех изменений, выполняемых при тестировании и отладке.

3. Что такое суффикс в контексте языка программирования? Приведите примеры использования.

Использование суффикса ".c" для имени файла с программой на языке Си отражает удобное и полезное соглашение, принятое в ОС UNIX. Для любого имени входного файла суффикс определяет какая компиляция требуется. Суффиксы и префиксы указывают тип объекта. Одно из полезных свойств компилятора Си — его способность по суффиксам определять типы файлов. По суффиксу .c компилятор распознает, что файл abcd.c должен компилироваться, а по суффиксу .o, что файл abcd.o является объектным модулем и для получения исполняемой программы необходимо выполнить редактирование связей. Простейший пример командной строки для компиляции программы abcd.c и построения исполняемого модуля abcd имеет вид: gcc -o abcd abcd.c.

4. Каково основное назначение компилятора языка C в UNIX?

В компиляции всей программы в целом и получении исполняемого модуля.

5. Для чего предназначена утилита make?

Для упрощения и автоматизации работы пользователя с командной строкой

6. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла.

Текст, следующий за точкой с запятой, и все последующие строки, начинающиеся с литеры табуляции, являются командами ОС UNIX, которые необходимо выполнить для обновления целевого файла. Таким образом, спецификация взаимосвязей имеет формат:

```
target1 [ target2...]: [:] [dependment1...]
```

```
[(tab)commands]
```

```
[#commentary]
```

```
[(tab)commands]
```

```
[#commentary],
```

где # — специфицирует начало комментария; : — последовательность команд ОС UNIX должна содержаться в одной строке make-файла (файла описаний), есть возможность переноса команд (\), но она считается как одна строка; :: — последовательность команд ОС UNIX может содержаться в нескольких последовательных строках файла описаний.

7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать?

Все программы отладки позволяют отслеживать состояние программы на любом из этапов ее исполнения. Для того чтобы эту возможность использовать необходимо изучить документацию по использованию определенного отладчика. Понять общие принципы отладки.

8. Назовите и дайте основную характеристику основным командам отладчика gdb.

– backtrace – выводит весь путь к текущей точке останова, то есть

названия всех функций, начиная от `main()`; иными словами, выводит весь стек функций;

– `break` – устанавливает точку останова; параметром может быть номер строки или название функции;

– `clear` – удаляет все точки останова на текущем уровне стека (то есть в текущей функции);

– `continue` – продолжает выполнение программы от текущей точки до конца;

– `delete` – удаляет точку останова или контрольное выражение;

– `display` – добавляет выражение в список выражений, значения которых отображаются каждый раз при остановке программы;

– `finish` – выполняет программу до выхода из текущей функции; отображает возвращаемое значение, если такое имеется;

– `info breakpoints` – выводит список всех имеющихся точек останова;

– `info watchpoints` – выводит список всех имеющихся контрольных выражений;

– `list` – выводит исходный код; в качестве параметра передаются название файла исходного кода, затем, через двоеточие, номер начальной и конечной строки;

– `next` – пошаговое выполнение программы, но, в отличие от команды `step`, не выполняет пошагово вызываемые функции;

– `print` – выводит значение какого-либо выражения (выражение передаётся в качестве параметра);

- run – запускает программу на выполнение;
 - set – устанавливает новое значение переменной
 - step – пошаговое выполнение программы;
 - watch – устанавливает контрольное выражение, программа остановится, как только значение контрольного выражения изменится;
9. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы.

Сначала я запустил отладчик для моей программы. Установил интересующую меня точку остановки. Запустил программу ожидая, что программа остановится на точке остановки. Узнал необходимые данные моей программы на текущем этапе ее исполнения путем ввода команд. Отобразил данные. Завершил программу, снял точки остановки.

10. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске.

В моем случае не было синтаксических ошибок. Были ошибки семантические. Компилятор

начал жаловаться на то, что программа по смыслу принимает указатель на char массив. В то время как я вводил не указатель, а прямое значение . Ошибка была исправлена

11. Назовите основные средства, повышающие понимание исходного кода программы.

- cscope - исследование функций, содержащихся в программе;
- lint -- критическая проверка программ, написанных на языке Си.

12. Каковы основные задачи, решаемые программой splint?

Splint- инструмент для статической проверки C-программ на наличие уязвимостей и ошибок