
Front matter

lang: ru-RU title: "Лабораторная работа №14" subtitle: " Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux" author: "Данилова Анастасия Сергеевна"

Formatting

toc-title: "Содержание" toc: true # Table of contents toc_depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt linestretch: 1.5 papersize: a4paper documentclass: screprpt polyglossia-lang: russian polyglossia-otherlangs: english mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase indent: true pdf-engine: lualatex header-includes: - \linepenalty=10 # the penalty added to the badness of each line within a paragraph (no associated penalty node) Increasing the value makes tex try to have fewer lines in the paragraph. - \interlinepenalty=0 # value of the penalty (node) added after each line of a paragraph. - \hyphenpenalty=50 # the penalty for line breaking at an automatically inserted hyphen - \exhyphenpenalty=50 # the penalty for line breaking at an explicit hyphen - \binoppenalty=700 # the penalty for breaking a line at a binary operator - \relpenalty=500 # the penalty for breaking a line at a relation - \clubpenalty=150 # extra penalty for breaking after first line of a paragraph - \widowpenalty=150 # extra penalty for breaking before last line of a paragraph - \displaywidowpenalty=50 # extra penalty for breaking before last line before a display math - \brokenpenalty=100 # extra penalty for page breaking after a hyphenated line - \predisplaypenalty=10000 # penalty for breaking before a display - \postdisplaypenalty=0 # penalty for breaking after a display - \floatingpenalty = 20000 # penalty for splitting an insertion (can only be split footnote in standard LaTeX) - \raggedbottom # or \flushbottom - \usepackage{float} # keep figures where there are in the text

- \floatplacement{figure}{H} # keep figures where there are in the text

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.

Задание

1. Создать файлы, в которых будут коды данные в теоретическом материале. 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.
2. С помощью gdb выполнить отладку программы calcul.
3. С помощью утилиты splint проанализируем коды файлов calculate.c и main.c.

Теоретическое введение

Стандартным средством для компиляции программ в ОС типа UNIX является GCC (GNU Compiler Collection). Это набор компиляторов для разного рода языков программирования (C, C++, Java, Фортран и др.). Работа с GCC производится при помощи одноимённой управляющей программы gcc, которая интерпретирует аргументы командной строки, определяет и осуществляет запуск нужного компилятора для входного файла.

Для сборки разрабатываемого приложения и собственно компиляции полезно воспользоваться утилитой make. Она позволяет автоматизировать процесс преобразования файлов программы из одной формы в другую, отслеживает взаимосвязи между файлами. Для работы с утилитой make необходимо в корне рабочего каталога с Вашим проектом создать файл с названием makefile или Makefile, в котором будут описаны правила обработки файлов Вашего программного комплекса.

Во время работы над кодом программы программист неизбежно сталкивается с появлением ошибок в ней. Использование отладчика для поиска и устранения ошибок в программе существенно облегчает жизнь программиста. В комплект программ GNU для ОС типа UNIX входит отладчик GDB (GNU Debugger). Для использования GDB необходимо скомпилировать анализируемый код программы таким образом, чтобы отладочная информация содержалась в результирующем бинарном файле. Для этого следует воспользоваться опцией -g компилятора gcc: gcc -c file.c -g

После этого для начала работы с gdb необходимо в командной строке ввести одноимённую команду, указав в качестве аргумента анализируемый бинарный файл: gdb file.o

Выполнение лабораторной работы

1. В домашнем каталоге создадим подкаталог ~/work/os/lab_prog с помощью mkdir
2. Создадим в нём файлы: calculate.h, calculate.c, main.c с помощью touch

Запишем в эти файлы программы, данные нам. (рис1) (рис2) (рис3) ▢

рис 1 ![] (C:\Users\nastd\Documents\прудн\4 модуль\операционные системы\Лаб 14\скрины\рис2.jpg) **рис 2** ▢ **рис 3**

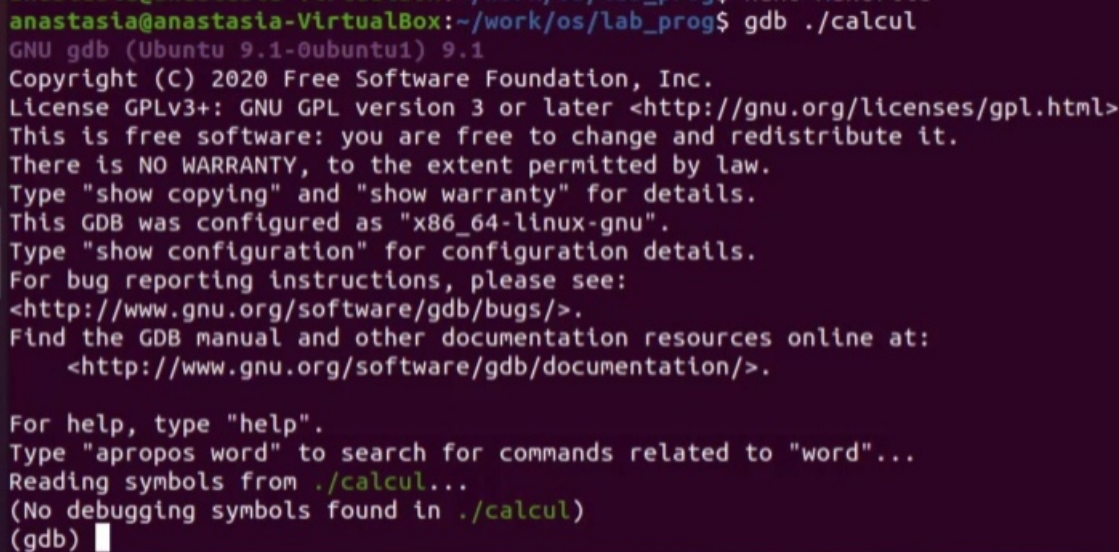
3. Выполним компиляцию программы посредством gcc: gcc -c calculate.c gcc -c main.c gcc calculate.o main.o -o calcul -lm (рис4) ![] (C:\Users\nastd\Documents\прудн\4 модуль\операционные системы\Лаб 14\скрины\рис4.jpg)

рис 4

4. Синтаксических ошибок не обнаружено
5. Создадим Makefile с данным нам кодом (рис 5) ▢ **рис 5**

6. С помощью gdb выполним отладку программы calcul:

- Запустим отладчик GDB, загрузив в него программу для отладки: `gdb ./calcul` (рис 6)

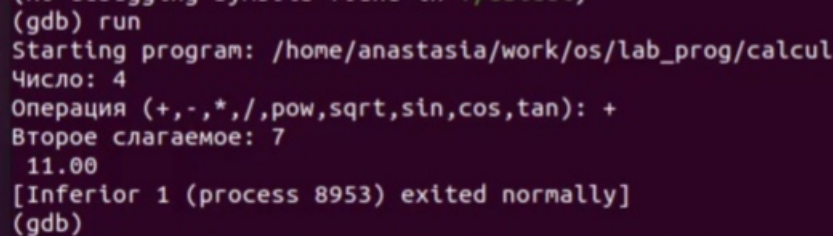


```
anastasia@anastasia-VirtualBox:~/work/os/lab_prog$ gdb ./calcul
GNU gdb (Ubuntu 9.1-0ubuntu1) 9.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
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 6

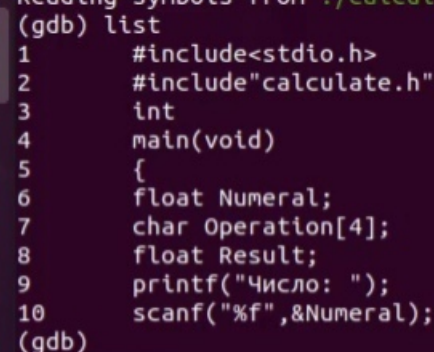
- Для запуска программы внутри отладчика введем команду `run` (рис 7)



```
(gdb) run
Starting program: /home/anastasia/work/os/lab_prog/calcul
Число: 4
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 7
11.00
[Inferior 1 (process 8953) exited normally]
(gdb)
```

рис 7

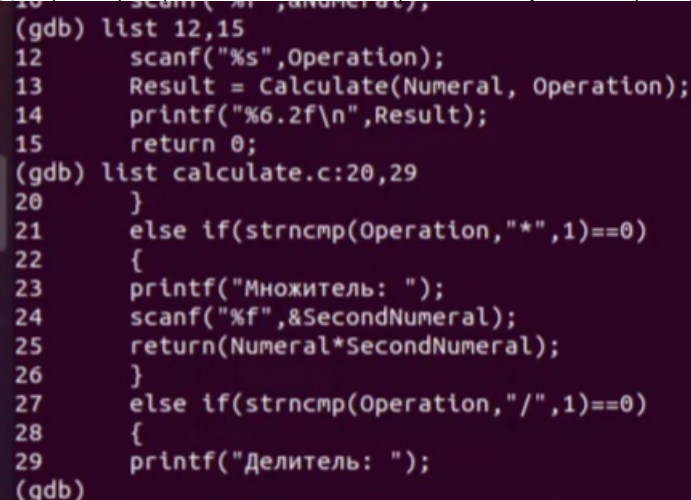
- Для постраничного просмотра исходного код используем команду `list` (рис 8)



```
(gdb) list
1      #include<stdio.h>
2      #include"calculate.h"
3      int
4      main(void)
5      {
6      float Numeral;
7      char Operation[4];
8      float Result;
9      printf("Число: ");
10     scanf("%f",&Numeral);
(gdb)
```

рис 8

- Для просмотра строк с 12 по 15 основного файла используем `list` с параметрами: `list 12,15` (рис 9)



```
(gdb) list 12,15
12     scanf("%s",Operation);
13     Result = Calculate(Numeral, Operation);
14     printf("%.2f\n",Result);
15     return 0;
(gdb) list calculate.c:20,29
20     }
21     else if(strncmp(Operation,"*",1)==0)
22     {
23     printf("Множитель: ");
24     scanf("%f",&SecondNumeral);
25     return(Numeral*SecondNumeral);
26     }
27     else if(strncmp(Operation,"/",1)==0)
28     {
29     printf("Делитель: ");
(gdb)
```

рис 9

- Для просмотра определённых строк не основного файла используем `list` с параметрами: `list calculate.c:20,29` (рис 9)
- Установим точку останова в файле calculate.c на строке номер 21: `list calculate.c:20,29 break 21` (рис 10)

```

20     }
21     else if(strncmp(Operation,"*",1)==0)
22     {
23     printf("Множитель: ");
24     scanf("%f",&SecondNumeral);
25     return(Numeral*SecondNumeral);
26     }
27     else if(strncmp(Operation,"/",1)==0)
(gdb) break 21
Breakpoint 1 at 0x1319: file calculate.c, line 21.
(gdb)

```

рис 10

- Выведем информацию об имеющихся в проекте точка останова: info breakpoints (рис 11)

```

(gdb) break 21
Breakpoint 1 at 0x1319: file calculate.c, line 21.
(gdb) info breakpoints
Num      Type           Disp Enb Address            What
1        breakpoint     keep y   0x00000000000001319 in Calculate
                                at calculate.c:21
(gdb)

```

рис 11

- Запустим программу внутри отладчика и убедимся, что программа остановится в момент прохождения точки останова: run - backtrace (рис 12)

```

(gdb) run
Starting program: /home/anastasia/work/os/lab_prog/calcul
Число: 6
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Breakpoint 1, Calculate (Numeral=6, Operation=0x7fffffffdf74 "*")
  at calculate.c:21
21     else if(strncmp(Operation,"*",1)==0)
(gdb) backtrace
#0 Calculate (Numeral=6, Operation=0x7fffffffdf74 "*") at calculate.c:21
#1 0x00005555555555bd in main () at main.c:13
(gdb)

```

рис 12

- Используем команду backtrace, которая покажет весь стек вызываемых функций от начала программы до текущего места. (рис 12)
- Посмотрим, чему равно на этом этапе значение переменной Numeral, введя: print Numeral. На экран должно быть выведено число 6. (рис 13)

```

(gdb) print Numeral
$1 = 6
(gdb) display Numeral
1: Numeral = 6
(gdb)

```

рис 13

- Сравним с результатом вывода на экран после использования команды: display Numeral (рис 13)
- Уберем точки останова: info breakpoints delete (рис 14)

```

(gdb) info breakpoints
Num      Type           Disp Enb Address            What
1        breakpoint     keep y   0x00005555555555319 in Calculate
                                at calculate.c:21
                                breakpoint already hit 1 time
(gdb) delete 1
(gdb)

```

рис 14

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c. (рис 15) (рис 16)


```

anastasia@anastasia-VirtualBox:~/work/os/lab_prog$ splint calculate.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:3:36: 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)
calculate.c:6:31: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:12:1: Return value (type int) ignored: scanf("%f", &Sec...
    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)
calculate.c:18:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:24:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:30:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:4: Dangerous equality comparison involving float types:
                    SecondNumeral == 0
    Two real (float, double, or long double) values are compared directly using
    == or != primitive. This may produce unexpected results since floating point
    representations are inexact. Instead, compare the difference to FLT_EPSILON
    or DBL_EPSILON. (Use -realcompare to inhibit warning)
calculate.c:32:7: Return value type double does not match declared type float:

anastasia@anastasia-VirtualBox:~/work/os/lab_prog$ splint main.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:3:36: 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:10:1: 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:12:1: Return value (type int) ignored: scanf("%s", Oper...

Finished checking --- 3 code warnings
anastasia@anastasia-VirtualBox:~/work/os/lab_prog$

```

рис 15

рис 16

Выводы

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

Библиография

<https://losst.ru/kak-polzovatsya-gdb>