

Лабораторная работа №13

Дисциплина: Операционные системы

Кондратьев Арсений Вячеславович

25.09.2022

Содержание

1	Цель работы	3
2	Теоретическое введение	4
3	Выполнение лабораторной работы	5
4	Выводы	13
5	Контрольные вопросы	14

1 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

2 Теоретическое введение

Процесс разработки программного обеспечения обычно разделяется на следующие этапы:

- планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;
- проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;
- непосредственная разработка приложения:
- кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);
- анализ разработанного кода;
- сборка, компиляция и разработка исполняемого модуля;
- тестирование и отладка, сохранение произведённых изменений;
- документирование.

3 Выполнение лабораторной работы

1. Создал подкаталог /lab_prog в котором создал файлы: calculate.h, calculate.c, main.c и написал в них реализацию простейшего калькулятора.(рис.3.1)

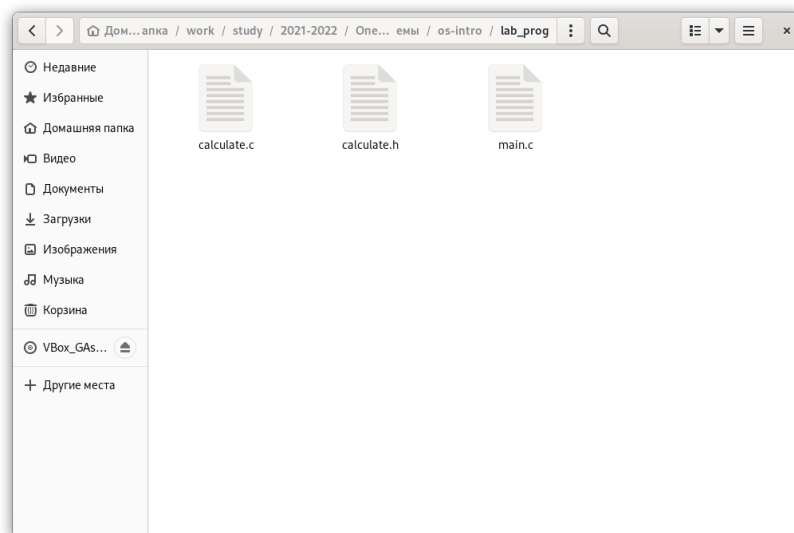


Figure 3.1: Рис. 1

2. Выполнил компиляцию программы посредством gcc(рис.3.2)

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

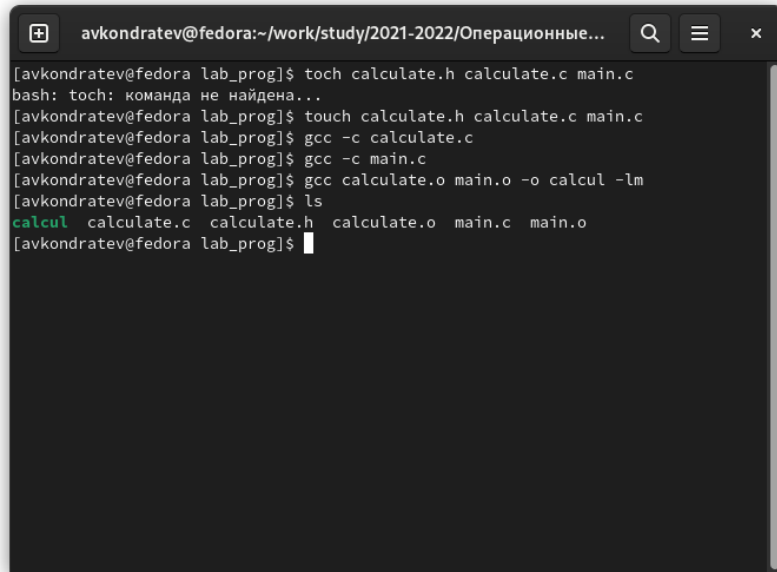
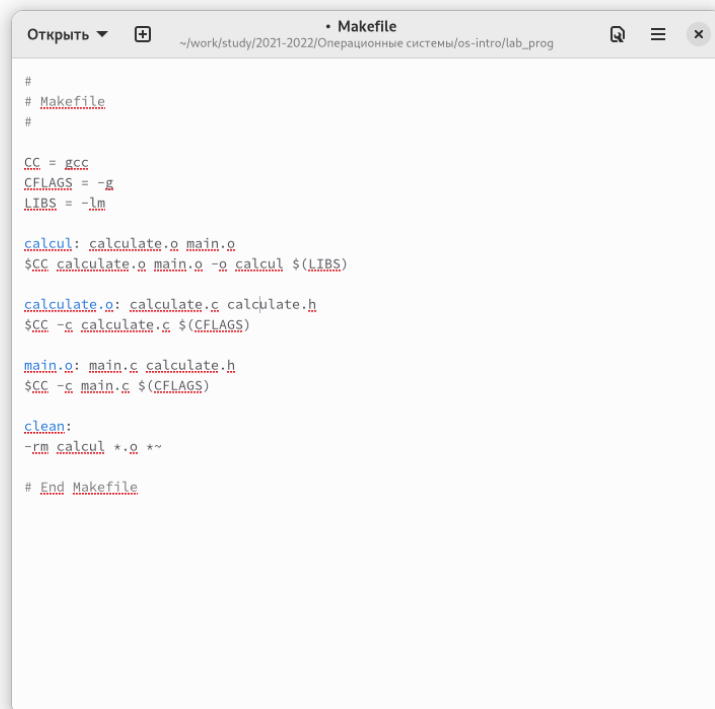
A terminal window with a dark background and light text. The window title is 'avkondratev@fedora:~/work/study/2021-2022/Операционные...'. The terminal shows a series of commands and their outputs. The first command is 'toch calculate.h calculate.c main.c', which results in an error message 'bash: toch: команда не найдена...'. The next command is 'touch calculate.h calculate.c main.c'. This is followed by 'gcc -c calculate.c' and 'gcc -c main.c'. Then, 'gcc calculate.o main.o -o calcul -lm' is executed. Finally, 'ls' is run, showing the files 'calcul', 'calculate.c', 'calculate.h', 'calculate.o', 'main.c', and 'main.o'. The prompt returns to '[avkondratev@fedora lab_prog]\$'.

Figure 3.2: Рис. 2

3. Создал Makefile в котором дописал опцию -g в CFLAGS для корректной компиляции объектных файлов и использовал переменную CC в которую помещен компилятор. Он необходим для автоматической компиляции calculate.c, main.c и объединения в один исполняемый файл calcul. Clean нужна для автоматического удаления файлов. Переменная CFLAGS отвечает за опции. Переменная LIBS отвечает за опции для объединения.(рис.3.3)



```
#
# Makefile
#

CC = gcc
CFLAGS = -g
LIBS = -lm

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

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

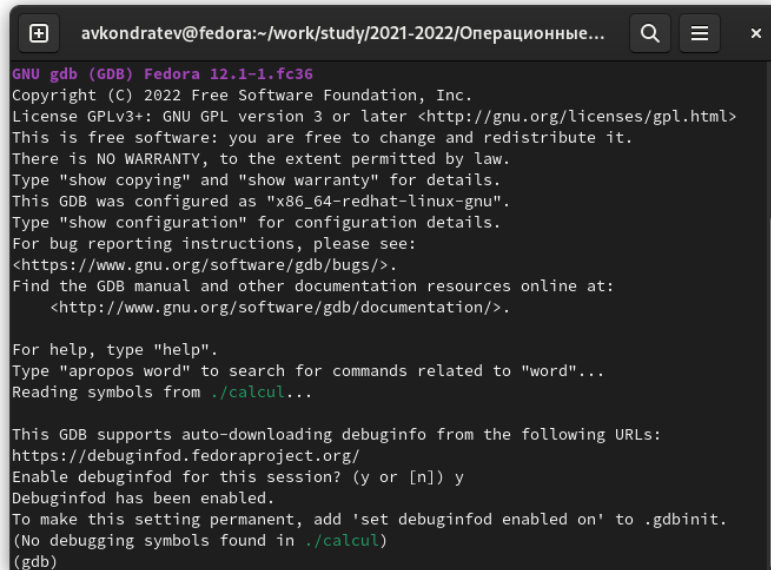
main.o: main.c calculate.h
$CC -c main.c $(CFLAGS)

clean:
-rm calcul *.o *~

# End Makefile
```

Figure 3.3: Рис. 3

4. Запустил отладчик GDB, загрузив в него программу для отладки(рис.3.4)



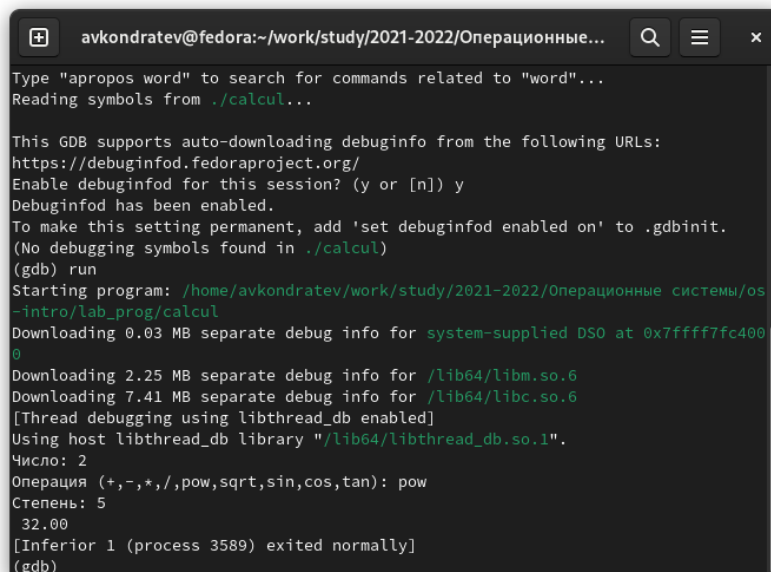
```
GNU gdb (GDB) Fedora 12.1-1.fc36
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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...

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
(No debugging symbols found in ./calcul)
(gdb)
```

Figure 3.4: Рис. 4

5. Запустил программу внутри отладчика(рис.3.5)



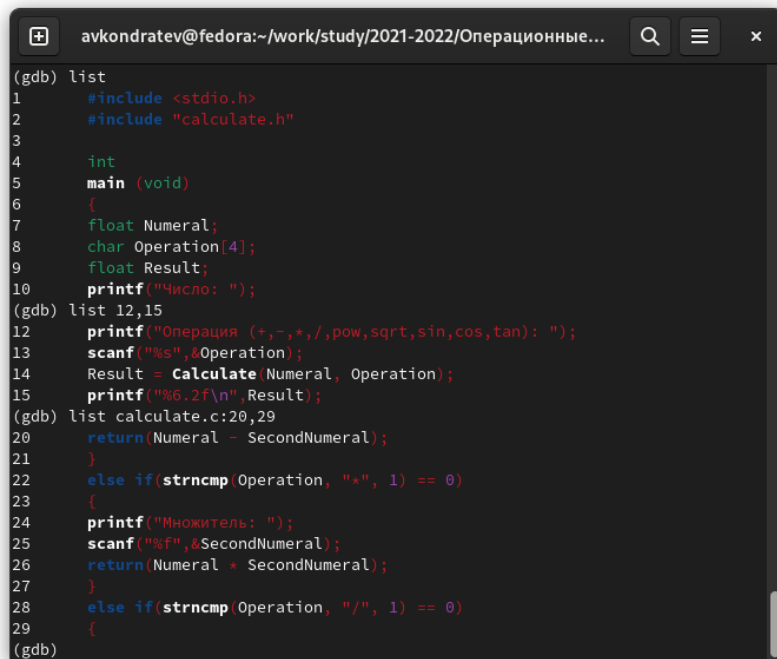
```
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/avkondratev/work/study/2021-2022/Операционные системы/os-intro/lab_prog/calcul
Downloading 0.03 MB separate debug info for system-supplied DS0 at 0x7ffff7fc400
0
Downloading 2.25 MB separate debug info for /lib64/libm.so.6
Downloading 7.41 MB separate debug info for /lib64/libc.so.6
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 2
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): pow
Степень: 5
32.00
[Inferior 1 (process 3589) exited normally]
(gdb)
```

Figure 3.5: Рис. 5

6. Использовал команду list для постраничного просмотра кода, затем просмотрел код с 12 по 15 строку, затем просмотрел определенные строки не

основного файла(рис.3.6)



```
(gdb) list
1  #include <stdio.h>
2  #include "calculate.h"
3
4  int
5  main (void)
6  {
7  float Numeral;
8  char Operation[4];
9  float Result;
10 printf("Число: ");
(gdb) list 12,15
12 printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
13 scanf("%s",&Operation);
14 Result = calculate(Numeral, Operation);
15 printf("%6.2f\n",Result);
(gdb) list calculate.c:20,29
20 return(Numeral - SecondNumeral);
21 }
22 else if(strncmp(Operation, "+", 1) == 0)
23 {
24 printf("Множитель: ");
25 scanf("%f",&SecondNumeral);
26 return(Numeral * SecondNumeral);
27 }
28 else if(strncmp(Operation, "/", 1) == 0)
29 {
(gdb)
```

Figure 3.6: Рис. 6

7. Установил точку останова в файле calculate.c на строке номер 21 и вывел информацию о точках останова с помощью info breakpoints(рис.3.7)

```
avkondratev@fedora:~/work/study/2021-2022/Операционные...
15     printf("%6.2f\n", Result);
(gdb) list calculate.c:20,29
20     return(Numeral - SecondNumeral);
21 }
22 else if(strncmp(Operation, "*", 1) == 0)
23 {
24     printf("Множитель: ");
25     scanf("%f", &SecondNumeral);
26     return(Numeral * SecondNumeral);
27 }
28 else if(strncmp(Operation, "/", 1) == 0)
29 {
(gdb) list calculate.c:20,27
20     return(Numeral - SecondNumeral);
21 }
22 else if(strncmp(Operation, "*", 1) == 0)
23 {
24     printf("Множитель: ");
25     scanf("%f", &SecondNumeral);
26     return(Numeral * SecondNumeral);
27 }
(gdb) break 21
Breakpoint 1 at 0x401247: file calculate.c, line 22.
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint      keep y   0x0000000000401247 in Calculate
                                           at calculate.c:22
(gdb)
```

Figure 3.7: Рис. 7

8. Убедился, что программа останавливается в момент точки останова, с помощью `print Numeral` выяснил что эта переменная равна 5 на тот момент и сравнил результат с командой `display Numeral`(рис.3.8)

```
avkondratev@fedora:~/work/study/2021-2022/Операционные сист...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 2
3.00
[Inferior 1 (process 4155) exited normally]
(gdb) break 21
Note: breakpoint 1 also set at pc 0x401247.
Breakpoint 2 at 0x401247: file calculate.c, line 22.
(gdb) break 20
Breakpoint 3 at 0x401234: file calculate.c, line 20.
(gdb) run
Starting program: /home/avkondratev/work/study/2021-2022/Операционные системы/os-int
ro/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 1

Breakpoint 3, Calculate (Numeral=5, Operation=0x7fffffffdd74 "-") at calculate.c:20
20     return(Numeral - SecondNumeral);
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb)
```

Figure 3.8: Рис. 8

9. Убрал точки останова(рис.3.9)

```
avkondratev@fedora:~/work/study/2021-2022/Операционные сист...
Breakpoint 3 at 0x401234: file calculate.c, line 20.
(gdb) run
Starting program: /home/avkondratev/work/study/2021-2022/Операционные системы/os-int
ro/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 1

Breakpoint 3, Calculate (Numeral=5, Operation=0x7fffffffdd74 "-") at calculate.c:20
20     return(Numeral - SecondNumeral);
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint        keep y   0x0000000000401247 in Calculate at calculate.c:22
2     breakpoint        keep y   0x0000000000401247 in Calculate at calculate.c:22
3     breakpoint        keep y   0x0000000000401234 in Calculate at calculate.c:20
      breakpoint already hit 1 time
(gdb) delte 1 2 3
Undefined command: "delite". Try "help".
(gdb) delete 1 2 3
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb)
```

Figure 3.9: Рис. 9

10. С помощью утилиты `splint` проанализировал коды файлов `calculate.c` и `main.c`(рис.3.10)

[illegible]

Figure 3.10: Рис. 10

4 Выводы

Я 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

5 Контрольные вопросы

1. С помощью справочной команды `man`
2. – планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;
– проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;
– непосредственная разработка приложения:
– кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);
– анализ разработанного кода;
– сборка, компиляция и разработка исполняемого модуля;
– тестирование и отладка, сохранение произведённых изменений;
– документирование.
3. Суффикс - это расширение
`gcc -c main.c`
Таким образом, `gcc` по расширению (суффиксу) `.c` распознает тип файла для компиляции и формирует объектный модуль — файл с расширением `.o`
4. Для запуска программ, написанных на других языках программирования
5. Для исполнения мэйкфайлов
6. Он состоит из списка зависимостей и команд, которые нужно выполнить

7. 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`
8. – `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 передается в качестве параметра);
- run – запускает программу на выполнение;
- set – устанавливает новое значение переменной
- step – пошаговое выполнение программы;
- watch – устанавливает контрольное выражение

9. Просмотрел код программы, поставил точку останова, запустил программу, на точке останова узнал значение переменной, закончил программу
10. Синтаксические ошибки отсутствовали
11. cscope – исследование функций, содержащихся в программе, lint – критическая проверка программ, написанных на языке Си
12. Splint анализирует программный код, проверяет корректность задания аргументов использованных в программе функций и типов возвращаемых значений.