**Отчет по лабораторной работе № 10** по курсу “Фундаментальная информатика”

Студент группы М80-103Б-21 Белоносов Кирилл Алексеевич, № по списку 3

Контакты почта kirillbelonosov@yandex.ru, telegram: @KiRiLLBElNOS

Работа выполнена: «4» ноября 2021г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан « » \_\_\_\_\_\_\_\_\_20\_\_ г., итоговая оценка \_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Тема:** Отладчик системы программирования ОС UNIX

1. **Цель работы:** Познакомится с отладчиком gdb и произвести отладку тестовой программы.
2. **Задание:** Изучить основные команды отладчика gdb и отладить программу из лабораторной работы №9.
3. **Оборудование** (студента):

*Процессор Intel Core i7-1165G7 @ 4x2.8GH с ОП 16384 Мб, НМД 512 Гб. Монитор 1920x1080*

1. **Программное обеспечение (**студента**):**

Операционная система семейства: *linux*, наименование: *ubuntu*, версия 20.04.3 *LTS*

интерпретатор команд: *bash* версия *5.0*.17(1)

Система программирования Visual studio code

Редактор текстов *emacs* версия *27.1*

Утилиты операционной системы --

Прикладные системы и программы

Местонахождение и имена файлов программ и данных на домашнем компьютере –

1. **Идея, метод, алгоритм**

Основной задачей является знакомство с отладчиком gdb обратимся к тексту лабораторной работы, а также к дополнительным источникам. На примере программы из 9 лабораторной рассмотрим основные команды gdb. Также посмотрим команды терминала для вызова отладчика.

**Список команд:**

help

list

break

run

set args

print

next

step

set var

ptype

backtrace

continue

quit

info locals

info breakpoints

del

watch

kill

**Примечание:**

Нажатие Enter приводит к выполнению последней команды.

1. **Сценарий выполнения работы.**
2. Изучим материалы по отладчику gdb
3. Запустим отладчик
4. Отработаем основные команды
5. Скопируем протокол
6. **Распечатка протокола**

kirill@kirill-Vostro-5402:~/c/lab9/Test$ gcc main.c -g -lm -Werror -o main

kirill@kirill-Vostro-5402:~/c/lab9/Test$ gdb main

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2

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/>.

--Type <RET> for more, q to quit, c to continue without paging--

For help, type "help".

Type "apropos word" to search for commands related to "word"...

Reading symbols from main...

(gdb) help

List of classes of commands:

aliases -- Aliases of other commands.

breakpoints -- Making program stop at certain points.

data -- Examining data.

files -- Specifying and examining files.

internals -- Maintenance commands.

obscure -- Obscure features.

running -- Running the program.

stack -- Examining the stack.

status -- Status inquiries.

support -- Support facilities.

--Type <RET> for more, q to quit, c to continue without paging--

tracepoints -- Tracing of program execution without stopping the program.

user-defined -- User-defined commands.

Type "help" followed by a class name for a list of commands in that class.

Type "help all" for the list of all commands.

Type "help" followed by command name for full documentation.

Type "apropos word" to search for commands related to "word".

Type "apropos -v word" for full documentation of commands related to "word".

Command name abbreviations are allowed if unambiguous.

(gdb) list

46 c = a / b;

47 if (b >= 0) {

48 return floor(c);

49 } else {

50 return ceil(c);

51 }

52 }

53

54 int mod(int a, int b)

55 {

(gdb)

56 return a - divs(a, b) \* b;

57 }

58

59 int main(void)

60 {

61 int i0;

62 int j0;

63 int l0;

64 int k = 0;

65 scanf("%d %d %d", &i0, &j0, &l0);

(gdb)

66 if (((i0 - 10) \* (i0 - 10) + (j0 - 10) \* (j0 - 10) <= 100) && ((i0 - 10) \* (i0 - 10) + (j0 - 10) \* (j0 - 10) > 25)) {

67 printf("Yes\n%d %d %d %d\n", i0, j0, l0, k);

68 return 0;

69 }

70 int i = i0;

71 int j = j0;

72 int l = l0;

73 int iLast = i0;

74 int jLast = j0;

75 int lLast = l0;

(gdb)

76 for (; k < 50; ++k) {

77 i = max(min(iLast + jLast - lLast - k, iLast - jLast + lLast - k), min(k + iLast - jLast - lLast, k - iLast - jLast + lLast));

78 j = jLast + mod(lLast \* sign(jLast), 20) + mod(k \* sign(iLast), 10);

79 l = ABS(iLast - jLast + lLast - k) \* sign(iLast) \* sign(jLast);

80 iLast = i;

81 jLast = j;

82 lLast = l;

83 if (((i - 10) \* (i - 10) + (j - 10) \* (j - 10) <= 100) && ((i - 10) \* (i - 10) + (j - 10) \* (j - 10) > 25)) {

84 printf("Yes\n%d %d %d %d\n", i, j, l, k + 1);

85 return 0;

(gdb)

86 }

87 }

88 printf("No\n%d %d %d %d\n", i, j, l, k);

89 return 0;

90 }

(gdb) set args

(gdb) set args -Wextra -Wall

(gdb) break 64

Breakpoint 1 at 0x12f8: file main.c, line 64.

(gdb) break 76

Breakpoint 2 at 0x13b9: file main.c, line 76.

(gdb) run

Starting program: /home/kirill/c/lab8/Test/main -Wextra -Wall

Breakpoint 1, main () at main.c:64

64 int k = 0;

(gdb) info locals

i0 = -8586

j0 = 32767

l0 = 1431655869

k = 21845

i = -135901240

j = 32767

l = 1431655792

iLast = 21845

jLast = 0

lLast = 0

(gdb) next

65 scanf("%d %d %d", &i0, &j0, &l0);

(gdb)

1 -30 1

66 if (((i0 - 10) \* (i0 - 10) + (j0 - 10) \* (j0 - 10) <= 100) && ((i0 - 10) \* (i0 - 10) + (j0 - 10) \* (j0 - 10) > 25)) {

(gdb) info locals

i0 = 1

j0 = -30

l0 = 1

k = 0

i = -135901240

j = 32767

l = 1431655792

iLast = 21845

jLast = 0

lLast = 0

(gdb) continue

Continuing.

Breakpoint 2, main () at main.c:76

76 for (; k < 50; ++k) {

(gdb) watch k

Hardware watchpoint 3: k

(gdb) next

77 i = max(min(iLast + jLast - lLast - k, iLast - jLast + lLast - k), min(k + iLast - jLast - lLast, k - iLast - jLast + lLast));

(gdb)

78 j = jLast + mod(lLast \* sign(jLast), 20) + mod(k \* sign(iLast), 10);

(gdb) step

sign (a=-30) at main.c:24

24 {

(gdb)

25 if (a < 0) {

(gdb)

26 return -1;

(gdb)

32 }

(gdb)

main () at main.c:78

78 j = jLast + mod(lLast \* sign(jLast), 20) + mod(k \* sign(iLast), 10);

(gdb)

mod (a=-30, b=30) at main.c:55

55 {

(gdb)

56 return a - divs(a, b) \* b;

(gdb)

divs (a=0, b=0) at main.c:44

44 {

(gdb) ptype a

type = double

(gdb) ptype b

type = double

(gdb) step

46 c = a / b;

(gdb)

47 if (b >= 0) {

(gdb)

48 return floor(c);

(gdb)

Hardware watchpoint 3: k

Old value = 0

New value = 1

0x000055555555551d in main () at main.c:76

76 for (; k < 50; ++k) {

(gdb) print i

$1 = 30

(gdb) next

77 i = max(min(iLast + jLast - lLast - k, iLast - jLast + lLast - k), min(k + iLast - jLast - lLast, k - iLast - jLast + lLast));

(gdb) step

min (a=-30, b=10) at main.c:35

35 {

(gdb) backtrace

#0 min (a=-30, b=10) at main.c:35

#1 0x00005555555553e5 in main () at main.c:77

(gdb) step

36 if (a <= b) {

(gdb)

39 return b;

(gdb)

41 }

(gdb)

min (a=74, b=-50) at main.c:35

35 {

(gdb)

36 if (a <= b) {

(gdb)

39 return b;

(gdb)

41 }

(gdb)

max (a=50, b=8) at main.c:15

15 {

(gdb)

16 if (a >= b) {

(gdb)

17 return a;

(gdb)

21 }

(gdb)

main () at main.c:78

78 j = jLast + mod(lLast \* sign(jLast), 20) + mod(k \* sign(iLast), 10);

(gdb) next

79 l = ABS(iLast - jLast + lLast - k) \* sign(iLast) \* sign(jLast);

(gdb)

80 iLast = i;

(gdb)

81 jLast = j;

(gdb)

82 lLast = l;

(gdb)

83 if (((i - 10) \* (i - 10) + (j - 10) \* (j - 10) <= 100) && ((i - 10) \* (i - 10) + (j - 10) \* (j - 10) > 25)) {

(gdb) print i

$2 = 8

(gdb) print j

$3 = 2

(gdb) print l

$4 = -8

(gdb) info breakpoints

Num Type Disp Enb Address What

1 breakpoint keep y 0x00005555555552f8 in main at main.c:64

breakpoint already hit 1 time

2 breakpoint keep y 0x00005555555553b9 in main at main.c:76

breakpoint already hit 1 time

3 hw watchpoint keep y k

breakpoint already hit 1 time

(gdb) del 1

(gdb) del 2

(gdb) set var k = 50

(gdb) next

84 printf("Yes\n%d %d %d %d\n", i, j, l, k + 1);

(gdb)

Yes

8 2 -8 51

85 return 0;

(gdb) kill

Kill the program being debugged? (y or n) y

[Inferior 1 (process 24262) killed]

(gdb) quit

1. **Выводы**

В данной лабораторной работе я познакомился с отладчиком gdb и изучил основные команды, а также научился отлаживать программу. Данная лабораторная работа интересна тем, что приобретенный опыт в отладке программы поможет мне в дальнейшем выполнении лабораторных работ, в частности в нахождении ошибок.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_