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

Студент группы М80-103Б-21 Березнев Никита Вадимович, № по списку 4

Контакты e-mail: nikita.berezneff@yandex.ru

Работа выполнена: «11» ноября 2021г.

Преподаватель: каф. 805 Севастьянов Виктор Сергеевич

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

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

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

1. **Цель работы:** Изучение механизмов работы отладчика gdb, а также его команд в ОС UNIX.
2. **Задание:** В ходе выполнения работы необходимо проделать и запротоколировать как минимум все описанные действия над некоторой программой на изучаемом языке программирования, в том числе и со специально внесенными ошибками.
3. **Оборудование:**

Процессор *Intel Core i3-3217U @ 4x 1.80GH* с ОП *7851* Мб, НМД *512* Гб. Монитор *1366x768*

1. **Программное обеспечение:**

Операционная система семейства: *linux*, наименование: *ubuntu*, версия *18.10 cosmic*

интерпретатор команд: *bash* версия *4.4.19*.

Система программирования -- версия --

Утилиты операционной системы:

Прикладные системы и программы --

Местонахождение и имена файлов программ и данных на домашнем компьютере --

**6. Идея, метод, алгоритм** решения задачи**:**

- Протестировать основные команды отладчика на корректно работающей программе, после чего, специально внеся ошибки в исходный код программы, вывести стек возникших ошибок. Затем с помощью точек останова и переприсваивания значений переменных пошагово разобрать программу и избавиться от источника ошибок.

**7. Сценарий выполнения работы:**

- Создать корректно работающую программу как минимум с одной функцией (для проверки команд step и next)

- Скомпилировать программу. Затем запустить сеанс отладки программы.

- После этого специально добавить в программу несколько ошибок. Затем скомпилировать программу и запустить сеанс отладки этой программы.

- Вывести стек ошибок. Расставить точки останова на тех строках, где встречаются ошибки.

- С помощью команды print <выражение> или info locals вывести значение переменной, создающей ошибку. Изменить значение переменной с помощью команды set var <имя> = <выражение> таким образом, чтобы программа работала корректно.

- Отключить точку останова на строке с измененной переменной в случае, если ошибки в этом месте не возникает. Установить точки останова с условием. С помощью команды continue дойти до ближайшей точки останова.

- Исправив все ошибки программы, удалить точки останова.

**8. Распечатка протокола:**

nikita@nikita-Virtual-Machine:~$ cat > Lab\_10.c

^C

nikita@nikita-Virtual-Machine:~$ emacs Lab\_10.c

nikita@nikita-Virtual-Machine:~$ cat Lab\_10.c

#include <stdio.h>

int sum(int a,int b)

{

return (a + b);

}

int main(void)

{

int a;

int b;

int a\_new;

int b\_new;

int count\_old;

int count\_new;

int i;

count\_old = 3;

scanf("%d", &a);

scanf("%d", &b);

for (i = 1 ; i < 10 ; i++) {

a\_new = sum(b, a) \* i;

b\_new = (b + a) % i;

count\_new = (count\_old + (a\_new \* b\_new)) % i;

count\_old = count\_new;

a = a\_new;

b = b\_new;

}

printf("%d %d %d\n", a, b, count\_new);

return 0;

}

nikita@nikita-Virtual-Machine:~$ gcc -std=c99 -Wall -Wextra -Werror -Wshadow -Wstrict-prototypes -Wold-style-definition -pedantic -g -Wno-unused-variable -Wno-sign-compare -Wno-unused-result -Wno-unused-parameter -o Lab\_10.out Lab\_10.c

nikita@nikita-Virtual-Machine:~$ gdb Lab\_10.out

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/>.

For help, type "help".

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

Reading symbols from Lab\_10.out...

(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.

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) run

Starting program: /home/nikita/Lab\_10.out

12 3

5726007 4 0

[Inferior 1 (process 14896) exited normally]

(gdb) quit

nikita@nikita-Virtual-Machine:~$ emacs Lab\_10.c

nikita@nikita-Virtual-Machine:~$ gcc -std=c99 -Wall -Wextra -Werror -Wshadow -Wstrict-prototypes -Wold-style-definition -pedantic -g -Wno-unused-variable -Wno-sign-compare -Wno-unused-result -Wno-unused-parameter -o Lab\_10.out Lab\_10.c

nikita@nikita-Virtual-Machine:~$ gdb Lab\_10.out

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/>.

For help, type "help".

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

Reading symbols from Lab\_10.out...

(gdb) list 1

1 #include <stdio.h>

2

3 int sum(int a,int b)

4 {

5 return (a + b);

6 }

7

8 int main(void)

9 {

10 int a;

(gdb)

11 int b;

12 int a\_new;

13 int b\_new;

14 int count\_old;

15 int count\_new;

16 int i;

17

18 count\_old = 3;

19

20 scanf("%d", &a);

(gdb)

21 scanf("%d", &b);

22

23 for (i = 1 ; i < 10 ; i++) {

24 a\_new = sum(b, a) \* i;

25 b\_new = (b % a) \* i;

26 count\_new = (count\_old + (a\_new \* b\_new)) % (i - 5);

27 count\_old = count\_new;

28 a = a\_new;

29 b = b\_new;

30 }

(gdb) run

Starting program: /home/nikita/Lab\_10.out

0 156

Program received signal SIGFPE, Arithmetic exception.

0x000055555555521b in main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb) backtrace

#0 0x000055555555521b in main () at Lab\_10.c:25

(gdb) continue

Continuing.

Program terminated with signal SIGFPE, Arithmetic exception.

The program no longer exists.

(gdb) break 25

Breakpoint 1 at 0x555555555214: file Lab\_10.c, line 25.

(gdb) run

Starting program: /home/nikita/Lab\_10.out

0 156

Breakpoint 1, main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb) ptype a

type = int

(gdb) print a

$3 = 0

(gdb) set var a = 12

(gdb) print a

$4 = 12

(gdb) next

26 count\_new = (count\_old + (a\_new \* b\_new)) % (i - 5);

(gdb) next 5

24 a\_new = sum(b, a) \* i;

(gdb) step

sum (a=156, b=0) at Lab\_10.c:4

4 {

(gdb)

5 return (a + b);

(gdb)

6 }

(gdb)

Breakpoint 1, main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb) next 6

24 a\_new = sum(b, a) \* i;

(gdb) next

Breakpoint 1, main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb) continue

Continuing.

Breakpoint 1, main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb)

Continuing.

Breakpoint 1, main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb)

Continuing.

Program received signal SIGFPE, Arithmetic exception.

0x000055555555523b in main () at Lab\_10.c:26

26 count\_new = (count\_old + (a\_new \* b\_new)) % (i - 5);

(gdb) next

Program terminated with signal SIGFPE, Arithmetic exception.

The program no longer exists.

(gdb) break 26 if i == 5

Breakpoint 2 at 0x555555555226: file Lab\_10.c, line 26.

(gdb) info break

Num Type Disp Enb Address What

1 breakpoint keep y 0x0000555555555214 in main at Lab\_10.c:25

breakpoint already hit 5 times

2 breakpoint keep y 0x0000555555555226 in main at Lab\_10.c:26

stop only if i == 5

(gdb) run

Starting program: /home/nikita/Lab\_10.out

0 156

Breakpoint 1, main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb) ptype a

type = int

(gdb) print a

$5 = 0

(gdb) set var a = 7

(gdb) disable 1

(gdb) info break

Num Type Disp Enb Address What

1 breakpoint keep n 0x0000555555555214 in main at Lab\_10.c:25

breakpoint already hit 1 time

2 breakpoint keep y 0x0000555555555226 in main at Lab\_10.c:26

stop only if i == 5

(gdb) print a

$6 = 7

(gdb) next

26 count\_new = (count\_old + (a\_new \* b\_new)) % (i - 5);

(gdb) continue

Continuing.

Breakpoint 2, main () at Lab\_10.c:26

26 count\_new = (count\_old + (a\_new \* b\_new)) % (i - 5);

(gdb) info locals

a = 3888

b = 48

a\_new = 19680

b\_new = 240

count\_old = 0

count\_new = 0

i = 5

(gdb) ptype i

type = int

(gdb) set var i = 6

(gdb) continue

Continuing.

10281600 120960 -1

[Inferior 1 (process 15104) exited normally]

(gdb) enable 1

(gdb) info break

Num Type Disp Enb Address What

1 breakpoint keep y 0x0000555555555214 in main at Lab\_10.c:25

breakpoint already hit 1 time

2 breakpoint keep y 0x0000555555555226 in main at Lab\_10.c:26

stop only if i == 5

breakpoint already hit 1 time

(gdb) run

Starting program: /home/nikita/Lab\_10.out

0 156

Breakpoint 1, main () at Lab\_10.c:25

25 b\_new = (b % a) \* i;

(gdb) set var a = 5

(gdb) delete 1

(gdb) info break

Num Type Disp Enb Address What

2 breakpoint keep y 0x0000555555555226 in main at Lab\_10.c:26

stop only if i == 5

(gdb) continue

Continuing.

Breakpoint 2, main () at Lab\_10.c:26

26 count\_new = (count\_old + (a\_new \* b\_new)) % (i - 5);

(gdb) set var i = 6

(gdb) delete 2

(gdb) info break

No breakpoints or watchpoints.

(gdb) continue

Continuing.

9858240 60480 -2

[Inferior 1 (process 15160) exited normally]

**9. Дневник отладки** должен содержать дату и время сеансов отладки, и основные события(ошибки в сценарии и программе,

нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
| 1 | дом | 11.10.21 | 12:46 | Точка останова работает некорректно, присваивая переменной из условия числовое значение из условия остановки | Замена “=” на “==” в условии остановки отладки | Когда-нибудь я забуду Pascal и запомню, что условие равенства обозначается “==”, а не “=”, как в Pascal |

1. **Замечания автора** по существу работы

Работу считаю необычайно полезной, поскольку возможность пошагово изучить код своей программы (или же отдельные участки программы) очень сильно упрощает процесс поиска ошибок в коде. А возможность переприсваивать новые значения переменным во время отладки программы не только помогает удостовериться в том, что ошибку порождало неправильное значение переменной, но и значительно сократить время на исправлении своего кода.

P.S. отсутствие чекера в этой работе восстановило моральное здоровье на 10 единиц.

1. **Выводы**

Работа мне понравилась. Было довольно занимательно изучать команды отладчика и применять их для обнаружения и исправления ошибок в коде. Благодаря данной работе я освоил команды отладчика gdb, благодаря чему я смогу в дальнейшем значительно быстрее найти ошибку в программе и искоренить ее. Информация, полученная мной в ходе выполнения данной лабораторной работы, поможет мне в дальнейшем при запуске программ, написанных мной.

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