Отчет по лабораторной работе №10 по курсу_____Фундаментальная информатика_____

Студент группы _____М8О-102Б-21_____, № по списку _9_- Кажекин Денис Андреевич

Контакты www, e-mail, icq, skype____deniskazhekin@mail.ru_____

Работа выполнена:

Преподаватель: _Никулин Сергей Петрович_____ каф. 806__

Входной контроль знаний с оценкой _____

Отчет сдан « » _____201 __ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Отладчик системы программирования ОС Unix
2. **Цель работы:** Научиться отлаживать простейшие программы, написанные на языке Си
3. **Задание -**
4. **Оборудование:**

Оборудование ПЭВМ студента, если использовалось:

Процессор: Apple M1, с ОП 8192 Мб, НМД 262144 Мб. Монитор: Retina 13,3; IPS 2560 x 1600.

5. Программное обеспечение ЭВМ студента, если использовалось

Операционная система семейства Linux, наименование Ubuntu, версия ____20.04.3 LTS____интерпретатор команд _bash_ версия __5.0.17(1)__

Редактор текстов GNU emacs, версия 27.2

Прикладные системы и программы: Компилятор языка C – gcc. отлалчик gdb.

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

- Скомпилируем программу (Из лабораторной работы №9) с ключом -g
- Посмотрим на значения переменных и убедимся в правильности работы программы

Команда gdb	Описание команды
help [раздел]	Подсказка по разделу отладчика. Без параметров выводит список разделов.
list [имя фукнции/файла:] [номер строки]	Распечатка текста функции/процедуры/файла или всей программы, начиная с указанной строки. По умолчанию распечатываются 10 строк программы. Распечатываемый файл становится текущим файлом исходного текста отлаживаемой программы.
break [номер строки/имя функции]	Задание точки остановки на строке/функции текущего исходного файла программы
run [параметры]	Запуск программы на выполнение. Могут указываться необязательные параметры командной строки и операции перенаправления

	ввода-вывода. gdb запоминает параметры и подставляет их для дальнейших вызовах run.
set args [параметры]	Предварительная установка параметров командной строки.
show args	Вывод параметров командной строки.
print [выражение]	Печать значения выражения, которое может включать и переменные, и вызовы функций программы.
next [n]	Выполнение очередной строки программы при пошаговой трассировке (процедуры и функции не трассируются, а выполняются за один такт). Необязательный параметр n указывает число строк программы для выполнения. По умолчанию n = 1.
step [n]	Выполнение очередной строки программы (с трассировкой процедур и функций). Перед выполнением next/step программа должна быть запущена командой run.
set var [имя] = [выражение]	Присваивание значения переменной.
ptype [имя переменной]	Выводит тип переменной.
backtrace или bt	Распечатка содержимого стека вызовов.
continue	Продолжение выполнения программы после остановки.
quit	Выход из отладчика.

7. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].
    1. Скомпилировать программу lr10.c (Программа из ЛР №9) с ключом -g
    2. Запустить отладку gdb
    3. Проверим значения переменных i, j, l на избранных итерациях
    4. Присвоить собственные значения переменным и посмотреть на правильность расчёта координаты точки при произвольных координатах, чтобы быть уверенными в правильности работы программы
    5. Ввести такие координаты вручную, чтобы проверить, выполняется ли вообще условие на попадание точки в лунку

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

Допущен к выполнению работы. **Подпись преподавателя**

_____

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
denis@ubuntu:~$ gcc -g lr10.c -lm
denis@ubuntu:~$ gdb a.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 "aarch64-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 a.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) help stack
Examining the stack.
The stack is made up of stack frames.  Gdb assigns numbers to stack frames
counting from zero for the innermost (currently executing) frame.

At any time gdb identifies one frame as the "selected" frame.
Variable lookups are done with respect to the selected frame.
When the program being debugged stops, gdb selects the innermost frame.
The commands below can be used to select other frames by number or address.

List of commands:
```

```
backtrace -- Print backtrace of all stack frames, or innermost COUNT frames.
bt -- Print backtrace of all stack frames, or innermost COUNT frames.
down -- Select and print stack frame called by this one.
faas -- Apply a command to all frames (ignoring errors and empty output).
frame -- Select and print a stack frame.
frame address -- Select and print a stack frame by stack address.
frame apply -- Apply a command to a number of frames.
frame apply all -- Apply a command to all frames.
frame apply level -- Apply a command to a list of frames.
frame function -- Select and print a stack frame by function name.
frame level -- Select and print a stack frame by level.
frame view -- View a stack frame that might be outside the current backtrace.
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) list lr10.c:42
37   int main(){
38       int i = 13, j =-9, l = -4, a, b, c;
39       for(int v = 0; v < 51; v++){
40            if (((sqrt(((-10 - i) * (-10 - i)) + ((-10 - j) * (-10 - j))) <= 10) &&
41                 (sqrt(((-20 - i) * (-20 - i)) + ((-20 - j) * (-20 - j))) <= 10))) {
42                printf("Bullseye!\n Time: %d\n Coordinates: %d, %d\n Param: %d\n", v, i, j, l);
43                break;
44            }
45            else{
46                printf("Miss!\t Time:%d\t Coordinates:%d,%d\t Param:%d\n",v,i,j,l);
(gdb) break 50
Breakpoint 1 at 0xba0: file lr10.c, line 50.
(gdb) break 51
Breakpoint 2 at 0xba8: file lr10.c, line 51.
(gdb) break 52
Breakpoint 3 at 0xbb0: file lr10.c, line 52.
(gdb) run
Starting program: /home/denis/a.out
Miss!      Time:0   Coordinates:13,-9            Param:-4

Breakpoint 1, main () at lr10.c:50
50                i = a;
(gdb) step

Breakpoint 2, main () at lr10.c:51
51                j = b;
(gdb) step

Breakpoint 3, main () at lr10.c:52
52                l = c;
(gdb) step
39       for(int v = 0; v < 51; v++){
(gdb) print i
$1 = 1
(gdb) print j
$2 = 0
(gdb) print l
$3 = -10
(gdb) continue
```

```
Continuing.
Miss!      Time:1   Coordinates:1,0   Param:-10

Breakpoint 1, main () at lr10.c:50
50              i = a;
(gdb) step

Breakpoint 2, main () at lr10.c:51
51              j = b;
(gdb)

Breakpoint 3, main () at lr10.c:52
52              l = c;
(gdb)
39      for(int v = 0; v < 51; v++){
(gdb) print i
$4 = 22
(gdb) print l
$6 = -9
(gdb) print j
$7 = 0
(gdb) set var i=1
(gdb) set var j=1
(gdb) set var l=1
(gdb) continue
Continuing.
Miss!      Time:2   Coordinates:1,1   Param:1

Breakpoint 1, main () at lr10.c:50
50              i = a;
(gdb) step

Breakpoint 2, main () at lr10.c:51
51              j = b;
(gdb)

Breakpoint 3, main () at lr10.c:52
52              l = c;
(gdb)
39      for(int v = 0; v < 51; v++){
(gdb) print i
$8 = 2
(gdb) print j
$9 = 3
(gdb) print l
$10 = -8
(gdb) ptype i
type = int
(gdb) set args 2
(gdb) show args
Argument list to give program being debugged when it is started is "2".
(gdb) backtrace
#0  main () at lr10.c:39
(gdb) continue
```

```
Continuing.
Miss!      Time:3   Coordinates:2,3   Param:-8

Breakpoint 1, main () at lr10.c:50
50                  i = a;
(gdb) step

Breakpoint 2, main () at lr10.c:51
51                  j = b;
(gdb)

Breakpoint 3, main () at lr10.c:52
52                  l = c;
(gdb)
39       for(int v = 0; v < 51; v++){
(gdb) set var i=-15
(gdb) set var j=-15
(gdb) continue
Continuing.
Bullseye!
 Time: 4
 Coordinates: -15, -15
 Param: 0
[Inferior 1 (process 6762) exited normally]
(gdb) quit
denis@ubuntu:~$
```

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. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы: Я научился отлаживать простейшие программы, написанные на языке Си

Подпись студента _____