Отчет по лабораторной работе №10 по курсу фундаментальная информатика

Студент группы М8О-112Б-22 Модин-Глазков Богдан Арсеньевич, № по списку 20

Контакты www, e-mail, icq, skype B.glazkov-modin@mail.ru

Работа выполнена: «28» октября 2022г.

Преподаватель: Никулин Сергей Петрович
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** (лабораторное):
   ЭВМ Intel Pentium G2140, процессор 3.30 GHz, имя узла сети Cameron с ОП 8096 Мб, НМД 7096 Мб. Терминал Asus адрес dev/pets/3. Принтер _____
   Другие устройства _____
   _____

   Оборудование ПЭВМ студента, если использовалось:
   Процессор Intel Core i7 2.6 GHz с ОП 16000Мб, НМД 58000 Мб. Монитор Встроенный дисплей Retina(16-дюймовый 3072 x 1920)
   Другие устройства _____
   _____

5. **Программное обеспечение (лабораторное):**
   Операционная система семейства Unix, наименование Ubuntu версия 4.15.0
   интерпретатор команд bash версия 4.4.20 Система программирования _____ версия_____
   Редактор текстов emacs версия 25.2.2
   Утилиты операционной системы:Команды nam
   Местонахождение и имена файлов программ и данных:_____
   Программное обеспечение ЭВМ студента, если использовалось:
   Операционная система семейства Unix, наименование macOS Big Sur версия 11.1
   интерпретатор команд bash версия 3.2.57(1)
   Система программирования Xcode версия _____
   Редактор текстов emacs версия 25.2.2
   Местонахождение и имена файлов программ и данных на домашнем компьютере bogdanmodin@mac

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

   - Скомпилируем исходную программу с помощью ключа компилятора -g
   - Введем данные, приводящие к ошибке. Проделаем все действия отладки над программой.

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
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
set var [имя] = [выражение]	Присваивание значения переменной.
pytype [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].

Тесты:
Input: 0
Output: Error

1. Скомпилируем при помощи команды gcc и ключа -g программу main.c (gcc -g main.c)
2. Запустим отладку с помощью команды gdb (gdb a.out)
3. Проделаем все действие по отладке программы.

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).

```
bogdanmodin@mac ~ $ gcc -g main.c
bogdanmodin@mac ~ $ gdb a.out
]GNU gdb (GDB) Fedora 9.2-7.fc33
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-redhat-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 running
Running the program.

List of commands:

advance -- Continue the program up to the given location (same form as args for
break command).
attach -- Attach to a process or file outside of GDB.
continue -- Continue program being debugged, after signal or breakpoint.
detach -- Detach a process or file previously attached.
detach checkpoint -- Detach from a checkpoint (experimental).
detach inferiors -- Detach from inferior ID (or list of IDS).
disconnect -- Disconnect from a target.
finish -- Execute until selected stack frame returns.
handle -- Specify how to handle signals.
inferior -- Use this command to switch between inferiors.
interrupt -- Interrupt the execution of the debugged program.
jump -- Continue program being debugged at specified line or address.
kill -- Kill execution of program being debugged.
kill inferiors -- Kill inferior ID (or list of IDs).
next -- Step program, proceeding through subroutine calls.
nexti -- Step one instruction, but proceed through subroutine calls.
queue-signal -- Queue a signal to be delivered to the current thread when it is
resumed.
reverse-continue -- Continue program being debugged but run it in reverse.
reverse-finish -- Execute backward until just before selected stack frame is
called.
reverse-next -- Step program backward, proceeding through subroutine calls.
reverse-nexti -- Step backward one instruction, but proceed through called
subroutines.
reverse-step -- Step program backward until it reaches the beginning of another
source line.
reverse-stepi -- Step backward exactly one instruction.
run -- Start debugged program.
signal -- Continue program with the specified signal.
start -- Start the debugged program stopping at the beginning of the main
procedure.
starti -- Start the debugged program stopping at the first instruction.
step -- Step program until it reaches a different source line.
stepi -- Step one instruction exactly.
taas -- Apply a command to all threads (ignoring errors and empty output).
target -- Connect to a target machine or process.
target core -- Use a core file as a target.
target ctf -- (Use a CTF directory as a target.
target exec -- Use an executable file as a target.
target extended-remote -- Use a remote computer via a serial line, using a gdb-
specific protocol.
target native -- Native process (started by the "run" command).
target record-btrace -- Collect control-flow trace and provide the execution
history.
target record-core -- Log program while executing and replay execution from log.
target record-full -- Log program while executing and replay execution from log.
target remote -- Use a remote computer via a serial line, using a gdb-specific
protocol.
```

```
target tfile -- Use a trace file as a target.
task -- Use this command to switch between Ada tasks.
tfaas -- Apply a command to all frames of all threads (ignoring errors and empty
output).
thread -- Use this command to switch between threads.
thread apply -- Apply a command to a list of threads.
thread apply all -- Apply a command to all threads.
thread find -- Find threads that match a regular expression.
thread name -- Set the current thread's name.
until -- Execute until past the current line or past a LOCATION.

--Type <RET> for more, q to quit, c to continue without paging--c
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
1  /* Лабораторная работа №10.
2   * Программа вычисления среднего арифмитического n целых чисел (программа с
возможной ошибкой во время исполнения).
3   * Число n вводится с клавиатуры
4   * Студент группы М8О-112Б-22 Модин-Глазков Б.А.*/
5
6  #include <stdio.h>
7  #include <math.h>
8  int main() {
9      int a, n, sum = 0;
10     int s = 0;
(gdb)
11     printf("%s\n", "Введите число n");
12     scanf("%d", &n); // Считывание количества чисел
13     printf("Введите числа \n");
14    for (int i = 0; i < n; i++) {
15        scanf("%d", &a);
16        sum += a;
17    }
18    s = sum/n;
19    printf("%f \n", s);
20    return 0;
21  }
(gdb) break 10
Breakpoint 1 at 0x401155: file main.c, line 10.
(gdb) break 18
Breakpoint 2 at 0x4011b7: file main.c, line 18.
(gdb) set args 1 1
(gdb) show args
Argument list to give program being debugged when it is started is "1 1".
(gdb) run
Starting program: /home/bogdanmodin/laba/a.out 1 1

Breakpoint 1, main () at main.c:10
10     int s = 0;
(gdb) print a
$1 = -7824
(gdb) print n
$2 = 0
(gdb) print p
$3 = 0
(gdb) set var a = 1000
(gdb) print a
$4 = 1000
(gdb) ptype a
type = int
(gdb) bt
#0  main () at main.c:10
(gdb) continue
Continuing.
Введите число n
0
Введите числа

Breakpoint 2, main () at main.c:18
18    s = p/n;
(gdb) print n
$5 = 0
(gdb) print s
```

```
$6 = 0
(gdb) set var n = 1
(gdb) continue
Continuing.
0.000000
[Inferior 1 (process 23551) exited normally]
(gdb) quit
```

9. **Дневник отладки** должен содержать дату и время сеансов отладки, и основные события (ошибки и 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 программы, написанные на Си

12. **Недочёты при выполнении задания могут быть устранены следующим образом:**

_____

_____

Подпись студента _____