**Отчет по лабораторной работе № 10 по курсу «Фундаментальная информатика»**

Студент группы М8О-109Б-22 Концебалов Олег Сергеевич

Контакты: telegram @baronpipistron

Работа выполнена: 15.11.2022

Преподаватель: каф.806 Сысоев Максим Алексеевич

Отчет сдан «» 20 г., итоговая оценка \_\_\_\_

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

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

**2. Цель работы:** Научиться пользоваться отладчиком и дебагером, находить ошибки и фиксить их

**3. Задание (вариант отсутствует):** Создать по две ошибки каждого вида (CE, RE, UB) и пофиксить их

**4. Оборудование (студента):**

Процессор AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz, ОП 16,0 Гб, SSD 512 Гб. Монитор 1920x1080 144 Hz

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

Операционная система семейства Linux, наименование Ubuntu, версия 18.10

Интерпретатор команд: bash, версия 4.4.19

Система программирования – версия --, редактор текстов Emacs, версия 25.2.2

Утилиты операционной системы –

Прикладные системы и программы –

Местонахождение и имена файлов программ и данных на домашнем компьютере –

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

Создаю необходимые ошибки в программе, умышленно нарушая синтаксис, правила написания программ и законы языка, а потом исправляю полученные ошибки

**7. Сценарий выполнения работы** *(план работы, первоначальный текст программы в черновике [можно на отдельном листе] и тесты, либо соображения по тестированию)*

1. Создаю две ошибки CE
2. Фикшу их
3. Создаю две ошибки RE
4. Фикшу их
5. Создаю две ошибки UB
6. Фикшу их

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

**Ошибка CE №1**

*Использование неинициализированной переменной z*

#include <stdio.h>  
  
int main(){  
 int j = 10;  
 int i = z + j;  
 printf("%d", i);  
 return 0;  
}

*Фикс – объявить переменную z*

#include <stdio.h>  
  
int main(){  
 int j = 10;  
 int z = 5;  
 int i = z + j;  
 printf("%d", i);  
 return 0;  
}

**Ошибка CE №2**

*Использование неверного аргумента в функции (while не может содержать . как аргумент)*

#include <stdio.h>  
  
int main(){  
 int j = 10;  
 while (.){  
 printf("%d", j);  
 }  
 return 0;  
}

*Фикс – задать нормально условие для функции while*

#include <stdio.h>  
  
int main(){  
 int j = 10;  
 while (j < 15){  
 j = j + 1;  
 printf("%d\n", j);  
 }  
 return 0;  
}

**Ошибка RE №1**

*Деление на 0 (переменную k)*

#include <stdio.h>  
  
int main(){  
 int n = 4, k = 0;  
 int x = n / k;  
 printf("%d", x);  
 return 0;  
}

*Фикс – заменить переменную k на любое другое число отличное от 0*

#include <stdio.h>  
  
int main(){  
 int n = 4, k = 4;  
 int x = n / k;  
 printf("%d", x);  
 return 0;  
}

**Ошибка RE №2**

*Выход за пределы массива*

#include <stdio.h>  
  
int main(){  
 int array[5];  
 array[10] = 15;  
 printf("%d", array[10]);  
 return 0;  
}

*Фикс – присвоить значение элементу, входящему в массив*

#include <stdio.h>  
  
int main(){  
 int array[5];  
 array[4] = 15;  
 printf("%d", array[4]);  
 return 0;  
}

**Ошибка UB №1**

*Вызов функции func, которая ничего не возвращает*

#include <stdio.h>  
  
int func(){  
}  
  
int main(){  
 return func();  
}

*Фикс – сделать так, чтобы func что-нибудь возвращала*

#include <stdio.h>  
  
int func(){  
 return 5;  
}  
  
int main(){  
 return func();  
}

**Ошибка UB №2**

*Целочисленное переполнение*

#include <stdio.h>  
  
int main(){  
 int x = 2147483647;  
 printf("%d", x + 1);  
}

*Фикс – использовать число x меньшее максимально возможного значения int*

#include <stdio.h>  
  
int main(){  
 int x = 222;  
 printf("%d", x + 1);  
}

**9. Дневник отладки** *(дата 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* | *Примечания* |
| 1 | дом | 15.11.22 | 21:04 | Не знаю, как получить RE кроме деления на 0 | Пошел гуглить | Ничего не понял, долго мучался |
| 2 | дом | 15.11.2022 | 23:15 | Нет идей как получить UB | Снова в гугл | Очень долгие поиски, пробы и ошибки |

**10. Замечания автора** *(по существу работы)*

Придумывать самому ошибки — это извращение

**11. Вывод**

Мне эта лабораторная не понравилась от слова совсем. Не понимаю и не вижу практической ценности данной лабораторки – да, можно попытаться поработать дебагером и отладчиком, но все ошибки ты придумываешь сам, значит и знаешь, что фиксить. А это несколько обесценивает мощь отладчика. Также хочется отметить всю боль, которую испытал, придумывая различные ошибки. Одно дело, когда они возникают по мере написания какой-либо программы и ты задумываешься откуда они взялись, пытаешься все починить и не поломать другое. И совершенно другой опыт, когда тебе надо самому целенаправленно придумать ошибку и потом сразу пофиксить ее (what?).

Но есть и положительные стороны в этой лабе: пока мучался с RE и UB пришлось облазить кучу сайтов в гугле, в основном иностранных, что помогло узнать немного новых слов и в целом дало небольшую практику в английском языке. За это лайк

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