Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра вычислительных систем

**ОТЧЕТ**

по практической работе 1

по дисциплине «**Программирование**»

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. ИС-241  «\_\_» февраля 2022 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Дмитрюк В.В./ |
|  |  |  |
| Проверил:  ст. преп. Кафедры ВС  «\_\_» февраля 2018 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Фульман В.О./ |

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_»

Новосибирск 2023

**ОГЛАВЛЕНИЕ**

[ЗАДАНИЕ 3](#_Toc35593781)

[ВЫПОЛНЕНИЕ РАБОТЫ 4](#_Toc35593782)

[ПРИЛОЖЕНИЕ 5](#_Toc35593783)

# **ЗАДАНИЕ**

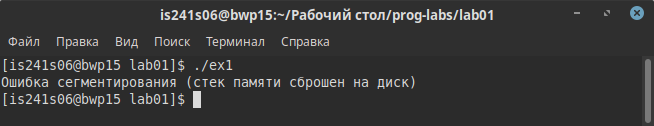
*Получить навыки отладки программ на примере использования отладчика GDB. Локализовать и исправить ошибки в приведённых программах с помощью отладчика.*

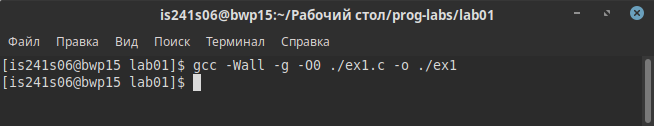
**ВЫПОЛНЕНИЕ РАБОТЫ**

Рассмотрим задание 1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <stdio.h>  #include <stdlib.h>  **void** **init**(**int**\* arr, **int** n)  {  arr = malloc(n \* **sizeof**(**int**));  **int** i;  **for** (i = **0**; i < n; ++i)  {  arr[i] = i;  }  }  **int** **main**(){  **int**\* arr = NULL;  **int** n = **10**;  init(arr, n);  **int** i;  **for** (i = **0**; i < n; ++i)  {  printf("%d**\n**", arr[i]);  }  **return** **0**;  } |

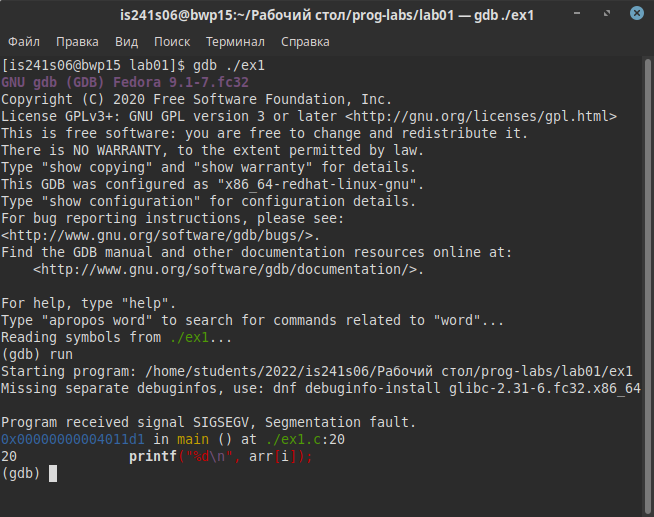
Очевидно, что функция init призвана инициализировать массив определённого размера инкрементально возрастающими числами. Однако, в ней допущена ошибка, из-за чего в результате работы программы происходит ошибка сегментирования.

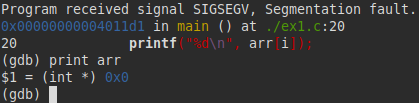
Компилятор не сообщил о предупреждениях даже с флагом «-Wall».   
Воспользуемся отладчиком для локализации ошибки. Для этого мы должны скомпилировать программу с флагами «-g -O0».



Вызовем отладчик gdb и запустим в нём нашу программу.

Вызовем отладчик gdb и запустим в нём нашу программу.

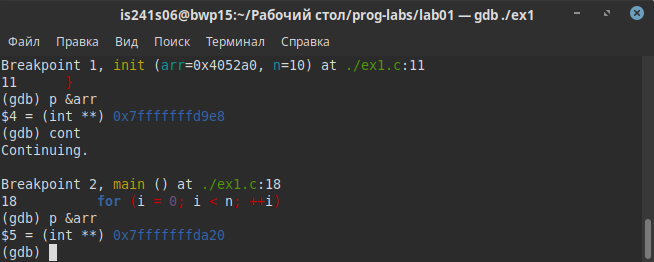
Ошибка произошла на строке 20, при попытке вывести значение массива в стандартный поток ввода-вывода. Ошибка сегментации чаще всего происходит при попытке обратиться к памяти, в которую у нас нет доступа. Выведем на экран значение указателя arr.



Действительно, ошибка произошла при попытке обратиться к памяти по нулевому указателю. Значение указателя в функции arr не изменилось, вопреки замыслу автора, поскольку переменные в функцию передаются по значению. Докажем это, сравнив значения указателей arr внутри функций init и main.

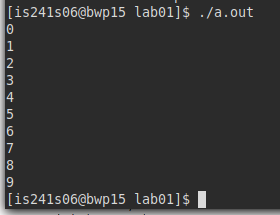
Поставим точку останова на строчках 11 и 17 — завершение работы функции init и строчка, следующая за вызовом функции init внутри функции main соответственно, и посмотрим в них значения переменной arr.

Адреса этих указателей тоже различаются:



Изменим программу таким образом, чтобы избежать проблемы передачи аргументов по значению: будем изменять указатель, определённый в функции main из функции init по указателю на него.  
Добавления выделены жёлтым цветом.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | #include <stdio.h>  #include <stdlib.h>  **void** **init**(**int**\*\* arr, **int** n)  {  \*arr = malloc(n \* **sizeof**(**int**));  **int** i;  **for** (i = **0**; i < n; i++)  {  (\*arr)[i] = i;  }  }  **int** **main**()  {  **int**\* arr = NULL;  **int** n = **10**;  init(&arr, n);  **int** i;  **for** (i = **0**; i < n; ++i)  {  printf("%d**\n**", arr[i]);  }  **return** **0**;  } |

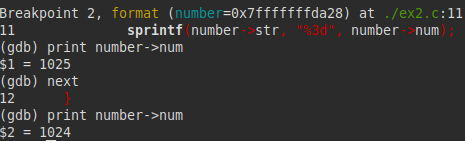
Теперь программа работает так, как и было задумано.

Рассмотрим задание 2.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <stdio.h>  **typedef** **struct**  {  **char** str[**3**];  **int** num;  } NumberRepr;  **void** **format**(NumberRepr\* number)  {  sprintf(number->str, "%3d", number->num);  }  **int** **main**()  {  NumberRepr number = { .num = **1025** };  format(&number);    printf("str: %s**\n**", number.str);  printf("num: %d**\n**", number.num);    **return** **0**;  } |

Задумка программиста — представить число 1025 в строковом и целочисленном представлении. Однако, программа выполнена неудачно — строковое и целочисленное представление не совпадают.

С помощью отладчика установлено, что переменная number->num приобретает другое значение после выполнения функции на строке 11.

Команда написана правильно — остаётся полагать, что str каким-то образом перезаписывает num.

Проверим это.

sprintf(number->str, "%3d", number->num) запишет в str 5 байт: четыре знака числа и символ конца строки.

# **ПРИЛОЖЕНИЕ**

Исходный код с комментариями;

**example.c**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #include <assert.h>  #include <stddef.h>  #include <stdint.h>  **size\_t** **encode\_varint**(**uint32\_t** value, **uint8\_t**\* buf)  {  assert(buf != NULL);  **uint8\_t**\* cur = buf;  **while** (value >= **0x80**) {  **const** **uint8\_t** byte = (value & **0x7f**) | **0x80**;  \*cur = byte;  value >>= **7**;  ++cur;  }  \*cur = value;  ++cur;  **return** cur - buf;  }  **uint32\_t** **decode\_varint**(**const** **uint8\_t**\*\* bufp)  {  **const** **uint8\_t**\* cur = \*bufp;  **uint8\_t** byte = \*cur++;  **uint32\_t** value = byte & **0x7f**;  **size\_t** shift = **7**;  **while** (byte >= **0x80**) {  byte = \*cur++;  value += (byte & **0x7f**) << shift;  shift += **7**;  }  \*bufp = cur;  **return** value;  } |