Lab 2 Exercise 2 – Segmentation Fault in Linux

Name: NOEL LIM XIAN
Student Number: A0211270Y
Email: E0493357@u.nus.edu
Date: 31 AUGUST 2022

Question for each part

a. What is the behavior? b. Is there a memory error? If there is a memory error, explain the rationale for the memory error. If there is no memory error, explain why it isn't a memory error. You should also explain the reason for the result and what happened. Your explanation should be clear, refer to the code and justify the behavior observed.

1

a)

Segmentation fault (core dumped)

b)

Case 1 has an instruction `p = 0;` which reassigns p to 0. We can add `printf("%p",NULL);` at the start of main to observe that NULL value is also 0. That is, p is assigned to the NULL pointer. Also, address 0x00 is owned by the operating system, not the program. Invoking `setvalue()`, program will attempt to dereference the NULL pointer `p` which is not owned by the program. Also, since 0x00 does not point to any variable, assigning an r-value to `*p` is illogical. Hence the segmentation fault.

2

a)

Segmentation fault (core dumped)

b)

`p` is assigned to `hello` which is a pointer to a string literal. String literals are static and not modifiable. That is, the character elements in `hello` cannot be replaced. However, reading/access is allowed.

```
*p; // legal
printf("%c",*p); // legal
*p = 'W'; // illegal, undefined behavior
```

See also: https://en.cppreference.com/w/c/language/string_literal

3

a)

program returns 0

b)

`string` is an array of element type `char`. Unlike c-string, it is not null-terminated and elements are modifiable.

4

a)

Segmentation fault (core dumped)

b)

`p` is assigned to NULL, and dereferencing NULL is an undefined behavior (RHS value is undefined in the assignment to int `choice`);

5

a)

program returns 0

b)

First, observe `printf("%i", sizeof(char *) == sizeof(int *));` is 1 (true), that is size of a char pointer and int pointer is the same. Size-wise, pointer type-cast from `int *` to `char *` is legal. Second, `sizeof(int) == 4` and `sizeof(char) == 1`. As long as we assign to `*p` an r-value that is within 1 byte (8-bits), the program will not complain. Otherwise, value will overflow.

8

a)

Segmentation fault (core dumped)

b)

0x40000000 is not mapped in the program (program is not aware of the space region).