Sidney Thomas
Github: siid14
ID: 918656419
CSC415 Operating Systems

Assignment 2– Struct and Buffer

Description:
This C program processes personal information from command-line arguments, allocates memory for data storage, writes the information, and processes variable-length strings. It manages data buffering and commits it in chunks while ensuring proper memory allocation and deallocation. The program returns an exit status based on the result of the checkIt function.

Approach / What I Did:
For this assignment, I was tasked with creating a C program that demonstrates how to access and print command-line arguments. Here's a breakdown of the steps I took and my thought process while working on the assignment:

1. Memory Allocation:
I allocated memory to store personal information using malloc and checked for allocation success.

2.Populating personalInfo:
I populated the personalInfo structure with data from command-line arguments, including first name, last name, student ID, grade level, programming languages, and a message.

3.Data Writing:
I called the writePersonalInfo function to perform a write operation with the personalInfo data.

4. Buffer Allocation:
I allocated a buffer for data processing using malloc, handling memory allocation errors.

5. Data Retrieval and Processing:
I used a loop to retrieve variable-length strings with the getNext function and processed them while managing buffer space efficiently.

6. Buffer Management and Commit:
I managed the buffer to store and commit data in smaller chunks, ensuring it didn't overflow. If the buffer reached its limit, you committed it using the commitBlock function.

7. Remaining Data Handling:
I ensured any remaining data in the buffer was properly committed before the program exited.

8. Memory Cleanup:

I freed allocated memory for both the personalInfo structure and the buffer to prevent memory leaks.

9. Exit Status:

Finally, I returned an exit status code based on the result of the checkIt function, indicating the success or failure of the program

10. Final Review and Submission:

Finally, I reviewed the code, comments, and the assignment requirements to ensure I hadn't missed anything. Once satisfied, I consider<ed the assignment complete and ready for submission.

Throughout the process, I focused on breaking down the task into manageable steps and ensuring that each step aligned with the assignment's requirements. This approach helped me create a functional and well-documented program.

Issues and Resolutions:

I had difficulties with printing data, I have faced issues related to incorrect format specifiers or function usage. The resolution did involve reviewing the printf statements, ensuring proper format specifiers (%s for strings, %d for integers, etc.), and checking that I was passing the correct variables/pointers to printf for output (debugging purpose)

I also mention that I initially had trouble committing the remaining data at the end. The resolution here was to implement a check within the loop that keeps track of the buffer position. When the buffer was full (reached 256 bytes), I called commitBlock(buffer) to ensure data was committed correctly.

At one point, I encountered a situation where the last line of data was not being committed to the buffer correctly. Initially, I overlooked this aspect of the program's logic, assuming that data would only be committed when the buffer was fully filled.

To address this issue, a helpful classmate reminded me to ensure that the last block of data is committed even if it's not fully filled. I implemented a check to commit the buffer if there was any remaining data in it, regardless of whether it reached the full buffer size. This fix ensured that no data was left unprocessed.

Screen shot of compilation:

```
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-2/csc415-assignment2-bufferandstruct-siid14$ make clean
rm Thomas_Sidney_HW2_main.o Thomas_Sidney_HW2_main
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-2/csc415-assignment2-bufferandstruct-siid14$ make
gcc -c -o Thomas_Sidney_HW2_main.o Thomas_Sidney_HW2_main.c -g -I.
gcc -o Thomas_Sidney_HW2_main Thomas_Sidney_HW2_main.o assignment2.o -g -I.
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-2/csc415-assignment2-bufferandstruct-siid14$
```

Screen shot(s) of the execution of the program:

```
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-2/csc415-assignment2-bufferandstruct-siid14$ make run
./Thomas_Sidney_HW2_main Sidney Thomas "Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Libert
y, and dedicated to the proposition that all men are created equal."
---------------------------------- CHECK ----------------------------------
Running the check for Sidney Thomas
Name check is 0 by 0
Student ID: 918656419, Grade Level: Senior
Languages: 264215 (40817)
Message:
Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived

The Check Succeded (0, 0)

END-OF-ASSIGNMENT
000000: 5B 82 DD C8 FD 7F 00 00  62 82 DD C8 FD 7F 00 00 | [????..b????..
000010: A3 95 C1 36 03 00 00 00  17 08 04 00 46 6F 75 72 | ???6........Four
000020: 20 73 63 6F 72 65 20 61  6E 64 20 73 65 76 65 6E |  score and seven
000030: 20 79 65 61 72 73 20 61  67 6F 20 6F 75 72 20 66 |  years ago our f
000040: 61 74 68 65 72 73 20 62  72 6F 75 67 68 74 20 66 | athers brought f
000050: 6F 72 74 68 20 6F 6E 20  74 68 69 73 20 63 6F 6E | orth on this con
000060: 74 69 6E 65 6E 74 2C 20  61 20 6E 65 77 20 6E 61 | tinent, a new na
000070: 74 69 6F 6E 2C 20 63 6F  6E 63 65 69 76 65 64 20 | tion, conceived

student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-2/csc415-assignment2-bufferandstruct-siid14$
```

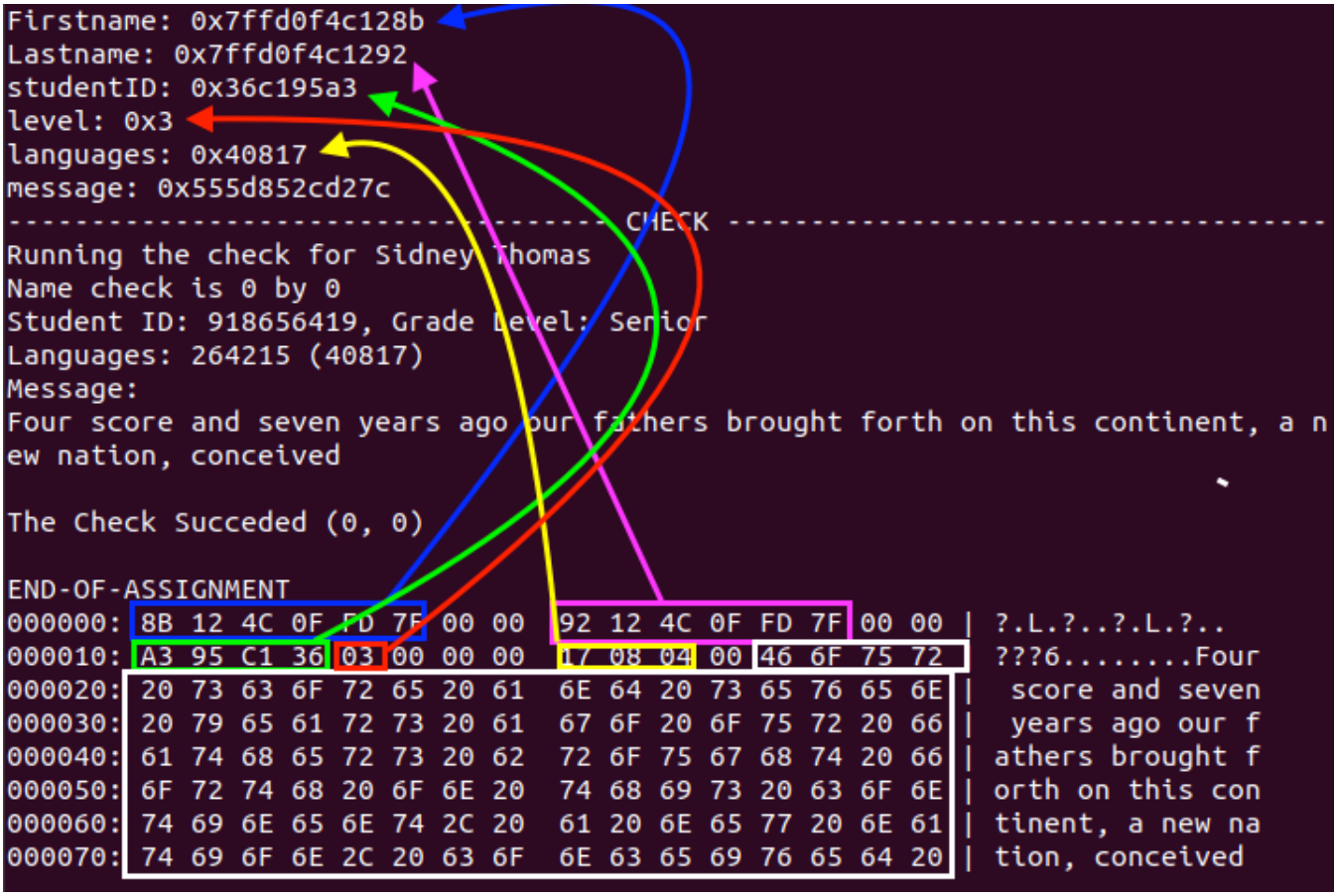
```
---------------------------------- CHECK ----------------------------------
Running the check for Sidney Thomas
Name check is 0 by 0
Student ID: 918656419, Grade Level: Senior
Languages: 264215 (40817)
Message:
Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived

The Check Succeded (0, 0)

END-OF-ASSIGNMENT
000000: 5B 82 DD C8 FD 7F 00 00  62 82 DD C8 FD 7F 00 00 | [????..b????..
000010: A3 95 C1 36 03 00 00 00  17 08 04 00 46 6F 75 72 | ???6........Four
000020: 20 73 63 6F 72 65 20 61  6E 64 20 73 65 76 65 6E |  score and seven
000030: 20 79 65 61 72 73 20 61  67 6F 20 6F 75 72 20 66 |  years ago our f
000040: 61 74 68 65 72 73 20 62  72 6F 75 67 68 74 20 66 | athers brought f
000050: 6F 72 74 68 20 6F 6E 20  74 68 69 73 20 63 6F 6E | orth on this con
000060: 74 69 6E 65 6E 74 2C 20  61 20 6E 65 77 20 6E 61 | tinent, a new na
000070: 74 69 6F 6E 2C 20 63 6F  6E 63 65 69 76 65 64 20 | tion, conceived

student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-2/csc415-assignment2-bufferandstruct-siid14$
```

Analysis (Step 8) :



On this Analysis, I did print the hexadecimal value of each data and find through my output produced via the function checkIt().

As you can notice we found the correspond hexadecimal value of the data in little endian.

The bytes at the address 000000 000001 – (blue) represents the little endian to firstName field with the hexadecimal of 0x07ffd0f4c128b

The bytes at the address 000000 000001 – (pink) represents the little endian to lastName field with the hexadecimal of 0x7ffd0f4c1292

The bytes at the address 000010 - 000011 – (green) represents the little endian to studentID field with the hexadecimal of 0x36C195a3

The bytes at the address 000010 - 000011 – (red) represents the little endian to level field with the hexadecimal of 0x3

The bytes at the address 000010 - 000011 represents the little endian to languages field with the hexadecimal of 0x40817

The bytes at the address 000010 - 000011 – (white) represents the little endian to message field with the hexadecimal of 0x555d852cd27c
And the followings are just at addresses that continuously represent the little endian to message field with the hexadecimal of 0x555d852cd27c.
The last 100 bytes represent the (char) message array.

It’s fair to indicate that the pointers are only valid during the execution of the program.