

Assignment 2 – Buffer and Struct

Description:

This assignment creates and allocates an instance of the `personalInfo` structure which is then populated using command line arguments and predetermined values and is written by calling `writePersonalInfo` and then deallocated. After this process is finished a character pointer named `buffer` is created to copy values returned by `getNext` and commit them when the buffer is full. Once this process is repeated until no more c-string values are returned by `getNext` and the final contents of the buffer are committed, the buffer is then deallocated.

Approach:

To complete each step expected in this assignment I plan to start by creating and allocating the `personalInfo` instance which will then be populated with values. To ensure this is done correctly I will use `malloc` to return a pointer to the `personalInfo` instance using the struct size. I will then populate the values using the first 3 command line arguments for the first and last names, and the message. The `studentId` will be initialized like a regular integer variable with my respective student id number, other values will be initialized using the predefined `KNOWLEDGE_OF` values in `assignment2.h` for the languages, the enum value of `SENIOR` for level as this is my respective grade level. The only `personalInfo` value I intend on populating alternatively is the message by using `strncpy`, the reason for this is that the message is an array of the set size of 100 characters as opposed to being a character pointer. Once the message is successfully populated, I will write the instance using `writePersonalInfo` and deallocate it using `free` since it won't be used any further.

Upon the completion of the deallocation of the personal info instance, I will create a character pointer and allocate it using `malloc` and the defined `BLOCK_SIZE`. Due to the buffer needing to have values from `getNext` copied into it, I will need to keep track of the space being used, and the space needed for the incoming c-string, and use these two variables to check if the buffer is near capacity before committing the buffer. I intend to use a while loop for the copying of `getNext` values since it will be able to monitor whether `getNext` has returned a null while iterating through `getNext` and the buffer. `Memcpy` will be used to copy values returned from `getNext` by using a temporary character pointer as the source and the buffer as the destination while using the values of `spaceUsed` as an offset for the buffer and `spaceNeeded` to define the size of what's being copied where I will then call `getNext` to receive the next c-string and continue the loop. To make sure the buffer isn't being filled once it's full I plan to use an if statement that checks if it's about to reach capacity, copy as much of the incoming string as it can to the buffer, and will then commit the buffer. Following this commit the space used will then be reset and the string value that was cut off will then be adjusted so it will start after the cut-off. Once the loop has ended I will check the buffer and if it still contains any leftover data I will commit it and deallocate the buffer before calling and returning `checkIt` and its return value upon termination.

Issues and Resolutions:

As I attempted to copy c-strings returned to the temp char pointer I ran into an issue using memcpy without having an offset value to specify the position that values from temp were going to be copied into inside the buffer. The manner in which I had originally done this is shown below.

```
memcpy(buffer, temp, spaceNeeded);
```

I was using syntax that is expected of strncpy which I typically used to copy strings from the beginning, however, due to memcpy being used to copy the memory over from a source to a destination you must specify the position in which the copy will be performed. This was a simple problem to fix by applying an offset which is where my counter spaceUsed that tracks the amount of space being occupied in the buffer came in. By adding the offset to the buffer through pointer arithmetic it then solved my problem as shown below.

```
memcpy(buffer + spaceUsed, temp, spaceNeeded);
```

Analysis:

When analyzing the hexdump produced by the program it's important to separate bytes based on their groupings and then translate them. Before the hexdump is output the checkIt function shows the results of the check and displays the values that were populated in our personalInfo in the order that will be shown in the hexdump. The order of these values is in the same order as declared in the personalInfo structure.

END-OF-ASSIGNMENT			
000000:	51 A3 04 C3 FD 7F 00 00	58 A3 04 C3 FD 7F 00 00	Q?..??..X?..??..
000010:	81 E1 08 37 14 00 00 00	1F 0C 44 01 46 6F 75 72	??..7.....D.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

In the image above each highlighted field represents values of the personalInfo struct in the order mentioned.

Both the first and last name fields are highlighted in yellow with Bytes at address 000000 - 000007 representing the address of the firstname character pointer, and Bytes at address 000008 - 00000F representing the address of the lastname character pointer. When looking at the ASCII representation of this on the right each time our program is executed both the hexadecimal and ASCII representations will change as these won't translate into text but are just representative of the memory location the pointers point to.

The Student ID field is highlighted in green with Bytes at address 000010 - 000013 being the little-endian hexadecimal translation of 923328897 which is my student id.

Enter decimal number		Little endian				
<input type="text" value="923328897"/>		Address	0	1	2	3
<div>= Convert × Reset ↕ Swap</div>		Data	81	E1	08	37
Hex number (8 digits)		Big endian				
<input type="text" value="3708E181"/>		Address	0	1	2	3
		Data	37	08	E1	81

The Grade Level field is highlighted in purple with Bytes at address 000014 - 000017 being the little-endian hexadecimal translation of 20 which is equivalent to the enum of SENIOR. In the personInfo struct FRESHMAN is set to the value of 17 and as one goes up in grade the value is incremented accordingly which is why SENIOR is equal to 20.

Enter decimal number		Little endian	
<input type="text" value="20"/>		Address	0
<div>= Convert × Reset ↕ Swap</div>		Data	14
Hex number (2 digits)		Big endian	
<input type="text" value="14"/>		Address	0
		Data	14

The Languages field is highlighted in pink with Bytes at address 000018 - 00001B being the little-endian hexadecimal translation of 21236767 which is the total value of all the languages I know thus far. The values of each respective coding language are declared in assignment2.h

```
KNOWLEDGE_OF_C | KNOWLEDGE_OF_JAVA | KNOWLEDGE_OF_JAVASCRIPT |  
KNOWLEDGE_OF_PYTHON | KNOWLEDGE_OF_CPLUSPLUS | KNOWLEDGE_OF_SQL |  
KNOWLEDGE_OF_HTML | KNOWLEDGE_OF_MIPS_ASSEMBLER | KNOWLEDGE_OF_R |  
KNOWLEDGE_OF_BASIC;
```

If we take the values for 10 I've listed from the definitions below we can calculate the total to be 21236767.

#define KNOWLEDGE_OF_C	1	#define KNOWLEDGE_OF_PROLOG	8192
#define KNOWLEDGE_OF_JAVA	2	#define KNOWLEDGE_OF_C_SHARP	16384
#define KNOWLEDGE_OF_JAVASCRIPT	4	#define KNOWLEDGE_OF_PL1	32768
#define KNOWLEDGE_OF_PYTHON	8	#define KNOWLEDGE_OF_INTEL_ASSEMBLER	65536
#define KNOWLEDGE_OF_PASCAL	16	#define KNOWLEDGE_OF_IBM_ASSEMBLER	131072
#define KNOWLEDGE_OF_FORTTRAN	32	#define KNOWLEDGE_OF_MIPS_ASSEMBLER	262144
#define KNOWLEDGE_OF_RUBY	64	#define KNOWLEDGE_OF_ARM_ASSEMBLER	524288
#define KNOWLEDGE_OF_ADA	128	#define KNOWLEDGE_OF_COBOL	1048576
#define KNOWLEDGE_OF_LISP	256	#define KNOWLEDGE_OF_APL	2097152
#define KNOWLEDGE_OF_SQL	512	#define KNOWLEDGE_OF_R	4194304
#define KNOWLEDGE_OF_HTML	1024	#define KNOWLEDGE_OF_OBJC	8388608
#define KNOWLEDGE_OF_SWIFT	4096	#define KNOWLEDGE_OF_BASIC	16777216
		#define KNOWLEDGE_OF_PHP	33554432
		#define KNOWLEDGE_OF_GO	67108864

Enter decimal number

21236767

= Convert

✕ Reset

↕ Swap

Hex number (7 digits)

1440C1F

Little endian

Address	0	1	2	3
Data	1F	0C	44	01

Big endian

Address	0	1	2	3
Data	01	44	0C	1F

The Message field is highlighted in blue with Bytes at address 00001C - 00007F being the hexadecimal translation of the message "Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived". Unlike the other fields, this can be translated through ASCII confirming its accuracy.

```
46 6F 75 72
20 73 63 6F 72 65 20 61 6E 64 20 73 65 76 65 6E
20 79 65 61 72 73 20 61 67 6F 20 6F 75 72 20 66
61 74 68 65 72 73 20 62 72 6F 75 67 68 74 20 66
6F 72 74 68 20 6F 6E 20 74 68 69 73 20 63 6F 6E
74 69 6E 65 6E 74 2C 20 61 20 6E 65 77 20 6E 61
74 69 6F 6E 2C 20 63 6F 6E 63 65 69 76 65 64 20
```

Four score and seven years ago our fathers brought forth on
this continent, a new nation, conceived

Screen shot of compilation:

```
student@student: ~/csc415-assignment2-bufferandstruct-ulicessgg-4
student@student:~/csc415-assignment2-bufferandstruct-ulicessgg-4$ make clean
rm Gonzalez_Ulices_HW2_main.o Gonzalez_Ulices_HW2_main
student@student:~/csc415-assignment2-bufferandstruct-ulicessgg-4$ make
gcc -c -o Gonzalez_Ulices_HW2_main.o Gonzalez_Ulices_HW2_main.c -g -rdynamic -I.
gcc -o Gonzalez_Ulices_HW2_main Gonzalez_Ulices_HW2_main.o assignment2.o -g -rdynamic -I.
student@student:~/csc415-assignment2-bufferandstruct-ulicessgg-4$ make run
```

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

```
gcc -o Gonzalez_Ulices_HW2_main Gonzalez_Ulices_HW2_main.o assignment2.o -g -rdynamic -I.
student@student:~/csc415-assignment2-bufferandstruct-ulicessgg-4$ make run
./Gonzalez_Ulices_HW2_main Ulices Gonzalez "Four score and seven years ago our fathers brought
t forth on this continent, a new nation, conceived in Liberty, and dedicated to the propositi
on that all men are created equal."
----- CHECK -----
Running the check for Ulices Gonzalez
Name check is 0 by 0
Student ID: 923328897, Grade Level: Senior
Languages: 21236767
Message:
Four score and seven years ago our fathers brought forth on this continent, a new nation, con
ceived

The Check Succeeded (0, 0)

END-OF-ASSIGNMENT
000000: 51 A3 04 C3 FD 7F 00 00 58 A3 04 C3 FD 7F 00 00 | Q?..??..X?..??..
000010: 81 E1 08 37 14 00 00 00 1F 0C 44 01 46 6F 75 72 | ??..7.....D.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:~/csc415-assignment2-bufferandstruct-ulicessgg-4$
```