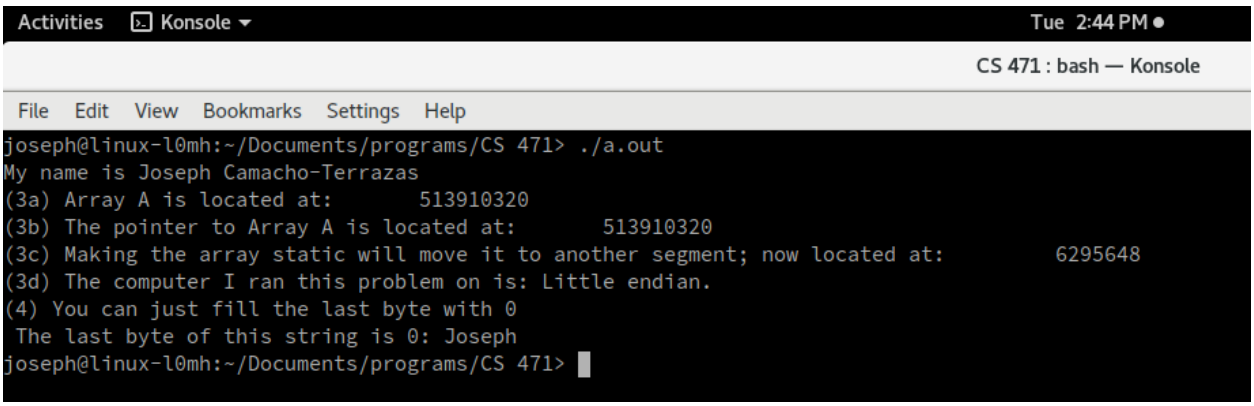
Programming #1 Simple C Aliasing Problem
Joseph Camacho-Terrazas
CS 471

```c
/*
* Joseph Camacho-Terrazas
* 8/28/2020
* Input: An array of 4-byte integers
* Output: Name printed in text
* Preconditions: None
* Postconditions: The ASCII integers will print my name as text.
*/

//Include
#include <stdio.h>
//Function prototype
void endianTest(int num);

int main() {

    //Declare array A to store integer values
    int A[100];
    //Declare pointer char S for casting later
    char *S;

    //ASCII values for J o s e
    A[0] = 74 + (111 * 256) + (115 * 256 * 256) + (101 * 256 * 256 * 256);
    //ASCII values for p h   C
    A[1] = 112 + (104 * 256) + (32 * 256 * 256) + (67 * 256 * 256 * 256);
    //ASCII values for a m a c
    A[2] = 97 + (109 * 256) + (97 * 256 * 256) + (99 * 256 * 256 * 256);
    //ASCII values for h o - T
    A[3] = 104 + (111 * 256) + (45 * 256 * 256) + (84 * 256 * 256 * 256);
    //ASCII values for e r r a
    A[4] = 101 + (114 * 256) + (114 * 256 * 256) + (97 * 256 * 256 * 256);
    //ASCII values for z a s
    A[5] = 122 + (97 * 256) + (115 * 256 * 256);
    //Terminator
    A[6] = 0;

    //Cast the integer array into char*
    S = (char*) A;

    //Final name print
    printf("My name is %s\n", S);

    //Question tests
```

Programming #1 Simple C Aliasing Problem
Joseph Camacho-Terrazas
CS 471

```
    //Test array for question 3
    //Declaring a static array will move it to another memory segment
    static int B[100];

    //ASCII byte test for question 4
    //Declare array C to store integer values
    int C[100];
    //Declare pointer char S2 for casting later
    char *S2;

    //ASCII values for J o s e
    C[0] = 74 + (111 * 256) + (115 * 256 * 256) + (101 * 256 * 256 * 256);
    //ASCII values for p h and make the last byte a terminator
    C[1] = 112 + (104 * 256) + (0 * 256 * 256);

    //Cast the integer array into char*
    S2 = (char*) C;

    printf("(3a) Array A is located at: %15u \n", A);
    printf("(3b) The pointer to Array A is located at: %15u \n", S);
    printf("(3c) Making the array static will move it to another segment; now loc
ated at: %15u \n", B);
    printf("(3d) The computer I ran this problem on is: ");
    endianTest(1);
    printf("(4) You can just fill the last byte with 0\n The last byte of this st
ring is 0: %s\n", S2);

    return (0);
}

//Simple endianess test. Casting the integer 1 to char* will only contain the fir
st byte.
//So if my computer is little endian c* will contain only 1, otherwise it's big e
ndian and c* contains a 0.
void endianTest(int num) {
    if ((char*)&num == 1) {printf("Big endian.\n");}
    else {printf("Little endian.\n");}
}
```

Programming #1 Simple C Aliasing Problem
Joseph Camacho-Terrazas
CS 471



Programming #1 Simple C Aliasing Problem
Joseph Camacho-Terrazas
CS 471

3a) The array is located in the stack.

```
(3a) Array A is located at:       513910320
```

3b) The pointer to the array is located in the stack as well, since it points to the address of the first element of the array.

```
(3b) The pointer to Array A is located at:       513910320
```

3c) To make the array be in the other segment, I declared a new static int array B. By declaring it as static or global, you can move it to another segment.

```
(3c) Making the array static will move it to another segment; now located at:         6295648
```

3d) The computer I ran this problem on is big endian.

```
(3d) The computer I ran this problem on is: Little endian.
```

3e) There is a difference between little and big endian because there are different processor architectures. Intel based processors use little endian and Motorola uses big endian. There is not one that is better per se, but it just depends on how a programmer chooses to handle a case of big endianness. Source: https://www.geeksforgeeks.org/little-and-big-endian-mystery/

4) You can just set that last byte as ‘0’ and get the correct result. To test this, I set the byte after the last ASCII integer of my name to ‘0’. My name was still printed correctly.

```
(4) You can just fill the last byte with 0
 The last byte of this string is 0: Joseph
```