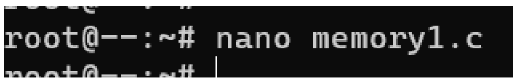
Ramzy Izza Wardhana
21/472698/PA/20322
IUP CS B

Assignment 10 – Memory Management

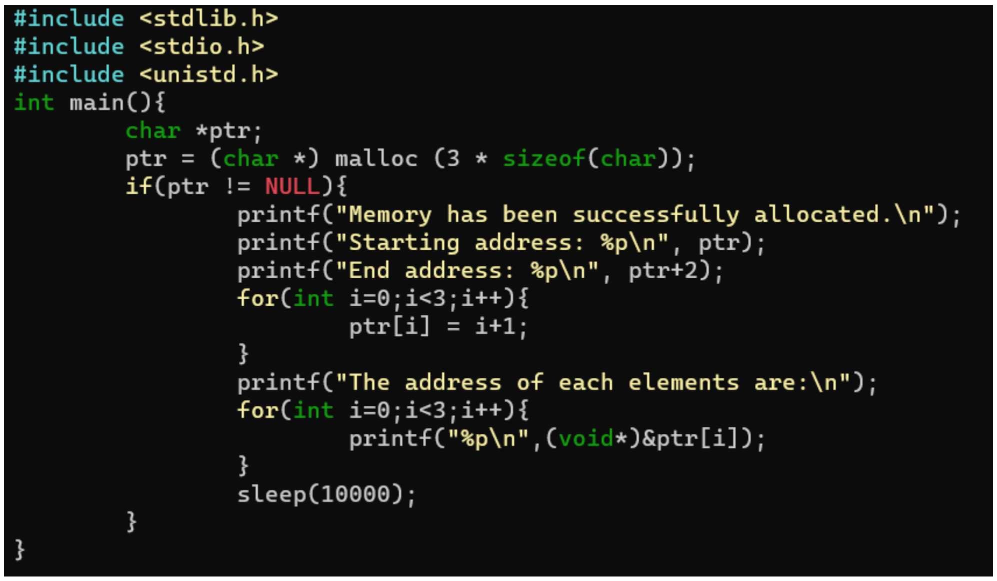
1. Write a program that declares a variable of any type. Print the memory address of the variable and find out in which segment the variable is stored in the virtual address space. Explain why the variable is stored there!

   Step 1: Create the memory1.c using nano



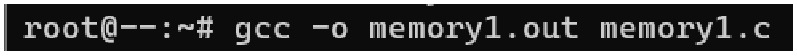
```
root@--:~# nano memory1.c
```

   Step 2: Write the code



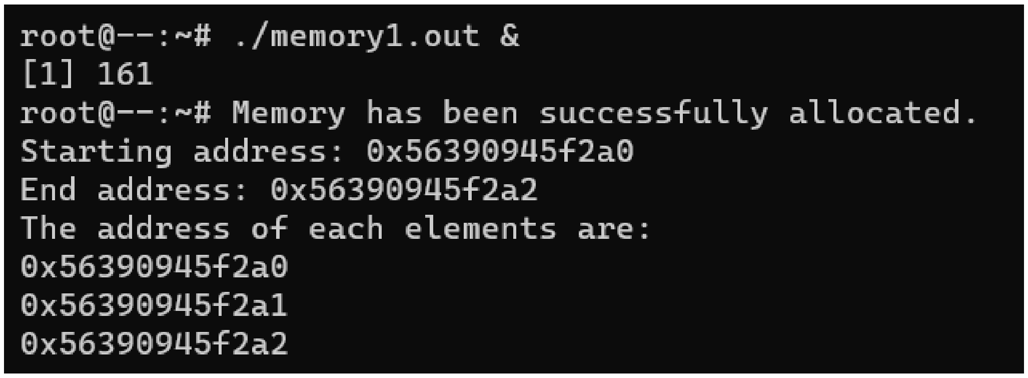
```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
int main(){
        char *ptr;
        ptr = (char *) malloc (3 * sizeof(char));
        if(ptr != NULL){
                printf("Memory has been successfully allocated.\n");
                printf("Starting address: %p\n", ptr);
                printf("End address: %p\n", ptr+2);
                for(int i=0;i<3;i++){
                        ptr[i] = i+1;
                }
                printf("The address of each elements are:\n");
                for(int i=0;i<3;i++){
                        printf("%p\n",(void*)&ptr[i]);
                }
                sleep(10000);
        }
}
```

   Step 3: Compile the code



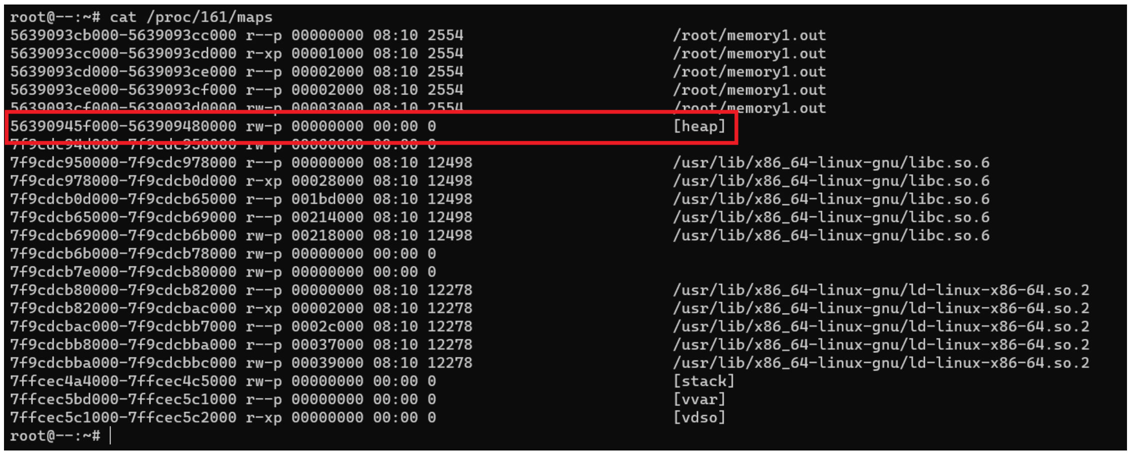
```
root@--:~# gcc -o memory1.out memory1.c
```

   Step 4: Run the code



```
root@--:~# ./memory1.out &
[1] 161
root@--:~# Memory has been successfully allocated.
Starting address: 0x56390945f2a0
End address: 0x56390945f2a2
The address of each elements are:
0x56390945f2a0
0x56390945f2a1
0x56390945f2a2
```

Step 5: Analyze the maps



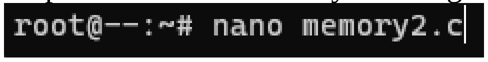
```
root@--:~# cat /proc/161/maps
5639093cb000-5639093cc000 r--p 00000000 08:10 2554                       /root/memory1.out
5639093cc000-5639093cd000 r-xp 00001000 08:10 2554                       /root/memory1.out
5639093cd000-5639093ce000 r--p 00002000 08:10 2554                       /root/memory1.out
5639093ce000-5639093cf000 r--p 00002000 08:10 2554                       /root/memory1.out
5639093cf000-5639093d0000 rw-p 00003000 08:10 2554                       /root/memory1.out
56390945f000-563909480000 rw-p 00000000 00:00 0                          [heap]
7f9cdc94d000-7f9cdc950000 rw-p 00000000 00:00 0
7f9cdc950000-7f9cdc978000 r--p 00000000 08:10 12498                      /usr/lib/x86_64-linux-gnu/libc.so.6
7f9cdc978000-7f9cdcb0d000 r-xp 00028000 08:10 12498                      /usr/lib/x86_64-linux-gnu/libc.so.6
7f9cdcb0d000-7f9cdcb65000 r--p 001bd000 08:10 12498                      /usr/lib/x86_64-linux-gnu/libc.so.6
7f9cdcb65000-7f9cdcb69000 r--p 00214000 08:10 12498                      /usr/lib/x86_64-linux-gnu/libc.so.6
7f9cdcb69000-7f9cdcb6b000 rw-p 00218000 08:10 12498                      /usr/lib/x86_64-linux-gnu/libc.so.6
7f9cdcb6b000-7f9cdcb78000 rw-p 00000000 00:00 0
7f9cdcb7e000-7f9cdcb80000 rw-p 00000000 00:00 0
7f9cdcb80000-7f9cdcb82000 r--p 00000000 08:10 12278                      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7f9cdcb82000-7f9cdcbac000 r-xp 00002000 08:10 12278                      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7f9cdcbac000-7f9cdcbb7000 r--p 0002c000 08:10 12278                      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7f9cdcbb8000-7f9cdcbba000 r--p 00037000 08:10 12278                      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7f9cdcbba000-7f9cdcbbc000 rw-p 00039000 08:10 12278                      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ffcec4a4000-7ffcec4c5000 rw-p 00000000 00:00 0                          [stack]
7ffcec5bd000-7ffcec5c1000 r--p 00000000 00:00 0                          [vvar]
7ffcec5c1000-7ffcec5c2000 r-xp 00000000 00:00 0                          [vdso]
root@--:~# |
```

Answer:
On the code implementation above, I declared the data type as char with size of 3 times of char size. Then by using the malloc(), I allocated the memory to store our value from index 0 until 2. Not only that, but I also print out the start address *(0x56390945f2a0)*, end address *(0x56390945f2a2)*, and each address of the value to easily evaluate the location of the virtual addresses. By compiling and running the program in the background using & (ampersand) and retrieve the PID [161], we then find the mapping file by typing cat /proc/161/maps to output the list of virtual addresses.

Picture depicted above, illustrated to us that the variable is stored in the [heap] *(observe the address starts with 56390945f000)*. This was caused by heap segment are specifically used to store the data that was dynamically allocated by the program, which I already done at the first segment of the code [*ptr = (char *) malloc (3 * sizeof(char));*].

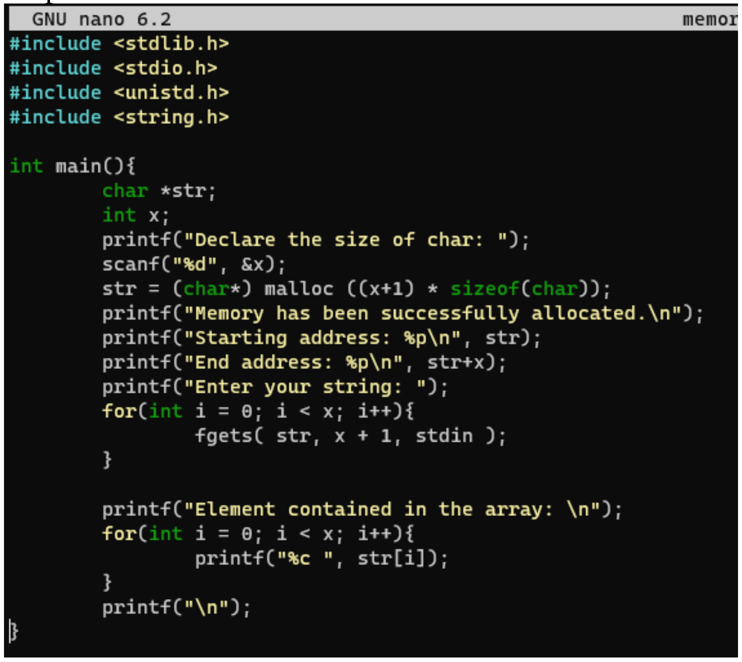
2. Write a program that accepts and displays a string value. The string should be stored in an array of characters allocated using the malloc function. First, the program accepts the length of string that needs to be stored as an input. Then, the program accepts the string to be displayed. Finally, the program displays the inputted string. You may use scanf and printf functions for this purpose.

   Step 1: Create the memory2.c using nano memory2.c



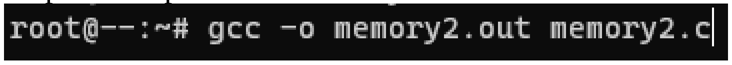
```
root@--:~# nano memory2.c
```

Step 2: Write the code



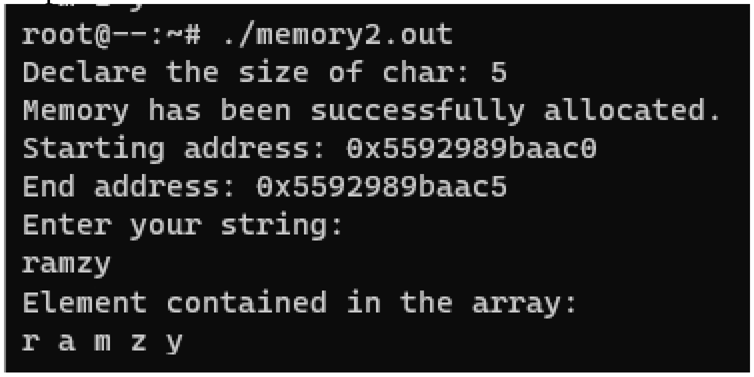
```
  GNU nano 6.2                                                       memor
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main(){
        char *str;
        int x;
        printf("Declare the size of char: ");
        scanf("%d", &x);
        str = (char*) malloc ((x+1) * sizeof(char));
        printf("Memory has been successfully allocated.\n");
        printf("Starting address: %p\n", str);
        printf("End address: %p\n", str+x);
        printf("Enter your string: ");
        for(int i = 0; i < x; i++){
                fgets( str, x + 1, stdin );
        }

        printf("Element contained in the array: \n");
        for(int i = 0; i < x; i++){
                printf("%c ", str[i]);
        }
        printf("\n");
}
```

Step 3: Compile the code



```
root@--:~# gcc -o memory2.out memory2.c
```

Step 4: Run the code



```
root@--:~# ./memory2.out
Declare the size of char: 5
Memory has been successfully allocated.
Starting address: 0x5592989baac0
End address: 0x5592989baac5
Enter your string:
ramzy
Element contained in the array:
r a m z y
```

Step 5: Analyze

Answer: From the output above, observe that I declared the size of char with the value of 5. After that by using malloc(), it has successfully allocated with the starting address of 0x5592989baac0 until 0x5592989baac5 denoting the end address. Next I entered my own name with the length of 5 and the program will stored in an array which outputted character by character illustrated above.