C-Strings and Valgrind

COMP201 Lab Session
Spring 2021

KOÇ
UNIVERSITY

Valgrind

Valgrind is a programming tool used for:

- memory debugging
- memory leak detection
- profiling

Memory Allocated but Never Used

main.c

```c
#include <stdlib.h>
int main()
{
    char *x = malloc(100);
    return 0;
}
```

Finding Invalid Pointer Use With Valgrind

main.c

```c
#include <stdlib.h>

int main()
{
    char *x = malloc(10);
    x[10] = 'a';
    return 0;
}
```

Valgrind Command

valgrind --tool=memcheck --leak-check=yes *filename*

Output:

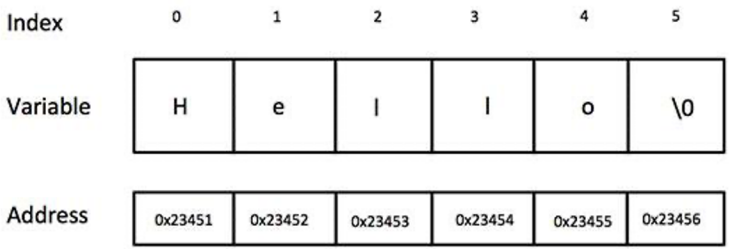
When 100 bytes are allocated but not used

```
==2330== 100 bytes in 1 blocks are definitely lost in loss record 1 of 1
==2330==    at 0x1B900DD0: malloc (vg_replace_malloc.c:131)
==2330==    by 0x804840F: main (main.c:5)
```

When Invalid pointer index is called

C-Strings

- 1-D array of characters
- Terminated by **null** or **\0**
- Initializing a String
  - char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
  - char greeting[] = "Hello";
  - char greeting[12] = "Hello";



String Functions in C

String functions	Description
strcat ( )	Concatenates str2 at the end of str1
strncat ( )	Appends a portion of string to another
strcpy ( )	Copies str2 into str1
strncpy ( )	Copies given number of characters of one string to another
strlen ( )	Gives the length of str1
strcmp ( )	Returns 0 if str1 is same as str2. Returns <0 if strl < str2. Returns >0 if str1 > str2
strcmpi ( )	Same as strcmp() function. But, this function negotiates case. “A” and “a” are treated as same.
strchr ( )	Returns pointer to first occurrence of char in str1
strrchr ( )	last occurrence of given character in a string is found
strstr ( )	Returns pointer to first occurrence of str2 in str1
strrstr ( )	Returns pointer to last occurrence of str2 in str1
strdup ( )	Duplicates the string
strlwr ( )	Converts string to lowercase
strupr ( )	Converts string to uppercase
strrev ( )	Reverses the given string
strset ( )	Sets all character in a string to given character
strnset ( )	It sets the portion of characters in a string to given character
strtok ( )	Tokenizing given string using delimiter

Using String functions

- Converting str1 to lowercase

  str1 = “Hello Comp201”;

  lcase = strlen(str1);

  printf("strlen(str1) : %d\n", len);

  //prints: ***strlen(str1) : 13***

- Concatenating two strings

  str1 = “Ahmed”;

  str2 = “Student”;

  strcat(str1, str2);

  printf("strcat(str1, str2):   %s\n", str1);

  //prints: *strcat(str1, str2): AhmedStudent*

Using String functions

- Converting str1 to Lowercase

  str1 = “Hello Comp201”;

  lwr = strlwr(str1);

  printf("strlwr(str1) : %s\n", lwr);

  //prints: ***strlwr(str1) : hello comp201***

- Comparing two strings

  str1 = “Ahmed”;

  str2 = “ahmed”;

  Str3 = strcmpi(str1, str2);

  printf("strcmpi(str1, str2):   %d\n", str3);

  //prints: *strcat(str1, str2): 0*

Strings In Memory

- Strings is a char array in the memory. We can change each character because we can change contents of array.
- There is a difference between char * and char []:
  - When a string is created as a char *, its characters cannot be modified because its memory lives in the data segment. We can set a char * equal to another value, because it is a reassignable pointer.
  - We cannot set a char[] equal to another value, because it is not a pointer; it refers to the block of memory reserved for the original array. If we pass a char[] as a parameter, set something equal to it, or perform arithmetic with it, it’s automatically converted to a char *.

Treating like an Array

- Find length without using strlen()

```
/*
 * We define a function countChars that counts the characters in the
string str
 * returns the last index i
 */
int countChars(char str[])
{
        int i=0;

        while ( str[i]! = '\0' ){
                        i++;
        }
        return i;
}
```

Print individual characters of string in reverse order

```
void main(){
    char str[100]; /* Declares a string of size 100 */
    int l,i;
       printf("Input the string : ");
       fgets(str, sizeof str, stdin);
            l=strlen(str);
            printf("The characters of the string in reverse are : \n");
       for(i=l ; i>=0 ; i--){
          printf("%c  ", str[i]);
        }
    printf("\n");
}
```