CS 240 Programming in C

Strings

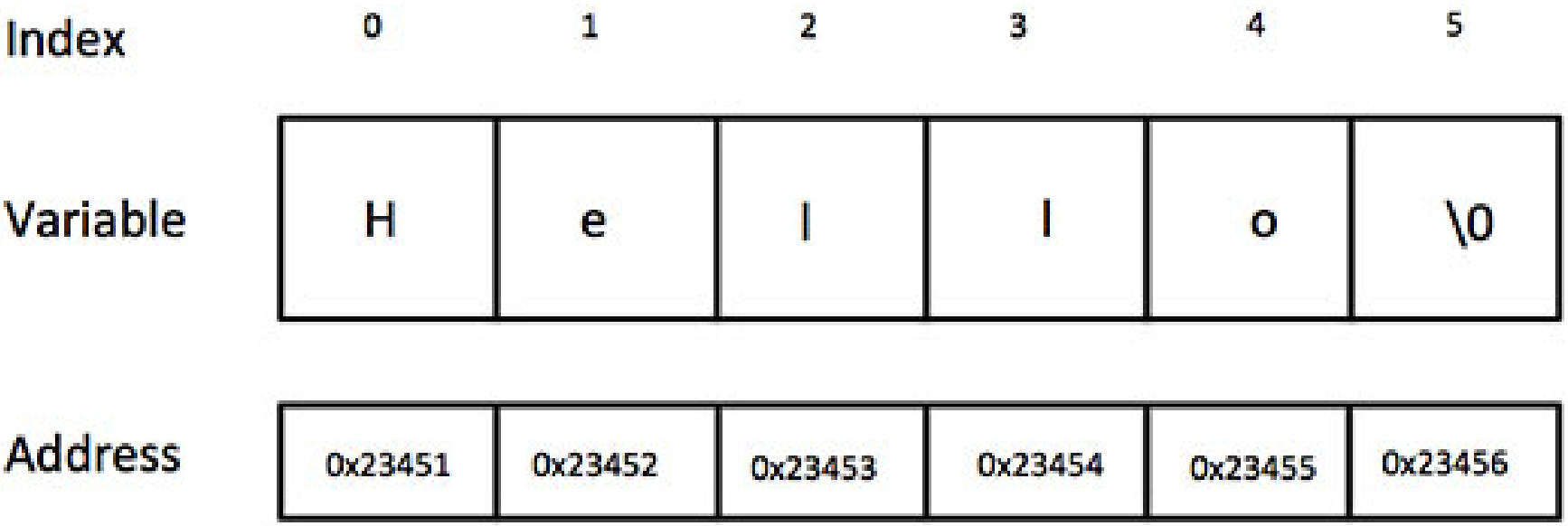
September 29, 2022

Strings in C

String in C programming is a sequence of characters terminated with a null character /0. Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a unique character /0. [1]

[1] https://www.geeksforgeeks.org/strings-in-c/

String in C



2

[2] https://www.tutorialspoint.com/cprogramming/c_strings.htm

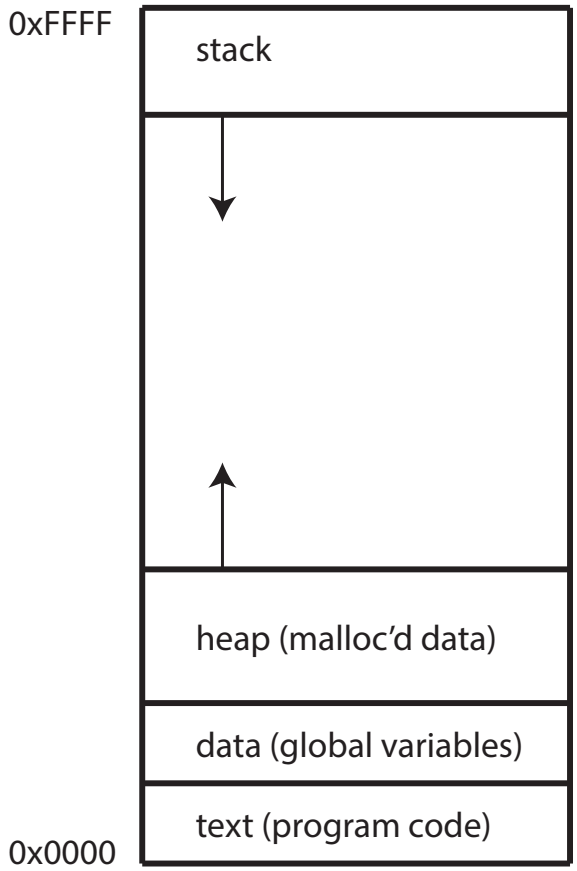
Initialization

Different ways to initialize string in C

- char str[] = "CS240";
- char str[] = { 'C', 'S', '2','4', '0', ''};
- char str[10] = "CS240";
- char str[10] = { 'C', 'S', '2','4', '0', '' };

Runtime Memory Layout

- Text segment: machine instructions of program
- Data segment: constants, global variables (static or not) and local static variables
- Heap: dynamically allocated space, `malloc()`
- Stack: runtime call stack
  - Actual parameters (arguments) for functions
  - Local (automatic) variables
  - Return address
  - Returned value



Stack frame

- C program's stack segment is consisted of nested function frames or stack frames.
- the main frame is the most outside frame, and the starting point of the program.
- Each function call will start one new frame in the stack, and after exit of function, its stack frame gets destroyed with every thing in it.
- Stack frames works like stack structure, the first created frame gets destroyed at last, which is the main, and the last created stack frame gets destroyed first. FILO.
- Stack frame growing from high address to low address.

Storage for Strings

Strings declared as character arrays are stored in C like other types of arrays. If str[] is an auto variable, it is stored in the stack segment; if it is a global or static variable, it is stored in the data segment.

```
char str[4] = "ABC"; /*One extra for string terminator*/
/* OR */
char str[4] = {'U', 'M', 'B', '\0'};
```

Storage for Strings

Read only string in a shared segment.

```
char *str = "ABC";
```

Storage for Strings

Dynamically allocated in heap segment.

```
char *str;
int size = 4; /*one extra for ‘\0’*/
str = (char *)malloc(sizeof(char)*size);
*(str+0) = ’A’;
*(str+1) = ’B’;
*(str+2) = ’C’;
*(str+3) = ’\0’;
```

Fgets and puts

```
#include <stdio.h>
int main()
{
    char name[30];
    printf("Enter name: ");
    fgets(name, sizeof(name), stdin);  // read string
    printf("Name: ");
    puts(name);    // display string
    return 0;
}
```

String Functions

- strcpy(s1, s2) - Copies string s2 into string s1.
- strcat(s1, s2) - Concatenates string s2 onto the end of string s1.
- strlen(s1) - Returns the length of string s1.
- strcmp(s1, s2) - Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
- strchr(s1, ch) - Returns a pointer to the first occurrence of character ch in string s1.
- strstr(s1, s2) - Returns a pointer to the first occurrence of string s2 in string s1.
- strdup(s) - Returns a pointer to the duplicated string s.
- strtok(s, c) - Returns a pointer to the first token found in the string.

strchr vs strpbrk

```
char *strpbrk(const char *s, const char *accept);

  The strpbrk() function locates the first occurrence
  in the string s of any of the bytes in the string accept.
```

```
char *strchr(const char *s, int c);

  The strchr() function returns a pointer to
  the first occurrence of the character c in the string s.
```

strtok

```
char * strtok ( char * str, const char * delimiters );
```

A sequence of calls to this function split str into tokens, which are sequences of contiguous characters separated by any of the characters that are part of delimiters.
On a first call, the function expects a C string as argument for str, whose first character is used as the starting location to scan for tokens. In subsequent calls, the function expects a null pointer and uses the position right after the end of the last token as the new starting location for scanning. [3]

[3] https://cplusplus.com/reference/cstring/strtok/

Code Samples

- strcpy - compare_string.c
- strcat - join_string.c
- strlen - strlen.c
- strcmp - compare_string.c
- strcoll - compare_string.c
- strdup - duplicate_string.c
- strpbrk - char_match.c
- strchr - char_match.c
- strspn - match_substring.c