

Strings



Advanced C

S_____s - Fill in the blanks please ;)



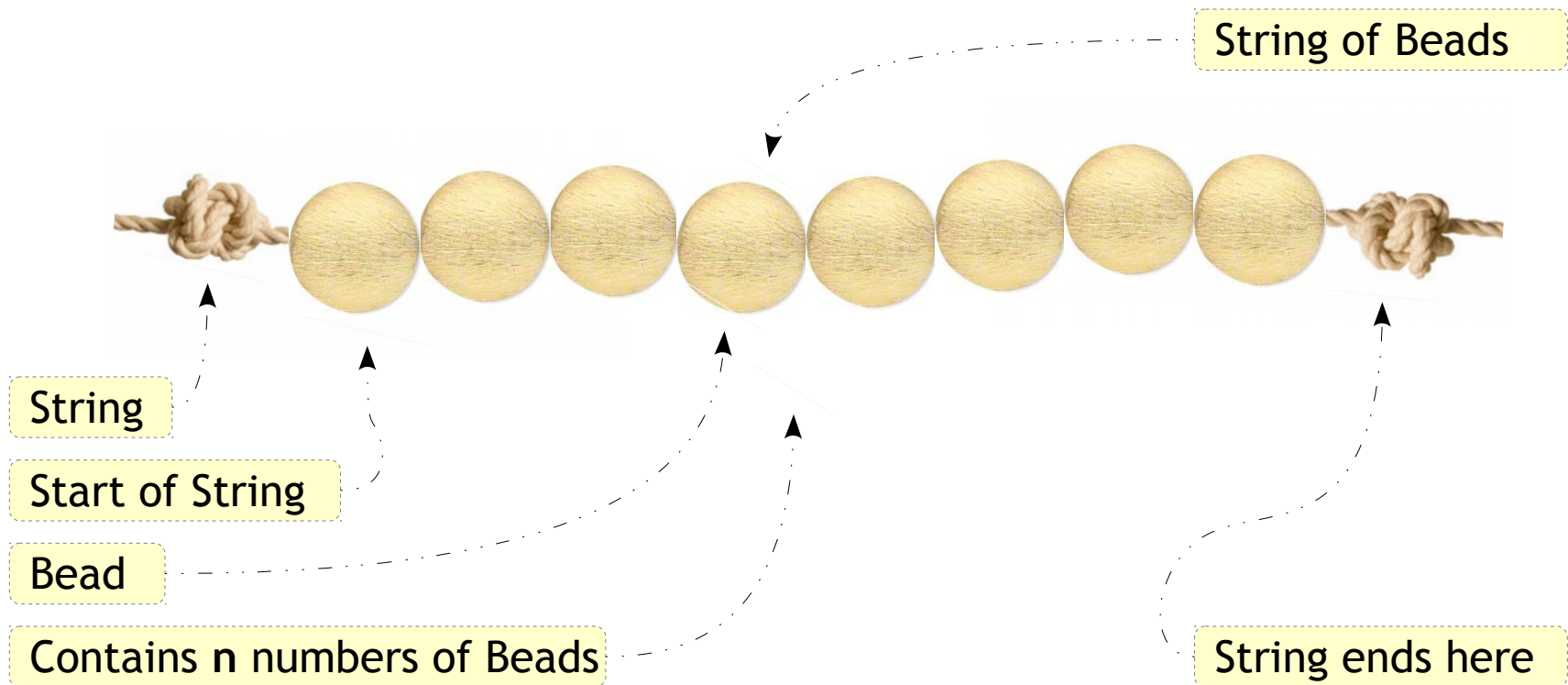
Advanced C

Strings



A set of things tied or threaded together on a thin cord

Source: Google



Advanced C

Strings



- Contiguous sequence of characters
- Stores printable ASCII characters and its extensions
- End of the string is marked with a special character, the null character '\0'
- '\0' is implicit in strings enclosed with “”
- Example

“You know, now this is what a string is!”

Advanced C

Strings



- Constant string
 - Also known as string literal
 - Such strings are read only
 - Usually, stored in read only (code or text segment) area
 - String literals are shared
- Modifiable String
 - Strings that can be modified at run time
 - Usually, such strings are stored in modifiable memory area (data segment, stack or heap)
 - Such strings are not shared

Advanced C

Strings - Initialization



001_example.c

`char char_array[5] = {'H', 'E', 'L', 'L', 'O'};` ← Character Array

`char str1[6] = {'H', 'E', 'L', 'L', 'O', '\0'};` ← String

`char str2[] = {'H', 'E', 'L', 'L', 'O', '\0'};` ← Valid

`char str3[6] = {"H", "E", "L", "L", "O"};` ← Invalid

`char str4[6] = {"H" "E" "L" "L" "O"};` ← Valid

`char str5[6] = {"HELLO"};` ← Valid

`char str6[6] = "HELLO";` ← Valid

`char str7[] = "HELLO";` ← Valid

`char *str8 = "HELLO";` ← Valid

Advanced C

Strings - Memory Allocation


Example

```
char str1[] = {'H', 'E', 'L', 'L', 'O', '\\0'};
```

```
char *str2 = "Hello";
```

str1	
'H'	1000
'E'	1001
'L'	1002
'L'	1003
'O'	1004
'\\0'	1005

str2	
1000	996
?	997
?	998
?	999
'H'	1000
'E'	1001
'L'	1002
'L'	1003
'O'	1004
'\\0'	1005



Advanced C

Strings - Size



002_example.c

```
#include <stdio.h>

int main()
{
    char char_array_1[5] = {'H', 'E', 'L', 'L', 'O'};
    char char_array_2[] = "Hello";

    sizeof(char_array_1);
    sizeof(char_array_2);

    return 0;
}
```

The size of the array
is calculated so,

5, 6

003_example.c

```
int main()
{
    char *str = "Hello";

    sizeof(str);

    return 0;
}
```

The size of pointer
is always constant
so,
4 (32 Bit Sys)

Advanced C

Strings - Size



004_example.c

```
#include <stdio.h>

int main()
{
    if (sizeof("Hello" "World") == sizeof("Hello") + sizeof("World"))
    {
        printf("WoW\n");
    }
    else
    {
        printf("HuH\n");
    }

    return 0;
}
```

Advanced C

Strings - Manipulations



005_example.c

```
#include <stdio.h>

int main()
{
    char str1[6] = "Hello";
    char str2[6];

    str2 = "World";

    char *str3 = "Hello";
    char *str4;

    str4 = "World";

    str1[0] = 'h';
    str3[0] = 'w';

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

    return 0;
}
```

Not possible to assign a string to a array since its a constant pointer

Possible to assign a string to a pointer since its variable

Valid. str1 contains "hello"

Invalid. str3 might be stored in read only section.
Undefined behaviour

Advanced C

Strings - Sharing



006_example.c

```
#include <stdio.h>

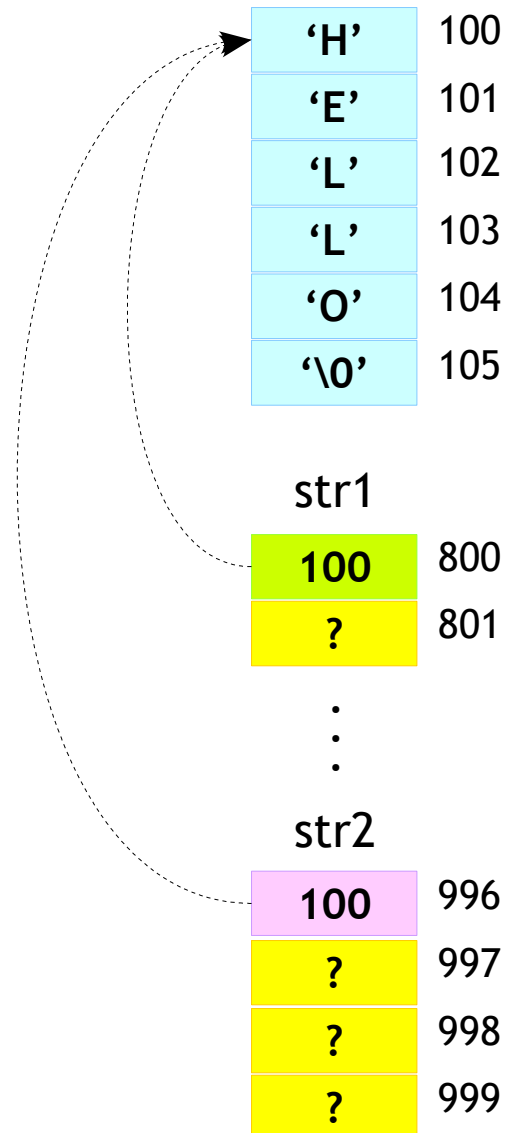
int main()
{
    char *str1 = "Hello";
    char *str2 = "Hello";

    if (str1 == str2)
    {
        printf("Hoo. They share same space\n");
    }
    else
    {
        printf("No. They are in different space\n");
    }

    return 0;
}
```

Advanced C

Strings - Sharing



Advanced C

Strings - Empty String



007_example.c

```
#include <stdio.h>
#include <string.h>

int main()
{
    char *str = "";
    int ret;

    ret = strlen(str);
    printf("%d\n", ret);

    return 0;
}
```

Advanced C

Strings - Passing to Function



008_example.c

```
#include <stdio.h>

void print(const char *str)
{
    while (*str)
    {
        putchar(*str++);
    }
}

int main()
{
    char *str = "Hello World";

    print(str);

    return 0;
}
```

Advanced C

Strings - Reading



009_example.c

```
#include <stdio.h>

int main()
{
    char str[6];

    gets(str);
    printf("The string is: %s\n", str);

    return 0;
}
```

- The above method is not recommended by the gcc. Will issue warning while compilation
- Might lead to stack smashing if the input length is greater than array size!!

Advanced C

Strings - Reading



010_example.c

```
#include <stdio.h>

int main()
{
    char str[6];

    fgets(str, 6, stdin);
    printf("The string is: %s\n", str);

    scanf("%5[^\n]", str);
    printf("The string is: %s\n", str);

    return 0;
}
```

- fgets() function or selective scan with width are recommended to read string from the user

Advanced C

Strings - DIY



- WAP to calculate length of the string
- WAP to copy a string
- WAP to compare two strings
- WAP to compare two strings ignoring case
- WAP to check a given string is palindrome or not

Advanced C

Strings - Library Functions



Purpose	Prototype	Return Values
Length	<code>size_t strlen(const char *str)</code>	String Length
Compare	<code>int strcmp(const char *str1, const char *str2)</code>	$str1 < str2 \rightarrow < 0$ $str1 > str2 \rightarrow > 0$ $str1 = str2 \rightarrow = 0$
Copy	<code>char *strcpy(char *dest, const char *src)</code>	Pointer to dest
Check String	<code>char *strstr(const char *haystack, const char *needle)</code>	Pointer to the beginning of substring
Check Character	<code>char *strchr(const char *s, int c)</code>	Pointer to the matched char else NULL
Merge	<code>char *strcat(char *dest, const char *src)</code>	Pointer to dest

Advanced C

Strings - Quiz



- Can we copy 2 strings like, `str1 = str2`?
- Why don't we pass the size of the string to string functions?
- What will happen if you overwrite the `'\0'` (null character) of string? Will you still call it a string?
- What is the difference between `char *s` and `char s[]`?

Advanced C

Strings - Quiz - Pointer vs Array



Pointer	Array
A single variable designed to store address	A bunch of variables; Each variable is accessed through index number
Size of pointer depends on size of address (Ex - 32 bit or 64 bit)	Size of Array depends on number of elements and their type
Pointer is a lvalue - <ul style="list-style-type: none">Pointers can be modified (can be incremented/decremented or new addresses can be stored)	Array name is not lvalue - <ul style="list-style-type: none">Array name represents either whole array (when operand to sizeof operator) or base address (address of first element in the array)Can't be modified (Array can't be incremented/decremented or base address can't be changed)

Advanced C

Strings - DIY



- WAP to reverse a string
- WAP to compare string2 with string1 up to n characters
- WAP to concatenate two strings

Advanced C

Strings - DIY



- Use the standard string functions like
 - strlen
 - strcpy
 - strcmp
 - strcat
 - strstr
 - strtok

Advanced C

Strings - DIY



- WAP to print user information -
 - Read : Name, Age, ID, Mobile number
 - Print the information on monitor
 - Print error “Invalid Mobile Number” if length of mobile number is not 10
- WAP to read user name and password and compare with stored fields. Present a puzzle to fill in the banks
- Use strtok to separate words from string “www.emertxe.com/bangalore”