Foundations of C Programming (Structured Programming)

- String(字符串)

Outline

- String concept
- Input and output a string
- String functions

String

- A string is an array of chars terminated (终结) with a null character, '\0'.
- String declaration
  - E.g.,
    - char `str[10];`
    - a string that contains at most 9 characters and 1 for '\0'
- Several ways to initialize a string
  - `char str[20] = {'H', 'e', 'l', 'l', 'o', '\0'};`
  - `char str[20] = "Hello";` // '\0' is automatically set to str[5]
  - `char str[] = "Hello";` //str's size is 6
- After the string declaration, the array name can be taken as a string variable (字符串变量).

Class Exercise

- Are the following initialization (初始化) correct?
  - `char str[10] = {'y', 'e', 's'};`
  - `char str[10] = "Good Morning";`
  - `char str = "Hi";`
  - `char str[] = 'Good';`
  - `char str[] = "O";`

Output (Write) A String

- To output a string to the screen, two functions can be used
  - printf (vs. sprintf)
  - puts
- These functions are defined in stdio library, so we need to include stdio.h in the program

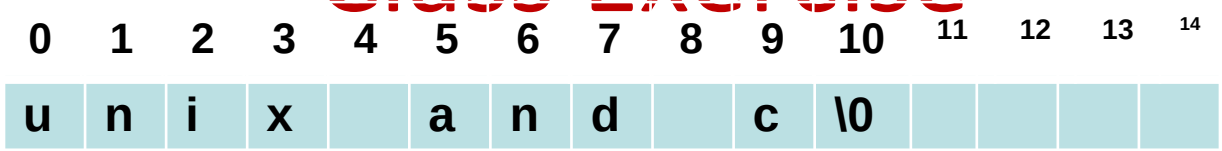
printf

- %s must be used in the format string to indicate a string is to be printed
  - E.g.,

```
char str[] = "This is a message";
printf("%s", str);
```

- printf stops printing when it meets '\0'.

Class Exercise



- What is the output of the following code

```
char str[15]="unix and c";

printf("%s", str);
printf("\n");

str[6]='\0';
printf("%s", str);
printf("\n");

str[2]='%';
printf("%s", str);
printf("\n");
```

What is the output?

sprintf Function

- Similar to `printf`. The function does not produce output to screen, but to a string variable.
- Composes a string with the same text that would be printed.
- Format
  - `int sprintf( char *str, const char *format, ... );`
  - An example

```
char buffer[50];
int n, a = 5, b = 3;

n = sprintf(buffer, "%d plus %d is %d", a, b, a+b);
```

buffer: "5 plus 3 is 8"
n: 13

puts

- Simpler than `printf`
  - E.g.,

```
char str[] = "This is a message";
puts(str);
```

- puts stops printing when it meets '\0'.

Class Exercise

```
char name1[16] = "Frans Coenen";
char name2[16] = {'F','r','a','n','s',' ','C','o','e','n','e','n','\0'};
char name3[16] = {70,114,97,110,115,32,67,111,101,110,101,110,0};

puts(name1);
puts(name2);
puts(name3);
```

What is the output?

Input (Read) A String

- To input a string from the keyboard, two functions can be used
  - scanf
  - gets
- Both functions are in stdio library, so we need to include stdio.h in the program

scanf

- %s or %*ns* is used in the format string to indicate a string is to be read
  - %s: scans up to the next space(空格) character
  - %ns: scans up to *n* characters or the next space character (空格) depending on which comes first.

```
char str1[20], str2[20], str3[20];
scanf("%s%2s%10s", str1, str2, str3);
```

  - E.g.,

    Assume the input is

    ```
    UIC Computer Science Students
    ```

    Then, `str1 = "UIC"`
    `str2 = "Co"`
    `str3 = "mputer"`

Attention: No & is put before the string variable.

Class Exercise

- What is the output of the following code?

```
char lName[81], fName[81];
int id_num;

puts("Enter the last name, firstname, ID number
       separated by spaces ");
puts("then press Enter \n");
scanf("%s%s%d", lName, fName, &id_num);
printf("3 items entered: %s %s %d\n",
       fName,lName,id_num);
```

The input is

```
Tony Towey 23456
```

gets

- gets( ) gets a line from the standard input.
  - E.g.,

```
char your_line[100];
printf("Enter a line:\n");
gets(your_line);
puts("Your input follows:\n");
puts(your_line);
```

- Be careful, do not overflow the string buffer (exceed the size of array). Instead, the following is safe
  - `fgets(your_line, sizeof(your_line), stdin)`

String Functions

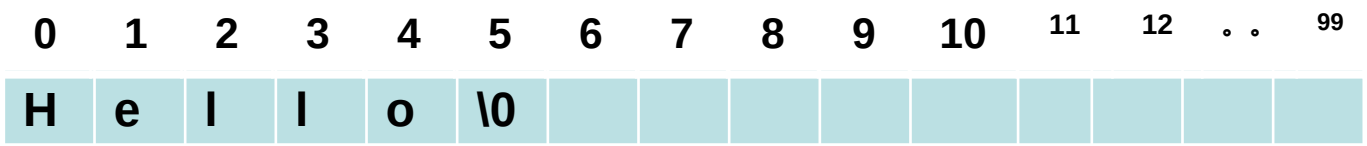
- Common string functions include functions
  - computing the length of a string (strlen)
  - copying strings (strcpy, strncpy)
  - comparing strings (strcmp, strncmp)
  - concatenating strings (strcat)
  - more …
- To use these functions, we must include the file string.h, e.g., #include <string.h>

strlen

- Call: strlen(str);
- Objective : Calculating the length of *str*
- Return: length of str, not including '\0';
- E.g.,

```
char your_line[100] = "Hello";
int len;

len = strlen(your_line);
printf("The length of your_line is %d", len);
```

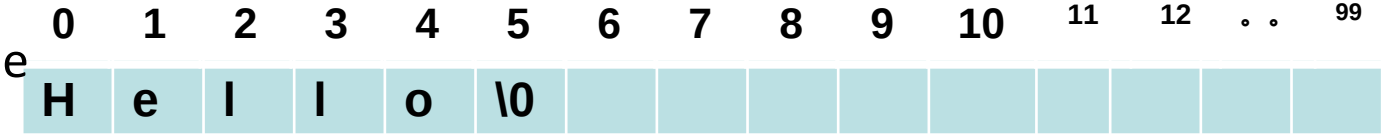


- `The length of your_line is 5`

strcpy

- Call: strcpy(destination, source);
- Objective : copy string from *source* to *destination*
- Return: same as destination
- E.g.,

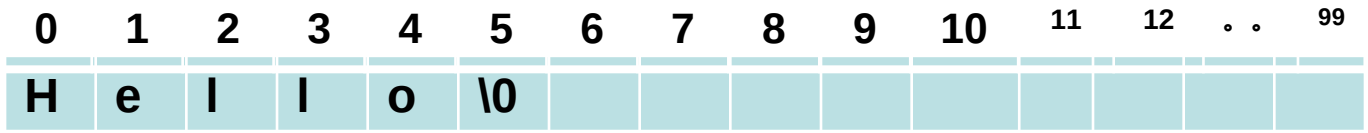
Your_line




```
char my_line[100];
char your_line[100] = "Hello";

strcpy(my_line, your_line)
printf("my_line is %s", my_line);
```

my_line



- `my_line is Hello`
- Attention: *destination* should have enough room to store the string.

strncpy

- Call: strncpy(destination, source, n);
- Objective : copy *n* characters from *source* to *destination*
- Return: same as *destination*
- E.g.,

```
char my_line[100];
char your_line[100] = "Hello";

strncpy(my_line, your_line, 2)
my_line[2] = '\0';
printf("my_line is %s", my_line);
```

- `my_line is "He"`
- Attention: *destination* should have enough room to store the string
- '\0' must be added to the end.

strcmp

- Call: strcmp(str1, str2);
- Objective : Compare *str1* and *str2* based on ASCII
- Return: < 0 if str1 is less than str2
  = 0 if str1 equals str2
  > 0 if str1 is greater than str2
- E.g.,

```
char my_line[100] = "Hello World";
char your_line[100] = "Hello world";

int r;
r = strcmp(my_line, your_line);
```

str1[0]	str1[1]	str1[2]	str1[3]	str1[4]					
str2[0]	str2[1]	str2[2]	str2[3]	str2[4]					

Compare until str1[i]!=str2[i]

- r > 0 or r < 0 or r = 0?

strncmp

- Call: strncmp(str1, str2, n);
- Objective : Compare *n* characters of *str1* and *str2* based on ASCII
- Return: < 0 if str1 is less than str2
  = 0 if str1 equals str2
  > 0 if str1 is greater than str2

  (the beginning *n* characters)
- E.g.,

```
char my_line[100] = "Hello World";
char your_line[100] = "Hello world";
int r1, r2;

r1 = strncmp(my_line, your_line, 6);
r2 = strncmp(my_line, your_line, 7);
```

- r1? r2?

strcat

- Call: strcat(destination, source);
- Objective : add source to destination
- Return: concatenated (连接) string, same as destination
- E.g.,

```
char my_line[100] = "Hello ";
char your_line[100] = "world";

strcat(my_line, your_line);
printf("The linked string is %s", my_line);
```

- The linked string is `Hello world`

Char, String, Number

- We must distinguish clearly numbers, chars, strings.
  - '1': char
  - "1": string
  - 1: number
- Are the following expressions correct?
  - str[0] = "h";
  - printf('\n');
  - Str[10] = 'hello';

String to Number Functions

- Some functions can transfer number string to numbers
  - atoi
  - atof
- If use these functions, must include stdlib.h, e.g., #include<stdlib.h>

atoi

- Call: atoi(str)
- Objective: convert *str* to an int number, starting at beginning and continuing until something non-convertible is encountered
- Space, +, - are acceptable
- E.g.,

String	Value returned
"157"	157
"-1.6"	-1
"+50x"	50
"twelve"	0
"x506"	0

atof

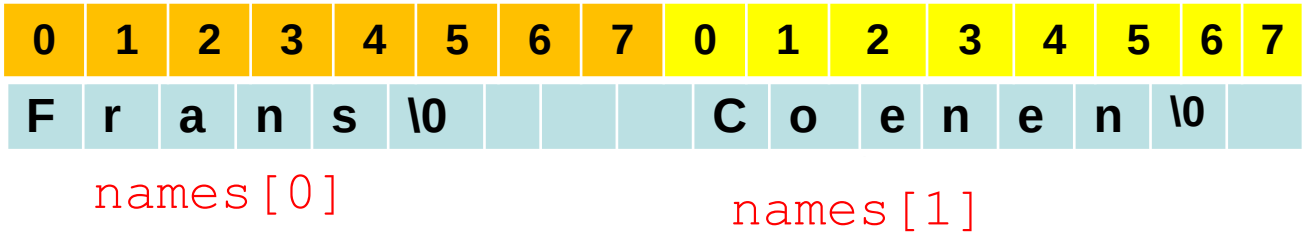
- Call: atof(str)
- Objective: convert *str* to a float number, starting at beginning and continues until something non-convertible is encountered
- Space, +, - are acceptable
- An E or e (exponent) is acceptable
- A decimal point is acceptable
- E.g.,

String	Value returned
"12"	12.000000
"-0.123"	-0.123000
"123E+3"	123000.000000
"123.1e-5"	0.001231

Array of Strings

- Each element of an array is a string
- It is usually declared as a two-dimensional array of chars

```
#include< stdio.h>
void main(void)
{
  char names[2][8] = {"Frans", "Coenen"};
  printf("names = %s, %s\n",names[0],names[1]);
  printf("Initials = %c.%c.\n",names[0][0],names[1][0]);
 }
```



Implementing String Functions

- We can write code to implement(实现，写) the string functions too
- The following function is similar to strcpy function.

```
void copyString(char dest[], char src[])
{
  int i = 0;
  while(src[i] != '\0') {
    dest[i] = src[i];
    i++;
  }
  dest[i] = '\0';
}
```

	src	dest
0	H	H
1	E	E
2	L	L
3	L	L
4	O	O
5	\0	\0

After the class, try to write your own functions to serve the same objectives as strlen() and strcmp() !

Summary

- String is a set of characters ending with '\0'
- Different ways can be used to initialize a string
- String functions are very helpful in handling string operations.