



# Strings!

CS2263 – Systems Software Development

1

## Lecture Learning Outcomes

At the conclusion of this presentation students should be able to:

- Explain how strings are represented and stored in C
- Assign values to strings and perform simple operations on them.
- List some common string functions and what they do



2

## References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York. Pp 9-27 (Chapter 6)
- Kernighan, B and D Ritchie. 1988. The C Programming Language. Prentice Hall. 272pp (Section 5.5, Appendix B, Section B3)



3

## What is a String

- Characteristics of a string in real-life
  - A set of characters
  - What could a string be used to hold?
  - How long is a string?
- String Representation in C
  - A block of memory holding the characters
  - Character access as array elements
  - End-of-string represented with `NULL`
- The C library (`string.h`, `stdlib.h`, `ctype.h`) provides a collection of functions for working with strings.



4

## String Constants/Literals

```
//constants.c
char *name = "Eric";
char *p;
p = "abc";
char *q = "abc";
```

### String constants must not be modified

- Strings being stored once by the compiler



5

## Character arrays versus Character Pointers

```
char date[] = "June 14";
```

- Declares date to be an array of characters
- Characters can be modified, like the elements of any array
- date is an array name; a **pointer constant** (a const pointer; address cannot change)

```
char *date = "June 14";
```

- Declares date to be a pointer to a string constant
- Characters shouldn't be modified
- date is a **pointer variable** that can be made to point to other strings during program execution.



6

## End-of-String Character

```
char date [] = "June 14";
```

J	u	n	e		1	4	\0
---	---	---	---	--	---	---	----

NULL, '\0'

- How many strings are in this array?
- Implications of this statement?

```
date[4] = NULL;
```



7

## Processing Strings

- String functions only need the pointer to a string. Why?

```
//same as strlen in string.h
int length(const char *s){
    const char *p; //warning if p not const
    for(p=s; *p; p++) // *p != '\0'
        ;
    return p - s;
}
```

- How long is date?
- How much storage does it consume?
- What are the indices of the actual string value?



8

## String Functions (I)

```
char *strcpy(char *dest, const char *src);
```

- `strcpy` does not verify that enough space in `dest`

```
char *strncpy(char *dest, const char *src, size_t n);
```

- Copies only `n` characters, terminates `dest` string

```
int strcmp(const char *str1, const char *str2);
```

Compares ASCII character codes and returns:

- `<` if `str1` less than `str2`
- `0` if `str1` equals `str2`
- `>` if `str1` greater than `str2`



9

## String Functions (II)

```
char *strstr(const char *haystack, const char *needle);
```

- Returns pointer to first occurrence of `needle` in `haystack`
- Returns `NULL` if `needle` is not a substring of `haystack`

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

- converts `int c` to `char`, returns pointer to first occurrence of character or `NULL`



10

## String i/O

### Character by character

- `getchar()`, `fgetc()`
- `putchar()`, `fputc()`

### Line by line:

- `fgets()` ► hazards?



11

## Character I/O

- `int getchar()` to input a single character
  - returns int value of character or EOF if at end of file
- `int putchar(int)` to output a single character

```
int c;
c = getchar();
while(c != EOF)
{
    putchar(c);
    c = getchar();
}
```



12

## Line-Oriented I/O

```
char *fgets(char *buf, int n, FILE *in);
```

- Reads a line from the file in (can use `stdin` for terminal input), and stores it in the block pointed to by `buf`. Stops reading when:
  - `n-1` characters have been read
  - end-of-line has been encountered; (NULL is stored at the end of string)
  - end-of-file has been encountered.
- In any case, `buf` is always properly terminated (NULL is stored).
- The function returns `buf` if successful and NULL if no
- characters have been read or there has been a reading error.



13

## Print the Longest Line

```
#define LINEL 200
int main(void){
    char line[LINEL+1]
    char longest[LINEL+1]={'\0'};
    int maxline = 0
    int length;
    while(fgets(line, LINEL+1, stdin) != NULL){
        length = strlen(line)-1;
        if(length > maxline){
            maxline = length;
            strcpy(longest, line);
        }
    }
    printf("longest line: %s", longest);
    return 0;
}
```



14

## String to Number Conversions

```
double strtod(const char *s, char **p);
long strtol(const char *s, char **p, int base);
```

- Convert a string *s* to a number. If the conversion failed: *\*p* is set to the value of the original string *s*
- Otherwise, *\*p* is set to point to the first character in the string *s* immediately following the converted part of this string.
- A default base, signified by *0*, is decimal, hexadecimal or octal, and it is derived from the string. (It also may be any number from 2 to 36).



15

## Conversion Example

```
#include <stdio.h>
#include <stdlib.h>
#define LINEL 200
int main(void){
    char line[LINEL+1];
    printf("enter 2 numbers: ");
    fgets(line, LINEL+1, stdin);
    char *p;
    double d1 = strtod(line, &p);
    double d2 = strtod(p, NULL);
    printf("your 2 numbers are %f and %f\n", d1, d2);

    printf("enter a binary integer: ");
    fgets(line, LINEL+1, stdin);
    int num = strtol(line, NULL, 2);
    printf("your number is %d\n", num);

    return 0;
}
```



16



## Arguments of main()

- `int main(void);`
- `int main(int argc, char **argv);`
- `int main(int argc, char *argv[]);`

