

Strings in C

Strings

A string is an Null-Terminated Character Array

Allocate space for a string just like any other array:

```
char outputString[16];
```

Space for string must contain room for **terminating zero**

Special syntax for initializing a string:

```
char outputString[] = "Result = ";
```

...which is the same as:

```
outputString[0] = 'R';
outputString[1] = 'e';
outputString[2] = 's';
...
outputString[9] = '\0'; // Null terminator
```

CIT 593

2/17

I/O with Strings

```
char outputString [ ] = "Hello";
```

```
char inputString [100];
```

Printf and scanf use "%s" format character for string

Printf -- print characters up to terminating zero

```
printf("%s", outputString);
```

Note: Just like PUTS in LC3

Scanf -- read characters until whitespace,
store result in string, and terminate with zero

```
scanf("%s", inputString);
```

Why no & operator?

CIT 593

3/17

Strings

Although there is not string data type in C, C has library `<string.h>` that can perform actions on strings.

All the functions in `<string.h>` have parameters or return values as character arrays terminated with null character

Examples of String functions

- `strlen` – returns string length
- `strcpy` – copy one string to another location

CIT 593

4/17

String Length - Array Style

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

CIT 593

5/17

String Length - Pointer Style

```
int strlen(char* str)
{
    int i = 0;
    while (*str != '\0') {
        i++;
        str++;
    }
    return i;
}
```

Note: array and pointer declarations interchangeable as **function formal parameters** because the whole array is never actually passed to a function (the address of the array is)

CIT 593

6/17

Usage of strlen

```
#include<string.h>
#include <stdio.h>
int main(){
    char array[] = "Hello";
    for(i = 0; i < strlen(array);i++){
        printf("%c\n",array[i]);
    }
}
```

Output:

H
e
l
l
o

CIT 593

7/17

String Copy - Array Style

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

CIT 593

8/17

String Copy - Array Style #2

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

CIT 593

9/17

String Copy - Pointer Style

```
void strcpy(char* dest, char* src)
{
    while ((*dest = *src) != '\0') {
        dest++;
        src++;
    }
}
```

CIT 593

10/17

String Copy - Pointer Style #2

```
void strcpy(char* dest, char* src){
    while ((*dest++ = *src++) != '\0') {
        // nothing
    }
}
```

Note:

*src++ is the character that src point before src was incremented (postfix ++ doesn't change src until after this character has been fetched)

Difficult to read

- "Experienced C programmers would prefer..." - K&R
- However confusing: try avoid this type of code

What happens if dest is too small?

- Bad things...

CIT 593

11/17

C String Library

C has a limited string library

- All based on null-terminated strings
- #include <string.h> to use them

Functions include

1. int strlen(char* str)
2. void strcpy(char* dest, char* src)
3. int strcmp(char* s1, char* s2)
 - Returns 0 on equal, -1 or 1 if greater or less
 - Remember, 0 is false, so equal returns false!

CIT 593

12/17

More String Library Functions

1. `strcat(char* dest, char* src)`
 - string concatenation (appending two strings)
2. `strncpy(char* dest, char* src, int max_length)`
3. `strncmp(char* s1, char* s2, int max_length)`
4. `strncat(char* dest, char* src, int max_length)`

Plus some more...

CIT 593

13/17

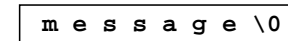
String Declaration

What's the difference between:

- `char amessage[] = "message"`
- `char *pmessage = "message"`

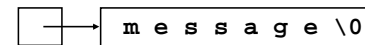
Answer:

- `char amessage[] = "message" // single array`



```
m e s s a g e \0
```

- `char *pmessage = "message" // pointer and array`



```
→ m e s s a g e \0
```

CIT 593

14/17

Main(), revisited

Main supports command line parameters

In Java

```
public static void main(String[] args)
```

In C:

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

An array of strings

By convention `argv[0]` is name by which program is invoked so `argc` is atleast 1.

CIT 593

15/17

Disassembler program – hw5part2

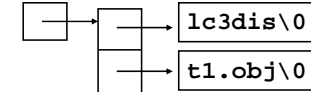
```
int main(int argc, char **argv) {
    FILE* f;

    if (argc != 2) {
        printf("Usage: %s <obj-file>\n",
            argv[0]);
        return 1;
    }

    f = fopen(argv[1], "r");
    if (f == NULL) {
        printf("Error opening file %s\n",
            argv[1]);
        return 1;
    }
    .....//rest of the code
}
```

```
gcc -o lc3dis lc3dis.c main.c
lc3dis t1.obj
```

argv:



```
→ lc3dis\0
→ t1.obj\0
```

What is:

`argv[0][0] = ?`

`argv[1][2] = ?`

CIT 593

16/17

When manipulating text in C

Use char array to hold strings

Extremely important:

- If you creating temporary arrays to copy string, make sure that array size must be string length + 1
- why??

Use the string library (string.h) when possible instead of creating your own functions