C Strings

Kurt Schmidt
Skipjack Solutions
November 1, 2021
Intended audience: Student who has working knowledge of Python
Target compiler: I’ll try to center the discussion on C99 using gcc 7.4
- Code examples might have an accompanying link
  - Follow link to step through example at pythontutor.com
  - Does a nice job of graphically showing variables in memory, the heap, and the stack
Strings
Introduction

- Stored in a character array
- String (in the array) is null-terminated
  - Special sentinel, the *null-terminator character*, (\0), marks the end of the string
- Use `strcpy` to assign into a string
  - Destination must be large enough for string, plus 1 for ’\0’

```c
int main( int argc, char *argv[] )
{
    char name[10] ;
    strcpy( name, "Kurt" ) ;

    printf( "My name is %10s\n", name )

    return 0 ;
}
```
Initialisation

String Initialisation

Only use the assignment operator ( = ) on strings at initialisation.

```c
int main( int argc, char *argv[] )
{
    char last[20] = "Schmidt" ;
    char first[] = { 'K', 'u', 'r', 't', '\0' } ;

    printf( "My first name is %12s, last is %20s\n", first, last )

    return 0 ;
}
```

- last array has size 20 (can hold 20 characters)
  - Can only store a string of length 19
- first is of just sufficient size
  - 5
Library Functions
Basic Library Functions

All available in `<string.h>`

- `int strlen( char* s )` – Returns length of `s`
- `char* strcpy( char* d, char* s )` – How we assign. Copies contents from `s` to `d`¹
- `char* strcat( char* d, char* s )` – Appends contents of `s` to `d`
- `int strcmp( char* s1, char* s2 )` – Compare 2 strings. Return -1 if `s1 < s2`, 0 if equal, and 1 if `s1 > s2`

String Assignment

Do not use the assignment operator (=) for strings. Use `strcpy`.

¹ Assumes `d` has sufficient space
More Library Functions

All available in `<string.h>`

- `char* strchr(char* s, char c)` – Return pointer to first occurrence of `c` in `s`
- `char* strstr(char* haystack, char* needle)` – Substrings. Return pointer to first occurrence of `needle` in `haystack`
- `char* strdup(char* s)` – Duplicates `s` to heap memory, returns pointer to new memory
  - We’ve not discussed heap memory yet

---

Actually passed as an `int`, cast to a `char`
Command-Line Arguments
Command-Line Arguments

- In `main`, `argv` is an array of `char*` (strings)
- There are `argc` of them
- `argv[0]` is the program name (invocation)
  - Note, `argc` is always $\geq 1$

```c
int main( int argc, char *argv[] )
{
    int i ;

    for( i=0; i<argc; ++i )
        printf( "%3d  %s\n", i, argv[i] )

    return 0 ;
}
```
Practise Functions
// strcpy - copies contents of s into t
// s must be null-terminated
// t must point to buffer of sufficient size

void myStrcpy( char* t, char* s )
{
    while( *s != '\0' ) {
        // *(t++) = *(s++) ;
        *t = *s ;
        ++t ;
        ++s ;
    }
    *t = *s ; // copy \0 over
}

int main()
{
    char name[30] ;

    myStrcpy( name, "Jaga" ) ;

    return 0 ;
}