Personal SE

Strings & Command Line Arguments

Strings in C

- A string is just an array of chars:

```
char welcome[] = "Hello" ;  // C permits this
```

- This is an array of 8-bit bytes holding ASCII characters.

Strings in C

- A string is just an array of chars:

```
char welcome[] = "Hello" ;  // C permits this
```

- This is an array of 8-bit bytes holding ASCII characters
- The array in memory looks like this:

'H'	'e'	'l'	'l'	'o'	'\0'

- Whoa! What's that *last* character????

Strings in C

- A string is just an array of chars:

```
char welcome[] = "Hello" ;  // C permits this
```

- This is an array of 8-bit bytes holding ASCII characters
- The array in memory looks like this:

- Whoa! What's that *last* character????
- In C, proper strings *must* be terminated with a NUL (0) character.
- We *always* need an *extra byte* to hold the terminator!

Strings in C

- Assume we are reading and processing lines of text, where at most the first 80 characters of a line are useful
- How would we declare an array to hold the line as a string?

Strings in C

- Assume we are reading and processing lines of text, where at most the first 80 characters of a line are useful
- How would we declare an array to hold the line as a string?

```
#define MAXLINE (80)
char line[ MAXLINE + 1 ] ;    // 1 extra character for the NUL
```

Strings in C

- Assume we are reading and processing lines of text, where at most the first 80 characters of a line are useful
- How would we declare an array to hold the line as a string?

```
#define MAXLINE (80)
char line[ MAXLINE + 1 ] ;    // 1 extra character for the NUL
```

- How would we read in such a line?

Strings in C

- Assume we are reading and processing lines of text, where at most the first 80 characters of a line are useful
- How would we declare an array to hold the line as a string?

```
#define MAXLINE (80)
char line[ MAXLINE + 1 ] ;    // 1 extra character for the NUL
```

- How would we read in such a line?

```
void readline( char line[], int maxsize ) {
    int i = 0 ;
    int ch ;

    for ( ch = getchar() ; ch != '\n' && ch != EOF ; ch = getchar() ) {
        if ( i < maxsize ) {
            line[ i++ ] = ch ;
        }
    }
    line[ i ] = '\0' ;
    return ;
}
```

Strings in C

- How can we copy one string to another?
- Modify acopy to strcpy:

```
void strcpy( char sto[], char sfrom[] ) {
    int i ;

    for ( i = 0 ; sto[ i ] = sfrom[ i ] ; ++i )
        ;
}
```

Strings in C

- How can we copy one string to another?
- Modify acopy to strcpy:

```
void strcpy( char sto[], char sfrom[] ) {
    int i ;

    for ( i = 0 ; sto[ i ] = sfrom[ i ] ; ++i )
        ;
}
```

Copy the ith character.
If this was a NUL, exit the loop.

String Library

```
#include <string.h>
int strlen( char str[] ) ;
   Note: strlen("Hello") == 5

void strcpy( char sto[], char sfrom[] ) ;
void strncpy( char sto[], char sfrom[], unsigned n );
```

Note: Copies 'n' characters to 'sto' from 'sfrom', padding with '\0' as necessary.

Note: If 'sfrom' is too long to fit in 'sto', then 'sto' will *NOT* be NUL terminated.

```
int strcmp( char str1[], char str2[] ) ;
```

Note: comparison is in dictionary order.

Note: returns -1, 0, 1 if 'str1' is less than, equal to, or greater than 'str2', respectively.

Command Line Arguments

The full declaration of main is:

```
int main( int ac, char **argv ) ;
```

Command Line Arguments

The full declaration of main is:

```
int main( int ac, char **argv ) ;
```

ac = argument count (the number of command line arguments).

ac >= 1, as the program name is the 0th argument.

Command Line Arguments

The full declaration of main is:

```
int main( int ac, char **argv ) ;
```

ac = argument count (the number of command line arguments).

Includes the program name as the 0th argument.

Example: ac == 5

Command Line Arguments

The full declaration of main is:

```
int main( int ac, char **argv ) ;
```

`argv` = the argument vector - allows access to the arguments

it's a pointer, but don't worry - treat it like a 2D array.

`argv[ i ]` is i^{th} argument as a string (array).

`argv[ i ][ j ]` is the j^{th} character of the i^{th} argument.

Example – Echo Arguments

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

int main( int ac, char **argv ) {
    int i ;

    printf( "Program name = %s\n", argv[0] ) ;

    for( i = 1 ; i < ac ; ++i ) {
        printf( "argv[%d] = %s ", i, argv[i] ) ;
        printf( "and its length is %d\n", strlen( argv[i]
  ) ) ;
    }

    return 0 ;
}
```