Array-based C-Strings

How you create a C-string determines where the characters are stored in memory. To copy characters **into user memory** where they can be modified, write this:

```
char s1[] = "String #1";
```

The C-string `s1 contains` **exactly** `10` characters; the `9` that appear in "`String #1`" and the terminating `NUL` character. Space for these characters is **allocated on the stack or static storage area**. The **actual characters are copied** into this "user space". This declaration is shorthand for:

```
 char s1[] = {'S','t','r','i','n','g',' ','#','1','\0'};
```

Because the characters have been copied into memory that you control, you can **change them if you like** using the normal array subscripting operations.

```
s1[0] = 'C';  // OK; all characters are read-write
```

```
const size_t kLen = 1024;     // small strings
char s2[kLen] = "String #2";
```

The declaration for `s2` is slightly different. While the effective size of the string is also `9` characters, its **allocated size** is set by `kLen` or `1024` in this case. Use `s2` if you want to add information to the end of the string, similar to partially-filled arrays.