Adam Donegan
100861846

Strings (String Literals vs Char*)

String Literal

```
char str1[] = "hello";
```

The line of code above does 1 thing; it defines a char array containing 6 characters: ‘`h`’, ‘`e`’, ‘`l`’, ‘`l`’, ‘`o`’ followed by the null terminator ‘`\0`’. The char array `str1` is allocated on the stack, and can be modified freely without error. If declared inside a function, it will live for the length of that function. The compiler will know the size of the array when it initializes it for you, so you can use the `sizeof()` funtion to get its size.

Char*

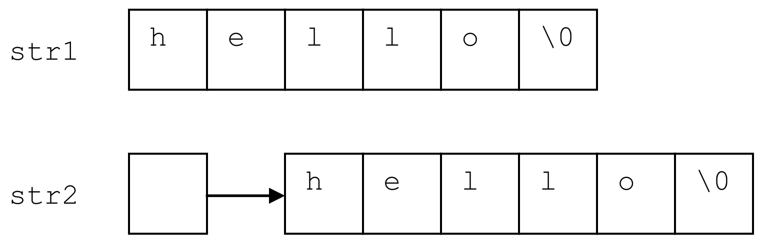
```
char *str2 = "hello";
```

The line of code seen here does two things:

- Defines a string literal containing 6 characters: ‘`h`’, ‘`e`’, ‘`l`’, ‘`l`’, ‘`o`’ followed by the null terminator ‘`\0`’. This literal is allocated in the read-only part of storage (the data segment).
- Defines a char pointer that points to the address of the first character in our read-only string literal from the previous point.

As the string literal that str2 points to is located in the read-only part of storage, it is not recommended that you try to modify its contents, as this can lead to undefined behaviour. Using the `sizeof()` function here doesn't help us, as it will only return the size of a char pointer. Instead, we must use the `strlen()` function to determine the size of our string.

Here is a visual representation of what is happening in both cases:


str1
h
e
l
l
o
\0
str2
h
e
l
l
o
\0