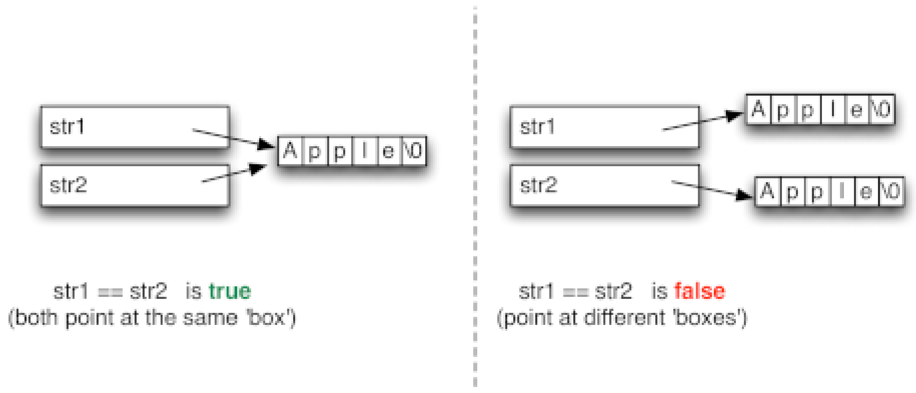
String Equality | Coursera

String Equality

String Equality

Often, programmers want to compare two strings to see if they are equal. Our first inclination might be to use the == operator, which we have already seen. However, the == operator will compare pointer equality. That is, if we write *str1* == *str2*, it will check if *str1* and *str2* are arrows **pointing at the same place**. Sometimes pointer equality is what we mean, but more often, we want to check to see if the two strings have the same sequence of characters, even if they are in different locations in memory. This concept is illustrated in the figure below:



This figure illustrates applying the == operator to two 'string' just compares the pointers for equality.

You can write a function to test if two strings have the same contents—that is, if they contain exactly the same sequence of characters. We will walk through use of The Seven Steps to devise and implement such a function in the next video. This task is common enough that there is already a function to do it—called *strcmp*—in *string.h* of the C library. However, working through the process using The Seven Step sis still a great example of how to write code that manipulates strings.

The C library *strcmp* function behaves slightly differently than the one we are about to write in the video—it returns 0 if the strings are equal, and non-zero if they are different. In fact, it returns a positive number if the first string is "greater than" the second string and a negative number if the first string is "less than" the second string. Here "greater than" and "less than" refer to *lexicographic order* —what you would think of as "alphabetical ordering," but extended to encompass the fact that strings may have non-letters. The comparison is case sensitive (so *abc* is different from *Abc*), but there is another function, *strcasecmp* which performs case-insensitive comparison.