to system):

0x0065fd24 = array address, 32 = sizeof cookies
0x0065fd24 = arr, 4 = sizeof arr
Total cookies eaten: 255
0x0065fd24 = arr, 4 = sizeof arr
First three eaters ate 7 cookies.
0x0065fd34 = arr, 4 = sizeof arr
Last four eaters ate 240 cookies.

Program Notes

Listing 7.6 illustrates some very interesting points about array functions. First, note that cookies and arr both evaluate to the same address, exactly as claimed. But sizeof cookies is 16, whereas sizeof arr is only 4. That's because sizeof cookies is the size of the whole array, whereas sizeof arr is the size of the pointer variable. (This program execution takes place on a system using 4-byte addresses.) By the way, that's why you have to pass explicitly the size of the array rather than use sizeof arr in sum_arr().

Because the only way sum_arr() knows the number of elements in the array is through what you tell it with the second argument, you can lie to the function. For example, the second time the program uses the function, it makes this call:

sum = sum_arr(cookies, 3);

By telling the function that cookies has but three elements, you get the function to calculate the sum of the first three elements.

Why stop there? You also can lie about where the array starts:

sum = sum_arr(cookies + 4, 4);

Because cookies acts as the address of the first element, cookies + 4 acts as the address of the fifth element. This statement sums the fifth, sixth, seventh, and eighth elements of the array. Note in the output how the third call to the function assigns a different address to arr than the first two calls did. And yes, you can use &cookies[4] instead of cookies + 4 as the argument; both mean the same thing.