Logical Operators Shortcut Evaluation in C

In **C**,
- **logical AND and OR**
- **stop evaluating operands**
- **when the operator's result is known**.

For example,

```
0 && this_function_crashes ()
```

does NOT call the function.

The **first operand is false** (0 in **C**), so the **second operand** (the function call) is **not evaluated**.

Logical AND Stops on False, Logical OR Stops on True

Similarly, if we write

```
1 || this_function_crashes ()
```

does NOT call the function.

The **first operand is true** (not 0 in **C**), so the **second operand** (the function call) is **not evaluated**.

Use Shortcutting to Protect Unsafe/Undesired Actions

Here's a more realistic example…

```
if (1 == scanf ("%d", &age) &&
    0 <= printf ("Salary? ") &&
    1 == scanf ("%d", &salary)) {
    // use age and salary
}
```

scanf in these cases returns 1 on success, and **printf** returns 8 (characters) on success.

Use Shortcutting to Protect Unsafe/Undesired Actions

And another one…

```
if (0 <= dist_sq &&
    walk_p (me, sqrt (dist_sq))) {
    // go for a walk
}
```

Calculating the square root (**sqrt**) of a negative number may cause a crash.

