

Let's break down the loop example in a more detailed and step-by-step manner to understand how `ptr` progresses through a linked list:

Understanding a Linked List

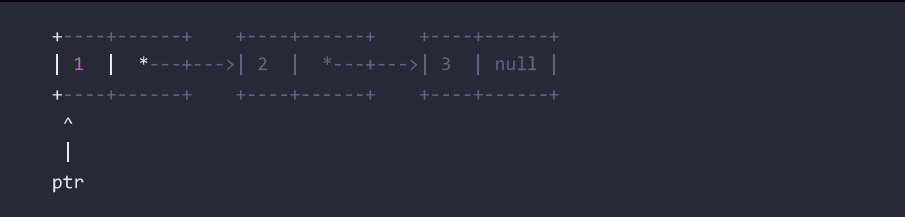
First, imagine a linked list as a series of connected nodes. Each node has two parts:

- 1 **Data:** The value or information stored in the node.
- 2 **Next Pointer:** A reference (pointer) to the next node in the list.

The last node's next pointer is `null`, indicating the end of the list.

Visual Representation

Consider a simple linked list with three nodes for illustration:



Here, `ptr` initially points to the first node (with data `1`). Each node points to the next node, and the last node points to `null`.

The Loop in Detail

Let's walk through the loop step by step:

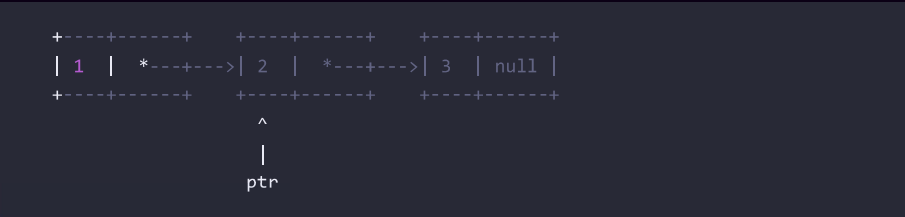
Initial State

- `ptr` points to the first node in the list.

First Iteration

- The loop checks if `ptr` is not `null`. It's true because `ptr` points to the first node.
- Inside the loop, `ptr` is set to `ptr->next`. So, `ptr` now points to the second node.

Updated state:



Second Iteration

- Again, check if `ptr` is not `null`. It's true because `ptr` points to the second node.
- Update `ptr` to `ptr->next`. Now, `ptr` points to the third node.

Updated state:



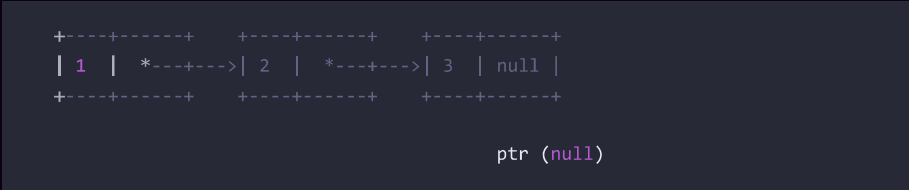
Third Iteration

- Check if `ptr` is not `null`. It's true because `ptr` points to the third node.
- Update `ptr` to `ptr->next`. Since the next of the third node is `null`, `ptr` now becomes `null`.

Termination

- The loop checks if `ptr` is not `null`. This time it's false, because `ptr` is `null`.
- The loop exits.

Final state:



`ptr` has moved through each node in the list and stops when it reaches `null`, indicating the end of the list.