Linked-List using C Programming Part 1

Learning step 1 : Create, List, Nodes

This was a little bit complicated concept for me, first we need 2 structures Linkedlist to hold the nodes and node itself.
We create a node when we need to append it to the list.
List is created only once

```
/*
 *Initialize structure for a single node and a list
 *Node containes data and next variable pointer that holds address to Node* structure
 *LinkedList only needs address for head of the node created
  */
typedef struct Node{
        int data;
        struct Node* next;
} Node;

typedef struct {
        Node* head;
} LinkedList;

/*
 *Createlist function that makes a list and allocated memory for future nodes
 *as list is made initially, make the head point to null as there are no nodes
 */

LinkedList* createList(){
        LinkedList* list = (LinkedList*)malloc(sizeof(LinkedList));
        list->head = NULL;
        return list;
}

/*Create node with value as parameter
 *update data to value, and point next pointer to null as it is a new node
 */

Node* createNode(int val){
        Node* nnode = (Node*)malloc(sizeof(Node));
        nnode->data = val;
        nnode->next = NULL;
        return nnode;
}
```

Comments should give you better idea about the structure.

Learning step 1 : Append, Traverse list

First user creates a list and the list head points to null. Append checks if list is empty, and then changes list head from pointing to null to new node. If list is not empty just simply parse through the list till you find a node that points to null which is basically the last node.

Traverse similarly parses through the list till there is no node while printing the node values.

```c
/*Append takes list and value for a node as input
 *creates node and check if it is empty, if yes make the new node head
 *if not use temp node to parse till next pointer for any node is null, basically add node to the last position
 */

void append(LinkedList* list, int val){
        Node* nnode = createNode(val);
        if(list->head == NULL){
                list->head = nnode;
                return;
        }

        Node* tnode = list->head;

        while(tnode->next != NULL){

                tnode = tnode->next;
        }
        tnode->next = nnode;
}

/*
 *Lastly traverse use temp node to parse through all the nodes till temp node is null
 *Print node data while you iterate
 */
void traverseNodes(LinkedList* list)
{
        Node* tnode = list->head;
        while(tnode != NULL){
        printf("%d - > ", tnode->data);
        tnode = tnode->next;}
        printf("NULL\n");
}
```

Call the function and execute.

```c
int main()
{
        LinkedList* list = createList();

        append(list, 10);
        append(list, 20);
        append(list, 30);

        traverseNodes(list);

        return 0;
}
```

```
phenodiss1de@Phen0:~/ccode/DSA-DAY1/C-Project-based-Learning/DSA/LinkedList/Day1$ vim linked_list.c
phenodiss1de@Phen0:~/ccode/DSA-DAY1/C-Project-based-Learning/DSA/LinkedList/Day1$ gcc -g linked_list.c -o append
phenodiss1de@Phen0:~/ccode/DSA-DAY1/C-Project-based-Learning/DSA/LinkedList/Day1$ ./append
10 - > 20 - > 30 - > NULL
```