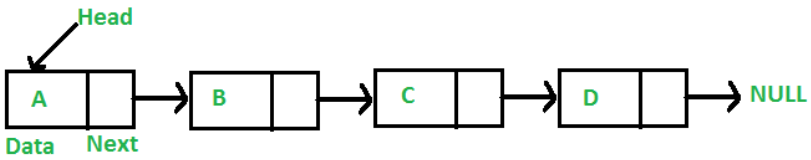
Linked List | Set 1 (Introduction)

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.



Why Linked List?

Arrays can be used to store linear data of similar types, but arrays have following limitations.

1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.

2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.

For example, in a system if we maintain a sorted list of IDs in an array id[].

id[] = [1000, 1010, 1050, 2000, 2040].

And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).

Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

Advantages over arrays

1) Dynamic size

2) Ease of insertion/deletion

Drawbacks:

1) Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists.

2) Extra memory space for a pointer is required with each element of the list.

3) Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

Representationv:

A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.

Each node in a list consists of at least two parts:

1) data

2) Pointer (Or Reference) to the next node

In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.

In Java, LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.

C

```c
// A linked list node
struct Node
{
  int data;
  struct Node *next;
};
```

Run on IDE

Java

```java
class LinkedList
{
    Node head;  // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;

        // Constructor to create a new node
        // Next is by default initialized
        // as null
        Node(int d) {data = d;}
    }
}
```

Run on IDE

Python

```python
# Node class
class Node:

    # Function to initialize the node object
    def __init__(self, data):
        self.data = data  # Assign data
        self.next = None  # Initialize
                          # next as null

# Linked List class
class LinkedList:

    # Function to initialize the Linked
    # List object
    def __init__(self):
        self.head = None
```

Run on IDE

First Simple Linked List in C Let us create a simple linked list with 3 nodes.

C

```
// A simple C program to introduce
// a linked list
#include<stdio.h>
#include<stdlib.h>

struct Node
{
  int data;
  struct Node *next;
};

// Program to create a simple linked
// list with 3 nodes
int main()
{
  struct Node* head = NULL;
  struct Node* second = NULL;
  struct Node* third = NULL;

  // allocate 3 nodes in the heap
  head = (struct Node*)malloc(sizeof(struct Node));
  second = (struct Node*)malloc(sizeof(struct Node));
  third = (struct Node*)malloc(sizeof(struct Node));

  /* Three blocks have been allocated  dynamically.
     We have pointers to these three blocks as first,
     second and third
       head           second           third
        |               |               |
        |               |               |
    +---+-----+     +----+----+     +----+----+
    | #  | #  |     | #  | #  |     |  # |  # |
    +---+-----+     +----+----+     +----+----+

   # represents any random value.
   Data is random because we haven't assigned
   anything yet  */

  head->data = 1; //assign data in first node
  head->next = second; // Link first node with
                       // the second node

  /* data has been assigned to data part of first
     block (block pointed by head).  And next
     pointer of first block points to second.
     So they both are linked.

       head          second         third
        |              |              |
        |              |              |
    +---+---+     +----+----+     +-----+----+
    | 1  | o----->| #  | #  |     |  #  | #  |
    +---+---+     +----+----+     +-----+----+
  */

  // assign data to second node
  second->data = 2;

  // Link second node with the third node
  second->next = third;

  /* data has been assigned to data part of second
     block (block pointed by second). And next
     pointer of the second block points to third
     block. So all three blocks are linked.

       head         second         third
```

```
        |            |            |
        |            |            |
    +---+---+     +---+---+     +----+----+
    | 1   | o-----> | 2 | o-----> |  #  |  # |
    +---+---+     +---+---+     +----+----+      */

  third->data = 3; //assign data to third node
  third->next = NULL;

  /* data has been assigned to data part of third
    block (block pointed by third). And next pointer
    of the third block is made NULL to indicate
    that the linked list is terminated here.

     We have the linked list ready.

           head
             |
             |
        +---+---+     +---+---+       +----+------+
        | 1   | o-----> |  2  | o-----> |  3 | NULL |
        +---+---+     +---+---+       +----+------+


    Note that only head is sufficient to represent
    the whole list.  We can traverse the complete
    list by following next pointers.    */

  return 0;
}
```

Run on IDE

Java

```
// A simple Java program to introduce a linked list
class LinkedList
{
    Node head;  // head of list

    /* Linked list Node.  This inner class is made static so that
       main() can access it */
    static class Node {
        int data;
        Node next;
        Node(int d)  { data = d;  next=null; } // Constructor
    }

    /* method to create a simple linked list with 3 nodes*/
    public static void main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head  = new Node(1);
        Node second = new Node(2);
        Node third  = new Node(3);

        /* Three nodes have been allocated  dynamically.
          We have refernces to these three blocks as first,
          second and third

          llist.head        second              third
             |                |                   |
             |                |                   |
         +----+------+     +----+------+     +----+------+
         | 1  | null |     | 2  | null |     |  3 | null |
         +----+------+     +----+------+     +----+------+ */

        llist.head.next = second; // Link first node with the second node

        /*  Now next of first Node refers to second.  So they
            both are linked.

         llist.head        second              third
            |                |                   |
```

```
            |                |                  |
        +----+------+     +----+------+     +----+------+
        | 1  |  o--------->| 2  | null |     |  3 | null |
        +----+------+     +----+------+     +----+------+ */

        second.next = third; // Link second node with the third node

        /*  Now next of second Node refers to third.  So all three
            nodes are linked.

         llist.head        second              third
            |                |                  |
            |                |                  |
        +----+------+     +----+------+     +----+------+
        | 1  |  o--------->| 2  |  o--------->|  3 | null |
        +----+------+     +----+------+     +----+------+ */
    }
}
```

Run on IDE

Python

```
# A simple Python program to introduce a linked list

# Node class
class Node:

    # Function to initialise the node object
    def __init__(self, data):
        self.data = data  # Assign data
        self.next = None  # Initialize next as null


# Linked List class contains a Node object
class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None


# Code execution starts here
if __name__=='__main__':

    # Start with the empty list
    llist = LinkedList()

    llist.head  = Node(1)
    second = Node(2)
    third  = Node(3)

    '''
    Three nodes have been created.
    We have references to these three blocks as first,
    second and third

    llist.head        second              third
         |                |                  |
         |                |                  |
    +----+------+     +----+------+     +----+------+
    | 1  | None |     | 2  | None |     |  3 | None |
    +----+------+     +----+------+     +----+------+
    '''

    llist.head.next = second; # Link first node with second

    '''
    Now next of first Node refers to second.  So they
    both are linked.

    llist.head        second              third
         |                |                  |
         |                |                  |
    +----+------+     +----+------+     +----+------+
    | 1  |  o--------->| 2  | null |     |  3 | null |
```

```
    +----+------+     +----+------+     +----+------+
    '''

    second.next = third; # Link second node with the third node

    '''
    Now next of second Node refers to third.  So all three
    nodes are linked.

    llist.head        second              third
         |              |                   |
         |              |                   |
    +----+------+     +----+------+     +----+------+
    | 1  |  o-------->| 2  |  o-------->|  3 | null |
    +----+------+     +----+------+     +----+------+
    '''
```

Run on IDE

Linked List Traversal

In the previous program, we have created a simple linked list with three nodes. Let us traverse the created list and print the data of each node. For traversal, let us write a general purpose function printList() that prints any given list.

We strongly recommend that you click here and practice it, before moving on to the solution.

C

```c
// A simple C program for traversal of a linked list
#include<stdio.h>
#include<stdlib.h>

struct Node
{
  int data;
  struct Node *next;
};

// This function prints contents of linked list starting from
// the given node
void printList(struct Node *n)
{
  while (n != NULL)
  {
     printf(" %d ", n->data);
     n = n->next;
  }
}

int main()
{
  struct Node* head = NULL;
  struct Node* second = NULL;
  struct Node* third = NULL;

  // allocate 3 nodes in the heap
  head  = (struct Node*)malloc(sizeof(struct Node));
  second = (struct Node*)malloc(sizeof(struct Node));
  third  = (struct Node*)malloc(sizeof(struct Node));

  head->data = 1; //assign data in first node
  head->next = second; // Link first node with second

  second->data = 2; //assign data to second node
  second->next = third;

  third->data = 3; //assign data to third node
  third->next = NULL;
```

```
  printList(head);

  return 0;
}
```

Run on IDE

Java

```
// A simple Java program for traversal of a linked list
class LinkedList
{
    Node head;  // head of list

    /* Linked list Node.  This inner class is made static so that
       main() can access it */
    static class Node {
        int data;
        Node next;
        Node(int d)  { data = d;  next=null; } // Constructor
    }

    /* This function prints contents of linked list starting from head */
    public void printList()
    {
        Node n = head;
        while (n != null)
        {
            System.out.print(n.data+" ");
            n = n.next;
        }
    }

    /* method to create a simple linked list with 3 nodes*/
    public static void main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head       = new Node(1);
        Node second      = new Node(2);
        Node third       = new Node(3);

        llist.head.next = second; // Link first node with the second node
        second.next = third; // Link first node with the second node

        llist.printList();
    }
}
```

Run on IDE

Python

```
# A simple Python program for traversal of a linked list

# Node class
class Node:

    # Function to initialise the node object
    def __init__(self, data):
        self.data = data  # Assign data
        self.next = None  # Initialize next as null


# Linked List class contains a Node object
class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None
```

```
    # This function prints contents of linked list
    # starting from head
    def printList(self):
        temp = self.head
        while (temp):
            print temp.data,
            temp = temp.next


# Code execution starts here
if __name__=='__main__':

    # Start with the empty list
    llist = LinkedList()

    llist.head  = Node(1)
    second = Node(2)
    third  = Node(3)

    llist.head.next = second; # Link first node with second
    second.next = third; # Link second node with the third node

    llist.printList()
```

Run on IDE

Output:

```
1  2  3
```

Important Links :

- Practice MCQ Questions on Linked List
- Linked List Data Structure Page
- Coding Practice Questions on Linked List.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.