養天地正氣 法古今完人

指针与链式结构

溢出问题

- 顺序存储结构将所有数据存储在数组中，数组的容量固定，需要在编写程序时确定，在程序运行中不能更改（常量）——写程序时需要确定数组容量的上限

- 数组容量确定不当，程序运行过程中空间不足，就会出现overflow

指针

- 能有效解决上述溢出问题

- 指针有时称为链域或引用，
- 它是一个对象，通常是一个变量。它存储其它对象，通常是含有我们希望处理的数据的有结构对象的地址。
- 如果我们用指针来定位我们关心的所有数据，就不用考虑这些数据到底放在那里，因为使用指针可以使计算机系统自己定位我们要找的数据。

有关图的约定

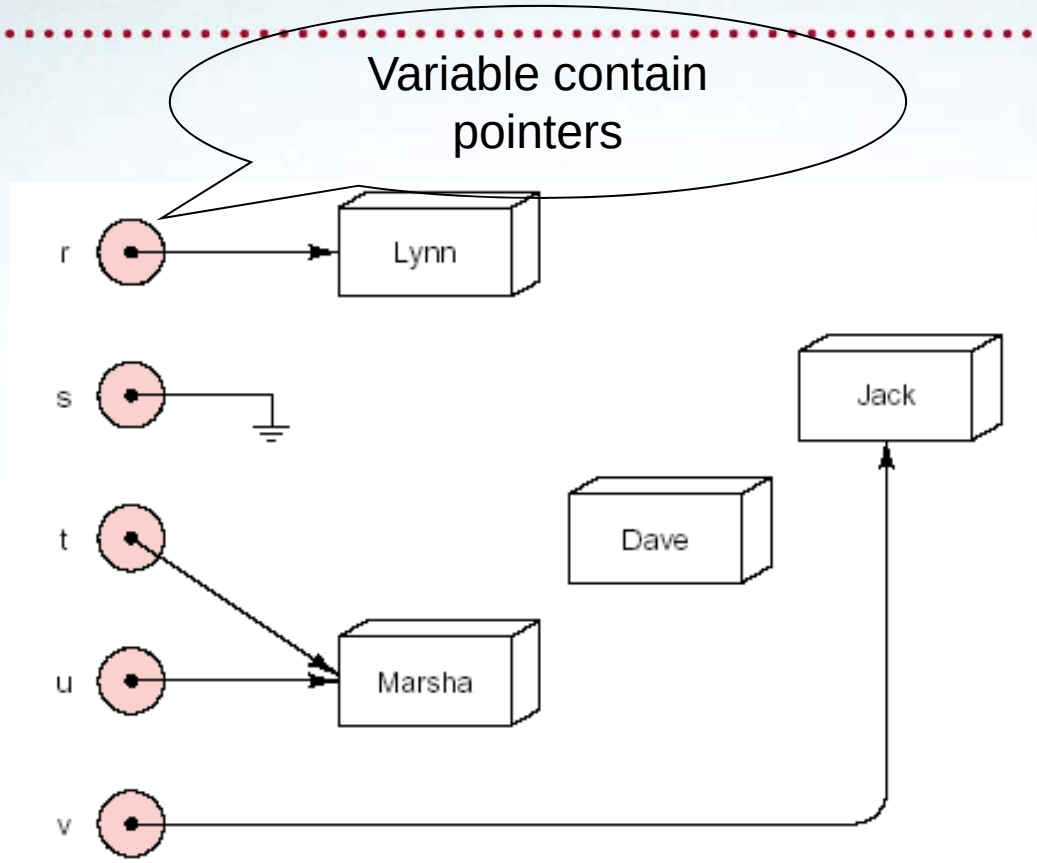


Figure 4.1. Pointers to objects

链式结构

链表的基本思想是借助指针指示线性表中下一个元素的位置

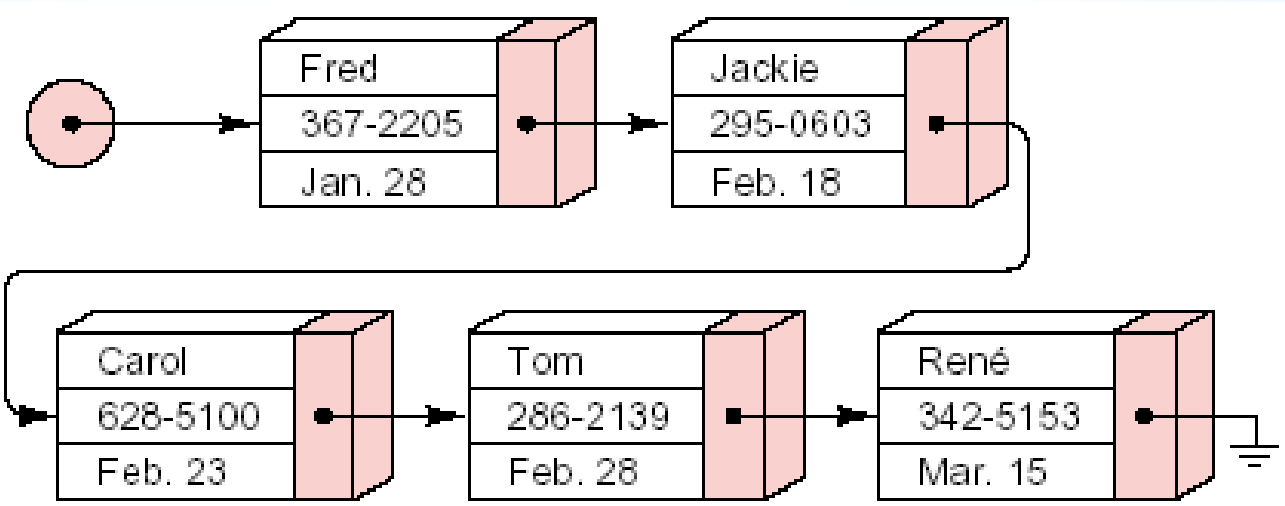


Figure 4.2. A linked list

动态内存分配

- 动态分配内存

  优势在于：开始时占用足够小的内存，运行中按需要分配内存，运行更有效，而且内存的限制就是计算机系统存储空间的大小

- 自动对象

  程序中明确命名，存储空间由系统按生存周期分配

- 动态对象

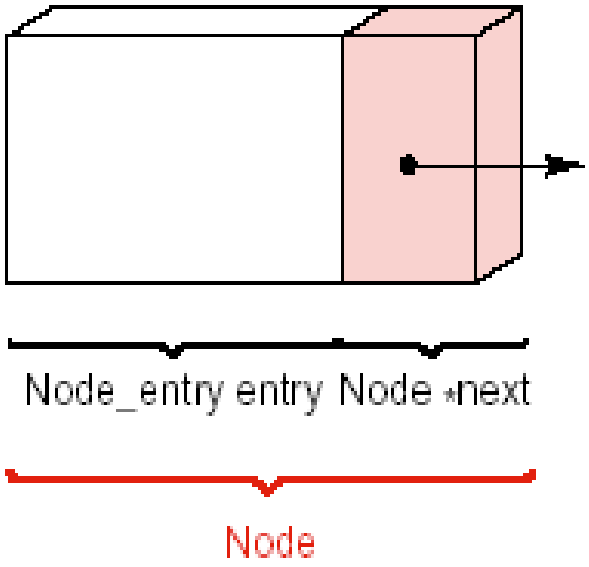
  程序执行过程中动态生成和销毁的，由程序员控制。

基本链式结构

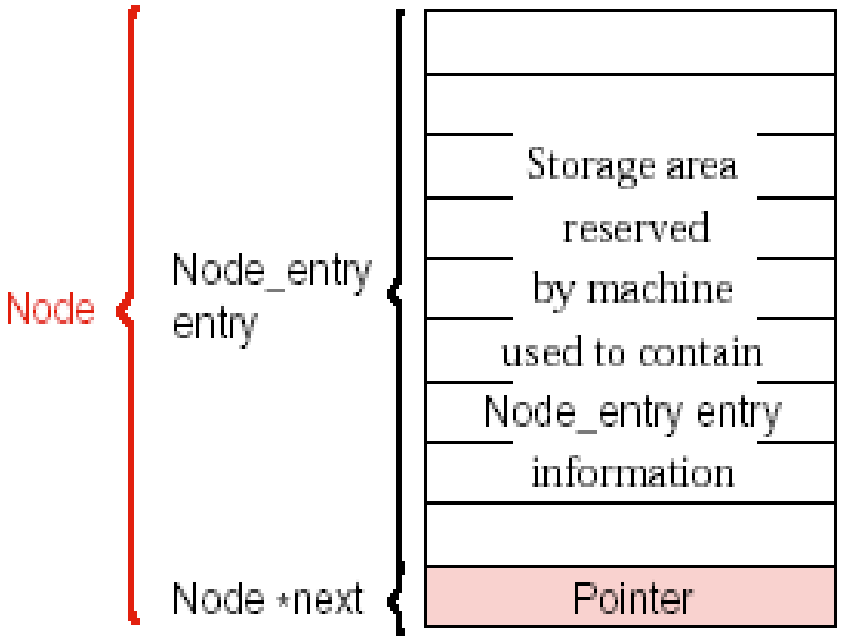
- Nodes and Type Declarations

```
struct Node {
// data members
Node_entry entry;
Node *next; //use before definition
// constructors
Node( );
Node(Node_entry item, Node *add_on = NULL);
};
```

基本链式结构



(a) Structure of a Node



(b) Machine storage representation of a Node

Figure 4.7. Structures containing pointers

基本链式结构

- Node Constructors

```
Node :: Node( )
{
next = NULL;
}
```

The second form accepts two parameters for initializing the data members.

```
Node :: Node(Node_entry item, Node *add_on)
{
entry = item;
next = add_on;
}
```

基本链式结构

```
Node first_node('a'); // Node first_node stores data 'a'.
Node *p0 = &first_node; // p0 points to first_Node.
Node *p1 = new Node('b'); // A second node storing 'b' is created.
p0->next = p1; // The second Node is linked after first_node.
Node *p2 = new Node('c', p0); // A third Node storing 'c' is created.
    // The third Node links back to the first node, *p0.
p1->next = p2; // The third Node is linked after the second Node.
```

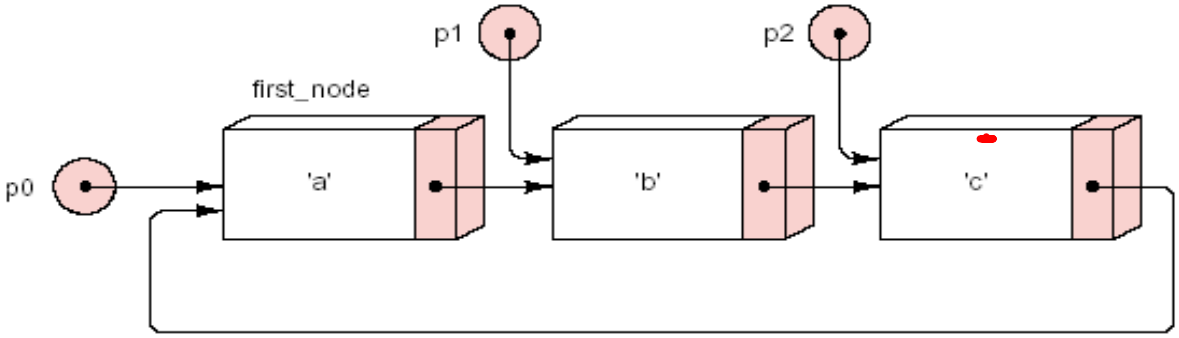


Figure 4.8. Linking nodes