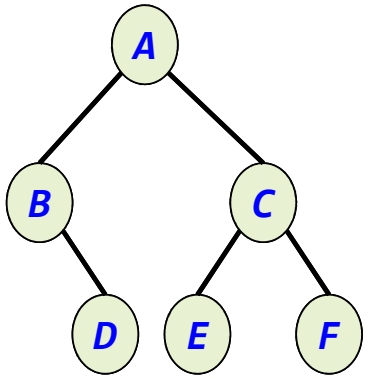
6.3 线索二叉树

6.3.1 什么是线索

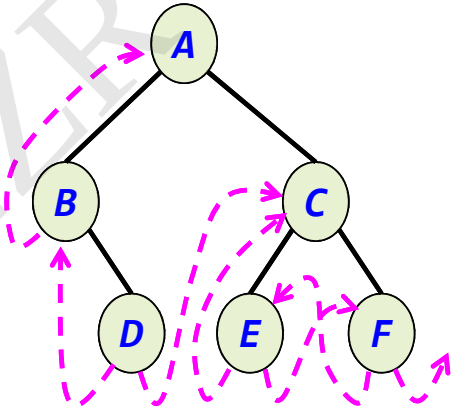
- 对于*n*个结点的二叉树，在二叉链存储结构中有*n*+1个空链域。
- 利用这些空链域存放在某种遍历次序（线性序列）下该结点的前驱结点和后继结点的指针，这些指针称为线索，加上线索的二叉树称为线索二叉树。
- 线索二叉树分为先序、中序和后序线索二叉树。

图中虚线为线索。



二叉树

先序线索二叉树



先序序列：***ABDCEF***

先序序列：***ABDCEF***

6.3.2 线索二叉树的存储结构

在原二叉链表中增加了ltag和rtag两个标志域。

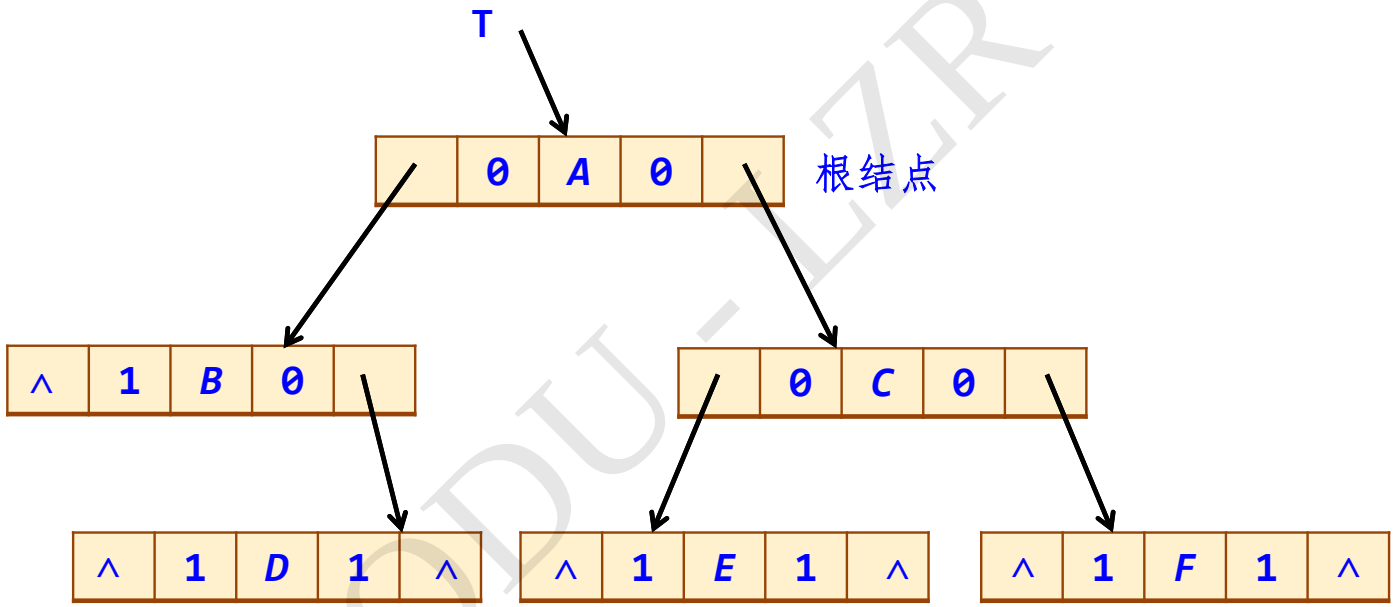
ltag= { 0 表示lchild指向结点的左孩子
ltag= { 1 表示lchild指向结点的前驱结点即为线索

rtag= { 0 表示rchild指向结点的左孩子
rtag= { 1 表示rchild指向结点的后继结点即为线索

线索二叉树的类型定义如下：

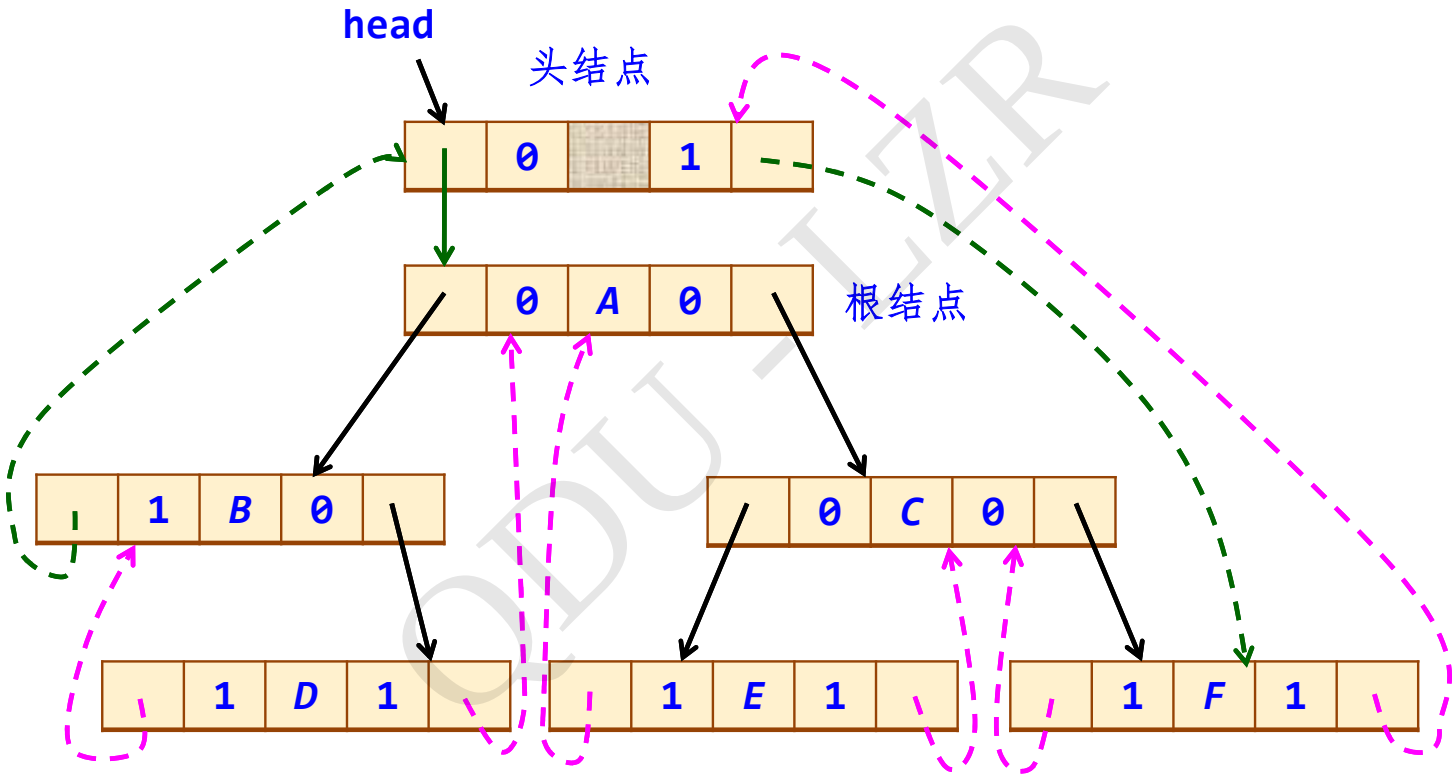
```
// ------ 二叉树的二叉线索存储表示  --------
enum PointerTag {Link, Thread}; // Link(0)：指针，Thread(1)：线索
struct BiThrNode {
    TElemType data;
    BiThrNode *lchild, *rchild;      // 左右孩子指针
    PointerTag LTag, RTag;           // 左右标志
};
typedef BiThrNode *BiThrTree;
```

下面以**中序线索二叉树**为例，讨论线索二叉树的建立和相关算法。为了方便算法实现，为线索二叉树增加一个头结点。



中序序列：**B D A E C F**

下面以**中序线索二叉树**为例，讨论线索二叉树的建立和相关算法。为了方便算法实现，为线索二叉树增加一个头结点。



中序序列：*B D A E C F*

创建线索二叉树算法

- 建立线索化二叉树称之为二叉树线索化。
- 以中序线索化一棵二叉树为例，实质上就是中序遍历一棵二叉树，在遍历过程中，检查当前结点的左、右指针域是否为空；如果为空，将它们改为指向前驱结点或后继结点的线索。

- 算法思想：先创建一个头结点head，在进行中序遍历过程中需保留当前结点p的前驱结点的指针，设为pre（全局变量，初值时指向头结点）。
- 在p不空的情况下：

✓ 遍历左子树（即左子树线索化）。

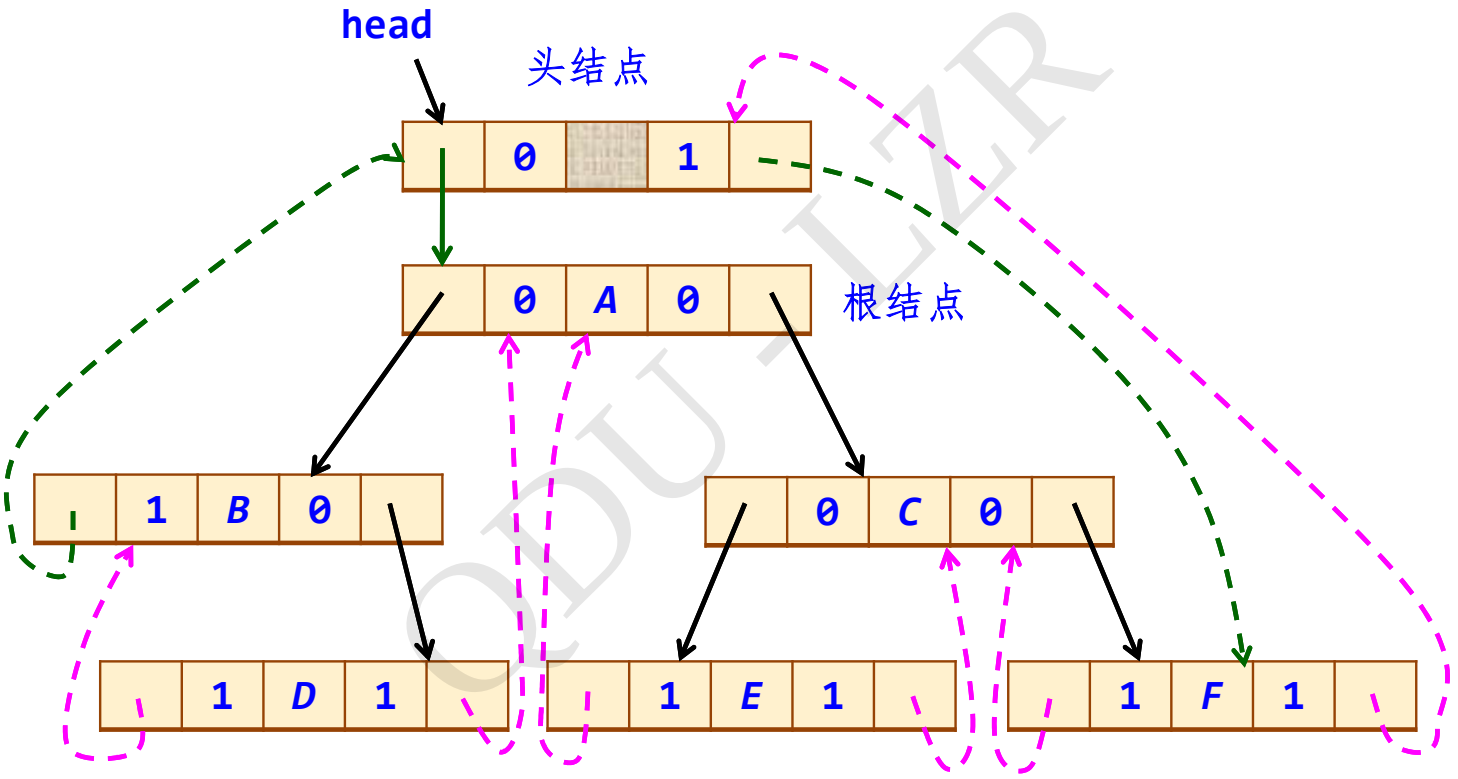
✓ 对空指针线索化。

若p->lchild为空，则置p->ltag=1，且p->lchild=pre；

若p->rchild为空，则置pre->rtag=1，且pre->rchild=p；pre=p；

✓ 遍历右子树（即右子树线索化）。

中序线索二叉树过程：



中序序列：*B D A E C F*

```
BiThrNode *pre;                    //定义pre为全局变量
void Thread(BiThrNode *&p)
//对以p为根结点的二叉树进行中序线索化
{
    if(p != NULL) {
        Thread(p->lchild);                  //左子树线索化
        if(p->lchild == NULL) {             //前驱线索
            p->lchild = pre;                //给结点p添加前驱线索
              p->ltag = 1;
        }
        else
            p->ltag = 0;
        if(pre->rchild == NULL) {
            pre->rchild = p;                //给结点pre添加后继线索
              pre->rtag = 1;
        }
        else
            pre->rtag = 0;
        pre = p;
        Thread(p->rchild);                  //右子树线索化
    }
}
```

```
BiThrNode *CreaThread(BiThrNode *bt)
//对以bt为根结点的二叉树中序线索化,并增加一个头结点head
{
    BiThrNode *head;
    head = (BiThrNode *)malloc(sizeof(BiThrNode));
    head->ltag = 0;
    head->rtag = 1; //创建头结点head
    head->rchild = bt;
    if(bt == NULL)                  //bt为空树时
        head->lchild = head;
    else {
        head->lchild = bt;
        pre = head;                 //pre是p的前驱结点,供加线索用
        Thread(bt);                 //中序遍历线索化二叉树
        pre->rchild = head;         //最后处理,加入指向根结点的线索
        pre->rtag = 1;
        head->rchild = pre;         //根结点右线索化
    }
    return head;
}
```

END