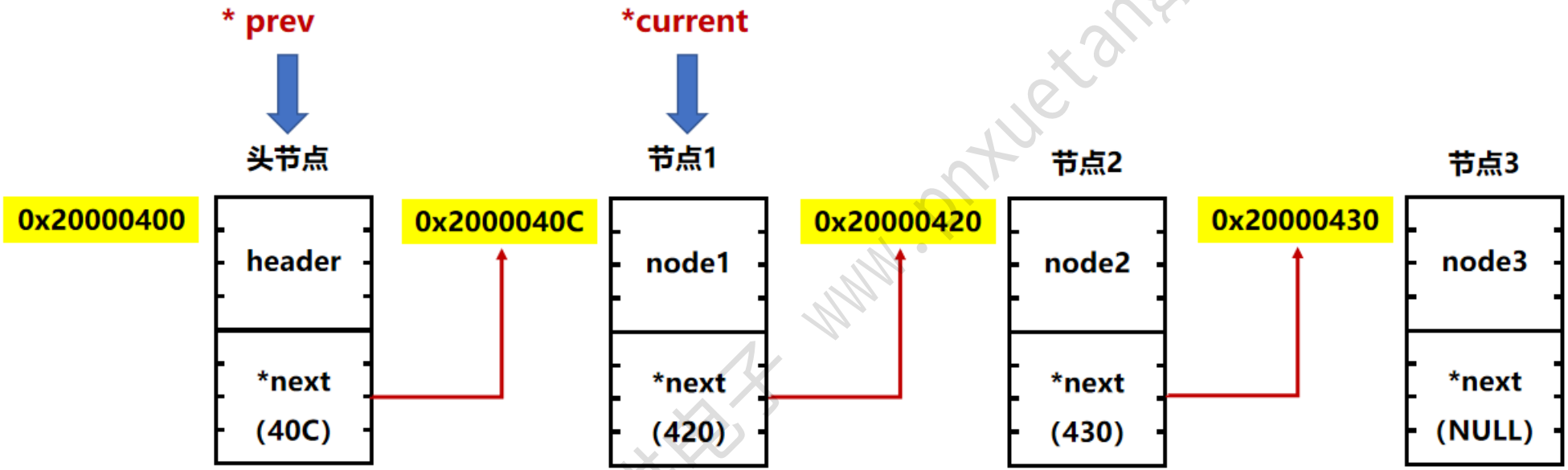
嵌入式C语言之-

双向循环链表

讲师：叶大鹏

助力你成为优秀的电子工程师！

单向链表

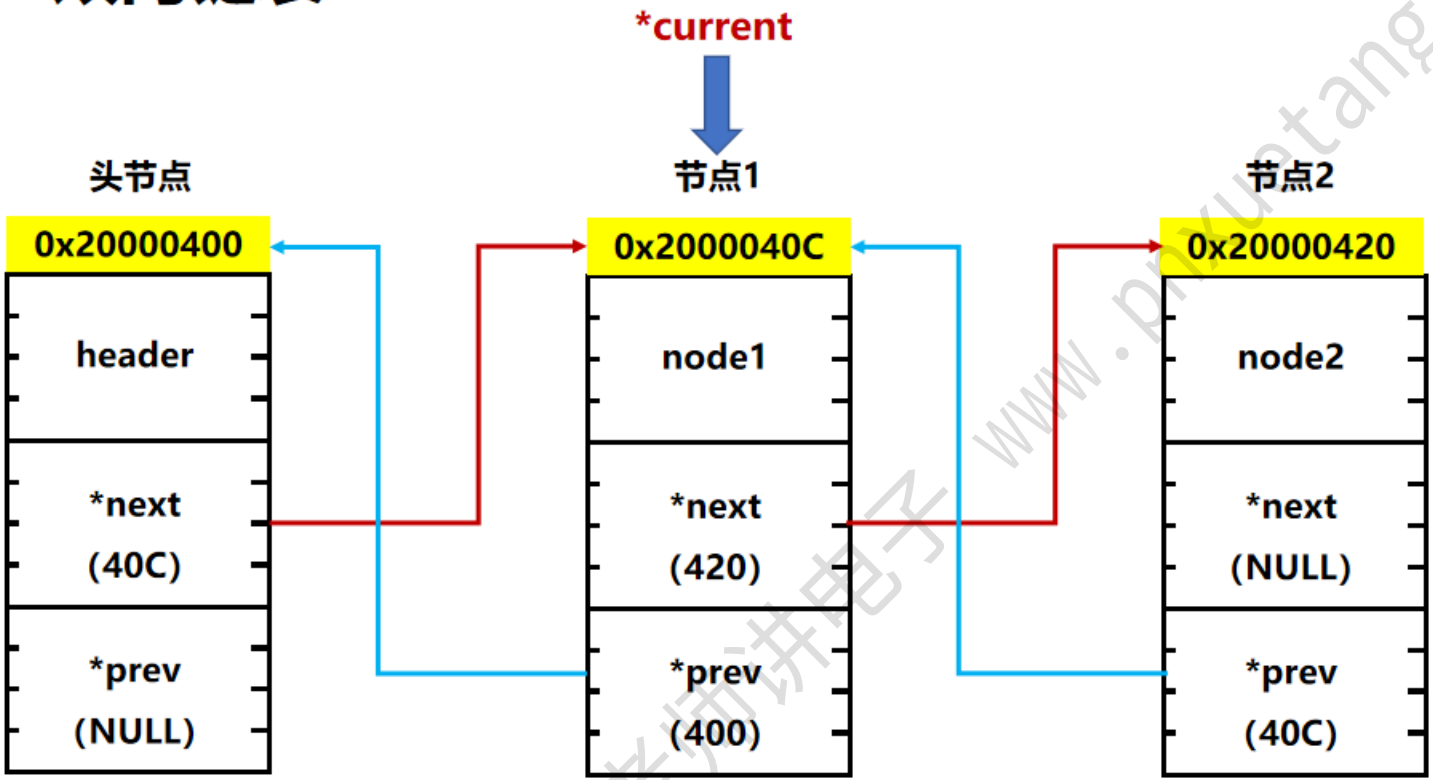


- 1. 使用两个指针变量prev和current，prev一直指向current的前向节点，初始化为：

  TempHumiListNode *prev = g_header;

  TempHumiListNode *current = g_header->next;

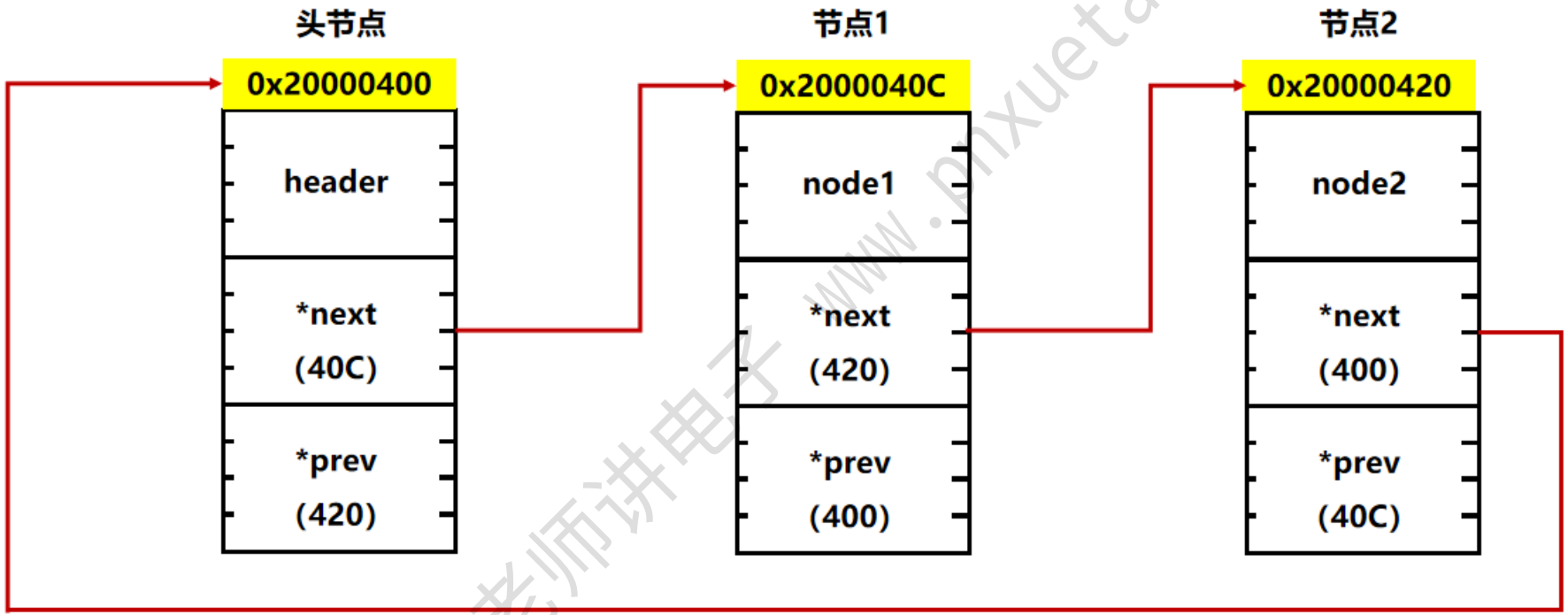
双向链表



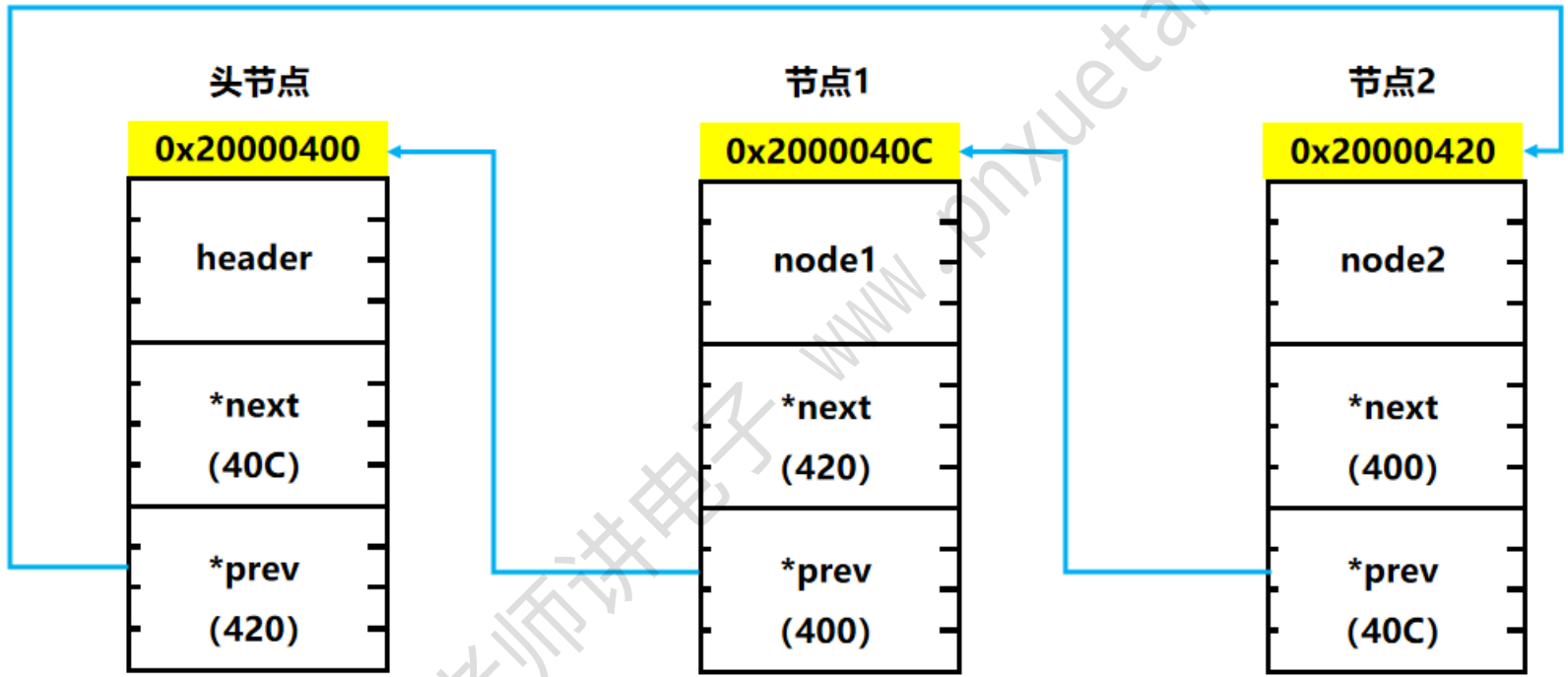

```
typedef struct TempHumiListNode
{
  uint32_t id;
  float temp;
  uint8_t humi;
  struct TempHumiListNode *next;
  struct TempHumiListNode *prev;
} TempHumiListNode;
```

- 双向链表，通过一个指针变量，就可以找到前向和后向节点，方便写程序。

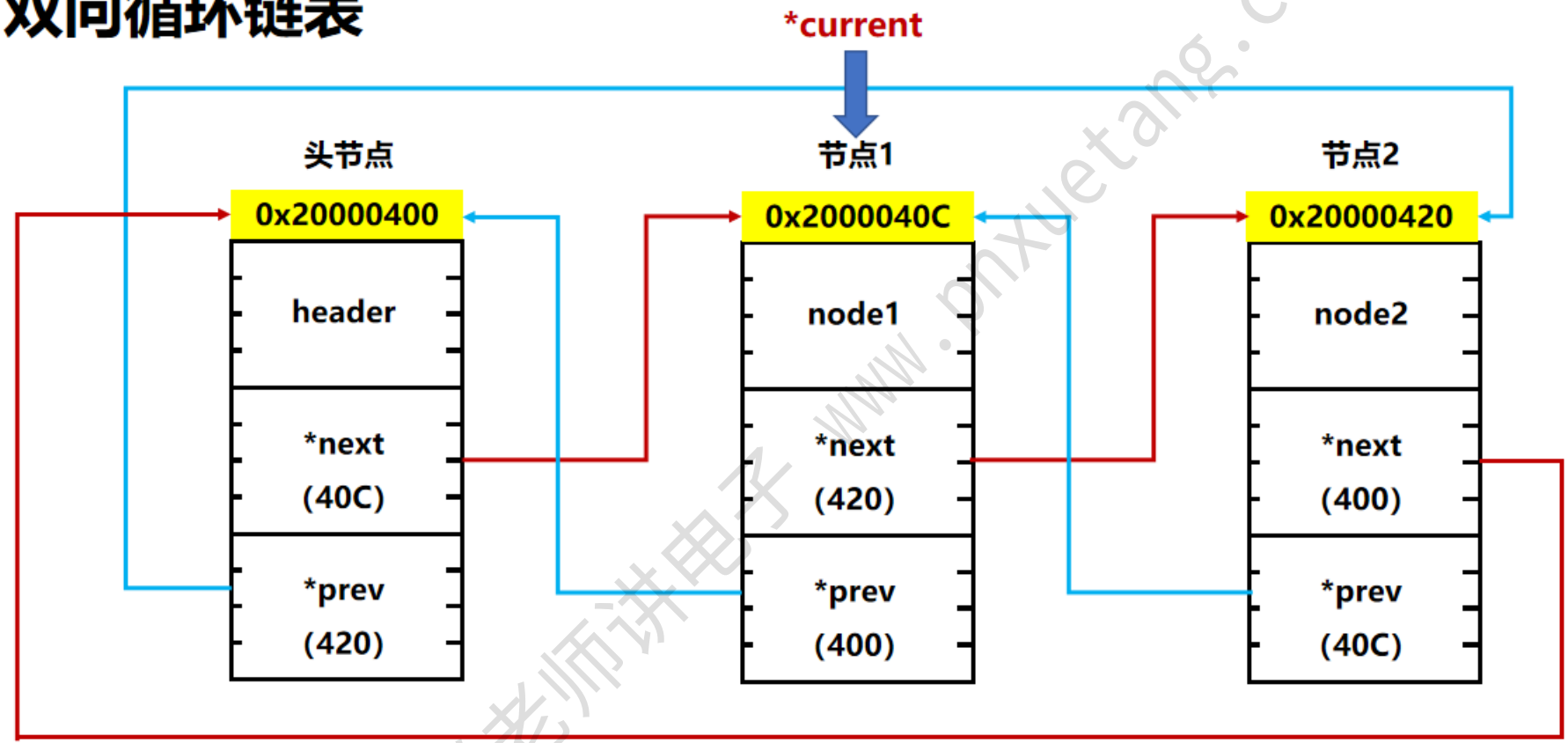
双向循环链表



双向循环链表

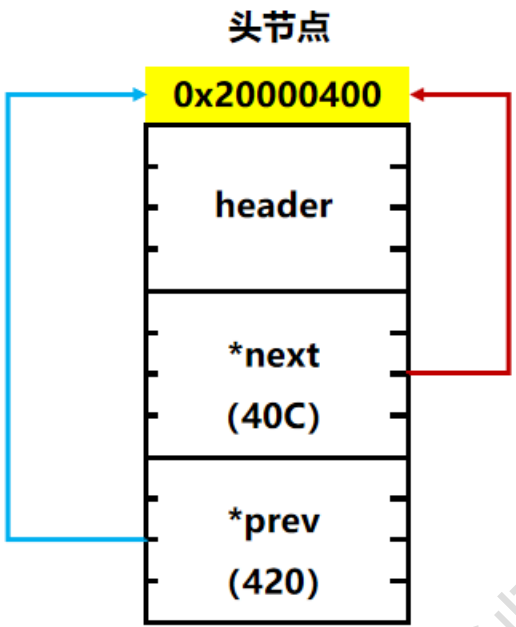


双向循环链表



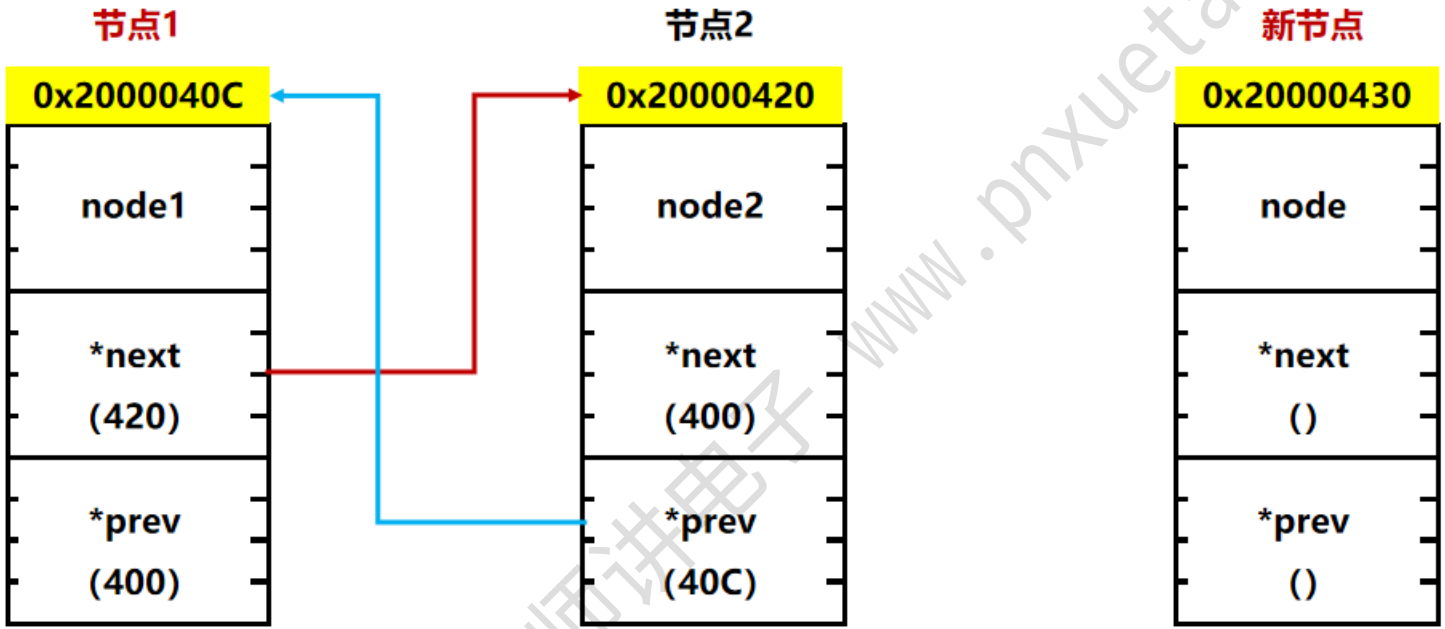
- 双向循环链表，通过一个指针变量，就可以找到前向和后向节点，方便写程序；而且可以从任意节点开始遍历整个链表。

链表初始化

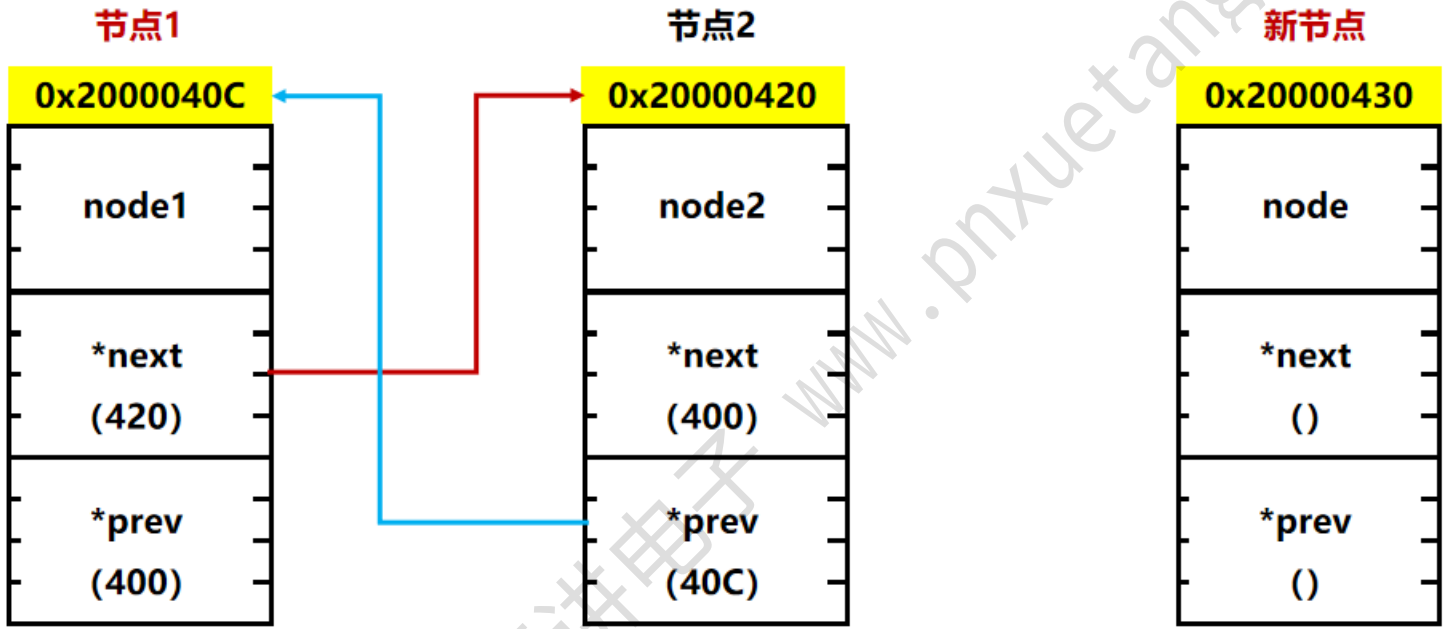



```
header->next = header;
header->prev = header;
```

在任意节点后面添加节点



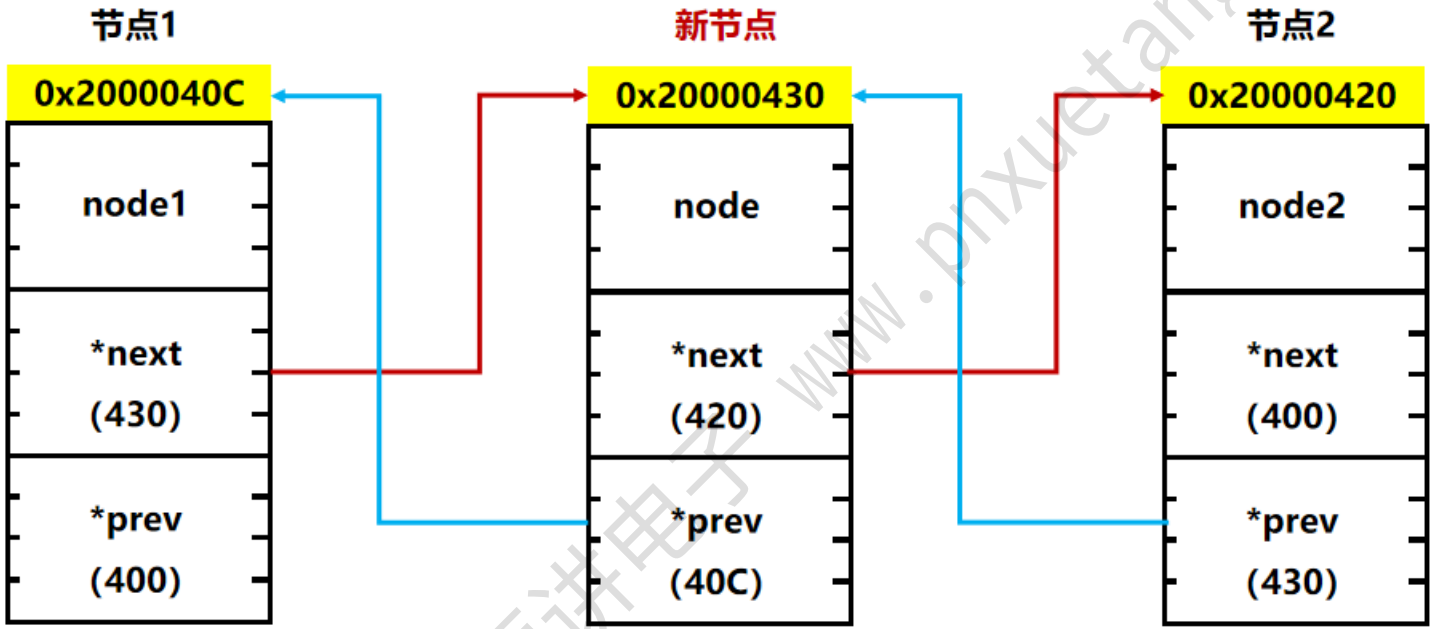
在任意节点后面添加节点



- 假如要在节点1后面添加新节点：

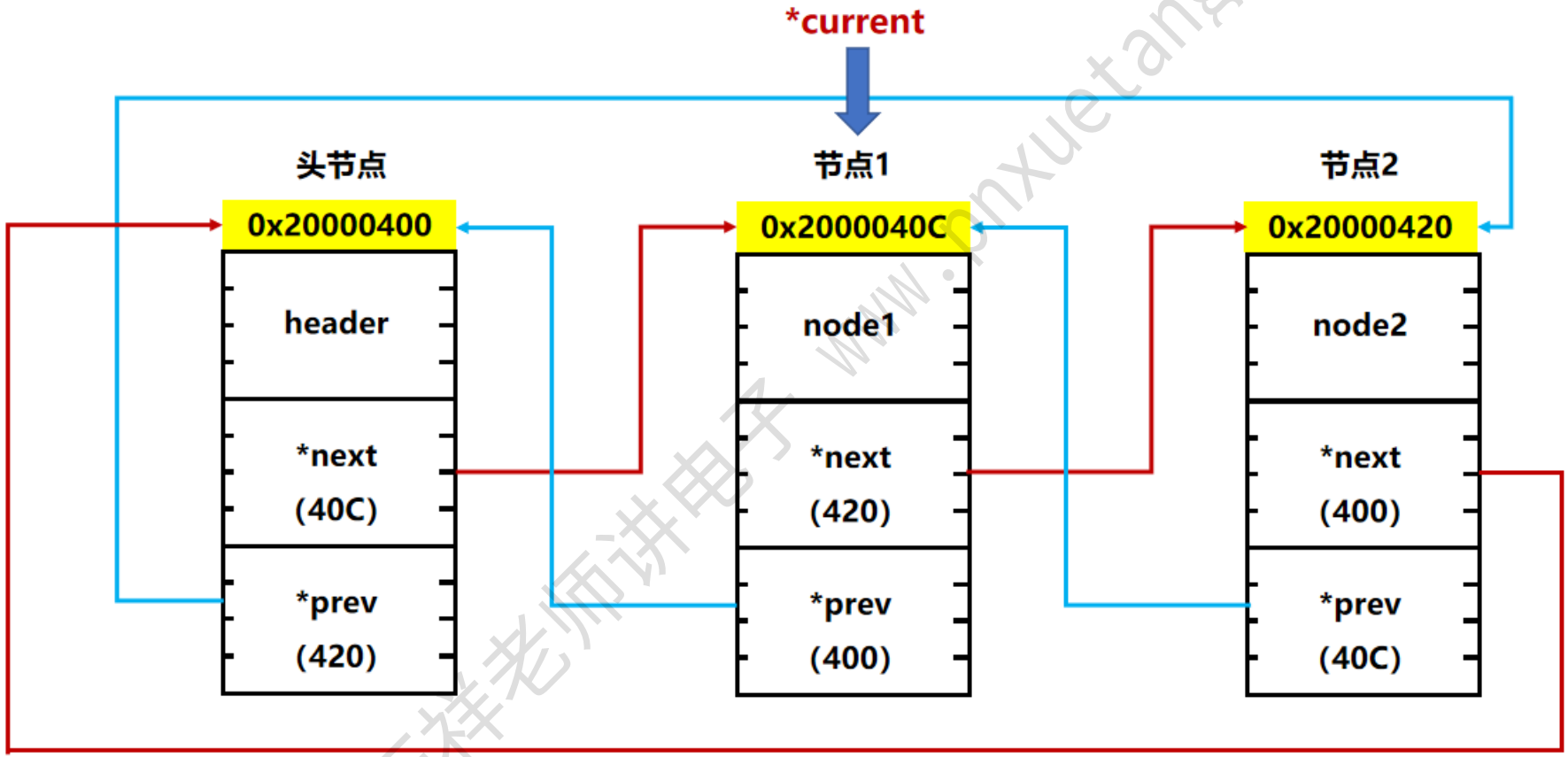
1. 修改新节点的next指向，指向节点2的首地址420；
2. 修改新节点的prev指向，指向节点1的首地址40C；
3. 修改节点2的prev指向(oldNode->next->prev)，从指向节点1首地址40C，改为指向新节点的首地址；
4. 修改节点1的next指向（oldNode->next），从指向节点2首地址420，改为指向新节点的首地址。

在任意节点后面添加节点



1. 修改新节点的next指向，指向节点2的首地址420：newNode->next = oldNode->next;
2. 修改新节点的prev指向，指向节点1的首地址40C：newNode->prev = oldNode;
3. 修改节点2的prev指向，从指向节点1首地址40C，改为指向新节点的首地址：oldNode->next->prev = newNode;
4. 修改节点1的next指向，从指向节点2首地址420，改为指向新节点的首地址：oldNode->next = newNode;

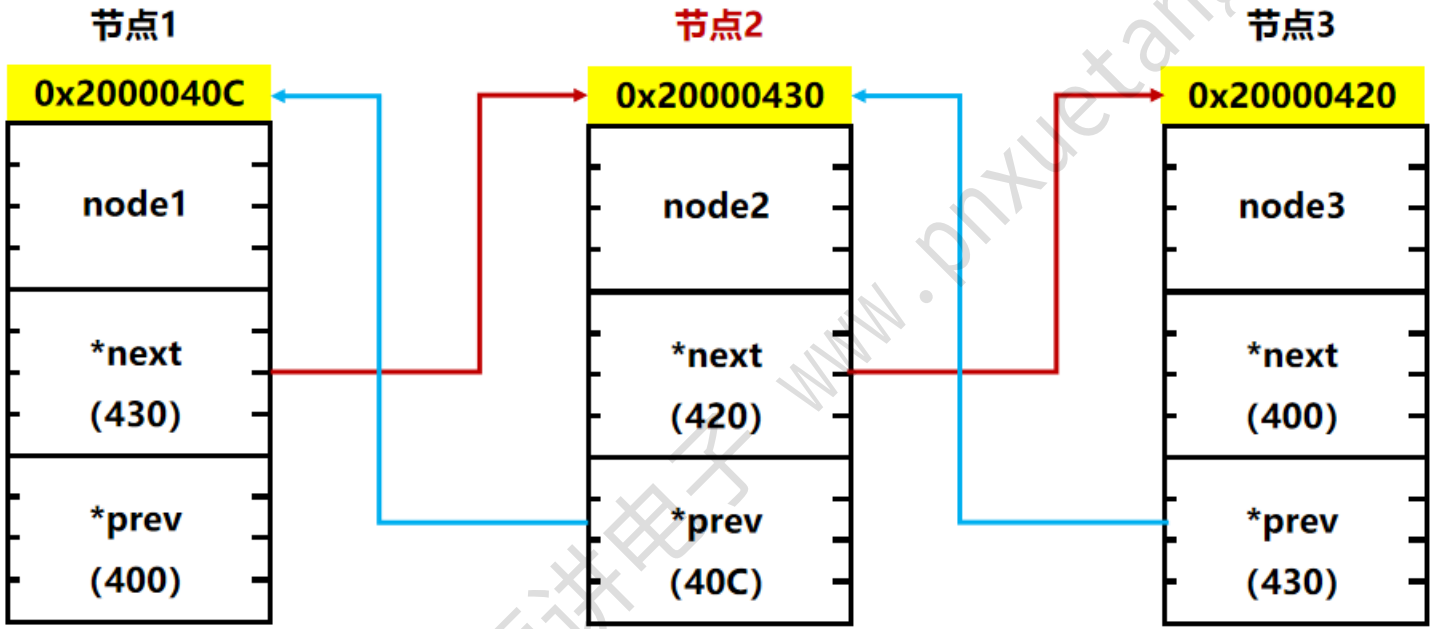
循环链表的遍历



循环链表的遍历

```
void PrintSensorData(TempHumiListNode *header)
{
    TempHumiListNode *current;
    current = header->next; //从头节点下一个节点开始遍历
    if (current == header) //判断是否只有一个头节点
    {
        printf("List has no node!\n");
        return;
    }
    while (current != header) //遍历到头节点，不再执行循环语句，退出循环
    {
        printf("\nSensor id:%d,temp = %.1f,humi = %d.\n", current->id, current->temp, current->humi);
        current = current->next;
    }
}
```

删除节点



1. 修改节点1（node->prev）的next指向（node->prev->next），从指向节点2首地址430，改为指向节点3的首地址（node->next）：node->prev->next = node->next;
2. 修改节点3（node->next）的prev指向（node->next->prev），从指向节点2首地址430，改为指向节点1的首地址（node->prev）：node->next->prev = node->prev;

THANK YOU!