Linked List

王慧妍

why@nju.edu.cn

南京大学 计算机科学与技术系 计算机软件研究所

回顾一点struct和指针

- 一些有趣的事情
  - struct-p-func.c

```
struct {
    int a;
    char b;
    float c;
} x[10], *p;
```

```
struct {
        char *name;
        char *des;
        int (*handler)(char *);
    } cmd_table[] = {
        {"i", "Print info", cmd_i},
        {"q", "Exit", cmd_q},
        {"c", "Continue", cmd_c}
    };
```

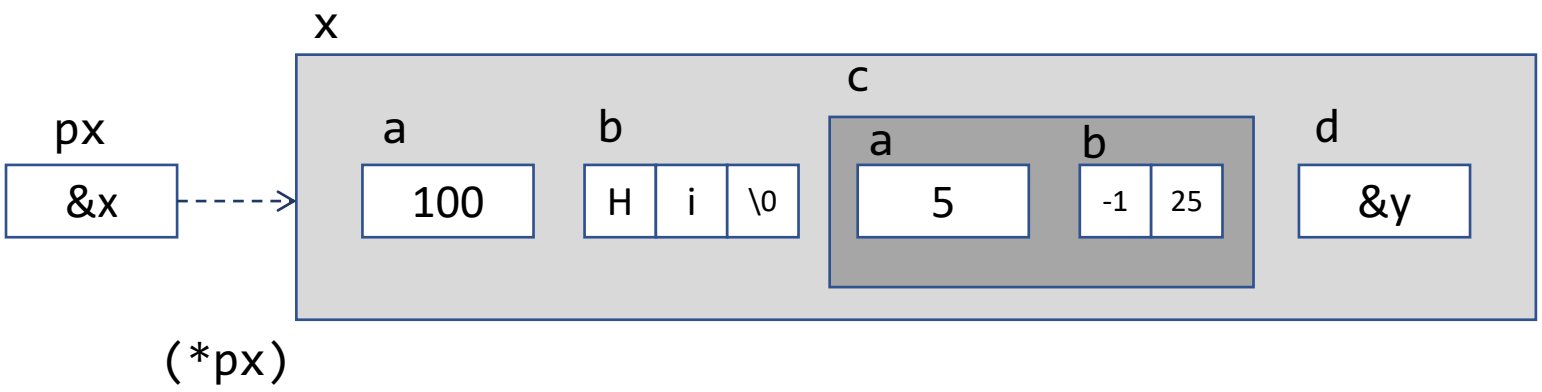
结构体的内存示意图

```
typedef struct {
    int a;
    short b[2];
} EX2;
```

```
typedef struct EX{
    int a;
    char b[3];
    EX2 c;
    struct EX *d;
} EX;
```

```
EX x;
EX *px = &x;
```

*px->c.b



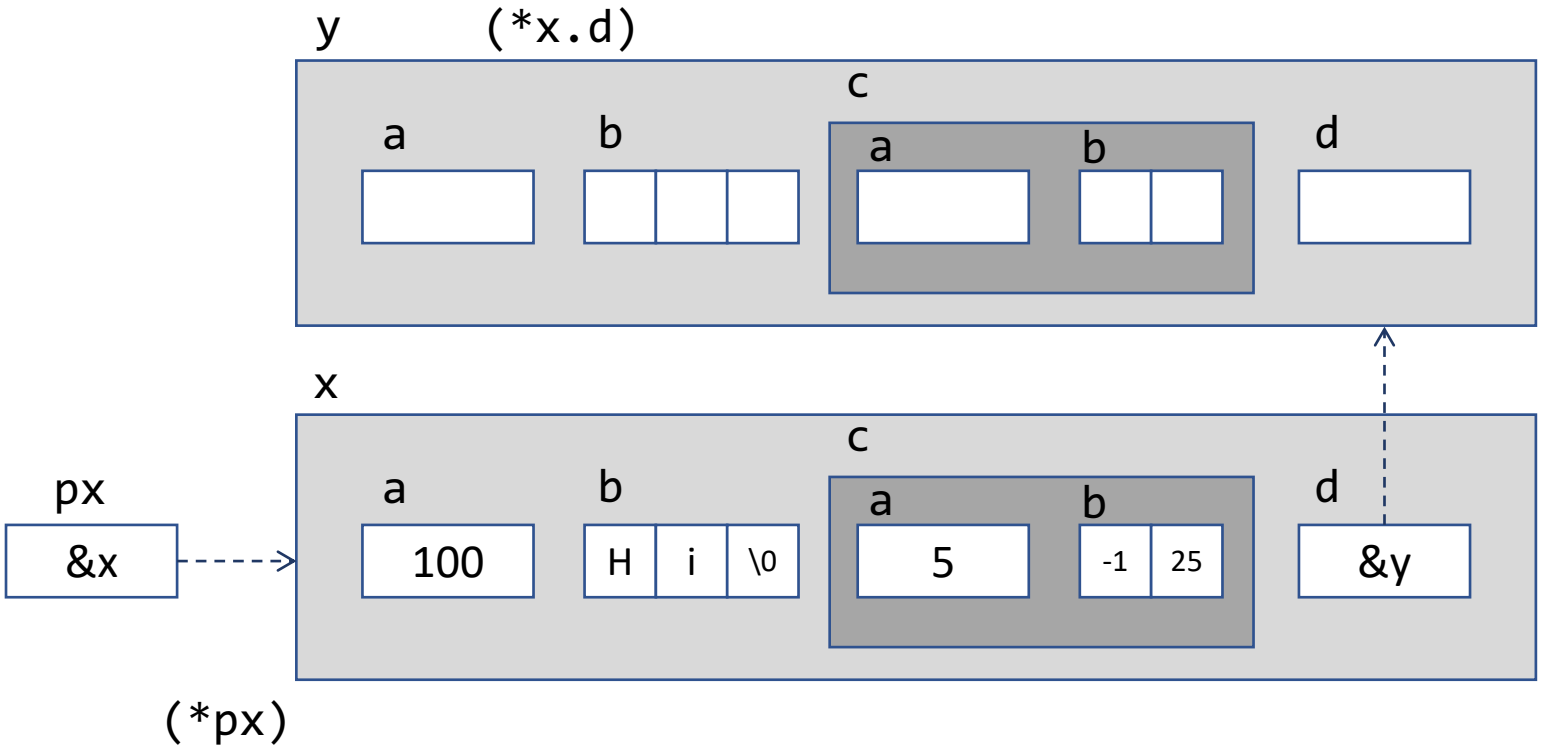

```
EX x = {100, {'H', 'i', '\0'}, {5, {-1, 25}}, 0};
```

```c
typedef struct {
    int a;
    short b[2];
} EX2;
```

```c
typedef struct EX{
    int a;
    char b[3];
    EX2 c;
    struct EX *d;
} EX;
```

```c
EX x;
EX *px = &x;
```

```c
EX y;
x.d = &y;
```




```c
EX x = {100, {'H', 'i', '\0'}, {5, {-1, 25}}, 0};
```

struct自引用

- 不同自引用的例子（分别合法么？）
  - 什么含义？

```
struct SELF{
    int a;
    struct SELF b;
    float c;
};
```

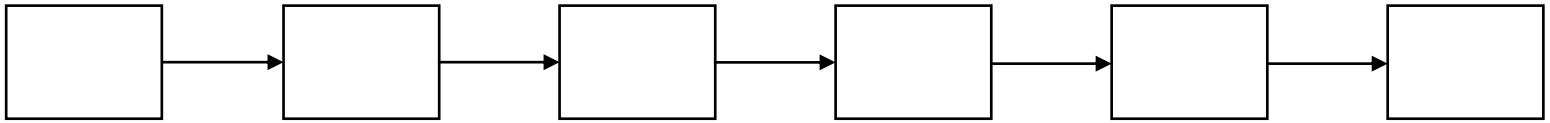
```
struct SELF{
    int a;
    struct SELF *b;
    float c;
};
```

```
typedef struct {
    int a;
    SELF *b;
    float c;
} SELF;
```

```
typedef struct SELF_TAG {
    int a;
    struct SELF_TAG *b;
    float c;
} SELF;
```

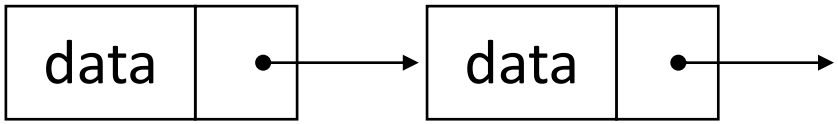
Linked List

- 链表：内存非连续的线性结构



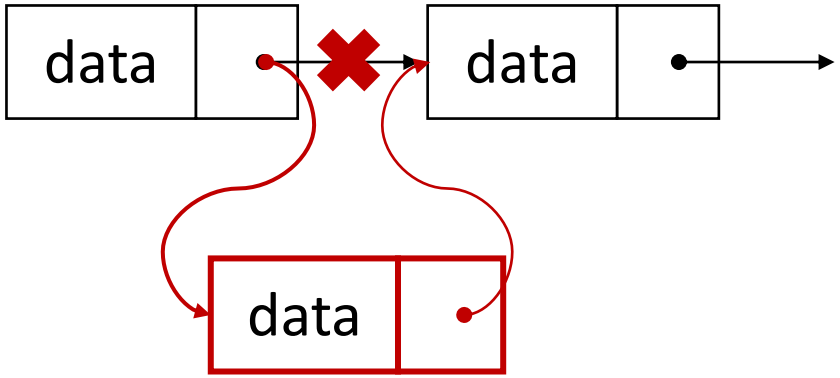
- 链表节点
  - 存储有价值的数据
  - 指向下一个链表节点的指针

  - `struct node {int data; struct node *next;};`

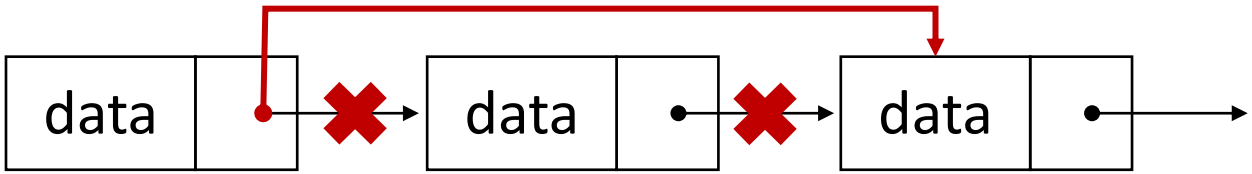


链表的节点插入和删除

- 新链表节点的插入



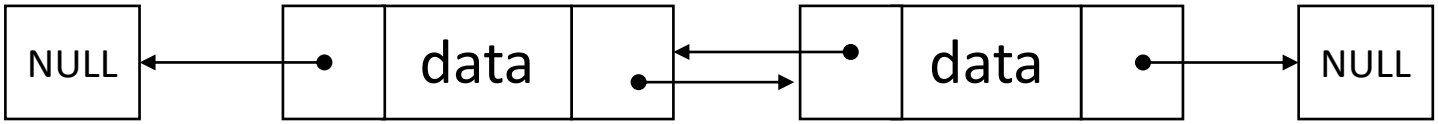
- 已有链表节点的删除



单向链表和双向链表

- 单向链表
  - `struct node {int data; struct node *next;};`

- 双向链表
  -
```
struct node {
  struct node * prev;
  int data;
  struct node *next;
};
```

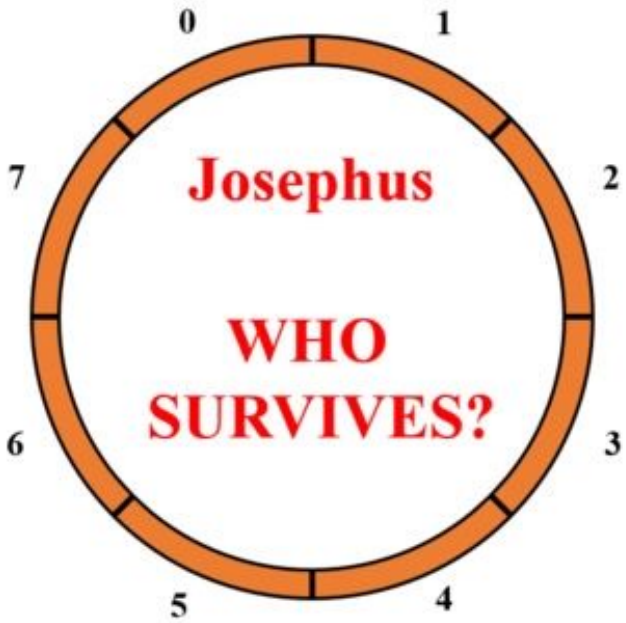


循环链表

- 与单向链表相比，区别在于
  - 尾节点的指针成员不再指向NULL，而是指向首节点

约瑟夫问题

- 由来
  - 在罗马人占领乔塔帕特后，39 个犹太人与Josephus 及他的朋友躲到一个洞中，39个犹太人决定宁愿死 也不要被敌人抓到，于是决定了一个自杀方式。41个人排成一个圆圈，由第1个人开始报数，每报数 到第3人该人就必须自杀，然后再由下一个重新报数 ，直到所有人都自杀身亡为止。Josephus要他的朋友先假装遵从，他将朋友与自己安排在第16个与第31个位置，于是逃过了这场死亡游戏 。



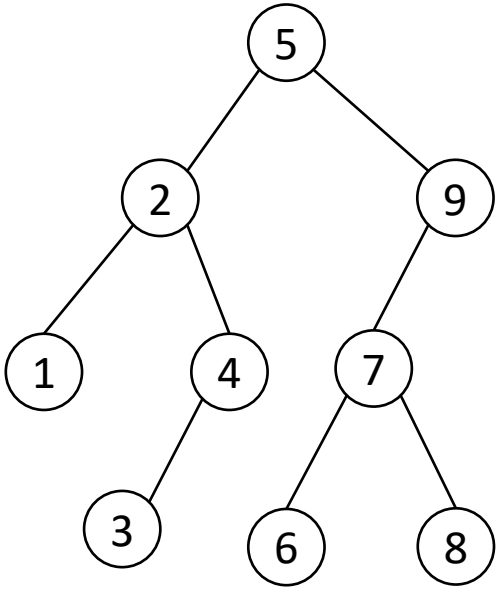
约瑟夫问题

- 用数组实现约瑟夫环？

- 试试用循环链表来实现约瑟夫环？
  - Joseph.c

更复杂点场景：链式结构实现二叉树

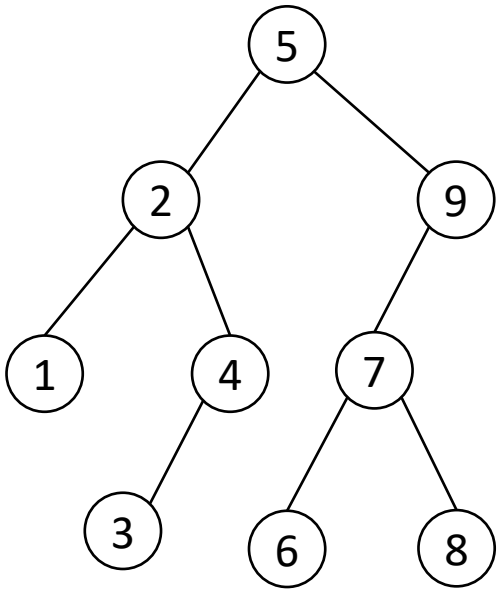
- 二叉搜索树(binary search tree)
  - 每个节点有其左孩子和右孩子
  - 额外属性：左孩子比当前节点值小，右孩子比当前节点值大




```
struct BNode {
    int value;
    struct BNode *left;
    struct BNode *right;
} *root;
```

更复杂点场景：链式结构实现二叉树

- 二叉树的遍历
  - 前序pre-order
    - 521439768
  - 中序in-order
    - 123456789
  - 后序post-order
    - 134268795



- BST.c

各种树

- 二叉树BST
- 平衡二叉树AVL
- 红黑树
- B树，B+树

Bit field

- 位域
  - 用一个字节中的不同二进制位来表达不同信息
  - 变量名：位数
  - Bitfield.c

```
typedef union inst {
   struct { u8 rs : 2, rt: 2,  op: 4; } rtype;
   struct { u8 addr: 4,        op: 4; } mtype;
} inst_t;
```

```
1  struct tcphdr {
2      __be16    source;
3      __be16    dest;
4      __be32    seq;
5      __be32    ack_seq;
6  #if defined(__LITTLE_ENDIAN_BITFIELD)
7      __u16    res1:4,
8           doff:4,
9           fin:1,
10          syn:1,
11          rst:1,
12          psh:1,
13          ack:1,
14          urg:1,
15          ece:1,
16          cwr:1;
17 #elif defined(__BIG_ENDIAN_BITFIELD)
26          syn:1,
27          fin:1;
28 #else
29 #error    "Adjust your <asm/byteorder.h> defines"
30 #endif
31     __be16    window;
32     __sum16    check;
33     __be16    urg_ptr;
34 };
```

End