



§. 链表参数传递思考题

1. 链表的建立是否正确

/* 2352018 大数据 刘彦 */

链表的建立逻辑大体上是正确的，但head指针的使用有一定问题。



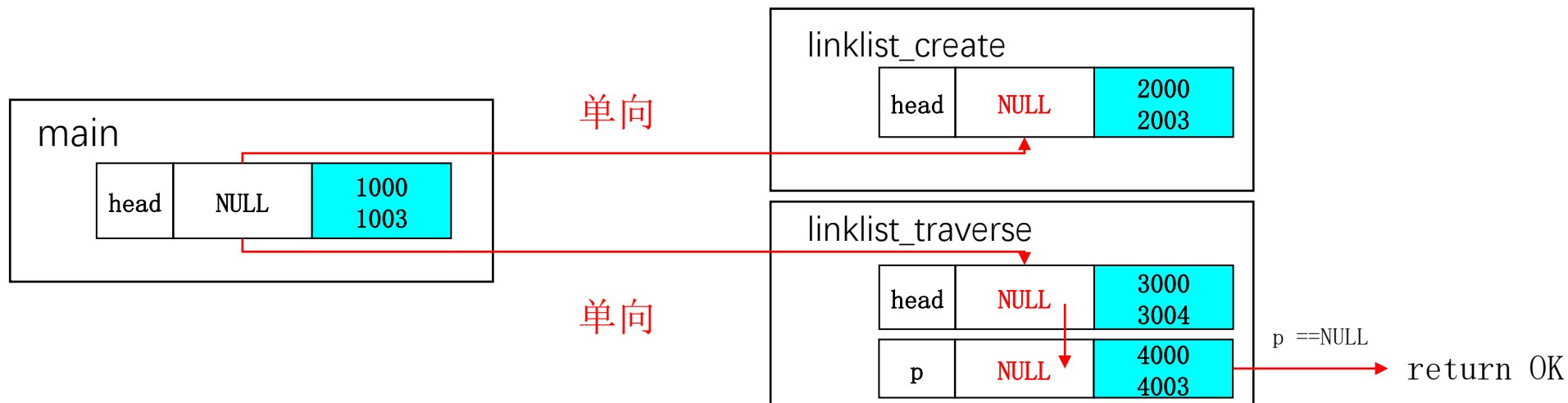
§. 链表参数传递思考题

2. 为什么遍历不成功

在linklist_create函数中，对head的赋值是直接在函数内进行的，这样会导致在main函数中head仍然为NULL。

原因是在原始代码中，head是在main函数中声明的，并且它是一个局部指针。然后，当你调用linklist_create时，传递的是head的值（即NULL），而不是它的地址。这意味着在linklist_create函数中对head的任何修改都不会影响main函数中的head指针。导致在linklist_traverse中遍历时没有节点可遍历。

为了让head指向链表的第一个节点，需要将它作为指向指针的参数传递。





§. 链表参数传递思考题

3. 链表的销毁是否成功了

链表中的每个节点都被释放，销毁是成功的。

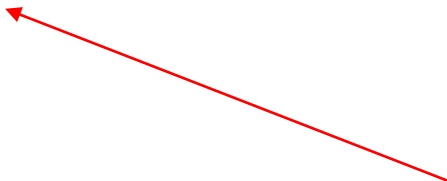


§. 链表参数传递思考题

4. 程序是否有内存丢失情况发生，如果有，发生在哪个函数被调用的阶段

在linklist_create函数被调用时，在节点分配阶段，某一节点的内存分配失败（malloc返回NULL），会直接返回ERROR，而没有释放已经分配的节点。这将导致已分配内存的丢失。

```
int linklist_create(struct student* head) {  
    ...  
    p = (struct student*)malloc(sizeof(struct student));  
    if (p == NULL)  
        return ERROR; // 注：此处未释放之前的链表节点  
    ...  
}
```

A red arrow points from the bottom right towards the 'return ERROR;' statement, highlighting the memory leak issue.



§. 链表参数传递思考题

5. 只允许修改某个函数的参数类型/该函数的声明, 并在该函数内部改动一个地方, main函数调用处改动一个地方, 使程序正确, 应该如何改动

该函数的声明

第16行, 原 int linklist create(struct student* head);, 新 int linklist create(struct student** head);

函数的参数类型

第27行, 原 int linklist create(struct student* head) {, 新 int linklist create(struct student** head) {

在该函数内部改动一个地方

第38行, 原 head = p;, 新 *head = p;

main函数调用处改动一个地方

第99行, 原 if (linklist create(head) == OK) {, 新 if (linklist create(&head) == OK) {