Linux内核链表

- 数据结构是构建操作系统的基础，和其它大型项目一样，Linux内核实现了通用数据结构，包括链表、队列、映射和二叉树。
- 在内核2.1中，首次引入了官方内核链表使用，Linux内核中使用了大量的链表结构来组织数据，包括设备列表以及各种功能模块中的数据组织。
- 链表代码在[include/linux/list.h]中，这是一个相当精彩的链表数据结构。为了做到通用性，在linux内核中的链表和普通链表的定义有着显著的区别，独树一帜。

Hacker精神--追求极致

Linux内核链表

Linux内核链表数据结构的定义

```
struct  list_head
{ struct  list_head  *next;
  struct  list_head  *prev;
}
```

- 内核的链表具备双链表功能，实际上，通常它都组织成**双循环链表**
- 和前面介绍的双链表结构模型不同，这里的**list_head没有数据域?**
- 在Linux内核链表中，**不是在链表结构中包含数据，而是在数据结构中包含链表节点?**

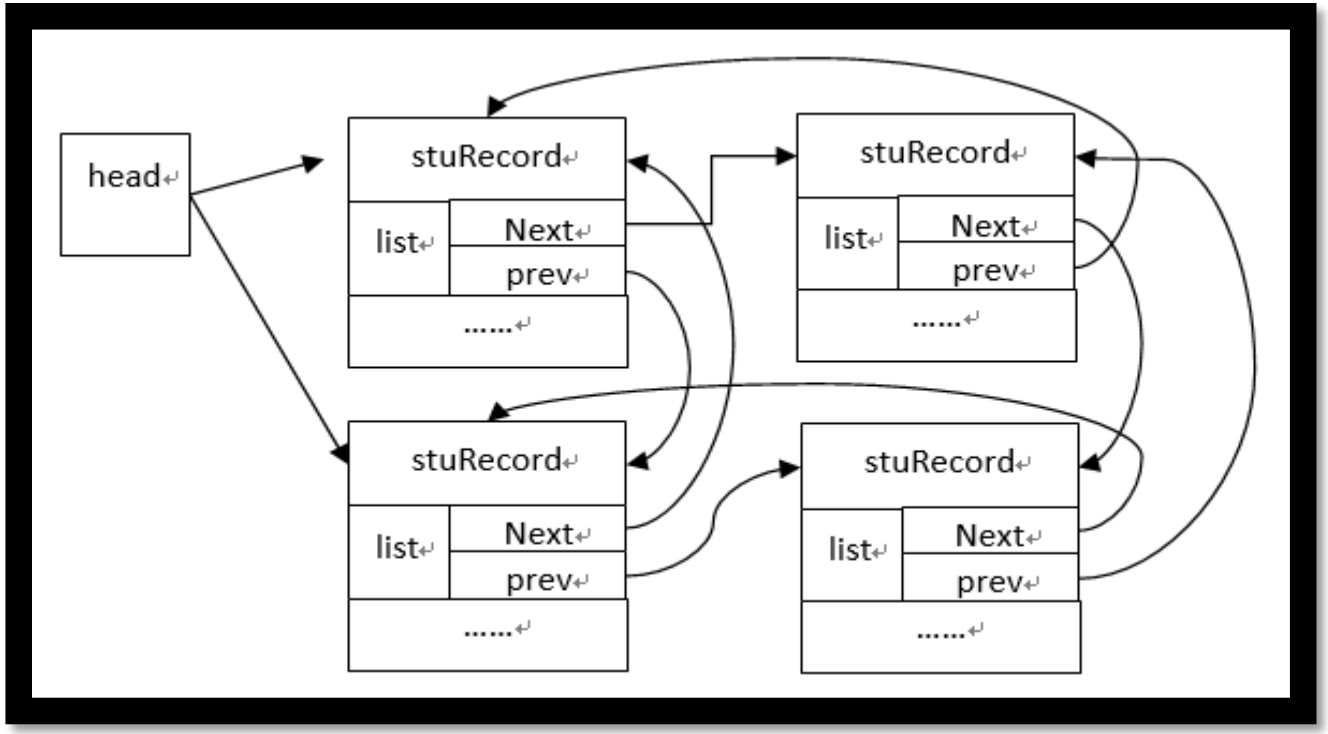
```
struct Student
{
    long num;
    char name[20];
    char sex[4];
    int  age;
    struct Student *next;
    struct Student *pre;
};
```

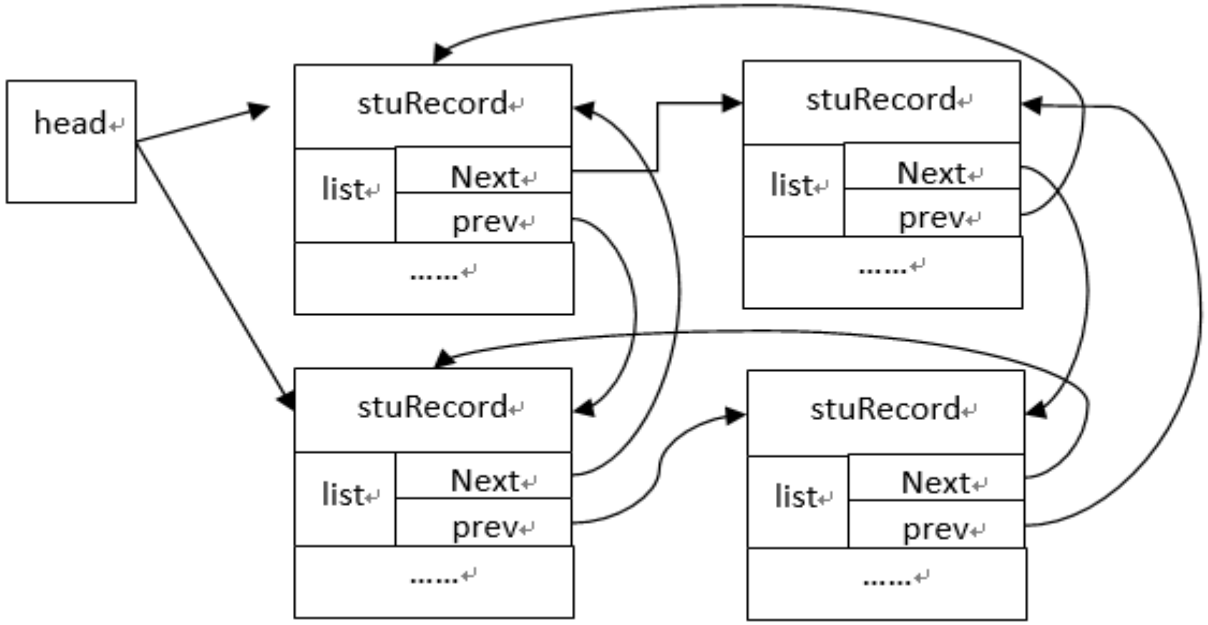
```
struct Book
{
    char num;
    char bname[50];
    char press[50];
    int NOWquantity;
    struct Student *next;
    struct Student *pre;
};
```

请注意：因为在不同的应用中数据类型不同，这样需要对每一种数据项类型都需要定义各自的链表结构。

Linux的简捷实用、不求完美和标准的风格，在这里体现得相当充分

```
struct StuRecord
{
    long num;
    char name[20];
    char sex[4];
    int  age;
    struct list_head  list; //内核链表
};
```





- list_head是结构StuRecord的成员变量，那么如何从链表指针list_head这个成员找到父结构StuRecord中包含的节点数据?
- 这里使用的是一个利用编译器技术的小技巧，即先求得结构成员在与结构中的偏移量，然后根据成员变量的地址反过来得出属主结构变量的地址

offsetof ()宏、container_of()宏和ListEntry ()宏

```
#define offsetof(TYPE, MEMBER) ((size_t) &((TYPE *)0)->MEMBER)
#define containerof(ptr, type, member) ({  (type *)( (char *)ptr - offsetof(type,member) );})
#define ListEntry (ptr, type, member)  containerof(ptr, type, member)
```

内核提供了一组接口操作链表，这些函数都要使用一个或多个list_head结构体指针参数，它们都是用C语言的内联函数形式实现的。所有这些函数的时间复杂度都是常数级的，即无论链表的大小，它们都在恒定时间内完成

- 初始化一个链表
- 插入结点
- 删除结点
- 遍历

思考1

任务：抽取linux内核链表接口实现一个基本的学生信息管理

思考2：Reuseability

任务：怎样将自己写的链表写成接口，并具有通用性呢?

说到通用性你想到了什么?

Stemplate

STL

.JAR

.LIB

DLL

.SO