kernel中的双链表的使用Linux认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_kernel_E4_B8_AD_E7_c103_644744.htm 参考例子如下： 你们可以把下面的代码粘贴到 自己的 代码里面去试验， 我给的只是个代码片段而已。 自己调试。 关于链表操作的具体实现， 可以看kernel代码里面的 include/linux/list.h 也可以在kernel代码中搜索 list_add/list_del/list_empty 等例子。 //声明一个链表 LIST_HEAD(bob_list). struct bob_jobs { unsigned long id. struct list_head list. }. #define this_job(p) list_entry(p, struct bob_jobs, list) static __init int chardev_init(void) { struct list_head *p = NULL. //begin struct bob_jobs job1 = { .id = 10UL, //.list = LIST_HEAD_INIT(list), 因为我们的结构体刚定义，list指针本身就是未知的，需要在list_add()才能对其进行操作，不用没法赋值。 }. struct bob_jobs job2 = { .id = 20UL, //.list = LIST_HEAD_INIT(list), 因为我们的结构体刚定义，list指针本身就是未知的，需要在list_add()才能对其进行操作，不用没法赋值。 }. Major = register_chrdev(0, DEVICE_NAME, lt. 0) { dbg("Registering the character device failed with %d\n", Major). return Major. } dbg("only debug the functions in list.h\n"). INIT_LIST_HEAD(amp.(job1.list),amp.(job2.list),amp.bob_list) { struct bob_jobs *job = NULL. job = this_job(p). dbg("id=%lu\n",job-amp.bob_list) { struct bob_jobs *job = NULL. p = bob_list.next. job = this_job(p). //0delete a entry after header // dbg("0delete a entry after header , id = %lu\n",(this_job(p))-gt.id). list_del(bob_list.next). } dbg("0delete over \n"). return 0. } 100Test