Operating Systems hw1 **105061212 王家駿**

Code and description:

1. Header files and list structure:

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/list.h>

MODULE_DESCRIPTION("Sample_module");
MODULE_LICENSE("GPL");

// declare the list structure with four integers and one pointer
struct numlist{
    int id;
    int yy;
    int mm;
    int dd;
    struct list_head list;
};

struct numlist numhead;  // create new list numhead
```

At first, we import the header flies we would use, including <linux/list.h> for creating the link-list. Then we declare the list structure, with nodes containing four integers: student ID, year, month, day and one list_head type “list” which contains two pointers pointing to the previous and the next nodes. And then we create a new list called “numhead”.

2. Construction function:

```c
//consturction function
static int __init sample_init(void)
{
    struct numlist *listnode;  // temporary node for appending behind
    struct list_head *itr;  // iteration pointer
    struct numlist *p;  // temporary node for printing results
    int i;  // loop index
    int stid = 105061212;  // initial value of student ID
    int yr = 1998;  // initial value of year
    int mn = 5;  // initial value of month
    int dy = 20;  // initial value of day

    INIT_LIST_HEAD(&numhead.list);  // initializing the list

    for(i=0; i<5; i++)  // repeat five times
    {
        listnode = (struct numlist *)vmalloc(sizeof(struct numlist));  // allocate dynamic memories
        listnode->id = stid;  // write values
        listnode->yy = yr;
        listnode->mm = mn;
        listnode->dd = dy;

        list_add_tail(&listnode->list,&numhead.list);  // append the node at the end of the list

        stid+=2; yr+=1; mn-=1; dy+=2;  // data increment
    }

    list_for_each(itr,&numhead.list){  // for each list node
        p = list_entry(itr,struct numlist,list);  // point to each node
        printk("%d, %d-%d-%d.\n", p->id, p->dd, p->mm, p->yy);  // print the data
    }

    return 0;
}
```

When this kernel module is loaded, the list is created and initialized by INIT_LIST_HEAD function. Then for the next five iteration steps, we assign the value of each node, and append them to the end of the list by list_add_tail function. The input data are slightly changed by -1~+2 to show the difference of the data.

In order to print out the data in the list, we iterates over the list, and show out all the contents in the kernel load buffer.

3. Destruction function:

```c
// destruction function
static void __exit sample_exit(void)
{
    struct list_head *itr,*n;  // iteration pointers
    struct numlist *p;  // temporary node for deleting

    list_for_each_safe(itr, n, &numhead.list)  // for each list node
    {
        list_del(itr);  // delete the node
        p = list_entry(itr, struct numlist, list);  // point to each node
        vfree(p);  // release memories
    }
}

// call functions
module_init(sample_init);
module_exit(sample_exit);
```

When this kernel module is removed, we iterate through the list, and clear the node. Then we release the dynamic memories of the node cleared.

Execution results:

```
[ 7936.810099] 105061212, 20-5-1998.
[ 7936.810100] 105061214, 22-4-1999.
[ 7936.810101] 105061216, 24-3-2000.
[ 7936.810101] 105061218, 26-2-2001.
[ 7936.810102] 105061220, 28-1-2002.
```

The list is constructed successfully, with the correct content in each node.