МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №5

з дисципліни «Дослідження та проектування комп’ютерних систем - 2»

Виконав:

студент гр. ІО-91мн

Глушенок В.О.

Перевірив:

Каплунов А.В.

Київ 2020

**Завдання**

Завдання Basic:

Оголосити структуру даних для розміщення у списку, яка крім елемента struct list\_head містить поле типу ktime\_t

(include/linux/ktime.h у вашому репозиторії linux-stable).

Створити статичну змінну голови списку.

Перед кожним друком привітання виділити пам'ять для екземпляра оголошеної структури, занести в неї поточний час ядра, отриманий функцією ktime\_get().

У функції hello\_exit() пройти по списку і надрукувати час кожної події в наносекундах, вилучити елемент списку і звільнити виділену пам'ять. Приклад проходу по списку з вилученням елемента є у appendix.

На даному етапі досить виділяти пам'ять викликом

ptr = kmalloc(sizeof(\*ptr), GFP\_KERNEL);

і звільнити її викликом

kfree(ptr);

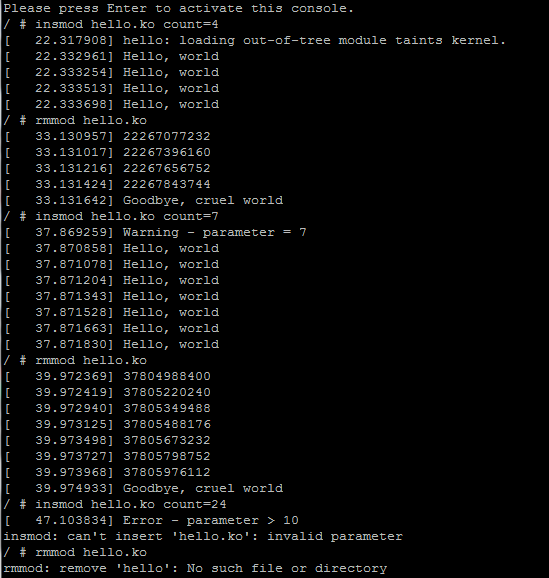
Докладніше у додатку Allocating Memory.

Завдання Advanced:

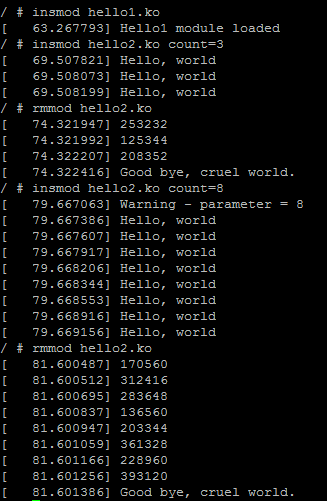
У цьому завданні вся робота зі списком виконується в модулі hello1. Також додати ще одне поле типу ktime\_t і засікати час до та після виклику функції друку, а на вивантаженні модуля надрукувати час, який пішов на кожен друк.

**Приклад виконання програми**

Basic:



Advanced:



**Програмний код**

**hello.c**

|  |
| --- |
| #include <linux/init.h> |
|  | #include <linux/module.h> |
|  | #include <linux/moduleparam.h> |
|  | #include <linux/errno.h> |
|  | #include <linux/ktime.h> |
|  | #include <linux/slab.h> |
|  |  |
|  | MODULE\_LICENSE("Dual BSD/GPL"); |
|  |  |
|  | static uint count = 1; |
|  | module\_param(count, uint, S\_IRUGO); |
|  | MODULE\_PARM\_DESC(count, "Here is parameter description"); |
|  |  |
|  | struct time\_list { |
|  | struct list\_head header; |
|  | ktime\_t time; |
|  | }; |
|  |  |
|  | static LIST\_HEAD(time\_list\_head); |
|  |  |
|  | void list\_free(struct list\_head \*head) |
|  | { |
|  | struct list\_head \*pos, \*n; |
|  |  |
|  | list\_for\_each\_safe(pos, n, head) { |
|  | kfree(pos); |
|  | } |
|  |  |
|  | INIT\_LIST\_HEAD(head); |
|  | } |
|  |  |
|  | static int \_\_init hello\_init(void) |
|  | { |
|  | if (count > 10) { |
|  | printk(KERN\_ERR "Error - parameter > 10\n"); |
|  | return -EINVAL; |
|  | } |
|  |  |
|  | if (!count || (count >= 5 && count <= 10)) |
|  | printk(KERN\_WARNING "Warning - parameter = %u\n", count); |
|  |  |
|  | struct time\_list \*next; |
|  | uint i; |
|  |  |
|  | for (i = 0; i < count; ++i) { |
|  | next = kmalloc(sizeof(struct time\_list), GFP\_KERNEL); |
|  | if (next) { |
|  | list\_add\_tail(&next->header, &time\_list\_head); |
|  |  |
|  | next->time = ktime\_get(); |
|  |  |
|  | printk(KERN\_NOTICE "Hello, world\n"); |
|  | } else { |
|  | list\_free(&time\_list\_head); |
|  | return -ENOMEM; |
|  | } |
|  | } |
|  |  |
|  | return 0; |
|  | } |
|  | module\_init(hello\_init); |
|  |  |
|  | static void \_\_exit hello\_exit(void) |
|  | { |
|  | struct list\_head \*pos; |
|  |  |
|  | list\_for\_each(pos, &time\_list\_head) { |
|  | printk(KERN\_NOTICE "%lld", ktime\_to\_ns(list\_entry(pos, struct time\_list, header)->time)); |
|  | } |
|  |  |
|  | list\_free(&time\_list\_head); |
|  |  |
|  | printk(KERN\_NOTICE "Goodbye, cruel world\n"); |
|  | } |
|  | module\_exit(hello\_exit); |

**Makefile**

|  |
| --- |
| # If KERNELRELEASE is defined, we've been invoked from the |
|  | # kernel build system and can use its language. |
|  | ifneq ($(KERNELRELEASE),) |
|  | obj-m := hello.o |
|  | # Otherwise we were called directly from the command |
|  | # line; invoke the kernel build system. |
|  | else |
|  | KDIR ?= /lib/modules/$(shell uname -r)/build |
|  | PWD := $(shell pwd) |
|  | default: |
|  | $(MAKE) -C $(KDIR) M=$(PWD) modules |
|  | endif |

**hello1.c**

|  |
| --- |
| #include "hello1.h" |
|  |  |
|  | #include <linux/errno.h> |
|  | #include <linux/ktime.h> |
|  | #include <linux/slab.h> |
|  |  |
|  | struct time\_list { |
|  | struct list\_head header; |
|  | ktime\_t begin\_time, end\_time; |
|  | }; |
|  |  |
|  | static LIST\_HEAD(time\_list\_head); |
|  |  |
|  | void list\_free(struct list\_head \*head) |
|  | { |
|  | struct list\_head \*pos, \*n; |
|  |  |
|  | list\_for\_each\_safe(pos, n, head) { |
|  | kfree(pos); |
|  | } |
|  |  |
|  | INIT\_LIST\_HEAD(head); |
|  | } |
|  |  |
|  | static int \_\_init hello1\_init(void) |
|  | { |
|  | pr\_info("Hello1 module loaded\n"); |
|  | return 0; |
|  | } |
|  | module\_init(hello1\_init); |
|  |  |
|  | int print\_hello(uint n) |
|  | { |
|  | struct time\_list \*next; |
|  | uint i; |
|  |  |
|  | if (n > 10) { |
|  | pr\_err("Error - parameter > 10\n"); |
|  | return -EINVAL; |
|  | } |
|  |  |
|  | if (!n || (n >= 5 && n <= 10)) |
|  | pr\_warn("Warning - parameter = %u\n", n); |
|  |  |
|  | for (i = 0; i < n; ++i) { |
|  | next = kmalloc(sizeof(struct time\_list), GFP\_KERNEL); |
|  | if (next) { |
|  | list\_add\_tail(&next->header, &time\_list\_head); |
|  |  |
|  | next->begin\_time = ktime\_get(); |
|  |  |
|  | pr\_info("Hello, world\n"); |
|  |  |
|  | next->end\_time = ktime\_get(); |
|  | } else { |
|  | list\_free(&time\_list\_head); |
|  | return -ENOMEM; |
|  | } |
|  | } |
|  |  |
|  | return 0; |
|  | } |
|  | EXPORT\_SYMBOL(print\_hello); |
|  |  |
|  | void print\_time(void) |
|  | { |
|  | struct list\_head \*pos; |
|  |  |
|  | list\_for\_each(pos, &time\_list\_head) { |
|  | struct time\_list \*struct\_buff = list\_entry(pos, |
|  | struct time\_list, header); |
|  |  |
|  | pr\_info("%lld", ktime\_to\_ns(struct\_buff->end\_time) |
|  | - ktime\_to\_ns(struct\_buff->begin\_time)); |
|  | } |
|  |  |
|  | list\_free(&time\_list\_head); |
|  | } |
|  | EXPORT\_SYMBOL(print\_time); |
|  |  |
|  | static void \_\_exit hello1\_exit(void) |
|  | { |
|  | pr\_info("Hello1 module unloaded\n"); |
|  | } |
|  | module\_exit(hello1\_exit); |

**hello2.c**

|  |
| --- |
| #include "hello1.h" |
|  |  |
|  | #include <linux/moduleparam.h> |
|  |  |
|  | static uint count = 1; |
|  | module\_param(count, uint, 0444); |
|  | MODULE\_PARM\_DESC(count, "Here is parameter description"); |
|  |  |
|  | extern int print\_hello(uint n); |
|  | extern void print\_time(void); |
|  |  |
|  | static int \_\_init hello2\_init(void) |
|  | { |
|  | return print\_hello(count); |
|  | } |
|  | module\_init(hello2\_init); |
|  |  |
|  | static void \_\_exit hello2\_exit(void) |
|  | { |
|  | print\_time(); |
|  |  |
|  | pr\_info("Good bye, cruel world.\n"); |
|  | } |
|  | module\_exit(hello2\_exit); |

**Makefile**

|  |
| --- |
| # If KERNELRELEASE is defined, we've been invoked from the |
|  | # kernel build system and can use its language. |
|  | ifneq ($(KERNELRELEASE),) |
|  | obj-m := hello1.o hello2.o |
|  |  |
|  | ccflags-y := -I$(src)/inc |
|  | # Otherwise we were called directly from the command |
|  | # line; invoke the kernel build system. |
|  | else |
|  | KDIR ?= /lib/modules/$(shell uname -r)/build |
|  | PWD := $(shell pwd) |
|  | default: |
|  | $(MAKE) -C $(KDIR) M=$(PWD) modules |
|  | endif |