|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 25/10/1403 | تمرین 4 | آزمایشگاه سیستم عامل | 4001830235 | محمد مهدی رسول امینی |

ماژول هسته – روشن و خاموش شدن چراغ LED

استاد کد ها برای من اجرا نشدن بخش پیاده سازی کد رو با خانم موسوی انجام دادیم.

**خلاصه اهداف، روش‌ها و نتایج:**

**هدف:** هدف این پروژه، طراحی و پیاده‌سازی یک ماژول هسته لینوکس است که بتواند چراغ LED مربوط به Caps Lock صفحه‌کلید را به‌صورت چشمک‌زن کنترل کند. این پروژه به ما امکان می‌دهد تا با نحوه کار تایمرها، دستگاه‌های ورودی و ارتباط ماژول با هسته آشنا شویم.

**روش‌ها:** برای رسیدن به هدف، ماژول به زبان C نوشته شد و از زیرسیستم ورودی لینوکس (Linux Input Subsystem) برای مدیریت LEDها استفاده شد. همچنین از تایمرهای هسته برای ایجاد عملکرد دوره‌ای استفاده گردید.

### **اطلاعات پیش‌زمینه‌ای و معرفی موضوع**

ماژول‌های هسته لینوکس، قابلیت گسترش عملکرد هسته بدون نیاز به کامپایل مجدد آن را فراهم می‌کنند. این پروژه به عنوان یک تمرین عملی، نحوه ایجاد و مدیریت یک ماژول ساده که با سخت‌افزار صفحه‌کلید در ارتباط است را بررسی می‌کند.

### **ابزارها و نرم‌افزارهای استفاده‌شده**

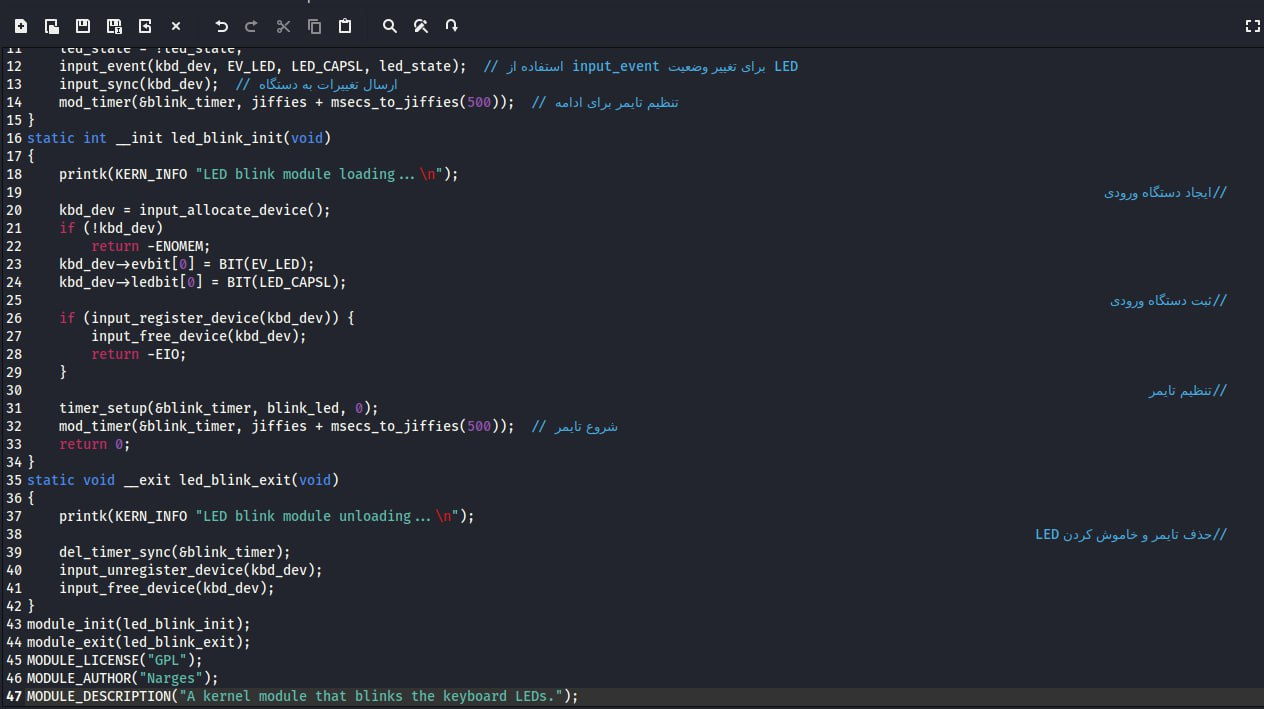
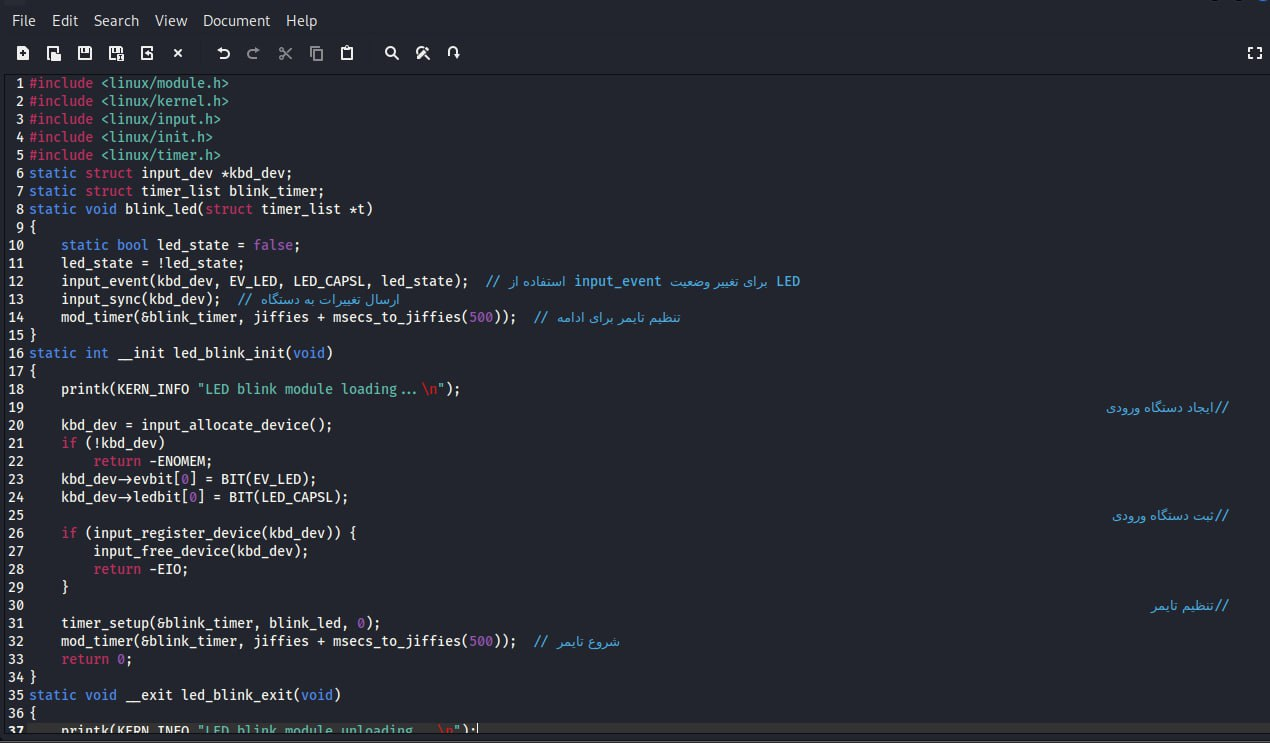
1. **سیستم‌عامل:** لینوکس (Kali Linux)
2. **ابزارهای توسعه:** GCC، Makefile
3. **زبان برنامه‌نویسی:** C
4. **مستندات مرجع:** مستندات هسته لینوکس و راهنمای برنامه‌نویسی ماژول‌های هسته

### **مراحل انجام پروژه به‌صورت گام‌به‌گام**

1. نوشتن کد ماژول در فایل c با استفاده از زیرسیستم ورودی و تایمرهای هسته.
2. ایجاد فایل Makefile برای کامپایل ماژول.
3. کامپایل کد با استفاده از دستور make.
4. بارگذاری ماژول با استفاده از دستور insmod.
5. مشاهده عملکرد ماژول از طریق روشن و خاموش شدن چراغ LED صفحه‌کلید.
6. حذف ماژول از هسته با دستور rmmod.

#### ****پیاده سازی بخش کد****

توضیحات کامل و خط به خط کد



### **توضیح خط به خط کد:**

### **هدر فایل‌ها**

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/input.h>

#include <linux/init.h>

#include <linux/timer.h>

* **linux/module.h**: شامل توابع ماژول‌های هسته است (مانند module\_init و module\_exit).
* **linux/kernel.h**: شامل توابع و ماکروهای مرتبط با هسته است (مانند printk).
* **linux/input.h**: برای مدیریت دستگاه‌های ورودی (مانند صفحه‌کلید) استفاده می‌شود.
* **linux/init.h**: شامل ماکروهایی برای مشخص کردن توابع ورودی و خروجی ماژول است.
* **linux/timer.h**: برای مدیریت تایمرها در هسته استفاده می‌شود.

### **تعریف متغیرها**

static struct input\_dev \*kbd\_dev;

static struct timer\_list blink\_timer;

* **kbd\_dev**: اشاره‌گری به ساختار دستگاه ورودی که برای کنترل صفحه‌کلید استفاده می‌شود.
* **blink\_timer**: یک تایمر که زمان‌بندی روشن و خاموش شدن LED را کنترل می‌کند.

### **تابع کنترل LED**

static void blink\_led(struct timer\_list \*t)

{

static bool led\_state = false; // وضعیت LED (روشن یا خاموش)

led\_state = !led\_state; // تغییر وضعیت LED (برعکس کردن حالت قبلی)

input\_event(kbd\_dev, EV\_LED, LED\_CAPSL, led\_state); // ارسال رویداد تغییر وضعیت LED

input\_sync(kbd\_dev); // همگام‌سازی تغییرات با دستگاه

mod\_timer(&blink\_timer, jiffies + msecs\_to\_jiffies(500)); // برنامه‌ریزی مجدد تایمر برای 500 میلی‌ثانیه دیگر

}

* **led\_state**: یک متغیر ثابت که وضعیت روشن یا خاموش بودن LED را ذخیره می‌کند.
* **input\_event**: رویداد تغییر وضعیت LED را به هسته ارسال می‌کند.
  + **EV\_LED**: نوع رویداد ورودی (کنترل LED).
  + **LED\_CAPSL**: LED مربوط به Caps Lock.
  + **led\_state**: وضعیت جدید LED.
* **input\_sync**: تغییرات انجام‌شده را به دستگاه اعمال می‌کند.
* **mod\_timer**: تایمر را برای اجرای مجدد تابع بعد از 500 میلی‌ثانیه تنظیم می‌کند.

### **تابع بارگذاری ماژول**

static int \_\_init led\_blink\_init(void)

{

printk(KERN\_INFO "LED blink module loading...\n"); // پیام در زمان بارگذاری ماژول

kbd\_dev = input\_allocate\_device(); // تخصیص دستگاه ورودی جدید

if (!kbd\_dev) // اگر تخصیص دستگاه ورودی ناموفق باشد

return -ENOMEM; // بازگشت خطای کمبود حافظه

kbd\_dev->evbit[0] = BIT(EV\_LED); // تنظیم پشتیبانی از رویدادهای LED

kbd\_dev->ledbit[0] = BIT(LED\_CAPSL); // فعال‌سازی LED Caps Lock

if (input\_register\_device(kbd\_dev)) { // ثبت دستگاه ورودی در هسته

input\_free\_device(kbd\_dev); // آزادسازی دستگاه در صورت خطا

return -EIO; // بازگشت خطای ورودی/خروجی

}

timer\_setup(&blink\_timer, blink\_led, 0); // تنظیم تایمر و اتصال آن به تابع blink\_led

mod\_timer(&blink\_timer, jiffies + msecs\_to\_jiffies(500)); // شروع تایمر برای اجرای اولین بار

return 0; // موفقیت‌آمیز بودن بارگذاری ماژول

}

* **printk**: پیام به لاگ هسته ارسال می‌کند.
* **input\_allocate\_device**: یک دستگاه ورودی جدید ایجاد می‌کند.
* **kbd\_dev->evbit و kbd\_dev->ledbit**: تنظیم رویدادهای ورودی و LED پشتیبانی‌شده.
* **input\_register\_device**: دستگاه ورودی را در سیستم ثبت می‌کند.
* **timer\_setup**: تایمر را تنظیم کرده و آن را به تابع blink\_led متصل می‌کند.

### **تابع خروج ماژول**

static void \_\_exit led\_blink\_exit(void)

{

printk(KERN\_INFO "LED blink module unloading...\n"); // پیام در زمان حذف ماژول

del\_timer\_sync(&blink\_timer); // حذف تایمر

input\_unregister\_device(kbd\_dev); // لغو ثبت دستگاه ورودی

input\_free\_device(kbd\_dev); // آزادسازی حافظه دستگاه

}

* **del\_timer\_sync**: تایمر را حذف می‌کند و اطمینان می‌دهد که دیگر اجرا نمی‌شود.
* **input\_unregister\_device**: دستگاه ورودی را از سیستم حذف می‌کند.
* **input\_free\_device**: حافظه اختصاص‌داده‌شده برای دستگاه را آزاد می‌کند.

### **ماکروهای ماژول**

module\_init(led\_blink\_init);

module\_exit(led\_blink\_exit);

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("Narges");

MODULE\_DESCRIPTION("A kernel module that blinks the keyboard LEDs.");

* **module\_init و module\_exit**: توابع ورودی و خروجی ماژول را مشخص می‌کنند.
* **MODULE\_LICENSE**: نوع مجوز ماژول را مشخص می‌کند (GPL).
* **MODULE\_AUTHOR**: نویسنده ماژول را مشخص می‌کند.
* **MODULE\_DESCRIPTION**: توضیحی کوتاه درباره ماژول ارائه می‌دهد.

### **اطلاعات ماژول**

### MODULE\_LICENSE("GPL"); // مجوز استفاده

MODULE\_AUTHOR("Narges"); // نویسنده ماژول

MODULE\_DESCRIPTION("A kernel module that blinks the keyboard LEDs."); // توضیح ماژول

* **MODULE\_LICENSE**: نوع مجوز استفاده از ماژول را مشخص می‌کند. در اینجا از GPL (GNU General Public License) برای استفاده آزاد انتخاب شده است.
* **MODULE\_AUTHOR**: نام نویسنده یا توسعه‌دهنده ماژول را نمایش می‌دهد.
* **MODULE\_DESCRIPTION**: توضیح مختصری درباره عملکرد ماژول ارائه می‌دهد. در اینجا این ماژول برای چشمک زدن چراغ‌های LED صفحه‌کلید طراحی شده است.

### **فایل Makefile و توضیحات آن**

برای ساخت ماژول‌های هسته لینوکس، باید فایل Makefile در کنار کد ماژول ایجاد شود. این فایل به کامپایلر دستور می‌دهد که چگونه کد منبع را به یک ماژول بارگذاری‌شده (.ko) تبدیل کند.

**محتوای فایل Makefile:**

Makefile

obj-m += narges.o

all:

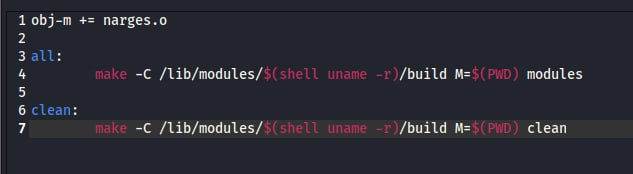
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:

make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

### **توضیحات فایل Makefile**

1. **obj-m += narges.o**
   * این خط به سیستم می‌گوید که ماژول narges.o باید ساخته شود.
   * نام narges.o بر اساس نام فایل کد منبع ماژول (در اینجا narges.c) انتخاب می‌شود.
2. **هدف all:**
   * اجرای این هدف (با دستور make) فرآیند کامپایل ماژول را آغاز می‌کند.
   * دستور make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules:
     + **-C**: دایرکتوری ساخت هسته لینوکس را مشخص می‌کند.
     + **$(shell uname -r)**: نسخه فعلی هسته را بازیابی می‌کند.
     + **M=$(PWD)**: دایرکتوری فعلی (محل فایل کد) را به کامپایلر معرفی می‌کند.
     + **modules**: به کامپایلر دستور می‌دهد که ماژول را بسازد.
3. **هدف clean:**
   * اجرای این هدف (با دستور make clean) فایل‌های موقت تولیدشده در حین کامپایل را حذف می‌کند:
     + narges.o: فایل شیء مربوط به ماژول.
     + narges.mod.o: فایل متادیتای ماژول.
     + narges.ko: فایل ماژول قابل بارگذاری.

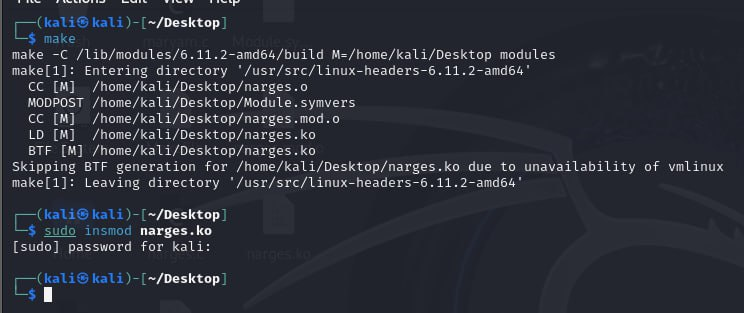


### **مراحل کامپایل و اجرای ماژول**

1. **ایجاد فایل Makefile**:
   * فایل Makefile را در کنار فایل کد ماژول (مثلاً narges.c) ذخیره کنید.
2. **کامپایل ماژول**:
   * با اجرای دستور زیر ماژول کامپایل شده و فایل narges.ko ایجاد می‌شود:

make

1. **بارگذاری ماژول**:
   * برای اضافه کردن ماژول به هسته لینوکس:
   * sudo insmod narges.ko
2. **مشاهده عملکرد ماژول**:
   * پس از بارگذاری، چراغ LED Caps Lock صفحه‌کلید شروع به چشمک زدن می‌کند.
3. **حذف ماژول**:
   * با دستور زیر ماژول از هسته حذف می‌شود:
   * sudo rmmod narges
4. **پاکسازی فایل‌ها**:
   * برای حذف فایل‌های موقت تولیدشده:
   * make clean



### **نتیجه اجرای ماژول**

پس از بارگذاری ماژول، چراغ LED Caps Lock صفحه‌کلید به صورت متناوب روشن و خاموش می‌شود. این فرآیند با استفاده از تایمرهای هسته و مدیریت دستگاه ورودی (input device) انجام می‌گیرد. با حذف ماژول، تایمر متوقف شده و LED در حالت خاموش باقی می‌ماند.

### **خلاصه کد، فرآیند اجرا و خروجی**

این کد یک ماژول هسته لینوکس است که با هدف چشمک زدن LED مربوط به Caps Lock در صفحه‌کلید طراحی شده است. پس از بارگذاری ماژول در هسته، چراغ LED Caps Lock با استفاده از تایمرهای هسته به صورت متناوب هر 500 میلی‌ثانیه روشن و خاموش می‌شود. فرآیند شامل مراحل زیر است:

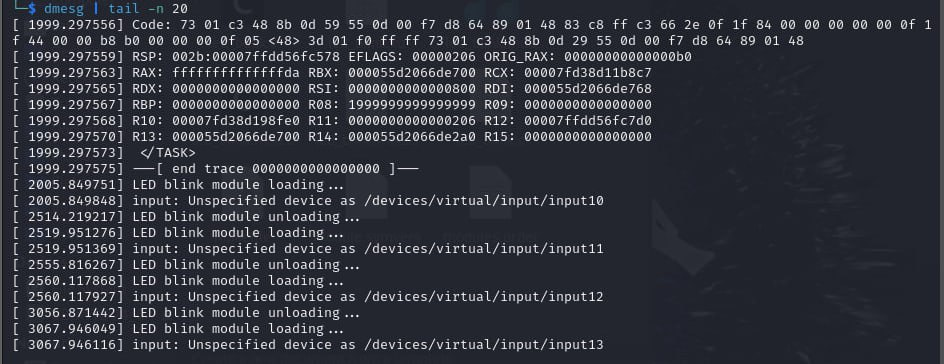
### **فرآیند اجرا**

1. **تعریف دستگاه ورودی**:
   * دستگاه ورودی (صفحه‌کلید) شبیه‌سازی شده و برای مدیریت LED تنظیم می‌شود.
2. **تنظیم تایمر**:
   * تایمری تعریف می‌شود که هر 500 میلی‌ثانیه تابعی را برای تغییر وضعیت LED فراخوانی می‌کند.
3. **تغییر وضعیت LED**:
   * در هر بار اجرای تایمر، وضعیت LED (روشن یا خاموش) تغییر می‌کند و تغییرات به صفحه‌کلید ارسال می‌شود.
4. **حذف ماژول**:
   * هنگام حذف ماژول، تایمر غیرفعال شده، دستگاه ورودی از سیستم لغو ثبت شده و حافظه آزاد می‌شود.

### **خروجی برنامه**

* چراغ LED Caps Lock صفحه‌کلید پس از بارگذاری ماژول، هر 500 میلی‌ثانیه چشمک می‌زند.
* هنگام حذف ماژول، LED در حالت خاموش باقی می‌ماند و تایمر متوقف می‌شود.

این ماژول نمونه‌ای کاربردی از مدیریت دستگاه‌های ورودی و استفاده از تایمرهای هسته در لینوکس است و به صورت پویا عملکرد دستگاه‌های سخت‌افزاری را کنترل می‌کند.



### **نتیجه‌گیری:**

1. پیاده‌سازی موفقیت‌آمیز ماژول نشان داد که می‌توان با استفاده از تایمرها و زیرسیستم ورودی، سخت‌افزار صفحه‌کلید را کنترل کرد.
2. ماژول به‌صورت پایدار اجرا شد و هیچ مشکلی در عملکرد سیستم ایجاد نکرد.
3. این پروژه درک بهتری از نحوه کار هسته لینوکس و تعامل آن با دستگاه‌های ورودی فراهم کرد.

### **منابع:**

1. مستندات رسمی لینوکس:
   * https://tldp.org/LDP/lkmpg/
2. مستندات مربوط به Linux Input Subsystem:
   * https://www.kernel.org/doc/html/latest/input/index.html
3. مستندات تایمرهای هسته لینوکس:
   * https://www.kernel.org/doc/html/latest/core-api/timer.html
4. پروژه نمونه:
   * <https://github.com/gagath/linux_keyblink>
5. کتاب‌ها و مقالات آموزشی:
   * Love, R. (2010). *Linux Kernel Development*. Addison-Wesley.