



دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

گزارش کار آزمایشگاه آزمایشگاه سیستم‌های عامل

گزارش آزمایش شماره ۹
(آشنایی با آشنایی با وقفه‌ها)

شماره‌ی گروه:	۲۰
گروه:	ارشیا یوسف‌نیا (۴۰۱۱۱۰۴۱۵)
استاد درس:	محمدعارف زارع زاده (۴۰۱۱۰۶۰۱۷)
تاریخ:	دکتر بیگی
	تابستان ۱۴۰۴

فهرست مطالب

۱	آزمایش ۱	۱
۱	۱.۱ توضیح کد	۱
۲	۲.۱ نحوه‌ی اجرای کد و نیازمندی‌ها	۲
۲	آزمایش ۲	۲
۲	۱.۲ توضیح کد	۲
۳	۲.۲ نیازمندی‌ها و نحوه‌ی اجرا	۳
۴	۳.۲ خروجی کد	۴

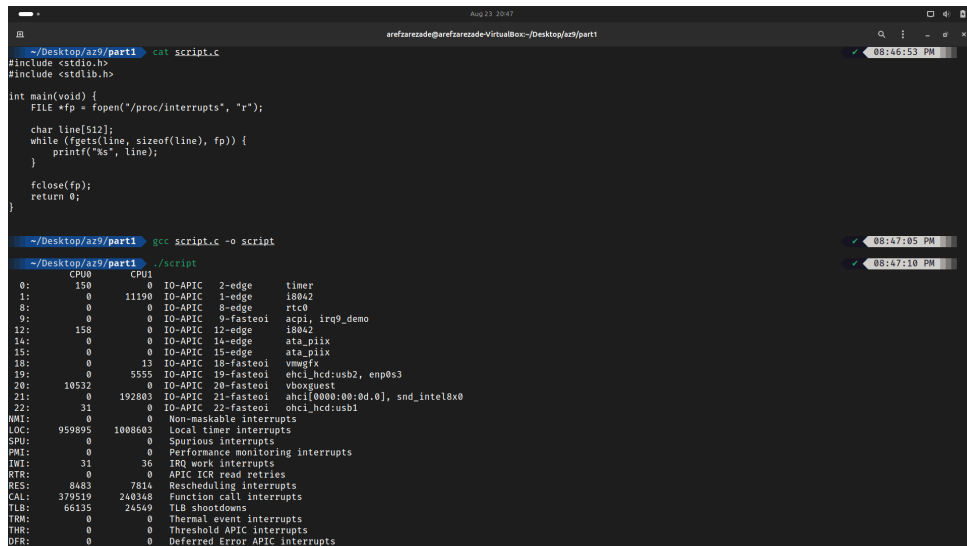
لیست تصاویر

۱	کد مشاهده‌ی تمام وقفه‌های سیستم عامل لینوکس	۱
۱	خروجی کد مشاهده‌ی تمام وقفه‌های سیستم عامل لینوکس	۲
۲	کد اضافه کردن handler به IRQ	۳
۳	کد اجرا کردن وقفه	۴
۳	فایل Makefile برای تست راحت تر	۵
۴	خروجی کد اضافه کردن وقفه	۶

۱ آزمایش

۱.۱ توضیح کد

کد این بخش را در آدرس source code/part1/script.c می‌توان مشاهده کرد. در ادامه آن کد را توضیح می‌دهیم. کد را در بالای شکل ۱ می‌توان مشاهده کرد.



```
~/Desktop/az9/part1 cat script.c
#include <stdio.h>
#include <stdlib.h>

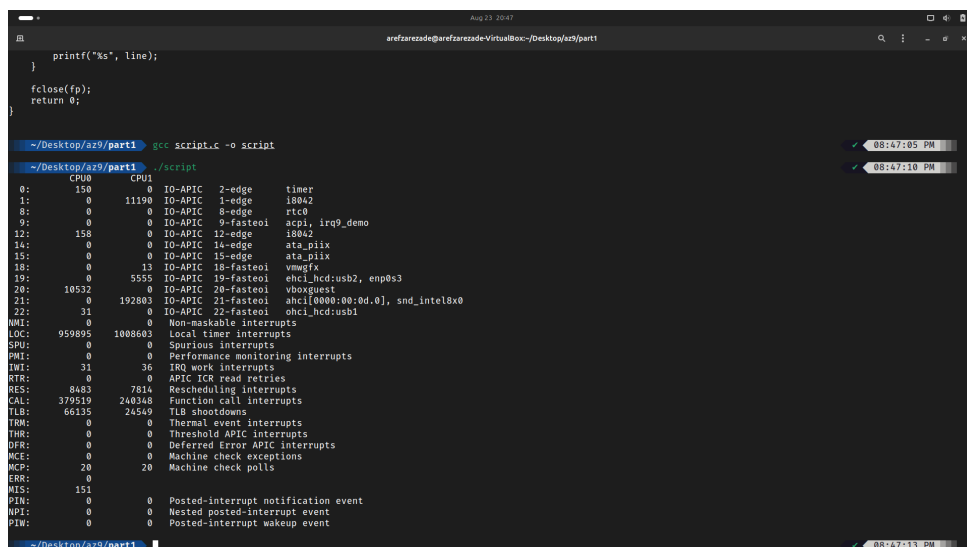
int main(void) {
    FILE *fp = fopen("/proc/interrupts", "r");
    char line[512];
    while (fgets(line, sizeof(line), fp)) {
        printf("%s", line);
    }
    fclose(fp);
    return 0;
}

~/Desktop/az9/part1 gcc script.c -o script

~/Desktop/az9/part1 ./script
0: 150 CPU0 0 IO-APIC 2-edge timer
1: 0 11190 IO-APIC 1-edge i8042
8: 0 0 IO-APIC 8-edge rtc0
9: 0 0 IO-APIC 9-fastest acpi, irq9_demo
12: 158 0 IO-APIC 12-edge i8042
14: 0 0 IO-APIC 14-edge ata_piix
15: 0 0 IO-APIC 15-edge ata_piix
18: 0 13 IO-APIC 18-fastest vmwgfx
19: 0 5555 IO-APIC 19-fastest ehci_hcd:usb2, enp0s3
20: 10532 0 IO-APIC 20-fastest vboxguest
21: 0 192803 IO-APIC 21-fastest ahci[0000:00:0d.0], snd_intel8x0
22: 31 0 IO-APIC 22-fastest ohci_hcd:usb1
NMI: 0 0 Non-maskable interrupts
LOC: 959895 1008603 Local timer interrupts
SPU: 0 0 Spurious interrupts
PMI: 0 0 Performance monitoring interrupts
IWI: 31 36 IRQ work interrupts
RTR: 0 0 APIC ICR read retries
RES: 8483 7814 Rescheduling interrupts
CAL: 379519 240348 Function call interrupts
TLB: 66135 24549 TLB shutdowns
TRM: 0 0 Thermal event interrupts
THM: 0 0 Threshold APIC interrupts
DFR: 0 0 Deferred Error APIC interrupts
MCE: 0 0 Machine check exceptions
MCP: 20 20 Machine check polls
ERR: 0 0
MIS: 151 0 Posted-interrupt notification event
PINS: 0 0 Nested posted-interrupt event
NPI: 0 0 Nested posted-interrupt event
PIW: 0 0 Posted-interrupt wakeup event
```

شکل ۱: کد مشاهده‌ی تمام وقفه‌های سیستم عامل لینوکس

طبق [۱] می‌دانیم لیست تمام وقفه‌های سیستم عامل لینوکس در فایل /proc/interrupts قابل مشاهده است. کد ما هم به این شیوه عمل می‌کند که ابتدا این فایل را خوانده و خروجی آن را خط به خط در یک رشته می‌ریزد. در نهایت آن را چاپ می‌کند.



```
~/Desktop/az9/part1 gcc script.c -o script

~/Desktop/az9/part1 ./script
0: 150 CPU0 0 IO-APIC 2-edge timer
1: 0 11190 IO-APIC 1-edge i8042
8: 0 0 IO-APIC 8-edge rtc0
9: 0 0 IO-APIC 9-fastest acpi, irq9_demo
12: 158 0 IO-APIC 12-edge i8042
14: 0 0 IO-APIC 14-edge ata_piix
15: 0 0 IO-APIC 15-edge ata_piix
18: 0 13 IO-APIC 18-fastest vmwgfx
19: 0 5555 IO-APIC 19-fastest ehci_hcd:usb2, enp0s3
20: 10532 0 IO-APIC 20-fastest vboxguest
21: 0 192803 IO-APIC 21-fastest ahci[0000:00:0d.0], snd_intel8x0
22: 31 0 IO-APIC 22-fastest ohci_hcd:usb1
NMI: 0 0 Non-maskable interrupts
LOC: 959895 1008603 Local timer interrupts
SPU: 0 0 Spurious interrupts
PMI: 0 0 Performance monitoring interrupts
IWI: 31 36 IRQ work interrupts
RTR: 0 0 APIC ICR read retries
RES: 8483 7814 Rescheduling interrupts
CAL: 379519 240348 Function call interrupts
TLB: 66135 24549 TLB shutdowns
TRM: 0 0 Thermal event interrupts
THM: 0 0 Threshold APIC interrupts
DFR: 0 0 Deferred Error APIC interrupts
MCE: 0 0 Machine check exceptions
MCP: 20 20 Machine check polls
ERR: 0 0
MIS: 151 0 Posted-interrupt notification event
PINS: 0 0 Nested posted-interrupt event
NPI: 0 0 Nested posted-interrupt event
PIW: 0 0 Posted-interrupt wakeup event
```

شکل ۲: خروجی کد مشاهده‌ی تمام وقفه‌های سیستم عامل لینوکس

۲.۱ نحوه‌ی اجرای کد و نیازمندی‌ها

اجرای آن نیاز به پیشنیاز خاصی ندارد. صرفاً کامپایلری مانند gcc برای کامپایل کد کافی است. البته می‌توان به صورت ساده‌تری و با دستور `cat /proc/interrupts` به همین خروجی رسید، اما طبق خواسته‌ی صورت آزمایش، کدی نوشتیم که این کار را انجام دهد.

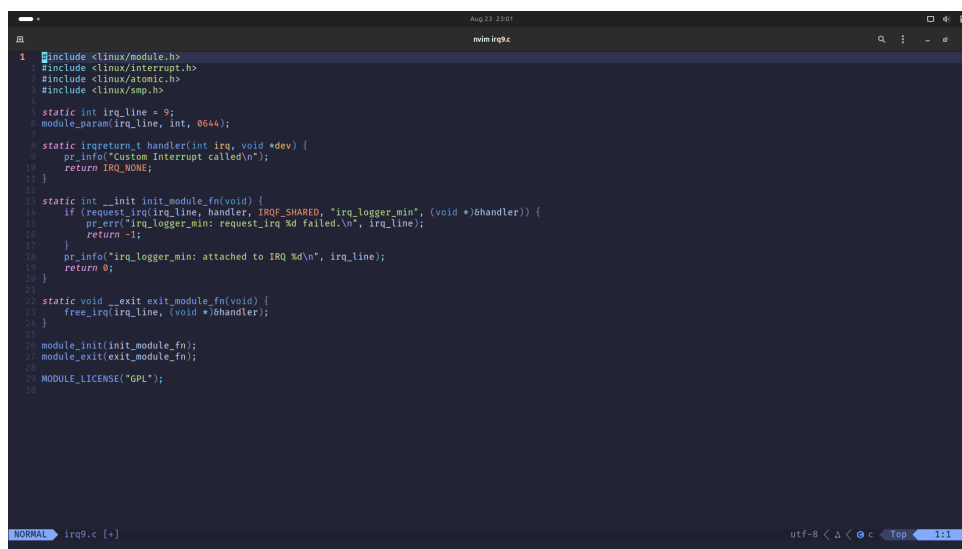
۲ آزمایش ۲

۱.۲ توضیح کد

برای اضافه کردن وقفه به سیستم عامل، به این صورت عمل می‌کنیم: در سیستم عامل لینوکس، برای اجرای وقفه‌ها، می‌توان از Interrupt Request یا IRQ استفاده کرد. اینها یک تابع handler دارند که هنگام اجرا شدن وقفه، اجرا می‌شوند.

طبق [۲] می‌دانیم که IRQ ۹ نخستین IRQ است که برای کاری رزرو نشده است، پس از آن استفاده می‌کنیم. اما مشکلی که وجود دارد، این است که این نوع از وقفه فقط توسط هسته قابل اجرا است، و کاربر امکان اجرای آن را ندارد. بنابراین نیاز است که یک ماژول هسته بنویسیم که آن وقفه را صدا می‌زند. پس مسیر کلی ما به این صورت است: ابتدا یک ماژول هسته می‌نویسیم که تابع handler را به IRQ ۹ اختصاص می‌دهد. سپس یک ماژول هسته‌ی دیگر می‌نویسیم که آن را اجرا می‌کند. در شکل ۳ می‌توان کد ماژول هسته‌ای که تابع handler را اضافه کرده و به IRQ ۹ اختصاص می‌دهد مشاهده کرد.

این کد ابتدا شماره‌ی IRQ و سطح دسترسی آن را تعیین می‌کند. سپس خود تابع handler را می‌توان مشاهده کرد، که صرفاً یک عبارت را چاپ می‌کند. سپس تابع init را داریم که درخواست می‌کند که تابع handler به آن شماره‌ی IRQ اختصاص یابد، و موفقیت یا عدم موفقیت را چاپ می‌کند. تابع exit هم صرفاً خروج را چاپ می‌کند.

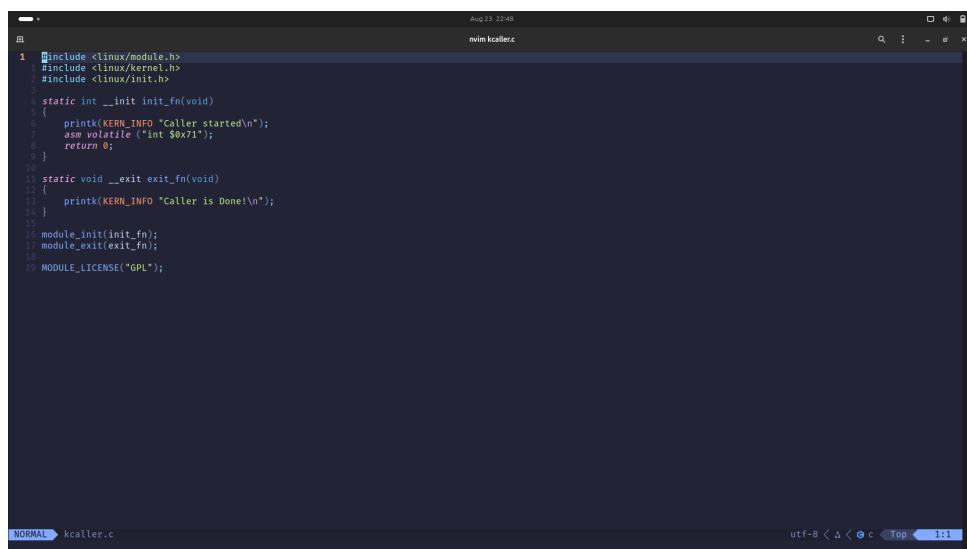


```
1 #include <linux/module.h>
2 #include <linux/interrupt.h>
3 #include <linux/atomic.h>
4 #include <linux/smp.h>
5
6 static int irq_line = 9;
7 module_param(irq_line, int, 0644);
8
9 static irqreturn_t handler(int irq, void *dev) {
10     pr_info("Custom Interrupt called\n");
11     return IRQ_NONE;
12 }
13
14 static int __init init_module_fn(void) {
15     if (request_irq(irq_line, handler, IRQF_SHARED, "irq_logger_min", (void *)handler)) {
16         pr_err("irq_logger_min: request_irq %d failed.\n", irq_line);
17         return -1;
18     }
19     pr_info("irq_logger_min: attached to IRQ %d\n", irq_line);
20     return 0;
21 }
22
23 static void __exit exit_module_fn(void) {
24     free_irq(irq_line, (void *)handler);
25 }
26
27 module_init(init_module_fn);
28 module_exit(exit_module_fn);
29 MODULE_LICENSE("GPL");
```

شکل ۳: کد اضافه کردن handler به IRQ

کد زیر، یک ماژول هسته است که از آن وقفه‌ی اضافه شده استفاده می‌کند. طبق [۳] می‌دانیم شماره‌ی بردار

وقفه‌ی آن 9 IRQ مساوی با 0x71 است. پس صرفا آن را اجرا می‌کنیم. آن را در شکل ۴ می‌توان مشاهده کرد.



```
1 #include <linux/module.h>
2 #include <linux/kernel.h>
3 #include <linux/init.h>
4
5 static int __init init_fn(void)
6 {
7     printk(KERN_INFO "Caller started\n");
8     asm volatile ("int $0x71");
9     return 0;
10 }
11
12 static void __exit exit_fn(void)
13 {
14     printk(KERN_INFO "Caller is Done!\n");
15 }
16
17 module_init(init_fn);
18 module_exit(exit_fn);
19
20 MODULE_LICENSE("GPL");
```

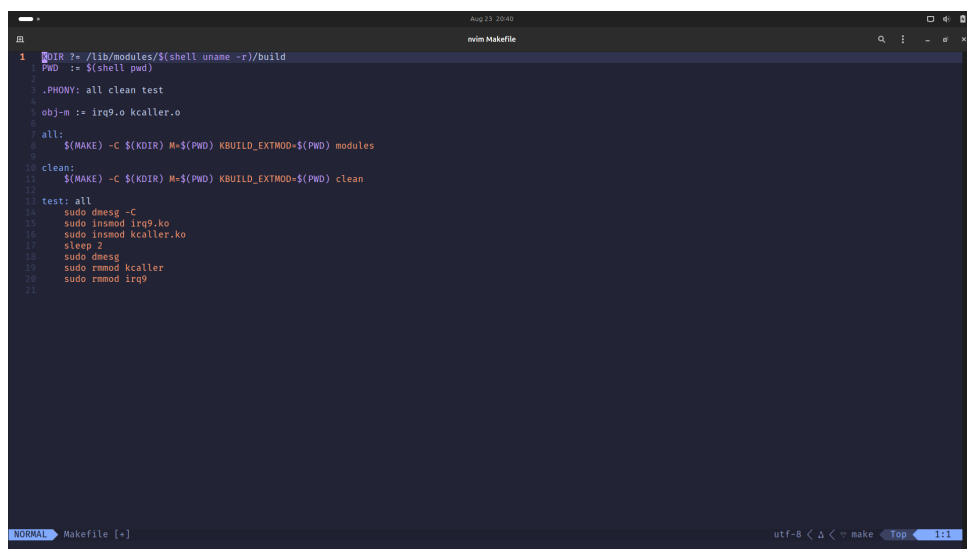
شکل ۴: کد اجرا کردن وقفه

۲.۲ نیازمندی‌ها و نحوه‌ی اجرا

تنها نیازمندی‌های لازم برای اجرای این کد، نیازمندی‌های استفاده از ماژول‌های هسته است که در آزمایش 8 به مفصل توضیح داده شد. در اینجا صرفا اشاره می‌کنیم که با اجرای دستور

`sudo apt install make build-essential linux-headers-`uname -r``

می‌توان نیازمندی‌های آن را نصب کرد. همچنین برای استفاده‌ی راحت تر، در Makefile مطابق شکل ۵ یک دستور `make test` اضافه کردیم که تست کردن را راحت تر کند. این دستور ابتدا ماژول‌ها را کامپایل و سپس به ترتیب مناسب اضافه می‌کند. سپس 2 ثانیه صبر می‌کند تا اجرا شوند و درنهایت به ترتیب مناسب آنها را خارج می‌کند.

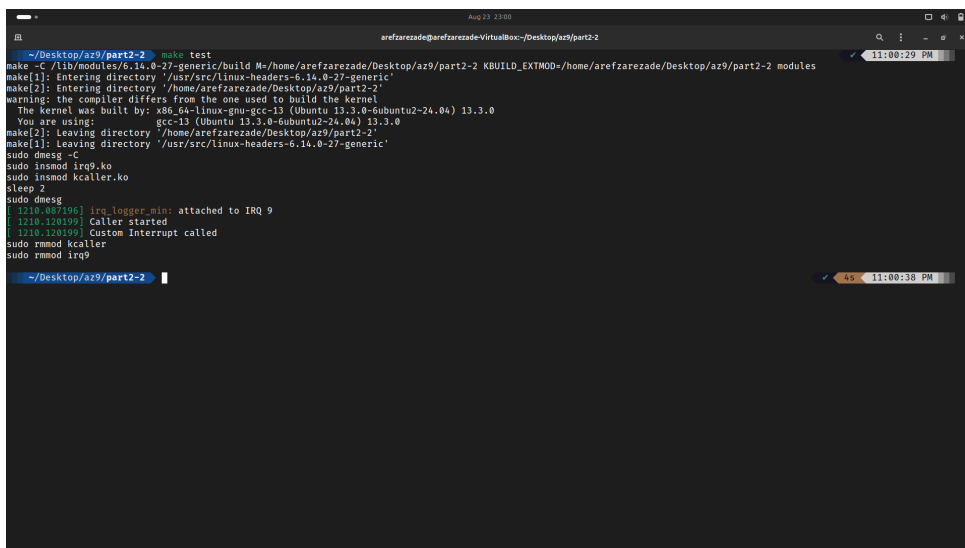


```
1 KDIR ?= /lib/modules/$(shell uname -r)/build
2 PWD := $(shell pwd)
3
4 .PHONY: all clean test
5
6 obj-m := irq9.o kcaller.o
7
8 all:
9     $(MAKE) -C $(KDIR) M=$(PWD) KBUILD_EXTMOD=$(PWD) modules
10
11 clean:
12     $(MAKE) -C $(KDIR) M=$(PWD) KBUILD_EXTMOD=$(PWD) clean
13
14 test: all
15     sudo dmesg -C
16     sudo insmod irq9.ko
17     sudo insmod kcaller.ko
18     sleep 2
19     sudo dmesg
20     sudo rmmod kcaller
21     sudo rmmod irq9
```

شکل ۵: فایل Makefile برای تست راحت تر

۳.۲ خروجی کد

در شکل ۶ می‌توان خروجی کد را مشاهده کرد، و می‌بینیم که کد به درستی کار می‌کند (زیرا عبارت Custom Interrupt called چاپ شده است).



```
arefzarezaade@arefzarezaade-VirtualBox: ~/Desktop/az9/part2-2
~/Desktop/az9/part2-2 make test
make -C /lib/modules/6.14.0-27-generic/build M=/home/arefzarezaade/Desktop/az9/part2-2 KBUILD_EXTMOD=/home/arefzarezaade/Desktop/az9/part2-2 modules
make[1]: Entering directory '/usr/src/linux-headers-6.14.0-27-generic'
make[2]: Entering directory '/home/arefzarezaade/Desktop/az9/part2-2'
warning: the compiler differs from the one used to build the kernel
The kernel was built by: x86_64-linux-gnu-gcc-13 (Ubuntu 13.3.0-6ubuntu2-24.04) 13.3.0
You are using: gcc-13 (Ubuntu 13.3.0-6ubuntu2-24.04) 13.3.0
make[2]: Leaving directory '/home/arefzarezaade/Desktop/az9/part2-2'
make[1]: Leaving directory '/usr/src/linux-headers-6.14.0-27-generic'
sudo dmesg -C
sudo insmod irq9.ko
sudo insmod kcaller.ko
sleep 2
sudo dmesg
[ 3210.007196] irq_logger_min: attached to IRQ 9
[ 3210.120199] Caller started
[ 3210.120199] Custom Interrupt called
sudo rmmod kcaller
sudo rmmod irq9
~/Desktop/az9/part2-2
```

شکل ۶: خروجی کد اضافه کردن وقفه

مراجع

- [١] Linux man-pages project. *proc_interrupts - Linux manual page*. Accessed: 2025-08-23. 2023. URL: https://man7.org/linux/man-pages/man5/proc_interrupts.5.html.
- [٢] Webopedia. *What Are IRQ Numbers?* Accessed: 2025-08-23 2010. URL: <https://www.webopedia.com/reference/irqnumbers/>.
- [٣] OSDev Wiki. *8259 PIC*. Accessed: 2025-08-23. 2023. URL: http://wiki.osdev.org/8259_PIC.