Dinamikus megszakításkérés-kiosztás illesztőprogramokhoz

A megszakítás az az eszköz, amellyel a hardver felhívja magára a szoftver figyelmét. Vizsgáljuk meg a működését!

Egy számítógép a vele szemben támasztott elvárásoknak csak akkor tud megfelelni, ha külső eszközökkel is képes kapcsolatba lépni. Az eszközök és a processzor közötti kapcsolattartáshoz az átjárót a megszakítások biztosítják. A megszakításkérési vonalak eszközökhöz való hozzárendelése és a megszakítások kezelésének módja kulcsszerepet játszik az illesztőprogramok fejlesztésében. Mivel a megszakításkérési vonalak száma korlátozott, nagyobb számú eszközhöz csak a megszakítások megosztásával biztosíthatunk hozzáférést. Ugyanakkor, ha megpróbálunk kiosztani egy már használatban lévő megszakítást, a rendszer összeomlik. Írásomban a megszakításokkal és a megszakításkezeléssel kapcsolatos alapismereteket szeretném átadni, valamint be szeretnék mutatni egy karakteres eszközökhöz használható ***megszakításkérés-kiosztó (IRQ)*** megoldást.

Minden eszköz célja valamilyen hasznos feladat ellátása, és ennek teljesítéséhez kapcsolatot kell tartania a processzorral. Ha a processzor közölni szeretne valamit az egyik eszközzel, akkor a megfelelő eszközvezérlőnek küldi el utasításait. Az eszközvezérlő irányítja az eszköz működését. Hasonlóan, ha az eszköz válaszolni szeretne a processzornak, mert például a kért adatok készen állnak a beolvasásra, akkor egy megszakítás létrehozásával vonja magára a processzor figyelmét. A megszakítás egy hardveres, a processzor és az eszköz közötti kommunikációt lehetővé tévő megoldás.

A ***Linux*** egészen a 2.6-os változatig nem preemptív működésű volt, ami azt jelenti, hogy ha egy folyamat rendszermag módban fut, és a futtatásra kész folyamatok várólistájára felkerül egy nagyobb fontosságú folyamat, akkor a kisebb fontosságú folyamattól nem lehet elvenni a futás jogát, amíg vissza nem tér felhasználó módba. A megszakítások azonban akkor is elvonhatják a processzor figyelmét, amikor éppen egy rendszermag módban futó folyamattal foglalkozik. Ezzel a megoldással növelni lehet a rendszer átviteli sebességét. Amikor megszakítás érkezik, a processzor felfüggeszti az éppen futó folyamatot, majd a megszakítást okozó esemény jellegétől függően lefuttatja a megfelelő kódot.

A számítógép minden eszköze rendelkezik egy eszközvezérlővel, valamint egy olyan hardveres érintkezővel, amelyet akkor használ jelzésre, amikor szüksége van a processzor közreműködésére. Ez az érintkező a processzor megfelelő megszakításlábához csatlakozik, ezen keresztül folyik az adatcsere. A processzor vezérlőhöz csatlakozó lábát megszakításkérési vonalnak nevezzük. Egy processzor több ilyen lábbal is rendelkezik, vagyis több eszközzel is képes tartani a kapcsolatot. A korszerű operációs rendszereknél egy ***programozható megszakításvezérlő (programmable interrupt controller, PIC)*** kezeli a processzor és az eszközvezérlők közötti ***IRQ***-vonalakat. Az egyes rendszerekben a szabad ***IRQ***-k száma korlátozott, ám a ***Linux*** rendelkezik egy olyan megoldással, amelynek révén több eszköz is osztozhat ugyanazokon a megszakításokon.

A megszakítások használata a programozók munkájához hasonlítható. A programozó megnyitja a postaládáját, majd nekikezd szokásos munkájának. Amikor új levele érkezik, akkor rövid hangjelzést hall, valamint a képernyő alján valamilyen egyéb jelzés is megjelenik. Azonnal elmenti a programot, amin éppen dolgozik, majd átvált a postaládára. Elolvassa a levelet, küld egy visszaigazolást, majd folytatja imént abbahagyott munkáját. Az általa végrehajtott lépések részletes listáját valamikor később küldi el. Hasonlóan, amikor egy processzor futtat egy folyamatot, az eszközök az egyes feladatokhoz kapcsolódóan megszakításokat küldhetnek neki, például jelezve, hogy valamilyen adat készen áll az átvitelre. Amikor befut egy megszakítás, a processzor azonnal elmenti a programszámláló aktuális értékét a rendszermag módú verembe, majd végrehajtja a megfelelő ***megszakításkiszolgáló eljárást (interrupt service routine, ISR)***. Az ***ISR*** egy függvény a rendszermagban, feladata a megszakítás jellegének meghatározása és a szükséges művelet végrehajtása, például egy adatblokk másolása a merevlemezről a központi memóriába. Az ***ISR*** lefuttatása után a processzor visszaállítja a korábbi folyamatot, és folytatja a futtatását.

Az illesztőprogram egy a rendszermagba beépülő, az alkalmazások kéréseit fogadó programmodul. Amikor egy alkalmazás adatokat szeretne olvasni egy eszközről, akkor azonnal meghívásra kerül a megfelelő illesztőprogram, majd megnyílik olvasásra a kívánt eszköz. Ha a rendszer sokat vár a lassú eszközökre, akkor nem tud a feladataival foglalkozni. A rendszermag fejlesztőinek egyik fő célja az erőforrások minél jobb kihasználásának biztosítása. Mivel nem akar a hardvereszközök által szállított adatokra várakozni,

a rendszermag kiadja a feladatot az eszközvezérlőnek, majd visszatér a felfüggesztett folyamathoz. Amikor az olvasás befejeződött, az eszköz egy megszakítás segítségével értesíti a processzort, amely végrehajtja a megfelelő *ISR*-t.

A megszakítások osztályozása

A megszakításokat két fő kategóriára osztjuk, szinkron és aszinkron megszakításokra. A szinkron megszakításokat a processzor vezérlőegysége hozza létre valamilyen utasítás végrehajtásakor. A vezérlőegység akkor adja ki a megszakítást, amikor végzett az utasítások végrehajtásával, innen származik a szinkron jelző. Az aszinkron megszakításokat a hardvereszközök hozzák létre, véletlenszerűen, bár a processzoróra állásához igazodva. Az *Intel* alapú gépeknél az első típust kivételnek, a másodikat pedig megszakításnak nevezzük. A megszakításokat egy előjel nélküli, egybájtos egész azonosítja, ezt vektornak nevezzük. A vektortartomány a 0 - 255. Az első 32 (0 - 31 sorszámú) vektor a kivételeket, más néven nem maszkolható megszakításokat azonosítja, ezekről *„Linuxos jelzések alkalmazásfejlesztői szemszögből”* (*Linuxvilág*, 2003. július) című cikkemben volt szó. A 32 - 47 tartomány a maszkolható megszakításoké, ezeket hozzák létre az *IRQ*-k (a 0 - 15 számú *IRQ*-vonalak).
Az utolsó tartomány a 48 - 255, ebbe a szoftveres megszakítások tartoznak. Ilyen például a rendszerhívások megvalósítására használt 128-as megszakítás (*int 0X80 assembly* utasítások).

IRQ-kiosztás

A rendszer már használatban lévő megszakításairól a */proc* könyvtárban találunk pillanatképet. A `$cat /proc/interrupt` parancs a megszakításokkal kapcsolatos adatokat jeleníti meg. A saját gépemen a következő volt a kimenete:

```
CPU0
  0:   82821789          XT-PIC  timer
  1:        122          XT-PIC  i8042
  2:          0          XT-PIC  cascade
  8:          1          XT-PIC  rtc
 10:     154190          XT-PIC  eth0
 12:        100          XT-PIC  i8042
 14:      21578          XT-PIC  ide0
 15:         18          XT-PIC  ide1
NMI:          0
ERR:          0
```

Az első oszlopban az *IRQ*-vonalak száma (32 - 47 vektortartomány) látható, a másodikban pedig az, hogy az egyes megszakítások a rendszer elindítása óta hányszor jutottak el a processzorhoz. A harmadik oszlop a *PIC*-kel kapcsolatos, az utolsó pedig azoknak az eszközöknek a nevét tartalmazza, amelyek kezelőt jegyeztek be az egyes megszakításokhoz.
Ha egy illesztőprogramot dinamikusan akarunk betölteni, akkor először használaton kívüli *IRQ*-vonalat kell találnunk a rendszerben. A `request_irq` függvény segítségével meghatározott azonosítójú *IRQ*-vonalat rendelhetünk az eszközünkhöz. A `request_irq` szintaxisának meghatározása a *linux/sched.h* fájlban található:

```
int
request_irq (unsigned int irq,
             void (*handler) (int, void *,
                              struct pt_regs *),
             unsigned long flags,
             const char *device, void *dev_id);
```

A függvény átadott értékeinek szerepe a következő:

- `unsigned int irq`: A rendszertől kért megszakítás száma.
- `void (*handler) (int, void *, struct pt_regs *)`: Megszakítás kérésekor gondoskodnunk kell a kezelését végző *ISR*-ről, egyébként a processzor egyszerűen nyugtázza, majd nem történik semmi. Az átadott érték a kezelőfüggvényre mutat. A kezelőfüggvény szintaxisa a következő:

  ```
  void
  handler (int irq, void *dev_id,
           struct pt_regs *regs);
  ```

  Az első átadott érték az *IRQ* száma, erről a `request_irq` függvény kapcsán már volt szó. A második átadott érték egy fő- és egy alrészből álló eszközazonosító, ez adja meg az aktuális megszakítási esemény kezelését végző eszközt. A harmadik átadott értéket a folyamat környezetének a rendszermag verembe mentésére használjuk, mielőtt a processzor megkezdené a megszakításkezelő függvény futtatását. A korábbi folyamat visszaállításakor a rendszer ezt az adatszerkezetet használja fel. Normál esetben az illesztőprogramok íróinak ezzel az átadott értékkel nem kell foglalkozniuk.
- `unsigned long flags`: A `flags` változó a megszakításkezelésben jut szerephez. Az `SA_INTERRUPT` jelzőt gyors megszakításkezelőknél használjuk, ez minden maszkolható megszakítást letilt. Az `SA_SHIRQ`-t akkor használjuk, ha az *IRQ*-t egynél több eszköz közt akarjuk megosztani; az `SA_PROBE` jelzőt akkor, ha az *IRQ*-vonal segítségével le szeretnénk kérdezni egy hardvereszközt; az `SA_RANDOM` jelző pedig a rendszermag véletlenszám-generátorának kezdeti értékadására használható. A jelzőről további részleteket a */usr/src/linux/drivers/char/random.c* fájlban találni.
- `constant char *device`: Az IRQ-t használó eszköz neve.
- `void *dev_id`: Az eszköz azonosítója, egy mutató az eszköz adatszerkezetére. Ha a megszakítás meg van osztva, akkor ez a mező meghatározott eszközre mutat.

A `request_irq` függvény siker esetén nullával, a foglalás sikertelenségekor pedig `-EBUSY` értékkel tér vissza. Az `EBUSY` a 16-os számú hiba, leírása a */usr/src/linux/include/asm/errno.h* fájlban található. A `free_irq` függvény elengedi az adott eszköz *IRQ*-ját. Szintaxisa a következő:

```
free_irq (unsigned int irq, void *dev_id);
```

Az argumentumok jelentése a fentiekkel egyezik meg.
ISR meghívására megszakítás fellépésekor kerül sor. A megszakítás hatására végrehajtandó műveletek leírása az *ISR*-ben szerepel. A rendszermag fenntart a memóriában egy a megszakítási eljárások címeit (megszakításvektorokat)

1. kódrészlet my_module.c

```
#include <linux/init.h>
#include <linux/fs.h>
#include <linux/module.h>
#include <linux/sched.h>
#include <linux/interrupt.h>
static struct file_operations fops;
static int Major, irq = 7;
static void OurISR (int irq, void *device,
                    struct pt_regs *regs)
{
  /* fontos és azonnali, időkritikus
  ➥feladatok */
}
static int __init my_init_module(void)
{
    int status;
    Major = register_chrdev(0, “OurDevice”,
    ➥&fops);
    if (Major == -1) {
        printk (“ A dinamikus főazonosító “
                “foglalása sikertelen.\n”);
        return Major;
    }
    status = request_irq(irq,
                         (void *)OurISR,
                         SA_INTERRUPT,
                         “OurDevice”, &fops);
    if (status == -EBUSY) {
        printk (“Az IRQ-azonosító foglalása
        ➥sikertelen.\n”);
        unregister_chrdev(Major, “OurDevice”);
        return status;
    }

    printk (“A modul sikeresen betöltve.\n”);
    printk (“Az eszköz főazonosítója:   %d\n”,
            Major);
    printk (“Az eszköz IRQ-ja:     %d\n”,
            irq);
    return 0;
}
static void __exit my_cleanup_module (void)
{
    printk(“A(z) %d főazonosító és a(z) %d
    ➥IRQ-azonosító “
            “elengedése megtörtént.\n”, Major,
            ➥irq);
    free_irq(irq, &fops);
    unregister_chrdev(Major, “OurDevice”);
    printk(“A modul sikeresen eltávolítva.\n”);
}
module_init (my_init_module);
module_exit (my_cleanup_module);
MODULE_LICENSE(“GPL”);
```

tartalmazó táblázatot. Amikor egy megszakítás érkezik, a processzor lekérdezi az ***ISR*** címét a megszakításvektorok táblázatából, majd lefuttatja. Az ***ISR*** feladata, hogy a megszakítás jellegétől – mint például írás vagy olvasás – függően válaszoljon az eszköznek. Általában az ***ISR*** alvó folyamatokat ébreszt fel az eszközön, ha a megszakítás olyan eseményt jelez, amire éppen várnak.

Azt az időt, amit a processzor egy-egy megszakítás kezeléséhez igényel, megszakítási lappangási időnek nevezzük. A megszakítási lappangási idő része a hardveres jelterjedési idő, a regiszterek elmentésének ideje és a szoftveres terjedési idő. A rendszer teljesítményének javításához a megszakítási lappangási időt a lehető legkisebbre kell csökkenteni, ezért az ***ISR***-nek rövidnek kell lennie, és a megszakításokat csak rövid időre tilthatja le. A megszakítások letiltása alatt is jelentkezhetnek további megszakítások, ám a processzor figyelmen kívül hagyja őket, amíg a megszakítások újraengedélyezése meg nem történik. Ha egynél több megszakítás van blokkolva, a processzor fontossági sorrendben engedélyezi őket, amikor késszé válik a kezelésükre.

Az illesztőprogramok készítőinek az illesztőprogramok kódjában csak akkor szabad letiltaniuk a megszakításokat, ha valóban szükséges, mert ilyenkor a rendszer nem frissíti az időzítőket, nem továbbítja a hálózati csomagokat a pufferek felé és felől, és így tovább. Az ***ISR***-eket úgy kell megírni, hogy más folyamatok számára is biztosítsák a processzor elérhetőségét. A való világban viszont az ***ISR***-ek hosszas feladatokat kezelnek. Ilyenkor az ***ISR*** csak az időkritikus adatcseréket végezheti el a hardverrel, és a ***taskletet*** kell használnia a tényleges adatátvitel lebonyolítására. A ***tasklet*** a legújabb ***Linux*** rendszermag egy különleges szolgáltatása, amely a megszakításokkal kapcsolatos műveletek egy részének megfelelő időben való elvégzésére alkalmas. A ***tasklet*** a szoftveres megszakítás, és más megszakítások által megszakítható. A megszakítások belső világáról ***Bovet*** és ***Cesati*** írásából (lásd az internetes forrásokat) lehet tájékozódni, a megszakítások megvalósítását – az illesztőprogramok nézőpontjából szemlélve – pedig ***Rubini*** és ***Corbet*** tárgyalja.

Egy egyszerű megvalósítás

A rendszermagmodulok illesztőprogramokat tartalmaznak, ezeket a meglévő rendszermaghoz, akár a rendszer működése közben lehet betölteni. A dinamikus ***IRQ***-kiosztást az ***1. kódrészletben*** szereplő egyszerű modullal szeretném szemléltetni. Egy egyszerű karakteres eszközhöz készült illesztőprogramról van szó, az eszközt nevezzük ***OurDevice***-nak (***SajátEszköz***), ehhez foglalunk dinamikusan ***IRQ***-vonalat. A modul beillesztésekor az init_module függvény fut le. Ha a foglalás sikeres, akkor kiírjuk az eszköz főazonosítóját és a kapott ***IRQ*** számát. Ettől a pillanattól az ***IRQ*** kiosztását a ***/proc*** könyvtárban is ellenőrizhetjük. A kapott ***IRQ***-t a modul eltávolításakor elengedjük. ***IRQ***-azonosítót a kódon belül legjobban egy nyílt belépési ponton lehet bejegyezni, amely a megfelelő pillanatban az elengedésről is gondoskodik.

A ***my_module.c*** fájlt a 2.6.0-0.test2.1.29 rendszermag alatt fordítottam le. A ***kernel-2.6.0-0.test2.1.30.i586.rpm*** fájlt az összes szükséges ***RPM***-el együtt kell letölteni és telepíteni. Az ***RPM***-et a ***people.redhat.com/arjanv/2.5/RPMS.kernel*** címen érhetjük el, az illesztőprogram fordítását pedig a következőképpen kell elvégeznünk:

```
gcc -Wall -O3 -finline-functions \
-Wstrict-prototypes -falign-functions=4 \
-I/lib/modules/2.6.0-0.test2.1.29/build/include \
-I/lib/modules/2.6.0-0.test2.1.29/build/include/
?asm/mach-default
-I./include -D__KERNEL__ -DMODULE -DEXPORT_SYMTAB \
-DKBUILD_MODNAME=my_module -c my_module.c -o \
my_module.o
```

A *my_module.o* betöltése után – ha a főazonosító és az *IRQ* foglalása sikeres – a megfelelő üzenetek jelennek meg. Ha az *IRQ*-azonosítót egy másik eszköz már használta, akkor a rendszermag törli az eszköz bejegyzését, és felszabadítja a főazonosítót. A `$cat /proc/interrupt` parancs a következő kimenetet jeleníti meg:

```
           CPU0
  0:   82887219          XT-PIC  timer
  1:        122          XT-PIC  i8042
  2:          0          XT-PIC  cascade
  7:          0          XT-PIC  OurDevice
  8:          1          XT-PIC  rtc
 10:     154769          XT-PIC  eth0
 12:        100          XT-PIC  i8042
 14:      21636          XT-PIC  ide0
 15:         18          XT-PIC  ide1
NMI:          0
ERR:          0
```

A kimenetben látható az `OurDevice`-hoz tartozó bejegyzés, illetve az eszköz *IRQ*-ja. Amikor a modult eltávolítjuk, a rendszermag felszabadítja az *IRQ*-azonosítót és a főazonosítót, valamint törli a készülék bejegyzését.

Összefoglalás

Remélem, hogy sikerült tisztáznom a megszakításokkal és a kezelésükkel kapcsolatos alapokat. A `request_irq` és a `free_irq` függvényt elsősorban illesztőprogramok készítésekor érdemes ismerni, a dinamikus *IRQ*-foglalási eljárást pedig egy egyszerű karakteres eszköz illesztőprogramjával szemléltettük.

Köszönetnyilvánítás

Köszönettel tartozom *C. Surestnek* a cikk előkészítéséhez nyújtott segítségéért.

Linux Journal 2005. április, 132. szám

A cikk forrásai: ➲ *www.linuxjournal.com/article/8064*

Dr B. Thangaraju (bt_raju@vsnl.net) fizikából szerzett PhD fokozatot, és öt évig kutató munkatársként dolgozott az Indiai Tudományos Intézetnél. Jelenleg műszaki vezető Indiában, a Wipro Technologies Talent Transformation, Embedded Systems Focus csoportjánál. Munkája során a Linux belső világával, a rendszermaggal, illesztőprogramokkal, valamint beágyazott és valós idejű Linuxokkal foglalkozik.