Az USB-soros illesztőprogram-réteg (2. rész)

Hogyan készíthetsz olyan USB-s eszközt, ami az általános USB-soros illesztőprogrammal használható? Olvass tovább, és hamarosan megtudod!

Első részében írásunknak (Linuxvilág, 2003. március) bemutattam az USB-soros réteget, illetve azokat az alapvető műveleteket, amelyekkel egy illesztőprogram a rétegnél bejegyezhető. Ez alkalommal arról lesz szó, hogy az adatok miként mozognak a rétegen keresztül, illetve hogyan jelennek meg az USB-soros eszközök `sysfs` alatt.

Általános USB-soros eszközök

Az első részben említettem, hogy az általános USB-illesztőprogram segítségével könnyedén, egyedi rendszermagszintű programozás nélkül is munkára foghatunk egy USB-s eszközt. Sajnos arról elfelejtettem szót ejteni, hogy pontosan milyen módon is kell ezt véghezvinni – szerencsére az olvasóktól érkezett kérdések hamar egyértelművé tették: nem sumákolhatok.
Ha az általános USB-soros illesztőprogrammal használható USB-s eszközt szeretnél létrehozni, mindössze két tömeges adatmozgatásra alkalmas USB-végpontra van szükséged, egy ki- és egy bemenetre. Az általános USB-soros illesztőprogram ezt a két végpontot egyetlen tty-eszközzé fogja össze, amiről olvasni, illetve amire írni lehet a felhasználói területről. Például egy eszköz végpontjainak a leírói a */proc/bus/usb/devices* alatt találhatók, ez maga mégis egykapus eszközként kezelhető. Csatlakoztatáskor az alábbi rendszermagüzeneteket láthatjuk:

```
Generic converter detected
(A rendszer ÆltalÆnos Ætalak t t ismert fel.)
Generic converter now attached to ttyUSB0
(Az ÆltalÆnos Ætalak t  a ttyUSB0 csom ponton
csatlakozik.) (devfs hasznÆlatakor usb/tts/0)
```

Ezt követően a */dev/ttyUSB0* csomóponton keresztül bármely felhasználó adatokat küldhet az eszköznek.
Ha egy eszköz egynél több tömeges adatmozgatásra szolgáló ki- és bemenetpárral rendelkezik, akkor a rendszer több kaput is rendel hozzá. Például: egy eszköz végpontjainak leírói a */proc/bus/usb/devices* alatt találhatók, az eszköz pedig kétkapusként jelenik meg. Csatlakoztatáskor az alábbi rendszermagüzenetek láthatók:

```
Generic converter detected
(A rendszer ÆltalÆnos Ætalak t t ismert fel.)
Generic converter now attached to ttyUSB0
(Az ÆltalÆnos Ætalak t  a ttyUSB0 csom ponton
csatlakozik.) (devfs hasznÆlatakor usb/tts/0)
Generic converter now attached to ttyUSB1
(Az ÆltalÆnos Ætalak t  a ttyUSB1 ponton
csatlakozik.) (devfs hasznÆlatakor usb/tts/1)
```

Ebben az esetben a */dev/ttyUSB0* és a */dev/ttyUSB1* egyaránt használható adatcserére.
A végpontok sorrendje lényegtelen, akár azt is megtehetjük, hogy először felsoroljuk az összes bemenetet, majd a kimeneteket vesszük sorra. (Az előző példában váltakozva szerepeltek.)

Az USB-soros mag az összes ki- és bemeneti végpontot kezelni fogja, és az imént látott módon fogja őket párosítani. Minden tömeges adatmozgatásra szolgáló párhoz egy-egy megszakításvégpontot is rendel, ha létezik ilyen, de a megszakításvégpontot az általános illesztőprogram nem fogja használni, azt csak a rendszermag USB-soros illesztőprogramjai vehetik igénybe.
Ha az általános USB-soros illesztőprogramot hozzá szeretnénk kötni az eszközhöz, az eszköz USB-gyártóazonosítóját (USB vendor ID) és termékazonosítóját (product ID) betöltéskor átadott értékként kell az `usbserial` modul tudomására hozni. Ha például az imént látott eszközt `ffff` gyártóazonosítóval és `fff8` termékazonosítóval szeretnénk kötésbe hozni, az alábbi parancsot kell kiadni:

```
modprobe usbserial vendor=0xffff
product=0xfff8
```

Ha a felhasználótól nem várhatjuk el, hogy a megfelelő eszközazonosítóval betöltse az `usbserial` modult, vagy ha az USB-soros illesztőprogrammal egynél több eszközazonosítót is használni akarunk, akkor egy aprócska illesztőprogram írása a megoldás. Erre listánk (➲ http://www.linuxvilag.hu/USB) szolgáltat példát. Ebben az illesztőprogramban nincsenek visszahívó függvények, mindössze a vezérelni kívánt eszközök gyártó- és termékazonosítóit találjuk benne. Ennek alapján az `usb_serial_device_type` adatszerkezet megadása a lista 1-es szakaszában látható.
A megfelelő gyártó- és termékazonosítókat az `id_table` mutatóban kell felsorolni (a lista 2-es szakasza).
Az illesztőprogram mindössze két függvényt – egy két- és egy háromsorost –, valamint három változómegadást tartalmaz. Segítségével a megadott eszközökre vonatkozóan az általános USB-soros illesztőprogram összes szolgáltatását igénybe vehetjük. Az illesztőprogram betöltése önműködően lezajlik, amikor az eszközt csatlakoztatjuk a számítógéphez – azért ez sem rossz dolog. Alighanem ez az egyik legkisebb működő linuxos rendszermag-illesztőprogram. Fordítása az alábbi paranccsal történik:

```
echo "obj-m := tiny_tiny_usbserial.o"
➥> Makefile
make -C <a/rendszermag/elØrØsi/œtja>
➥ SUBDIRS=$PWD modules
```

A Windows USB OPOS-soros illesztőprogramon keresztül a Windows operációs rendszer is támogatja az ilyen jellegű, az eszköz számára virtuális „COM”-kapukat létrehozó eszközfelületet. Az eszközök fejlesztői, gyártói így olyan USB-s termékeket dobhatnak piacra, amelyekhez – szerencsés módon – nem kell külön linuxos és windowsos illesztőprogramot írni.

Az USB-soros eszközök életciklusa

Amikor egy USB-soros eszközt csatlakoztatunk a számítógéphez, különféle műveletek hosszú sora szükséges hozzá, hogy

Az „aprócska” USB-soros illesztőprogram

```
#include <linux/config.h>

#ifdef CONFIG_USB_SERIAL_DEBUG
       static int debug = 1;
       #define DEBUG
#else
       static int debug;
#endif

#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/tty.h>
#include <linux/module.h>
#include <linux/usb.h>
#include <../drivers/usb/serial/usb-serial.h>

#define TERM KAZONOS˝T     0xfff0
#define ESZK ZAZONOS˝T     0xfff0

/* 2-es szakasz */
static struct usb_device_id id_table [] = {
       {USB_DEVICE(TERM KAZONOS˝T ,
                  ➥ESZK ZAZONOS˝T )},
       { }    /* A bejegyzØs lezÆrÆsa */
};
/* 2-es szakasz vØge */

/*
 * az illesztıprogram  nmßk dı bet ltØsØnek
 * lehetıvØ tØtele
 */
MODULE_DEVICE_TABLE (usb, id_table);

static struct usb_driver tiny_driver = {
   .name =             "tiny",
   .probe =            usb_serial_probe,
   .disconnect =       usb_serial_disconnect,
   .id_table =         id_table,
};

/* 1-es szakasz */
/* Minden, a Tiny eszk z Æltal igØnyel adat */
static struct usb_serial_device_type  *
               ➥tiny_device = {
   .owner =            THIS_MODULE,
   .name =             "Tiny USB serial",
   .short_name =       "tiny",
   .id_table =         id_table,
   .num_interrupt_in = NUM_DONT_CARE,
   .num_bulk_in =      NUM_DONT_CARE,
   .num_bulk_out =     NUM_DONT_CARE,
   .num_ports =        1,
};
/* 1-es szakasz vØge */

static int __init tiny_init (void)
{
       usb_serial_register (&tiny_device);
       usb_register (&tiny_driver);
       return 0;
}

static void __exit tiny_exit (void)
{
       usb_deregister (&tiny_driver);
       usb_serial_deregister (&tiny_device);
}

module_init(tiny_init);
module_exit(tiny_exit);

MODULE_LICENSE("GPL");
```

adott USB-soros illesztőprogram egy tty-eszközt vezérelhessen. Ezek a műveletek a következők:

- Az USB-elosztó (hub) illesztőprogramja felismeri az új eszközt. Hozzárendel egy USB-azonosítót, kiolvassa belőle a fontosabb USB-leírókat, majd ezeket egy `usb_device` adatszerkezetbe helyezi, és kellő számú, a teljes USB-eszközt képviselő `usb_interfaces` adatszerkezettel párosítja.
- Az USB-mag gondozásába veszi az eszközt, és az USB-felületeket bejegyzi a rendszermag illesztőprogrammagjánál.
- A rendszermag illesztőprogrammagja végignézi a már bejegyzett USB-illesztőprogramok listáját, és megvizsgálja, hogy ezek valamelyike kezelni tudja-e az eszközt.
- Mivel USB-soros eszközről van szó, az USB-soros mag átveszi az eszköz vezérlését a rendszermag illesztőprogrammagjától.
- Az USB-soros mag `usb_serial` névvel egy adatszerkezetet hoz létre, majd ezzel az adatszerkezettel meghívja a megfelelő USB-soros illesztőprogram `probe()` függvényét.
- Az USB-soros illesztőprogram `probe()` függvénye – ha szükséges – üzembe helyezi (inicializálja) az eszközt, majd a vezérlést visszaadja az USB-soros magnak.
- Az USB-soros mag létrehozza az `usb_serial_port` adatszerkezeteket – az adott eszközön lévő soros kapuk számától függően –, majd meghívja az USB-soros illesztőprogram `attach()` függvényét, ha az rendelkezik ilyennel.
- Az `attach()` függvény visszatérését követően kerül sor az egyes `usb_serial_port` adatszerkezetek bejegyzésére a rendszermag illesztőprogrammagjánál.
- A rendszermag illesztőprogrammagja minden egyes kapuhoz egy visszahívást hajt végre az USB-soros mag felé.
- Az USB-soros mag meghívja a kapu USB-soros illesztőprogramjának egyedi `port_probe()` függvényét – ha van ilyen –, majd bejegyzi a kaput a tty-rétegnél, befejezve ezzel az üzembe helyezés folyamatát.

A folyamat lejátszódását követően a tty-eszközcsomópont az egyedi USB-soros kapuhoz kötődik. Amikor valamelyik felhasználó megnyitja az eszközcsomópontot , a rendszermagban az alábbi lépésekre kerül sor:

- A rendszermag megvizsgálja az eszközcsomópontot, és megállapítja, hogy azt a tty-réteg jegyezte be, tehát meghívja a tty-réteg `open` függvényét.
- A tty-réteg megvizsgálja az eszközt, és megállapítja, hogy az USB-soros mag jegyezte be nála a csomópontot, tehát a

drivers/usb/serial/usb-serial.c fájl `serial_open()` függvényét hívja meg.
- A `serial_open()` függvény meghatározza, hogy melyik egyedi USB-soros illesztőprogram van ehhez a csomóponthoz bejegyezve.
- A megadott USB-soros illesztőprogram modulszámlálóját a rendszer növeli, ezzel előzi meg a modul eltávolítását, miközben egy felhasználó esetleg éppen használja az eszközt.
- Ha a megadott USB-soros illesztőprogramnak van `open()` függvénye, akkor sor kerül ennek a meghívására, és átadott értékként a kapu `usb_serial_port` adatszerkezetét kapja meg.
- Az USB-soros illesztőprogram ekkor bármely szükséges, kifejezetten az eszközre jellemző megnyitó jellegű műveletet el tud végezni, és el tudja küldeni az eszköztől érkező adatok fogadásának megkezdéséhez szükséges URB-ket (USB-kérésblokkokat).

Ha egy felhasználó az eszközcsomópontra a `write()` függvényt hívja meg, vagyis a megadott soros kapura adatokat akar küldeni, a rendszermag az alábbi lépéseket hajtja végre:
- A rendszermag meghívja a tty-mag `tty_write()` függvényét. Korábban – az open híváskor – ezt a mutatót már beállította, így nem kell újra előkeresnie.
- A `tty_write()` meghívja a vonalszabályozó (line discipline) `write()` függvényét a megadott tty-eszközre.
- A vonalszabályozó meghívja az USB-soros mag `serial_write()` függvényét.
- A `serial_write()` függvény meghatározza, hogy pontosan melyik USB-soros illesztőprogramot használja ez a fájl, majd annak `write()` függvényét hívja meg.
- Az USB-soros illesztőprogram ekkor megkezdheti az adatoknak egy átmeneti tárba történő másolását, illetve az USB-kapcsolaton keresztül az eszköznek való elküldésüket, közben gondoskodik az eszköz által esetlegesen támasztott különleges formázási elvárások teljesítéséről.
- Miután az adatok küldése befejeződött, az illesztőprogram felébreszti a tty-eszközt, hogy az küldje el neki az esetlegesen átmeneti tárban tárolt adatokat. Mindez egy egyszerű hívással történik:

```
schedule_work(&port->work);
```

Ha adott kapu USB-soros illesztőprogramja adatokat kap, akkor a kapu átmeneti tárához (flip puffer) rendelt tty-adatszerkezetbe kell helyeznie őket:

```
for (i = 0; i < data_size; ++i) {
    if (tty->flip.count >= TTY_FLIPBUF_SIZE)
        tty_flip_buffer_push(tty);
    tty_insert_flip_char(tty, data[i], 0);
}
tty_flip_buffer_push(tty);
```

Amikor egy felhasználó az eszközcsomóponton `read()` hívást hajt végre, a kapu `tty flip` átmeneti tárában található adatokat kapja meg.
Ha a felhasználó lezárja az eszközcsomópontot, a rendszermag az alábbi lépéseket hajtja végre:
- A rendszermag meghívja a tty-mag `tty_release()` függvényét.
- A `tty_release()` meghatározza, hogy ez volt-e az utolsó hivatkozás erre az eszközcsomópontra. (Ne feledjük, egy eszközcsomópontot egyszerre több program is megnyithat.) Ha igen, az USB-soros mag `serial_close()` függvényének meghívására is sor kerül.
- A `serial_close()` függvény meghívja az USB-soros illesztőprogram `close()` függvényét, így lezárulnak a függőben lévő USB-átvitelek, az illesztőprogram pedig készenléti állapotba kerül.
- Az USB-soros mag csökkenti az USB-soros illesztőprogram modulszámlálóját, ami akár el is távolítható.

Az USB-soros eszközök megjelenése sysfs alatt

Az USB-soros eszköz USB-soros illesztőprogramhoz való kötésének fenti leírásában többször is sor került a rendszermag illesztőprogrammagjának meghívására. Erre azért van szükség, mert az USB-soros mag külön buszként jelenik meg a rendszermag illesztőprogram-modelljében – így egyetlen USB-eszközön több kapu is lehet.
Például az alábbi egy nyolckapus USB-soros eszköz, ami a rendszer első USB-buszán található. Sysfs alatt az elérési útja a következő: */sys/devices/pci0/00:09.0/usb1/1-1/1-1.1*. Ezen a könyvtáron belül az alábbi könyvtárakat és fájlokat találjuk: ***1-1.1:0/, bcdDevice, bConfigurationValue, bDeviceClass, bDeviceProtocol, bDeviceSubClass, bmAttributes, bMaxPower, bNumConfigurations, bNumInterfaces, idProduct, idVendor, manufacturer, name, power, product, serial, speed, ttyUSB0/, ttyUSB1/, ttyUSB2/, ttyUSB3/, ttyUSB4/, ttyUSB5/, ttyUSB6/*** és ***ttyUSB7/***.
A könyvtár állományaiból minden, az eszközzel kapcsolatos adatot megtudhatunk, ahogy az ***1-1.1:0/*** könyvtár állományaiból is, ez egyébként az első kapcsolódási felület az eszköz felé. A `ttyUSB*` könyvtárakat az USB-soros mag hozza létre, és az alábbi fájlokat tartalmazzák: ***dev***, ***name*** és ***power***.
A ***dev*** fájl az adott eszköz al- és főszámát tartalmazza; ha párbeszédet akarunk folytatni az eszközzel, ennek alapján található meg pontosan a megfelelő eszközcsomópont. A */sys/bus/usb* könyvtár szerint ez az USB-eszköz az `io_edgeport` USB-illesztőprogramhoz kötődik. Van egy `usb-serial` busz is, ami a rendszermagnál bejegyzett egyes USB-soros kapukat mutatja. Mivel ezek a kapuk tty-eszközök, a tty-osztály könyvtárában is megjelennek.
Mivel a hivatkozások mindegyike ugyanahhoz az USB-eszközhöz vezet, az USB-eszköz típusa, tty-kapuinak a száma és a vezérléséhez szükséges USB-soros illesztőprogram típusa könnyedén meghatározható. Így sokkal több adathoz juthatunk hozzá, mint amit a */proc/tty/driver/usb-serial* fájlban találtunk, lásd írásom első részében.
A `sysfs` felülettel kapcsolatban most szűkszavú leszek, ám érdemes tudni, hogy rengeteg adatot szolgáltat a rendszerben pillanatnyilag jelen lévő fizikai és logikai eszközökről. A `sysfs`-nek és a rendszermag illesztőprogram-modelljének bővebb ismertetését ***Pat Mochel*** idei ***linux.conf.au*** írásában, a ➲ http://www.kernel.org/pub/linux/kernel/people/mochel/doc/lca címen találod meg.

Linux Journal 2003. április, 108. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux USB és a PCI Hot Plug rendszermag felelőse. Az IBM-nél dolgozik, ahol számos, a Linux rendszermagjával kapcsolatos kérdéssel foglalkozik.