I2C illesztőprogramok (1. rész)

Az I2C-busz segítségével figyelemmel követhetjük számítógépünk egészségi állapotát. Tekintsük át, hogyan írhatunk egy minden szükséges állapotadatot begyűjtő illesztőprogramot!

A Linuxvilág 2003. júniusi és augusztusi számában a Linux-rendszermag illesztőprogram-modelljével foglalkoztam, példaként pedig az I2C alrendszert hoztam fel. Írásomban az I2C alrendszer feladataival és a hozzá való illesztőprogramok írásával fogok foglalkozni.
Az I2C egy, két vezetéket használó soros buszprotokoll neve, ami eredetileg a Philips fejlesztése. Beágyazott rendszerekben gyakran használják a különféle összetevők közötti kapcsolattartásra, a PC-s alaplapok pedig az érzékelőlapkákkal való párbeszéd céljából alkalmazzák. Az érzékelők általában a ventilátorok fordulatszámát, a processzor hőmérsékletét és más alkatrészek egyéb jellemzőit mérik és jelentik. Bizonyos RAM-lapkák is ilyen protokollon keresztül továbbítják a DIMM-modullal kapcsolatos adatokat az operációs rendszernek.
Az I2C rendszermagkód életének jelentős részét a fő rendszermagfából kimaradva töltötte, fejlesztése egyébként a 2.0-s rendszermag idején kezdődött. A 2.4-es rendszermag már tartalmaz valamennyi I2C-támogatást, elsősorban bizonyos megjelenítő-illesztőprogramokhoz. A 2.6-os rendszermaggal az I2C-kód jelentős része bekerült a fő rendszermagfába, azoknak rendszermag-fejlesztőknek köszönhetően, akik az adatcsere-felületeket a rendszermag-fejlesztői közösség számára elfogadhatóbbakká alakították át. Néhány illesztőprogram még most is csak a külső CVS-fában található meg, a fő ➲ http://kernel.org fában nem, de csupán idő kérdése, hogy ezeknek az átültetése is megtörténjen.
Az I2C rendszermagkód logikailag több részre bontható, amelyek a következők: I2C-mag, I2C buszillesztőprogramok, I2C algoritmus-illesztőprogramok és I2C lapka-illesztőprogramok. Írásomban az I2C-mag működésével nem foglalkozom, inkább a busz- és az algoritmus-illesztőprogramok írására összpontosítok. Az I2C lapka-illesztőprogramok írásának témájára a második részben kerül sor.

I2C buszillesztőprogramok

Az I2C buszillesztőprogramok leírása az `i2c_adapter` adatszerkezetben található meg, ennek megadása az ***include/linux/i2c.h*** fájlban szerepel. A buszillesztőprogramnak csak a következő mezőknek kell értéket adnia:

- `struct module *owner` – értéke (`THIS_MODULE`) segítségével a modulra való hivatkozások számlálása oldható meg.
- `unsigned int class` – az illesztőprogram által támogatott I2C osztályeszközök. Értéke általában I2C_ADAP_CLASS_SMBUS.
- `struct i2c_algorithm *algo` – mutató az i2c_algorithm adatszerkezetre, amely az I2C buszvezérlőn folyó adatcsere mikéntjét írja le. Az adatszerkezetről bővebb leírást is fogok adni.
- `char name[I2C_NAME_SIZE]` – az I2C buszillesztőprogram beszédes neve. Az itt megadott érték az I2C-csatolóhoz tartozó `sysfs` fájlnévben jelenik meg.

Az alábbi kódrészlet – amely az `i2c_adapter` adatszerkezet értékadását szemlélteti – egy ***tiny_i2c_adap.c*** nevű I2C-csatoló példa-illesztőprogramból származik. Az illesztőprogram az 57. CD Magazin/I2C könyvtárában található.

```
static struct i2c_adapter tiny_adapter = {
    .owner   = THIS_MODULE,
    .class   = I2C_ADAP_CLASS_SMBUS,
    .algo    = &tiny_algorithm,
    .name    = "tiny adapter",
};
```

Az I2C-csatoló bejegyzését az illesztőprogram az `i2c_add_adapter` függvény meghívásával végzi el, amelynek egy, az `i2c_adapter` adatszerkezetre hivatkozó mutatót ad át:

```
retval = i2c_add_adapter(&tiny_adapter);
```

Ha az I2C-csatoló olyan típusú eszköz felett üzemel, amelyhez `device` adatszerkezet van hozzárendelve – ilyenek például a PCI- vagy USB-eszközök –, akkor az `i2c_add_adapter` meghívása előtt a csatolóeszköz szülőjének a mutatóját erre az eszközre kell állítani.
A ***drivers/i2c/busses/i2c-piix4.c*** illesztőprogramban a mutató beállítása például a következőképpen történik:

```
/* sysfs csatolás beállítása a szülőeszközre */
piix4_adapter.dev.parent = &dev->dev;
```

Ha a mutató értékét nem adjuk meg, akkor az I2C-csatoló

1. lista A tiny_access függvény

```
static s32 tiny_access(struct i2c_adapter *adap,
                       u16 addr,
                       unsigned short flags,
                       char read_write,
                       u8 command,
                       int size,
                       union i2c_smbus_data *data)
{
 int i, len;

 dev_info(&adap->dev, "%s hívására a következő
 ➥átadott értékekkel "
        "került sor:\n",
        __FUNCTION__);
 dev_info(&adap->dev, "cím = %.4x\n", addr);
 dev_info(&adap->dev, "kapcsolók = %.4x\n",
flags);
 dev_info(&adap->dev, "olvasas_iras = %s\n",
          read_write == I2C_SMBUS_WRITE ?
          "iras" : "olvasas");
 dev_info(&adap->dev, "parancs = %d\n",
          command);

  switch (size) {
 case I2C_SMBUS_PROC_CALL:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_PROC_CALL\n");
    break;
 case I2C_SMBUS_QUICK:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_QUICK\n");
    break;
 case I2C_SMBUS_BYTE:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_BYTE\n");
    break;
 case I2C_SMBUS_BYTE_DATA:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_BYTE_DATA\n");
    if (read_write == I2C_SMBUS_WRITE)
        dev_info(&adap->dev,
                 "adat = %.2x\n", data->byte);
    break;
 case I2C_SMBUS_WORD_DATA:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_WORD_DATA\n");
    if (read_write == I2C_SMBUS_WRITE)
        dev_info(&adap->dev,
                 "adat = %.4x\n", data->word);
    break;
 case I2C_SMBUS_BLOCK_DATA:
    dev_info(&adap->dev,
             "méret = I2C_SMBUS_BLOCK_DATA\n");
    if (read_write == I2C_SMBUS_WRITE) {
        dev_info(&adap->dev, "adat = %.4x\n",
                 data->word);
        len = data->block[0];
        if (len < 0)
            len = 0;
        if (len > 32)
            len = 32;
        for (i = 1; i <= len; i++)
            dev_info(&adap->dev,
                     "adat->blokk[%d] = %x\n",
                     i, data->block[i]);
    }
    break;
 }
 return 0;
}
```

a `legacy` (örökölt) buszra kerül, és a `sysfs` fában a */sys/devices/legacy* elérési út alatt látszik. Nézzük, hogy mi történik példa-illesztőprogramunkkal bejegyzéskor:

```
$ tree /sys/devices/legacy/
/sys/devices/legacy/
|-- detach_state
|-- floppy0
|   |-- detach_state
|   `-- power
|       `-- state
|-- i2c-0
|   |-- detach_state
|   |-- name
|   `-- power
|       `-- state
`-- power
    `-- state
```

Mint a rendszermag illesztőprogram-modelljével foglalkozó korábbi írásokból kiderül, az I2C-csatoló a */sys/class/i2c-adapter* könyvtárban is megjelenik:

```
$ tree /sys/class/i2c-adapter/
/sys/class/i2c-adapter/
`-- i2c-0
    |-- device -> ../../../devices/legacy/i2c-0
    `-- driver ->
../../../bus/i2c/drivers/i2c_adapter
```

Az I2C-csatoló bejegyzésének törlését az illesztőprogram az `i2c_del_adapter` függvény – átadott értéke egy, az `i2c_adapter` adatszerkezetre hivatkozó mutató – meghívásával végezheti el:

```
i2c_del_adapter(&tiny_adapter);
```

Az I2C algoritmus-illesztőprogramok

Az I2C-algoritmust az I2C buszillesztőprogram használja az I2C-busszal való kapcsolattartásra. A legtöbb I2C buszillesztőprogram saját I2C-algoritmust ad meg és használ. Az algoritmus jellege szorosan összefügg a buszillesztőprogram

és az adott eszköz közti párbeszéd módjával. Bizonyos I2C buszillesztőprogram-osztályokhoz már több I2C algoritmus-illesztőprogram is készült; példaként a ***drivers/i2c/i2c-algo-ite.c*** fájl által kezelt ITE csatolókat, a ***drivers/i2c/i2c-algo-ibm_ocp.c által*** vezérelt IBM PPC 405 csatolókat és a ***drivers/i2c/i2c-algo-bit.c*** fájlban található általános I2C bit-eltoló algoritmust említhetném. Mindegyik algoritmus saját függvényekkel bír, használatukhoz az I2C buszillesztőprogram oldaláról külön bejegyzés szükséges. Minderről a rendszermagfa ***drivers/i2c/i2c-algo-*.c*** fájljaiban találhatunk bővebb tudnivalókat.

Példa-illesztőprogramunkhoz saját I2C algoritmus-illesztőprogramot fogunk készíteni. Az algoritmus-illesztőprogram megadása az `i2c_algorithm` adatszerkezetben, az ***include/linux/i2c.h*** fájlban található. A gyakran használt mezők közül íme néhánynak a leírása:

- `char name[32];` – az algoritmus neve.
- `unsigned int id;` – az adatszerkezet által megadott algoritmus típusának a leírása. A különféle típusok megadása az ***include/linux/i2c-id.h*** fájlban található. Az összes típus neve az `I2C_ALGO_` karakterlánccal kezdődik.
- `int (*master_xfer)(struct i2c_adapter *adap,struct i2c_msg msgs[], int num);` – függvénymutató, értéket akkor szükséges neki adni, ha az adott algoritmus-illesztőprogram közvetlen I2C-elérést is végezhet. Ha az értékét megadjuk, akkor mindig ennek a függvénynek a meghívására kerül sor, ha egy I2C lapka-illesztőprogram kapcsolatba szeretne lépni a lapkával. Ha az értéke `NULL`, akkor ezt a szerepet az `smbus_xfer` függvény veszi át.
- `int (*smbus_xfer) (struct i2c_adapter *adap, u16 addr, unsigned short flags, char read_write, u8 command, int size, union i2c_smbus_data *data);` – függvénymutató, akkor kell neki értéket adni, amikor az algoritmus-illesztőprogram hozzáférhet az SMB-buszhoz. A legtöbb PCI alapú I2C-buszillesztőprogram képes erre, tehát esetükben ennek a függvénymutatónak értéket kell adnunk. Ha ezt megtesszük, akkor mindig ennek a függvénynek a meghívására kerül sor, amennyiben egy I2C lapka-illesztőprogram kapcsolatba szeretne lépni a lapkával. Ha az értéke `NULL`, akkor ezt a szerepet a `master_xfer` függvény veszi át.
- `u32 (*functionality) (struct i2c_adapter *);` – függvénymutató, a megadott függvényt az I2C-mag hívja meg annak meghatározásához, hogy az I2C-csatoló illesztőprogram milyen jellegű írásokra és olvasásokra képes.

Az I2C-csatoló példa-illesztőprogramunkban az `i2c_adapter` adatszerkezet a `tiny_algorithm` változóra hivatkozott. Ennek az adatszerkezetnek a megadása a következő:

```
static struct i2c_algorithm tiny_algorithm = {
    .name           = "tiny algorithm",
    .id             = I2C_ALGO_SMBUS,
    .smbus_xfer     = tiny_access,
    .functionality  = tiny_func,
};
```

A `tiny_func` függvény egészen kisméretű, a feladata mindössze az, hogy az I2C-magot tájékoztassa az algoritmus által támogatott I2C üzenettípusokról. Ettől az illesztőprogramtól az alábbi néhány I2C üzenettípus támogatását várjuk el:

```
static u32 tiny_func(struct i2c_adapter *adapter)
{
    return I2C_FUNC_SMBUS_QUICK |
           I2C_FUNC_SMBUS_BYTE |
           I2C_FUNC_SMBUS_BYTE_DATA |
           I2C_FUNC_SMBUS_WORD_DATA |
           I2C_FUNC_SMBUS_BLOCK_DATA;
}
```

Az összes I2C üzenettípus az ***include/linux/i2c.h*** fájlban van megadva, és az `I2C_FUNC_` karakterlánccal kezdődik.

A `tiny_access` függvény meghívására akkor kerül sor, amikor egy I2C ügyfél-illesztőprogram az I2C-busszal szeretne párbeszédet kezdeményezni. Példafüggvényünk meglehetősen egyszerű: mindössze naplózza az I2C lapka-illesztőprogram kéréseit a `syslog`-ba, valamint a hívó felé jelenti a sikeres végrehajtást. A napló alapján minden különböző cím- és adattípus összesíthető, amelyet egy I2C lapka-illesztőprogram valaha is igényelhet. A példafüggvény megvalósítása az *1. listán* látható.

2. lista Az „lm75” I2C ügyfél-illesztőprogram

```
$ modprobe lm75
$ tree /sys/bus/i2c/
/sys/bus/i2c/
|-- devices
|   |-- 0-0048 -> ../../../devices/legacy/i2c-0/0-0048
|   |-- 0-0049 -> ../../../devices/legacy/i2c-0/0-0049
|   |-- 0-004a -> ../../../devices/legacy/i2c-0/0-004a
|   |-- 0-004b -> ../../../devices/legacy/i2c-0/0-004b
|   |-- 0-004c -> ../../../devices/legacy/i2c-0/0-004c
|   |-- 0-004d -> ../../../devices/legacy/i2c-0/0-004d
|   |-- 0-004e -> ../../../devices/legacy/i2c-0/0-004e
|   `-- 0-004f -> ../../../devices/legacy/i2c-0/0-004f
`-- drivers
    |-- i2c_adapter
    `-- lm75
        |-- 0-0048 -> ../../../../devices/legacy/i2c-0/0-0048
        |-- 0-0049 -> ../../../../devices/legacy/i2c-0/0-0049
        |-- 0-004a -> ../../../../devices/legacy/i2c-0/0-004a
        |-- 0-004b -> ../../../../devices/legacy/i2c-0/0-004b
        |-- 0-004c -> ../../../../devices/legacy/i2c-0/0-004c
        |-- 0-004d -> ../../../../devices/legacy/i2c-0/0-004d
        |-- 0-004e -> ../../../../devices/legacy/i2c-0/0-004e
        `-- 0-004f -> ../../../../devices/legacy/i2c-0/0-004f
```

Elkészült a tiny_i2c_adap illesztőprogram, lefordítottuk, betöltöttük – de mire jó? A válasz prózai lesz: önmagában semmire. Az I2C buszillesztőprogramnak I2C ügyfél-illesztőprogramra van szüksége ahhoz, hogy bármi értelmeset tudjon tenni, azon kívül, hogy elücsörög a sysfs fában.
Az lm75 I2C ügyfél-illesztőprogram betöltéskor tehát a tiny_i2c_adap illesztőprogram segítségével próbálja megtalálni azt a lapkát, amelyhez őt írták (lásd a 2. *listát*).
Mivel a tiny_i2c_adap illesztőprogram minden olvasási és írási kérésre sikert jelezve válaszol, az A tiny_access függvény lm75 I2C lapka-illesztőprogram azt hiszi, hogy az lm75 lapka lehetséges elérési címeinek mindegyikén talált egy ilyen lapkát. A bőséges címválaszték az oka annak, hogy a 0-0048 – 0-004f tartományban több I2C-eszköz is létrejött.
Ha benézünk valamelyik eszköz könyvtárába, akkor a lapka-illesztőprogram érzékelőfájljait láthatjuk:

```
$ tree /sys/devices/legacy/i2c-0/0-0048/
/sys/devices/legacy/i2c-0/0-0048/
|-- detach_state
|-- name
|-- power
|   `-- state
|-- temp_input
|-- temp_max
`-- temp_min
```

A ***detach_state*** fájlt és a ***power*** könyvtárat a rendszermag-illesztőprogram magja hozza létre, mindkettő az energiakezelésben jut szerephez. Ismétlem, ezeket nem az lm75 illesztőprogram hívja életre. Tekintsük át a könyvtár további állományainak a feladatát!
Ha az lm75 illesztőprogramból lekérdezzük a temp_max pillanatnyi értékét, a következőt kapjuk:

```
$ cat /sys/devices/legacy/i2c-0/0-0048/temp_max
1000
```

Az érték lekérdezéséhez az lm75 illesztőprogram az I2C-busz bizonyos címeiről végzett olvasásra kérte a tiny_i2c_adap illesztőprogramot. A kérést a syslog alapján lehet követni (lásd a 3. listát az 57. CD Magazin/IRC könyvtárában).
A napló alapján kiderül, hogy a tiny_access függvény meghívása három alkalommal történt meg. Az első parancs egy word típusú adatot akart kiolvasni a 0048-as című eszköz 0 számú regiszteréből. A második és a harmadik olvasás ugyanezen eszköz 3-as és 2-es regiszterére irányult.
A parancsok a ***drivers/i2c/chips/lm75.c*** fájlban található lm75_update_client függvény kódjának a következő részeivel társíthatók:

```
data->temp_input = lm75_read_value(client,
                        LM75_REG_TEMP);
data->temp_max = lm75_read_value(client,
                         LM75_REG_TEMP_OS);
data->temp_hyst = lm75_read_value(client,
                       LM75_REG_TEMP_HYST);
```

Ugyanezen fájl lm75_read_valu*e függvénye az alábbi kódot tartalmazza:

```
/* Minden regiszter word méretű, kivéve a
beállítóregisztert. Az LM75 a felsőbb helyi értékű
bájtot helyezi előre, pontosan fordítva ahhoz
képest, ahogy azt megszoktuk. */
static int lm75_read_value(struct i2c_client
                          *client, u8 reg)
{
  if (reg == LM75_REG_CONF)
       return i2c_smbus_read_byte_data(client,
                                reg);
  else
       return swap_bytes(
             i2c_smbus_read_word_data(client,
                                      reg));
}
```

Amikor tehát az lm75 illesztőprogram ki akarja olvasni a legnagyobb hőmérsékleti értéket, a regiszter számával meghívja az lm75_read_value függvényt, amely viszont az I2C-mag i2c_smbus_read_word_data függvényének adja át a szót. Az I2C-magnak ez a függvénye megkeresi, hogy az ügyféleszköz melyik I2C-buszon található, majd az átvitel végrehajtása érdekében az ehhez az I2C-buszhoz társított algoritmust hívja meg. A tiny_i2c_adap illesztőprogramunk tehát ezzel a módszerrel kapja meg az átvitel elvégzésére vonatkozó kérést.
Ha ugyanebbe a ***sysfs*** fájlba írni szeretnénk, az lm75 illesztőprogram ugyanilyen eljárással kéri meg a tiny_i2c_adap illesztőprogramot az I2C-busz megadott címére végzendő adatírásra. A syslog-ban a kérés adatai is megjelennek:

```
$ echo 300 > /sys/devices/legacy/i2c-0/
➥0-0048/temp_max
$ dmesg
i2c_adapter i2c-0: tiny_access hívására
➥a következő átadott értékekkel került sor:
i2c_adapter i2c-0: cím = 0048
i2c_adapter i2c-0: kapcsolók = 0000
i2c_adapter i2c-0: olvasas_iras = iras
i2c_adapter i2c-0: parancs = 3
i2c_adapter i2c-0: méret = I2C_SMBUS_WORD_DATA
i2c_adapter i2c-0: adat = 8000
```

Összegzés

Ez alkalommal összefoglaltuk az I2C illesztőprogram-alrendszer alapvető jellemzőit, valamint áttekintettük, hogyan lehet egyszerű I2C busz- és I2C algoritmus-illesztőprogramot írni, amely bármely meglévő I2C ügyfél-illesztőprogrammal együttműködik. A teljes illesztőprogram ***dmn-09-i2c-adap.c*** néven az 57. CD Magazin/I2C könyvtárában található.
Az I2C lapkaillesztőprogramok írásának témájával a második részben foglalkozunk.

Linux Journal 2003. december, 116. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux-rendszerek USB és gyors csatlakoztatású (PCI Hot Plug) egységei rendszermagba épített meghajtóprogramjainak a fejlesztője. Az IBM-nél dolgozik és rendszermaggal kapcsolatos kérdésekkel foglalkozik.