I2C illesztőprogramok (2. rész)

Lássuk, hogyan tudjuk leolvasni a gép hőmérsékletét, a ventilátorok fordulatszámát és az egyéb fontos jellemzőket mérő érzékelők jeleit!

Sorozatunk első részében az I2C buszillesztőprogramok és az I2C algoritmus-illesztőprogram működését tárgyaltuk, valamint egy egyszerű I2C buszillesztőprogramot is összeállítottunk. Ebben a hónapban az I2C lapka-illesztőprogramok működését tekintjük át, majd üzem közben is megvizsgálunk egyet.

Az I2C lapka-illesztőprogramok az I2C-buszra csatlakozó I2C-eszközökkel folytatott párbeszédeket vezérlik. Az I2C-lapkák általában az alaplapra csatlakozó különféle rendszerelemek fizikai jellemzőit figyelik, például a ventilátorok fordulatszámát, a hőmérsékleti és a feszültségértékeket.

Az `i2c_driver` adatszerkezet egy I2C lapka-illesztőprogramot ír le. Megadása az ***include/linux/i2c.h*** fájlban található. Működő illesztőprogram létrehozásához mindössze a következő mezőkre van szükségünk:

- `struct module *owner;` – értéke `THIS_MODULE`, segítségével a modulra való hivatkozások számlálása oldható meg.
- `char name[I2C_NAME_SIZE];` – az I2C lapka-illesztőprogram beszédes neve. Ez az érték jelenik meg az egyes I2C-eszközökhöz létrehozott sysfs alatti ***name*** fájlokban.
- unsigned int flags; – értékét az `I2C_DF_NOTIFY` értékével megegyezően kell beállítani, így a lapka-illesztőprogram értesülhet arról, ha betöltése után további I2C-eszközök betöltésére is sor kerül. Ez a mező hamarosan valószínűleg eltűnik, mivel az illesztőprogramok túlnyomó része maga gondoskodik a beállításáról.
- `int (*attach_adapter)(struct i2c_adapter *);` – meghívására akkor kerül sor, amikor a rendszer új I2C buszillesztőprogramot tölt be. Később még bővebben is szó lesz róla.
- `int (*detach_client)(struct i2c_client *);` – meghívására az `i2c_client` eszköz rendszerből való eltávolításakor kerül sor. Kicsit később részletesebben is kitérünk rá.

Az alábbi kódrészlet a ***tiny_i2c_chip.c*** nevű I2C példalapka-illesztőprogramból származik. A program a Linuxvilág honlapjáról (➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C könyvtára) címről tölthető le. Az `i2c_driver` adatszerkezet a következőképpen épül fel:

```
static struct i2c_driver chip_driver = {
    .owner          = THIS_MODULE,
    .name           = "tiny_chip",
    .flags          = I2C_DF_NOTIFY,
    .attach_adapter = chip_attach_adapter,
    .detach_client  = chip_detach_client,
};
```

Az illesztőprogram bejegyzése

Az I2C lapka-illesztőprogram bejegyzéséhez az `i2c_add_driver` függvényt kell meghívni, átadott értéke egy mutató lesz az `i2c_driver` adatszerkezetre:

```
static int __init tiny_init(void)
{
    return i2c_add_driver(&chip_driver);
}
```

Az I2C lapkaillesztőprogram bejegyzésének törlését az `i2c_del_driver` függvénnyel végezhetjük el, ennek átadott értéke szintén az `i2c_driver` adatszerkezetre mutat:

```
static void __exit tiny_exit(void)
{
    i2c_del_driver(&chip_driver);
}
```

Az I2C lapka-illesztőprogram bejegyzését követően az I2C buszillesztőprogram betöltésekor az `attach_adapter` visszahívó (callback) függvény meghívására kerül sor. Ez a függvény ellenőrzi, hogy látható-e valamilyen I2C-eszköz azon az I2C-buszon, amelyhez az ügyfél-illesztőprogram csatlakozni próbál. A vizsgálathoz szinte minden I2C lapka-illesztőprogram az I2C-mag `i2c_detect` függvényét használja. A ***tiny_i2c_chip.c*** illesztőprogram például a következőképpen jár el:

```
static int
chip_attach_adapter(struct i2c_adapter *adapter)
{
    return i2c_detect(adapter,
           &addr_data,chip_detect);
}
```

Az `i2c_detect` függvény az `addr_data` adatszerkezetben megadott címeken végez próbákat az I2C-csatolóval. Ha talál

valamilyen eszközt, a `chip_detect` függvényt hívja meg. Ha közelebbről is megvizsgáljuk a forráskódot, észrevehetjük, hogy az `addr_data` adatszerkezet sehol nem szerepel. Ennek az az oka, hogy az adatszerkezet létrehozása a `SENSORS_INSMOD_1` makró feladata. A makró megadása az ***include/linux/i2c-sensor.h*** fájlban található, és nem nevezném könnyed délutáni olvasmánynak. Az illesztőprogram által támogatott különféle lapkatípusok és az elérésükre alkalmas címek alapján a makró egy `addr_data` nevű statikus változót hoz létre. Ezeket az értékeket később a modulnak átadott értékek segítségével lehet megváltoztatni. Egy I2C lapka-illesztőprogramnak a következő változókat kell rendelkezésre bocsátania: `normal_i2c`, `normal_i2c_range`, `normal_isa` és `normal_isa_range`. Ezek a változók adják meg az illesztőprogram által támogatott i2c smbus és i2c isa címeket. Valójában címeket tartalmazó tömbökről van szó – mindegyiket egy különleges érték zárja, az `I2C_CLIENT_END` és az `I2C_CLIENT_ISA_END`. Egy-egy I2C lapkatípus általában csak korlátozott számú címen bukkanhat fel.
A ***tiny_i2c_client.c*** illesztőprogram az említett változókat a következő módon adja meg:

```
static unsigned short normal_i2c[] =
  { I2C_CLIENT_END };
static unsigned short normal_i2c_range[] =
  { 0x00, 0xff, I2C_CLIENT_END };
static unsigned int normal_isa[] =
  { I2C_CLIENT_ISA_END };
static unsigned int normal_isa_range[] =
  { I2C_CLIENT_ISA_END };
```

A `normal_i2c_range` változó azt tudatja velünk, hogy ezt a lapkát bármelyik I2C smbus címen megtalálhatjuk, vagyis illesztőprogramunkat gyakorlatilag bármelyik I2C buszillesztőprogrammal ki tudjuk próbálni.

Mit kell tenni a lapka megtalálása után?

A ***tiny_i2c_chip.c*** illesztőprogramnál az I2C-eszköz megtalálása után az I2C-mag a `chip_detect` függvényt hívja meg. Ennek megadásában a következő átadott értékek szerepelnek:

```
static int
chip_detect(struct i2c_adapter *adapter,
            int address, int kind);*
```

Az `adapter` változó azt az I2C adapter-adatszerkezetet adja meg, amelyen a lapka fellelhető. Az `address` változó azokat a címeket tartalmazza, amelyeken a lapka elérhető, a `kind` változó pedig a megtalált lapka pontos típusát jelzi. A `kind` változót a legtöbb esetben figyelmen kívül hagyja a rendszer, de vannak olyan I2C lapka-illesztőprogramok, amelyek különböző típusú I2C-lapkákat is támogatnak, ezek pontosan a `kind` segítségével tudakolják meg a gépben található lapka típusát.
Ez a függvény felelős az `i2c_client` adatszerkezet létrehozásáért, amely később az I2C-magnál bejegyzésre kerül. Az I2C-mag az adatszerkezetet önálló I2C-eszközként használja. Az adatszerkezet létrehozását a `chip_detect` függvény az 1. listán látható módon végzi el (➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C).

Első lépésként az `i2c_client` adatszerkezet és `chip_data` névvel egy különálló, helyi adatszerkezet jön létre és kap nulla kezdőértéket. Fontos, hogy az `i2c_client` kezdeti nullás értékadása megtörténjen, ugyanis ennek hiányában előfordulhat, hogy a rendszermag-illesztőprogram alsó rétegei nem működnek helyesen. A memória sikeres lefoglalását követően az `i2c_client` adatszerkezet néhány mezőjének az értékét úgy módosítjuk, hogy a tényleges eszközre és illesztőprogramra mutassanak. Kicsit kézzelfoghatóbban: az `addr`, az `adapter` és a `driver` változók kezdeti értékadását kell elvégezni. Az `i2c_client` adatszerkezet nevét úgyszintén be kell állítani, feltéve, hogy azt szeretnénk, hogy a `sysfs` fának az I2C-eszközhöz tartozó bejegyzése helyesen jelenjen meg.
Az `i2c_client` adatszerkezetet kezdeti értékadása után be kell jegyezni az I2C-magnál, ezt az `i2c_attach_client` függvénnyel tehetjük meg:

```
/* Szólunk az I2C-rétegnek, hogy új ügyfél
érkezett. */
err = i2c_attach_client(new_client);
if (err)
    goto error;
```

Ha a függvényhívás hibajelzés nélkül tér vissza, akkor az I2C-eszköz üzembe helyezése a rendszermag oldaláról megtörtént.

I2C és sysfs

A 2.0-s, 2.2-es és 2.4-es sorozatszámot viselő rendszermagokban az I2C-kód az I2C-eszközöket a ***/proc/bus/i2c*** könyvtárba helyezi. A 2.6-os rendszermagnál minden I2C-eszköz és -csatoló a `sysfs` fájlrendszerben jelenik meg. Az I2C-eszközök a ***/sys/bus/i2c/devices*** elérési úton találhatók meg, csatoló- és lapkacímek szerint rendezve.
Ha például egy gépen betöltjük a `tiny_i2c_chip` illesztőprogramot, ennek hatására a 2. listán látható (➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C) `sysfs` faszerkezet jön létre.

Négy különböző I2C-eszköz jelenik tehát meg, s mindegyiket ugyanaz a `tiny_chip` illesztőprogram kezeli. A vezérlést végző illesztőprogramot úgy találhatjuk meg, hogy átnézzük a ***/sys/bus/i2c/drivers*** könyvtár tartalmát vagy belépünk a lapka eszköz saját könyvtárába, és megnézzük a ***name*** fájl tartalmát:

```
$ cat /sys/devices/pci0000:00/0000:00:06.0/
i2c-0/0-0009/name
tiny_chip
```

Az I2C lapka-illesztőprogramok a különféle érzékelőkről leolvasott értékeket az I2C-eszköz könyvtárában lévő `sysfs` fájlokon keresztül teszik elérhetővé. A fájlok neve szabványosított, ahogy az értékek értelmezésére használt mértékegységek is. A vonatkozó leírások a rendszermagfa ***Documentation/i2c/sysfs-interface*** állományában találhatók (lásd a *táblázatot*).
Mint az 1. táblázat alapján is látható, minden fájlban csak egy érték szerepel. Az összes fájl olvasható, továbbá bizo-

Az érzékelők sysfs fájlokon keresztül elérhető értékei

temp_max[1-3]	A legmagasabb hőmérséklet fixpontos, XXXXX formátumú érték, ezerrel elosztva Celsius-fokban mért hőmérsékletet kapunk. Olvasható és írható érték.
temp_min[1-3]	A legmagasabb hőmérséklet- vagy hiszterézisérték fixpontos, XXXXX formátumú érték, ezerrel elosztva Celsius-fokban mért hőmérsékletet kapunk. Általában hiszterézisérték, abszolút hőmérsékletet jelent, és nem a legnagyobbtól való eltérést ad meg. Olvasható és írható érték.
temp_input[1-3]	Bemeneti hőmérsékleti érték. Csak olvasható.

nyos fájlokat – a megfelelő jogosultság birtokában – a felhasználók írhatnak is.
A ***tiny_i2c_chip.c*** illesztőprogram egy I2C-eszközt emulál, amely képes a hőmérsékleti értékek jelentésére. A `sysfs` alatt létrehozza a ***temp_max1***, a ***temp_min1*** és a ***temp_input1*** fájlt; az általa visszaadott érték a fájlok minden egyes olvasásakor növekszik, így az egyes lapkaértékek könnyen megkülönböztethetők egymástól.
A `sysfs` alatt fájlt létrehozni a `DEVICE_ATTR` makróval lehet:

```
static DEVICE_ATTR(temp_max, S_IWUSR | S_IRUGO,
                   show_temp_max, set_temp_max);
static DEVICE_ATTR(temp_min, S_IWUSR | S_IRUGO,
                   show_temp_hyst, set_temp_hyst);
static DEVICE_ATTR(temp_input, S_IRUGO,
                   show_temp_input, NULL);
```

A makró egy adatszerkezetet hoz létre, amely a `chip_detect` függvény futásának végén a `device_create_file` függvénynek kerül átadásra:

```
/* sysfs fájlok bejegyzése */
device_create_file(&new_client->dev,
                   &dev_attr_temp_max);
device_create_file(&new_client->dev,
                   &dev_attr_temp_min);
device_create_file(&new_client->dev,
                   &dev_attr_temp_input);
```

Ezzel a hívással létrejönnek az eszközhöz tartozó `sysfs` fájlok:

```
/sys/devices/pci0000:00/0000:00:06.0/i2c-0/0-0009
|-- detach_state
|-- name
|-- power
|   `-- state
|-- temp_input
|-- temp_max
`-- temp_min
```

A ***name*** fájlt az I2C-mag hozza létre, a `detach_state` és a ***power/state*** fájlokat pedig az illesztőprogrammag.
Térjünk azonban vissza egy kicsit a `DEVICE_ATTR` makróra! A makrónak ismernie kell a létrehozandó fájl nevét, a rá

3. lista Függvényt létrehozó makró

```
#define show(value)
static ssize_t
show_##value(struct device *dev, char *buf)
{
  struct i2c_client *client = to_i2c_client(dev);
  struct chip_data *data =
      i2c_get_clientdata(client);

  chip_update_client(client);
  return sprintf(buf, "%d\n", data->value);
}
show(temp_max);
show(temp_hyst);
show(temp_input);
```

4. lista A set_temp_max függvény

```
#define set(value, reg)
static ssize_t
set_##value(struct device *dev,
            const char *buf, size_t count)
{
  struct i2c_client *client = to_i2c_client(dev);
  struct chip_data *data =
      i2c_get_clientdata(client);
  int temp = simple_strtoul(buf, NULL, 10);

  down(&data->update_lock);
  data->value = temp;
  up(&data->update_lock);
  return count;
}
set(temp_max, REG_TEMP_OS);
set(temp_hyst, REG_TEMP_HYST);
```

vonatkozó futtatási, olvasási és elérési engedélyeket, valamint annak a függvénynek a nevét, amelyet a fájl olvasásakor és írásakor meg kell hívni. A ***temp_max*** fájl esetében ezeknek a megadása a következőképpen történt:

```
static DEVICE_ATTR(temp_max, S_IWUSR | S_IRUGO,
                   show_temp_max, set_temp_max);
```

A fájl olvasásakor a `show_temp_max` függvényt kell meghívni; ennek megadása – sok más `sysfs` fájlhoz hasonlóan – egy másik, egy függvényt létrehozó makróval lehetséges (lásd a *3. listán*).
Annak oka, hogy a függvény létrehozása makróval történik, az, hogy így egyszerűen, a programkód többszörözése nélkül lehet további, gyakorlatilag ugyanolyan feladatokat ellátó, ám más névvel ellátott és más változók értékét kiolvasó sysfs fájlokat létrehozni. A makró három különböző függvényt hoz létre, ezekkel a `chip_data` adatszerkezet három különböző változójának az értékét lehet kiolvasni.
A függvény futása során a `device *` adatszerkezet

5. lista A detatch_client függvény

```
static int
chip_detach_client(
    struct i2c_client *client)
{
    struct chip_data *data =
       i2c_get_clientdata(client);
    int err;

    err = i2c_detach_client(client);
    if (err) {
      dev_err(&client->dev,
              "Ügyfél bejegyzésének törlése"
              " sikertelen, az ügyfél"
              " nem vált le.\n");
      return err;
    }
    kfree(client);
    kfree(data);
    return 0;
}
```

`i2c_client` * adatszerkezetté alakul át. Ezután a privát `chip_data` * adatszerkezet átvételére kerül sor az `i2c_client` * adatszerkezetből. A következő lépés a lapkaadatok frissítése a `chip_update_client` meghívásával. Onnan a kért változó egy átmeneti tárba (buffer) íródik, majd átadásra kerül az illesztőprogrammagnak, amely a felhasználónak adja tovább:

```
$ cat /sys/devices/.../0-0009/temp_input
1
```

A `chip_update_client` minden egyes meghívásakor eggyel növeli az összes értéket:

```
static void
chip_update_client(struct i2c_client *client)
{
    struct chip_data *data =
        i2c_get_clientdata(client);

    down(&data->update_lock);
    dev_dbg(&client->dev, "%s\n", __FUNCTION__);
    ++data->temp_input;
    ++data->temp_max;
    ++data->temp_hyst;
    data->last_updated = jiffies;
    data->valid = 1;
    up(&data->update_lock);
}
```

Így az értékre vonatkozó kérések mindegyike más lesz:

```
$ cat /sys/devices/.../0-0009/temp_input
2
$ cat /sys/devices/.../0-0009/temp_input
3
```

A `set_temp_max` függvény, amely a változók írását teszi lehetővé, szintén makró segítségével jön létre (lásd a *4. listán*).
A `show` függvényekhez hasonlóan ez a függvény is a `device` * adatszerkezetet `i2c_client` * adatszerkezetté alakítja, ezt követően kerül sor a privát `chip_data` * adatszerkezet létrehozására. Ezután a felhasználó által átadott adatokból a `simple_strtoul` függvény segítségével egy számot állítunk elő, majd a megfelelő változóba mentjük:

```
$ cat /sys/devices/.../0-0009/temp_max
1
$ echo 41 > /sys/devices/.../temp_max
$ cat /sys/devices/.../0-0009/temp_max
42
```

Rendrakás

Az I2C lapka-illesztőprogram rendszerből való eltávolításakor, történjen az akár az I2C buszillesztőprogram, akár az I2C lapka-illesztőprogram eltávolításával, az I2C-mag meghívja az `i2c_driver` adatszerkezetben megadott `detatch_client` függvényt. Ez – mint a példa-illesztőprogramból is kitűnik (*5. lista*) – általában egy egyszerű függvény.
Ahogy az `i2c_attach_client` függvény hívására volt szükség az `i2c_client` adatszerkezetnek az I2C-magnál történő bejegyzéséhez, úgy az `i2c_detach_client` függvény végzi el a bejegyzés törlését. Ha a függvény végrehajtása sikerrel jár, akkor – még a függvényből való visszatérés előtt – fel kell szabadítani az illesztőprogram által az I2C-eszköz számára lefoglalt memóriát.
A példa-illesztőprogram nem végzi el a `sysfs` fájlok kifejezett eltávolítását a `sysfs`-magból, erre az illesztőprogrammagban önműködően kerül sor, az `i2c_detach_client` függvény részeként. Ha a szerző úgy kívánja, a `device_remove_file` meghívásával a fájlok eltávolítását kézzel is elvégezheti.

Összegzés

Két írásomban áttekintettem a rendszermag I2C illesztőprogramok írásával, valamint az I2C algoritmus- és az I2C lapka-illesztőprogrammal kapcsolatos alapvető tudnivalókat. Az I2C illesztőprogramok írásával kapcsolatban rengeteg érdekes olvasnivaló található a rendszermagfa ***Documentation/i2c*** könyvtárában és az ***Lm_sensors*** oldalon (➲ http://secure.netroedge.com/~lm78).

A cikkhez kapcsolódó listák megtalálhatóak a ➲ http://melleklet.linuxvilag.hu/04_2004_Aprilis/I2C könyvtárában.

Linux Journal 2004. február, 118. szám

Greg Kroah-Hartman (greg@kroah.com)
Jelenleg a Linux-rendszermag különféle illesztőprogram-alrendszereiért felelős. Az IBM-nél dolgozik és a Linux-rendszermaggal kapcsolatos kérdésekkel foglalkozik.