Nyomtatókapu közvetlen programozása

A legtöbb számítógépben, legyen az hordozható vagy asztali gép, a mai napig megtalálható a párhuzamos vagy nyomtatókapu (port).

A Centronics kapu elnevezés burkoltan az alkalmazott protokollt is tartalmazza, de ez mára már nem teljesen igaz. Jelenleg a legtöbb nyomtató az IEEE 1284 szabvány ajánlásainak megfelelően működik, ami lehetővé teszi az önműködő felismerést, valamint az egyéb kétirányú kapcsolattartást.
Nos, mire is jó a nyomtatókapu? Önként adódik a válasz: nyomtatásra. Ez viszont ma egyre kevésbé igaz, ugyanis a legtöbb új nyomtató különböző újabb kapukat használ, leginkább az USB-t, de előfordul – különösen a hordozható változatoknál – az infra vagy a Bluetooth (rádiós) adatátvitel is. Korábban nyomtatókapus lapolvasók is léteztek, de ezek hamarabb álltak át az USB-kapura, mint maguk a nyomtatók.
Így viszont egy kissé haszontalannak tűnhet a nyomtatókapu, de ha már ott van, használjuk valamire! Régebben a nyomtatókapuhoz ethernet- (még régebben arcnet-) illesztők is megtalálhatók voltak, amelyek – ha némileg lassabban is, mint az „igazi" ethernet, de – használhatóak voltak. Ha viszont nincs ilyenünk, de két linuxos géppel is rendelkezünk és ezeket össze szeretnénk kötni, megtehetjük a dolgot magának a nyomtatókapunak a használatával is.

Hálózat a nyomtatókapun keresztül

Előfordulhat, hogy két linuxos gépet össze akarunk kötni, de nem azonos a bennük található hálózati kártyák hálózatkezelése, vagy csak nem akarjuk egy másik hálózatról átállítani, esetleg azt a másik hálózatot is el akarjuk érni. Megeshet, hogy a régebbi hordozható gépekben nincs is hálózati csatoló. Nos, ha van szabad nyomtatókapu a gépekben, megoldhatjuk a problémát. Mire van hozzá szükségünk?

- Mindkét gépben egy-egy szabad nyomtatókapura.
- Egy nullnyomtató kábelre, laplinkként is ismert, esetleg csak egy átalakítóra, s a nyomtatókábel végére.
- A gépeken telepített GNU/Linux-rendszerre (esetleg más Unixra).

Mit is kell tennünk?

1. Először csatlakoztatni kell a kábel két végét a gépekhez.
2. Ha az `lp` modult használjuk, akkor addig távolítsuk el, amíg a nyomtatókaput a hálózati feladatokhoz használjuk (`rmmod lp`).
3. Ha még nincs bent, töltsük be a `parport` modult (`insmod parport`).
4. Ezt követően a megfelelő értékekkel töltsük be a `parport_pc` modult (például `insmod parport_pc io=0x378 irq=7`).
5. A `plip` modult (`insmod plip`) is töltsük be.
6. A rendszerüzenetek között nézzük meg, hogy a plip kapunk (`dmesg`) milyen eszközszámot kapott, ez általában `plip0`.
7. A két gépen állítsuk be a csatolót (ez az IP-cím csupán szemléltetésül szolgál, a lényeg, hogy egymásra hivatkozzanak):
   `ifconfig plip0 192.168.10.1 ➥pointopoint 192.168.10.2 up`
   illetve
   `ifconfig plip0 192.168.10.2 ➥pointopoint 192.168.10.1 up`
8. A `ping` utasítás segítségével meggyőződhetünk arról, hogy a kapcsolat működik-e.
9. Ha az egyik gép a másikon keresztül más hálózatot is el akar érni, akkor ezen átjáróként állítsuk be a másik gépet:
   `route add default gw ➥192.168.10.1`
10. Továbbá a másik gépet állítsuk be útválasztónak (azaz engedélyezzük az IP-csomagtovábbítást).

A Linux nyomtatókapu-illesztője

Röviden tekintsük át, hogyan is használják a programok a párhuzamos kaput Linux-rendszer alatt. Előbb azonban ismét vessünk egy pillantást a múltba: a DOS időszakában a hozzáértők számára egyszerű volt a különböző kapuk és általában a gép alkatrészeinek az elérése – ezt a megfelelő kapucímek írásával és olvasásával lehetett megoldani. Például, ha a nyomtatókapu a 0x3bc címen helyezkedett el (hogy igazán régi legyen a példa, mert ezt a címet általában a mono-, Herkules és CGA kártyák beépített nyomtatókapuja használta), akkor egy 8 bites értéknek a nyomtatókapu adatvonalaira való kiírása a következőképpen történt:

```
outb(0x3bc,data);
```

vagy assemblerben:

```
mov     dx,0x3bx
out     dx,ax      ; ax=data
```

Nos, ez természetesen már a múlté… Mivel sokféle nyomtatókapu létezik (na jó, nem sok, de többféle, például a PC-s változat, a Macintoshé, a Sun munkaállomásé), valamint többféle csatlakoztatható eszköz van, célszerű volt egy illesztőfelületet írni, amit ***Tim Waugh*** mások segítségével meg is tett.

A felépítés lényege, hogy az eszközök elérése többrétegűvé vált (ez természetesen más elemekre is igaz, nem csak a nyomtatókapura). Legalul egy közvetlenül az adott vas típusától függő modul helyezkedik el, esetünkben a `parport_pc`. Ennek meg is adhatjuk a pillanatnyi jellemzőket, ha tudjuk – például a betöltése ilyesmi lehet:

```
parport_pc io=0x3bc irq=none
```

A következő modul a `parport`, amely egységes elérési felületet biztosít a különböző tényleges kapumegvalósításokhoz. Innentől kezdve a nyomtatókapu a gép megvalósításától függetlenül egységes módon érhető el. A `parport` programozási felületét használva működnek a „magasabb” szinten lévők: a leggyakoribb az `lp`, ez valóban nyomtatót illeszt a rendszerbe, de ilyen még az előbb látott `plip` is. A nyomtatókapu használata a `parport` modul függvényein keresztül lehetséges, azonban csak úgy, ha rendszermagmodult (drivert) készítünk. Ennek a modulnak a következő feladatai vannak:

- Be kell jegyeztetnie magát mint a nyomtatókapu felhasználóját (registration).
- Le kell kérdeznie, hogy milyen kapuk vannak a rendszerben (ezt visszahívó, azaz callback függvények segítségével valósítja meg: `attach` és `detach`).
- Le kell foglalnia a kaput (`claim`), ami blokkoló és nem blokkoló módon is történhet, azaz vagy vár, amíg szabad lesz a kapu, vagy nem, ekkor viszont nem biztos, hogy sikerült lefoglalni.
- Ezután a `parport` és `parport_dev` szerkezeteken keresztül használhatja a kaput.
- A kapu használatát `yield` típusú függvények segítségével ideiglenesen fel lehet adni.
- Végül a `release` függvény segítségével a használat jogát más folyamatoknak adhatja át.

A `parport` modul programozásáról, valamint egy „próba” `lp` meghajtó írásával kapcsolatban sokat megtudhatunk a Tim Waugh által írt parportbook dokumentumból vagy a Linux-mag forrásszövegéhez mellékelt leírásokból.

Közvetlen kapuelérés

Mint láthattuk, a kapu elérése meglehetősen bonyolult dolog. Ez nagy sorozatban gyártott eszközök esetén nem annyira nagy baj, mert előbb-utóbb úgyis megírja valaki a szükséges meghajtóprogramot (esetleg többen is, netán magunk is besegíthetünk). Ha azonban mi magunk akarjuk használni a kapukat a saját készülékünkkel való „beszélgetésre”, akkor nehezebb helyzetben vagyunk. Kedvenc vesszőparipám és munkám mellett hobbim is a mikrovezérlők programozása. Ezekhez szerettem volna egy programbetöltő készüléket (régebbi nevén „égetőt”) készíteni. Mindemellett elsőrendű célom az volt, hogy lehetőleg mindenféle rendszerből lehessen használni, azaz Linux, Windows és – horribile dictu! – DOS alól is. Ezért a nyomtatókapuhoz való csatlakoztatás mellett döntöttem. Ennek külön előnye, hogy még az újabb hordozható gépeken is megtalálható, így mindenfelé használható lesz.
A DOS-változattal a fent említettek szerint nincs gond, ugyanakkor a másik kettőnél jócskán kijutott belőle. Mire van szükség egy ilyen készülék illesztésénél? Digitális ki- és bemenetekre, mégpedig úgy, hogy mind beolvasni, mind kiírni bitenként lehessen őket – tehát nem elegendő, amit a nyomtatóillesztő programok nyújtanak, hogy tudniillik egész bájtokat küldünk a kapura és visszafelé csak a nyomtató állapotjeleit (illetve azok közül is csak néhányat) olvassuk. Lássuk legelőször azt, hogy mit is tud nyújtani a nyomtatókapu:

- egy 8 bites adatkaput, amely az újabb gépeken kétirányú, a régebbieken csak kimenet;
- egy 5 bites állapotkaput (status), amely bemenet;
- egy 4 bites vezérlő (control) kaput, amely nyitott kollektoros kimenet felhúzó ellenállással, és vissza is olvasható, így elvileg bemenetként szintén használható lenne, de ez nem minden gépen működik (esetenként más módon van megvalósítva).

A kapuk címei a báziscímhez (B, ami a fent említett 0x3bc mellett 0x278 és 0x378 szokott még lenni, az új BIOS-okban választhatóan) a következők:

- adatkapu B+0 (például: 0x278),
- állapotkapu B+1 (például: 0x279),
- vezérlőkapu B+2 (például: 0x27a).

Ezeket a címeket szeretnénk írásra és olvasásra elérni. Ha az `outb` és `inb` függvényekkel próbálkozunk, mint DOS-ban, akkor szakaszolási hibát (segmentation fault) kapunk, lévén, hogy az adott ki-bemeneti terület nem tartozik a programunkhoz. Tehát a magon (kernel) keresztül kell elérnünk. Szerencsére a `parport` modul szerzője `ioctl` hívásokon keresztül lehetővé tette a kapu közvetlen írását és olvasását. Ezt elég szegényes módon említik meg a leírásokban, és a neten is kevés ilyen jellegű útmutató található, ezért is osztom meg kutatásaim eredményét az érdeklődőkkel. Először is akciónk sikeréhez az kell, hogy a `parport` modul (ez pedig függ a `parport_pc` modultól) be legyen töltve. Itt említem meg, hogy a viszonylag újabb magok (2.4) esetében a `parport0` eszközfájl (***/dev/parport0***) független az adott kapu báziscímétől: mindig az elsőként beállított kapuhoz kapcsolódik, a `parport1` a másodikhoz stb. Régebben ezek a fizikai címekhez kötődtek.
Ezután arra van szükség, hogy a felhasználó, aki futtatni akarja a programot, írási és olvasási jogosultsággal bírjon a `parport0` eszközfájlon. Ha mindez megvan, akkor programunkban a következő fejlécfájlokat az alábbi módon be kell olvastatnunk:

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/io.h>
#include "ppdev.h"
```

Az utóbbi azért a pillanatnyi könyvtárból lesz beolvasva, mert ha nem rendszergazdaként és magfordítás közben dolgozunk, akkor nincs benne az ***include*** állományok elérési útjában (általában csak magfordításnál szükséges), ezért átmásoltam a forrásállományok könyvtárába. Tulajdonképpen ez az utóbbi tartalmazza a `parport` eszközhöz szükséges meghatározásokat. Régebbi rendszereken esetleg ***ppuser.h*** is lehet a neve, a használat szempontjából csaknem azonosak. Az adott fájlt a mag forrásfájában kell megkeresnünk.

Miután mindezt megtettük, a következő módon használhatjuk a kaput:

- `hLpt= open("/dev/parport0", O_RDWR);` – megnyitja az eszközfájlt.
- `ioctl(hLpt, PPEXCL,0);` – kizárólagosságot kér a kapura, a következő hívás előtt kell beállítani.
- `ioctl(hLpt,PPCLAIM,0);` – használati igényt „jelent be” a kapura. 0 visszaadott érték esetén megvan, egyébként nincs.
- `ioctl(hLpt,PPRELEASE,0);` – visszaadja a rendszernek a kapu használati jogát, utána a `close(hLpt)` hívással be lehet zárni az eszközfájlt.
- `ioctl(hLpt,PPRSTATUS,&data);` – az `unsigned char data` változóba másolja az állapotkapu bitjeinek az értékét.
- `ioctl(hLpt,PPRDATA,&data);` – az `unsigned char data` változóba másolja az adatkapu bitjeinek az értékét; ez csak kétirányú változat esetén használandó.
- `ioctl(hLpt,PPWDATA,&data);` – a `data` változó tartalmát az adatkapura írja. Amit ide kiírunk, az a következő kiírásig változatlan marad, azaz a megfelelő logikai értéke lesznek a csatlakozó érintkezőin. (Ez a következő vezérlőkapura is igaz). Így felhasználható a külső eszközünk vezérlésére.
- `ioctl(hLpt,PPWCONTROL,&data);` – a vezérlőkapura írja a `data` változó tartalmát.
- `ioctl(hLpt,PPRCONTROL,&data);` – beolvassa a vezérlőkapu bitjeit. Ez nem biztos, hogy ugyanaz, mint amit kiírtunk, mivel külső eszköz megváltoztathatja a logikai szinteket, valamint a lentebb említett módon bizonyos bitek invertálódnak.
- Emellett vannak még a kétirányú változatok adatirányát beállító, az ezekhez tartozó kibővített vezérlőregisztert, a FIFO tárolót, és a magasabbrendű protokollt segítő hívások, de ezek túlmutatnak cikkünk témáján.

Mint azt a korábbiakban megemlítettem, az állapot- és vezérlőkapunak van néhány furcsa sajátossága. Az állapotkapu csak 5 bites, ezek a bájt felső részén olvashatóak, azaz az alsó három nem valós érték. Ráadásul a legfelső bit invertált, azaz a csatlakozón lévő logikai szint ellentétét mutatja. (A miértekre nem tudok felelni, valószínűleg így jobban megfelelt az eredetileg kialakított protokollnak.) Tehát, ha a kapun lévő valódi adatot meg akarjuk kapni, akkor a `data=data ^ 0x80;` utasítással meg kell ezt a bitet fordítanunk (ez egy bináris `xor` művelet, ha a műveleti jel netán ismeretlennek tűnne).
A vezérlőkapunál is vannak érdekességek: ez csak 4 bites, de ezek a bájt alsó részén helyezkednek el. Ráadásul több is invertált. Így a következő módon kell kiírnunk az adatot, ha azt akarjuk, hogy az 1 állapotú bitek logikai 1 értéket jelentsenek a csatlakozón:

```
data=data ^ 0x0b;
```

A kiírást követően, hogy a felső bitek ne zavarjanak (mivel néhány esetben ezek közül valamelyik vezérli a kétirányú adatkapu irányát, esetleg a megszakítási jel előállítását is):

```
data=data & 0x0f;
```

Ezután már kiírhatjuk a kapott adatokat a fent látott `ioctl(hLpt, PPWCONTROL,&data);` hívással.
Mint látjuk, némi küzdelem árán elérhetjük, hogy kedvenc Linux-rendszerünkből is ugyanúgy kezeljük a párhuzamos kaput, mint régen, a „mindent szabad” DOS-os időszakban. Ez előnyös lehet, ha saját építésű vagy más okból a rendszer által nem támogatott készüléket akarunk működtetni. Ilyen esetben a nehézkes kipróbálás miatt elég munkás a további fejlesztést C-ben végezni; erre jelenthet megoldást a – talán egy későbbi cikkben ismertetésre kerülő – beágyazott forth rendszer használata, amelyből megfelelő ingyenes, nyílt forrású változatot Linuxra is találhatunk.

Havránek Ferenc
(hf@delvidek.hu)
Automatikamérnökként dolgozik. Kedvtelései közé tartozik mindenféle kétkerekű járművön (kerékpár és motor) való közlekedés. Ezenkívül szívesen tölti idejét programozással, nemcsak PC-s, hanem egyéb környezetben is, például mikrovezérlő programokat ír.