Séta a lemezes egységek körül (2. rész)

A számítógép adatokkal dolgozik, és ezeket az adatokat az úgynevezett beviteli-kiviteli eszközökön keresztül juttathatjuk a memóriába, illetve ezek segítségével ismerhetjük meg az adott művelet végeredményét. Azt is megállapítottuk, hogy a beviteli-kiviteli eszközöket nagyjából két csoportra oszthatjuk: karakteresekre és blokkosokra. A karakteres eszközök tárgyalásakor a terminálokkal foglalkoztunk, amelyek a felhasználó és a számítógép közötti kapcsolat megvalósításáért felelősek. A blokkeszközök közül a legjellemzőbbek a lemezek, amelyek adataink háttértárolására szolgálnak. Legutóbb a blokkeszközöket csak általában véve néztük meg, továbbá bemutattuk, hogy milyen elvet alkalmaz a Linux a blokkeszközök meghajtására. Itt is az eszközfüggetlenség elve érvényesült, tehát az volt a cél, hogy az operációs rendszer kódjának csak a lehető legkisebb része tudjon bármit is az eszköz kezelésének módjáról. A felsőbb rétegek az eszközöket csupán absztrakt utasítások formájában érhetik el. Ilyen utasítás például az „Olvasd be a 12. blokkot!", vagy a felhasználói szinten lévő programok esetében az „Írd ki ezt valamelyik állományba!".
Most továbbra is a blokkeszközökkel, pontosabban a lemezekkel foglalkozunk. Sorozatunk előző részében az eszközfüggetlen részt már alaposan áttanulmányoztuk, ám magáról a lemezkezelőről (ez tartja a kapcsolatot ténylegesen a lemezegységgel) kevés gyakorlati mozzanatot árultunk el. Ezért az elkövetkezendőkben tovább boncolgatjuk a lemezek felépítését és az operációs rendszer lemezkezelőjének a működését.
Sokakban az a kétely támadhat, hogy e cikk írója a terminálokat hátrányosan kezeli, ugyanis csak röpke két oldal erejéig foglalkozott velük, míg a lemezekkel két részen keresztül. Valójában nincs szó ilyesmiről. A fentiek oka egyrészt abban rejlik, hogy a karakteres eszközökkel az operációs rendszer könnyebben „elbír", mint a blokkosokkal. Másrészt ne feledkezzünk meg arról, hogy a terminál a felhasználóval áll közvetlen kapcsolatban. A felhasználó pedig nem eleve meghatározott jelenség, a lehető legelképzelhetetlenebb dolgokra képes. A terminálkezelő kódja és működése eléggé bonyolult, de erre azért is szükség van, hogy hatásosan meg tudja védeni a rendszert a felhasználó ténykedéseitől. Tehát bármit tegyen emberünk (akár csépelheti is a gombokat, vagy telemorzsázhatja a billentyűzetet), az ne legyen hatással a rendszer működésére (például ne tudja lefagyasztani magát a terminálkezelőt, ami rendkívül sok kellemetlenséggel járhat).
A lemezkezelőnek talán annyival könnyebb a dolga a terminálkezelővel szemben, hogy biztos lehet benne: az eszköztől az adatokat mindig ugyanúgy kapja vissza. Ennek köszönhetően meglehetősen kevés külső tényező befolyásolhatja a feladat végrehajtását, ezért nem is kell annyi óvintézkedést tennünk, mivel gépekkel és nem emberekkel van dolgunk. De ne higgyük, hogy egy lemezkezelő tervezése gyerekjáték. A nehézségek egész seregével kell megküzdenünk, és nem könnyű igazán jó megoldást találni.

A merevlemez

Az előző részben beszéltünk már ugyan a lemezek felépítéséről, de ahhoz, hogy még részletesebben bemutathassuk a merevlemez-kezelő működését, ki kell egészítenünk az ott leírtakat.
A merevlemezekről már tudjuk, hogy szektorokra, pályavonalakra (vagy sávokra), illetve cilinderekre tagolódnak. A pályavonal nem más, mint egy teljes körülfordulás alatt felírt bitsorozat, amely általában 512 bájt hasznos adatot tartalmazó szektorokból áll. A merevlemez több lemezt tartalmaz, amely egymás felett helyezkedik el, és mindegyikhez tartozik egy olvasófej (illetve kettő, mert a lemez mindkét oldalán lehet tárolni adatokat). Az olvasófejek mozgatókarjai rögzítve vannak egymáshoz, tehát egyik fej sem mozgatható külön-külön. Így mindig egymás alatt, minden lemeznek ugyanazon a pályavonalán tartózkodnak. Az egymás alatti pályavonalak együttesen egyetlen cilindert alkotnak. (Könnyű belátni, hogy a cilinderek száma megegyezik a lemezek sávjainak számával). Ezt a felosztást a merevlemez geometriai felépítésének nevezzük, amely nem minden esetben egyezik meg a valódi (fizikai) felépítéssel (lásd később).
Hogy a geometriai felépítést könnyebben megértsük, nézzük meg, miképp határozhatunk meg egy szektort a merevlemezen. Először is meg kell adnunk a cilinder számát, ami azt mondja meg, hogy az olvasófejeket hova kell mozgatnunk (és egyben azt is, hogy majd melyik pályavonalról szeretnénk olvasni). Ezután ki kell választanunk egy fejet (amellyel meghatározzuk, hogy egyrészt melyik lemezről, illetve annak melyik oldaláról szeretnénk olvasni). Legvégül át kell adnunk a szektor sorszámát. E három adat segítségével egyértelműen meghatározhatjuk a merevlemez bármelyik szektorát.
A mai merevlemezeknél a külső sávok több szektort tartalmaznak, mint a belsők. Ez lényegesen megnöveli a lemez tárolókapacitását, de bonyolultabb vezérlést követel. Szerencsére az operációs rendszernek ezzel nem kell foglalkoznia, ugyanis minden lemezegységhez tartozik egy vezérlő, ami tulajdonképpen egy kis számítógép saját processzorral és memóriával. Az operációs rendszer merevlemez-kezelője a vezérlővel tartja a kapcsolatot. A mai vezérlők programjai már nagyon bonyolultak és sok mindenre képesek, például saját gyorstárral rendelkeznek, illetve ha egy szektor meghibásodik, azt helyettesítik egy másikkal. (Minden pályavonalhoz tartozik pár tartalékszektor, amelyet kifejezetten ilyen célokra tartanak fenn. Ha egy szektor „beadja a kulcsot", a vezérlő a hibás szektor címét átirányítja az egyik tartalék szektorra, ennek köszönhetően a hibás szektorok elérhetetlenné válnak. Ez mind az eszköz szintjén történik, ezekről a dolgokról azonban az operációs rendszer az égvilágon semmit sem tud.)
A PC-s világban leggyakrabban használt lemezcsatolófelület az IDE (Integrated Drive Electronics, azaz beépített eszközelektronika), illetve az azt szinte már teljesen kiváltó EIDE (Extended IDE). Az IDE-lemezek a hajdani XT-hez készült Seagate 10 MB-os lemezéből fejlődtek ki, amelyeket az operációs rendszer BIOS-hívások segítségével tudott kezelni. Egy szektor eléréséhez a processzor regisztereibe be kellett töltenie a megfelelő fej-, cilinder- és szektorszámokat. Az IDE fejlesztői az együttműködés érdekében megtartották a BIOS-on keresztül zajló címzési módot.
A nehézség abban gyökeredzett, hogy egy szektor címzésében a fej megadásához 4, a cilinderhez 10, a szektorhoz pedig 6 bit állt rendelkezésre, azaz a BIOS segítségével csak olyan

merevlemezeket kezelhettünk, amelyben legtöbb 16 fej, 1024 cilinder és 63 szektor van (összesen körülbelül egymillió szektor). Ezzel nem is volt semmi gond, egészen addig, amíg meg nem jelentek az 528 MB-nál nagyobb tárterületű lemezek. Ezek 1024 cilindernél sokkal többet tartalmaztak, tehát már nem lehet őket a „hagyományos" módon címezni. Ezért a vezérlőket úgy módosították, hogy képesek legyenek a címek közötti bűvészkedésre: a BIOS-határon belüli értékeket fogadtak el, de azokat átszámolták maguknak a valódi fizikai értékre. Magyarán a felhasználó szemébe hazudtak, de ez legalább működő módszer volt, ha nem is túlzottan hatékony.

Nem sokkal ezt követően megjelent az EIDE, ami az IDE-vel szemben számtalan előnnyel rendelkezett. Lehetővé tette egy addig teljesen más címzési mód használatát is, ezt LBA-nak (Logical Block Addressing) hívták, ami nemcsak az 528 MB-os határ átlépését, de a szektorok egyszerűbb elérhetőségét is lehetővé tette. Az elv rendkívül egyszerű: a szektorokat beszámozzuk 0-tól 2^{24}-ig, a vezérlő pedig gondoskodik ennek az értéknek fej-, cilinder- és szektorszámmá való átalakításáról. Ez igazán kényelmes megoldás az operációs rendszer lemezkezelője számára. Az EIDE vezérlőhöz összesen négy meghajtót lehet csatlakoztatni, és a CD-ROM-meghajtókat is kezelni tudja. Manapság az EIDE-meghajtók olcsó áruk miatt nagyon elterjedtek, ezért szinte minden mai, Intel-kiépítésen is futó operációs rendszer támogatja (habár az EIDE ma már más számítógépeken is használatos). Mindezek ellenére az EIDE-lemezekkel számos gond akadt, ilyen például a viszonylag kis adatátviteli sebesség (ez például a videodigitalizáláskor jelenthet gondot), illetve az, hogy egyszerre csak egy eszköz működhet. A lemezek másik nagy családja, a SCSI azonban a gyorsabb átvitelt és több eszköz egyszerre való működését is lehetővé teszi.

A SCSI-ról csupán annyit említenénk meg, hogy ez több mint egy egyszerű vezérlő, valójában egy olyan adatsín, amelyre hét vagy tizenöt eszköz csatlakoztatható (például merevlemez, CD-író, lapolvasó vagy bármilyen más SCSI-s eszköz). A legtöbb ilyen egység külön kimenettel és bemenettel rendelkezik, ahol a kimenetet a másik bemenetére kell kapcsolni. A SCSI leginkább a hálózati kiszolgálóknál vagy a videodigitalizálásra használt gépekben elterjedt. Mi a továbbiakban az EIDE merevlemezkezelővel foglalkozunk, de a többi is ugyanezeket az alapelveket követi.

Folytassuk egy kis kitérővel. A PC-k esetében különösképpen fel kell készülnünk egy olyan nehézségre, aminek a gyökere magának a PC-nek az egyik jelentős tulajdonságából fakad. A PC ugyanis nem egy kézzelfogható számítógépfajta (mint például a Macintosh, ahol kapunk egy dobozt, amelybe minden elképzelhető dolog be van építve), hanem a különböző típusú egységekből felépülő számítógépek nagy családja, amelyek különböző processzorokat, sínrendszereket, memóriákat használhatnak. Gond abból fakad, hogy ezeknek az eszközöknek a programozása olykor gyökeresen eltérő lehet. Példaképpen megemlíthetjük a ma már múzeumba való XT-ket, amelyek 8-bites sínrendszerrel rendelkeztek. A 286-os, 386-os, 486-os és a Pentium kategóriás processzorokhoz azonban már az úgynevezett AT-s sín tartozik, ami 16 bites kapcsolódási felülettel bír. Ma pedig már a 32-bites PCI-sínek a legelterjedtebbek. A legtöbb számítógépben található egy úgynevezett gyári program (firmware), amely az alaptesten (PC-sen szólva az alaplapon) található, csak olvasható memóriában helyezkedik el. Ez tulajdonképpen nem más, mint a BIOS.

A BIOS egyik feladata az, hogy olyan eljárásokat nyújtson az operációs rendszer számára, amelyeket használva az figyelmen kívül hagyhatja az eszköz ehhez hasonló sajátosságait. Ez nagyon örvendetes, ugyanis – ha például a merevlemez-kezelést vesszük alapul – az operációs rendszernek, a BIOS merevlemezkezelő eljárásait használva, elvben nem kell olyannal foglalkoznia, hogy az adott vezérlő most XT-s vagy AT-s, vagy akármilyen más típusú, valószínűleg minden működni fog (el kell ismernünk, ez a példa egy kicsit sántít, ugyanis aki ma XT-n szeretne Linuxot futtatni, az „kemény arc"; akinek még sikerül elindítania is, az vagy hazudik, vagy valamiféle nagymester, mivel a Linux nem hajlandó védett módot nem támogató processzorokon futni; de azért vannak olyan operációs rendszerek – például a MINIX –, amelyek, ha nagyon muszáj, valós módban is elfutnak).

A BIOS-szal viszont alapvető gondok vannak, például az, hogy az MS-DOS-hoz fejlesztették ki, amely köztudottan 16-bites és nem is többfeladatos. Mi most azonban védett módban futó, többfeladatos operációs rendszerekben gondolkozunk. Ha az ilyen rendszerek a BIOS segítségével szeretnék a beviteli-kiviteli kapukat elérni, szembe kell nézniük azzal, hogy a processzort minden BIOS-hívásnál át kell kapcsolniuk valós módra, majd ismét védett módba. Az ilyen ide-odakapcsolgatás pedig iszonyatosan lassú művelet. Ezért ha valaki még azt is szeretné, hogy a rendszere valamennyire gyors legyen, úgy, ahogy van, el kell felejtenie a BIOS-t, és a beviteli-kiviteli eszközök kezelését saját magának kell elvégeznie.

Azért a BIOS-nak óriási szerepe van a gép elindításánál: sok operációs rendszer indítóprogramja is a BIOS-t hívja segítségül, hogy a rendszermagot a merevlemezről a memóriába töltse. Utána viszont már semmi szükség sincs rá; ma már legfeljebb az elavult DOS-os alkalmazások használják a BIOS eljárásait (de még azok sem mindig, mert iszonyatosan lassú).

El is érkeztünk igazi kérdésünkhöz: hogyan fér meg a rendszermagban például két merevlemez-kezelő? Értelemszerűen csak az egyikre lesz szükség, de valamilyen úton-módon el kell dönteni, hogy melyikre. Összesen három lehetőségünk van. Az első: a rendszermagot úgy fordítjuk le, hogy csak azokat a kezelőket tartalmazza, amelyekre valóban szükség lesz. A második és a harmadik lehetőségnél az összes létező támogatást befordítjuk, de míg az egyiknél a döntést a felhasználóra bízzuk, a másiknál maga a rendszermag próbálja kipuhatolni, hogy melyik kezelő is illik az adott eszközhöz.

Mind a három módszernek egyaránt vannak előnyei és hátrányai. Az elsőnél például semmi ilyen jellegű gond nincs, de az adott rendszermag kizárólag a mi beállításainkkal megegyező gépeken lesz hajlandó futni. Az sem feltétlenül jó, ha a választást a felhasználóra bízzuk, mivel nem biztos, hogy szakértője a gépben található eszközök pontos típusainak. Elengedhetetlen tehát, hogy az operációs rendszer önmaga is képes legyen felismerni a gépünkben található beviteli-kiviteli eszközöket, és több kezelő esetén ki tudja választani a megfelelőt. (Ám a PC-k esetében ez az „eszközfelismerősdi" sem olyan egyszerű, ugyanis a piacon rengeteg, a szabványnak csak részben megfelelő termékkel találkozhatunk. Amikor egy eszközt próbálunk felismerni, első lépésként a beviteli-kiviteli kapukat kezdjük el letapogatni. De ez nagyon veszélyes játék! Például véletlenül működésbe hozhatunk egy olyan eszközt, amelynek helytelen beállítása következtében az egész rendszer kiakadhat, és már csak a RESET segíthet rajtunk. Ezt elkerülendő a Linux, a Windows és a többi rendszer is bizonyos biztonsági eljárásokat alkalmaz. Ha például az eszköztől sokáig nem érkezik értékelhető válasz, az adott folyamat azonnal megsemmisül.)

De térjünk vissza a lemezkezelőre! Az előző részben említettük, hogy az eszközfüggő lemezkezelő a felsőbb rétegektől hatféle

utasítást kaphat. Ezek a következők voltak: DEV_OPEN, DEV_CLOSE, DEV_IOCTL, DEV_READ, DEV_WRITE és SCATTERED_IO. Most megnézzük, mit is jelentenek ezek az utasítások a merevlemez esetében!
A DEV_OPEN az eszköz első elérésekor hívódik meg. Feladata többnyire csak annyi, hogy beállítson bizonyos alapértékeket, amelyekre szükség lesz az eszközről történő olvasáshoz, illetve az arra való íráshoz – ilyenek például a merevlemezen található lemezrészek. A DEV_CLOSE például a CD-ROM-oknál lehet hasznos, segítségével az eszközt rábírhatjuk, hogy dobja ki a CD-t. A DEV_IOCTL a merevlemezek esetében a lemezfelosztási tábla módosításával foglalkozik, de például a CD-ROM esetében beállít egy értéket, ami megmondja a DEV_CLOSE-nak, hogy ki kell-e adnia a korongot a meghajtóból. A DEV_READ-hez és a DEV_WRITE-hoz csupán annyit fűznénk hozzá, hogy a beérkező kérések végrehajtásakor a kezelő nem alkalmazza az előző részben bemutatott liftes algoritmust. Helyette egy rendezési eljárást alkalmaz, ami összeszedi az egymás utáni szektorokra vonatkozó kéréseket. Ez azért jó, mert a merevlemezről az esetek többségében a fájlrendszer olvasni szeretne, és a művelet az esetek többségében több egymást követő szektor beolvasásával jár. Erre a fájlrendszer bemutatásánál még részletesebben visszatérünk.
A DEV_READ és a DEV_WRITE műveleteknél a feladat elvégzése után mindig hívódik egy befejező eljárás. De mivel gyakran van szükség egymás utáni szektorok beolvasására, illetve írására, a SCATTERED_IO nélkülözhetetlen a lemezműveletek felgyorsításához. Segítségével ugyanis egyszerre több, egymást követő szektort is beolvashatunk, de az a bizonyos befejező eljárás csak egyszer hívódik meg, miután már mindent beolvastunk, illetve kiírtunk, amit csak kellett.

Hajlékonylemezek

A hajlékonylemez-meghajtók jóval egyszerűbb felépítésűek, mint a merevlemezek, kezelésük azonban nagyobb kihívást jelent az operációs rendszer számára. Ennek oka több dologra is visszavezethető.
Először is a viszonylag egyszerűbb felépítésű eszközhöz egyszerűbb vezérlő is társul. Míg a merevlemezek esetében az operációs rendszer a bonyolult és összetett műveleteket a vezérlőre bízta, sok mindenről saját magának kell gondoskodnia. Vegyük például az olvasófejnek egyik pályavonalról a másikra történő átmozgatását! A merevlemezek esetében az olvasófej pozicionálásával nem kell foglalkoznunk, mivel a vezérlő azt elintézi helyettünk. Sőt egyáltalán nem biztos, hogy a cilinder-szektor-fej felosztása megegyezik a lemez fizikai felépítésével, például könnyen elképzelhető, hogy a külső cilindereken valójában több szektor található, mint a belsőkön. Vagy gondoljunk csak az LBA-ra, amikor is a vezérlő számolja ki a megadott logikai címből, hogy pontosan hová is kell mozgatnia az olvasófejet. Hajlékonylemezek esetében azonban nem is álmodhatunk az LBA-hoz hasonló segítségről, és a fej pozicionálásához sem kapunk sok segítséget a vezérlőtől. Itt az operációs rendszernek közvetlenül meg kell hívnia a SEEK nevű műveletet, amely a fej pályavonalak közti mozgatására szolgál. A SEEK művelet értékeként azt kell megmondanunk, hogy a fejet melyik irányban hány pályavonallal arrébb szeretnénk tuszkolni. Ha a SEEK valamilyen oknál fogva nem járna sikerrel (például nem arra a pályavonalra kerül, ahová kell), a RECALIBRATE nevű utasítás végrehajtására is szükség van, ami a fejet kiviszi a 0. pályavonalra, majd onnan ismét elkezdhetjük a megfelelő helyzetbe való állítást.
Ezenkívül van még néhány tényező, ami nagyban bonyolítja a hajlékonylemez kezelését, de erről nem az eszköz csekély tudású vezérlője tehet. Ilyen tényező, hogy a hajlékonylemez cserélhető eszköz. Ez önmagában nem is lenne baj, csak az a gond, hogy nem tudjuk érzékelni, ha valaki két művelet között orvul kicseréli a meghajtóban lévő lemezt. Magyarán nem biztosíthatjuk, hogy a kiírandó adat arra a lemezre kerül, amelyre azt szántuk. (Felmerülhet a kérdés, hogy mi a helyzet a CD-ROM esetében, mivel az is cserélhető eszköz. Ez igaz, de ott nem léphet fel ilyen nehézség, mivel az egység vezérlője érzékeli, ha kinyitják a meghajtó ajtaját. Erre a hajlékonylemez-egység vezérlője sajnos nem képes, ha tudná, nem lenne semmi gond). Mivel azt sem tudjuk megállapítani, hogy a meghajtó tartalmaz-e éppen lemezt vagy sem, további fejfájást okozó helyzetekkel is számolnunk kell. Mi történik például, ha a felhasználó kísérletet tesz a hajlékonylemez elérésére, ám ezt megelőzően elfelejtette a meghajtóba helyezni? Valószínűleg sokan tapasztalták már az ebből adódó kellemetlenségeket: az egész rendszer addig felfüggesztett állapotba kerül, amíg a meghajtó bőszen a bent nem lévő lemez elérésével próbálkozik. Ezért van szükség különböző óvintézkedésekre, ha például a kiadott művelet bizonyos idő alatt nem fejeződik be, az egészet gyorsan el kell felejtetni a lemezkezelővel.
Ha már a meghajtóba helyezhető lemezekről volt szó, érdemes megemlítenünk, hogy két típusú hajlékonylemezt ismerünk: az 5,25 és a 3,5 hüvelykeset (inch). Ezek közül csak az 5,25-öst tudjuk hajlítani, mivel a másikat egy merev külső tok védi. A „hajlékonyabbat" ma már szinte sehol sem használják, de minden operációs rendszernek tudnia kell kezelni. Hogy mégse legyen ennyire egyszerű az életünk, mind a kétféle lemezből van kisebb (LD – Low-Density) és nagyobb (HD – High-Density) sűrűségű is, amelyek más-más kapacitásúak (a 3,5 hüvelykes LD lemezek például 720 KB-osak, míg a HD-sek 1,44 MB-osak). Ez a sokféle lemezformátum annyiban kellemetlen a lemezkezelő számára, hogy meg kell tudnia állapítani, hogy a felhasználó éppen milyen lemezt helyezett be. Ennek megállapítására több módszer is kínálkozik, például a magasabb számozású szektorok és pályavonalak útján. Akárhogy is, a formátum megállapítása időigényes feladat, sőt ha a lemez hibás szektorokkal van teletűzdelve, az sem biztos, hogy felismerhető.
A merevlemezekkel ellentétben itt az olvasófej hozzáér a lemez felületéhez, és a súrlódás miatt hamar elkophat. Ezért a lemezt forgató motort csak akkor szabad bekapcsolni, amíg lemezművelet zajlik, egyébként érdemes leállítani, és a fejet visszahúzni. Csakhogy a lemeznek a megfelelő fordulatszámra való felpörgetése és a fej előrehúzása körülbelül fél másodperces művelet. Ezért nem jó megoldás, ha a motort minden lemezművelet után kikapcsoljuk. A lemezkezelők éppen ezért úgy működnek, hogy a művelet után egy kis ideig még bekapcsolva hagyják a motort, és csak akkor kapcsolják ki, ha adott időn belül nem érkezett további kérés.
Most már kétség sem férhet hozzá, egy hajlékonylemez-kezelő tervezése nem éppen az élet örömteli feladatai közé tartozik. Még egy jelentős beviteli-kiviteli eszközről nem ejtettünk szót, mégpedig az óráról. Elsőre kicsit meglepő, hogy az óra is B-K eszköz. Pedig az, mi több, nélkülözhetetlen eszköz az operációs rendszer számára. Hogy miért, erről a következő hónapban írunk. Az óra bemutatása után befejezzük a B-K eszközöket, és a memóriakezeléssel foglalkozunk.

Garzó András (garzoand@interware.hu)
Körülbelül három éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.