Holtpont kizárva!

Sorozatunk előző részében már nem maradt hely egy sok fejfájást okozó nehézség bemutatására, ezért most rögtön be is pótoljuk a mulasztást. A továbbiakban már valóban gyakorlatibb vizekre evezünk, és a beviteli-kiviteli eszközök kezelésével foglalkozunk, a terminálokon kezdve a sort.

A nehézséget, amiről most beszélni fogunk, az okozza, hogy léteznek olyan erőforrások, amelyeket ha egy folyamat megkap és elkezd használni, egészen addig nem vehetjük el tőle, amíg be nem fejezte a munkáját. Ezek az úgynevezett kizárólagos vagy monopol módú erőforrások; példaként a nyomtatót lehetne említeni.
Akadnak azonban megszakítható erőforrások is, amelyek a folyamattól bármikor elvehetők, azzal a feltétellel, hogy ugyanabban az állapotban kapják vissza, mint ahogyan elkoboztuk tőlük. Jellemzően ilyen erőforrás a memória. Egy folyamatot bármikor felfüggeszthetünk, kipakolhatunk a memóriából a merevlemezen lévő virtuális memóriába, és az így felszabadult memóriaszeletet másvalakinek adhatjuk át. Ezután az eredeti folyamatot visszatöltve az ugyanonnan folytathatja a futását, ahol abbahagyta. Nyilvánvaló, hogy például a nyomtató esetében ezt nem tehetnénk meg.
Ha megengedjük a monopol módú erőforrások létezését, fel kell rá készülnünk, hogy előbb-utóbb valamilyen gond is lesz. Kialakulhat az esetenként a folyamatok rejtélyes lefagyásával járó rendkívül kellemetlen jelenség, a holtpont.
Általánosan megfogalmazva: valamely két vagy több folyamatból álló halmaz akkor juthat holtpontba, ha a benne lévő összes folyamat egy olyan eseményre várakozik, amelyet csak egy szintén abban a halmazban lévő folyamat idézhetne elő. Magyarán: körkörös várakozás van érvényben: „A” folyamat vár a „B”-re, az pedig a „C”-re, a „C” meg az „A”-ra.
Zavaros? Nézzünk egy példát! Az első folyamat ki szeretne nyomtatni egy állományt a CD-ROM-ról, ezért lefoglalja a CD-meghajtót mint erőforrást. Ezután megpróbálja megszerezni a nyomtatót, de azt egy másik folyamat éppen használja, tehát várakoznia kell. Az éppen nyomtató folyamat azonban be szeretne valamit tölteni a CD-ről, ami szükséges a további adatok kinyomtatásához. A meghajtó viszont már foglalt, ezért ő is várakozásra van ítélve. Mindkét folyamat egymásra vár, vagyis holtpontról kell beszélnünk.
Ha még így sem tiszta teljesen, gondoljunk csak a két résszel ezelőtt említett étkező filozófusok kérdésére! A filozófusok jelen esetben a folyamatoknak, a villák pedig monopol módú erőforrásoknak felelnek meg.
Az operációsrendszer-tervezőket már régóta foglalkoztatja, hogy milyen módszerekkel lehetne elkerülni a holtpontok kialakulását. Rájöttek, hogy a monopol módú erőforrások jelenlétén kívül három további feltételnek is teljesülnie kell, hogy holtpont jöhessen létre. Az első az úgynevezett kölcsönös kizárás, azaz minden erőforrásnak két állapota lehet: vagy hozzá van rendelve valamelyik folyamathoz, vagy nincs. A második az, hogy egy folyamat bármikor anélkül kérhet egy erőforrást, hogy el kellene engednie egy már korábban lefoglaltat. Az utolsó feltétel pedig az előbb már említett körkörös várakozást követeli meg. Ha ezek közül csak az egyik nem teljesül, sohasem alakul ki holtpont.
Rengeteg algoritmust kitaláltak, amellyel megakadályozhatnánk bármelyik feltétel teljesülését, ám a Unix-rendszerek eléggé sajátos módon oldják meg ezt a nehézséget: egyszerűen nem foglalkoznak vele.
A Unixban ugyanis nincsenek monopol módú eszközök, és az operációs rendszer nem is teszi lehetővé a zárolásukat. Mégis létezik erre egy módszer: az úgynevezett lezáró (lock) állományok. Ha egy eszközre kizárólagos elérést szeretnénk, akkor csak egy ilyen állományt kell elhelyezni a megfelelő helyre a megfelelő névvel (amely az adott eszközre utal).
Ez azonban nem az operációs rendszer szolgáltatása! Maga a folyamat dönti el, hogy figyelembe veszi-e a lezáró állományokat vagy sem. Ha az utóbbi eset áll fenn, akkor könnyen lehet, hogy végeredményül valamiféle szörnyűséget kapunk, de holtpont akkor sem alakulhat ki, mivel a kölcsönös kizárás feltétele nem teljesül.
Ennyit a holtpontokról. A továbbiakban a különböző beviteli kiviteli eszközök kezelésével foglalkozunk, elsőként a terminálokkal.

Terminálok

A terminálok olyan eszközök, amelyek lehetővé teszik, hogy a felhasználók kapcsolatot tartsanak a számítógéppel, és minden számítógép rendelkezik legalább egy ilyennel. Monitorunk és billentyűzetünk együttesen szintén egy terminált alkot. A többfelhasználós operációs rendszerek (mint például a Linux) esetében ezt konzolnak nevezik.
Érdemes megjegyeznünk, hogy a többfelhasználós operációs rendszerek esetében a többi rész közül általában a terminálmeghajtó kódja a legnagyobb, mégis az egész témakört elintézzük röpke két oldalon. Ez azzal magyarázható, hogy például a folyamatkezelésnél (amelyet több mint két részen át tárgyaltunk) a terminálkezelés elve viszonylag egyszerűbb, de a megvalósítás sokkal „macerásabb”, mivel egyrészt egyszerre kell kezelni a billentyűzetet és a képernyőt (amelyek különkülön is elég nagy kihívást jelentenek), másrészt ennél az eszköz esetében számolni kell egy eléggé beszámíthatatlan külső tényezővel is: a felhasználóval. Tapasztalatból tudjuk, hogy egy rendszer sohasem támaszthat túl nagy követelményeket a felhasználóval szemben, aki akarva-akaratlan mindenféle ármánykodásra képes, például felrúgja a bevitelre vonatkozó szabályokat, vagy több karaktert ír be, mint amennyi egy sorban kifér (vagy amennyit az adott alkalmazás fogadni képes) stb.
Az ilyen helyzeteket a terminálmeghajtónak kell kezelnie, tehát gyakorlatilag minden eshetőségre fel kell készülnie. Ha ezt nem tenné meg, akkor akár az is előfordulhat, hogy a felhasználó ügyetlenkedése miatt elszállhat az adott alkalmazás, rosszabb esetben talán az egész rendszer is.

Sokféle színű, illatú, formájú terminál létezik, az operációs rendszer szempontjából azonban csak kapcsolódási felületükben különböznek egymástól (azaz abban, hogy milyen elv alapján cserélnek adatot a központi számítógéppel). Mi ezek közül most háromfélét említünk meg.

Tárcím-leképezéses terminálok

Általában az összes számítógép rendelkezik legalább egy ilyennel. Ebbe a csoportba tartozik billentyűzet-monitor párosunk is. Az ilyen típusú terminálok mindig beépített részét képezik a számítógépnek.
Ez a csoport a nevét onnan kapta, hogy a kijelző egységet (a képernyőt) egy úgynevezett video-RAM-on keresztül érhetjük el. Ez nem más, mint egy olyan memóriatartomány, ami része a főmemóriának, és ugyanúgy kell használni (címezni), mint a memória bármelyik másik rekeszét.
A rendszernek tehát csak el kell helyeznie a megjeleníteni kívánt szöveget (vagy képet) a memória megfelelő részére, ezután az ott tárolt adatot a grafikus kártyán található videovezérlő a monitor számára emészthető videojelekké alakítja át.
A többi terminháltípustól eltérően itt a bevitel teljesen el van választva a megjelenítéstől, a billentyűzet közvetlenül az alaplaphoz kapcsolódik. Amikor a felhasználó lenyom egy billentyűt, a billentyűzet belsejében található mikroprocesszor ezt érzékeli, és elküldi az adott billentyűhöz tartozó úgynevezett scan-kódot. Amint erről az alaplapon található billentyűzetvezérlő értesül, megszakítást vált ki. Ugyanez történik az adott billentyű felengedésekor is.

RS-232 (soros kapujú) terminálok

Ezek olyan kijelzőből és egy billentyűzetből álló eszközök, amelyek a számítógép soros kapujához kapcsolódnak. A soros kapu azonban lassú, és csak korlátozott mennyiségben áll rendelkezésre, ezért manapság leginkább csak a távoli, telefonvonalon és modemen keresztüli elérés biztosítására használják.

Hálózati és grafikus (X) terminálok

Ma már ott tartunk, hogy maguk a terminálok is számítógépek (esetleg PC-k), amelyek saját processzorral és memóriával rendelkeznek. Megjelentek a grafikus környezettel is megbirkózó úgynevezett X-terminálok. Ezek többnyire hálózaton keresztül kapcsolódnak a központi számítógéphez.
Ha karakteres terminált szeretnénk „gyártani” egy, a hálózatba kapcsolt PC-ből, csupán egy úgynevezett terminálemulátor programot kell rajta elindítanunk, vagyis egy egyszerű `telnet` ügyfelet. A `telnet` a világ egyik legegyszerűbb hálózati protokollja: a lenyomott billentyűket elküldi a kiszolgáló felé, az ügyfél pedig megjeleníti a beérkező karaktereket. Az `ssh` (Secure Shell) protokoll már egy kicsit bonyolultabb, habár szerepe megegyezik a `telnet`-ével. A fő különbség annyi, hogy az átvitel titkosított, tehát biztonságban érezhetjük magunkat a hálózaton hallgatózó fülektől.
Grafikus terminált lehet készen kapni, de tulajdonképpen bármilyen általános célú számítógép használható erre a célra, ami rendelkezik hálózati csatolóval, egérrel, billentyűzettel, és természetesen grafikus megjelenítővel.
Az X-terminál minden esetben egy X-kiszolgáló nevű alkalmazást futtat, amely a billentyűzetről, illetve az egérről beérkező adatokat továbbítja a gazdaszámítógéphez, továbbá fogadja az utasításait. Ezek a parancsok viszonylag alacsony szintűek, tehát olyasmire kell gondolni, mint egy pont kirajzolása vagy egy vonal húzása.
A gazdaszámítógépen úgynevezett X-ügyfélprogramok futnak. Ezek egyrészt azok, amelyeket a felhasználó a terminálról elindít, de a terminál képernyőjén jelennek meg. Másrészt X-ügyfélnek számít még az ablakkezelő, amely a terminálon megjelenő ablakok létrehozásáért, kinézetéért, mozgatásáért, átméretezéséért stb. felel.
Láthatjuk, hogy mindhárom különböző terminháltípust másképpen kell programoznunk. A felhasználói programok azonban nyilván nem szeretnének olyan dolgokkal törődni, mint hogy az őket futtató felhasználó éppen milyen típusú, felbontású terminál előtt ül. Tehát itt is eszközfüggetlen környezet létrehozása a cél.
A továbbiakban nagy vonalakban bemutatjuk a Linuxhoz hasonló Unix-rendszerek terminálmeghajtóinak működési elvét, főként a tárcím-leképezéses terminálokra kihegyezve.

Beolvasás a terminálról

Az első kérdés, amit egy operációs rendszer tervezésekor meg kell vizsgálni, az, hogy érdemes-e kettéválasztani a beolvasást, illetve a megjelenítést. Logikusnak tűnik, ha azt mondjuk, hogy igen, mivel a képernyő és a billentyűzet különálló eszköz, ám a Linux mégis együtt kezeli őket. Mindenesetre mi most kettéválasztva tárgyaljuk őket, a bevitelen kezdve a sort.
Alapvetően kétféle beviteli módot különböztetünk meg: a nyers módot és a feldolgozott vagy a Unix-rendszerek által kanonikusnak keresztelt módot. Az első esetben a billentyűzetmeghajtó a beérkezett karaktert egyből továbbítja a felhasználói alkalmazások felé. Ez nagyon hasznos például a szövegszerkesztők esetében, viszont zavaró lehet egy parancsértelmező számára. Ha ugyanis a felhasználó elgépelte a parancsot, akkor a BACKSPACE billentyű segítségével a beírt karaktereket egészen a rosszul írt részig kitörli, majd onnantól folytatja a parancs bevitelét. Nyilvánvaló azonban, hogy a parancsértelmező nem kíváncsi a felhasználó ügyetlenkedéseire, csak a kész, kijavított sor érdekli, ezért számára a feldolgozott módú bekérés lenne rokonszenvesebb. Szerencsére a Unix-rendszerek tartalmaznak olyan könyvtári eljárásokat (vagy rendszerhívásokat), amelyek segítségével az adatokat mindkét módon be lehet kérni.
A leütött billentyűk összegyűjtéséről a billentyűzetmeghajtónak kell gondoskodnia, amely a terminálkezelő folyamat része. A beérkező karakterek fogadásáról azonban a rendszermag legalján lévő megszakításkezelő gondoskodik, amely azt egy ideiglenes tárba (pufferbe) helyezi, majd valamilyen úton-módon értesíti a billentyűzetmeghajtót az új karakter megérkezéséről.
A billentyűzet csak a leütött billentyű scan-kódját (tulajdonképpen a leütött billentyű sorszámát) küldi tovább, de ez a legtöbb felhasználói alkalmazás számára emészthetetlen, ezért a billentyűzetmeghajtónak egy úgynevezett karakterkódot kell hozzárendelnie. Ezek közül a legelterjedtebb az ASCII (American Code for Information Interchange) kódtáblázat.
De ennek mi, magyar nyelvterületen élők nem sok hasznát vehetjük akkor, ha ékes anyanyelvünkön íródott szövegekkel is szeretnénk dolgozni az adott rendszer alatt, mivel az ASCII-kódtábla nem tartalmaz ékezetes betűket. Ezért a legtöbb operációs rendszer lehetőséget ad arra, hogy maga a felhasználó határozza meg, milyen táblázat szerint történjen az átalakítás. Ezeket a táblázatokat egyébként billentyűzettérképnek vagy kódlapnak is szokás nevezni.
Ha a bevitel bekérése kanonikus módban zajlik, a billentyűzetkezelőnek további kihívásokkal kell szembenéznie. Ilyen például a visszhangzás megvalósítása, vagyis a leütött billen-

tyűknek a képernyőn történő megjelenítése. A gond abban gyökeredzik, hogy olyasmire is figyelni kell, hogy a felhasználó egy 80 karakteres sorhosszúságú terminálon 80 karakternél többet gépel-e be, vagy valamilyen különleges billentyűkombinációt használ, mint például a soremelést, a BACKSPACE-t vagy a „fájl vége” karaktert.

Láthatjuk tehát, hogy egy jól működő terminálmeghajtó megtervezése és elkészítése nem kis feladat. Mindenesetre most nagy vonalakban kövessük nyomon, hogyan kér be például egy parancsértelmező egy utasítást a felhasználótól!

Amikor a parancsértelmező várja a felhasználó utasításait, olvasni próbál az úgynevezett szabványbemenetről, ami például a konzolon lévő első virtuális terminál esetében */dev/tty1* (erre a célra egy külön könyvtári eljárás létezik). Azt, hogy a szabványbemenetnek megadott eszközfájl alatt pontosan melyik eszközt is értjük, a fájlrendszer tudja megmondani, ezért a parancsértelmező egy üzenetet küld a fájlrendszerkezelő folyamatnak, ami tartalmazza az eszközfájlt, és egy memóriacímet, ahová a beérkező adat majd kerülni fog. Ezután a parancsértelmező blokkol.

Az eszközfájlok két értéket tartalmaznak: a fő, illetve a mellék eszközszámot. Ezek határozzák meg pontosan, hogy melyik eszközről is van szó, jelen esetben melyik eszköztől kell várnunk a bemenetet. Miután a fájlrendszer megállapította, hogy a megadott eszközfájlhoz melyik fő, illetve mellék eszközszám tartozik, üzenetet küld a terminálkezelőnek, hogy valaki bevitelre vár.

Ez a folyamat azonban a másodperc törtrésze alatt zajlik le, így a felhasználó valószínűleg még egy árva billentyűt sem ütött le, tehát a terminálkezelő még nem tudja teljesíteni a kérést.

A fájlrendszer azonban nem blokkol, csupán megjegyzi, hogy van itt valahol egy folyamat, amely bevitelre vár, majd folytatja a munkát a következő kérés végrehajtásával. A parancsértelmező folyamata azonban egészen addig aludni fog, amíg a kért adat meg nem érkezik.

Amikor leütünk egy billentyűt, két megszakítás is történik: egy, amikor lenyomtuk és még egy, amikor felengedtük. A rendszer csak azokkal a megszakításokkal foglalkozik, amelyek egy billentyű lenyomásakor jönnek létre. Kivételt képeznek ez alól a módosító billentyűk (a CTRL és a SHIFT), amelyeknek a lenyomását- és felengedését is kezelni kell.

A beérkező billentyűkódokat a megszakításkezelő fogadja, és beteszi egy, a terminálkezelő számára is elérhető memóriahelyre. Ezután felébreszti a terminál-kezelőt, amely a pillanatnyi billentyűzettérkép és a lenyomva tartott módosító billentyűk függvényében meghatározza, hogy milyen karaktert kell továbbküldenie az alkalmazásnak.

A feldolgozott beviteli módnál (mint jelen esetben is) a terminálkezelő addig gyűjti a billentyűzetről érkező karaktereket, amíg egy sorvége, fájlvége vagy soremelés karakter nem érkezik. Ha ez bekövetkezik, akkor a beérkezett karaktereket a parancsértelmező által kijelölt memóriarekeszbe teszi, majd üzenetet küld a fájlrendszernek, hogy a kért művelet befejeződött. Ezután a parancsértelmező felébredhet, és elkezdheti a felhasználótól érkezett karakterek feldolgozását.

A megjelenítés

A tárcím-leképezéses terminálokon való megjelenítés megvalósítása egyszerűbb a bekérésnél, bár bizonyos tényezők ezt is rendkívül túlbonyolíthatják. Például a legtöbb Unix-rendszer esetében a konzolra is megadhatunk több virtuális terminált, ami azzal jár, hogy a terminálkezelőnek figyelnie kell, hogy mindig csak az éppen működő képernyőre írjon. Mi azonban most csak a közvetlen képernyőkiírás menetét vizsgáljuk.

Az alkalmazások a `printf` könyvtári függvény segítségével írhatnak a szabványkimenetre. Hasonlóan a bekéréshez, itt is a fájlrendszernek küldünk üzenetet, ami tartalmazza a kimeneti eszközfájl nevét, továbbá egy memóriacímet, amelyen a kiírandó szöveg megtalálható. A terminálkezelőnek a feladata elvben annyi, hogy az ezen a memóriacímen található karaktersorozatot bemásolja a video-RAM-ba, a többit a gép elintézi.

Ám ez csak látszólag ilyen egyszerű. Gondoskodni kell ugyanis a különböző különleges karakterekről és a kilépési szekvenciákról (lásd később) is. A különleges karakterekre a két legjobb példa a soremelés (CTRL-J) és a csengetés (CTRL-G). Az utóbbi karakter segítségével csengetéshez hasonló hangot csalhatunk ki a hangszóróból. Látható, hogy ez a megjelenítéstől egy teljesen különböző világ, ám ennek kezeléséről mégis a terminálmeghajtónak kell gondoskodnia.

A soremelésnél előfordulhat, hogy a képernyő aljára értünk, és mindent egy sorral feljebb kell görgetnünk. Itt az a gond merül fel, hogyha ezt a video-RAM-on belüli adatok ide-oda pakolgatásával oldanánk meg, a megjelenítés rettentő lassú lenne. Szerencsére egy kis alkatrészes segítségre támaszkodhatunk. A legtöbb videovezérlőnek megmondhatjuk, hogy a video-RAM mely pontjától kezdje a megjelenítést. Így ezt a mutatót csak egy sorral lejjebb kell állítanunk, és máris felszabadul a legalsó sor. Ám ezt nem tehetjük örökké, mert a video-RAM mérete véges. Aggodalomra azonban itt sincs ok, mert a videovezérlő a video-RAM-ot körkörösen kezeli, azaz ha elért a végére, visszatér az elejére, és onnan folytatja a megjelenítést.

Bizonyos alkalmazások igényelik, hogy kicsit nagyobb szabadságot élvezzenek a megjelenítésben, azaz lehetőségük legyen például a kurzor mozgatására vagy a karakterek tulajdonságainak (elő- és háttérszínük) megváltoztatására. Ezeket az úgynevezett kilépési szekvenciák segítségével érhetik el.

Egy kilépési szekvencia több karakterből áll, amelyből az első mindig az ESC karakter. Ezt követi maga az utasítás, amelyet a megjelenítésért felelős kódnak végre kell hajtania. Egy ilyen a következőképpen nézhet ki:

```
ESC[1;1H
```

Ez a szekvencia a kurzort az 1,1-es helyzetbe (a képernyő bal felső sarkába) mozgatja. Ezek az utasítások nem egyszerűek, sőt kapcsolóik bizonyos esetekben elhagyhatóak, tehát ezeknek a kódoknak az értelmezése nem egyszerű feladat.

A kilépési szekvenciákat egyébként az ANSI szabvány tartalmazza, tehát sok más operációs rendszer is támogatja őket, például a régi MS-DOS is (igaz, csak akkor, ha a rendszer indulásakor az ANSI.SYS állományt betöltöttük).

Ennyit a terminálokról dióhéjban. A következő részben folytatjuk a beviteli–kiviteli eszközökkel való ismerkedést. Akkor egy viszonylag nagyobb lélegzetvételű témába kezdünk bele, méghozzá a lemezmeghajtókkal való ismerkedésbe.

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.