Héjprogramozás Linux alatt – feltételek (3. rész)

Cikksorozatunk előző két részében egy apró programot fejlesztettünk lépésről lépésre.

Célunk az volt, hogy megszámoljuk, hány alkönyvtár nyílik egy adott könyvtárból. Óriásinak egyelőre nem nevezhető fejlesztésünk jelenleg itt tart:

```
1: #!/bin/sh
2: lista=`ls -l $1 | grep ^d`
3: echo "$lista"
4: echo "Összesen"`echo -n "$lista" | wc -l`"
        ➥alkönyvtár"
```

A második sorban a `lista` nevű változóban összegyűjtjük az első parancssori értékben megadott könyvtár alkönyvtárainak nevét (és számos egyéb adatát). A harmadikban ezt a listát kiíratjuk, a negyedikben pedig megszámoljuk, hány sorból áll. Ez utóbbiban azért használtuk az `echo` parancs `-n` kapcsolóját, mert a program akkor is egy alkönyvtárat számolt meg, amikor valójában egy sem volt. Rájöttünk, hogy ezt az `echo`-nak az a furcsa szokása okozza, hogy akkor is kiküld egy újsorkaraktert, ha valójában nincs is mit kiírnia. A `wc` viszont éppen az újsorkarakterek alapján tájékozódik…

Újabb gondok

Egy program fejlesztése során gyakran támad olyan érzésünk, hogy most aztán már semmi baj nem érhet minket, hiszen mindenre gondoltunk. Aztán rendszerint mégis kiderül valami. Esetünkben is ez a helyzet, sőt rögtön két gondunk is akad. Először is még semmit nem tettünk annak érdekében, hogy kiszűrjük a nem létező könyvtárakat. Márpedig a felhasználónak bármilyen badarságot „jogában áll” parancssori értékként megadni. A második gond ennél összetettebb. Az előző részben ugyan nagy megelégedésünkre szolgált az `echo -n` kapcsolójának felfedezése, hiszen kijavítottunk vele egy hibát. Csakhogy ha közelebbről megvizsgáljuk programunk működését, hamar rájövünk, hogy most meg mindig eggyel kevesebb alkönyvtárat számol meg, mint amennyi valójában van. Az egyetlen helyzet, amikor tényleg helyesen működik az, amikor nincs mit megszámolnia. „Ügyesen” megoldottuk tehát egy speciális eset kezelését, és elrontottuk az összes többiét.
De mi is tulajdonképpen a baj? Ahogy az előbb említettük, a `wc` az újsorkarakterek alapján „tájékozódik”. A `-n` kapcsolóval viszont éppen a lezáró újsor kiírását tiltottuk meg az `echo`-nak! A hiba tehát röviden, hogy a `-n` kapcsolót vagy kizárólag különleges esetben kellene használnunk, vagy egy teljesen más megoldást kellene alkalmaznunk.
Nyilván az olvasó is érzi, hogy mindkét fenti nehézség kezelése egy „mi történik akkor, ha” jellegű kérdés programszerkezeti megvalósítását igényli. Kicsit tényszerűbben: feltételes utasításokat kell beépítenünk a programba.

Feltételes utasítások héjprogramokban

Héjprogramokban egy feltételes programblokk szerkezete nagyjából a következő:

```
if logikai_kifejezés
then
  a feltétel teljesülése esetén érvényes
  ➥parancsok
else
  a hamis ág
fi
```

Bővebb magyarázatot talán csak a „logikai kifejezés” használata igényel. Héjprogramokban a legtöbb ilyen kifejezést a `test` parancs segítségével fogalmazzuk meg. Ez a parancs mindenféle vizsgálatot (numerikus és nem numerikus összehasonlításokat, fájlok vagy könyvtárak létezésének vagy tulajdonságainak vizsgálatát) képes elvégezni, és a feltétel teljesülését a visszatérési értékén keresztül jelzi. Ha a logikai kifejezés igaz, akkor a visszatérési érték nulla, ha nem, akkor egy.
Fontos kihangsúlyozni, hogy visszatérési értéke minden Linux alatt futó programnak (beleértve a héjprogramokat is) van, tehát elvileg bármilyen parancssor használható „logikai kifejezésként”. Az, hogy az esetek többségében mégis a `test` parancsot alkalmazzuk, pusztán annak köszönhető, hogy ezt a parancsot kifejezetten erre találták ki, az összes többit pedig kimondottan másra. Az is lényeges, hogy a logikai kifejezésekre támaszkodó programszerkezeti elemek igaznak kizárólag a nulla visszatérési értéket tekintik. (A logikai hamis értéknek bármely nem nulla érték megfelel.)

1. táblázat

egyenlő	`-eq`
nem egyenlő	`-ne`
kisebb	`-lt`
nagyobb	`-gt`
kisebb vagy egyenlő	`-le`
nagyobb vagy egyenlő	`-ge`

Numerikus vizsgálatok

A Unix operációs rendszernek ma van néhány – finoman fogalmazva – a számítástechnika hőskorára emlékeztető szolgáltatása. Ilyenek a `test` parancs numerikus összehasonlításra szolgáló kapcsolói is, amiket az *1. táblázatban* foglaltam össze. Nézzünk egy egyszerű példát:

```
if test $szam -eq 1
then
...
```

Ennek a feltételes szerkezetnek akkor hajtódik végre az igaz ága, ha a `szam` nevű változó tartalma egy. Ügyeljünk rá, hogy a `test` parancs a változó tartalmára és nem a nevére kíváncsi, tehát a `$` használata kötelező. Szintén fontos mozzanat, hogy a numerikus műveletek csak egész számokkal működnek. Gyakori, hogy ugyanezt a szerkezetet egy másik, valamivel talán áttekinthetőbb formában fogalmazzuk meg:

```
if [ $szam -eq 1 ]
then
...
```

2. táblázat

`-z` *karakterlánc*	A karakterlánc hossza nulla.
`-n` *karakterlánc*	A karakterlánc hossza nem nulla.
`-e` név	A megadott fájl létezik.
`-f` név	A megadott fájl létezik és egyszerű fájl.
`-d` név	A megadott könyvtár létezik.
`-r` név	A fájl létezik és olvasható.
`-w` név	A fájl létezik és írható.
`-x` név	A fájl létezik és végrehajtható.

```
1: #!/bin/sh
2: if test -d $1
3: then
4:    lista=`ls -l $1 | grep ^d`
5:    echo "$lista"
6:    if [ `echo "$lista" | wc -w` -eq 0 ]
7:    then
8:       darab=0
9:    else
10:      darab=`echo "$lista" | wc -l`
11:   fi
12:   echo "Összesen" $darab "alkönyvtár"
13: else
14:   echo "A megadott könyvtár nem létezik!"
15:   exit 1
16: fi
```

Itt mindössze annyi a változás, hogy a `test` parancsot szögletes zárójelekkel helyettesítettük. Ez valóban csak helyettesítés, tehát ugyanannak a dolognak egy másik írásmódjáról van szó, semmi többről. Ügyeljünk rá, hogy a szögletes zárójeleket mindkét oldalon minden egyébtől legalább egy szóköznek kell elválasztania, ellenkező esetben misztikus, a tényleges hibára nem igazán utaló hibaüzenetet kapunk!

Nem numerikus vizsgálatok

A `test`-nek számos egyéb hasznos szolgáltatása is van. Fájlokat, könyvtárakat és karakterláncokat is képes kezelni, illetve a különböző részfeltételeket logikai ÉS illetve VAGY műveletekkel kombinálhatjuk is. Ezekről bővebben a `test` súgóoldalán olvashatunk, de néhány fontosabbat a *2. táblázatban* is felsoroltam. Számunkra első közelítésben csak a `-d` kapcsoló fontos, ami egy könyvtár létezését vizsgálja.

A javított változat

Programunk javított változatában. Mint korábban említettem, két feltételes utasításra is szükségünk lesz. Az egyikkel rögtön a program elején azt vizsgáljuk meg, hogy a parancssori értékként megadott könyvtár egyáltalán létezik-e. Ha nem, akkor a további műveletekkel nyilván kár próbálkozni.
A másik döntési szerkezetben azt kell megvizsgálnunk, hogy volt-e legalább egy alkönyvtár. Ha nem, akkor a nulla darabszámot határozott módon kell beállítanunk, mert mint láttuk, az `echo -n` általános használata nem megfelelő.
A kód egyik lehetséges megvalósítása az *1. listában* látható.
A megadott könyvtár létezését a 2., míg a lista hosszát a 6. sorban vizsgáljuk meg. Ha nincs a megadott névnek megfelelő könyvtár, hibaüzenetet küldünk, majd 1-es visszatérési értékkel

```
1: #!/bin/sh
2: if [ -d $1 ]
3: then
4:    if [ -r $1 -a -x $1 ]
5:    then
6:       lista=`ls -l $1 | grep ^d`
7:       echo "$lista"
8:       if [ -z "$lista" ]
9:       then
10:         darab=0
11:      else
12:         darab=`echo "$lista" | wc -l`
13:      fi
14:      echo "Összesen" $darab "alkönyvtár"
15:   else
16:      echo "A megadott könyvtár nem
               ➥listázható!"
17:      exit 2
18:   fi
19: else
20:   echo "A megadott könyvtár nem létezik!"
21:   exit 1
22: fi
```

leállunk (15. sor). (A nullától különböző értékkel arra utalunk, hogy a leállás valamilyen hibaállapot miatt történt.)
A 6. sorban a `test` parancs által vizsgált értéket parancsbehelyettesítéssel álltjuk elő. A lista hosszának vizsgálatával kapcsolatban azt használjuk ki, hogyha üres, akkor a benne található szavak (`-w`) száma biztosan nulla. Ugyanez a sorokra (`-l`) vagy a karakterekre (`-c`) nem igaz, mivel az egyetlen újsorkarakter is egy sornak, és egyben egyetlen karakternek számít. Ezen a ponton természetesen számos más megoldást is használhattunk volna. A legegyszerűbb azonban az, ha felfedezzük, hogy a `test` parancsnak eleve van egy erre szolgáló kapcsolója, a `-z`. Ezzel a 6. sor egyszerűen a következő alakot ölti:

```
if [ -z "$lista" ]
```

Az olvashatóság vizsgálata

Még mindig akad egy dolog, ami eddig elkerülte a figyelmünket. Linux alatt ahhoz, hogy egy könyvtár tartalmát az `ls` paranccsal megjeleníthessük, legalább olvasási (`r`) és végrehajtási (`x`) jogosultsággal kell hozzá rendelkeznünk. Elképzelhető tehát, hogy az értékként megadott könyvtár létezik ugyan, de nem látunk bele.
Ennek a feltételnek a vizsgálatához egy újabb `if...then...else` szerkezetet kell beszúrnunk az eddigi igaz ágba (lásd *2. listánkon*).
A 4. sorban a logikai ÉS (`-a` kapcsoló) segítségével kombináltuk az olvasási (`-r`) és a végrehajtási (`-x`) jogosultság vizsgálatát. Hiba esetén a 17. sorban a 2-es visszatérési értékkel jelezzük a feladat természetét.

Büki András (buki@vuk.chem.elte.hu)
Körülbelül kilenc éve dolgozik Linux operációs rendszerrel. Állandó szerzőtársa Prof. Dr. H. V. Kuksinak, akivel a Duna vagy a Tisza partján szoktak az élet és a tudomány viselt dolgairól töprengeni.