Héjprogramozás Linux alatt (2. rész)

Ismerkedjünk meg a héjváltozókkal, a parancssori kapcsolókkal és a feltételes utasításokkal!

Cikksorozatunk előző részében egy egyszerű héjprogramot kezdtünk el fejleszteni, ami a pillanatnyi könyvtár bejegyzései közül képes volt kiválogatni az alkönyvtárak neveit.
A forráskódja valahogy így festett:

```
1: #!/bin/sh
2: ls -l | grep ^d
3: echo " sszesen"‘ls -l | grep ^d | wc -l‘"
alk nyvtÆr."
```

A harmadik sorban látható parancsbehelyettesítés hajszálpontosan ugyanezt a műveletet tartalmazza, pusztán azért, mert a bejegyzéseket meg is akarjuk számolni. Nyilván az olvasó is érzi, hogy kétszer csinálni meg valamit ugyanabban a programban nem valami elegáns megoldás. Mennyivel szebb lenne, ha a második sor kimenetét átmenetileg tárolnánk valahol, és a harmadikban már csak fölhasználnánk! Nos, ezzel el is érkeztünk a változók használatához.
A héjprogramokban – akárcsak más programnyelvekben – gyakorlatilag tetszőleges számú változót használhatunk. Ezeket nem kell külön megadnunk, mivel abban a pillanatban, amikor értéket adunk nekik, önműködően létrejönnek. Előző példánk „elegánsabb kivitelben” a következőképpen fest:

```
1: #!/bin/sh
2: lista=‘ls -l | grep ^d‘
3: echo "$lista"
4: echo " sszesen"‘echo "$lista" | wc -l‘"
   alk nyvtÆr."
```

A második sor kimenete most egy `lista` nevű változóba kerül. Ezt a tartalmat a következő sorban egy `echo` paranccsal rögtön a képernyőre küldjük, a rákövetkezőben pedig ugyanezt a parancsot használva megszámoljuk és kiíratjuk, hogy a tartalma hány soros.
Figyeljük meg, hogy az új változót minden különösebb teketória nélkül vezettük be, egyszerűen a rá vonatkozó értékadással (2. sor). Amikor azonban a tartalmához akarunk hozzáférni, a $ dollárjelet kell közvetlenül a neve elé írni (3. és 4. sor).
Szintén érdemes megjegyezni, hogy a 4. sorban alkalmazott parancsbehelyettesítés szemmel láthatóan nem gátolja a változó tartalmára való hivatkozást. Ugyanakkor mindkét érték szerinti hivatkozást kettős idézőjelek közé zártuk – erre csak akkor van szükség, ha a változó tartalmában valamilyen különleges karakter (jelen esetben új sor) is található. A kettős idézőjel megakadályozza, hogy a héj értelmezze ezeket a karaktereket. Esetünkben, ha megszüntetjük az idézőjelezést, akkor az `echo` egyetlen hosszú sorként jeleníti meg a változó valójában többsoros tartalmát, és a `wc` is csak egy sort fog érzékelni (próbáljuk ki!).
Egy héjváltozóban bármit (karaktert, egy- vagy többsoros szöveget, számot) elhelyezhetünk, maga a héj azonban mindent szövegnek tekint. A héjváltozóknak tehát nincs a hagyományos értelemben vett típusuk. Értékadásnál a szóközöket is tartalmazó karakterláncot kettős idézőjelek közé kell zárni.
A fejlesztés következő lépéseként azt kellene megvalósítanunk, hogy programunk ne csak a pillanatnyi könyvtárban működjön, hanem parancssori kapcsolóként elfogadjon egy tetszőleges könyvtárnevet. Héjprogramokban a parancssori kapcsolókra a `$1`, `$2` stb. szimbólumokkal hivatkozhatunk; a parancssorban az egyes kapcsolók tartalmát egy vagy több szóköz választja el egymástól. Ha szóközt is tartalmazó karakterláncot akarunk egyetlen kapcsolóként megadni, akkor azt kettős idézőjelek közé kell zárnunk. Programunkban – legalábbis első megközelítésben – csupán a második sort kell módosítani:

```
2: lista=‘ls -l $1 | grep ^d‘
```

A dolog csaknem tökéletes, sőt új programunk még akkor is helyesen működik, ha elfelejtünk neki könyvtárnevet megadni. Ilyenkor „hagyományos módon” a pillanatnyi könyvtárral dolgozik. (Ebben ugyebár semmi természetfölötti nincs, hiszen a kapcsolók nélkül meghívott `ls` parancs is pontosan ezt teszi.) Egyéb, különleges helyzetekkel próbálkozva azonban hamar rájövünk, hogy akad még két apró gond. Ha a megadott könyvtárból egyáltalán nem nyílnak alkönyvtárak, programunk az összesítésnél akkor is azt állítja, hogy azok száma 1; ha pedig véletlenül nem létező könyvtárat adunk meg neki, akkor az `ls` parancs szabványos hibaüzenete jelenik meg, de a program az összesítést ilyenkor is elvégzi. Az utóbbi csak csúnya, az előbbi azonban kifejezetten hiba.
A téves összegzés okára kezdő héjprogramozók igen nehezen szoktak rájönni, és ez nem véletlen. Józanul gondolkodó ember ugyanis azt hinné, hogyha egy változóban nincs semmi, akkor az `echo` kimenetén is „semmi” fog megjelenni – ez azonban csak majdnem igaz. Az `echo` ugyanis alapértelmezésként akkor is kiküld egy újsor karaktert, ha semmi dolga nem lenne. Ezt pedig a wc boldogan megszámolja, hiszen számára az üres sor is sor. Az `echo` alapértelmezett viselkedését a `-n` kapcsolóval a 4. sorban lehet letiltani:

```
4: echo " sszesen"‘echo -n "$lista" | wc -l‘"
alk nyvtÆr."
```

A hibás kapcsolók kezeléshez természetesen egy logikai elágazást fogunk beiktatni, ami aszerint működik, hogy a parancssori kapcsolóként megadott néven létezik-e könyvtár. Ez azonban már sorozatunk következő részének a témája lesz.

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.