Héjprogramozás Linux alatt

A dolgok értelme, avagy miért érdemes megtanulnunk a héjprogramozást?

Egy operációs rendszer két legalapvetőbb feladata a számítógép erőforrásainak elosztása és a felhasználóval való kapcsolatteremtés. Bármilyen operációs rendszerrel legyen is dolgunk, biztosan lesz egy olyan része, amelyik a géppel, és egy olyan, amelyik a felhasználóval képes szót érteni. Ezek természetesen egymással is beszélő viszonyban vannak, másként a rendszer nem működhetne, de ez a belső viszony a gép előtt ülő számára mindvégig rejtve marad.
A Linux esetében az eszközökkel a rendszermag (kernel), a felhasználóval a héj (parancsértelmező) tartja a kapcsolatot. A héj egyfajta tolmácsként működik: értelmezi a felhasználó által begépelt parancsokat, ellenőrzi az írásmódjukat, majd ha mindent rendben talált, elindítja a szükséges programokat.

Parancsoljunk!

Mindebben semmi érdekes nincs egészen addig, amíg csak egy-egy parancsot akarunk végrehajtani. Begépeljük például hogy `ls`, és a Linux „rögtön tudja”, hogy a pillanatnyi könyvtár tartalmát szeretnénk a képernyőn viszontlátni. Az egész valahogy úgy működik, mint egy italautomata: bedobod a pénzt, megnyomod a gombot, és – egy kis szerencsével – megkapod, amire vágytál. Némi túlzással azt mondhatjuk, hogy a Unixtól különböző operációs rendszerek alapvetően valóban így működnek. Tényleges gondokra kézzelfogható megoldásokkal szolgálnak előre megírt programok formájában. Ezek természetesen kiválóan használhatók egészen addig, amíg a felhasználónak „ötletei nem támadnak”. Ha ugyanis olyasvalamit akar megvalósítani, ami egy-egy ilyen készen kapott eszközzel már nem megy, akkor vagy megkísérli ügyesen összekombinálni azokat, vagy ha ez nem sikerül, egyszerűen megírja magának a megfelelő programot, például C-ben.
A Unix rendszerek, így a Linux fő erőssége a dolgok kombinálása. Az operációs rendszerrel kapunk egy nagy halom, jobbára teljesen „szakbarbár” segédprogramot. Némelyik betűket cserélget a bemenő szövegben mindenféle körmönfont szabályok alapján, de számolni gyakorlatilag nem tud (`sed`, `tr`). Mások a számokat is ismerik (`expr`), megint mások tulajdonképpen önálló programozási nyelvnek is tekinthetők (`awk`).
Különlegességük folytán ezekkel az eszközökkel önmagukban semmire sem mennénk. Szerencsére van azonban egy olyan rendszerszolgáltatás, a cső (jele a | karakter), ami lehetővé teszi az egyes elemek sorba kapcsolását. Ennek segítségével az egyik program kimenetét rögtön egy másik bemeneteként használjuk, anélkül, hogy az fizikai formában (például átmeneti fájlként) bárhol megjelenne. Ha például a pillanatnyi könyvtárban található bejegyzések közül ki akarjuk válogatni az alkönyvtárak neveit, így járhatunk el:

```
ls -l | grep ^d
```

Már ez a végtelenül egyszerű példa is jól szemlélteti, hogy a „csövezés” segítségével az önmagukban buta elemekből nagyon is okos rendszereket építhetünk anélkül, hogy olyan alacsony szintű műveletekkel kellene foglalkoznunk, mint a fájlok megnyitása, vagy a belőlük való olvasás.
És ezzel el is érkeztünk most induló cikksorozatunk témájához, a héjprogramozáshoz, ami tulajdonképpen nem egyéb, mint az előbb említett „csövezés” egy kissé hivatalosabb megnevezése.

Írjunk egy egyszerű héjprogramot!

A héjprogramok első közelítésben egyszerű szövegfájlok, amelyek a héj számára értelmes utasítássorokat tartalmaznak. Ezeket akár a parancssorba is begépelhetnénk, de ha hosszúak vagy sokszor kell az adott műveletsort használnunk, célszerűbb belőlük héjprogramot írni. A fenti egyszerű példánál maradva a könyvtárválogató trükköt a következőképpen alakíthatjuk állandó segédeszközzé (a sorokat csak a jobb megértés miatt számoztuk):

```
1: #!/bin/sh
2: # Az alk nyvtÆrak neveinek listÆzÆsa
3: ls -l | grep ^d
```

Az első sor annak a parancsértelmezőnek a nevét és teljes elérési útvonalát adja meg, amelyik képes értelmezni a leírtakat. Linux esetében ez csaknem mindig a Bash, a */bin/sh* itt ugyanis valójában egy erre a héjra mutató hivatkozás. A második sor megjegyzés, amit a kettős kereszt (# karakter) után írhatunk. Ennek nem muszáj a sor elején kezdődnie, de a szöveg csak úgy lehet több soros, ha mindegyik elejére kiírjuk a kettős keresztet. A lényeget a harmadik sor tartalmazza. Hozzunk tehát létre egy tetszőleges szövegszerkesztővel egy ilyen tartalmú szövegfájlt. Legyen a neve, mondjuk *konyvtar.sh*. Ahhoz, hogy futtatni is tudjuk, végrehajtási jogot kell neki adnunk:

```
chmod 700 konyvtar.sh
```

Ezzel el is készült első héjprogramunk. Próbáljuk ki! Ahhoz, hogy bárhonnan elérhető legyen, természetesen egy olyan könyvtárban kell elhelyeznünk, amelyik szerepel a `PATH` környezeti változóban.
Mindez természetesen még csak a kezdet. A héjprogramok fejlesztése során számos, más programozási nyelvben is megtalálható vezérlési szerkezetet lehet és kell használnunk. Létrehozhatunk ciklusokat, döntési szerkezeteket, átmeneti fájlokat, kezelhetjük a parancssort és társaloghatunk a felhasználóval. Cikksorozatunkban az alapok tisztázása után az olyan elvontabb, de általánosan használható módszerek kerülnek sorra, mint a zárolás, a parancssori kapcsolók kezelése, vagy az önhívó programok írása.

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.