Bevezetés a szabályos kifejezések használatába (1. rész)

A Linuxot úgy is használhatjuk, mint bármelyik más grafikus operációs rendszert. Aki azonban „érezni akarja az erőt”, annak meg kell tanulnia varázsolni a kódokkal!

Számtalanszor olvashattuk már azt az állítást, amely szerint a Unix operációs rendszert programozók írták programozóknak. Hogy ez mennyire igaz, azt a Unix- és a Linux-rendszerek számos tulajdonsága és viselkedésformája is alátámasztja. Az „igazi” linuxos például komoly munka közben nem az egérrel kattogtat, hanem a közönséges földi halandók számára igencsak meglepő dolgokat irkál a parancssorba. Néha annyira furcsákat, hogy egyes esztéták szerint azokat látván az ember önkéntelenül és „elzöldült képpel kapott a légibetegség esetén használatos zacskó után”. (Az idézet ***David Gelernter*** „Ami működik, az csodálatos” című könyvéből származik, amely alcíme szerint a technika esztétikájáról szól.)

Lássuk csak, mi zavarhatja meg ennyire ezeknek az ártatlan embereknek az egyensúlyérzékét. Az első furcsaság számukra valószínűleg az lehet, hogy Linux alatt a feladatok megoldására szolgáló – néha valóban kissé ördögi – gépezeteket maguknak kell apró építőelemekből összeállítaniuk. Ezt hívjuk héjprogramozásnak, amiről egy egész cikksorozatot olvashat az, aki végiglapozza a Linuxvilág előző néhány számát. Abból az egyszerű felismerésből kiindulva, hogy az emberek többsége szeret legózni, nem valószínű, hogy a fent említett gondot egyedül a héjprogramozás okozná – kell itt lennie még valaminek.

A másik „nagy zavarkeltő” a szabályos kifejezések használata lehet. Ezekkel gyakorlatilag tetszőlegesen bonyolult, a keresés-sel, helyettesítéssel vagy más ezekhez hasonló szövegfeldolgozási műveletekkel kapcsolatos utasítás, algoritmus tömören – általában egyetlen sorban – leírható. Az általánosságnak azonban ára van: egy szabályos kifejezésekből felépített, logikai értelemben „gyönyörű” feldolgozási utasítás a kívülállók és a kezdő felhasználók számára rémálom. Az ember az első pillanatban általában azt se tudja, melyik végén kezdje el olvasni.

A nehézségeket csak fokozza, hogy a szabályos kifejezésekre támaszkodó segédprogramok (itt elsősorban a `grep`, `sed` és `awk` nevű eszközökről van szó) dokumentációja tartalmazza ugyan a megfelelő útmutatásokat, de egyáltalán nem olyan stílusban, ami igyekezne a kezdőket átsegíteni a dolog nehezén. Ezek a leírások inkább csak azok számára nyújtanak lexikonszerű segítséget, akik a dolog lényegével már tisztában vannak, vagyis legalább nagyjából tudják, hogy mit keresnek.

Az írásainkban bemutatott példák ezzel szemben fokozatosan nehezednek, és kifejezetten a gyakorlatra összpontosítva igyekeznek bevezetni az olvasót a szabályos kifejezések használatába.

A szabályos kifejezések kézzelfogható elemei

Gyakorlatias szempontból közelítve meg a kérdést a szabályos kifejezések arra szolgálnak, hogy a segítségükkel egy szöveg bizonyos tulajdonságokkal rendelkező részeit azonosíthassuk. Az azonosítás célja lehet keresés (`grep`) vagy valamilyen feldolgozás (`sed` és `awk`).

Tekintve, hogy egy szöveg legkisebb építőeleme egyetlen karakter, a legegyszerűbb szabályos kifejezés is egyetlen karakterből áll, ami – nem túl meglepő módon – önmagára illeszkedik. Amennyiben tehát végrehajtjuk a

```
cat szöveg.txt | grep "a"
```

műveletet, akkor a keresésre szolgáló és szabályos kifejezésekkel vezérelhető `grep` parancs a szövegnek minden olyan sorát megjeleníti, amelyben bárhol legalább egy „a” betű szerepel.

A következő egyszerű szabály szerint ha két szabályos kifejezést egymás mellé írunk, akkor ismét szabályos kifejezést kapunk. Ennek szellemében a

```
cat szöveg.txt | grep "ablak"
```

parancs az összes olyan sort megjeleníti, amelyben legalább egyszer szerepel az „ablak” szó. Ezen a ponton fontos megjegyezni, hogy a szabályos kifejezés illeszkedésének nem feltétele az, hogy a keresett szó magában álljon, vagyis az „ablak” illeszkedik az „ablaktörlő” és a „vakablak” szavakat tartalmazó sorokra is.

A szabályos kifejezések elvont elemei

Előfordulnak olyan esetek, amikor nem ennyire egyértelműen egyetlen szó vagy karakterlánc formájában írható le az a dolog, amit keresünk. Például előfordulhat, hogy kizárólag „a” betűvel kezdődő, ötbetűs szavakat akarunk keresni. Ez első látásra ugyan egyáltalán nem tűnik bonyolultnak, de mint hamarosan látni fogjuk, arra kiválóan alkalmas, hogy rajta keresztül bemutassuk a jól használható szabályos kifejezések megfogalmazásához szükséges gondolatmenetet.

Először is minden olyan sajátságot fel kell sorolnunk, ami a keresendő objektumot egyértelműen azonosítja. Kis „a” betűvel kezdődő pontosan ötbetűs szavakat keresünk, vagyis olyan karakterláncokat, amelyek első eleme az „a” betű, amit pontosan négy másik, amúgy tetszőleges karakter követ.

A szabályos kifejezésekben az „egy darab akármi” jele a pont (.). Ez tehát pontosan egy, de tetszőleges karakterre illeszkedik, ami esetünkben azt jelenti, hogy az „a betűvel kezdődő ötbetűs szó” fogalma a

```
"a...."
```

szabályos kifejezéssel írható le. Vagy mégsem? Hozzunk létre egy szövegfájlt a következő tartalommal:

```
Az almák édesek.
A vadalma fanyar.
```

És most nézzük, melyik sorra illeszkedik a fenti szabályos kifejezés:

```
cat szöveg.txt | grep "a...."
```

Mindkét sor megjelent a `grep` kimenetén, vagyis mindkettőre illeszkedett a logikai minta – pedig a másodikban sem ötbetűs,

A szabályos kifejezések alapelemei

Elem	Leírás.
karakter	A szabályos kifejezések legkisebb építőeleme. Bármely karakter önmagára illeszkedik.
\karakter	Olyan karakter, amely a szabályos kifejezésekben absztrakt jelentéssel is rendelkezik (pl.: ^ , $, [ ), de az adott helyen literális értelemben kívánjuk használni.
[lista]	Egyetlen karakter, amely a megadott lista bármely eleme lehet.
[^ lista]	Egyetlen karakter, amely a listában felsoroltakon kívül bármi lehet.
[a-z]	Bármilyen kisbetű (kivéve a magyar ékezetes karaktereket).
[A-Z]	Bármilyen nagybetű (kivéve a magyar ékezetes karaktereket).
[a-zA-Z]	Bármilyen betű (kivéve a magyar ékezetes karaktereket).
[0-9]	Bármilyen számjegy.
^	Sorkezdet.
$	Sorvég.

sem „a” betűvel kezdődő szó nincs. Ugyanakkor a „vadalma” szóban két olyan részlet is van („adalm” és „alma”), amely „a” betűvel kezdődik és azt négy karakter követi. Világos tehát, hogy nem elég egyértelműen fogalmaztuk meg a keresési feltételt.
Ha egy szó valóban pontosan ötbetűs, akkor előtte és utána is szóköznek kell állnia. Az előbbinél tehát sokkal alkalmasabb a

```
cat szöveg.txt | grep " a.... "
```

szabályos kifejezés; ez már valóban csak a megfelelő sort jeleníti meg a kimeneten.

További bonyodalmak

Eredeti célunknál maradva finomítsuk tovább egy kicsit a keresés szempontjait! Például meg lehet-e azt oldani, hogyha egy ötbetűs, „a” betűvel kezdődő szó a mondat elején található, és ezért nagybetűvel kezdődik, akkor is rátaláljon a `grep`? Vagy mi történjen akkor, ha egy ilyen szó a mondat végére kerül, és így nem szóköz, hanem pont van utána?
Mindezek kipróbálására az előbb használt példaszöveget egészítsük ki három újabb sorral:

```
Az almák édesek.
A vadalma fanyar.
A kép címe: Almák és körték.
És megint almák.
Langyos a sör.
```

Ha az illesztendő minta egy pontján vagylagosan többféle lehetőséget akarunk megengedni, szögletes zárójelek között kell felsorolnunk őket. Az `"[abcd]"` szabályos kifejezés tehát az ábécé első négy betűje közül bármelyikre illeszkedik, de továbbra is csupán egyetlen karakternek felel meg, tehát vagy egyetlen „a” betűnek, vagy egyetlen „b” betűnek és így tovább. Ha az összes kisbetűre így akarunk hivatkozni, azt az `"[a-z]"` formájú kifejezéssel tehetjük meg. Természetesen az összes nagybetűre is hivatkozhatunk ilyen módon, az `"[a-zA-Z]"` szabályos kifejezés pedig valamennyi betűnek megfelel.
Ennek fényében ha az ötbetűs szó elején a kis és nagy „a” betűt is meg akarjuk engedni, a végén pedig a szóközt és a mondatvégi pontot is, akkor a következő szabályos kifejezést szükséges használnunk:

```
cat szöveg.txt | grep " [Aa]....[ \.]"
```

A kifejezés végén a pont előtt egy \ is szerepel. Erre azért van szükség, mert ennek a karakternek a szabályos kifejezésekben különleges jelentése van (lásd fent), és valahogyan jeleznünk kell, hogy itt betű szerinti értelemben (literálisan) használjuk, vagyis tényleg pontot keresünk.
Most már minden nagyon szép és jó, csak egy gond van még: langyos a sör…
Ez a mondat is megjelent a kimeneten, pedig nincs benne öt betűből álló, „a” betűvel kezdődő szó. Vagy mégis? Nos, ami azt illeti, van benne egy olyan rész, hogy „ a sör. ”. Ez pont öt karakter, amit egy szóköz előz meg és egy pont követ. És mivel ráadásul az „a” betű az első, illeszkedett rá a megadott szabályos kifejezés. Igaz ugyan, hogy a közbenső betűk között van egy szóköz is, de azt nem is kötöttük ki eddig semmilyen formában, hogy az ilyen „ötbetűs” karakterláncokat nem szeretjük. Oldjuk meg tehát ezt is:

```
cat szöveg.txt | grep " [Aa][^0-9 ][^0-9 ]
➥[^0-9 ][^0-9 ][ \.]"
```

(Khm... Hova is tettem azt a zacskót?)
A `[^0-9 ]` felsorolás természetesen a számjegyeket és a szóközt jelenti. A nyitó szögletes zárójelet közvetlenül követő ^ karakter ugyanakkor a tagadás jele, vagyis ez a kifejezés mindenre illeszkedik, ami nem szám és nem szóköz.
Az olvasó most nyilván azt kérdezi, hogy minek kellett ezt ilyen faramuci módon megfogalmazni. Miért nem írtuk egyszerűen azt, hogy `"[a-zA-Z]"`? Nos, van itt egy kis gond. Az a helyzet, hogy a Unix világában a magyar ékezetes betűk nem számítanak betűnek, vagyis nem elemei az imént említett halmaznak. Viszont szerencsére nem is számok, így az általunk alkalmazott körmönfont megfogalmazás éppen megfelelő.
A következő kérdés az lesz, hogy feltétlenül le kell-e írni az illető halmazt négyszer egymás után, ha négy ilyen karaktert keresünk? És mi van, ha nem négyet, hanem százhuszonhetet akarunk azonosítani? Természetesen létezik egyszerűbb írásmód, de erről egy kicsit később lesz szó.

További elemek

A szabályos kifejezések jelkészletében létezik egy-egy elem a sor elejének és végének azonosítására is. A sorkezdetet a ^, a sorvéget a $ karakter jelöli. Ha tehát egy szabályos kifejezés elé odaírjuk a ^ jelet, azzal kikötjük, hogy az illesztést mindenképpen a sor elején akarjuk elvégezni. Hasonlóan a $ a sor végén illeszt. (Igen, egy kissé zavaró, hogy a ^ két különböző dolgot is jelent, de ennek nyilván történeti okai vannak.)
Fontos hangsúlyozni, hogy a sorkezdet- és sorvégjeleknek önmagukban is van értelme, bár karaktereket tartalmazó felsorolásokban azért nem szerepelhetnek. Így a

```
cat szöveg.txt | grep "^$" | wc -l
```

feldolgozási sor például megszámolja, hogy a bemenő szövegben hány teljesen üres sor van, a

```
cat szöveg.txt | grep -v "^$"
```

pedig az ilyen sorokat kiszűri a bemenetből, és csak a tömörített szöveget jeleníti meg. (A `grep -v` kapcsolója a tagadás jele, vagyis a megadott szabályos kifejezésnek nem megfelelő sorok fognak megjelenni a kimeneten.)
Az imént érintőlegesen említettem ugyan, de azért nem árt még egyszer hangsúlyozni, hogyha egy keresendő karakter szabályos kifejezés részeként is előfordulhat, akkor `\` karakternek kell megelőznie, jelezvén, hogy literálisan, vagyis betű szerinti értelmében használjuk. Ha tehát például a befejezetlen mondatot jelző három egymás követő pontot akarunk keresni egy szövegben, akkor a

```
cat szöveg.txt | grep "\.\.\."
```

kifejezést kell használnunk. Ha lefelejtjük a három `\` karaktert, akkor a `grep` három tetszőleges karaktert fog keresni, és így minden olyan sort megjelenít, amelyik legalább három karaktert tartalmaz.

Néhány példa

A szabályos kifejezésekről eddig tanultakat táblázatban foglaltam össze. Zárásként nézzünk meg néhány olyan gyakorlatias példát, amelyeket már ennyi ismeret birtokában is képesek vagyunk megoldani.
Ha sokat gépelünk, néha megesik, hogy elfelejtjük kitenni a mondat végére a pontot. Ráadásul az ilyen hibát utólag is elég nehéz észrevenni. Ugyanakkor a következő egészen egyszerű szabályos kifejezés segítségével kiszűrhetők azok a sorok, amelyekben ezt a hibát valószínűleg elkövettük:

```
cat szöveg.txt | grep "[a-z] [A-Z]"
```

Láthatóan azt használjuk ki, hogyha a mondat végéről hiányzik a pont, akkor egy kisbetűvel végződő szót egy szóköz után rögtön egy nagybetűs követ. A módszer ugyan nem tökéletes, hiszen ha tulajdonnév szerepel egy mondatban, akkor is lesz benne ilyen részlet. Ugyanakkor a valóban hibás sorok biztosan fönnakadnak a rostán. (Ha azt akarjuk, hogy ékezetes betűkkel is működjön a szűrés, akkor a korábban említett trükköt kell alkalmaznunk.)
Hasonlóan gyakori hiba a „z” és „y” felcserélése, ami „ay” és „egz” karakterláncok felbukkanását vonja maga után. Ezeket a következő egyszerű `sed` programmal cserélhetjük „magyar megfelelőjükre”:

```
cat szöveg.txt |\
sed "s/ ay / az /g
     s/ Ay / Az /g
     s/ egz / egy /g
     s/ Egz / Egy /g"
```

A csere miatt most a kis és a nagy kezdőbetűs alakok összevonására nem használhattuk a `" [Aa]y "`, illetve a `" [Ee]gz "` szabályos kifejezéseket. Bár már így is négy különböző cserét írtunk elő, a módszer mégsem tökéletes, mert nem fogja érzékelni a sor elején, illetve a sor végén található hibásan írt alakokat. (Ezek előtt, illetve mögött nem lesz meg az előírt szóköz.)
A teljes megoldás tehát – legalábbis jelenlegi ismereteink birtokában – tizenkét cseréből állna.
Végül nézzünk egy olyan példát, amely kigyűjti egy szövegből az elektronikus levélcímeket. Ennek a legegyszerűbb megvalósítása az lenne, ha egyszerűen a `grep` segítségével a @ karakterre végeznénk keresést. Ugyanakkor ez a megoldás eléggé spártai kinézetű kimenetet eredményezne, hiszen a `grep` találat esetén csak egész sorokat tud megjeleníteni. Ha tehát a címek nem külön sorokban szerepelnek, akkor az utómunkálatokat nekünk magunknak kellene elvégezni.
Használjuk ki tehát, hogy az `awk` soron belül is képes vizsgálódni, és ugyanúgy ismeri a szabályos kifejezéseket, mint a `grep`. Ha feltételezhető, hogy a feldolgozandó szövegben nemcsak elektronikus levélcímekben szerepel a @ karakter, akkor egy kifinomultabb keresési feltételt is használhatunk: egy levelezési címben a @ előtt és után is betűnek kell lennie. Mindezek fényében a következő egysorost használhatjuk a válogatás elvégzésére:

```
cat szöveg.txt | awk '/@/ {for(i=1;i<=NF;i++)
if($i~["[a-zA-Z]@[a-zA-Z]") print $i }'
```

A feldolgozási program előtt két / jel között megadott szabályos kifejezés a sorok címzésére szolgál. A program tehát eleve csak azokat a sorokat fogja vizsgálni, amelyekben – egyelőre bármilyen formában – szerepel a @ karakter. Az ilyen sorokon mezőnként végigmegy, és megvizsgálja, hogy melyikre illeszkedik a `"[a-zA-Z]@[a-zA-Z]"` szabályos kifejezés. (Az `awk` nyelvben az illeszkedik jele a ~ karakter.) Csak azokat a mezőket jeleníti meg, amelyeknél ez a feltétel teljesül.
Hasonlóan érdekes feladat egy szövegből a pénzösszegek kigyűjtése, például könyvelési célzattal. Az árakat arról lehet felismerni, hogy egy számot egy, a „Ft” karakterpárost is tartalmazó szó követ. Ez utóbbi ragozott alak is lehet (például „500 Ft-ba került”), illetve eshet a mondat végére is. Lássuk a válogatást végző programot:

```
#!/bin/sh
cat $1 | awk \
'/Ft/  {
  elozo=$1
  for(i=2;i<=NF;i++)
  {
    if($i~"Ft" && elozo~"[0-9]") print elozo, "Ft"
    elozo=$i
  }
}'
```

Tekintettel a feldolgozás összetettségére, ezt a példát már célszerűbb volt a fent látható módon héjprogram formájában megvalósítani. Ismét csak azokkal a sorokkal foglalkozunk, amelyekben legalább egyszer előfordul valahol a „Ft” karakterkettős. A feldolgozás lényege az, hogy a sor mezőit páronként vizsgáljuk. Ha a pillanatnyi mezőben felismerjük a „Ft” részletet (`$i~"Ft"`), az előző mezőben pedig van számjegy (`elozo~"[0-9]"`), akkor kiíratjuk az előző mezőt. A módszer egyetlen hibája, hogy ha az ár éppen egy sor végére, a „Ft” jel pedig a következő sor elejére kerül, akkor nem működik.
Némi további programozással természetesen ez is megoldható.

Büki András (buki.andras@insilico.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.