Operációs Rendszerek Gyakorlat

Második zh. – felkészülés, gyakorlás

2017 május 2. - Berta Árpád

1. feladat

Készíts egy **bash** scriptet, amely a paraméterről kapott szóközzel elválasztott kifejezések közül kiírja a szabályos születési dátumokat. Elfogadható dátumnak számít, amely 1900-2017 közötti évszámokat tartalmaz. Alakja pedig olyan, hogy pontokkal vannak elválasztva és számokkal vannak ábrázolva a hónapok (hónap, nap két számjegyű ábrázolás). Azzal most nem foglalkozunk, hogy egy hónap 28,30,31 napos-e, így a szökőévekkel sem. Feltesszük, hogy minden hónap esetében elfogadható a 31 nap. Arra viszont figyelj oda, hogy a dátum pontosan ilyen lehet, tehát bővebb dátumok sem fogadhatóak el.

```
?> ./1_beapait.sh 2010.01.01 2020.01.01 2017.12.31 1900.09.12 1920.1.10
1920.10.10 1920.10.102 1920.13.10 1920.12.32 1920.12.00 1920.00.10
1920A10A10 1920-10-10
2010.01.01
2017.12.31
1900.09.12
1920.10.10
```

Megoldás:

Az ilyen feladatokban, ahol különböző megfelelési szabályok kerülnek elő, amelyek befolyásolják, hogy mit írjunk ki, akkor a szabályoknak megfelelő reguláris kifejezést kell készítenünk. Mivel kihangsúlyozom, hogy bash scriptet kell írni, így ezekben a feladatokban közvetlenül nem írható awk script. Másrészt a paraméter átadást nem tanultuk awk esetében így ezért is nehéz lenne azzal megoldani.

A reguláris kifejezések illesztésére a bashban az **egrep** parancsot tanultuk. Úgyhogy ezt kell használnunk a feladat megoldásához.

Amit kiindulásképpen tudnunk kell az egrepről az az, hogy **a sorok esetében tud illesztést vizsgálni**. A példában viszont paraméterről kapjuk a dátumokat, amiket ugye a bash scriptünkben majd a **$*** változó nyomán tudjuk egy listában elérni. Ez a lista viszont egy szóközzel elválasztott lista, nekünk pedig olyan lista kell, amely soronként egy paramétert tartalmaz. Több különböző módon el lehet ezt érni. Viszont most csak a legegyszerűbb módszert fogom bemutatni: for ciklussal bejárjuk a paraméterek listáját és echoval kiírjuk a paramétert.

```
#! /bin/bash
for param in $*; do
 echo $param
done
```

Ha ezt kipróbálod láthatod, hogy soronként lett kiírva az összes paraméter. Ezután nincs más dolgunk, mint az echo parancsok kimenetét átirányítani egy egrepre, amely ugye kiírja azt a

szöveget, amit az echoval generáltunk, ha az általunk megadott szabály szerint megfelelő dátumformátumot tartalmaz.

A következő lépés a reguláris kifejezés megadása. A ilyen feladatok fő részét ez képezi, tehát amennyiben csak a szabály egyes részeit tudod jól megadni, akkor is szerezhető részpontszám.

Első lépésben próbáljunk meg az évszámra illesztést megadni. Gondoljuk végig milyen szabályt adhatunk meg erre? Első nekifutásra gondolhatunk arra, hogy olyan módon adjuk ezt meg, hogy az első két számjegy az lehet 19 vagy 20 és ezt 2 tetszőleges 0-9 szám kövessen: '(19|20)[0-9]{2}'. Ezzel az a probléma, hogy ilyen esetben 2099 is átmegy a szűrőn, ami viszont már nem elfogadható ebben a feladatban. Ilyen esetben úgy érdemes tovább bontani a szabályt, hogy a 2000-es évekre külön szabályt adunk meg és azt hozzá vagyoljuk az 1900-as évekre vonatkozó szabállyal. Tehát olyan szabályunk lesz az évekre, hogy '19[0-9]{2}|20[01][0-9]'. Ezzel továbbra sem vagyunk készen, hiszen a 2019 is beleesik az elfogadott évszámokba, ami pedig nem elfogadott a feladatban. Tehát amint láthatjuk a 2000-es éveket is tovább kell bontani. Az évekre vonatkozó szabály tehát a következő lesz: **19[0-9]{2}|200[0-9]|201[0-7]**

```
#! /bin/bash
for param in $*; do
 echo $param | egrep ”19[0-9]{2}|200[0-9]|201[0-7]”
done
```

A különböző részeket zárójeleznünk kell majd, mert a konkatenáció erősebb művelet a vagynál. Ezért az **évszámra vonatkozó kifejezést zárójelbe tesszük**. Ezután egy pont következik, ahol nagyon fontos odafigyelnünk arra, hogy a reguláris kifejezések linuxos jelölése esetén a pont az egy tetszőleges karaktert jelent, így jelölni kell, mint speciális karakter, amit hagyományos értelemben szeretnénk használni. Tehát **\.**

```
#! /bin/bash
for param in $*; do
 echo $param | egrep ”(19[0-9]{2}|200[0-9]|201[0-7])\.”
done
```

A következő lépés a hónap megadása. Itt az évszámhoz hasonló megfontolásokból a 0-val kezdődőekre és az 1-gyel kezdődőekre külön szabályt kell megadni és azokat vaggyal elválasztani. 0 esetében a második számjegy, ugye nem lehet 0 hiszen nincs 0. hónap. Az 1 után pedig ugye csak 0,1,2 jöhet, ugyanis a 12. az utolsó hónap. Ezeket is zárójelbe tesszük és egy ponttal folytatjuk. Tehát a reguláris kifejezés a következőképpen folytatódik:

```
#! /bin/bash
for param in $*; do
 echo $param | egrep ”(19[0-9]{2}|200[0-9]|201[0-7])\.(0[1-9]|1[0-2])\.”
done
```

A nap szabálya lesz a következő lépés. Mivel engedményt tettünk így ez 01-31 között vehet fel értéket. Hasonlóan az előzőekhez külön szabályt kell alkotni, minden olyan első számjegyre, ami után nem 0-9-ig jöhet egy tetszőleges szám. Ilyen a 0, ugyanis azután nem jöhet újabb 0, és ilyen a 3, hiszen utána csak 0 és 1 jöhet. Ezt is bezárójelezzük és a reguláris kifejezésünk így néz ki:

```
(19[0-9]{2}|200[0-9]|201[0-7])\.(0[1-9]|1[0-2])\.(0[1-9]|[12][0-9]|30|31)
```

Egyetlen utolsó simítás van még hátra. Ugyanis ez még elfogad olyan dátumokat, amelyek megfelelnek a fenti szabálynak, de utána vagy előtte egyéb karakterek vagy szöveg található. Az ilyen esetek általános kiküszöbölése az az, hogy megköveteljük, hogy a szabállyal kezdődjön és végződjön a sor. Ez esetben nem lehet más előtte vagy utána. Ezt ugye a sor elején ^, a sor végén pedig **$** jelöljük.

A végeredmény tehát:

```
#! /bin/bash
for param in $*; do
  echo $param | egrep "^(19[0-9]{2}|200[0-9]|201[0-7])\.(0[1-9]|1[0-2])\.
(0[1-9]|[12][0-9]|30|31)$"
done
```

2. feladat

Írj **AWK** szkriptet, amely paraméterként egy fájlt vár. A fájl a következő formátumú sorokból áll: <ország>;<tájegység>;<kiinduló pont>;<célpont>; A feladat: Kiírni az országokat az őket érintő útvonalak kezdő és végpontjainak neveinek a felsorolásával. Majd egy üres sor kihagyását követően a tájegységeket kiírni az őket érintő útvonalak összhosszai alapján. Figyelj oda, hogy nem csak a példában bemutatott négy sorra, néhány tájra vagy néhány országra kell működnie, hanem tetszőleges hasonló felépítésű bemeneti fájlra.

```
?> cat tajak.txt
Magyarország;Börzsöny;Nagybörzsöny;Márianosztra;18
Románia;Kelemen-havasok;Maroshévíz;Pietros;25
Magyarország;Mátra;Gyöngyös;Mátraszentistván;23
Magyarország;Börzsöny;Diósjenő;Nagy-hideg-hegy;12
?> ./3_beapait.awk tajak.txt
Országok
Magyarország: Nagybörzsöny-Márianosztra Gyöngyös-Mátraszentistván
Diósjenő-Nagy-hideg-hegy
Románia: Maroshévíz-Pietros

Tájegységek
Börzsöny: 30
Kelemen-havasok: 25
Mátra: 23
```

Megoldás:

A feladatban awk scriptet kell írnunk. Azokban a feladatokban, ahol több különböző dologhoz kapcsolódóan kell valamit összeszámolnunk, vagy szöveget összefűznünk, ott mindig tömbbel kell megoldanunk feladatot. Az, hogy létrehozunk országonként külön változót nem jó megoldás. Nyilván az sem jó megoldás, ha tájanként létrehozunk egyet. Tehát a tájaknak és az országoknak is külön tömböt kell létrehozunk. A tömböt az országok illetve a tájak neveivel kell majd indexelnünk. Tehát legyen egy **orszagok tömb**, amibe pl orszagok[”Magyarország”] vagy pl orszagok[”Románia”] címkét fogunk használni. Ehhez hasonlóan lesz egy **tajak tömb**ünk.

Mint minden awk feladatot ezt is azzal kell kezdenünk, hogy megvizsgáljuk a bemeneti állomány felépítését. Tehát, hogy az awk miként tudja majd ezt mezőkre bontani nekünk. Az alapértelmezett elválasztójel a szóköz, ehelyett itt viszont a pontosvessző az elválasztó jel. Tehát **az FS változót át**

kell állítanunk erre. Ezt pedig a BEGIN blokkban szokás elvégezni.

```
#! /usr/bin/awk -f
BEGIN {
  FS=";"
}
```

Ha ez megvan akkor a $1 változóban fogjuk tudni minden sorban elérni az országot, $2 a tájegység, $3 a kiindulási pont, $4 az érkezési pont, $5 pedig az útvonal hossza.

Gondoljuk végig azt, hogy mit kell összegyűjtenünk a feladatban. Országonként az útvonalak kezdő- és végpontjának a nevét. Ehhez a tömbértéknek szövegesnek kell lennie, amely tartalmazni fogja az említett mezők stringjeit kötőjellel illetve szóközzel elválasztva. A feladat tehát az, hogy minden sorban fűzzük hozzá az útvonal stringjét az eddig az országhoz eltárolt szöveghez. AWK-ban nem szükséges kezdőérték adása, alapból üres szövegről indulunk és első lépésben ahhoz fűzünk hozzá. Az összefűzés művelet a sima egymásmellé írással történik.

```
#! /usr/bin/awk -f
BEGIN {
  FS=";"
}
{
  orszagok[$1]=orszagok[$1]""$3"-"$4" "
}
```

Tehát, mivel nincs megadva semmilyen minta, ezért a most hozzáadott blokk minden sorra egyesével le fog futni. Soronként az *orszagok* tömböt a ország nevével indexeljük és az értéke, pedig a korábbi értékhez hozzáfűzött 3. és 4. mező tartalma lesz, tehát a kezdő- és a végpontja az útvonalnak egy kötőjellel elválasztva. A végére még hozzáfűzünk egy szóközt, hogy amennyiben lesz folytatás, akkor azt elválasszuk ezzel.

A feladat második része a tájegységeken húzódó útvonalak összeszámolása. Tehát, amennyiben egy tájegységen több különböző útvonal is húzódik, akkor azoknak a hosszát kilométerben összegezzük a tömb tájegységhez tartozó indexe értékében.

```
#! /usr/bin/awk -f
BEGIN {
  FS=";"
}
{
  orszagok[$1]=orszagok[$1]""$3"-"$4" "
  tajak[$2]+=$5
}
```

Tehát most a tájegység elnevezése lesz a tömb indexe, és így mindig az azonos tájegységhez tartozó km-eket összegezzük.

A feladatból az van még hátra, hogy a végén kiírjuk az eredményeket. Ezt az END blokkban tudjuk megtenni és írnunk kell benne két for ciklust, amik külön-külön bejárják a két tömböt.

```
#! /usr/bin/awk -f
BEGIN {
  FS=";"
}
{
  orszagok[$1]=orszagok[$1]""$3"-"$4" "
  tajak[$2]+=$5
}
END {
  print("Országok")
  for(orszag in orszagok){
    print(orszag": "orszagok[orszag])
  }
  print("")
  print("Tájegységek")
  for(taj in tajak){
    print(taj": "tajak[taj])
  }
}
```

3. feladat

Írj AWK scriptet, ami összehasonlítja az egymás utáni sorokat és kiírja, hogy az előző sorhoz képest hány azonos pozíción lévő karakterben történt változás. Tehát kiírja a sorok közötti különbséget, 0 ha a két sor azonos. Az első sorban az üres sorhoz képesti változást írja ki, tehát a sor hosszát. Az üres karakterből karakterbe, vagy a karakterből üres karakterbe történő változás is változás.

```
?>cat ism.txt
abcdefghijklmnop
bacdefghijklmnop
bacdefghijklmnopq
bacdefghijklmnopq
qponmlkjhgfedcba
qponmlkjhgfedcb
qponmlkjhgfedc
qponmlkjhgfed
qponmlkjhgfe
qpo
qp
q
?>awk -f 3_beapait.awk ism.txt
16
2
1
0
17
1
1
1
1
9
1
1
```

Megoldás:

Először értelmezzük a feladatot, és próbáljunk meg ötleteket gyűjteni azzal kapcsolatban, hogy lehetne ezt awkkal megoldani. A legelső megoldandó dolog az, hogy karakterenként tudjuk bejárni a sorokat. Ehhez módosítanunk kell, az elválasztójelet az üres stringre. Ekkor egy mező egy karakter lesz.

```
#! /usr/bin/awk -f
BEGIN{
  FS=""
}
```

Ami a legfontosabb ennél a feladatnál, hogy valahogyan el kell tárolnunk az előző sort, ugyanis ahhoz képest kell összehasonlítani majd a jelenlegit. A változók, így a tömbök is, olyanok az awkban, hogy sorokon átívelően megőrzik a tartalmukat. Tehát semmi más dolgunk nem lesz, mint létrehozni egy *elozoSor* tömböt, amelyben a mező indexe mentén helyezzük el a mezőértékeket. Ahhoz, hogy minden elemet megkapjunk egy for ciklussal be kell járnunk az adott sor összes mezőjét.

```
#! /usr/bin/awk -f
BEGIN{
  FS=""
}
{
  for(i=1;i<=NF;i++){
    elozoSor[i]=$i
  }
}
```

Ezek után elérhető lesz az előző sor tartalma a következő sorban. Kezdésnek az üres tömbhöz képest fogjuk megnézni, hogy mennyi változás történt. Ha történik változás, akkor növelünk egy *valtozas* változót. Fontos, hogy ezt azelőtt tegyük meg, mielőtt újra beállítanánk az előző sort. Mivel a változást soronként vizsgáljuk, ezért mindig le kell nullázni a *valtozas*t még a sor elején.

```
#! /usr/bin/awk -f
BEGIN{
  FS=""
}
{
  valtozas=0
  for(i=1;i<=NF;i++){
    if(elozoSor[i]!=$i){
       valtozas++
    }
    elozoSor[i]=$i
  }
  print(valtozas)
}
```

Azt hihetnénk, hogy ezzel készen is vagyunk, de van benne még egy elég nagy hiba. Méghozzá az *elozoSor* tömbünk, amennyiben egy hosszabb sort tartalmazott a jelenleginél a végén több sorral korábbról benne maradhatnak karakterek, amik hibát okoznak. Ahhoz, hogy ezt megoldjuk az előző sor mezőinek a számára (NF) van szükségünk a következő sorban. Ezt tároljuk el az *elozoNF*

változóban. Ez alapján, ha az *elozoNF*>NF, akkor indítani kell egy for ciklust NF+1-től (ugyanis az NF-et már megvizsgáltuk) *elozoNF*-ig és beállítani üres stringre (vagy 0-ra) az érintett indexeket. Ami még fontos, hogy ezek is változásnak számítanak, ezért a különbséget mindenképpen fel kell venni a *valtozas* változóban.

```
#! /usr/bin/awk -f
BEGIN{
  FS=""
}
{
  valtozas=0
  for(i=1;i<=NF;i++){
    if(elozoSor[i]!=$i){
       valtozas++
    }
    elozoSor[i]=$i
  }
  if(elozoNF>NF){
    for(i=NF+1;i<=elozoNF;i++){
      elozoSor[i]=""
      valtozas++
    }
  }
  print(valtozas)
  elozoNF=NF
}
```