10

AWK programozás,
minták,
vezérlési szerkezetek

AWK

- adatvezérelt szkriptnyelv
- text processing, adat kiterjesztés, tagolt adatok automatizált soronkénti feldolgozása
- a forrásállományt soronként beolvassa és feldolgozhatóvá teszi
- a sor egyes elemeire külön alapértelmezett változóként lehet hivatkozni (az elválasztási módszert definiálni lehet), rengeteg beépített változó érhető el
- BASH-nél lazább szintaxis

AWK delimiter

- Az awk soronkénti műveletvégzést tesz lehetővé, a sorokat változókba tagolja egy *delimiter* által:

  $1 $2 $3 $4 $5 …... $NF

  (NF változó a sor mezőinek számát tárolja)
- Az awk alapértelmezés szerint a whitespace az elválasztás
- Megadható:
  - kapcsolóval: `awk -F";" -f CODEFILE FILE`
  - FS változónak értékül adva: `FS = ";"`
- Delimiter lehet bármilyen karakter (akár escape) vagy **reguláris kifejezés**

Feladat

- Írjunk awk scriptet, ami összegzi a szamok.csv fájl 3. oszlopát. Figyelj oda az elválasztó jelre!
- Írjunk awk scriptet, ami a delim.dat fájl különböző elválasztó jelek mentén választja el. Próbáljuk ki a különböző lehetőségeket.

AWK futtatási módok

- AWK parancs, közvetlen programkódmegadás:
  - awk 'PROGRAMKÓD' FILE
  - példa: ls -l | awk '{print $1, $5}'
  - a programkód helyére minden indentálás nélkül beírod a programkódot → ez rettentően nehezen olvashatóvá teszi az egészet, csak a legegyszerűbb feladatok esetén javasolt !!!
- AWK programkód külső fájlból beolvasva:
  - awk -f CODEFILE FILE
  - ez a legkézenfekvőbb
- AWK program, mint futtatható AWK script:
  - parancsértelmező fejrész megadása:
    - #! /usr/bin/awk -f

      (ha nem ismeri akkor locate bin/awk)
  - ./AWK_SCRIPT FILE

Feladat: HelloWorld AWK futtatása ls -l kimenetén

Kifejezések

- +, –, *, /, %, ^
- ++, - -
- egymás mellé írás (string konkatenáció)
- mezőhivatkozás ($KIF)
- =, +=, -=,*=, /=, %=, ^=
- <, <=, >, >=, ==, != (stringek esetén lexikografikus)
- valtozo ~ / REGKIF /, valtozo !~ / REGKIF /
- INDEX in TÖMB (tömbindex létezése, true or false)
- !, &&, ||
- KIF1 ? KIF2 : KIF3

Egyszerű változók

- jelölése kisbetűkkel
- értékadás, hivatkozás C-hez hasonlóan
- dinamikusak: első használatkor jönnek létre
- típusuk az alapján állítódik be, amit értékül adunk:
  - numerikus
  - szöveges
  - tömb
- A szám szöveggé, a szöveg számmá automatikusan konvertálható (ha nem sikerül akkor 0)
  - de ez manuálisan is elvégezhető, kikényszeríthető:
    - str+0 (szöveg számmá)
    - num”” (szám szöveggé)

Változó értékek

- Szám:
  - egész: 12
  - valós, tizedesponttal: 12.0
  - valós, lebegőpont: 1.2e+1
- Szöveg:
  - ”SZOVEG\n” (escape karakterek használhatóak)
  - üres string: ””

Beépített változók

- $1 $2 $3 (módosíthatók, $-el ellátott változóval is hivatkozhatunk ezekre: $i)
- $0 – az egész sor egyben
- NF – number of fields
- NR – aktuális sor azonosítója
- FNR – több fájl esetén fájlonkénti aktuális sorának azonosítója
- FILENAME – bemeneti fájl neve (amit feldolgozunk)
- FS – delimiter (alapértelmezés szerint szóköz)
- OFS – kimeneti delimiter (alapértelmezés szerint szóköz)
- RS – sorhatároló karakter (alapértelmezetten \n)
- ORS – kimeneti sor delimiter (alapértelmezetten \n)
- IGNORECASE – ha nem nulla, akkor azonosak a kis és nagybetűk (alapértelmezés szerint mindig nulla)

  Feladat: próbáljuk ki a beépített változók viselkedését

egy AWK program felépítése

- Szabályokból épül fel:
  - MINTA{ AKCIÓ1; … AKCIÓ2; }
    - ha elhagyjuk a MINTA-t, akkor minden sorra végrehajtja
    - ha elhagyjuk a { AKCIÓ }-t akkor a MINTA teljesülése esetén {print $0} fog végrehajtódni (ha nem akarunk semmit, akkor {} -et kell hagynunk a MINTA után)
- Egyes parancsok végét ; -el jelöljük (soronként tagolva ez elhagyható)
- Nem érzékeny a szóköz, TAB tagolásra: akár az egész programkódot írhatjuk egyetlen sorba, de szét is tagolhatjuk valamilyen kódformázási protokoll alapján
- Kommentelés: #

Minták

- Soronkénti feltétel, ha teljesül, akkor végrehajtódik a blokk, különben nem.
- A változók felhasználhatóak a minta megadásakor
- Elemi minták:
  - MINTA: relációs kifejezés
  - (MINTA)
  - ! MINTA
  - / REGKIF / ha a reguláris kifejezés illeszkedik a sorban
  - BEGIN : awk program indulásakor elsőként fut le
  - END : awk program végén a kilépés előtt utoljára lefutó rész
    - Nem kombinálhatóak más mintákkal
- Összetett minták: &&, ||

Minták

- Soronként ellenőrzi és megadási sorrend alapján:
  - BEGIN{ AKCIÓ1; … AKCIÓ2; }
  - MINTA1{ AKCIÓ1; … AKCIÓ2; }

    .

    .

    .
  - MINTAn{ AKCIÓ1; … AKCIÓ2; }
  - END{ AKCIÓ1; … AKCIÓ2; }
- Feladat:

1) Köszöntő üzenet, számoljuk össze a sorokat és csak a végén írjuk ki.

2) Írjuk ki az első sor kivételével az összes sort

3) Az ls -l kimenetéből szűrjük ki a könyvtárakat és csak a nevüket írjuk ki

Vezérlési szerkezetek

- if (FELTÉTEL) {
  UTASÍTÁS
  } else if (FELTÉTEL) {
  UTASÍTÁS
  } else {
  UTASÍTÁS
  }
- while (FELTÉTEL) {}
- do {} while (FELTÉTEL)

Vezérlési szerkezetek

- for (KIF1; KIF2; KIF3;) {}
- for (INDEX in NÉV) {}
  - nem sorrendben járja be a tömböt!
  - csak ha nem számít a sorrend, pl: halmaz
- break, continue
- exit
  - (ha nem az END-ben van, akkor az END szekció még lefut!!)

Feladatok

1) Azokat a sorokat írjuk ki, ahol az utolsó elem 120000-nél nagyobb és a végén hogy hány ilyen van.

2) Azokat a sorokat írjuk ki és számoljuk meg, ahol az elemek összege 700000-nél nagyobb

3) A 2)-pontban megkapott sorok elemeinek összegének az átlagát írjuk ki (egyszer, a végén).