# 08

AWK

# AWK

- Mintakereső és -feldolgozó program saját programozási nyelvvel
- 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 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

# AWK felépítés - bevezetés

- 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
- Nem érzékeny a szóköz, TAB tagolásra: akár az egész programkódot írhatjuk egyetlen sorba, de szét is tagolhatjuk valamilyen kódformázási protokoll alapján
- Kommentelés: #

# 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 ";" CODEFILE FILE`
  - FS változónak értékül adva: `FS = ";"`
- Delimiter lehet bármilyen karakter (akár escape) vagy **reguláris kifejezés**

  Feladat: hozzunk létre egy fájlt, amiben soronként más elválasztó jel van, adjunk meg rá reguláris kifejezést, ami az awk számára feldolgozhatóvá teszi.

# 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:
    - str+0 (szöveg számmá)
    - num”” (szám szöveggé)

Feladat: próbáljuk ki a különböző típusokat és a konverziót.

# 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