

11

Reguláris kifejezések
AWK bevezetés

Reguláris kifejezések

- Komplex mintaillesztés megadása
- Szövegen belül ***bárho***l elkezdődő és ezen belül a ***leghosszabb*** illeszkedést értjük.
- Részletesebben a formális nyelvek (vagy számtud alapjai) kurzus témája, most csak mint egy eszköz nézzük meg
- Bashben használta: `egrep` paranccsal

Elemi KIFEJEZÉSEK MEGADÁSA

- `KARAKTER`: közösleges karakter egy példányára illeszkedik
- `\KARAKTER`: spec karakter is közösleges karakter lesz, annak példányára illeszkedik
- `()`: üres szóra illeszkedik (üres zárójelpár)
- `.` bármilyen karakter egy példányára illeszkedik (spec karaktereket kivéve)
- `[HALMAZ]`: felsorolt karakterek bármelyikének egy példányára illeszkedik, `[TÓL-IG]` hasonlóan, csak az intervallumba eső bármelyik karakterrel, `[^HALMAZ]` a nem felsorolt karakterek bármelyikének egy példányára illeszkedik. Például `[a-z],[A-Z],[0-9]`
- `^` sor elejére illeszkedik, `$` sor végére illeszkedik

Összetett kifejezés

- **iterálható:**
 - KIF^* : a KIF akárhány egymást követő példányra illeszkedik, de az üres szóra is
 - KIF^+ : a KIF legalább egy egymást követő példányra illeszkedik
 - $KIF?$: 0 vagy 1 előfordulásra illeszkedik (0 előfordulás az üres szó)
 - $KIF\{i\}$: pontosan i egymást követő példányra illeszkedik
 - $KIF\{i,\}$: legalább i egymást követő példányra illeszkedik
 - $KIF\{i,j\}$: legalább i egymást követő példányra illeszkedik, de legfeljebb j egymás után követőre ($i \leq j$ teljesülése mellett)
- $KIF1KIF2$: összefűzhető (**konkatenáció**)
- $KIF1|KIF2$: legalább az egyik kifejezésre illeszkedik (logikai vagy, **alternáció**)
- Műveletek erőssége: iteráció, konkatenáció, alternáció. De csoportosítható: $((KIF1)(KIF2))$

Reguláris kifejezés - bash

- `egrep 'REGKIF' ALLOMANY:`
 - csak a reguláris kifejezéshez illeszkedő sorokat írja ki a fájlból
 - ha nem adunk meg ALLOMANYt akkor stdin-ről olvas

egrep fontosabb kapcsolók

- c illeszkedések darabszáma
- n az illeszkedő sor sorszámát is kiírja
- v nem illeszkedő sorok
- f KIFFÁJL az illesztő kifejezéseket szöveges állományból olvassa (fájl minden sorában pontosan egy reguláris kifejezés lehet, VAGY)
- q eltünteti a kimenetet, visszatérési érték az exit státusz: igaz, ha talál illeszkedést, hamis ha nem (ezzel lehet if-fel használni)

Példák

- írjunk reguláris kifejezést, ami akárhány “a” illeszkedését vizsgálja
- írjunk reguláris kifejezést, ami az IP(v4) címekre illeszkedik
- írjunk reguláris kifejezést, ami egy e-mail címet fogad és megmondja, hogy helyes szerkezetű-e
- írjunk reguláris kifejezést, ami paraméterül kap egy könyvtárat, ahonnan ls -l listázva válasszuk ki a könyvtárakra vonatkozó sorokat
- írjunk reguláris kifejezést, ami olyan sorokra illeszkedik ami az alábbi négy szó valamelyikét kizárólagosan tartalmazza: asztalon, asztalra, asztalhoz, asztalrol

Egyik korábbi zh feladat

Írj bash scriptet, amelynek a paramétere egy a számok.csv-hez hasonló fájl (egy ilyen példa fájl megtalálható a feladat mappában) és elmenti a negyedik oszlop 7 legnagyobb értékét, rendezetten egy max.txt fájlba. Minden lehetséges érték csak egyszer szerepeljen.

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

Feladat

- az előbb megoldott zh feladatot oldjuk meg AWK segítségével