Operációs rendszerek alapjai

6. gyakorlat

Mai gyakorlat

- awk
- Feladatokhoz szükséges: G06.tar.gz

awk

- Alfred Aho, Peter Weinberger és Brian Kernighan írta
- Szöveg feldolgozás orientált nyelv
  - bemeneti stream-t az eredeti változtatása nélkül módosítja
- Használhatjuk
  - parancsként, a bemenetet pipe-olva, vagy argumentumként átadva (akár több fájl is)
  - interpreter direktívaként
    - sed-hez hasonlóan `-f` kapcsolóval megadhatjuk a kódot tartalmazó fájlt
- Interpretált nyelv
  - GNU Awk
  - New Awk
  - stb.

Programstruktúra

- awk program különböző részekre bontható
  - inicializáló rész (`BEGIN` kulcsszó)
    - egyetlen egyszer fut le, az első sor olvasása után
    - lehet többször használni (minden parancs egyszer fut el)
  - saját függvények definíciója
  - programtörzs
    - bizonyos mintázatok esetén adott parancsok lefuttatása
    - adott mintázathoz tartozó parancsok `{}`-ben, parancsok elválasztása `;`-el
    - üres mintázat minden sorra lefut
  - lezáró rész (`END` kulcsszó)
    - egyetlen egyszer fut le, az utolsó sor olvasása után
    - lehet többször használni (minden parancs egyszer fut le)

Szöveg lebontása

- Bemenet felosztva rekordokra és mezőkre
  - alapértelmezetten új sor a rekord elválasztó és whitespace a mező elválasztó, tehát egy rekord egy sornak felel meg, egy mező egy szónak
  - ez a működés átállítható
- Kiírás a `print` paranccsal
  - `$0` változó kiírja a rekordot
  - `$n` változó az n-edik mezőt írja ki
  - vesszővel elválasztva a mezők
  - precízebb kiíráshoz van `printf` is

G06F01

- Írj egy awk script-t, ami kiírja egy fájl összes sorát!
- Módosítsd a script-t, hogy kiírja egy fájl összes sorának első szavát!
- Próbáld ki a kezdes.txt fájlon!

Változók és kifejezések

- Numerikus konstansokat használhatunk hagyományos és tudományos módon
  - decimális, oktális (`0`-val kezdődik) és hexadecimális (`0x`-el kezdődik)
- String konstansokat “” között használhatunk
- Reguláris kifejezés konstansok `//` között, `@//` között strongly typed
- Változók használata szokásosan
  - előre definiálhatunk változókat, akár az inicializáló rész lefutása előtt a `-v foo=bar` szintaktikáva (csak parancssoros futtatás esetén)
- Előre definiált változók
  - `NR` az eddig feldolgozott rekordok száma
  - `NF` az adott rekord mezőinek száma
  - `RS`, `ORS`, `FS`, `OFS` a be- és kimeneti rekord- és mező szeparátor
    - alapértelmezetten rendre új sor, új sor, whitespace, whitespace

G06F02

- class.txt fájl múlt óráról tartalmazza a kedvenc tárgyad pontjait .csv formátumban
  - egy oszlop Neptun kód, két oszlop házi pontok, egy oszlop érdemjegy
- Írd egy awk script-t, ami kiírja a Neptun kódokat és a hozzájuk tartozó érdemjegyeket!
  - mező szeparátor átállítása az inicializáló blokkban
  - első és negyedik mező kiírása

Mintázatok

- Többféleképpen tudunk szűrni sorokat
  - `/regex/` esetén a megadott kifejezésnek megfelelő sorokat választja ki
  - `expression` esetén nem nulla vagy nem üres eredmény esetén a sor kiválasztása
    - kifejezések később bővebben
  - `begin_pattern,end_pattern`
- Mintázatok között értelmezhető a metszet (`&&`), unió (`||`) és negálás (`!`) művelete

G06F03

- Írj egy awk script-t, ami kiírja az ötös jegyet szerző hallgatókat a class.txt fájlból!
  - `5$` reguláris kifejezéssel szűrj

Kifejezések

- Szám és szöveg változók automatikus konverziója
  - ha valamiért nem működik jól magától, akkor expliciten kell konkatenálni számot üres string-el, vagy string-t összeadni nullával
- Számok között szokásos műveletek, értékadások és összehasonlítások
  - `+, -, *, /, %, ^, ++, --`
  - `=, +=, -=, *=, /=, %=, ^=`
  - `<, >, <=, >=, ==, !=`
- Szokásos logikai műveletek
  - `&&, ||, !`
  - ami nem nulla vagy nem üres, az `true`
- Regex-el összehasonlítás `~` és `!~` operátorral

G06F04

- Írj egy awk script-t, ami kiszámolja az átlagos érdemjegyet a class.txt fájlban!
  - inicializáló blokkban mező szeparátor beállítása
  - jegyek összeadása
  - lezáró blokkban összeg leosztása a feldolgozott rekordok számával (`NR`)

Elágazások

```
if (condition)
     commands
else if
     commands
else
     commands
```

```
condition ? true-expr : false-expr
```

```
switch (expression) {
case value:
     commands
     break
case regexp:
     commands
     break
default:
     commands
}
```

while

```
while (condition)
     commands

do
     commands
while (condition)
```

- `break` és `continue` szokásosan
- `next` befejezi az adott rekord feldolgozását
- `nextfile` befejezi az adott fájl feldolgozását
- `exit` befejezi a futást, opcionális visszatérési értéket is megadhatunk

for

```
for (initialization; condition; increment)
     commands

for (val in array)
     commands
```

- `break`, `continue`, `next`, `nextfile`, `exit` szokásosan

G06F05

- Switch segítségével számoljuk meg az ügyes (elégtelen és elégséges érdemjegy), ügyesebb (közepes vagy jó érdemjegy) és legügyesebb (jeles érdemjegy) hallgatókat a class.txt fájlban!
  - inicializáló blokkban mező szeparátor beállítása
  - hallgatók számlálása
  - lezáró blokkban kiírás

G06F06

- Írj egy awk script-t, ami kiszámolja egy adott szám faktoriálisát!
  - próbáld ki while és for ciklus segítségével is
  - `-v` kapcsolóval add a számot
    - `./G06F06.awk -v n=5`

Tárolók

- Asszociatív tömbök vannak csak
  - használható hagyományos tömbként, ha a kulcsok egész számok
  - többdimenziós tömböt is lehet gyártani kulcsok felsorolásával
  - gawk-ban tömbök tömbje is
- Tömb kulcs és érték lehet szám vagy string
- Deklaráció nélkül lehet egyből használni
- Szokásosan lehet indexelni és értékül adni
  - `array[bar]=foo`
- Lehet törölni adott elemet
  - `delete array[bar]`
- Iterálható

G06F07

- A matrix.txt fájl egy bűvös négyzetet tartalmaz .csv formátumban
  - minden sor, oszlop és a két átló összege ugyanannyi
- Írj egy awk script-t, ami ellenőrzi, hogy valóban bűvös négyzet-e!
  - inicializáló blokkban mező szeparátor beállítása
  - törzsben a mátrix 2D tömbbe mentése
  - lezáró blokkban a tulajdonság ellenőrzése

Beépített függvények

- Numerikus függvények
  - `atan2(y,x)`, `cos(x)`, `sin(x)`, `exp(x)`, `log(x)`, `sqrt(x)`, `int(x)`, `srand(x)` és `rand()`
- String függvények
  - `index(sub,str)`, `length(str)`, `sub(regex,replacement,target)`, `gsub(regex,replacement,target)`, `sprintf(format,expression,...)`, `substr(str,pos,len)`, `tolower(str)`, `toupper(str)`
- Shell elérése
  - `system(command)`
  - `print command | "/bin/bash"`
- Egyéb függvények: I/O műveletek, bitenkénti operátorok stb.

Saját függvények

- Hagyományos szintaxis

```
function bar(parameters) {
    commands
}
```

- Alapértelmezetten minden változó globális
  - lokális változót fel kell sorolni a szignatúrában, de nem kell átadni (konvenció szerint extra space-k)
    `function bar(    i)` meghívása `bar()`
- Tömbök referencia szerint, minden más érték szerint átadva
- Visszatérési értékek a `return` kulcsszóval
- Indirect function call a @ karakterrel
  - ha a `bar` változó értéke `foo`, akkor `@bar()` meghívja a `foo` függvényt

G06F08

- Írj két függvényt awk script-ben, ami megkeresi egy 2D tömb minimumát, illetve maximumát!
- Bemeneti paraméter legyen, hogy melyik függvényt hívod meg!
- Futtasd le a matrix.txt fájlon!
- Lépések
  - inicializáló blokkban mező szeparátor beállítása
  - törzsben a mátrix 2D tömbbe mentése
  - lezáró blokkban a bemenetnek megfelelő függvény meghívása indirect function call-al

Következő gyakorlat

- Terminál-internet interakció

Érdekességek, olvasnivalók

- GNU Awk User's Guide
- The AWK Programming Language book
- Ronald P. Loui: Why GAWK for AI?
- Computerphile - What's your Favourite Programming Language?
- How To Write Unmaintainable Code