AWK & GREP

1. AWK

Awk este un limbaj de programare specializat in prelucrari de text, deci putem crea scripturi ce contin functii sau structuri de control awk.

In continuare in cadrul laboratoarelor ne vom focusa doar pe comanda **awk**.

Comanda awk permite prelucrarea facila a fisierelor ce contin date structurate pe coloane, separate prin spatii, permitand astfel crearea de statistici si rapoarte.

Sintaxa comenzi awk este :

awk 'pattern {acțiuni}' fișier

Sau daca folosim parametri sintaxa devine:

awk <parametri> 'pattern {actiuni}' fisier

Pattern reprezinta sablonul cautat si este un parametru optional.

Comanda awk este folosita atunci cand fisierul din care trebuie extrase date are un format binedefinit, impartit pe coloane.

Separatorul (field separator) predefinit pentru comanda awk este spatiul sau tab-ul.

In cazul comenzii awk, ca si in cazurile prezentate anterior, variabilele $0,$1,$2...au o semnificatie speciala:

- $0 - reprezinta o linie intreaga
- $1, $2, …$n - reprezinta itemul de la indexul (coloana) 1,2,...,n

```
#awk <params> '<conditions> { <actions> }' <file name>
awk '{ print $0 }' testData/datafile
awk '{ print $2, $5 }' testData/datafile
ls -l | awk '{ print $9 }'
```

Comanda awk poate fi conectata la alte comenzi prin intermediul operatorului pipe.

In exemplele de mai sus a fost folosita doar sintaxa de baza si nu s-a utilizat partea de pattern (conditie) si nici cea de parametri.

Daca se doreste aplicarea unei conditii, de exemplu: sa se afiseze randurile in care coloana a 5-a sa fie >100:

```
awk '$5>100 { print $0 }' testData/datafile
```

Daca se doreste afisarea coloanelor 1 si 4 atunci cand coloana a 5-a e >100:

```
awk '$5>100 { print $1,$4 }' testData/datafile
```

Daca se doresete sa se aplice conditii multiple se pot folosi operatorii SI (&&) respectiv SAU (||).

De ex: coloana 5 >100 si coloana 5 < 500

```
awk '$5>100 && $5<500 { print $0 }' testData/datafile
```

Pentru a utiliza variabile in loc de valori hard-codate in conditiile comenzi awk, ele trebuie sa fie definite in prealabil.

```
#use shell variables in conditions
# syntax: awk -v <awk var name>="$<shell var name>" '[...]'

upperLimit=500
awk -v awkVar="$upperLimit" '$5<awkVar {print $0}' testData/datafile
```

Daca se doreste utilizarea mai multor variabile trebuie pusa constructia **-v numeVar si valoare** pentru fiecare variabila awk. De ex:

```
upperLimit=500
lowerLimit=100

awk -v awkVarUpper="$upperLimit" -v awkVarLower="$lowerLimit"
'$5>awkVarLower && $5<awkVarUpper {print $0}' testData/
datafile
```

Exista cazuri in care sunt fisiere cu format special, insa nu spatiul sau tab-ul este separatorul de coloane, ci ca si separator este : sau ; sau altceva.

Pentru a putea utiliza comanda awk si pe alte formate si alte separatoare se poate utiliza optiunea **-F** .

```
awk -F : '{print $3}' testData/awk.txt
```

2. GREP - General Regular Expression Parser

3.

Se cauta in fisiere liniile care se potrivesc unui pattern(sablon)

Sintaxa:

```
# grep <parameters> <pattern> <file name>
```

Pentru a afisa liniile ce corespund pattern-ului:

```
grep data testData/lsResult
```

Pentru a numara liniile care corespund pattern-ului

```
grep -c data testData/lsResult
```

Pentru a afisa toate liniile ce corespund pattern-ului in modul case insensitive (pentru a ignora caracterele mici/mari)

```
grep -i DATA testData/lsResult
```

Pentru a afisa toate liniile ce NU corespund unui pattern (inverted search)

```
grep -v data testData/lsResult
```

Se pot utiliza si combinat optiuni pentru comanda grep:

Ex: pentru a numara numarul de linii care nu corespund unui pattern:

```
grep -vc data testData/lsResult
```

Ex: pentru a numara numarul de linii in mod case insensitive care corespund unui pattern:

```
grep -ic data testData/lsResult
```

Pentru a realiza cautari mai avansate se poate folosi comanda grep in combinatie cu expresii regulate.

Pentru a afisa liniile ce incep cu un pattern:

```
grep ^data testData/lsResult
```

Pentru a afisa liniile ce se termina cu un pattern

```
grep sh$ testData/lsResult
```

Pentru a afisa liniile ce contin exact un pattern:

```
grep ^dummy$ testData/lsResult
```

Pentru a afisa liniile ce au 5 caractere:

```
grep ^.....$ testData/lsResult
```

Daca dorim sa putem avea si alte cuvinte inaintea unui pattern:

```
grep .*data testData/lsResult

grep .sh testData/lsResult
```

In caz de mai sus, punctul ar tine locul oricarui caracter.

Daca dorim ca sablonul nostru sa contina caractere speciale cum ar fi **.** (din extensia fisierului) trebuie sa precedam caracterele speciale prin **'\.'** sau **\\.sh** sau **"\."**sh. Pentru ca daca rulam urmatoarea comanda:

```
grep '\.'sh testData/lsResult
grep \\.sh testData/lsResult
grep "\."sh testData/lsResult
```

Daca dorim sa extragem liniile cu extensia sh, deci sa se termine cu .sh, mai trebuie sa adaugam caracterul $ la exemplele anterioare

```
grep '\.'sh$ testData/lsResult
grep "\."sh$ testData/lsResult
grep \\.sh$ testData/lsResult
```

Pentru a extrage liniile care incep cu cifre:

```
grep ^[0-9] testData/lsResult
```

Pentru a extrage liniile care incep cu un caracter intre a si m :

```
grep ^[a-m] testData/lsResult
```