Λειτουργικά Συστήματα Ι

Ασκήσεις Πράξης

Τμήμα Μηχανικών Πληροφορικής & Υπολογιστών

Σχολή Μηχανικών

awk

awk

Η **awk** είναι μια πανίσχυρη εντολή που σχεδιάσθηκε για επεξεργασία κειμένου και χρησιμοποιείται συνήθως ως εργαλείο ανάκτησης δεδομένων και δημιουργίας αναφορών. Δημιουργήθηκε το 1978 στα εργαστήρια Bell Labs και η προέλευση του ονόματός της προέρχεται από τα επίθετα των δημιουργών της Alfred **A**ho, Peter **W**einberger και Brian **K**ernighan.

Σύμφωνα με τους δημιουργούς της αποτελεί μια γλώσσα προγραμματισμού με βασική λειτουργία την αναζήτηση προτύπων σε αρχεία κειμένου και εκτέλεση συγκεκριμένων ενεργειών στα πεδία των γραμμών που περιέχουν το πρότυπο.

awk

Η σύνταξη της εντολής είναι:

awk [OPTION]… [PROGRAM]… [FILE]…

όπου PROGRAM είναι οι awk εντολές που καθορίζουν το είδος της επεξεργασίας που θέλουμε να εφαρμόσουμε στα περιεχόμενα του αρχείου FILE. Αν δε δοθούν αρχεία εισόδου η εντολή διαβάζει από την προκαθορισμένη είσοδο.

Οι επιλογές της εντολής awk είναι:

-F value θέτει στον διαχωριστή πεδίων FS (field separator) την τιμή *value*

-f file οι εντολές του προγράμματος διαβάζονται από το αρχείο *file* αντί από τη γραμμή εντολών

-v var=value θέτει στη μεταβλητή *var* την τιμή *value*

awk

Πιο συγκεκριμένα, ένα πρόγραμμα awk αποτελείται από μια σειρά από κανόνες στη μορφή:

pattern { action }

Κάθε κανόνας καθορίζει το πρότυπο αναζήτησης και την ενέργεια που θα εκτελεστεί στη γραμμή που ταιριάζει με αυτό.

Αν σε έναν κανόνα λείπει το πρότυπο, η ενέργεια εκτελείται για όλες τις γραμμές.

Αν λείπει η ενέργεια από τον κανόνα, εμφανίζονται όλες οι γραμμές που ταιριάζουν με το πρότυπο.

awk

Το πρώτο μας παράδειγμα τυπώνει όλες τις γραμμές του αρχείου /etc/group που ταιριάζουν με το πρότυπο *systemd*:

```
$ awk '/systemd/' /etc/group  # sed -n '/systemd/'
systemd-journal:x:101:
systemd-timesync:x:102:
```

Το πρότυπο είναι μια λογική συνθήκη ή μια κανονική έκφραση ανάμεσα σε δεξιές πλάγιες. Για παράδειγμα το πρότυπο /^[a-zA-Z]/ ταιριάζει σε όλες τις γραμμές που αρχίζουν με πεζό ή κεφαλαίο λατινικό γράμμα. Η ενέργεια είναι μία ή περισσότερες εντολές που χωρίζονται με το ελληνικό ερωτηματικό ή τον χαρακτήρα αλλαγής γραμμής και περικλείονται σε άγκιστρα **{ }**.

awk

Η προηγούμενη εντολή θα μπορούσε να γραφεί ορίζοντας και την ενέργεια, που στη συγκεκριμένη περίπτωση είναι η προκαθορισμένη εντολή **print**:

```
$ awk '/systemd/ {print}' /etc/group
```

awk

Κατά την ανάγνωση της εισόδου της, η awk χωρίζει κάθε γραμμή σε πεδία και τα αναθέτει κατά σειρά στις μεταβλητές **$1**, **$2**, **$3** κ.ο.κ.. Η δε μεταβλητή **$0** αντιπροσωπεύει ολόκληρη τη γραμμή, οπότε οι εντολές *print* και *print $0* είναι ισοδύναμες.

Η εντολή awk που ακολουθεί δέχεται είσοδο από σωλήνωση και τυπώνει το 9ο πεδίο των γραμμών που περιέχουν το πρότυπο *bigfile*:

```
$ ls -l | awk '/big/ {print $9}' # print filenames
bigfile.txt
verybigfile
```

awk

Αν θέλουμε να τυπώσουμε εκτός από το όνομα του αρχείου και το μέγεθός του, θα προσθέσουμε και το 5ο πεδίο διαχωρίζοντάς τα με έναν χαρακτήρα της επιλογής μας, όπως είναι για παράδειγμα το κενό διάστημα:

```
$ ls -l | awk '/big/ {print $9 " " $5}'   # size
bigfile.txt 2171286
verybigfile 10856456
```

awk

Αν τα πεδία δεν χωρίζονται με το κενό διάστημα αλλά κάποιον άλλο χαρακτήρα, θα πρέπει να τον προσδιορίσουμε με την επιλογή **-F**.

Η επόμενη εντολή τυπώνει όλες τις ομάδες χρηστών και τα αντίστοιχα μέλη:

```
$ awk -F':' '{print $1 " " $4}' /etc/group
root                              # groups and members
...
students alice,bob,carol,dave
```

awk

Προσθέτοντας κείμενο στην εντολή print, η έξοδος μπορεί να γίνει πιο περιεκτική:

```
$ awk -F':' '{print "Group: " $1 \     # multiline
> "\tMembers: " $4}' /etc/group
Group: root     Members:
Group: daemon   Members:
Group: bin      Members:
Group: sys      Members:
...
Group: students Members: alice,bob,carol,dave
```

awk

Εκτός από print, η awk παρέχει και την εντολή **printf** που διαθέτει παραπλήσια σύνταξη με την ομώνυμη συνάρτηση της C. Στο παράδειγμα που ακολουθεί φιλτράρουμε τις γραμμές του αρχείου *file.txt* που ταιριάζουν με το πρότυπο *2522* και χρησιμοποιούμε την εντολή printf για να τυπώσουμε το 1ο, 2ο και 5ο πεδίο:

```
$ awk '/2522/ {printf "%8s %-15s %10s\n", \
> $1, $2, $5}' file.txt    # minus sign left-justify
   Peter Frampton       252241313
    Anna Frampton       252241313
```

awk

Η προσθήκη επικεφαλίδων στον παραπάνω πίνακα μπορεί να γίνει εύκολα με το ειδικό πρότυπο **BEGIN**. Οι ενέργειες του τμήματος BEGIN εκτελούνται μία μόνο φορά πριν διαβαστεί η πρώτη γραμμή της εισόδου της awk:

```
$ awk 'BEGIN {printf "%8s %-10s %10s\n", \
> "Lastname", "Firstname", "Phone";
> printf "============================="} /2522/ \
> {printf "%8s %-10s %10s\n", $1, $2, $5}' file.txt
Lastname Firstname       Phone
=============================
   Peter Frampton   2522041313
    Anna Frampton   2522041313
```

awk

Αντίστοιχα υπάρχει και το ειδικό πρότυπο **END**, οι ενέργειες του οποίου εκτελούνται μία μόνο φορά μετά την επεξεργασία όλων των γραμμών εισόδου της awk. Το επόμενο παράδειγμα τυπώνει μία φράση μετά την επεξεργασία:

```
$ ls -l | awk 'END {print "That\x27s all"} \
> /big/ {print $9, $5}'
bigfile.txt 2171286
verybigfile 10856456
That's all      # \x27 ascii code of single quote
```

Τα προγράμματα awk αποτελούνται από τρία τμήματα: το τμήμα BEGIN, το τμήμα επεξεργασίας και το τμήμα END. Οποιοδήποτε από τα τμήματα αυτά μπορεί να παραληφθεί.

awk

Παράδειγμα με χρήση σεναρίου δύο μόνο γραμμών: δύο κανόνες οι οποίοι εκτελούνται σειριακά:

```
$ cat script-01.awk
/2522/ {print $1, $2, $5}
/2521/ {print $1, $2, $5}
$ awk -f script-01.awk file.txt     # κλήση με -f
Peter Frampton 2522041313
Anna Frampton 2522041313
Wolfgang Hartmann 2521091466
```

Προφανώς οι δύο πρώτες γραμμές είναι αποτέλεσμα του πρώτου κανόνα και η τρίτη γραμμή το αποτέλεσμα του δεύτερου κανόνα.

awk

Εναλλακτικά μπορούμε να κάνουμε χρήση του **Shebang** (**#!**) στην πρώτη γραμμή του αρχείου για να καθορίσουμε τον διερμηνευτή που θα χρησιμοποιηθεί:

```
#!/usr/bin/awk -f
/2522/ {print $1, $2, $5}
/2521/ {print $1, $2, $5}
```

Αφού προσθέσουμε το δικαίωμα εκτέλεσης στον ιδιοκτήτη του αρχείου:

```
$ chmod u+x script-02.awk
```

το εκτελούμε καλώντας το με όρισμα ένα αρχείο δεδομένων:

```
$ ./script-02.awk file.txt
```

Μεταβλητές και τελεστές

Οι μεταβλητές στην awk ορίζονται με απλή εκχώρηση τιμής χρησιμοποιώντας τον τελεστή ανάθεση τιμής “=”. Η γενική μορφή είναι:

variable = value

Τα ονόματα των μεταβλητών ακολουθούν τους κλασικούς ονομασίας, δηλαδή αρχίζουν από γράμμα και μπορούν να περιέχουν οποιοδήποτε συνδυασμό πεζών και κεφαλαίων λατινικών γραμμάτων, ψηφίων και της κάτω παύλας. Να επισημάνουμε επίσης πως γίνεται διάκριση πεζών και κεφαλαίων.

Μεταβλητές και τελεστές

Η αρχικοποίηση μεταβλητών συνιστάται να γίνει στο τμήμα BEGIN το οποίο εκτελείται μία φορά στην αρχή. Οι μεταβλητές δεν έχουν τύπο δεδομένων και ο τρόπος που τις χειρίζεται η awk εξαρτάται από τα συμφραζόμενα. Τα παραδείγματα που ακολουθούν δείχνουν τη μετατροπή αλφαριθμητικών σε αριθμούς και το αντίστροφο:

```
$ awk 'BEGIN{one = 1; two = 2; print (one two) + 3}'
15
```

Οι αριθμητικές τιμές των μεταβλητών *one* και *two* μετατρέπονται στην αλφαριθμητική τιμή “12”, η οποία προστίθεται με το 3 και δίνει το τελικό αποτέλεσμα 15.

Μεταβλητές και τελεστές

Αντίστοιχα, όταν έχουμε πρόσθεση μεταξύ μεταβλητών με αλφαριθμητικά γίνεται πρώτα μετατροπή σε αριθμό και μετά η πρόσθεση:

```
$ awk 'BEGIN{one = 1; two = 2; print one + two}'
3
```

Τα αλφαριθμητικά που δεν μπορούν να μετατραπούν σε έγκυρους αριθμούς μετατρέπονται σε μηδέν. Το παράδειγμα που ακολουθεί δείχνει τη μετατροπή αλφαριθμητικών σε αριθμούς:

```
$ echo Alice 4.5 4,5 | awk '{print $1-1, $2-1, $3-1}'
-1 3.5 3
```

Αριθμητικοί τελεστές

Τελεστής	Πράξη	Παράδειγμα	Αποτέλεσμα
+	Πρόσθεση	5 + 4	9
-	Αφαίρεση	5 – 4	1
*	Πολλαπλασιασμός	5 * 4	20
/	Διαίρεση	5 / 4	1.25
%	Υπόλοιπο ακέραιας διαίρεσης	5 % 4	1
^	Ύψωση στη δύναμη	5 ^ 4	625

```
$ awk 'BEGIN {x = 5; y = 3;
> printf "%d / %d = %f\n", x, y, x/y}'
5 / 3 = 1.666667
```

Τελεστές συσχέτισης

Τελεστής	Περιγραφή	Παράδειγμα	Αποτέλεσμα
==	Σύγκριση για ισότητα	3 == 4	0
!=	Σύγκριση για ανισότητα	3 != 4	1
<	Μικρότερο	3 < 4	1
<=	Μικρότερο ή ίσο	3 <= 4	1
>	Μεγαλύτερο	3 > 4	0
>=	Μεγαλύτερο ή ίσο	3 >= 4	0

Οι τελεστές συσχέτισης συγκρίνουν δύο εκφράσεις και επιστρέφουν την τιμή **1** αν η συνθήκη που καθορίζεται από τον τελεστή ικανοποιείται ή την τιμή **0** αν ισχύει το αντίθετο. Όπως ακριβώς και με τη γλώσσα C, η τιμή 0 θεωρείται ψευδής ενώ η τιμή 1 αληθής.

Τελεστές συσχέτισης

Τα επόμενα παραδείγματα επιλέγουν τις γραμμές που έχουν στο 3ο πεδίο τιμή μικρότερη από 180 και στο 4ο πεδίο ίση με 1963 αντίστοιχα:

```
$ awk '$3 < 180' file.txt     # height lt 180
Anna Frampton 170 1963 2522041313
Alice Bauhaus 167 1965 2106399547
$ awk '$4 == 1963' file.txt   # birth year eq 1963
Anna Frampton 170 1963 2522041313
Steffen Reichert 188 1963 2310620789
```

Τελεστές συσχέτισης

Ο τελεστής “**>**” όταν συμμετέχει σε παράσταση της print πρέπει να περικλείεται από παρενθέσεις καθώς σε διαφορετική περίπτωση εκλαμβάνεται ως ανακατεύθυνση εξόδου.

Η επόμενη εντολή στέλνει τις τιμές του 3ου πεδίου στο αρχείο 180:

```
$ awk '{print $3 > 180}' file.txt
$ cat 180
188
...
189
```

Λογικοί τελεστές

Τελεστής	Περιγραφή	Παράδειγμα	Αποτέλεσμα
&&	Λογική σύζευξη (And)	1 && 0	0
||	Λογική διάζευξη (Or)	1 || 0	1
!	Λογική άρνηση (Not)	!0	1

Η εντολή που ακολουθεί επιλέγει τις γραμμές για τις οποίες το 3ο πεδίο είναι μεγαλύτερο ή ίσο από την τιμή 180 και το 4ο πεδίο μικρότερο από την τιμή 1963:

```
$ awk '$3 >= 180 && $4 < 1963' file.txt
Peter Frampton 188 1962 2522041313
Wolfgang Hartmann 189 1958 2521091466
```

Τελεστές σύνθετων αναθέσεων

Τελεστής	Παράδειγμα	Ισοδύναμη έκφραση
+=	x += a	x = x + a
-=	x -= a	x = x - a
*=	x *= a	x = x * a
/=	x /= a	x = x / a
%=	x %= a	x = x % a
^=	x ^= a	x = x ^ a

Οι παραπάνω τελεστές μας δίνουν τη δυνατότητα να συνδυάσουμε μια αριθμητική πράξη με τον τελεστή ανάθεσης τιμής.

```
$ awk 'BEGIN {x = 1; x += 5; print "x =", x}'
x = 6
```

Τελεστές αύξησης και μείωσης κατά ένα

Τελεστής	Παράδειγμα	Ισοδύναμη έκφραση
++	x++	x = x + 1
--	x--	x = x – 1

Οι δύο τελεστές μπορούν να προηγούνται ή να έπονται του τελεστέου. Στην περίπτωση που ο τελεστής προηγείται καλείται **προθεματικός** (prefix), ενώ όταν έπεται καλείται **επιθεματικός** (postfix). Όταν ο τελεστής είναι προθεματικός, εκτελείται η πράξη αύξησης ή μείωσης, πριν χρησιμοποιηθεί η τιμή του τελεστέου. Αντίθετα, όταν ο τελεστής είναι επιθεματικός, χρησιμοποιείται η τιμή της μεταβλητής και στη συνέχεια γίνεται η αύξηση ή η μείωση.

Τελεστές αύξησης και μείωσης κατά ένα

Τα επόμενα παραδείγματα επιδεικνύουν τη διαφορά επιθεματικών και προθεματικών τελεστών:

```
$ awk 'BEGIN {x = 5; y = x++;
>              printf "x = %d, y = %d\n", x, y}'
x = 6, y = 5
$ awk 'BEGIN {x = 5; y = ++x;
>              printf "x = %d, y = %d\n", x, y}'
x = 6, y = 6
```

Ταιριάζει και δεν ταιριάζει

Οι τελεστές **~** (ταιριάζει) και **!~** (δεν ταιριάζει) συγκρίνουν μια κανονική παράσταση με κάποιο συγκεκριμένο πεδίο της γραμμής εισόδου.

```
$ awk '$2 ~ /Fram/' file.txt      # contains Fram
Peter Frampton 188 1962 2522041313
Anna Frampton 170 1963 2522041313
$ awk '$4 !~ /1963/' file.txt     # don't contains
Peter Frampton 188 1962 2522041313
Alice Bauhaus 167 1965 2106399547
Wolfgang Hartmann 189 1958 2521091466
```

Τριαδικός τελεστής ?:

Τέλος, η awk διαθέτει τον ίδιο τριαδικό τελεστή “**?:**” με τη γλώσσα C, ο οποίος παρέχει έναν εναλλακτικό τρόπο γραφής της εντολής if-else.

Το παράδειγμα που ακολουθεί υπολογίζει τον μεγαλύτερο από δύο αριθμούς:

```
$ awk 'BEGIN {x = 5; y = 8; (x > y) ? \
>                max = x : max = y; print max}'
8
```

Ενσωματωμένες μεταβλητές

Πέρα από τις μεταβλητές που ορίζονται από τον χρήστη, η awk διαθέτει αρκετές ενσωματωμένες μεταβλητές που παίζουν σημαντικό ρόλο στην δημιουργία σεναρίων.

$n *n*-στό πεδίο της γραμμής (το $0 αντιπροσωπεύει ολόκληρη τη γραμμή)

FS Διαχωριστής πεδίων, εξ ορισμού το κενό και ο στηλοθέτης (Field Separator)

OFS Διαχωριστής των πεδίων εξόδου, εξ ορισμού το κενό (Output FS)

NR Ο αριθμός των γραμμών/εγγραφών εισόδου που έχει επεξεργαστεί η awk από την αρχή της εκτέλεσής της (Number of Record)

Ενσωματωμένες μεταβλητές

NF Πλήθος των πεδίων (Number of Fields)

RS Διαχωριστής γραμμών/εγγραφών αρχείου εισόδου, εξ ορισμού ο χαρακτήρας αλλαγής γραμμής (Record Separator)

FNR Τρέχον αριθμός γραμμής στο τρέχον αρχείο

FILENAME Όνομα αρχείου εισόδου

ARGC, ARGV Τα ορίσματα της γραμμής εντολών (πίνακας ARGV) και το πλήθος τους (ARGC). Σε αντίθεση με τους πίνακες της awk, ο πίνακας ARGV έχει δείκτες από 0 έως και ARGC-1.

ENVIRON Συσχετιστικός πίνακας που περιέχει τις μεταβλητές περιβάλλοντος

Παράδειγμα με NR και FNR

```
$ awk 'BEGIN {print "RecNr Name Telephone";
>              print "-------------------------"}
>      {printf "%5d %-10s %10s\n", NR, $1, $5}
>      END {print "-------------------------";
>           print "Total records:", FNR}' file.txt
RecNr Name              Telephone
-------------------------------
    1 Peter             2522041313
    2 Anna              2522041313
    3 Alice             2106399547
    4 Steffen           2310620789
    5 Wolfgang          2521091466
-------------------------------
Total records: 5
```

Παράδειγμα υπολογισμού μέσου όρου

Ο υπολογισμός του μέσου όρου του 3ου πεδίου όλων των εγγραφών του αρχείου *file.txt* γίνεται με την ακόλουθη εντολή:

```
$ awk 'BEGIN {sum = 0} {sum += $3}
>       END {print "Average:", sum/NR}' file.txt
Average: 180.4
```

Παράδειγμα με OFS

Η επόμενη εντολή επιλέγει το 1ο (username) και το 5ο (GECOS) πεδίο κάθε γραμμής του αρχείου /etc/passwd που αρχίζει με τον χαρακτήρα “n” και τα αποθηκεύει σ΄ ένα άλλο αρχείο αφού πρώτα αλλάξει τον διαχωριστή πεδίου εξόδου σε κόμμα.

```
$ awk -F':' 'BEGIN {OFS=","}
> /^[n]/ {print $1, $5}' /etc/passwd > users
$ cat users
news,news
nobody,nobody
nemo,Captain Nemo,,,
```

Πίνακες

Πέρα από τις απλές μεταβλητές, η awk παρέχει και μονοδιάστατους συσχετιστικούς (associative) πίνακες. Κάθε πίνακας αποτελείται από μία συλλογή ζευγαριών: έναν δείκτη και την τιμή του αντίστοιχου στοιχείου του πίνακα. Οι συγκεκριμένοι πίνακες δεικτοδοτούνται από αλφαριθμητικά και στην περίπτωση αριθμητικών δεικτών αυτοί μετατρέπονται αυτόματα σε αλφαριθμητικά. Η δημιουργία ενός πίνακα γίνεται με μια εντολή εκχώρησης τιμής:

array[index] = value

και όπως γίνεται φανερό, δεν γίνεται καμία δήλωση του μεγέθους του πίνακα, καθώς αυτό μπορεί και αλλάζει δυναμικά κατά τη διάρκεια της εκτέλεσης.

Πίνακες

Το παράδειγμα που ακολουθεί δηλώνει έναν πίνακα με τρία στοιχεία και εμφανίζει το ένα από αυτά:

```
$ awk 'BEGIN {
>       colors["red"] = "Rot";
>       colors["green"] = "Grün";
>       colors["blue"] = "Blau";
>       print "Meine Lieblingsfarbe ist",
>       colors["green"]}'
Meine Lieblingsfarbe ist Grün
```

Πίνακες

Για να ελέγξουμε αν υπάρχει ένα συγκεκριμένο στοιχείο σ' έναν πίνακα χρησιμοποιούμε την εντολή **in**:

index in array

ενώ η διαγραφή ενός στοιχείου γίνεται με την εντολή:

delete array[index]

ενώ η εντολή

delete array

διαγράφει όλα τα στοιχεία του πίνακα (πίνακας συνεχίζει όμως να υπάρχει).

Πίνακες

Το επόμενο παράδειγμα δείχνει τη χρήση τους:

```
$ awk 'BEGIN {
>       colors["red"] = "Rot";
>       colors["blue"] = "Blau";
>       delete colors["red"];
>       print "red" in colors, "blue" in colors}'
0 1                                    # false, true
```

Άσκηση 1

Να μετρηθούν οι λέξεις του αρχείου /usr/share/dict/american-english. Το αρχείο έχει μορφή:

```
$ cat /usr/share/dict/american-english
A
A's
AMD
...
zygote
zygote's
zygotes
$ dict=/usr/share/dict/american-english
```

Άσκηση 1 - Ενδεικτικές λύσεις

```
$ awk 'END {print NR}' $dict
104334
$ awk 'BEGIN {count=0} {count++}
>      END {print count}' $dict
104334
$ wc -l < $dict
104334
```

Άσκηση 1 - Ενδεικτικές λύσεις

```
$ sed -n '$=' $dict
104334   # $ refers to the last line of each file
$ export dict              # pass to child processes
$ cat linecounter.sh
LINECNT=0
while read -r LINE
do
  (( LINECNT++ ))
done < $dict
echo $LINECNT
$ ./linecounter.sh
104334
```

for

Η σάρωση όλων των στοιχείων ενός πίνακα γίνεται με τον ειδικό για πίνακες βρόχο **for-in** που έχει σύνταξη:

for (index in array)

do something with array[index]

Οι τιμές της ενσωματωμένης μεταβλητής ENVIRON είναι:

```
$ awk 'BEGIN {
>       for (key in ENVIRON)
>         print ENVIRON[key]}'
/home/nemo/bin:/home/nemo/usr/local/sbin
Ubuntu
...
```

for

Πέρα από τον βρόχο for-in που αναπτύχθηκε ειδικά για πίνακες, η awk παρέχει και τα τρία είδη βρόχων της γλώσσας προγραμματισμού C: **for**, **while** και **do-while**. Ο βρόχος for έχει σύνταξη:

for (*initialization*; *condition*; *increment*)

body

```
$ awk 'BEGIN {
>       sum = 0;
>       for (i = 1; i <= 10; i++)
>         sum += i
>       printf "%.2f\n", sum/10 }'
5.50
```

Παράδειγμα με for και ARGC/ARGV

```
$ cat script-04.awk
BEGIN {
  for (i = 0; i < ARGC; i++)
    print "ARGV[" i "]:", ARGV[i]
}
```

Ένα στιγμιότυπο εκτέλεσης του σεναρίου είναι:

```
$ awk -f script-04.awk Δράμα Καβάλα Σέρρες
ARGV[0]: awk
ARGV[1]: Δράμα
ARGV[2]: Καβάλα
ARGV[3]: Σέρρες
```

while – do while

Η γενική μορφή σύνταξης των βρόχων **while** και **do-while** είναι:

```
initialization
while (condition)
    statements
    increment
```

```
initialization
do
    statements
    increment
while (condition)
```

Παράδειγμα με while

Το πρώτο μας παράδειγμα τυπώνει 50 φορές τον χαρακτήρα “@”:

```
$ awk 'BEGIN {
>       i = 1
>       while (i++ <= 50)
>         ch = ch "@"                # concat
>       print ch
>     }'
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@
```

Παράδειγμα με do while

```
$ awk 'BEGIN{
>       i = 1
>       do {
>         print "This will be printed 3 times."
>         i++
>       } while (i < 4)
>     }'
This will be printed 3 times.
This will be printed 3 times.
This will be printed 3 times.
```

if

Με την εντολή **if** ελέγχεται η τιμή μιας έκφρασης και ανάλογα εκτελούνται ή όχι οι εντολές του σώματός της. Η εντολή if έχει την παρακάτω γενική μορφή σύνταξης:

if *(condition)*

statements

else

statements

Μέγιστη και ελάχιστη τιμή 3η στήλης

```
$ cat script-05.awk
# script-05.awk - max/min of 3rd column
{
  if (NR == 1){
     max = $3
     min = $3
  } else {
     if ($3 < min) min = $3
     if ($3 > max) max = $3
  }
}
END {print "Minimum:", min, "Maximum:", max}
```

Μέγιστη και ελάχιστη τιμή 3η στήλης

Η εκτέλεση του σεναρίου και η έξοδος είναι:

```
$ awk -f script-05.awk file.txt
Minimum: 167 Maximum: 189
```

Παράδειγμα – if, break, continue

```
$ cat script-06.awk
BEGIN {
    sum = i = 0
    while (1) {                         # infinite loop
        i++
        if (i % 2 == 0) continue        # odd or even
        sum += i
        if (sum > 30) break
    }
    print "Sum =", sum
}
```

Παράδειγμα – if, break, continue

Η εκτέλεση του σεναρίου και η έξοδος έχει ως εξής:

```
$ awk -f script-06.awk
Sum = 36
```

Παράδειγμα – if else if, frequencies

```
$ cat script-07.awk
BEGIN{
  freq["A"] = freq["B"] = freq["C"] = freq["D"] = 0
}
{
  if ($3 <= 180) freq["A"]++
  else if ($3 > 180 && $3 <= 185) freq["B"]++
  else if ($3 > 185 && $3 <= 190) freq["C"]++
  else freq["D"]++
}
```

Παράδειγμα - if else if, frequencies

```
END{
  print "Frequencies\n----------"
  for (key in freq)
    print key, freq[key]
}
```

Η εκτέλεση του σεναρίου και η έξοδος έχει ως εξής:

```
$ awk -f script-07.awk file.txt
Frequencies
----------
A 2
B 0
C 3
D 0
```

Αριθμητικές συναρτήσεις

Συνάρτηση	Περιγραφή
cos(x)	Επιστρέφει το συνημίτονο του x (το x σε ακτίνια).
exp(x)	Επιστρέφει το e^x.
int(x)	Επιστρέφει το ακέραιο μέρος του x.
log(x)	Επιστρέφει τον φυσικό λογάριθμο του x.
rand()	Επιστρέφει έναν τυχαίο αριθμό μεγαλύτερο ή ίσο του 0 και μικρότερο του 1.
sin(x)	Επιστρέφει το ημίτονο του x (το x σε ακτίνια).
sqrt(x)	Επιστρέφει την τετραγωνική ρίζα του x.
srand([x])	Αρχικοποιεί τη γεννήτρια τυχαίων αριθμών. Αν παραληφθεί το όρισμα χρησιμοποιείται η τρέχουσα ημερομηνία και ώρα του συστήματος.

Αλφαριθμητικές συναρτήσεις

Συνάρτηση	Περιγραφή
gsub(regex, sub[, t])	Αντικαθιστά όλες τις εμφανίσεις του regex με το sub. Το 3ο όρισμα είναι προαιρετικό και έχει προκαθορισμένη τιμή $0. Επιστρέφει το πλήθος των αντικαταστάσεων που έγιναν.
index(s, sub)	Ψάχνει την πρώτη εμφάνιση του sub στο s και επιστρέφει τη θέση στην οποία βρέθηκε αλλιώς επιστρέφει την τιμή μηδέν.
length([s])	Επιστρέφει το πλήθος των χαρακτήρων του s. Χωρίς όρισμα επιστρέφει το μήκος του $0.
match(s, regex)	Ψάχνει την μεγαλύτερη ταύτιση του regex στο s και επιστρέφει τη θέση στην οποία βρέθηκε, αλλιώς μηδέν. Ενημερώνει τις προκαθορισμένες μεταβλητές RSTART και RLENGTH.

Αλφαριθμητικές συναρτήσεις

split(s, array [, sep])	Καταχωρεί τα τμήματα του s που χωρίζονται μεταξύ τους με τον χαρακτήρα sep στα στοιχεία του πίνακα array. Προκαθορισμένος διαχωριστικός χαρακτήρας είναι η τιμή της μεταβλητής FS.
sprintf(format, expr1, …)	Δημιουργία ενός αλφαριθμητικού που προκύπτει από το format, αφού γίνει η αντικατάσταση των τιμών της λίστας ορισμάτων.
sub(regex, sub[, t])	Αντικαθιστά την πρώτη εμφάνιση του regex με το sub. Το 3ο όρισμα είναι προαιρετικό και έχει προκαθορισμένη τιμή $0. Επιστρέφει το πλήθος των αντικαταστάσεων που έγιναν (0 ή 1).

Αλφαριθμητικές συναρτήσεις

substr(s, start [, length])	Επιστρέφει το τμήμα του αλφαριθμητικού s, που αρχίζει από τη θέση start και έχει μήκος length. Αν παραληφθεί το 3ο όρισμα επιστρέφει ολόκληρο το τμήμα από τη θέση start.
tolower(s)	Επιστρέφει ένα αντίγραφο του s, στο οποίο τυχόν κεφαλαία λατινικά γράμματα μετατράπηκαν σε πεζά.
toupper(s)	Επιστρέφει ένα αντίγραφο του s, στο οποίο τυχόν πεζά λατινικά γράμματα μετατράπηκαν σε κεφαλαία.

Άλλες Συναρτήσεις

Τέλος, θα πρέπει να αναφέρουμε τις συναρτήσεις **system(**cmd**)**, η οποία εκτελεί την εντολή cmd και επιστρέφει τον κωδικό εξόδου της εντολής και την **getline(**[s]**)**, η οποία αποθηκεύει την επόμενη γραμμή του αρχείου εισόδου στην μεταβλητή s ή στην $0 εφόσον κληθεί χωρίς όρισμα.

Παράδειγμα με συναρτήσεις - 1

```
$ awk 'BEGIN {          # sin() and cos() example
>   PI = 3.14159
>   a = 60
>   c = cos(angle * PI / 180.0)
>   s = sin(angle * PI / 180.0)
>   printf "Cosine of %d degrees is %.2f\n", a, c
>   printf "Sinus of %d degrees is %.2f\n", a, s
> }'
Cosine of 60 degrees is 0.50
Sinus of 60 degrees is 0.87
```

Παράδειγμα με συναρτήσεις - 2

```
$ awk 'BEGIN {          # rand() and int() example
>   num = rand()
>   print "Random number", num
>   print "Random integer", int(6 * num)
> }'
Random number 0.906608
Random integer 5
```

Παράδειγμα με συναρτήσεις - 3

```
$ cat script-08.awk
# script-08.awk - string build-in functions
BEGIN {
  str = "das ist das Haus vom Nikolaus"
  print length(str)
  print index(str, "ist")
  print tolower(str), "\n", toupper(str)
  print gsub(/das/, "this", str), str
  split(str, array); print array[1], array[6]
  pi = sprintf("π = %.2f (approximate)", 22/7);
  print pi
}
```

Παράδειγμα με συναρτήσεις - 3

```
$ awk -f script-08.awk
29
5
das ist das haus vom nikolaus
 DAS IST DAS HAUS VOM NIKOLAUS
2 this ist this Haus vom Nikolaus
this Nikolaus
π = 3.14 (approximate)
```

Συναρτήσεις (UDF)

Ο ορισμός συναρτήσεων χρήστη μπορεί να γίνει σε οποιοδήποτε σημείο του προγράμματος καθώς πρώτα διαβάζεται ολόκληρος ο κώδικας και στη συνέχεια γίνεται η εκτέλεσή του. Η σύνταξη μιας συνάρτησης έχει την παρακάτω γενική μορφή:

function name(*[parameter-list]***)**
{
   statements
   return *[expression]*
}

Συναρτήσεις (UDF)

```
$ cat script-09.awk
function testfunc()
{
  if ($5 != "")    # if GECOS field is not empty
    printf "'%s' has gecos field: '%s'\n", $1, $5
  else
    printf "'%s' has no gecos field\n", $1
}
BEGIN{ FS = ":" } # change Field Separator to ":"
{
  testfunc()      # call function testfunc()
}
```

Συναρτήσεις (UDF)

Η εκτέλεση και έξοδος του σεναρίου script-09.awk έχει ως εξής:

```
$ awk -f script-09.awk /etc/passwd
'root' gecos field: 'root'
...
'nemo' gecos field: 'Captain Nemo,K16.208,5322,'
'alice' gecos field: 'Alice Cooper,OS Lab,9876,'
'bob' gecos field: 'Bob Marley,OS Lab,9876,'
'carol' gecos field: 'Carol Spencer,OS Lab,9876,’
'heidi' has no gecos field
'ivan' has no gecos field
```

Άσκηση 2

Να μετατρέψετε σε κεφαλαία τα τυχόν πεζά γράμματα μιας φράσης:

```
$ words="Magister dixit"
$ echo $words
Magister dixit
```

Άσκηση 2 - Ενδεικτικές λύσεις

```
$ echo $words | tr [a-z] [A-Z]
MAGISTER DIXIT
$ echo $words | tr [:lower:] [:upper:]
MAGISTER DIXIT
$ echo $words | awk '{ print toupper($0) }'
MAGISTER DIXIT
$ echo ${words^^}
MAGISTER DIXIT
```

Άσκηση 3

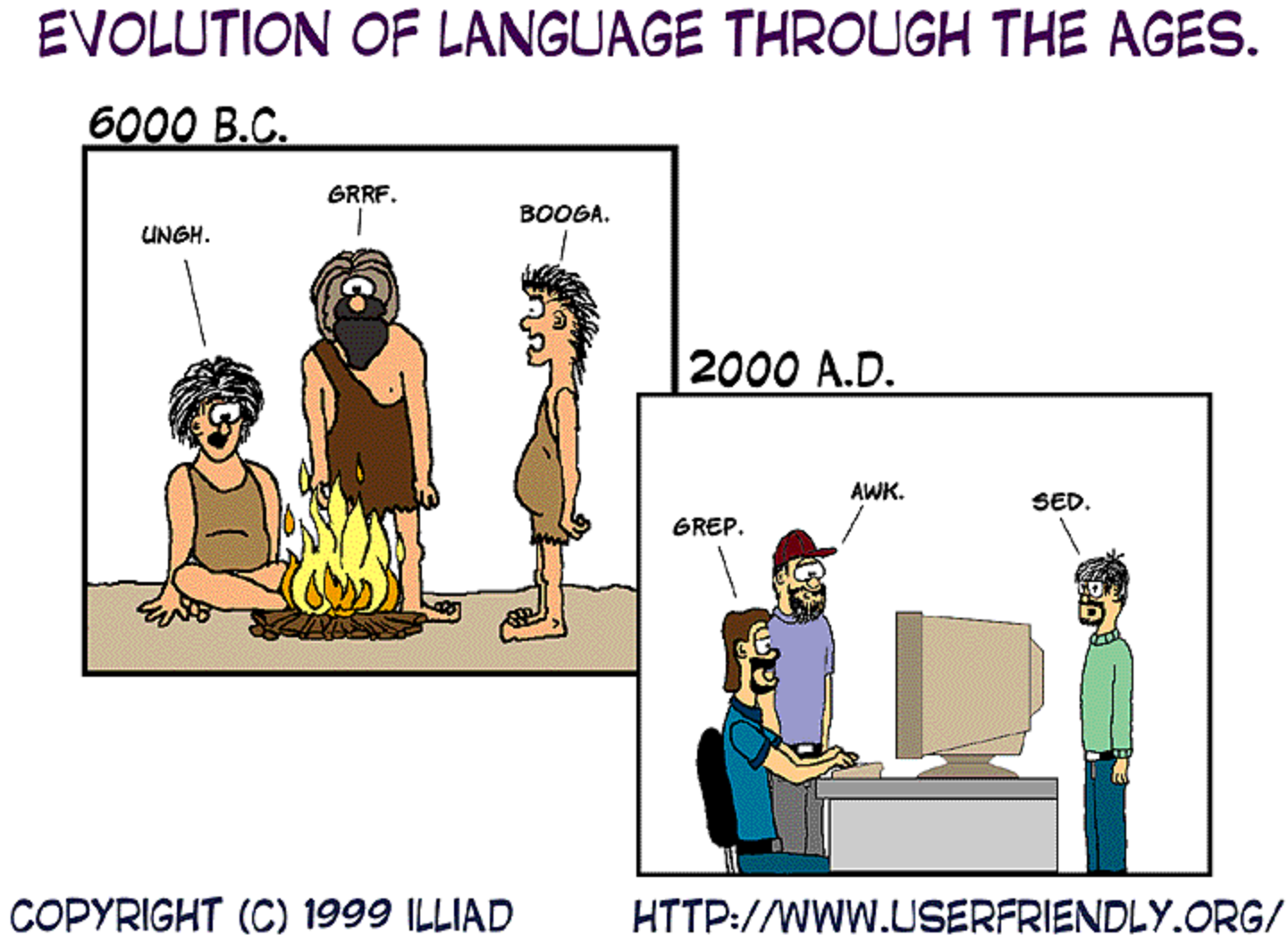
Να υπολογιστεί η ηλικία κάθε εγγραφής του αρχείου *file.txt*. Το 4ο πεδίο περιέχει το έτος γέννησης κάθε ατόμου.

```
$ cat file.txt
Peter Frampton 188 1962 2522041313
Anna Frampton 170 1963 2522041313
Alice Bauhaus 167 1965 2106399547
Steffen Reichert 188 1963 2310620789
Wolfgang Hartmann 189 1958 2521091466
```

Άσκηση 3 - Ενδεικτικές λύσεις

```
$ cat script-10.awk
{ print $1 " is " cyear-$4 " years old." }
$ awk -v cyear=`date +%Y` -f script-10.awk file.txt
Peter is 62 years old.
...
Wolfgang is 66 years old.
$ cat script-11.awk
BEGIN { cyear = strftime("%Y", systime()) }  # gawk
{ print $1 " is " cyear-$4 " years old." }
$ awk -f script-11.awk file.txt
Peter is 62 years old.
...
```

Ερωτήσεις


EVOLUTION OF LANGUAGE THROUGH THE AGES.
6000 B.C.
UNGH.
GRRF.
BOOGA.
2000 A.D.
GREP.
AWK.
SED.
COPYRIGHT (C) 1999 ILLIAD
HTTP://WWW.USERFRIENDLY.ORG/