



Διάλεξη 7 Εργαλεία Ωφελιμότητας Unix: Awk

Δημήτρης Ζεϊναλιπούρ



Επεξεργασία Ρεύματος

- sed (Stream EDitor)
- **awk** (Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan)
 - Δημιουργείται στα Bell Labs από τον ιδρυτή της γλώσσας C και του UNIX.
 - Η πιο ισχυρή της έκδοση “ονομάζεται” Perl
 - ο Larry Wall, δημιουργός της Perl, εμπνέεται από την awk.

Σημείωση: Μελετήστε τα εγχειρίδια που υπάρχουν στην ιστοσελίδα και το man στο UNIX.

Εισαγωγή στην awk



- Μια **γλώσσα προγραμματισμού** σχεδιασμένη για να **βρίσκει, ταιριάζει** **πρότυπα** και να **εκτελεί ενέργειες** σε **αρχεία – ρεύματα εισόδου**.
 - **παίρνει είσοδο από αρχεία, ανακατεύθυνση, διοχέτευση και απευθείας από το προκαθορισμένο ρεύμα εισόδου**

Εισαγωγή στην awk



- Υπάρχουν πολλοί τρόποι να τρέξουμε ένα *awk* πρόγραμμα
 - ***awk 'program' input_file(s)***
 - *program* και *input files* παρέχονται **ως ορίσματα γραμμής εντολής**
 - ***awk 'program'***
 - *program* είναι ένα όρισμα γραμμής εντολής. Η είσοδος δίνεται από το προκαθορισμένο **ρεύμα εισόδου** ή από **διοχέτευση**
 - ***awk -f program_file input_files***
 - *program* διαβάζεται από αρχείο
 - ***awk is ERE***

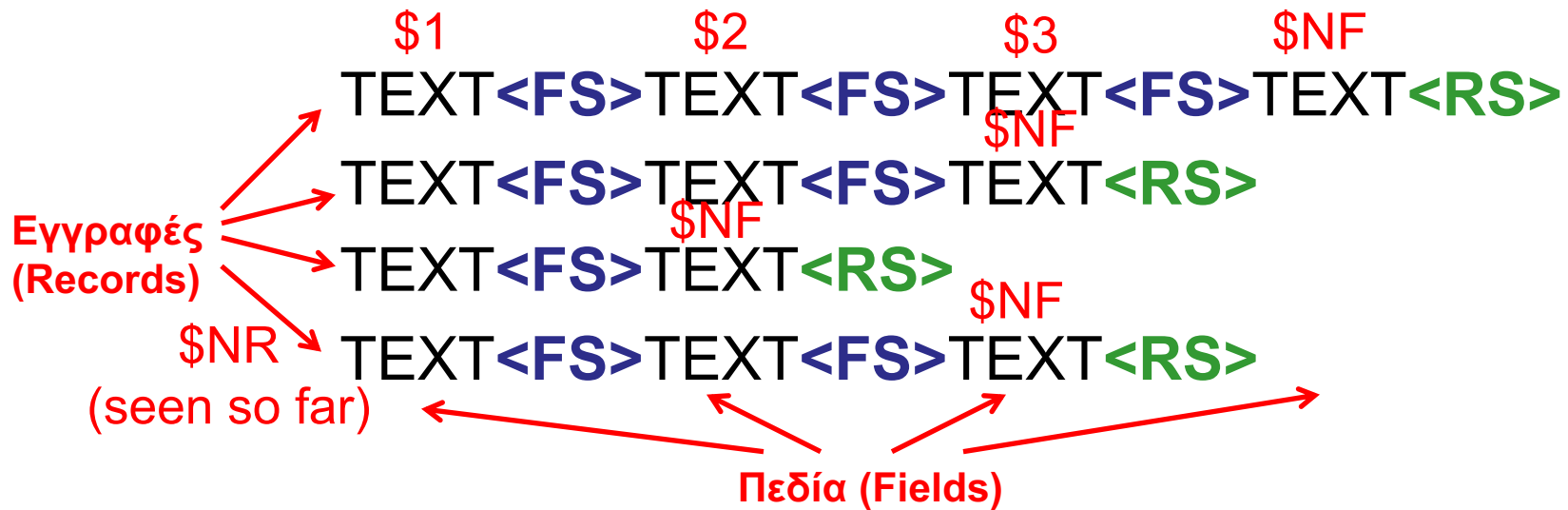
Λογική Επεξεργασίας της awk

Περιγραφή της AWK από τον Alfred V. Aho (ένα από τους δημιουργούς της AWK):

- Η AWK είναι **γλώσσα** για **επεξεργασία** αρχείων κειμένου.
- Ένα αρχείο επεξεργάζεται ως μια **ακολουθία εγγραφών** (records), και εξ' ορισμού κάθε **γραμμή** είναι **ένα record**.
- Κάθε **γραμμή** τέμνεται σε **πεδία (fields)**.
- Ένα AWK πρόγραμμα είναι μια ακολουθία (sequence) από εκφράσεις **pattern-action**.
- Το AWK διαβάζει από το input **μία γραμμή κάθε φορά** και για κάθε γραμμή **εξετάζεται** το κάθε **pattern ξεχωριστά**.
- Για κάθε **pattern** που ταιριάζει εκτελείται το αντίστοιχο **action**.

Λογική Επεξεργασίας της awk

Ροή Εισόδου στην AWK



\$0: complete line

FS: Input File Separator (space by default)

RS: Record Separator (newline by default)

NR: Number of Records seen so far in input

NF: Number of Fields in current input record

Λογική Επεξεργασίας της awk

Ροή Εξόδου από την AWK

text<OFS>text<OFS>text<OFS>text<ORS>

OFS: Output File Separator (space by default)

ORS: Output Record Separator (newline by default)

Δομή Προγράμματος awk (Παράδειγμα Εκτέλεσης)



Παράδειγμα

> ls -al

total 17600

drwxr-xr-x 2 dzeina faculty 4096 May 5 2009 .

drwxr-xr-x 10 dzeina faculty 4096 Jan 14 15:55 ..

-rw-r--r-- 1 dzeina faculty 967047 Feb 1 01:53 01.pdf

-rw-r--r-- 1 dzeina faculty 519830 Feb 1 01:53 02.pdf

-rw-r--r-- 1 dzeina faculty 432031 Feb 1 01:54 03.pdf

-rw-r--r-- 1 dzeina faculty 738217 Feb 2 13:56 04.pdf

Εκτύπωση 3ης και 4ης στήλης (όπως όμοιο με την ls -al | cut -d" " -f4,5)

> ls -al | awk -F" " '{print \$3" "\$4}'

dzeina faculty

dzeina faculty

dzeina faculty

dzeina faculty

dzeina faculty

dzeina faculty

Δομή Προγράμματος awk (Ζεύγη Pattern {Action})



- Όπως αναφέραμε, η awk **σαρώνει** τις γραμμές **εισόδου μια-μια**, ψάχνοντας να δει ποια γραμμή ταιριάζει με ένα **σύνολο προτύπων (patterns)** ή **συνθηκών (conditions)** που δίνονται στην awk.
- Για κάθε **pattern**, προσδιορίζεται μια πράξη (**action**). Η πράξη εκτελείται όταν το πρότυπο ταιριάζει αυτό του input line.

```
awk <options> pattern { action }; pattern { action }
```

- Τα Actions συμπεριλαμβάνονται μέσα σε **curly παρενθέσεις** και διαχωρίζονται με **semi-colon ‘;’**

Δομή Προγράμματος awk (Επιλογές – Options)



awk **<options>** *pattern { action }; pattern { action }*

- Υπάρχουν κάποια **POSIX options** (με -) και κάποια άλλα **GNU options** (με --)
- Οι περισσότερες επιλογές δε θα είναι πολύ χρήσιμες στα πλαίσια του μαθήματος.
- Κάποιες χρήσιμες επιλογές είναι οι ακόλουθες:
 - **-F fs** : Προσδιορισμός του Field Separator από το command line.
Π.χ., **ls -al | awk -F"." '{print \$2}'**.
 - Εναλλακτικά μπορεί να προσδιοριστεί στο BEGIN block που θα δούμε σε λίγο!
 - **-W help**: Εκτύπωση μιας σύνοψης επίλογων.
 - **-f program-file**: Ανάγνωση προγράμματος της sed από **αρχείο** (στο οποίο περιέχεται ένα **pattern-action** ανά **γραμμή**). Επίσης πολλαπλά αρχεία μπορούν να προσδιοριστούν με -f

Δομή Προγράμματος awk (`'pattern {action}'`)



- Στο προηγούμενο παράδειγμα **ΔΕΝ** υπήρχε το **pattern**. Υπήρχε απλά το action:

```
awk '{print $3"\t"$4}'
```

→ Κενό(Pattern)
Εκτέλεση Action σε
κάθε γραμμή

– Αυτό υποδήλωνε ότι το action θα έπρεπε να εκτελείται σε κάθε γραμμή εισόδου.

- Ήταν δηλαδή αντίστοιχο της ακόλουθης εντολής (όπου το 1 υποδηλώνει το pattern TRUE, δηλ., ισχύει πάντα)

```
awk -F" " '1 {print $3"\t"$4} '
```

- Κατ' αντίστοιχο τρόπο, θα μπορούσε να είχε παραληφθεί το action, π.χ.,

```
awk -F" " '1;1 {print $3"\t"$4} '
```

← Κενό(Action)
Εκτύπωση Ολόκληρης της
γραμμής και μετά της
στήλης 3-4

Δομή Προγράμματος awk



- Πιο συγκεκριμένα, ένα awk πρόγραμμα αποτελείται από:
 - Το **προαιρετικό τμήμα *BEGIN***
 - εκτελείται **πριν** την επεξεργασία της πρώτης γραμμής εισόδου.
 - Ένα ή περισσότερα (***pattern {action}***)
 - επεξεργασία δεδομένων εισόδου
 - Για κάθε πρότυπο που ταιριάζει, η αντίστοιχη ενέργεια εκτελείται
 - Το **προαιρετικό τμήμα *END***
 - εκτελείται μετά το τέλος επεξεργασίας της τελευταίας γραμμής εισόδου.

```
[BEGIN  
{action;action};]  
  
pattern {action;action};  
pattern {action};  
  
.  
.  
.  
  
pattern { action};
```

```
[END {action}]
```

Σημείωση:

(η σειρά διατύπωσης
begin, end μπορεί να
είναι τυχαία)

Πρότυπα (Patterns) στην awk

- **Πρότυπο (Pattern):** Προσδιορισμός κατά πόσον μια *ενέργεια-δράση* πρόκειται να εκτελεστεί
 - Μπορεί να είναι:

1. Το ειδικό *token* **BEGIN** ή **END**

```
• $ ls | awk '  
  > BEGIN {print "Start"; print "NOW";} → Action 1  
  > {print "LINE: "; {print "IS:", $0} → Action 2  
  > END {print "End"}' → Action 3
```

→ Action 4

- **Επιστρέφει:**

```
Start → Action 1 BEGIN  
NOW  
-----  
LINE: → Action 2  
IS:01.pdf → Action 3  
LINE: → Action 2  
IS:02.pdf → Action 3  
LINE: → Action 2  
IS:03.pdf → Action 3  
LINE: → Action 2  
IS:04.pdf → Action 3  
End → Action 4 END
```

Πρότυπα (Patterns) στην awk

2. Κανονική έκφραση (ERE) (εσωκλειόμενο μέσα σε / /), όπως με την sed -E, egrep, κτλ.

```
- ls | awk
> 'BEGIN {print "Start..."}
> /03/ {print $0}
> END {print "....End"} '
```

Diagram illustrating the awk command structure with red arrows pointing to the components:

- Red arrow from `/03/` to the word **Pattern**.
- Red arrow from `{print $0}` to the word **Action**.

– Επιστρέφει:	Pattern {Action}
Start...	Εκτύπωση γραμμής εισόδου
03.Pdf	που ικανοποιεί την κανονική
....End	έκφραση /03/ (περιέχει το 03)

Πρότυπα (Patterns) στην awk

3. Λογικοί και Σχεσιακοί Τελεστές:

– Μοναδιαίος Λογικός Τελεστής (Άρνησης): !

- Π.χ., `ls | awk BEGIN {print "Start..."} !/03/ {print $0} END {print "....End"} '`
- Εκτυπώνει τα πάντα που **δεν** ικανοποιούν την Κανονική Έκφραση `/03/`

– Σχεσιακοί Τελεστές: > < >= <= ==

– Δυαδικοί Λογικοί Τελεστές: && ή ||

- π.χ. Δεν περιέχει το 03 και είναι μεγαλύτερο του 10 (αλφαριθμητικά)
- `ls | awk '`
 - > `BEGIN {print "Start..."}`
 - > `!/03/ && substr($0,0,2)>=10 {print $0}`
 - > `END {print "....End"} '`
 - η **substr** είναι συνάρτηση που εξαγάγει από την γραμμή \$0 από την 0 στήλη μέχρι την 2 στήλη). Περισσότερα για συναρτήσεις αργότερα.

– **Macro Expressions: Pattern1 ? Pattern2 : Pattern3**-16

- *If Pattern1 is true, then Pattern2, else Pattern3*

Παραδείγματα Εκτέλεσης awk



- Παραδείγματα

```
bash-3.1$ cat example.awk
```

```
Line number 1  
Line number 2  
Line number 3  
Line number 4
```

Τυπώνει την γραμμή που περιλαμβάνει το 2

```
bash-3.1$ awk '/2/ {print}' example.awk
```

```
Line number 2
```

```
bash-3.1$ awk '{print $3}' example.awk
```

```
1  
2  
3  
4
```

Τυπώνει την τρίτη στήλη του αρχείου (χωρίς πρότυπο), συνεπώς για κάθε γραμμή.

Παραδείγματα Εκτέλεσης awk



- Παραδείγματα

```
bash-3.1$ cat example.awk
```


```
Line number 1
```

```
Line number 2
```

```
Line number 3
```

```
Line number 4
```

To BEGIN block εκτυπώνει το "Hello you"
To main τυπώνει την γραμμή που περιλαμβάνει το
2



```
bash-3.1$ awk '/2/ {print $0} BEGIN {print "Hello  
you"} END {print "goodbye"}' example.awk
```

```
Hello you
```

```
Line number 2
```

```
goudbye
```

Παραδείγματα Εκτέλεσης awk



- Παραδείγματα

```
bash-3.1$ cat example2.awk
```

Just a text file. Nothing to see here.

Some lines have

more fields than others

and some

← Κενή γραμμή

are blank.

Εκτύπωσε τις γραμμές των οποίων τα πεδία είναι > 3

```
bash-3.1$ awk 'NF>3 {print $0}' example2.awk
```

Just a text file. Nothing to see here.

more fields than others

NF: Number of Fields

Παραδείγματα Εκτέλεσης awk



- Παραδείγματα

```
bash-3.1$ cat example2.awk
```

Just a text file. Nothing to see here.

Some lines have

more fields than others

and some

← Κενή γραμμή

are blank.

↑ Εκτύπωσε τις γραμμές των οποίων τα πεδία είναι > 3 ή είναι κενές

```
bash-3.1$ awk 'NF>3 || /^$/ {print $0}' example2.awk
```

Just a text file. Nothing to see here.

more fields than others

← Κενή γραμμή

NF: Number of Fields

Παραδείγματα Εκτέλεσης awk



- Παραδείγματα

```
bash-3.1$ cat example2.awk
```

Just a text file. Nothing to see here.

Some lines have

more fields than others

and some

are blank.

If (NF>3) then if /file/ then print \$0

Else if /^and/ then print \$0

Εκτύπωσε τις γραμμές των οποίων τα πεδία είναι > 3 και καλύπτουν το πρότυπο /file/ αλλιώς εκτύπωσε τις γραμμές των οποίων τα πεδία είναι <=3 και ξεκινούν με το and

```
bash-3.1$ awk 'NF>3 ? /file/ : /^and/ {print $0}' example2.awk
```

Just a text file. Nothing to see here.

and some

Όνομα Αρχείου

Μέγεθος αρχείου

```
file+1.txt 0
output.txt 4898
processes.txt 12953
test-cut.txt 55
test-sort.txt 124
test-tr.txt 40
```

There you go!!

Εκτύπωσε το όνομα του αρχείου,
που έχει προέκταση .txt και το
μέγεθός του

- BRE: `.*[]^$ \ + | \?` BRE Char: `\ . * ^ $ \ \ V + | ?`
- ERE: `.*[]^$ \ + | ?` ERE Char: `\ . * ^ $ \ \ V | + | \?`

Παραδείγματα Εκτέλεσης awk

Number of Fields

- Παραδείγματα από Actions
 - **{print NF, \$1, \$NF}**
 - ΤΥΠΩΝΕΙ ΤΟΝ **αριθμό των πεδίων**, το **πρώτο πεδίο**, και το **τελευταίο πεδίο** στο υφιστάμενο record
 - **{print \$(NF-2)}**
 - ΤΥΠΩΝΕΙ ΤΟ **τρίτο από το τέλος πεδίο**.
 - **{print \$1, \$2*\$3}**
 - ΤΥΠΩΝΕΙ ΤΟ **πρώτο πεδίο** και το **αποτέλεσμα του υπολογισμού** – (πολλαπλασιασμός του δεύτερου και τρίτου πεδίου)
 - **{print NR, \$0}**
 - Προσθέτει τον αριθμό της υφιστάμενης γραμμής πριν κάθε γραμμή
 - **{print "total pay for", \$1, "is", \$2*\$3}**
 - προσθέτεις κείμενο για εκτύπωση στην έξοδο μαζί με τα πεδία της υφιστάμενης γραμμής
 - Το κείμενο εσωκλείεται με ""

Number of Records

Αριθμητικές Πράξεις στην AWK

- Ενέργειες – Δράσεις
 - Περιλαμβάνει τους γνωστούς τελεστές, όπως στη C
 - **++** Increment, **--** Decrement
 - **^** Exponentiation
 - **+** **-** **!** Plus, Minus, NOT
 - ***** **/** **%** Multiplication, division, modulus

Χρήση Συνθήκης στο Πρότυπο (Pattern)



- **Επιλογή βάσει Σύγκρισης**
 - `$2 >= 5 { print } // ίδιο με print $0`
- **Επιλογή βάσει Υπολογισμού**
 - `$2 * $3 > 50 { printf ("%6.2f for %s\n", $2 * $3, $1) }`
- **Επιλογή βάσει Περιεχομένου Κειμένου**
 - `$1 == "NYU"`
 - `$2 ~ /NYU/ => Το $2 περιέχει το REGEX /NYU/.`
Πολύ σημαντικός **τελεστής** στις κανονικές εκφράσεις, εφόσον το `/NYU/` θα εφαρμοζόταν σε όλη την γραμμή!
- **Επιλογή βάσει λογικής έκφρασης**
 - `$2 >= 4 || $3 >= 20`

Διαμόρφωση Εξόδου με την printf()



- Καλύτερη Διαμόρφωση Εξόδου
 - Χρήση της **printf** " " όπως στη C
 - **awk '{ printf [(] "format", var1, var2, var3, ... [)] }'**

– Παράδειγμα

Χαρακτήρας Δολαρίου στο output

```
awk '{ printf "total pay for %s is  
$%.2f\n", $1, $2 * $3 }'
```

- Όταν χρησιμοποιούμε την **printf()** η διαμόρφωση της εξόδου είναι στα χέρια μας (κενά διαστήματα, νέα γραμμή, κλπ)

```
{ printf("%-8s %6.2f\n", $1, $2 * $3 ) }
```

Left align within 8 spaces (by default right alignment)

6 μονάδες + 2 ψηφία δεκαδικής ακρίβειας

Παραδείγματα Εκτέλεσης awk

- Παραδείγματα

```
bash-3.1$ ls -l | awk 'BEGIN {print "List of all .txt  
files"} /\.txt$/ && $5>0 && NR>15 {print "line number:"  
NR, "file", $9, "of size:", $5} END {print "There you  
go!!"}'
```

List of all .txt files

line number:16 file test-cut.txt of size: 55

line number:19 file test-sort.txt of size: 124

line number:20 file test-tr.txt of size: 40

There you go!!

Παραδείγματα Εκτέλεσης awk

- Παραδείγματα

```
bash-3.1$ ls -l | awk 'BEGIN {print "List of all .txt  
files"} /\.txt$/ && $5>0 && NR>15 && $6=="Feb" {print  
"line number:" NR, "file", $9, "of size:", $5} END  
{print "There you go!!"}'
```

List of all .txt files

line number:16 file test-cut.txt of size: 55

line number:20 file test-tr.txt of size: 40

There you go!!

ή

```
bash-3.1$ ls -l | awk 'BEGIN {print "List of all .txt  
files"} /\.txt$/ && $5>0 && NR>15 && $6=="Feb"  
{printf("line number: %3d\tfile %15s of size %4d\n", NR,  
$9, $5)} END {print "There you go!!"}'
```

List of all .txt files

line number:	16	file	test-cut.txt of size	55
--------------	----	------	----------------------	----

line number:	20	file	test-tr.txt of size	40
--------------	----	------	---------------------	----

Μεταβλητές στην awk



- Μεταβλητές

- Η *awk* μπορεί να **ορίσει** και να **χρησιμοποιήσει** **μεταβλητές**

```
$ ls | awk '
```

```
> BEGIN { sum = 0 }
```

```
> {sum++}
```

```
> END {print sum} '
```

Όχι απαραίτητο

- Οι μεταβλητές αυτές μπορούν να πάρουν **αριθμητική (ακέραια ή πραγματική)** τιμή ή **συμβολοσειρά**

- ΔΕΝ ΔΗΛΩΝΕΤΑΙ ο τύπος τους αλλά αναγνωρίζεται από τα συμφραζόμενα. Εξ' ορισμού, οι μεταβλητές που ορίζουμε αρχικοποιούνται με την αριθμητική τιμή 0 (→ *null* string)

- Εάν θέλουμε να κάνουμε cast ένα float σε ακέραια τιμή εκτελούμε την συνάρτηση **int(sum)**.

- Υπάρχουν διαφορές μαθ. συναρ. (cos, log, ran, κτλ), ελεξτε-29 το manual.

Παραδείγματα Εκτέλεσης awk (Μεταβλητές)



- Παραδείγματα

```
awk $3 > 15 { emp = emp + 1 }  
END { print emp, "employees worked more than 15 hrs" }
```

Τυπώνει πόσοι employees δούλεψαν πάνω από 15 ώρες

```
{ pay = pay + $2 * $3 }  
END { print NR, "employees"  
      print "total pay is", pay  
      print "average pay is", pay/NR  
}
```

Υπολογίζει το άθροισμα και τον μέσο όρο των μισθών που υπολογίζεται από την 2^η και 3^η στήλη.

Συνένωση Συμβολοσειρών στην awk



- Συνένωση Συμβολοσειρών
 - Νέες συμβολοσειρές μπορούν να δημιουργηθούν με τη συνένωση παλιών

```
{ names = names $1 " " } END { print names }
```

- Παράδειγμα

Concat column \$9 to string, then print out line number. At the end print out name variable.

```
bash-3.1$ ls -l | awk 'BEGIN {print "List of all  
.txt files"} /\..txt$/ && $5>0 && NR>15 && $6=="Feb"  
{names = names $9 " "; printf("line number:  
%3d\tfile %15s of size %4d\n", NR, $9, $5)} END  
{print "There you go!!"; print names}'
```

List of all .txt files

line number: 16	file	test-cut.txt	of size	55
line number: 20	file	test-tr.txt	of size	40

There you go!!

test-cut.txt test-tr.txt

← string concatenation

Έλεγχος Ροής Δεδομένων AWK Control Statements



- **if (*condition*) then-body [else else-body]**
 - awk '{ if (x % 2 == 0) print "x is even"; else print "x is odd" }'
- **while (*condition*) body**
 - awk '{ i = 1 while (i <= 3) { print \$i i++ } }'
- **do body while (*condition*)**
 - awk '{ i = 1 do { print \$0 i++ } while (i <= 10) }'
- **for (*initialization; condition; increment*) body**
 - awk '{ for (i = 1; i <= 3; i++) print \$i }'
- Άλλα: **break, continue, exit, next (record)**, κτλ.

Εκτύπωση i-οστής στήλης

Εκτύπωση Όλων των στηλών πλην των πρώτων δυο



```
$ ls -al
```

```
total 4
drwx----- 2 dzeina faculty 70 Feb 28 11:48 .
drwx-----x 36 dzeina faculty 4096 Feb 28 11:11 ..
-rw----- 1 dzeina faculty 0 Feb 28 11:48 a
-rw----- 1 dzeina faculty 0 Feb 28 11:48 b
-rw----- 1 dzeina faculty 0 Feb 28 11:48 c
-rw----- 1 dzeina faculty 0 Feb 28 11:48 d
-rw----- 1 dzeina faculty 0 Feb 28 11:48 e
```

```
$ ls -al | awk '{$1=$2=""; print $0}'
```

```
dzeina faculty 70 Feb 28 11:48 .
dzeina faculty 4096 Feb 28 11:11 ..
dzeina faculty 0 Feb 28 11:48 a
dzeina faculty 0 Feb 28 11:48 b
dzeina faculty 0 Feb 28 11:48 c
dzeina faculty 0 Feb 28 11:48 d
dzeina faculty 0 Feb 28 11:48 e
```


Εκτύπωση των τελευταίων 5 στηλών



```
$ ls -al
```

```
total 4
```

```
drwx----- 2 dzeina faculty 70 Feb 28 11:48 .
```

```
drwx-----x 36 dzeina faculty 4096 Feb 28 11:11 ..
```

```
-rw----- 1 dzeina faculty 0 Feb 28 11:48 a
```

```
-rw----- 1 dzeina faculty 0 Feb 28 11:48 b
```

```
-rw----- 1 dzeina faculty 0 Feb 28 11:48 c
```

```
-rw----- 1 dzeina faculty 0 Feb 28 11:48 d
```

```
-rw----- 1 dzeina faculty 0 Feb 28 11:48 e
```

```
$ ls -al | awk '{for(i=6;i<=NF;i++) printf $i" "; print ""}'
```

```
or ls -al | awk '{for(i=1;i<=5;i++) $i="";print $0}'
```

```
Feb 28 11:48 .
```

```
Feb 28 11:11 ..
```

```
Feb 28 11:48 a
```

```
Feb 28 11:48 b
```

```
Feb 28 11:48 c
```

```
Feb 28 11:48 d
```

```
Feb 28 11:48 e
```