ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

Λειτουργικά Συστήματα

SED and AWK

Σύνδεση με τα προηγούμενα

- Από πού αντλείται η δύναμη του προγραμματισμού SHELL ?
  - Στο Unix όλα είναι αρχεία
  - Output redirection και pipes είναι βασικά εγγραφή/ανάγνωση κατάλληλων file descriptors.
  - Το περιβάλλον του κελύφους έχει λογική όπως οι γλώσσες προγραμματισμού: for, foreach, if-else, switch, variables, evaluation ...
  - Υπάρχουν άπειρες μικρές εντολές των οποιών τα αποτελέσματα, χρήσει των παραπάνω, μπορούν να συνδυαστούν:
    finger Bob | grep Login | awk ’{print$2":"$4""$5}’ | lp

- Η διαχείριση κειμένου δίνει επιπλέον δύναμη
  - **sed** και **awk.**

sed (stream editor)

- Είσοδος από αρχείο ή pipe / Έξοδος αντίστοιχα
- Φιλτράρει και επεξεργάζεται το κείμενο εισόδου και το δίνει στην έξοδο, τροποποιημένο κατά τον τρόπο που επισημάνθηκε και δίχως ισχυρή επανατροφοδότηση: συνήθως με κάποια εύρεση pattern και αντικατάσταση
- Επεξεργάζεται αρχεία κειμένου ανά **γραμμές**
- Τον χρησιμοποιούμε όταν η γνώση για την πληροφορία δεν βρίσκεται τόσο στη δομή του αρχείου (γραμμές, στήλες) όσο στο κείμενο του

Αρχή Λειτουργίας

- **pattern space** = το υφιστάμενο κείμενο όπως επεξεργάζεται (data buffer)

while (!EOF)
{
  - διαβάζει μια γραμμή εισόδου στο pattern space
  - εκτελεί μια σειρά από εντολές επεξεργασίας επί του pattern space
  - γράφει το pattern space στο stdout

}

Σύνταξη εντολής

- `sed <options> '<address> <command>'`

- **address**
  - ένας **αριθμός γραμμής** του κειμένου εισόδου ή ένα **pattern** που εμπεριέχεται σε slashes (/pattern/).
  - Καθορίζει **πού** θα εφαρμοστεί η εντολή, δηλαδή σε ποιές γραμμές
  - Αν δεν ορίζεται, η sed θα εφαρμοστεί σε **κάθε** γραμμή της εισόδου
  - Το pattern περιγράφεται με regular expressions, όπως στη grep
  - Εάν δοθούν δυο διευθύνσεις χωρισμένες με κόμμα, η εντολή λειτουργεί στο πεδίο γραμμών μεταξύ της πρώτης και δεύτερης διεύθυνσης, συμπεριλαμβανομένων των ιδίων
  - ! = NOT (εφαρμογή στις διευθύνσεις εκτός αυτών που περιγράφτηκαν)

Συνήθεις Εντολές Επεξεργασίας

a\	Προσθήκη κειμένου μετά την τρέχουσα γραμμή
c\	Αλλαγή του τρέχοντος κειμένου σε «νέο κείμενο»
d	Διαγραφή κειμένου
i\	Προσθήκη κειμένου πριν την τρέχουσα γραμμή
p	Εκτύπωση κειμένου
r	Ανάγνωση αρχείου
s	Αναζήτηση και αντικατάσταση κειμένου
w	Εγγραφή σε αρχείο
-e	Για τον προσδιορισμό πολλαπλών εντολών
-f SCRIPT_FILE	Ορισμός αρχείου sed εντολών
-n	Εκτυπώνονται μόνο οι p εντολές

Σημείωση: Όπου βλέπετε «\» σημαίνει ότι γράφετε κάποιο string.

Αντικατάσταση

- Η πλέον συνήθης εντολή:

  `sed s/pattern/replacement/<flags>`

- **pattern:** πρότυπο που αναζητούμε

- **replacement:** το string που θα αντικαταστήσει το πρότυπο

- **flags** (προαιρετικά):
  - **n** (number): ποια εμφάνιση (π.χ., $1^{η}$, $2^{η}$, …) του προτύπου θα αντικατασταθεί
  - **g** (global): αντικατάσταση όλων (όχι μόνο της πρώτης εμφάνισης) των προτύπων που εμφανίζονται στην τρέχουσα γραμμή
  - **p** (print): εκτύπωση του περιεχομένου του pattern space

Αρχείο για παραδείγματα

```
# cat -n example.sed

1 This is the first line of an example text.
2 It is a text with erors.
3 Lots of erors.
4 So many erors, all these erors are making me sick.
5 This is a line not containing any errors.
6 This is the last line.
```

Παράδειγμα 1

```
# sed 's/erors/errors/g' example.sed

This is the first line of an example text.
It is a text with errors.
Lots of errors.
So many errors, all these errors are making me sick.
This is a line not containing any errors.
This is the last line.
```

- Τι θα συμβεί αν στην εντολή αντικαταστήσουμε το g με τον αριθμό 2;
- Τι αν το απομακρύνουμε εντελώς;

Παράδειγμα 2

- Δύο ειδικά σύμβολα:
  - ^ Αρχή γραμμής
  - $ Τέλος γραμμής

```
# sed 's/^/> /' example.sed

> This is the first line of an example text.
> It is a text with erors.
> Lots of erors.
> So many erors, all these erors are making me sick.
> This is a line not containing any errors.
> This is the last line.
```

- Τι θα συμβεί αν στην εντολή αντικαταστήσουμε το ^ με $;

Παράδειγμα 3

#sed -e 's/erors/errors/g' -e 's/last/final/g' example.sed

(ή, εναλλακτικά)

sed 's/erors/errors/g; s/last/final/g' example.sed

```
This is the first line of an example text.
It is a text with errors.
Lots of errors.
So many errors, all these errors are making me sick.
This is a line not containing any errors.
This is the final line.
```

Άλλοι ειδικοί χαρακτήρες

- **_** ή **,** ή οποιοσδήποτε άλλος χαρακτήρας μπορεί να πάρει τη θέση του **/** για αναγνωσιμότητα
- ****: escape character
- **&**: Αντιστοιχεί στο πρότυπο που έχει βρεθεί (αναφερόμαστε πάντα στην τρέχουσα γραμμή)
- Προσοχή στο τι αποτελεί σύμβολο regular expression και τι όχι

Παράδειγμα 4

- Εκτύπωση μόνο των γραμμών που υπακούουν στο πρότυπο μετά την αλλαγή τους κατά τους όρους της αντικατάστασης που ορίστηκε:

- **`# sed -n 's/erors/errors/gp' example.sed`**

```
It is a text with errors.
Lots of errors.
So many errors, all these errors are making me sick.
```

Παράδειγμα 5

- **[^abcd]** σημαίνει «οποιοσδήποτε χαρακτήρας πλην των a, b, c, d»

- **# sed 's/[^]*/(&)/1' example.sed**

```
(This) is the first line of an example text.
(It) is a text with erors.
(Lots) of erors.
(So) many erors, all these erors are making me sick.
(This) is a line not containing any errors.
(This) is the last line.
```

Εστίαση σε συγκεκριμένες γραμμές

- Μπορούμε να εστιάσουμε σε συγκεκριμένες γραμμές τις αλλαγές μας, δηλώνοντας τις γραμμές με τον αριθμό τους.

- **# sed '1,3 s/erors/errors/g' example.sed**

```
This is the first line of an example text.
It is a text with errors.
Lots of errors.
So many erors, all these erors are making me sick.
This is a line not containing any erors.
This is the last line.
```

Εστίαση σε συγκεκριμένες γραμμές

- Το ίδιο μπορούμε να κάνουμε δίνοντας ένα κοινό τους pattern

- **# sed '/line/s/li/re/' example.sed**

```
This was the first rene of an example text.
It is a text with errors.
Lots of errors.
So many errors, all these erors are making me sick.
This was a rene not containing any errors.
This was the last rene.
```

Διαγραφή

d	όλες τις γραμμές
6d	τη γραμμή 6
`/^$/d`	όλες τις κενές γραμμές
`/^\./d`	όλες τις γραμμές που ξεκινούν με .
`1,10d`	τις γραμμές 1 -10
`1,/^$/d`	από τη 1η γραμμή μέχρι και την πρώτη κενή γραμμή
`/^$/,$d`	από την πρώτη κενή γραμμή μέχρι και τη τελευταία γραμμή

sed -- απλά Παραδείγματα

- Αντικατάστησε το "foo" με το "bar" μόνο στις γραμμές που περιέχουν το "baz"
  - `sed '/baz/s/foo/bar/g'`
- Σβήσε τα κενά από την αρχή και από το τέλος κάθε γραμμής
  - `sed 's/^[ \t]*//;s/[ \t]*$//'`
- Πρόσθεσε 5 κενά στην αρχή κάθε γραμμής
  - `sed 's/^/     /'`
- Σβήσε όλες τις κενές γραμμές από ένα αρχείο (όπως η "grep '.' ")
  - `sed '/^$/d'`

awk (Aho, Weinberger, Kernighan)

- Mini γλώσσα προγραμματισμού σχεδιασμένη να βρίσκει και να ταιριάζει πρότυπα και να εκτελεί εντολές σε αυτά → Perl
- Μπορεί να ενεργήσει επί των δεδομένων εισόδου της με σύνθετες, αναδρομικές συναρτήσεις (όπως στη C)
- Επεξεργάζεται αρχεία κειμένου **ανά πεδίο**
  - Ενώ το sed δούλευε **ανά γραμμή**
- Τη χρησιμοποιούμε περισσότερο όταν η γνώση για την πληροφορία βρίσκεται στη δομή του αρχείου (γραμμές, στήλες)
- Για πιο σύνθετες πράξεις, όπως αναδρομική εργασία σε συμβολοσειρές ή η μορφοποίηση εξόδου
- Όταν είναι απαραίτητες ευκολίες μιας υψηλού επιπέδου C-like γλώσσας, όπως πίνακες και εντολές ελέγχου ροής

Χαρακτηριστικά

- Σε σύνταξη μοιάζει με τη C
- Χειρίζεται την είσοδο, τον διαχωρισμό πεδίων και τη διαχείριση μνήμης αυτόματα
- Περιέχει
  - τύπους δεδομένων για συμβολοσειρές και αριθμούς
  - συναρτήσεις αρίθμησης και συμβολοσειρών
- Όχι δηλώσεις τύπων μεταβλητών
- Μεταβλητές και ροή ελέγχου στις ενέργειες
- Βολικός τρόπος πρόσβασης πεδίων μέσα σε γραμμές
- Ευέλικτη εκτύπωση

Δομή

- Ένα πρόγραμμα awk αποτελείται από:
  - Ένα προαιρετικό βήμα που εκτελείται πριν το διάβασμα εισόδου BEGIN
  - Ένα ζευγάρι πρότυπο – ενέργεια (pattern -- action)
    - επεξεργάζεται τα δεδομένα εισόδου
    - εφαρμόζει την εντολή action για κάθε matching
  - Ένα προαιρετικό βήμα που εκτελείται μετά το διάβασμα εισόδου END

```
BEGIN {action}
pattern {action}
pattern {action}
   .
   .
   .
pattern { action}
END {action}
```

Πεδία και τελεστές

- Ένα πρόγραμμα awk μπορεί να είναι ιδιαίτερα πολύπλοκο.
- Για να τρέξουμε ένα awk πρόγραμμα μπορούμε να δώσουμε ένα σύντομο program και input files (προαιρετικά, αφού τα δεδομένα μπορεί να έρχονται από pipe ή redirection) ως ορίσματα γραμμής εντολής
  - awk ’program’ [input_file(s)]
- Ή στη θέση του ’program’ να δώσουμε ένα αρχείο με τις εντολές που αυτό θα περιέχει:
  - awk -f program_file [input_file(s)]

Δομή προτύπων - ενεργειών

- Κάθε εντολή προγράμματος πρέπει να έχει ένα πρότυπο ή μια ενέργεια ή και τα δυο
  - Εξ' ορισμού (προκαθορισμένο) πρότυπο είναι να ταιριάξει με όλες τις γραμμές
  - Εξ' ορισμού (προκαθορισμένη) ενέργεια είναι να εκτυπώσει την υφιστάμενη καταγραφή
- Τα πρότυπα απλά παρατίθενται, ενώ οι ενέργειες εσωκλείονται σε **{ }**
- Η awk ανιχνεύει μια ακολουθία γραμμών, μια-μια, ψάχνοντας για γραμμές που ταιριάζουν με το πρότυπο

Πρότυπα

- Προσδιορισμός κατά πόσον μια ενέργεια-δράση πρόκειται να εκτελεστεί

- Μπορεί να είναι:
  - Το ειδικό *token* **BEGIN** ή **END**
  - Κανονική έκφραση (εσωκλειόμενη μέσα σε / /)
  - Σχεσιακές ή *string* αντιστοιχίες εκφράσεων – > < >= <= ==
  - Το σύμβολο **!** δίνει το αντίθετο της αντιστοιχίας
  - Συνδυασμός των πιο πάνω χρησιμοποιώντας λογικούς τελεστές && ή ||
    - π.χ. /NYU/ && (name == “UNIX Tools”)

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

Ενέργειες-δράσεις

- Περιλαμβάνει λίστα από εκφράσεις όπως στη *C*
  - *++* Increment, -- Decrement
  - ^ Exponentiation
  - *+-!* Πρόσθεση, Αφαίρεση , Άρνηση
  - ** / %* Πολλαπλασιασμός, διαίρεση, modulo

- Εκτελείται για κάθε γραμμή που ταιριάζει το πρότυπο
  - Εάν δεν δοθεί πρότυπο, η δράση εκτελείται για κάθε γραμμή εισόδου
  - Εάν δεν δοθεί δράση, όλες οι γραμμές που ταιριάζουν με το πρότυπο στέλνονται στο stdout

Εγγραφές (records)

- Κάθε γραμμή της εισόδου είναι ένα *record*
  - Το υφιστάμενο *record* μπορεί να παραπεμφθεί με το σύμβολο *$0*
- Πεδίο – Field
  - Κάθε λέξη σε ένα *record* ονομάζεται πεδίο *(field)*
  - Κάθε πεδίο αριθμείται και παραπέμπεται ως εξής: – *$1* είναι η πρώτη λέξη, *$2* είναι η δεύτερη λέξη κλπ

Ειδικές προκαθορισμένες μεταβλητές

- *FS*
  - Ο χαρακτήρας που δρα ως ο οριοθέτης του πεδίου (*field separator*)
  - Εξ' ορισμού είναι το *white space (*κενό διάστημα*, tab* κλπ*)*
  - Μπορεί να ξανα-ορισθεί με την –F επιλογή
    - **» awk –Fc** : ρυθμίζει το *FS* με τον χαρακτήρα *c*
    - Μπορεί να αλλάξει και στην *BEGIN* ενέργεια
- *RS*
  - Ο χαρακτήρας που δρα ως ο οριοθέτης του *record*
  - Εξ' ορισμού είναι το τέλος της γραμμής (*newline*)
  - Μπορεί να αλλάξει στην *BEGIN* ενέργεια
- *NF*
  - Αριθμός πεδίων στο υφιστάμενο *record*
- *NR*
  - Συνολικός αριθμός *records* που έχουν διαβαστεί μέχρι στιγμής
- *OFS*
  - *Output Field Separator*
  - Εξ' ορισμού είναι το μονό κενό διάστημα
- *ORS*
  - *Output Record Separator*

Παράδειγμα 1

- Εκτύπωσε όλη την είσοδο τυπώνοντας πριν ένα START και μετά ένα STOP
  - awk ’BEGIN {print "START"}
  - {print $0}
  - END {print "STOP"}’
  - foo.txt

- Εκτύπωσέ μου τα ονόματα των αρχείων στον τρέχοντα κατάλογο και τους χρήστες τους

```
ls -l | awk ’
BEGIN { print "File\tOwner" }
      { print $8, "\t", $3}
END   { print "done"}
’
```

Προσέξτε τα μονά εισαγωγικά (a.k.a. αυτάκια)! Αν τα ξεχάσετε, το πρόγραμμά σας δεν θα τρέξει!

Παράδειγμα 2

- bash-3.1# cat example.awk

Line number 1

Line number 2

Line number 3

Line number 4

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

Hello you

Line number 2

goodbye

Παράδειγμα 3

- 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

```
Just a text file. Nothing to see here.
Some lines have
more fields than others
and some
                              ( <- blank line)
are blank.

bash-3.1# awk 'NF>3 ? /file/ : /^and/ {print $0}'
example2.awk
Just a text file. Nothing to see here.
and some
```

Παράδειγμα 4

- Εκτύπωσε όλα τα αρχεία μη μηδενικού μεγέθους με κατάληξη .txt

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

List of all .txt files
line number:9 file file+1.txt of size: 10
line number:12 file output.txt of size: 4898
line number:13 file processes.txt of size: 12953
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
OK!!
```

awk -- απλά Παραδείγματα (1)

- Τύπωσε τις δύο πρώτες στήλες (default FS=" ") με αντίστροφη σειρά
  - awk '{ print $2, $1 }' file
- Τύπωσε τη στήλη 3 αν η στήλη 1 είναι μεγαλύτερη από τη στήλη 2
  - awk '$1 > $2 {print $3}' file
- Τύπωσε τη γραμμή εμφάνισης, τον αριθμό των στηλών και τη στήλη 1 κάθε γραμμής
  - awk '{print NR, NF, $1}' file
- Εκτύπωσε το 2ο πεδίο σε ένα αρχείο όπου τα πεδία είναι χωρισμένα με ?
  - awk –F "?" '{print $2}' file
- Τύπωσε το αρχείο file αφού διαγράψεις τη 2η στήλη του
  - awk '{$2 = ""; print}' file

Μεταβλητές

- Η *awk* μπορεί να ορίσει και να χρησιμοποιήσει μεταβλητές
  - BEGIN { sum = 0 } { sum ++ } END { print sum }

- Οι μεταβλητές αυτές μπορούν να πάρουν αριθμητική (ακέραια ή πραγματική) τιμή ή συμβολοσειρά
  - ΔΕΝ ΔΗΛΩΝΟΝΤΑΙ
  - Εξ' ορισμού, οι μεταβλητές που ορίζουμε αρχικοποιούνται με την αριθμητική τιμή 0 (->*null* string)

awk -- απλά Παραδείγματα (2)

- Τύπωσε όλες τις γραμμές που διαφέρουν ανά δύο στην πρώτη στήλη
  - awk ’$1 != prev { print; prev = $1 }’ file
- Τύπωσε μόνο την τελευταία γραμμή
  - awk ’{line = $0} END {print line}’

awk - μεταβλητές, ροή ελέγχου

Λειτουργίες με μεταβλητές:

- Βρες το άθροισμα και τον μέσο όρο των αριθμών της 1ης στήλης
  - awk ’{ s += $1 } END {print "sum is", s, " average is", s/NR}’

Ροή ελέγχου: for loop

- Τύπωσε όλες τις στήλες με αντίθετη σειρά
  - awk ’{ for (i = NF; i > 0; --i) print $i }’ file

Ροή ελέγχου: if-else

- Τύπωσε όλες τις μη κενές γραμμές των οποίων οι δυο πρώτες στήλες είναι ίσες
  - awk ’NF > 0 { if($1 == $2) print $0}’ file

Μορφοποίηση εξόδου και συνδυασμοί

- **Μορφοποίηση με printf**
- Τύπωσε και στοίχισε όλα τα αρχεία < 1000 bytes και τα μεγέθη τους
  - ls -l | awk ’{ if ($5<1000) printf("File: %10s - Size %5d\n", $9, $5)}’

- **Συνδυαστικό παράδειγμα**
- Πρόσθεσε τους αριθμούς από τις πρώτες 10 γραμμές του file

```
awk ’
{
  while(counter <= 10)
  {
    for(i=1; i<=NF; i++) {sum += $i;}
    counter++;
  }
}
END {printf("sum of first 10 lines: %3d\n", sum)}’ file
```