Εργαστήριο 3: Εργαλεία Συστήματος UNIX

Στο εργαστήριο θα μελετηθούν:

- *Ομάδες για παρουσίαση*
- awk (Alfred Aho, Peter Weinberger, and Brian Kernighan)
- *Παραδείγματα χρήσης awk*

Διδάσκων: Γιώργος Χατζηπολλάς

Ομάδες Παρουσίασης

Group Name	Description	Members
Group 1	Διαχείριση Συστήματος Υποδομής Πλέγματος EGEE	Christodoulos Efstathiades Doros Georgiou
Group 2	Ruby	Marios Hadjipollas Katerina Kounouni
Group 3	Windows: Sockets	Marinos Argyrou Aristodimos Pipis
Group 4	Windows: Threads and Concurrency	Orestis Spanos Christos Kyriakou
Group 5	Windows: Process Management and IPC	Panagiotis Petsas ????

awk

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

- Η *awk* επεξεργάζεται *πεδία* ενώ η *sed* επεξεργάζεται μόνο γραμμές

awk - χαρακτηριστικά

- Η γλώσσα μοιάζει λίγο με τη *C* αλλά αυτόματα χειρίζεται την είσοδο, το διαχωρισμό πεδίων, και τη διαχείριση μνήμης
  - Ενσωματωμένοι τύποι δεδομένων για συμβολοσειρές και αριθμούς
  - Όχι δηλώσεις τύπων μεταβλητών
  - Κατάλληλη αριθμητική επεξεργασία
  - Μεταβλητές και ροή ελέγχου στις ενέργειες
  - Βολικός τρόπος πρόσβασης πεδίων μέσα σε γραμμές
  - Εύκαμπτη εκτύπωση
  - Ενσωματωμένες συναρτήσεις αρίθμησης και συμβολοσειρών
  - Σύνταξη γλώσσας μοιάζει της *C*

awk – σύνταξη

- Ένα awk πρόγραμμα αποτελείται από:
  - ένα προαιρετικό τμήμα *BEGIN*
    - εκτελείται πριν το διάβασμα εισόδου
  - Ζευγάρι *προτύπου – ενέργειας* (*pattern - action*)
    - Επεξεργασία δεδομένων εισόδου
    - Για κάθε πρότυπο που ταιριάζει, η αντίστοιχη ενέργεια εκτελείται
  - ένα προαιρετικό τμήμα *END*
    - εκτελείται μετά το τέλος εισόδου δεδομένων

BEGIN {action}

pattern {action}

pattern {action}

.

.

.

pattern { action}

END {action}

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

- Πρότυπα και Ενέργειες
  - Ψάχνει ένα σύνολο αρχείων για *πρότυπα*
  - Εκτελεί τις ενέργειες στις γραμμές ή πεδία που περιέχουν τις περιπτώσεις των προτύπων.
  - Δεν αλλάζει τα αρχεία εισόδου
  - Επεξεργάζεται μια γραμμή εισόδου κάθε φορά

awk

- Δομή Προτύπων και 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 μέσα σε **{ }**
  - Η awk ανιχνεύει μια ακολουθία γραμμών, μια-μια, ψάχνοντας για γραμμές που ταιριάζουν με το πρότυπο

awk - πρότυπα

- Πρότυπα
  - Προσδιορισμός κατά πόσον μια *ενέργεια-δράση* πρόκειται να εκτελεστεί
  - Μπορεί να είναι:
    - Το ειδικό *token* **BEGIN** ή **END**
    - Κανονική έκφραση (εσωκλειόμενο μέσα σε / /), όπως *egrep*
    - 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*

awk - ενέργειες

- Ενέργειες – Δράσεις
  - Περιλαμβάνει λίστα από εκφράσεις όπως στη *C*
    - ++, -- Increment, Decrement
    - ^ Exponentiation
    - +, -, ! Plus, Minus, NOT
    - *, /, % Multiplication, division, modulus
  - Εκτελείται για κάθε γραμμή που ταιριάζει το *πρότυπο*
    - Εάν δε δοθεί *πρότυπο*, η δράση εκτελείται για κάθε γραμμή εισόδου
    - Εάν δε δοθεί δράση, όλες οι γραμμές που ταιριάζουν με το πρότυπο στέλλονται στο ρεύμα εξόδου

awk – εγγραφές

- Εγγραφή – Record
  - 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

awk – προκαθορισμένες μεταβλητές

- Ειδικές προκαθορισμένες μεταβλητές *awk*
  - *RS*
    - Ο χαρακτήρας που δρα ως ο οριοθέτης του *record*
      - Εξ' ορισμού είναι το τέλος της γραμμής (*newline*)
      - Μπορεί να αλλάξει στην ενέργεια *BEGIN*
  - *FS*
    - Ο χαρακτήρας που δρα ως ο οριοθέτης του πεδίου (*field*)
      - Εξ' ορισμού είναι το *white space (κενό διάστημα, tab κλπ)*
      - Μπορεί να ξανα-ορισθεί με την –F επιλογή
        - » awk –Fc επιλογή ρυθμίζει το *FS* με το χαρακτήρα *c*
      - Μπορεί να αλλάξει και στην ενέργεια *BEGIN*
  - *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*

awk - Παραδείγματα

```
bash-3.1$ cat example.awk
Line number 1
Line number 2
Line number 3
Line number 4

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

bash-3.1$ awk '{print $3}' example.awk
1
2
3
4
```

awk - Παραδείγματα

```
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
```

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.

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

Εκτύπωσε τις γραμμές των οποίων τα πεδία είναι > 3

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

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

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.

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

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

awk - Παραδείγματα

- Παραδείγματα

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

```
bash-3.1$ ls -l | awk 'BEGIN {print "List of all .txt
   files"} /\.txt$/ {print $9, $5} END {print "There you
   go!!"}'
List of all .txt files
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!!
```

awk - Παραδείγματα

- Παραδείγματα
  - {print NF, $1, $NF}
    - τυπώνει τον αριθμό των πεδίων, το πρώτο πεδίο, και το τελευταίο πεδίο στο υφιστάμενο record
  - {print $(NF-2)}
    - τυπώνει το τρίτο από το τέλος πεδίο
  - {print $1, $2*$3}
    - τυπώνει το πρώτο πεδίο και το αποτέλεσμα του υπολογισμού – πολλαπλασιασμός του δεύτερου και τρίτου πεδίου
  - {print NR, $0}
    - τυπώνει κάθε γραμμή τοποθετώντας στην αρχή τον αριθμό της υφιστάμενης γραμμής
  - {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 “”

awk

- Παραδείγματα

No comma → no empty space at output

```
bash-3.1$ ls -l | awk 'BEGIN {print "List of all .txt
  files"} /\.txt$/ {print "line number:" NR, "file",
  $9, "of size:", $5} END {print "There you go!!"}'
List of all .txt files
line number:9 file file+1.txt of size: 0
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
There you go!!
```

awk

- Καλύτερη Διαμόρφωση Εξόδου
  - Χρήση της *printf()* όπως στη *C*
    - *printf(format, val1, val2, val3, …*)

```
{ 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 ) }
```

awk

- Επιλογή
  - Τα *awk* πρότυπα μπορούν να επιλέγουν συγκεκριμένες γραμμές από την είσοδο για περαιτέρω επεξεργασία
  - Επιλογή βάση Σύγκρισης
    - **`$2 >= 5 { print }`**
  - Επιλογή βάση Υπολογισμού
    - **`$2 * $3 > 50 { printf(“%6.2f for %s\n”, $2 * $3, $1) }`**
  - Επιλογή βάση Περιεχομένου Κειμένου
    - **`$1 == "NYU"`**
    - **`$2 ~ /NYU/`**
  - Συνδυασμοί προτύπων
    - **`$2 >= 4 || $3 >= 20`**
  - Επιλογή βάση αριθμού γραμμής
    - **`$2 >= 4 || $3 >= 20`**

awk

- Παραδείγματα

```
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 "There you
   go!!"}'
List of all .txt files
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
There you go!!
```

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!!
```

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!!
```

```
bash-3.1$ ls -l | awk 'BEGIN {print "List of all .txt
ή    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
There you go!!
```

Χρήση μεταβλητών στην awk Παραδείγματα

- Η awk μπορεί να ορίσει και να χρησιμοποιήσει μεταβλητές
  - Οι μεταβλητές αυτές μπορούν να πάρουν αριθμητική (ακέραια ή πραγματική) τιμή ή συμβολοσειρά
    - ΔΕ ΔΗΛΩΝΟΝΤΑΙ
    - Εξ' ορισμού, οι μεταβλητές που ορίζουμε αρχικοποιούνται με την αριθμητική τιμή 0 (→ *null* string)

- Παραδείγματα

```
BEGIN { sum = 0 }
{ sum ++ }
END { print sum }
```

Μετρά τον αριθμό των γραμμών της εισόδου

Χρήση μεταβλητής NR

```
awk 'END{print NR}'
```

Χρήση μεταβλητών στην awk Παραδείγματα

- Έστω ότι έχουμε ένα αρχείο με 3 στήλες
- Ταυτότητα Μισθός ανά ώρα Ώρες Εργασίας

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

Τυπώνει τον αριθμό των υπαλλήλων που δούλεψαν περισσότερες από 15 ώρες

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

Τυπώνει τον αριθμό των υπαλλήλων, το συνολικό κόστος πληρωμής και το μέσο μισθό

awk - Παραδείγματα

- Διαχείριση συμβολοσειρών
  - Παράδειγμα
    - Το πρόγραμμα αυτό βρίσκει τον εργοδοτούμενο που έχει πληρωθεί το μέγιστο ποσό ανά ώρα:

συμβολοσειρά

αριθμητική τιμή

```
# Fields: employee, payrate
$2 > maxrate { maxrate = $2; maxemp = $1 }
END { print “highest hourly rate:”, maxrate, “for”, maxemp }
```

awk

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

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

  - Παράδειγμα

multiple statements in a row

```
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

awk

- Εντολές ελέγχου ροής

```
if (condition)
{
  Statements
}
else
{
   Statements
}
```

```
while (Condition){
   Statements
}
```

```
do {
       Statements
     }
while (condition)
```

```
for (Declaration ; Condition ; Increment )
{
   Statements
}
```

Χρήση μεταβλητών στην awk Παραδείγματα Αρίθμησης και Υπολογισμού

- Τυπώστε το συνολικό μέγεθος των αρχείων σε Mb που βρίσκονται στον τρέχον φάκελο και έχουν αλλαχτεί κατά το μήνα Νοέμβριο.

```
ls -lg | awk '$5 == "Nov" { sum += $6 } END { print sum “Mb” }‘
```

- Τυπώστε όλες τις γραμμές για τις οποίες το πρώτο πεδίο είναι διαφορετικό από την προηγούμενη

```
awk '$1 != prev { print; prev = $1 }' file
```

- Τυπώστε και ταξινομίστε με αντίστροφη αλφαβητική σειρά όλα τα login names (πρώτη στήλη) που βρίσκονται στο αρχείο /etc/passwd

```
awk -F ":" '{ print $1 }' /etc/passwd | sort -r
awk -F ":" '{ print $1 | "sort -r" }' /etc/passwd
```

Χρήση μεταβλητών στην awk Παραδείγματα Χρήσης Βρόγχων

- Γράψετε ένα awk script το οποίο παίρνει ένα αρχείο με τρείς στήλες, προσθέτει τις στήλες κάθε γραμμής και υπολογίζει το άθροισμά τους. Το αποτέλεσμα πρέπει να φαίνεται ως μαθηματικές πράξεις

```
BEGIN {print "Print Totals"}
    {total = $1 + $2 + $3}
    {print $1 " + " $2 " + " $3 " = "total}
END {print "End Totals"}
```

- Τυπώστε κάθε πεδίο κάθε γραμμής του αρχείου *filename* σε ξεχωριστή γραμμή

```
awk '{for(i=1;i<=NF;i++) print $i }' filename
```

- Αντίστρεψε τις γραμμές του αρχείου *filename*

```
awk '{a[i++]=$0} END {for (j=i-1; j>=0;) print a[j--] }' filename
```