Λειτουργικά Συστήματα Ι

Ασκήσεις Πράξης

Τμήμα Μηχανικών Πληροφορικής & Υπολογιστών

Σχολή Μηχανικών

Φίλτρα

Λίστα εντολών

Λίστα εντολών είναι μια σειρά από δύο ή περισσότερες σωληνώσεις χωρισμένες μεταξύ τους με τα σύμβολα **;** ή **||** ή **&&**. Ο τελεστής ; εκτελεί διαδοχικά τις εντολές της λίστας σε αντίθεση με τους τελεστές ελέγχου διάζευξης || και σύζευξης && που τις εκτελούν υπό συνθήκη. Η παράθεση εντολών έχει τη μορφή:

command1 ; command2

και η δεύτερη εντολή εκτελείται πάντα μετά την πρώτη. Μία λίστα εντολών με διάζευξη έχει τη μορφή:

command1 || command2

και η εντολή command2 εκτελείται μόνο αν η εντολή command1 επιστέψει μη μηδενική τιμή εξόδου.

Λίστα εντολών

Μία λίστα εντολών με σύζευξη έχει μορφή:

command1 && command2

και η εντολή command2 εκτελείται μόνο αν η εντολή command1 έχει τιμή εξόδου μηδέν. Πιο συγκεκριμένα, η δεύτερη εντολή εκτελείται μόνο αν η πρώτη εκτελέσθηκε με επιτυχία.

Η τιμή εξόδου μιας λίστας εντολών είναι η τιμή εξόδου της τελευταίας εντολής που εκτελέσθηκε.

Λίστα εντολών

```
$ date; echo -n "My kernel is "; uname -r
Sun Dec 24 03:55:06 PM CET 2023
My kernel is 6.2.0-39-generic
```

Η επόμενη εντολή ταξινομεί ένα αρχείο κειμένου και μόνο αν δημιουργηθεί το ταξινομημένο αρχείο θα διαγράψει το πηγαίο αρχείο:

```
$ sort file.txt > sorted.txt && rm file.txt
```

Η εντολή mkdir που ακολουθεί θα δημιουργήσει έναν κατάλογο μόνο στην περίπτωση που αποτύχει η εκτέλεση της εντολής ls, όταν δηλαδή δεν υπάρχει ο κατάλογος *mydir*:

```
$ ls mydir 2> /dev/null || mkdir mydir
```

Φίλτρα

- Ο όρος φίλτρο (filter) αναφέρεται σε ένα απλό και εξειδικευμένο πρόγραμμα, το οποίο τροποποιεί ή μετασχηματίζει τα στοιχεία εισόδου του. Αν και τα προγράμματα αυτά μπορούν να χρησιμοποιηθούν αυτόνομα, συνήθως αποτελούν μέλος μιας σωλήνωσης, η οποία παράγει ένα εξειδικευμένο αποτέλεσμα.
- Παραδείγματα φίλτρων είναι οι εντολές **head** και **tail** που επιλέγουν γραμμές εισόδου και η εντολή **wc** που μετράει χαρακτήρες, λέξεις και γραμμές.

Σωληνώσεις

Ένα από τα βασικά στοιχεία της φιλοσοφίας του UNIX είναι η δημιουργία ισχυρών εργαλείων τα οποία βασίζονται σε περισσότερες απλούστερες εντολές. Η διαδικασία αυτή επιτυγχάνεται με τη σύνδεση μεταξύ δύο διεργασιών, όπου τα στοιχεία εξόδου της πρώτης δίνονται ως στοιχεία εισόδου στη δεύτερη. Με αυτόν τον τρόπο είναι δυνατή η διαδοχική επεξεργασία ενός αρχείου εισόδου από πολλά προγράμματα. Η λειτουργία αυτή ονομάζεται σωλήνωση ή διοχέτευση (pipeline) και η σύνδεση των διεργασιών γίνεται με τη χρήση του τελεστή | (κάθετος).

Σωληνώσεις

Η ακολουθία εντολών **command1 | command2** στέλνει την πρότυπη έξοδο της εντολής command1 στην πρότυπη είσοδο της εντολής command2.

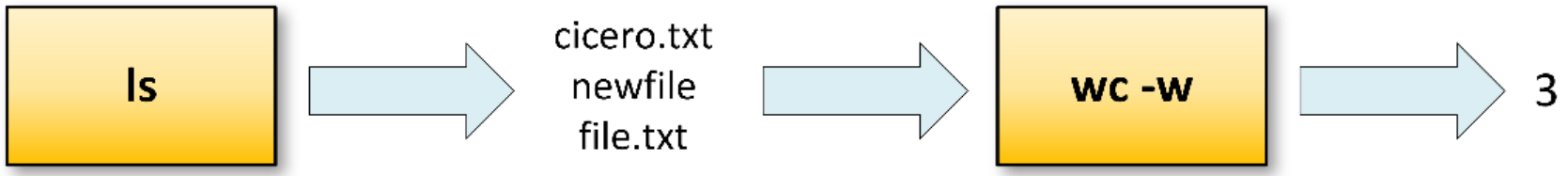
Πιο συγκεκριμένα, η πρώτη διεργασία θα δημιουργήσει την πληροφορία, η οποία μέσα από τη σωλήνωση θα δοθεί ως είσοδος στη δεύτερη διεργασία. Η διαδικασία αυτή μπορεί να συνεχίζεται με οποιοδήποτε αριθμό διεργασιών, αρκεί η εντολή που βρίσκεται στα αριστερά της σωλήνωσης να έχει πρότυπη έξοδο την οθόνη και η εντολή που βρίσκεται δεξιά της σωλήνωσης να έχει ως πρότυπη είσοδο το πληκτρολόγιο.

Σωληνώσεις

Για παράδειγμα μπορούμε να χρησιμοποιήσουμε μια σωλήνωση για να υπολογίσουμε το πλήθος των αρχείων του τρέχοντος καταλόγου:

```
$ ls | wc -w                    # count files
3
```

Η παραπάνω λειτουργία μπορεί να αναπαρασταθεί με την επόμενη εικόνα:



sort

- Η εντολή **sort** ταξινομεί γραμμές κειμένου.
- Η σύνταξή της είναι:

  sort [OPTION]… [FILE]…
- Χωρίς τη χρήση επιλογών και αρχείων η εντολή sort διαβάζει δεδομένα από το πληκτρολόγιο και με το πέρας της εισόδου (μόλις δηλαδή πατήσουμε τον συνδυασμό πλήκτρων [Ctrl]+[D]), εμφανίζει τις γραμμές ταξινομημένες αλφαβητικά κατ’ αύξουσα σειρά στην οθόνη.

sort

Οι πιο σημαντικές επιλογές της sort είναι:

- **-b** αγνόηση αρχικών κενών (ignore **b**lanks)
- **-c** έλεγχος αν ένα αρχείο είναι ταξινομημένο (**c**heck)
- **-d** λαμβάνονται υπόψη μόνο κενά, γράμματα και ψηφία (**d**ictionary order)
- **-f** δε κάνει διάκριση μεταξύ πεζών και κεφαλαίων (**f**old lowercase to upper)
- **-i** εξετάζονται μόνο εκτυπώσιμοι χαρακτήρες (**i**gnore non printing)
- **-k** ταξινόμηση σύμφωνα με κλειδί (**k**ey)

sort

- **-m** συγχώνευση ήδη ταξινομημένων αρχείων (**m**erge)
- **-M** σύγκριση κατά μήνα (**M**onth)
- **-n** αριθμητική ταξινόμηση (**n**umerically order)
- **-o FILE** το αποτέλεσμα της ταξινόμησης στέλνεται στο αρχείο (**o**utput)
- **-r** φθίνουσα σειρά ταξινόμησης (**r**everse)
- **-R** τυχαία σειρά (**R**andom)
- **-t SEP** καθορισμός του SEP ως διαχωριστή πεδίων (**t**ransition)

sort

```
$ ls | sort  # sort contents of current directory
cicero.txt
...
newfile
$ sort -r file.txt                  # sort reverse
Wolfgang Hartmann 189 1958 2521091466
Steffen Reichert 188 1963 2310620789
Peter Frampton 188 1962 2522041313
Anna Frampton 170 1963 2522041313
Alice Bauhaus 167 1965 2106399547
```

sort

```
$ sort file.txt > sortedfile.txt
$ sort -c sortedfile.txt      # no output, sorted
$ sort -c file.txt    # report first unsorted data
sort file.txt:2: disorder: Anna Frampton 170 1963
$ cat > months          # create file months
Feb
Mar
Jan
$ sort -M months      # 'JAN' < ... < 'DEC'
Jan
Feb
Mar
```

sort

- Υπό κανονικές συνθήκες, η sort ταξινομεί με βάση ολόκληρη τη γραμμή. Συγκρίνει τον πρώτο χαρακτήρα κάθε γραμμής και αν βρεθεί ταύτιση συγκρίνει τους δεύτερους χαρακτήρες κ.ο.κ..
- Με την επιλογή **-k** έχουμε τη δυνατότητα να κάνουμε σύγκριση σύμφωνα με ένα ή και περισσότερα πεδία, θεωρώντας ως διαχωριστή πεδίων το κενό ή το στηλοθέτη.

sort

Αντίστροφη αριθμητική ταξινόμηση στο 3ο πεδίο:

```
$ sort -nr -k 3 file.txt
Wolfgang Hartmann 189 1958 2521091466
Steffen Reichert 188 1963 2310620789
Peter Frampton 188 1962 2522041313
Anna Frampton 170 1963 2522041313
Alice Bauhaus 167 1965 2106399547
```

Η πλήρης σύνταξη της επιλογής είναι **-k F.C,F.C** όπου το **F** αντιπροσωπεύει το πεδίο και το **C** αντιπροσωπεύει τη θέση του χαρακτήρα μέσα στο αντίστοιχο πεδίο. Αν παραλείπεται η θέση, η ταξινόμηση γίνεται με το πλήρες μήκος του πεδίου.

sort

Ταξινόμηση με τους πρώτους 4 χαρακτήρες του 5ου πεδίου:

```
$ sort -t" " -k 5.1,5.4 file.txt
Alice Bauhaus 167 1965 2106399547
Steffen Reichert 188 1963 2310620789
Wolfgang Hartmann 189 1958 2521091466
Anna Frampton 170 1963 2522041313
Peter Frampton 188 1962 2522041313
```

Τέλος, το αποτέλεσμα της ταξινόμησης ενός αρχείου μπορεί να καταχωρηθεί στο ίδιο αρχείο με τη χρήση της επιλογής **-o**:

```
$ sort file.txt -o file.txt  # output to same file
```

grep

- Η βασική λειτουργία της εντολής **grep** (**g**lobal search with the **r**egular **e**xpression and **p**rinting) είναι να αναζητά ένα πρότυπο σε αρχεία κειμένου και να εμφανίζει τις γραμμές που το περιέχουν. Το όνομα της εντολής προέρχεται από την εντολή **g/re/p** του συντάκτη κειμένου **ed** που έχει παρόμοια λειτουργία.
- Η σύνταξή της έχει ως εξής:

  grep [OPTION]… PATTERN [FILE]…
- Αν δεν δοθεί κάποιο αρχείο ως όρισμα, η εντολή διαβάζει από την πρότυπη είσοδο, οπότε μπορεί να χρησιμοποιηθεί σε σωλήνωση κάτι που είναι και το σύνηθες.

grep

Οι πιο σημαντικές επιλογές της εντολής grep είναι:

- **-c**, **--count** εμφάνιση του πλήθους γραμμών που περιέχουν το πρότυπο
- -**f FILE**, **--file=FILE** χρήση των πρότυπων από το αρχείο FILE (ένα ανά γραμμή)
- **-i**, **--ignore-case** δε γίνεται διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων
- **-l**, **--files-with-matches** εμφανίζει μόνο τα ονόματα των αρχείων
- **-L**, **--files-without-match** εμφανίζει τα ονόματα των αρχείων στα οποία δεν βρέθηκε ταύτιση με το πρότυπο

grep

- **-n**, **--line-number** εμφάνιση αύξοντα αριθμού γραμμής
- **-r**, **--recursive** αναδρομική αναζήτηση
- **-s**, **--no-messages** δεν εμφανίζονται μηνύματα λαθών
- **-v, --invert-match** επιστρέφει τις γραμμές που δε ταιριάζουν με το πρότυπο
- **-w**, **--word-regexp** το πρότυπο ταιριάζει με ολόκληρες λέξεις
- **-x**, **--line-regexp** ταύτιση προτύπου με πλήρη γραμμή

grep

```
$ grep Frampton file.txt      # search pattern
Peter Frampton 188 1962 2522041313
Anna Frampton 170 1963 2522041313
$ grep Frampton *.txt         # prefix file name
file.txt:Peter Frampton 188 1962 2522041313
file.txt:Anna Frampton 170 1963 2522041313
sortedfile.txt:Anna Frampton 170 1963
2522041313
sortedfile.txt:Peter Frampton 188 1962
2522041313
```

grep

```
$ grep -i frampton file.txt      # ignore case
Peter Frampton 188 1962 2522041313
Anna Frampton 170 1963 2522041313
$ grep -l Frampton *.txt         # file names only
file.txt
sortedfile.txt
$ grep -L Frampton *.txt         # file names only
cicero.txt
months.txt
```

grep

```
$ grep -n Frampton file.txt    # line number
1:Peter Frampton 188 1962 2522041313
2:Anna Frampton 170 1963 2522041313
$ grep -v 252 file.txt                 # invert
Alice Bauhaus 167 1965 2106399547
Steffen Reichert 188 1963 2310620789
$ grep -c 2522 file.txt                # count
2
$ grep -x "Peter Frampton" file.txt  # whole line
```

grep - regular expressions

Ο πίνακας παρουσιάζει τους ειδικούς χαρακτήρες και τη σημασία τους στη δημιουργία κανονικών εκφράσεων:	
^	Το στοιχείο που έπεται βρίσκεται στην αρχή της γραμμής
$	Το στοιχείο που προηγείται βρίσκεται στο τέλος της γραμμής
.	Ταιριάζει με οποιονδήποτε χαρακτήρα
[ ]	Ταιριάζει με οποιονδήποτε από τους περιεχόμενους χαρακτήρες
?	Το στοιχείο που προηγείται είναι προαιρετικό και ταιριάζει το πολύ μία φορά
*	Το στοιχείο που προηγείται ταιριάζει μηδέν ή περισσότερες φορές
+	Το στοιχείο που προηγείται ταιριάζει μία ή περισσότερες φορές
{n}	Το στοιχείο που προηγείται ταιριάζει ακριβώς n φορές
{n,}	Το στοιχείο που προηγείται ταιριάζει n ή περισσότερες φορές
{,m}	Το στοιχείο που προηγείται ταιριάζει το πολύ m φορές
{n,m}	Το στοιχείο που προηγείται τουλάχιστον n φορές και όχι περισσότερες από m φορές
|	Τελεστής OR, ταυτίζει μια από τις δύο κανονικές εκφράσεις
\	Αναστέλλει τη σημασία του μεταχαρακτήρα που ακολουθεί

grep

Στα πρώτα δύο παραδείγματα γίνεται αναζήτηση όλων των γραμμών του αρχείου /usr/share/common-licenses/GPL-3 που αρχίζουν και τελειώνουν αντίστοιχα με το αλφαριθμητικό “free”:

```
$ grep ^free /usr/share/common-licenses/GPL-3
$ grep free$ /usr/share/common-licenses/GPL-3
```

Το επόμενο παράδειγμα μετράει τις κενές γραμμές του αρχείου:

```
$ grep -c ^$ /usr/share/common-licenses/GPL-3
```

grep

Αναζήτηση στο λεξικό του συστήματος για λέξεις που αρχίζουν με τον χαρακτήρα “t”, ακολουθούν δύο οποιοιδήποτε χαρακτήρες και μετά υπάρχει ένας ακόμη χαρακτήρας “t”:

```
$ grep ^t..t /usr/share/dict/words
tact
tactful
...
twitters
twitting
```

grep

Αν θέλουμε να γίνουμε πιο συγκεκριμένοι, μπορούμε αντί για τις τελείες να χρησιμοποιήσουμε τις αγκύλες. Έτσι ο δεύτερος χαρακτήρας του πρότυπου μπορεί να είναι το “a” ή το “o” και ο τρίτος χαρακτήρας το “s” ή το “r”:

```
$ grep ^t[ao][sr]t /usr/share/dict/words
taste
torte
...
tasting
tasty
```

grep

Για την αναζήτηση όλων των γραμμών που αρχίζουν με κεφαλαίο γράμμα μπορούμε να χρησιμοποιήσουμε τις αγκύλες και να ορίσουμε εύρος χαρακτήρων με την παύλα:

```
$ grep ^[A-Z] /usr/share/common-licenses/GPL-3
GNU General Public License for most of our
software; it applies
...
```

Το **[\s]** για λευκούς χαρακτήρες, **-P** για Perl-compatible regex

```
$ grep -P '^[\s]+[A-Z][a-z]+' /usr/share/common-
licenses/GPL-3
                    Version 3, 29 June 2007
```

grep

Αν ο πρώτος χαρακτήρας μέσα στις αγκύλες είναι ο χαρακτήρας **^**, τότε το ταίριασμα γίνεται μ' έναν οποιο-δήποτε χαρακτήρα εκτός αυτών που αναφέρονται στις αγκύλες:

```
$ grep -i [^d]isk /usr/share/common-licenses/GPL-3
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY OF THE
PROGRAM
```

grep

Το παράδειγμα που ακολουθεί επιλέγει τις γραμμές με λέξεις από το λεξικό του συστήματος που αρχίζουν με “gre” και τελειώνουν με “r” :

```
$ grep -w gre.*r /usr/share/dict/words
greacier
greater
...
```

Για την αναζήτηση όλων των γραμμών που τελειώνουν με τελεία, θα πρέπει να προστατέψουμε την τελεία προτάσσοντας την αριστερή πλάγια:

```
$ grep '\.$' /usr/share/common-licenses/GPL-3
```

grep

Η εντολή grep διαθέτει διάφορες παραλλαγές όπως την **fgrep** (fixed-character) για σταθερά πρότυπα αναζήτησης, χωρίς δηλαδή κανονικές εκφράσεις και την **egrep** (extended) για αναζητήσεις με εκτεταμένες κανονικές εκφράσεις. Οι δύο αυτές παραλλαγές προσομοιώνονται από την grep με τη χρήση των επιλογών **-F** και **-E** αντίστοιχα. Το παράδειγμα που ακολουθεί ψάχνει στο λεξικό του συστήματος για όλες τις λέξεις που περιέχουν 2 ή το πολύ 3 χαρακτήρες 'a':

```
$ grep -E 'a{2,3}' /usr/share/dict/words
Africaans
Baal
...
```

cut

Η εντολή **cut** εμφανίζει επιλεγμένα πεδία των γραμμών ενός αρχείου κειμένου στην έξοδο. Η σύνταξή της είναι:

cut OPTION… [FILE]…

Αν δεν δοθεί κάποιο αρχείο ως όρισμα, η εντολή διαβάζει από την πρότυπη είσοδο. Οι πιο σημαντικές επιλογές της εντολής cut είναι:

- **-c**, **--characters=LIST** επιλογή μόνο των συγκεκριμένων χαρακτήρων
- -**d**, **--delimiter=DELIM** χρήση του DELIM για διαχωριστή πεδίων αντί του TAB
- -**f**, **--fields=LIST** επιλογή μόνο των συγκεκριμένων πεδίων

cut

```
$ cut -d' ' -f2,4 file.txt # delimiter is space
Frampton 1962
Frampton 1963
Bauhaus 1965
Reichert 1963
Hartmann 1958
$ cut -c1-4 file.txt          # get columns 1 to 4
Pete
Anna
Alic
Stef
Wolf
```

tr

Η εντολή **tr** (translate) διαβάζει από την πρότυπη είσοδο και γράφει στην πρότυπη έξοδο αφού πρώτα αντικαταστήσει επιλεγμένους χαρακτήρες. Η σύνταξη της εντολής είναι:

tr [OPTION]… SET1 [SET2]

Η tr μπορεί επίσης να διαγράψει χαρακτήρες και να μετατρέψει επαναλαμβανόμενους χαρακτήρες σε έναν.

Οι πιο σημαντικές επιλογές της εντολής tr είναι:

- **-d**, **--delete** διαγραφή χαρακτήρων του SET1 και όχι μετάφραση
- **-s**, **--squeeze-repeats** αντικατάσταση κάθε συνεχόμενου επαναλαμβανόμενου χαρακτήρα από το SET1 που εμφανίζεται πολλές φορές

tr

Η επόμενη εντολή μετατρέπει τους πεζούς λατινικούς χαρακτήρες της εισόδου της στα αντίστοιχα κεφαλαία:

```
$ tr [a-z] [A-Z] < file.txt
PETER FRAMPTON 188 1962 252241313
ANNA FRAMPTON 170 1963 252241313
ALICE BAUHAUS 167 1965 2106399547
STEFFEN REICHERT 188 1963 2310620789
WOLFGANG HARTMANN 189 1958 2521091466
```

tr

Η εντολή που ακολουθεί μετατρέπει κάθε κενό χαρακτήρα της εισόδου της στον χαρακτήρα TAB:

```
$ echo 'Hannibal ante portas' | tr ' ' '\t'
Hannibal        ante    portas
```

Η επόμενη εντολή μετατρέπει επαναλαμβανόμενα κενά σε ένα:

```
$ echo 'Nemo   sine vitio   est' | tr -s ' '
Nemo sine vitio est
```

Με την επιλογή **-d** μπορούμε να διαγράψουμε επιλεγμένους χαρακτήρες:

```
$ echo "My fav club is FC Schalke 04" | tr -d [0-9]
My fav club is FC Schalke
```

tee

Η εντολή **tee** είναι η μοναδική εντολή που έχει δύο εξόδους. Πιο συγκεκριμένα, μας επιτρέπει να διοχετεύσουμε μια ροή δεδομένων στην πρότυπη έξοδο και σ' ένα αρχείο. Η σύνταξη της εντολής είναι:

tee [OPTION]… [FILE]…

Η πιο σημαντική επιλογή της εντολής είναι η append (**-a**) με την οποία γίνεται προσθήκη των δεδομένων στο αρχείο.

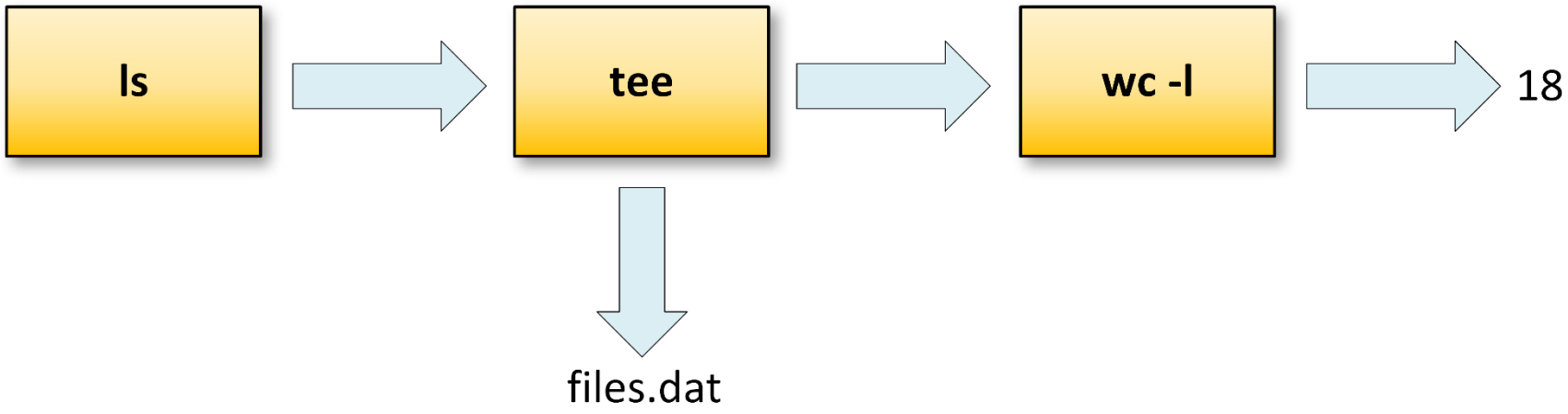
Η εντολή tee στο παράδειγμα που ακολουθεί, αποθηκεύσει την έξοδο της ls στο αρχείο files.dat και ταυτόχρονα τη διοχετεύει στην εντολή wc:

```
$ ls | tee files.dat | wc -l
18
```

tee

Η ίδια εντολή με τη χρήση της επιλογής **-a** θα προσθέσει άλλες 18 γραμμές στο αρχείο με συνέπεια να γίνουν 36 συνολικά:

```
$ ls | tee -a files.dat | wc -l   # append lines
18
```



uniq

Η εντολή **uniq** χρησιμοποιείται κατά τη δημιουργία αναφορών καθώς έχει τη δυνατότητα να εντοπίζει και να αναφέρει ή να παραλείπει επαναλαμβανόμενες γραμμές. Η εντολή κάνει σύγκριση συνεχόμενων γραμμών και για αυτό το λόγο συνδυάζεται συνήθως με την εντολή ταξινόμησης sort. Η σύνταξή της είναι:

uniq [OPTION]… [INPUT[OUTPUT]]

Οι πιο σημαντικές επιλογές της εντολής είναι:

- **-c** εμφανίζει τις γραμμές και τον αριθμό εμφάνισης κάθε μιας από αυτές
- **-d** εμφανίζει μόνο τις επαναλαμβανόμενες γραμμές, μια για κάθε ομάδα

uniq

- **-D** εμφανίζει όλες τις επαναλαμβανόμενες γραμμές
- **-f** N παραλείπει τα πρώτα N πεδία κατά τη σύγκριση
- **-i** δεν κάνει διάκριση μεταξύ πεζών και κεφαλαίων κατά τη σύγκριση
- **-s** N παραλείπει τους πρώτους N χαρακτήρες κατά τη σύγκριση
- **-u** εμφανίζει μόνο τις μοναδικές γραμμές
- **-w** N συγκρίνει τους πρώτους N χαρακτήρες κάθε γραμμής

uniq – αφαίρεση επαναλαμβανόμενων γραμμών

```
$ cat lines.txt
This is a line.
This is also a line.
This is also a line.
This is a line too.
This is a line.
THIS IS A LINE.
```

```
$ uniq lines.txt
This is a line.
This is also a line.
This is a line too.
This is a line.
THIS IS A LINE.
```

uniq

Παρατηρούμε πως η 1η γραμμή του αρχείου είναι ίδια με την 4η γραμμή του. Για να γίνει σωστή ομαδοποίηση ίδιων γραμμών θα πρέπει η είσοδος της εντολής να είναι ταξινομημένη. Προσθέτοντας στην εντολή την επιλογή -c εμφανίζεται και το πλήθος των επαναλήψεων κάθε γραμμής:

```
$ sort lines.txt | uniq -c      # prefix by number
      2 This is a line.         #  of occurrences
      1 THIS IS A LINE.
      1 This is a line too.
      2 This is also a line.
```

uniq

```
$ sort lines.txt | uniq -d          # print only
This is a line.                  # duplicate lines,
This is also a line.          # one for each group
$ sort lines.txt | uniq -D     # duplicate lines
This is a line.
This is a line.
This is a line.
This is also a line.
This is also a line.
$ sort lines.txt | uniq -u      # unique lines
This is a line too.
```

Ερωτήσεις