"Contrary to popular belief, Unix is user friendly. It just happens to be very selective about who it decides to make friends with. "

Εργαστήριο $2^ο$: Κανονικές Εκφράσεις και Editors

Τσαδήμας Ανάργυρος, Τμήμα Πληροφορικής & Τηλεματικής

Οι κανονικές εκφράσεις είναι ένας πανίσχυρος μηχανισμός που μας επιτρέπει να ψάχνουμε για συγκεκριμένα πρότυπα (patterns) σε κείμενα. Χρησιμοποιείται σαν όρισμα σε αρκετές εντολές του UNIX αλλά και σχεδόν σε όλες τις γλώσσες προγραμματισμού.

1 Console Editors

Οι πιο δημοφιλής editors στο UNIX είναι οι παρακάτω:

- vi / vim [3]. Υπάρχει σε κάθε UNIX σύστημα. Δείτε το manual
- nano / pico. Εύχρηστος, απλός editor. Δείτε το manual
- emacs. Παλαιότερος όλων. Διαφέρει αρκετά από τους άλλους editors.

Για να μπορέσετε να χρησιμοποιήσετε τους console editors χρειαζεται να έχει οριστεί το terminal emulation package και το πιο δημοφιλές είναι το xterm. Σε περίπτωση που δεν σας λειτουργεί (Unknown terminal type error) ο editor δοκιμάστε την εξής εντολή `export TERM=xterm`. Περισσότερα για τη μεταβλητή TERM εδώ.

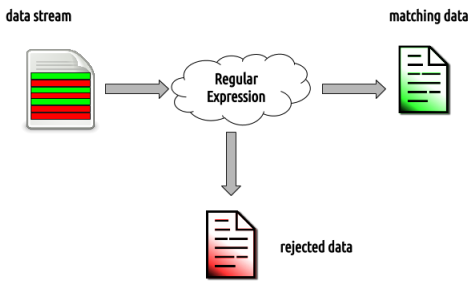
vifm

Αν ενδιαφέρεστε για έναν treminal file manager με επιρροές από τον vim editor, δοκιμάστε τον vifm. Δείτε τον οδηγό χρήσης του vifm.

2 Κανονικές εκφράσεις

Είδαμε ότι με την εντολή grep μπορούμε να κάνουμε αναζήτηση σχηματισμών σε αρχεία. Μπορούμε να χρησιμοποιήσουμε πιο πολύπλοκες εκφράσεις για αναζήτηση, οι οποίες ονομάζονται κανονικές εκφράσεις (regular expressions ή regex) [4][2]. Οι κανονικές εκφράσεις είναι ένας μηχανισμός ταιριάσματος προτύπων. Αντίθετα με το μηχανισμό που χρησιμοποιεί ο φλοιός για τη δημιουργία ονομάτων αρχείων, οι κανονικές εκφράσεις προσφέρουν κυρίως ισχύ και όχι ευκολία χρήσης. Ωστόσο, με την εξάσκηση και προσεκτική επισκόπηση όσων κανονικών εκφράσεων συναντάτε, μπορείτε να εκμεταλλευτείτε αυτό το δυνατό εργαλείο. Ανάμεσα στα πολλά utilities που τα χρησιμοποιούν, είναι οι: grep, sed, less, vi και awk.

Οι κανονικές εκφράσεις χρησιμοποιούνται για να φιλτράρουμε κάποιο κείμενο [1]. Αποτελούν όρισμα σε αρκετές εντολές και γλώσσες προγραμματι-



Σχήμα 1: Η λογική των κανονικών εκφράσεων

σμού και είναι ένας πανίσχυρος μηχανισμός ταιριάσματος προτύπων. Η λογική με την οποία λειτουργούν είναι απλή και φαίνεται στην εικόνα 1. Υπάρχουν δυο εκδοχές των κανονικών εκφράσεων στο UNIX:

- Basic Regular Expressions (BRE). Υποστηρίζονται από αρκετές εντολές του UNIX. Κάποιες ωστόσο υποστηρίζουν ακόμη και ένα υποσύνολο των BRE. Σε κάθε περίπτωση πρέπει να βλέπετε το manual των εντολών.

- Extended Regular Expressions (ERE). Συνήθως υποστηρίζεται στις γλώσσες προγραμματισμού. Υποστηρίζουν περισσότερα σύμβολα και λειτουργίες. Κάποιες εντολές υποστηρίζουν τις ERE όταν βάλουμε κάποιο συγκεκριμένη παράμετρο (option).

Υπάρχουν δυο ειδών regular expressions: τα βασικά και τα extended. Κάποια προγράμματα χρησιμοποιούν τα βασικά, όπως ο vi, sed, grep, more και άλλα τα extended, όπως το awk, egrep.

Οι κανονικές εκφράσεις μπορούν να χωριστούν σε τρεις κατηγορίες:

- wildcards, χρησιμοποιούνται στη θέση κάποιου άλλου χαρακτήρα
- modifiers, χρησιμοποιούνται για να τροποποιήσουν τον προηγούμενο χαρακτήρα
- anchors, χρησιμοποιούνται για να «αγκιστρώσουν» μία συμβολοσειρά στην αρχή ή στο τέλος μίας γραμμής ή λέξης

2.1 wildcards - quotations

Κάποιοι χαρακτήρες είναι ειδικοί χαρακτήρες όταν χρησιμοποιούνται σε κανονικές εκφράσεις. Αυτοί είναι οι:

```
. * [] {} ^ \$ \ + ? | ()
```

Μερικές φορές θέλουμε να αναφερθούμε σε πολλά αρχεία με ένα πέρασμα. Για παράδειγμα αν θέλουμε να αντιγράψουμε όλα τα αρχεία που έχουν κατάληξη .c σε ένα άλλο φάκελο, θα χρησιμοποιήσουμε wildcards. Είναι κάποιοι χαρακτήρες σαν το * και το ?, οι οποίοι αναπαριστούν οποιονδήποτε ή ένα σύνολο από χαρακτήρες. Είναι κάτι σαν τον μπαλαντέρ στα χαρτιά. Τα σύμβολα που μπορούν να είναι wildcards παρουσιάζονται στον πίνακα 1.

2.1.1 Special Character Classes

Στις κανονικές εκφράσεις υπάρχουν και κάποιες συγκεκριμένες character classes τις οποίες μπορούμε να χρησιμοποιούμε [1]. Αυτές οι κλάσεις[2] φαίνονται στον Πίνακα 2.

Είναι σημαντικό να ξέρουμε ότι το κέλυφος «απλώνει» τους χαρακτήρες-μπαλαντέρ: Δηλαδή όταν δίνουμε μια εντολή το πρόγραμμα δεν ξεκινά με μια παράμετρο που περιέχει χαρακτήρα wildcard, αλλά το κέλυφος «ανοίγει» το wildcard χαρακτήρα πρώτα και μετά εκτελεί την εντολή με τη λίστα των παραμέτρων που προκύπτουν από την αντικατάσταση των μπαλαντέρ με όλες τις πιθανές τιμές που παίρνουν.

Για παράδειγμα δοκιμάστε να δημιουργήσετε καταλόγους στον προσωπικό σας φάκελο, της μορφής dir1, dir2, ..., dir9 και μετά να τους διαγράψετε.

Χρειάζεται ιδιαίτερη προσοχή όταν διαγράφουμε αρχεία και καταλόγους και χρησιμοποιούμε wildcards. Τι θα γίνει αν εκτελέσουμε την εντολή `rm *` ; Μην το κάνετε!

[1] Στο Solaris η grep παίρνει σαν όρισμα κανονικές εκφράσεις. Προσοχή όμως γιατί η /usr/bin/grep δέχεται μόνο ένα υποσύνολο των κανονικών εκφράσεων. Αν θέλουμε να χρησιμοποιήσουμε τα special character classes θα πρέπει να χρησιμοποιήσουμε την /usr/xpg4/bin/grep βάζοντας και την παράμετρο -E. Στο Linux η grep τα υποστηρίζει κανονικά.

[2] Πηγή https://goo.gl/ncAEBm

.	ταιριάζει οποιονδήποτε χαρακτήρα εκτός από το newline
?	ταιριάζει έναν οποιονδήποτε χαρακτήρα. Προσοχή: Υποστηρίζεται μόνο στις Extended Regular Expressions. Για να χρησιμοποιηθεί σε κάποιες εντολές όπως η grep θα πρέπει να χρησιμοποιήσουμε κάποια παράμετρο (-E στην grep)
*	ταιριάζει 0 ή περισσότερους χαρακτήρες, εκτός αν ο αρχικός είναι η τελεία
$[C_1C_2$ ... $C_n]$	Αναφέρεται σε οποινδήποτε χαρακτήρα που εμπεριέχεται στη λίστα (character class)
$[C_1$ - $C_n]$	Αναφέρεται σε οποινδήποτε χαρακτήρα που εμπεριέχεται στη λίστα, η λίστα ξεκινάει από τον πρώτο χαρακτήρα μέχρι τον τελευταίο $[C_1$ -$C_n]$.
$[^abc]$	Αναφέρεται σε οποινδήποτε χαρακτήρα που δεν εμπεριέχεται στη λίστα
" "	«άπλωσε» τις εντολές και τις μεταβλητές
' '	παρέθεσε ακριβώς την έκφραση
\	παρέθεσε ακριβώς τον επόμενο χαρακτήρα

Πίνακας 1: Wildcards και quotations

[:alnum:]	Αλφαριθμητικοί χαρακτήρες. Περιλαμβάνει γράμματα και αριθμούς. Δεν περιλαμβάνει τα σημεία στίξης
[:alpha:]	Περιλαμβάνει μόνο τα γράμματα
[:blank:]	Χαρακτήρες όπως το κενό και το tab
[:cntrl:]	Χαρακτήρες control (μη εκτυπώσιμοι)
[:digit:]	Αριθμοί από 0-9
[:graph:]	Τα [:punct:], [:upper:], [:lower:], [:digit:] μαζί
[:lower:]	Οποιοσδήποτε πεζός αλφαβητικός χαρακτήρας
[:print:]	Οποιοσδήποτε εκτυπώσιμος χαρακτήρας
[:punct:]	Οποιοσδήποτε χαρακτήρας στίξης
[:space:]	Οποιοσδήποτε white space χαρακτήρας: space, Tab, NL, FF, VT, CR
[:upper:]	Οποιοσδήποτε κεφαλαίος αλφαβητικός χαρακτήρας
[:xdigit:]	Οποιοδήποτε χαρακτήρας του δεκαεξαδικού

Πίνακας 2: RE character classes

Προσοχή! Η αναζήτηση σχηματισμών γίνεται μόνο για τα υπάρχοντα ονόματα αρχείων. Δεν μπορούμε να δημιουργήσουμε νέα ονόματα αρχείων ή καταλόγων χρησιμοποιώντας σχηματισμούς.[3]

Ας δημιουργήσουμε κάποιους καταλόγους με δομή κεφαλαίων ενός βιβλίου (chapter1.1, chapter 1.2 ,..., chapter2.1, chapter2.2, ...).Πως θα γίνει αυτό με μια εντολή; Τώρα που τους δημιουργήσατε δοκιμάστε να τους μετονομάσετε σε directory1.1 κτλ. Τι παρατηρείτε;

Το σύμβολο * μπορεί να χρησιμοποιηθεί σε ονόματα διαδρομών. Για παράδειγμα πως θα αναφερθούμε σε όλα τα αρχεια .profile των χρηστών;

Μια εξαίρεση: το σύμβολο . (τελεία) δεν αντικαθίσταται εάν είναι το πρώτο σύμβολο στο όνομα ενός αρχείου (ή εάν είναι αμέσως μετά απο ένα /). Δοκιμάστε την ls *.

Ας υποθέσουμε ότι έχουμε ένα αρχείο το οποίο περιλαμβάνει τα εξής:

```
bt
bat
bet
btt
baat
baaeeet
baeeaeeat
baakeeet
```

Δοκιμάστε να βρείτε το pattern b[ae]*t και αιτιο-

[3]Για να δημιουργήσουμε νέα αρχεία ή καταλόγους χρησιμοποιούμε τις {...}, το οποίο σημαίνει «εκτέλεσε τις εντολές που ορίζονται από τα ... ». Για να δημιουργήσουμε καταλόγους π.χ. dir1, ... ,dir9 εκτελούμε την εντολή `mkdir dir{1..9}`.

/etc/rc.????	αναφέρεται σε όλα τα αρχεία στο φάκελο /etc τα οποία ξεκινούν από rc. και το όνομά τους έχει μέγεθος 7 χαρακτήρων
*.c	Αναφέρεται σε όλα τα αρχεία που έχουν κατάληξη .c (C προγράμματα)
*.[Cc]	Αναφέρεται σε όλα τα αρχεία που έχουν κατάληξη .c ή .C (C ή C++ προγράμματα)
*.[a-z]	Αναφέρεται σε όλα τα αρχεία που έχουν κατάληξη από .a μέχρι .z

Πίνακας 3: Παραδείγματα χαρακτήρων μπαλαντέρ

λογήστε το αποτέλεσμα.

2.2 modifiers

Ακολουθούν συνήθως ένα χαρακτήρα και τον προσδιορίζουν ποσοτικά. Υποστηρίζονται στις Advanced Regular Expressions.

- * μηδέν ή περισσότεροι από τον προηγούμενο χαρακτήρα
- + ένας ή περισσότεροι από τον προηγούμενο χαρακτήρα
- ? μηδέν ή ένας από τον ποηγούμενο χαρακτήρα
- {i} ακριβώς i από τον προηγούμενο χαρακτήρα
- {i,} i ή περισσότεροι από τον προηγούμενο χαρακτήρα
- {i,j} i έως j από τον προηγούμενο χαρακτήρα

Παραδείγματα: a*, ab*c, [a-z][a-z][0-9]*

Οι αγκύλες { και } όταν έπονται του χαρακτήρα \(backslash) συμπεριφέρονται ως μετρητές (counters), και προσδιορίζουν τον αριθμό του προηγούμενου χαρακτήρα που μας ενδιαφέρει. Παραδείγματα: r\{6\} , ca\{5,\}t

2.2.1 Pipe

Το σύμβολο (|) μας επιτρέπει να καθορίσουμε δύο ή περισσότερα patterns τα οποία θα ταιριάξει ο μηχανισμός των RE. Αν υποθέσουμε ότι έχουμε το αρχείο:

```
Hello there
1 dollar
there are 3 pets inside
well done
bt
bat
bet
btt
baat
baaeeet
baeeaeeat
baakeeet
```

Αν θέλουμε να φέρουμε όλες τις γραμμές οι οποίες περιέχουν λέξεις οι οποίες περιέχουν ένα b το οποίο ακολουθείται από ένα ή περισσότερα a και μετά από ένα t και όλες τις γραμμές οι οποίες περιέχουν αριθμούς, τότε θα εκτελέσουμε την εντολή:

```
grep -E "ba{1,}t|[[:digit:]]" file.txt
```

όπου file.txt το αρχείο.

2.2.2 Grouping

Οι παρενθέσεις χρησιμοποιούνται για να ομαδοποιούμε τις κανονικές εκφράσεις. Όταν ομαδοποιούμε κάτι, αυτό πλέον αντιμετωπίζεται όπως ένας απλός χαρακτήρας. Παράδειγμα:

```
grep -E "b(aee){1,2}" file.txt
```

2.3 anchors

Πολλά εργαλεία κειμένου στο UNIX είναι προσανατολισμένα στις γραμμές, δηλαδή κάνουν αναζήτηση για patterns τα οποία περιέχονται μεταξύ δυο end of line (EOL) seperators. Οι κανονικές εκφράσεις γενικά εξετάζουν το κείμενο το οποίο βρίσκεται ενδιάμεσα από αυτούς τους seperators. Αν θέλουμε να αναζητήσουμε ένα pattern το οποίο βρίσκεται στο ένα ή το άλλο άκρο, χρησιμοποιούμε τους anchors. Οι anchors αναφέρονται σε πρότυπα σχετικά με την αρχή ή το τέλος μίας γραμμής ή λέξης. Οι anchors είναι:

Παραδείγματα: ^Hi, the$, ^Hello$, \<cat

Όταν χρησιμοποιείτε κανονικές παραστάσεις στη γραμμή εντολών, πρέπει να τις χρησιμο-

^	Η γραμμή αρχίζει με
$	Η γραμμή τελειώνει με
\<	Η λέξη αρχίζει με
\>	Η λέξη τελειώνει με

Πίνακας 4: Anchors

ποιείτε εντός (μονών κατά προτίμηση) εισαγωγικών. Ο λόγος είναι ότι ο φλοιός θα αντικαταστήσει όλους τους ειδικούς χαρακτήρες (όπως *, [ ] και \) και αν στη διαδικασία αυτή ταιριάξουν ονόματα αρχείων, τα αποτελέσματα δεν θα είναι τα επιθυμητά. Ωστόσο, πρέπει να μην χρησιμοποιείτε εισαγωγικά για τα πρότυπα δημιουργίας ονομάτων αρχείων.

✎ Άσκηση

Βρείτε την κανονική έκφραση που να περιγράφει το εξής: Να ξεκινάει από C, να είναι η πρώτη λέξη της γραμμής, το δεύτερο γράμμα να είναι φωνήεν, να έχει μήκος 3 γράμματα και το 3ο γράμμα να είναι σύμφωνο.

✎ Άσκηση

Πειραματιστείτε με τις λέξεις που βρίσκονται στο /usr/share/lib/dict/words στο Solaris ή στο /usr/share/dict/words στο Linux.

✎ Άσκηση

Βρείτε την κανονική έκφραση η οποία αποτυπώνει τους Ταχυδρομικούς Κώδικες της Ελλάδας. (Δείτε εδώ)

✎ Άσκηση

Βρείτε την κανονική έκφραση ώστε να εντοπίζουμε τηλεφωνικούς αριθμούς της μορφής XXXXXXXXXX, XXX-XXXXXXX, XXX XXXXXXX όπου οι αριθμοί είναο από 0-9 εκτός από τον πρώτο που δεν πρέπει να είναι 0.

2.4 Υλικό για μελέτη

- Regular Expressions
- VI manual
- VI cheat sheet

Αναφορές

[1] R. Blum. Linux command line and shell scripting bible, volume 481. John Wiley & Sons, 2008.

[2] S. G. Kochan and P. Wood. Shell programming in unix, linux and os x. 2016.

[3] D. Neil. Practical Vim: Edit Text at the Speed of Thought. Pragmatic Bookshelf, 2nd edition, 2015.

[4] E. Nemeth. UNIX and Linux System Administration Handbook, 4/e. Pearson Education India, 2011.

Σχήμα 2: vim cheetsheet https://imgur.com/gallery/wjtaaxh