Λεπτομέρειες σχετικά με την εργασία στα Λειτουργικά Συστήματα

Στραβοράβδης Σπύρος

Απρίλιος 2017

Περίληψη

Οι ακόλουθες σημειώσεις είναι για το 1ο Θέμα της εργασίας, το shell script.

Είναι γραμμένο σε GNU Bash. Περισσότερες λεπτομέρειες σχετικά με τα διάφορα shells βρίσκονται παρακάτω, στην ενότητα *Συμβατότητα*.
Δημιουργήθηκε και δοκιμάστηκε σε Ubuntu 16.04 (Bash 4.3).
Το script δουλεύει στη μνήμη, δεν δημιουργεί περιττά αρχεία, απλά διαβάζει το αρχείο που δίνεται και στο ερώτημα G το τροποποιεί.
Για να διορθωθούν κάποια σφάλματα χρησιμοποιήθηκε το πολύ χρήσιμο *ShellCheck*[1].
Πηγές που μου χρησίμευσαν (μεταξύ άλλων): *Bash Guide for Beginners*[2], *Advanced Bash-Scripting Guide*[3], *The Bash Hackers Wiki*[4], αρκετές απαντήσεις στο *Stack Overflow* και φυσικά τα σχετικά manpages.

1 Λεπτομέρειες

1.1 Γενική δομή

Υπάρχει η κύρια συνάρτηση (που κάνει το διάβασμα από το shell) και μερικές υπο-συναρτήσεις (που υλοποιούν συγκεκριμένες χρήσεις, ανάλογα με το ερώτημα):

[1] https://github.com/koalaman/shellcheck
[2] http://www.tldp.org/LDP/Bash-Beginners-Guide/html/
[3] http://tldp.org/LDP/abs/html/
[4] http://wiki.bash-hackers.org/start

Για το ερώτημα A: `printFile()`
Για το ερώτημα B: `printId()`
Για το ερώτημα C: `printFirstnames()`
Για το ερώτημα D: `printLastnames()`
Για το ερώτημα E: `printBornSince()` ή `printBornUntil()` ή `printSinceUntil()`
Για το ερώτημα F: `printBrowsers()`
Για το ερώτημα G: `editColumn()`
Επιπλέον συνάρτηση: `usage()` (εμφανίζει οδηγίες για τη σωστή χρήση του script)

> Στο script οι υπο-συναρτήσεις βρίσκονται πριν από την κύρια συνάρτηση. Οι παρενθέσεις είναι προαιρετικές, τις έβαλα κυρίως για να ξεχωρίζουν.

1.2 Parsing

Χρησιμοποίησα την `getopts` για να διαβάσω τις εντολές από το shell. Ωστόσο επειδή αυτή δέχεται μόνο arguments ενός χαρακτήρα, πρέπει να δημιουργηθούν προσωρινά μονοψήφια ονόματα για κάθε argument που θα δώσει ο χρήστης. Οι συντομεύσεις αυτές φαίνονται στον *Πίνακα 1*.

Πλήρες όνομα εντολής	Συντόμευση
-id	-d
–firstnames	-i
–lastnames	-l
–born-since	-s
–born-until	-u
–browsers	-b
–edit	-e
–help	-h

Πίνακας 1: Αντιστοίχηση εντολών για χρήση από την `getopts`. Αριστερά οι εντολές που θα δίνει ο χρήστης. Δεξιά οι μονοψήφιες εντολές όπως θα δοθούν στην getopts. Η `--help` είναι επισημασμένη επειδή δεν απαιτείται από την εκφώνηση.

Θέλουμε ο χρήστης να μπορεί να δώσει τα arguments με όποια σειρά θέλει. Ωστόσο πριν εκτελεστούν τα υπόλοιπα arguments θα πρέπει να έχει φορτωθεί το file (αφού δε γίνεται να εμφανίσουμε output ή να κάνουμε αλλαγές χωρίς να διαβάσουμε τα δεδομένα). Γι' αυτό εκτελούμε 2 επαναλήψεις: 1 για να βρούμε το file (πριν από οτιδήποτε άλλο) και στη συνέχεια 1 ακόμα επανάληψη για τα υπόλοιπα arguments.

1.3 Ερωτήματα

Αν δοθεί μόνο το όνομα του script (πχ. `./myscript.sh`), χωρίς επιπλέον arguments, τυπώνουμε τον Αριθμό Μητρώου και κάνουμε exit.

> Αν δοθεί το όνομα του script + 1 argument (πχ. `./myscript.sh --help` ή `./myscript.sh --blabla`) καλείται η συνάρτηση `usage()` που εμφανίζει τον τρόπο χρήσης του script. Αυτό δεν είναι υποχρεωτικό από την εκφώνηση, ωστόσο δεν δημιουργεί κάποιο πρόβλημα.

1.3.1 Ερώτημα A

Σε περίπτωση που διαβάσουμε το `$file` και το `$#` (ο πίνακας με τα arguments) περιέχει μόνο 2 στοιχεία (άρα μας έχει δοθεί κάποια εντολή του τύπου `./myscript.sh -f users.dat`) εκτελούμε την `printFile()`.

Αλλιώς ελέγχουμε με την getopts και εκτελούμε την κατάλληλη συνάρτηση από τις υπόλοιπες.

1.3.2 Ερωτήματα B, C, D

Είναι αρκετά αυτονόητα, αναλόγως την περίπτωση που διαβάζουμε από το πληκτρολόγιο εκτελούμε και την κατάλληλη υποσυνάρτηση.

Στα ερωτήματα C και D χρησιμοποιούμε και τα προγράμματα `sort` και `cut`. Πρώτα ταξινομούμε με το `sort` (με βάση την ζητούμενη στήλη) και στη συνέχεια περνάμε το output στο `cut` το οποίο αφαιρεί τις υπόλοιπες στήλες. Στην συνέχεια εμφανίζουμε αυτό που απομένει.

1.3.3 Ερώτημα E

Το ερώτημα E έχει την δυσκολία ότι δεν ξέρουμε αν θα δοθεί `--born-since`, `--born-until` ή και τα δύο ταυτόχρονα, γι' αυτό ορίζουμε 3 υποσυναρτήσεις και καλούμε την κατάλληλη όταν βρούμε ποια από τις παραπάνω

περιπτώσεις ισχύει.

Ορίζουμε 2 flags: το `counterOfBorn` που το αρχικοποιούμε με 1 και το `flagOfBorn` που είναι false. Στη συνέχεια διαβάζουμε από το πληκτρολόγιο.

Αν έχει ζητηθεί να τυπώσουμε το `--born-since`, το `counterOfBorn` θα μειωθεί κατά 1 (θα γίνει δηλαδή 0), το `flagOfBorn` θα γίνει true και θα εκτελεστεί η συνάρτηση `printBornSince`.
Αν έχει ζητηθεί να τυπώσουμε το `--born-until`, το `counterOfBorn` θα αυξηθεί κατά 1 (θα γίνει δηλαδή 2), το `flagOfBorn` θα γίνει true και θα εκτελεστεί η συνάρτηση `printBornUntil`.
Αν έχει ζητηθεί κάποια εντολή που περιέχει και το `--born-since` και το `--born-until`, το `counterOfBorn` θα αυξηθεί κατά 1 και θα μειωθεί κατά 1 (θα παραμείνει δηλαδή 1) , το `flagOfBorn` θα γίνει true και θα εκτελεστεί η συνάρτηση `printSinceUntil`.

Σε περίπτωση που το `flagOfBorn` παραμείνει false δεν εκτελείται καμία από τις παραπάνω συναρτήσεις!

1.3.4 Ερώτημα F

Πρώτα αφαιρούμε με το `cut` όλες τις στήλες εκτός από αυτή με τους browsers. Ταξινομούμε το output με το `sort` και το στέλνουμε στο `uniq` το οποίο εμφανίζει κάθε όνομα μόνο μία φορά (και με το `-c` μετράει πόσες φορές παρουσιάζεται).

Υπάρχει μία μικρή απόκλιση από την εκφώνηση.
Οι browsers εμφανίζονται με αυτή τη σειρά:
1 Firefox
1 Internet Explorer
ενώ ζητείται το αντίθετο. Θα μπορούσαν να εμφανιστούν σωστά αν αποθήκευα το output σε ένα προσωρινό αρχείο και στη συνέχεια το αντέστρεφα με το awk (`awk 'print $2,$1' tmp && rm tmp`) , ωστόσο θέλησα το πρόγραμμα να μην χρειάζεται να γράφει στο δίσκο αλλά να τρέχει από την μνήμη.

1.3.5 Ερώτημα G

Επειδή έχουμε τρία arguments στο `--edit`, η getopts τα διαβάζει και τα τοποθετεί σε ένα μικρό πίνακα, με τη σειρά που δόθηκαν. Ελέγχουμε αν το πρώτο στοιχείο αντιστοιχεί σε κάποιο ID του αρχείου. Αν ναι, το 2ο στοιχείο αναφέρει τη στήλη που πρέπει να γίνει η αλλαγή. Ορίζουμε το περιεχόμενο του συγκεκριμένου σημείου στη μεταβλητή `columnContent`. Υπάρχει και η μεταβλητή `columnName`, η οποία όμως είναι προαιρετική, χρησιμοποιείται απλά στο output.

> Υπάρχει μία μικρή απόκλιση από την εκφώνηση.
> Αντί για `--edit <id> <column> <value>`, τα arguments του edit πρέπει να δοθούν με κόμμα (`--edit <id>,<column>,<value>`) ή, αν αλλάξουμε τον field separator από κόμμα σε κενό (`IFS=' '`), με εισαγωγικά (`--edit "<id> <column> <value>"`)

Λόγω των περιορισμών της εκφώνησης χρειαζόμαστε ένα πρόγραμμα που να κάνει απευθείας επεξεργασία (in-place editing) μόνο στο σημείο του αρχείου που καθορίζουμε. Το sed παρέχει αυτή την δυνατότητα. Το χρησιμοποιούμε για να χρησιμοποιήσουμε το `columnContent` με το τρίτο στοιχείο του πίνακα.

> Η επεξεργασία του αρχείου πιθανόν να μπορούσε να γίνει και με GNU awk, καθώς από την έκδοση 4.1 (2013) υποστηρίζει in-place editing όπως το sed.
> Ωστόσο ενώ προσπάθησα αρκετά, δεν μπόρεσα να το πετύχω (έσβηνε πάντα όλο το αρχείο).
> Μέσα στον κώδικα υπάρχει σαν σχόλιο στην `editColumn()` η γραμμή που δοκίμασα να υλοποιήσω με το awk.

2 Συμβατότητα

Το script δουλεύει σε Bash και (κατά τύχη) σε ksh93 και διαδόχους του, όπως το mksh.

Σκεφτόμουν να κάνω αλλαγές, ώστε να εξαλείψω τυχόν bashisms και να δουλεύει σε οποιοδήποτε άλλο shell υλοποιεί το POSIX standard (πχ. dash), γι' αυτό πχ. χρησιμοποίησα κυρίως `printf` αντί για `echo` (το output του οποίου διαφέρει ελαφρώς σε κάθε shell).

Ωστόσο έπειτα διαπίστωσα ότι τα arrays (τα οποία χρησιμοποιούνται στο ερώτημα G για τα τρία arguments του `--edit`) στο POSIX είναι undefined, δηλαδή συμπεριφέρονται διαφορετικά σε κάθε shell.

Αφού μια λύση χωρίς arrays δεν θα ήταν ιδιαίτερα πρακτική, άφησα το script σε Bash.

Οι περισσότερες από τις υπόλοιπες ασυμβατότητες φαίνονται να μπορούν να γραφτούν με τρόπο που ορίζεται από το POSIX, απλά με κάπως πιο δυσνόητο κώδικα.[5] [6]

[5] https://wiki.ubuntu.com/DashAsBinSh

[6] http://mywiki.wooledge.org/Bashism