Λειτουργικά Συστήματα

Εργασία 1

Η εργασία θα υλοποιηθεί σε σύστημα linux, οπότε είτε με εικονική μηχανή είτε αν έχετε λειτουργικό linux όπως το Ubuntu να έχετε πρόσβαση στον υπολογιστή σας σε λειτουργικό σύστημα ανοικτού λογισμικού.
Το shell (κέλυφος) είναι ένα πρόγραμμα που δέχεται εντολές από το πληκτρολόγιο και δημιουργεί processes (διεργασίες) για να τις εκτελέσει. Όποτε πληκτρολογείτε μια εντολή στο τερματικό ενός συστήματος Unix (π.χ., ls, ps, date, wc, κτλ.), ουσιαστικά το πρόγραμμα που διαβάζει αυτά που γράφετε και εκτελεί τις αντίστοιχες εντολές είναι το shell.
Συνήθως τα shells παρέχουν πολλές επιπλέον λειτουργίες, όπως π.χ., την εκτέλεση scripts (μέσω ενός ενσωματωμένου interpreter), διαχείριση environment variables, και άλλες. Στην παρούσα άσκηση, ο στόχος είναι η υλοποίηση ενός μινιμαλιστικού shell, του οποίου η μόνη λειτουργία θα είναι να διαβάζει εντολές από το πληκτρολόγιο και να τις εκτελεί.

Θέμα 1

Γράψτε ένα πρόγραμμα σε BASH shell για τη διαχείριση ενός αρχείου χρηστών κοινωνικού δικτύου, το οποίο περιέχει στοιχεία στο παρακάτω format:

```
id|lastName|firstName|gender|birthday|joinDate|IP|browser
```

Γραμμές που ξεκινούν με # θεωρούνται σχόλια και πρέπει να αγνοούνται. Το πρόγραμμά σας θα πρέπει να ονομάζεται project.sh, και θα πρέπει να υποστηρίζει τις παρακάτω λειτουργίες:
`./project.sh`
Όταν το τρέχετε χωρίς καμία παράμετρο, θα εμφανίζει τον ΑΜ σας, χωρίς KANENAN άλλο χαρακτήρα. Παράδειγμα:
```
5000
```
```
   A)    ./project.sh -f <file>
```
Όπου `<file>` είναι το όνομα του αρχείου χρηστών (π.χ., `users. dat`). (Εδώ καλό θα ήταν να έχουμε 5 χρήστες).Θα εμφανίζει όλα τα περιεχόμενα του αρχείου, χωρίς τις γραμμές-σχόλια που πρέπει να έχουν αγνοηθεί!
```
   B)    ./project.sh -f <file> -id <id>
```
Θα εμφανίζει το όνομα, επώνυμο, και ημερομηνία γέννησης του χρήστη με το συγκεκριμένο `id`, χωρισμένα με μονά κενά. Για παράδειγμα:
```
 $  ./project.sh -f users.dat -id  Vasilis Papadopoulos 1997-12-3
```
```
   C)    ./project.sh --firstnames -f <file>
```
Θα εμφανίζει όλα τα διακριτά πρώτα ονόματα (πεδίο `firstname`) που περιέχονται στο αρχείο, σε αλφαβητική σειρά, ένα ανά γραμμή.
```
   D)    ./project.sh --lastnames -f <file>
```
Θα εμφανίζει όλα τα διακριτά επώνυμα (πεδίο lastname) που περιέχονται στο αρχείο, σε αλφαβητική σειρά, ένα ανά γραμμή.
```
   E)    ./project.sh    --born-since    <dateA>
         --born-until    <dateB>    -f <file>
```
Θα εμφανίζει μόνο τις γραμμές που αντιστοιχούν σε χρήστες που έχουν γεννηθεί από την ημερομηνία dateA μέχρι την ημερομηνία dateB. Μπορεί να δοθεί και το ένα από τα δύο, π.χ., για να εμφανίσει όλους του χρήστες γεννημένους από μια μέρα κι έπειτα, ή γεννημένους μέχρι κάποια μέρα. Για κάθε χρήστη που πληροί τα κριτήρια, να εμφανίζεται αυτούσια η γραμμή όπως ακριβώς ήταν στο αρχείο. Προφανώς δεν θα περιλαμβάνονται σχόλια.

```
   F)    ./project.sh --browsers -f <file>
```
Θα εμφανίζει όλους τους browsers που χρησιμοποιούνται, σε αλφαβητική σειρά, και μετά από κάθε browser τον αριθμό των χρηστών που τον χρησιμοποιούν (με ακριβώς ένα κενό για χώρισμα). Για παράδειγμα:
```
Chrome 143
Firefox 251
Internet Explorer 67
```

G) `./project.sh -f <file> --edit <id> <column> <value>`
Η εντολή αυτή θα μεταβάλει το αρχείο. Συγκεκριμένα, για τον χρήστη με κωδικό `<id>`, θα αντικαθιστά τη στήλη `<column>` με την τιμή `<value>`. Αν δεν υπάρχει κανένας χρήστης με αυτό το `id`, ή η στήλη δεν είναι μεταξύ των αποδεκτών στηλών 2 έως 8 (η στήλη 1 που αντιστοιχεί στο ίδιο το `id` δεν επιτρέπεται να αλλάξει), η εντολή αυτή δεν θα μεταβάλει τίποτα. Πέρα από τη ζητούμενη αλλαγή, δεν θα πρέπει να μεταβάλλεται τίποτα άλλο, δηλαδή πρέπει να διατηρείται η αρχική ταξινόμηση των εγγραφών συμπεριλαμβανομένων των σχολίων.

Σε όλα τα παραπάνω, οι παράμετροι πρέπει να μπορούν να δίνονται με οποιαδήποτε σειρά. Για παράδειγμα τα παρακάτω είναι ταυτόσημα:

```
./project.sh --born-since <A> --born-until <B> -f <F>
./project.sh --born-until <B> -f <F> --born-since <A>
./project.sh –f <F> --born-until <B> --born-since <A>
./project.sh –f <F> --born-since <A> --born-until <B>
```

Θέμα 2: Υλοποίηση των shells

Θα πρέπει να υλοποιήσετε τέσσερα shells, που έχουν βαθμιαία μεγαλύτερη πολυπλοκότητα και δυσκολία. Τα shells αυτά θα πρέπει να ονομαστούν `osshl, ossh2, ossh3, και ossh4`. Τα τέσσερα αυτά shells θα πρέπει να παρέχουν τις εξής λειτουργίες:

1. Το `osshl` διαβάζει το όνομα ενός προγράμματος από το πληκτρολόγιο, και στη συνέχεια το εκτελεί. Σημειωτέον, το προς εκτέλεση πρόγραμμα μπορεί να βρίσκεται σε οποιοδήποτε `directory` του `$path.`

Για να το εκτελέσει, δημιουργεί μια νέα διεργασία (διεργασία-παιδί), η οποία εκτελεί το εν λόγω πρόγραμμα. Το `shell` (διεργασία-πατέρας) περιμένει να ολοκληρωθεί η εκτέλεση του προγράμματος πριν δεχτεί την επόμενη εντολή. Για παράδειγμα, αν δώσετε στο `prompt` του osshl την εντολή `"is"`, θα πρέπει να εκτελεστεί το γνωστό `/bin/ls` και να εμφανίσει στην οθόνη (πιο σωστά: στο `standard output`) τη λίστα αρχείων στο τρέχον `directory`. Μια ειδική περίπτωση είναι η εντολή `"exit"` η οποία δεν αντιστοιχεί σε εκτελέσιμο πρόγραμμα, αλλά απλά τερματίζει το `ossh1`.

Κάθε shell, για να δείξει ότι είναι έτοιμο να δεχθεί την επόμενη εντολή, εμφανίζει στην οθόνη ένα `prompt`. Στα πλαίσια της άσκησης, το prompt του osshl πρέπει να είναι το "$ ", δηλαδή ακριβώς το σύμβολο του δολαρίου ακολουθούμενο από ένα κενό, χωρίς κανέναν άλλο χαρακτήρα ούτε πριν ούτε μετά.

2. Το `ossh2` θα αποτελεί μια επέκταση του osshl. Συγκεκριμένα, πέρα από την εκτέλεση προγραμμάτων βάσει μόνο του ονόματος, θα πρέπει να υποστηρίζει και παραμέτρους οι οποίες θα περνιούνται στο πρόγραμμα που θα εκτελεστεί.

Για παράδειγμα, το νέο σας `shell` θα πρέπει να υποστηρίζει εντολές όπως η `"is -l /tmp''`. Επίσης, το `ossh2` θα πρέπει να υποστηρίζει και την εντολή `cd`, η οποία θα πρέπει να υποστηρίζει και absolute `paths (π.χ., cd /home/user)` και relative paths `(π.χ., cd ../Pictures)`. Για την υλοποίηση του `cd`, θα πρέπει να διαβάσετε τα `manual pages των chdir (2) και getcwd(3)`.

3. Το `ossh3` είναι περαιτέρω επέκταση του `ossh2`. Προσθέτει υποστήριξη για `pipes`, όπως η εντολή `"ls /tmp | wc -l"`. Μπορείτε να υποθέσετε ότι οι εντολές θα περιέχουν το πολύ ένα `pipe`, δηλαδή δεν χρειάζεται να υποστηρίξετε εντολές με δύο ή περισσότερα pipes όπως η `"sort foo | uniq -c | wc -l''`.

4. Το `ossh4` θα είναι σαν το `ossh3` αλλά θα υποστηρίζει και πολλαπλά pipes, π.χ., εντολές όπως η

```
cut -f 2 myfile.txt | sort -n | uniq | wc.
```

Κατά την υλοποίηση του ossh3, θα χρειαστείτε να χρησιμοποιήσετε το `system call dup(2)` ή εναλλακτικά το `dup2(2)`. Ανατρέξτε στα manual pages (man dup) για την ακριβή λειτουργία που παρέχουν. Για να κάνετε parsing της εντολής που έχει δώσει ο χρήστης, ίσως σας βοηθήσει η χρήση της `strtok (3)`, χωρίς όμως να είναι υποχρεωτικό.

Προφανώς, η χρήση της εντολής `system()` μπορεί να αποφευχθεί σε αυτή την άσκηση. Εξίσου προφανώς μπορεί να αποφευχθεί το `invocation` ενός άλλου shell (π.χ., `/bin/sh`) που θα κάνει τη δουλειά για λογαριασμό σας

Σε περίπτωση λαθεμένου input, π.χ., αν η εντολή που πληκτρολογήθηκε δεν αντιστοιχεί σε υπάρχον εκτελέσιμο, εννοείται πως το shell σας δεν θα πρέπει να τερματίζει ή να κολλάει. Σε τέτοια περίπτωση το shell σας ΛΕΝ ΘΑ ΠΡΕΠΕΙ ΝΑ ΒΓΑΖΕΙ ΚΑΝΕΝΑ OUTPUT στο `stdout`. Καλύτερα να μην βγάζει κανένα απολύτως μήνυμα, αλλά αν θέλετε να τυπώσετε οπωσδήποτε μήνυμα Command not found, αυτό επιτρέπεται μόνο στο `stderr`!!

Οδηγίες παράδοσης

Καλείστε να υποβάλετε την εργασία σας στο περιβάλλον της ασύγχρονης στην σελίδα του μαθήματος στο εδάφιο της εργασίας μέχρι τις 28/4.
Θα δημιουργήσετε ένα συμπιεσμένο αρχείο OSProgectAM, όπου AM θα θέσετε τον αριθμό μητρώου σα (το συμπιεσμένο αρχείο μπορεί να είναι σε όποια μορφή θέλετε tar,targzip,zip κ.α)ς. Η εργασία σας θα πρέπει να περιέχει τα αρχεία

1. Το πρόγραμμά σας να ονομάζεται project.sh για το θέμα 1
2. Τα τέσσερα αρχεία ossh[1234] για το θέμα 2
3. Την αναφορά της εργασίας σας σε pdf μορφή που θα περιγράφει την λογική που ακολουθήσατε και προβλήματα που αντιμετωπίσατε με όνομα details.pdf

Μην αποστείλετε την εργασία με email

Καλή επιτυχία!!