Λειτουργικά Συστήματα

Project εξαμήνου
Φεβρουάριος 2018, ΤΗΜΜΥ ΑΠΘ

Φανάκης Απόστολος
ΑΕΜ: 8261

Table of Contents

1) Περίληψη

Στο project του εξαμήνου ζητήθηκε να αναπτύξουμε ένα πρόγραμμα κελύφους (shell) σε γλώσσα C εφαρμόζοντας τις γνώσεις που αποκτήσαμε κατά τη διάρκεια του εξαμήνου. Συγκεκριμένα για την συγγραφή του προγράμματος χρειάστηκαν, εκτός από γενικές γνώσεις προγραμματισμού, κατανόηση των διεργασιών του λειτουργικού (proccesses), γνώση τεχνικών γέννησης (spawn) διεργασιών σε περιβάλλον linux, εκτέλεση προγραμμάτων με κλήση από τη γραμμή εντολών (terminal) και η αντίστοιχη διαδικασία μέσα από ένα πρόγραμμα C και άλλες γνώσεις σχετικές με τα linux (ή unix γενικότερα) λειτουργικά.

Ζητήθηκε το κέλυφος να λειτουργεί με διαδραστικό (interactive) τρόπο ερμηνεύοντας και εκτελώντας εντολές που δίνει ο χρήστης κατά την εκτέλεση του προγράμματος shell, ή σε λειτουργία batch διαβάζοντας και εκτελώντας εντολές αποθηκευμένες σε κάποιο αρχείο δέσμης (script).

2) Κώδικας προγράμματος και περιορισμοί

Ο κώδικας βρίσκεται ανεβασμένος στο repository:

https://gitlab.com/Apostolof/authOperatingSystemsCourseProject

Η εργασία περιλαμβάνει τα εξής παραδοτέα αρχεία:

1. αρχεία `.c` και `.h` : κώδικας εργασίας
2. `Makefile` : αρχείο για τη μεταγλώττιση του κώδικα
3. `apostolofShellManpage` : τεκμηρίωση του κωδικά, άνοιγμα με χρήση της εντολής `man ./apostolofShellManpage`
4. `testScript` : αρχείο δοκιμής του batch mode

Ο κώδικας χωρίζεται σε δύο μέρη:

1. την main (αρχείο `apostolofShell.c`)
2. ένα header (αρχείο `apostolofShellFunctions.h`) που περιέχει τις δηλώσεις των συναρτήσεων και σταθερών και το αντίστοιχο αρχείο κώδικα που περιέχει την υλοποίηση τους (αρχείο `apostolofShellFunctions.c`)

Το πρόγραμμα αναπτύχθηκε ακολουθώντας κάποιες παραδοχές και περιορισμούς, όπως:

- η είσοδος περιορίζεται σε 512 χαρακτήρες τη φορά
- το κέλυφος μπορεί να ερμηνεύσει **εκτελέσιμες** εντολές και τους χαρακτήρες διπλό ampersand (&&) και semicolon (;), όχι όμως εντολές που δεν είναι εκτελέσιμες και άλλους ειδικούς χαρακτήρες όπως:
  - pipeline (|)
  - single ampersand suffix (&)
  - cd, exit, set, unset, export, history και λοιπές μη εκτελέσιμες εντολές

  κατά την είσοδο των οποίων η έξοδος του προγράμματος είναι απροσδιόριστη

Η λειτουργικότητα του προγράμματος βασίζεται στις εξής 5 βασικές συναρτήσεις:

```
char *trimWhitespaces(char *string)
```

αλλάζει τη θέση του pointer και του null terminator '\0' ώστε να απορρίψει τους χαρακτήρες κενού[1] στην αρχή και το τέλος του string, επιστρέφει το τροποποιημένο string

```
Parameters:
string - input to trim

Returns:
    modified string
```

```
void parseCommand(char *input, char **command, char ***argv, int *argc)
```

δεσμεύει μνήμη για τα ορίσματα command, argv, argc, σπάει την είσοδο σε κομμάτια χωρισμένα στον χαρακτήρα κενό ' ', αποθηκεύει την εντολή στο όρισμα command, τα ορίσματά της εντολής στο όρισμα argv και τον αριθμό των ορισμάτων στο όρισμα argc

```
Parameters:
input - input to parse
command - pointer to a string for storing the parsed command
argv - pointer to an array of strings for storing the parsed arguments
argc - pointer to an int for storing the number of arguments

Returns:
    void
```

```
int execute(char *command, char **argv)
```

δημιουργεί μία δευτερεύουσα (θυγατρική) διεργασία η οποία εκτελεί την εντολή command με ορίσματα argv και επιστρέφει την τιμή εξόδου της εκτέλεσης

```
Parameters:
command - command to execute
argv - arguments of the command

Returns:
    return value of the command execution
```

[1]: *Για τον διαχωρισμό των χαρακτήρων κενού από τους υπόλοιπους χαρακτήρες κειμένου χρησιμοποιείται η συνάρτηση* `isspace` *της βιβλιοθήκης* `ctype.h`*, σύμφωνα με την οποία χαρακτήρες κενού θεωρούνται οι: space ' ', horizontal tab '\t', newline '\n', vertical tab '\v', feed '\f', carriage return '\r'*

```
void conditionalRun(char *input, int *quit_called)
```

σπάει το input σε κομμάτια χωρισμένα στο ζεύγος χαρακτήρων ampersand '&&', κάνει parse την εντολή και τα ορίσματα κάθε κομματιού, εκτελεί κάθε κομμάτι διαδοχικά μόνο αν όλα τα προηγούμενα είχαν επιτυχημένη εκτέλεση, αν κάποια από τις εντολές είναι η εντολή "quit" η σημαία `quit_called` γίνεται αληθής και η συνάρτηση επιστρέφει

Parameters:
`input - input to parse`
`quit_called - pointer to an int flag`

Returns:
`void`

```
void unconditionalRun(char *input, int *quit_called)
```

σπάει το input σε κομμάτια χωρισμένα στο χαρακτήρα semicolon ';', καλεί την `conditionalRun` διαδοχικά για κάθε κομμάτι ανεξάρτητα από την επιτυχία ή αποτυχία εκτέλεσης των προηγούμενων κομματιών, αν κάποιο από τα κομμάτια περιέχει την εντολή "quit" η σημαία `quit_called` γίνεται αληθής και η συνάρτηση επιστρέφει

Parameters:
`input - input to parse`
`quit_called - pointer to an int flag`

Returns:
`void`

Επιπλέον των παραπάνω χρησιμοποιούνται δύο συναρτήσεις οι οποίες καλούνται από την main και υλοποιούν τα δύο modes του shell:

```
void interactiveMode()
```

διαβάζει εντολές από τον χρήστη και τις εκτελεί μέχρι να δοθεί η εντολή "quit"

Parameters:

Returns:
`void`

```
void batchFileMode(char *filename)
```

διαβάζει εντολές από το αρχείο filename εκτελεί μέχρι να διαβάσει την εντολή "quit" ή να τελειώσει το αρχείο

Parameters:
`filename - path and filename of the script`

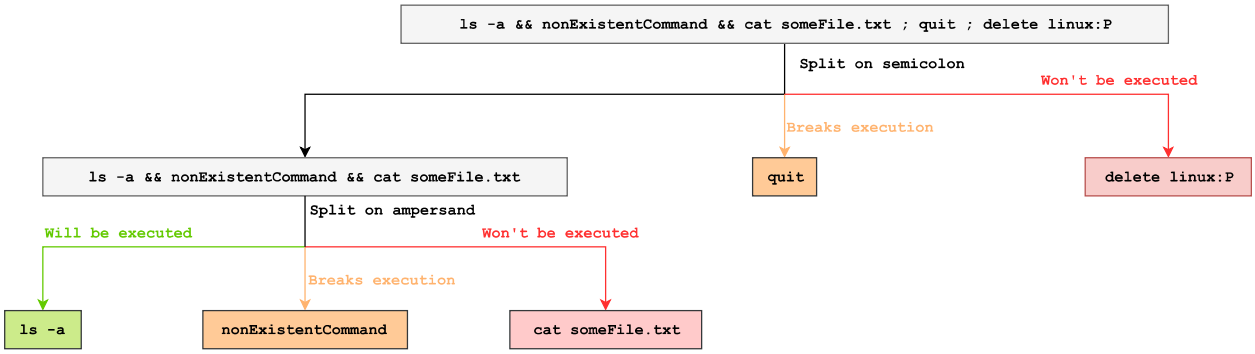
Returns:
`void`

3) Λειτουργία

Για τη λειτουργία του προγράμματος απαιτείται η μεταγλώττιση του κώδικα, εκτελώντας την εντολή `make` από τον φάκελο στον οποίο βρίσκεται. Η ερμηνεία των εντολών από το πρόγραμμα γίνεται με το ίδιο τρόπο ανεξάρτητα από το mode στο οποίο λειτουργεί το shell. Κάθε input επεξεργάζεται ως εξής:

- Πρώτα χωρίζεται στα semicolons
- Κάθε κομμάτι χωρίζεται περαιτέρω στα ampersands
- Κάθε κομμάτι εκτελείται αν τα προηγούμενα chunks εκτελέστηκαν επιτυχώς

Ένα παράδειγμα εκτέλεσης:
Input: `ls -a && nonExistentCommand && cat someFile.txt ; quit ; delete linux :P`



Μέρος 3.1) Interactive

Για την interactive λειτουργία του προγράμματος αρκεί να εκτελεστεί το αρχείο apostolofShell που προκύπτει από τη μεταγλώττιση του κώδικα χρησιμοποιώντας την εντολή `./apostolofShell`. Έπειτα μέσα στο πρόγραμμα μπορούμε να εκτελέσουμε τις επιθυμητές εντολές.

```
fanakis_8261> pwd
/home/apostolof/Documents/Workspaces/C/operatingSystemsCourseProject
fanakis_8261> ls
apostolofShell    apostolofShellDeclarations.c  Makefile  testScript
apostolofShell.c  apostolofShellDeclarations.h  tests
fanakis_8261> quit
```

Screenshot 1: Interactive λειτουργία του shell

Μέρος 3.2) Batch

Για την batch λειτουργία του προγράμματος ο χρήστης πρέπει να προσδιορίσει το αρχείο δέσμης, δίνοντας τη σχετική ή απόλυτη διαδρομή (relative/absolute path) προς το αρχείο σαν όρισμα γραμμή εντολών κατά την εκτέλεση του shell χρησιμοποιώντας την εντολή `./apostolofShell /path/to/script`.

```
apostolof@apostolof:/media/apostolof/0E88958A889570C7/Workspaces/C/operatingSyst
emsCourseProject$ cat testScript
mkdir someDir && touch someDir/file
mkdir someDir/dir2 ; touch someDir/file2


ls someDir
touch someDir/dir2/file3
apostolof@apostolof:/media/apostolof/0E88958A889570C7/Workspaces/C/operatingSyst
emsCourseProject$ ./apostolofShell testScript
dir2  file  file2
```

Screenshot 2: Batch λειτουργία του shell