Λειτουργικά Συστήματα

Τομέας Ηλεκτρονικής και Υπολογιστών

Τμήμα ΗΜΜΥ

Α.Π.Θ.

Φεβρουάριος 2019

Λειτουργικά Συστήματα

Project: Creating a Shell in Unix Environment

Δημήτριος Παππάς

ΑΕΜ: 8391

Email: dspappas@auth.gr

Λειτουργικά Συστήματα

Τομέας Ηλεκτρονικής και Υπολογιστών

Τμήμα ΗΜΜΥ

Α.Π.Θ.

Φεβρουάριος 2019

Τι είναι το Shell;

Το Shell είναι ένας command-line interpreter, δηλαδή ένα πρόγραμμα το οποίο δέχεται εντολές από το πληκτρολόγιο μέσω Terminal ή από κάποιο αρχείο και χρησιμοποιείται από το λειτουργικό για τον έλεγχο και την εκτέλεση εντολών.

Βασικές λειτουργίες

Με το παρακάτω Shell είναι δυνατή η εκτέλεση διάφορων ενερειών, όπως εκτέλεση απλών ή και πολλάπών εντολών σε ένα command line, piping, redirection, change direction, αναγνώριση και εκτέλεση εντολών από αρχείο, διαχείριση κενών χαρακτήρων, κενής γραμμής στο τερματικό και σε αρχείο. Με το πέρας μιας εντολής το Shell τυπώνει το prompt “Pappas_8391> ”. Οι συναρτήσεις που χρησιμοποιήθηκαν είναι οι εξής:

void init_shell()

Κάνει clear το terminal με την εντολη clear() και καλωσορίζει τον user που άνοιξε το Shell, βρίσκοντας το όνομα του μέσω της build-in συνάρτησης printDir().

int cd(char* path)

Εκτελούμε change direction με μία build-in συνάρτηση, διότι το execvp(), αδυνατεί να την εκτελέσει.

int getInput(char* input)

Εξασφαλίζει την εμφάνιση του prompt “Pappas_8391> ” και διαβάζει και αποθηκεύει την εντολή που έδωσε ο χρήστης στον char pointer input. Αποθηκεύει την εντολή στο ιστορικό του Shell με την add_history(buff) και επιστρέφει 0. Αν δεν κατάφερε να υλοποιηθεί επιστρέφει 1.

void parseSpace(char* input, char** parsed)

Με τη βοήθεια της ‘strsep’, κάθε φορά που υπάρχει space στο input, διαχωρίζουμε το string και αποθηκεύουμε τα τμημματα που δημιουργούνται στο double pointer parsed. Η διαδικασία σταματάει όταν διαβάσει NULL ή όταν φτάσει στο άνω όριο χαρακτήρων εντολής, που είναι οι MAXCHAR = 512 χαρακτήρες.

void parseSymbols(int* nc, int* ns, char* input, char** commands, char* symbols)

Λειτουργεί με παρόμοιο τρόπο με την parseSpace, με την διαφορά ότι το split του sting γίνεται όταν βρει τους χαρακτήρες ‘;’ και ‘&’. Επιπλέον, αποθηκεύει των αριθμό και την σειρά τον συμβόλων στον πίνακα symbols.

int check(char* input)

Ελέγχει αν στο input υπάρχουν οι χαρακτήρε ‘;’ και ‘&’ και μας αν υπάρχει έστω και ένας σημαίνει ότι δόθηκαν πολλαπλές εντολές και επιστρέφει 1. Αν υπάρχει μόνο μια εντολή επιστρέφει 0.

void executeCommand(char** parsed)

Καλείται όταν υπάρχει απλή εντολή. Καλούμε την fork(). Το παιδί εκτελεί την εντολή, που είναι αποθηκευμένη στο parsed[0] και τα ορίσματα της στα parsed[1], parsed[2], … , parsed[…] =NULL. Η εκτέλεση της εντολής γίνεται με την execvp, η οποία σκοτώνει το παιδί με το πέρας της. Σε περίπτωση μη σωστής εκτέλεσης, μας ενημερώνει με μήνυμα και κάνουμε exit. Ο πατέρας περιμένει να τελειώσει το παιδί.

void executeMultiCommands(int nc, int ns, char** commands, char* symbols)

Λειτουργεί όπως η executeCommand μόνο που διαχειρίζεται την εκτέλεση πολλαπλών εντολών. Μέσα σε μια for καλει fork() και για καθε επαναληψη/εντολη καλειται η parsedSpace ωστε να χωρισει την εντολη σε υποπίναες που περιεχουν την εντολη και τα ορίσμα της. Το παιδί εκτελει την εντολή και ο πατέρας το περιμένει. Στην περίπτωση που πριν την εκτολή προς εκτέλεση υπάρχει ‘;’ η διαδικασία συνεχίζει ανεξαρτήτως την ορθή εκτέλεση της προηγούμενης εντολής. Στην περίπτωση του ‘&’ αν η προηγούμενη εντολή δεν εκτελέστηκε σωστά, η εντολή δεν εκτελείται και ελέγχουμε την επόμενη.

int parsePipe(char* input, char** pipeArgs)

Ελέγχει αν υπάρχει pipe και αν ναι, χωρίζει την εντολή στα δύο τμήμματα του Pipe και τα αποθηκεύει σε έναν πίνακα. Επσιτρέφει 0 αν δεν βρήκε pipe και 1 αν βρήκε.

void executePipeCommands(char** parsePipeRead, char** parsePipeWrite)

Στην περίπτωση που υπάρχει pipe ορίζουμε τον pd που περιέχει το read end στο pd[0] και το write end της σωλήνωσης στο pd[1]. Καλούμε δύο fork(). Το 1^{ο} παιδί ανακατευθήνει το write end στο STDOUT μεσω της dup2 ενώ το 2^{ο} ανακατευθήνει το read end στο STDIN. Ο πατέρας περιμένει και τα δύο παιδια να ολοκληρωθούν μετα την εκτέλεση των execvp.

int parseRedirection(char* input, char** redirArgs)

Ελέγχει αν υπάρχει redirection command και αν ναι, χωρίζει την εντολή σε δύο τμήμματα και τα αποθηκεύει σε έναν πίνακα. Επσιτρέφει 0 αν δεν βρήκε ‘>’ και 1 αν βρήκε.

void executeRedirection(char** parseRedir, char** redirArgs)

Δημιουργεί ένα αρχείο με όνομα την τιμή του 2^{ου} μέλους του redirection που είναι αποθηκευμένο στο redirArgs[1]. Σε περίπτωση σφάλματος στο άνοιγμα του αρχείου, μας τυπώνει το error στο αρχείο "cerr.log". Πάλι με την dup και την dup2 κάνουμε ανακατεύθηνση της εξόδου από το STDOUT στο file και εκτελούμε την εκτολή παρομοίως. Στο τέλος κάνουμε fflush για να καθαρίσει το κανάλι.

main

Input from terminal:

Το πρόγραμμα παραμένει σε αδράνεια μέχρι να δωθεί εντολή μλέσω τερματικού. Η εντολή αποθηκεύεται και στη συνέχεια πραγματοποιούνται οι έλεγχοι για το αν υπάρχει Pipe, Redirection ή πολλαπλές εντολές. Στη συνέχεια μέσα από εμφωλευμένες if ελέγχουμε αν πρέπει να εκτελεστουν αρχικά οι build-in συναρτήσεις και έπειτα εκτελείτε η κατάλληλη execute συνάρτηση με βάση τους παραπάνω ελέγχους. Μετά το τέλος των execute το πρόγραμμα ξανα μπαίνει σε αδράνεια και περιμένει νέα εντολή.

Input from file:

Ο file pointer διαβάζει το αρχείο μέχρι να βρεί το τέλος του αρχείου ή την εντολή quit. Η εκτέλεση των εντολών πραγματοποιείται με ττον ίδιο τρόπο όπως και στο τερματικό. Μετά το τέλος των execute το πρόγραμμα τερματίζει.

Στιγμιότυπα και Αποτελέσματα

cd:

```
USER is: @dimitris
Pappas_8391> cd /bin
/bin
Pappas_8391> pwd
/bin
Pappas_8391>
```

Simple command

```
****************************************


        ****  D-SHELL  ****

****************************************


USER is: @dimitris
Pappas_8391> ls -l
total 116
-rw-r--r-- 1 dimitris dimitris 10970 Jan 27 15:32  backup2.c
-rw-r--r-- 1 dimitris dimitris 10969 Jan 30 11:13  backup.c
-rw-r--r-- 1 dimitris dimitris     0 Feb  5 12:02  brandNewFile
-rw------- 1 dimitris dimitris     0 Feb  5 10:03  cerr.log
-rw-r--r-- 1 dimitris dimitris   143 Jan 30 17:33  dFile
-rwxr-xr-x 1 dimitris dimitris 18480 Feb  5 11:58  dShell
-rw-r--r-- 1 dimitris dimitris 13821 Feb  5 11:36  dShell.c
-rwxr-xr-x 1 dimitris dimitris 13568 Jan 27 12:09  exampleShell
-rw-r--r-- 1 dimitris dimitris  5471 Jan 20 13:42  exampleShell.c
-rw-r--r-- 1 dimitris dimitris   636 Jan 20 15:05  execvp.c
-rw------- 1 dimitris dimitris   360 Feb  5 12:02 ' lsFile '
-rw-r--r-- 1 dimitris dimitris    89 Jan 30 16:23  makefile
-rw-r--r-- 1 dimitris dimitris 13821 Feb  5 11:36  myshell.c
Pappas_8391>
```

Multiple Commands

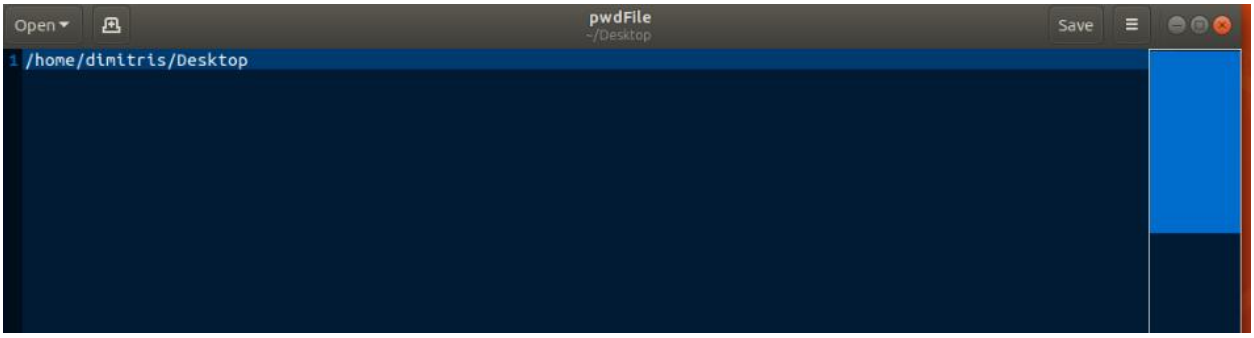
```
Pappas_8391> ls ; kati & pwd
 backup2.c      cerr.log   dShell.c         execvp.c    myshell.c
 backup.c       dFile      exampleShell     ' lsFile '
 brandNewFile   dShell     exampleShell.c   makefile
execvp [1] failed
Symbol '&' prevents the execution of  pwd
Pappas_8391>
```

Pipe

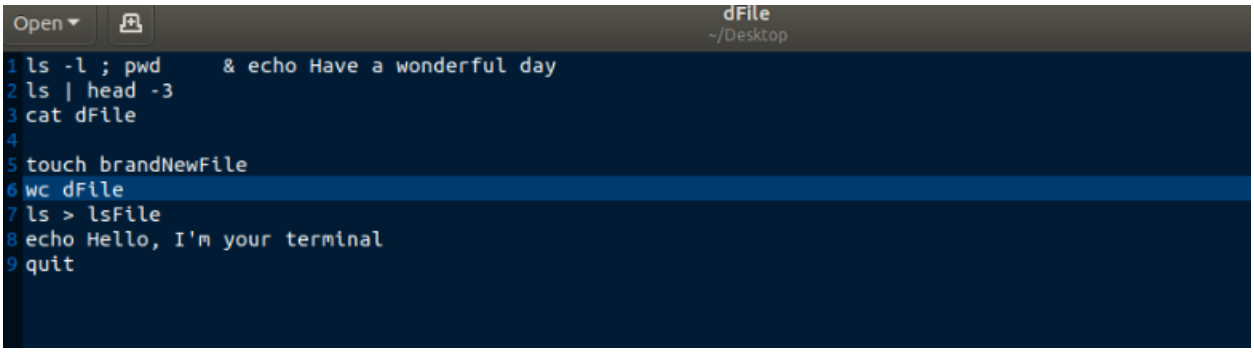
```
Pappas_8391> ls | head -4
backup2.c
backup.c
brandNewFile
cerr.log
Pappas_8391>
```

Redirection

```
Pappas_8391> pwd > pwdFile
Pappas_8391>
```



File



```
****************************************


        ****  D-SHELL  ****

****************************************


USER is: @dimitris
total 116
-rw-r--r-- 1 dimitris dimitris 10970 Jan 27 15:32  backup2.c
-rw-r--r-- 1 dimitris dimitris 10969 Jan 30 11:13  backup.c
-rw------- 1 dimitris dimitris     0 Feb  5 10:03  cerr.log
-rw-r--r-- 1 dimitris dimitris   143 Jan 30 17:33  dFile
-rwxr-xr-x 1 dimitris dimitris 18480 Feb  5 11:58  dShell
-rw-r--r-- 1 dimitris dimitris 13821 Feb  5 11:36  dShell.c
-rwxr-xr-x 1 dimitris dimitris 13568 Jan 27 12:09  exampleShell
-rw-r--r-- 1 dimitris dimitris  5471 Jan 20 13:42  exampleShell.c
-rw-r--r-- 1 dimitris dimitris   636 Jan 20 15:05  execvp.c
-rw------- 1 dimitris dimitris   232 Feb  5 10:04 ' lsFile '
-rw-r--r-- 1 dimitris dimitris    89 Jan 30 16:23  makefile
-rw-r--r-- 1 dimitris dimitris 13821 Feb  5 11:36  myshell.c
/home/dimitris/Desktop
Have a wonderful day
backup2.c
backup.c
cerr.log
ls -l ; pwd     & echo Have a wonderful day
ls | head -3
cat dFile

touch brandNewFile
wc dFile
ls > lsFile
echo Hello, I'm your terminal
quit
  9  29 143 dFile
Hello, I'm your terminal

Goodbye!
Do NOT come back (>_<)
dimitris@ubuntu:~/Desktop$
```

Space handling


Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>
Pappas_8391>