ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ - ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Λειτουργικά Συστήματα — Άσκηση 2

Εργασία Φοιτητή:
Δαμιανός Ντούμη - Σιγάλας, 6157
nsigalas@ceid.upatras.gr

Υλοποίηση των mysh1 και mysh2

Ο κώδικας που αφορά το `mysh1` και το `mysh2` είναι κοινός καθώς το δεύτερο αποτελεί προεκτασή της πρώτης περίπτωσης. Επομένως τα δύο αρχεία (`mysh1.c, mysh2.c`) έχουν πανομοιότυπο περιεχόμενο. Η λογική που ακολουθήθηκε περιγράφεται παρακάτω.

Αρχικά, εμφανίζεται το prompt στην οθόνη (`$ `) αναμένοντας την είσοδο του χρήστη, η οποία αποτελείται από το όνομα του ζητούμενου προγράμματος ή εντολής και πιθανόν διαφορές παραμέτρους που εξειδικεύουν την λειτουργία του. Έπειτα η είσοδος χωρίζεται σε κομμάτια με βάση τον κενό χαρακτήρα, δημιουργώντας έτσι έναν πίνακα αλφαριθμητικών σε κάθε θέση του οποίου έχουμε και από ένα μέρος της εντολής.

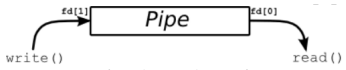
Στη συνέχεια ελέγχεται αν η είσοδος αποτελεί κάποιο built-in command του shell (στην περίπτωσή μας `exit` και `cd`) ή είναι απαραίτητη η κλήση κάποιου άλλου προγράμματος ώστε να πραγματοποιηθούν οι κατάλληλες ενέργειες και να εκτελεστεί το αντίστοιχο πρόγραμμα.

Στην περίπτωση που ο χρήστης επιθυμεί την εκτέλεση ενός τρίτου προγράμματος τότε με χρήση της fork η διεργασία γονέας (δηλαδή το shell μας) δημιουργεί και αναμένει την εκτέλεση μιας διεργασίας παιδιού η οποία αναλαμβάνει να εκτελέσει ένα στιγμιότυπο του ζητούμενο προγράμματος (με χρήση του system call `execvp`).

Το πρόβλημα που αντιμετώπισα σε αυτό το σημείο αφορά την σωστή διαχείριση της εισόδου του χρήστη καθώς το μέγεθος και ο αριθμός των παραμέτρων μιας εντολής δεν είναι γνωστός. Επομένως απαιτείται η δυναμική εκχώρηση μνήμης σχετικά με την δέσμευση ενος array of strings, σε κάθε θέση του οποίου αποθηκεύεται και ένα μέρος του user input, χωρισμένο με βάση τον κενό χαρακτήρα. Η λογική αυτή υλοποιείται στη συνάρτηση `char** parse_command();` η οποία αναλαμβάνει να φέρει το αρχικό αλφαριθμητικό σε μορφή κατάλληλη για παράμετρο της `execvp`.

Υλοποίηση των mysh3 και mysh4

Σε αυτά τα δύο shells προστίθεται η υποστήριξη του piping . Όταν μεταξύ δύο εντολών παρεμβάλλεται ένας vertical bar χαρακτήρας τότε η εντολή που βρίσκεται στα αριστερά ανακατευθύνει την έξοδο της στο άκρο εγγραφής ενός pipe ώστε η δεξια εντολή να ανακατευθύνει την είσοδο της στο άκρο ανάγνωσης του ίδιου pipe λαμβάνοντας τα δεδομένα που έγραψε σε αυτό η πρώτη.



`leftmost command | (...) | rightmost command`

Αφού χωρίσουμε την είσοδο με βάση τα vertical bars, κάθε token που προκύπτει πρέπει να χωριστεί με βάση το κενό και να έρθει σε μορφή κατάλληλη για την `execvp`. Στην περίπτωση του `mysh3` που γνωρίζουμε εκ των προτέρων ότι θα έχουμε μόνο δύο εντολές (επομένως είναι σταθερός ο αριθμός των εντολών και άγνωστο το πλήθος των παραμέτρων κάθε εντολής) η διαδικασία είναι παρόμοια με αυτή που περιγράφεται στα πρώτα δύο shells. Στο `mysh4` το πλήθος των pipes που αναμένουμε δεν είναι γνωστό. Έτσι πέρα από το άγνωστο πλήθος παραμέτρων κάθε εντολής το πρόγραμμα θα πρέπει να διαχειρίζεται και άγνωστο πλήθος εντολών. Για τον σκοπό αυτό δημιουργείται ένας πίνακας τύπου `char***` κάθε θέση του οποίου αποθηκεύει δείκτες σε array of strings (`char**`) καθένα από τα οποία περιέχει μία εντολή με την μορφή που περιγράφεται παραπάνω.

Για παράδειγμα με είσοδο `"ls -l -a | sort | wc -w"` ο πίνακας θα έχει την μορφή:

```
char** cmd[3] = { {"ls", "-l", "-a", NULL}, {"sort", NULL}, {"wc", "-w", NULL} }
```

Έπειτα εκτελείται η κάθε εντολή ξεχωριστά (το πλήθος των εντολών ισούται με τον αριθμό των pipes που διαβάστηκαν προσαυξημένο κατά ένα) φροντίζοντας να γίνουν οι κατάλληλες ανακατευθύνσεις των file descriptors κάθε διεργασίας ώστε να διαβάζει και να γράφει χρησιμοποιώντας τα δύο άκρα του pipe . Η πρώτη διεργασία διαβάζει απ΄ το `stdin` ενω η τελευταία γράφει στο `stdout`. Η διαδικασία αυτή επιτυγχάνεται με την χρήση του system call `dup2`.

Υλοποίηση του mysh5

Επιπρόσθετα με τις παραπάνω δυνατότητες το `mysh5` θα πρέπει να υποστηρίζει περισσότερες απο μία εντολές χωρισμένες με τον χαρακτήρα του semicolon, οι οποίες θα εκτελούνται σειριακά. Για να υλοποιηθεί αυτό το χαρακτηριστικό, η είσοδος του χρήστη υπόκειται σε επιπλέον επεξεργασία η οποία εξασφαλίζει τον σωστό κατακερματισμό της (συγχώνευση κενών χαρακτήρων και διαγραφή τους όπου χρειάζεται). Τα tokens που προκύπτουν, τροφοδοτούνται στον κώδικα που υλοποιεί το τέταρτο shell καθώς από το σημείο αυτό και μετά η διαδικασία είναι πανομοιότυπη.

Για παράδειγμα η είσοδος `"du -sh .; cat myfile.txt | sort | uniq; df -h"` θα χωριστεί στα εξής tokens:

1. `du -sh .`
2. `cat myfile.txt | sort | uniq`
3. `df -h`

Τα οποία οδηγώντας σειριακά τον κώδικα του τέταρτου shell, θα μετασχηματιστούν στην μορφή:

1. `{ "du", "-sh", ".", NULL }`
2. `{ {"cat", "myfile.txt", NULL}, {"sort", NULL}, {"uniq", NULL} }`
3. `{ "df", "-h", NULL }`

Παρατήρηση: Κάθε επόμενο shell ικανοποιεί τις απαιτήσεις του προηγούμενου του. Πράγματι το feedback που λαμβάνεται με την υποβολή ίδιου κώδικα για όλα τα ερωτήματα στον αυτόματο διορθωτή το επιβεβαιώνει. Ο λόγος που επέλεξα την ξεχωριστή παράδοση τους είναι ότι γίνεται εμφανής η σειρά που υλοποιήθηκαν οι ζητούμενες λειτουργίες κάνοντας φανερή την λογική μετάβαση από το ένα στο άλλο.