ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ 1

ΑΣΚΗΣΗ 2

Κωνσταντίνος Μπουραντάς
bourantas@ceid.upatras.gr
ΑΜ: 23 6145

~

Νοέμβριος 2016

- ***Στην συγκεκριμένη υλοποίηση του shell δόθηκε μεγάλη έμφαση στον ορθό και έξυπνο χειρισμό των εισόδων που δέχεται το πρόγραμμα από τον χρήστη . Αρχικά , με την εισαγωγή εντολών στο shell το πρόγραμμα μας κάνει πάντοτε κάποιες συγκεκριμένες ενέργειες ενημερώνοντας ορισμένες global μεταβλητές και καλώντας συγκεκριμένες συναρτήσεις.***

- read_line(): Πρώτα το πρόγραμμα μας καλεί την συνάρτηση read_line() η οποία διαβάζει την γραμμή που εισάγει ο χρήστης και την επιστρέφει στο κύριο πρόγραμμα . Επιπλέον κάνει μια σειρά από άλλες λειτουργίες οι οποίες θα βοηθήσουν στην μετέπειτα ροή του προγράμματος. Αρχικά η συνάρτηση μας διατρέχει όλους τους χαρακτήρες της συγκεκριμένης γραμμής και εντοπίζει τους χαρακτήρες που υποδηλώνουν ότι υπάρχει pipe στην εντολή και ενημερώνει το ανάλογο counter για το πόσα pipes περιέχει η εντολή.

- Έπειτα σκαναρει ξανά την γραμμή και μετράει τις λέξεις που περιέχονται για κάθε εντολή στην περίπτωση που έχουμε pipe μεταξύ των εντολών και τις αποθηκεύει σε έναν πίνακα τύπου int input_numb[] για κάθε εντολή. Με αυτόν τον τρόπο το πρόγραμμα εισάγει για κάθε εντολή σύμφωνα με την σειρά που έχει στην εντολή του shell πόσους παραμέτρους έχει . Δηλαδή :

  πχ.

  cmd: 0 1 2
  ls -l | grep mysh | sort

  input_numb[0]=1
  input_numb[1]=2 κλπ.

- Επίσης ενημερώνει το πρόγραμμα μας για το πόσες συνολικά εντολές & παραμέτρους έχει η κάθε γραμμή που εισάγεται από τον χρήστη. Η παραπάνω υλοποίηση έγινε έτσι ώστε να λυθούν διάφορα προβλήματα που προέκυψαν όπως η είσοδος εντολών χωρίς κενά και διάφορων άλλων όπως θα δούμε στην συνεχεία.

- split_line() : Μέτα από την read_line() σειρά έχει η split_line() ο οποία έχει ως ρόλο να δημιουργήσει έναν πινάκα ο οποίος θα περιεχέι όλες τις εντολές και παραμέτρους που εισάγει κάθε φορά ο χρήστης. Με την χρήση της συνάρτηση strtok() της C καταφέρνουμε το επιθυμητό αποτέλεσμα. Τέλος η συνάρτηση βάζει στην τελεφταια θέση του πίνακα την τιμή NULL έτσι ώστε ο πίνακα να είναι συμβατός με τις απαιτήσεις της συνάρτηση execvp όπως θα δούμε παρακάτω.

- Τέλος σειρά έχει η συνάρτηση η οποία καλείται από το πρόγραμμα για την εκτέλεση των εντολών που εισήχθησαν, η launch_pipe , η οποία καλείται μόνο στην περίπτωση που έχουμε pipe ανάμεσα των εντολών. Διαφορετικά καλείται η συνάρτηση launch. Η συνάρτηση launch δημιουργήθηκε στα προηγούμενα ερωτήματα της άσκησης για αυτό επέλεξα να την κρατήσω . Κανονικά η συνάρτηση launch_pipe μπορεί να κληθεί και για μια μόνο εντολή όμως αυτό θα άλλαζε το συνοχή της υλοποίησης.

- Αρχικά η launch_pipe δημιουργεί όσα pipes εισήγαγε ο χρήστης με σε μια επανάληψη for. Αώου το πρόγραμμα μας δέσμευση όλη την απαραίτητη μνήμη την οποία χρειάζεται προχωρά

σε μια άλλη επανάληψη for στην οποία θα εκτελεστούν τελικά οι εντολές. Σε κάθε επανάληψη αλλάζουμε το περιεχόμενο του πίνακα exec_input τον οποίο εισάγουμε στην εντολή exec_cvp. Για να ξέρουμε κάθε στιγμή πια εντολή θα φορτώσουμε στον exec_input και πόσους παραμέτρους χρησιμοποιήσουμε ως δείκτη για τον πίνακα που περιεχέι όλες τις εισόδου του χρήστη που δημιουργεί η split_line , την μεταβλητή index η οποία αρχικά έχει τιμή 0 και έπειτα την τιμή της seen_rags η οποία αυξάνεται προσθέτοντας τον αριθμό των παραμέτρων κάθε εντολής και τον εαυτό της.

Πχ .

0 1 2 3 4
Ls -l | grep sh | sort

ls : input_numb[0] = 2
index =0

while input_numb : index++; exec_inputs[i]= command[index]

index ← index + input_numb[0]

index = 2 : οπου υπάρχει η επόμενη εντολή , grep. Κλπ.

- Στο συγκεκριμένο σημείο το πρόγραμμα έχει bug το οποίο αφορούσε το memory allocation και την επανεγγραφή στις ίδιες θέσεις μνήμης για τον δείκτη exec_input . Το συγκεκριμένο bug διορθώθηκε βγάζοντας το memory allocation έξω από την επανάληψη while.(β. Κώδικα)

- Αφού δημιουργήσουμε την είσοδο της εντολής execvp κάνουμε fork() για να δημιουργήσουμε ενα καινουργικαινούργιο process έτσι ώστε να εκτελεστή η κάθε εντολή . Για κάθε εντολή δημιουργούμε ένα καινούργιο process μέσα στην επανάληψη. Στην συνεχεία εκτελούμε την εντολή dup2(pipes_fd[j – 2]) εάν η εντολή μας δεν είναι η πρώτη στην σειρά για να διαβάσει από τ pipe της προηγούμενης εντολής. Διαφορετικά εάν είναι η πρώτη γραφεί στο ανάλογο pipe. Αφού η κάθε εντολή διαβάσει ότι πρέπει κλείνουμε όλα τα ανοιχτά file descriptors και στην συνεχεία εκτελούμε την εντολή execvp για την εκτέλεση της κάθε εντολής. Τείος κλείνουμε όλα τα pipes και το γονικό process περιμένει για όλα του τα παιδιά να τελειώσουν να εκτελούνται.

- Η συνάρτηση launch αποτελεί μια πολύ πιο απλή υλοποίηση της launch_pipe στην οποία έγινε χρίση της switch() για απλούστερη υλοποίηση.

- Στην main συνάρτηση του προγράμματος εκτελούμε ένα loop το οποίο δεν τελειώνει παραμονή όταν ο χρήστης πληκτρολόγηση “exit” . Βλέπουμε μια σειρά από ifs τα οποία χρησιμοποιούνται για τις διακριτές λειτουργίες του shell . Η εντολή cd δεν εκτελείται απευθείας, πρέπει να κάνουμε κλίση της συνάρτησης change_dir. Στα τελευταία ifs βλέπουμε τον διαχωρισμό για την κλίση των συναρτήσεων εκτέλεσης των εντολών οποί όπως προαναφέραμε θα μπορούσαμε να έχουμε μόνο την μια από της δυο.