ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ ΤΜΗΜΑ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

7ο ΕΞΑΜΗΝΟ

Όνομα : Ιωάννης-Παναγιώτης Μπουντουρίδης
Α.Ε.Μ. : 8872

ΘΕΣΣΑΛΟΝΙΚΗ 2018

Ανάλυση Κώδικα

Προτού ξεκινήσουμε την εκτέλεση των εντολών του shell είναι σημαντικό να διακρίνουμε την επιλογή λειτουργίας του χρήστη (interactive ή batch mode) κατά την εκτέλεση. Για αυτό τον σκοπό ελέγχεται η τιμή της μεταβλητής argc. Όταν αυτή είναι ίση με την μονάδα εκτελείται ο κώδικας της λειτουργίας interactive ενώ όταν είναι 2 εκτελείται ο κώδικας για το κομμάτι της υλοποίησης batch mode.

Στο κομμάτι του interactive mode μέσα σε μια while αποθηκεύουμε αυτό που εισάγει ο χρήστης σε έναν πίνακα *cmd μέσω της εντολής fgets και στην συνέχεια καλούμε την command_execution() με όρισμα τον πίνακα cmd που περιέχει την εντολή προς εκτέλεση. Η while() τερματίζεται όταν ο χρήστης εισάγει quit. Η void promptMessage() τυπώνει σωστά το "mpountouridis_8872>" πριν εισάγουμε την πρώτη εντολή καθώς επίσης και μετά από κάθε ολοκλήρωση εντολής που εισάγει ο χρήστης.

Για την λειτουργία του batch mode αρχικά ανοίγουμε το αρχείο μέσω της εντολής fopen και διαβάζουμε ένα ένα τα γράμματα που περιέχονται σε αυτό. Το όνομα του αρχείου υπάρχει αποθηκευμένο στην μεταβλητή argv[1].Το σύνολο των γραμμάτων που υπάρχουν στο αρχείο αποθηκεύονται σε μια μεταβλητή int, την count_steps. Κατά την εκτέλεση της εντολής fgetc()!=EOF μέσα σε μια while φάνηκε να δημιουργείται ένας ατέρμονος βρόγχος όταν η εντολή fork εκτελούνταν εντός του βρόγχου αυτού. Για να μην έχουμε όμως ατέρμονους βρόχους κάνουμε μια καταμέτρηση των γραμμάτων του αρχείου και αποθηκεύουμε το αποτέλεσμα στην μεταβλητή count_steps. Στην συνέχεια κλείνουμε και ξανά ανοίγουμε το αρχείο αυτό Τώρα σε έναν πεπερασμένο βρόγχο for που εκτελείται τόσες φορές όσες μετρήσαμε παραπάνω, δηλαδή όσο μας επιτρέπει η count_steps. Με αυτό τον τρόπο λύθηκε το bug που παρατηρήθηκε να εκτελεί σε μια άπειρη λούπα τις ίδιες εντολές ασταμάτητα. Εντός του πεπερασμένου βρόγχου for γίνονται πολλαπλοί έλεγχοι. Καταρχάς αποθηκεύουμε κάθε χαρακτήρα σε μια μεταβλητή χαρακτήρων c. Αν η μεταβλητή c εντοπίσει αλλαγή γραμμής τότε σε έναν πίνακα χαρακτήρων *cmd_ αποθηκεύεται η εντολή μέχρι εκείνο το σημείο που διαβάστηκε από το αρχείο. Όμοια με την interactive mode η void command_excecution() με όρισμα τον *cmd_ κάνει οτι χρειάζεται για να εκτελεστούν σωστά οι εντολές που του δίνονται.

Στην περίπτωση που το αρχείο δεν περιέχει quit το πρόγραμμα ενημερώνει τον χρήστη πως η εντολή quit δεν βρέθηκε και τερματίζει.

Παραπάνω βλέπουμε πως και στις δύο περιπτώσεις την εκτέλεση των εντολών που εισάγει ο χρήστης την αναλαμβάνει η void command_execution(). Αναφέρθηκε ήδη πως το μοναδικό όρισμα της είναι ο πίνακας χαρακτήρων cmd τα οποία τα αντιγράφουμε σε έναν πίνακα temp. Πάνω σε αυτόν πειραματιζόμαστε και τον σπάμε σε κομμάτια με βάσει τα κενά. Μετράμε τις φορές που τον σπάσαμε σε μια μεταβλητή counter_key. Στην συνέχεια αν αυτή η μεταβλητή είναι μεγαλύτερη του μηδενός (δηλαδή αν δεν δόθηκε κενή εντολή) μέσα σε έναν πίνακα parts αποθηκεύουμε τον νέο "σπασμένο" πινάκα με βάσει τα κενά. Επειδή τα κενά δεν εξασφαλίζουν τον σωστό διαχωρισμό των εντολών πρέπει να ξεχωρίσουμε από τις εντολές τυχόν σκουπίδια που θα μας δυσκολεύσουν στην εκτέλεση τους. Για παράδειγμα, αν ο χρήστης εισάγει ls; ls. Τότε ο πίνακας parts θα περιέχει την πρώτη εντολή ls με ένα κολλημένο ερωτηματικό. Για να αποθηκεύονταν λοιπόν σωστά η εντολή θα έπρεπε να σπάγαμε τον πίνακα όχι μόνο ως προς τα κενά άλλα και ως προς τους χαρακτήρες (";" "&&" ">" "<"). Ελέγχουμε αν βρέθηκαν στον πίνακα parts ελληνικά ερωτηματικά εντός των εντολών του. Σε περίπτωση που βρεθούν αποθηκεύουμε στον δισδιάστατο πίνακα b τις εντολές χωρίς τα σκουπίδια που δεν μπόρεσε ο διαχωρισμός των κενών να εκπληρώσει. Αντίστοιχα για τον πίνακα b ελέγχουμε αν υπάρχουν εντός του σκουπίδια τύπου &&. Αν βρεθούν τότε αποθηκεύονται με την σωστή μορφή στον δισδιάστατο πίνακα d. Με παρόμοιο τρόπο συνεχίζεται αυτή η διαδικασία για τις περιπτώσεις ">" "<" και "|". Έχουμε πλέον στην διάθεση μας την εντολή στην επιθυμητή μορφή σε έναν δισδιάστατο πίνακα command_end και το μέγεθος του αποθηκευμένο στην μεταβλητή int array_max. Σε μια for για i από 0 έως array_max διαβάζουμε μια μια τις εντολές του από τον δισδιάστατο πίνακα command_end. Για την σωστή ανάγνωση των εντολών χρησιμοποιούμε τις ακέραιες μεταβλητές end και start οι οποίες έχουν αρχικά την τιμή 0. Σε περίπτωση που βρούμε τον χαρακτήρα ";" κατά την ανάγνωση της εντολής σταματάμε την ανάγνωση και εκτελούμε την εντολή που διαβάσαμε μέχρι εκείνο το σημείο. Προφανώς ενημερώνουμε της μεταβλητές start και end για την σωστή ανάγνωση της επόμενης εντολής. Αποθηκεύουμε αυτή την εντολή σε έναν νέο πίνακα και καλούμε την fork για την δημιουργία ενός νέου process. Το process αυτό καλεί την execvp και στην περίπτωση η εντολή είχε κάποιο λάθος τυπώνουμε error. Όμοια

κατα την ανάγνωση της εντολής από τον δισδιάστατο πίνακα command_end, στην περίπτωση που βρούμε τους χαρακτήρες "&&" εκτελούμε την εντολή μέχρι το σημείο που διαβάσαμε και ενημερώνουμε τις μεταβλητές end και start για την σωστή ανάγνωση της επόμενης εντολής. Με την προϋπόθεση ότι η εντολή που εκτελέστηκε προηγουμένως ήταν έγκυρη εκτελούμε την επόμενη. Ξανά αποθηκεύουμε αυτή την εντολή σε έναν νέο πίνακα και καλούμε την fork για την δημιουργία ενός νέου process. Το process αυτό καλεί την execvp και στην περίπτωση η εντολή είχε κάποιο λάθος τυπώνουμε error. Στην περίπτωση που βρούμε τον χαρακτήρα ">" κατά την ανάγνωση της εντολής από τον δισδιάστατο πίνακα command_end δημιουργούμε ένα νέο process το οποίο ανοίγει ένα αρχείο που έδωσε ο χρήστης (στην περίπτωση που δεν υπάρχει το δημιουργεί). Μέσω της dup2 υλοποιείται σωστά η αυτοκατεύθυνση εξόδου έτσι ώστε το αποτέλεσμα της εντολής πριν του χαρακτήρα ">" να αποθηκεύεται στο αρχείο μετά του χαρακτήρα ">" προφανώς και σε αυτή την περίπτωση ενημερώνουμε τις μεταβλητές end και start για την σωστή ανάγνωση της επόμενης εντολής. Τέλος στην περίπτωση που εντοπίζουμε τον χαρακτήρα "<" κατά την ανάγνωση της εντολής από τον δισδιάστατο πίνακα command_end δημιουργούμε ένα νέο process το οποίο υλοποιεί αντίστοιχα την αυτοκατεύθυνση εισόδου. Στην περίπτωση που δε βρεθεί κάποιο από τους παραπάνω χαρακτήρες και έχουμε μια μονό απλή εντολή τότε προφανώς ελέγχουμε αν η for θα εκτελεστεί για τελευταία φορά και βάσει των start και end αποθηκεύουμε την εντολή σωστά και την εκτελούμε. Στην σωστή λειτουργία της void command_execution() βοήθησε αρκετά και η συνάρτηση int count(). Η συνάρτηση αυτή έχει ως ορίσματα τον δισδιάστατο πίνακα εντολών cmd, το μέγεθος του int size, την περίπτωση ελέγχου: (π.χ. αυτή του ελληνικού ερωτηματικού ";") και την θέση του δισδιάστατου πίνακα εντολών στην οποία θα γίνει ο έλεγχος. Η συνάρτηση int count() επίσης ελέγχει αν οι εντολές περιέχουν τους ειδικούς χαρακτήρες που αναφέραμε παραπάνω με λάθος συντακτικό τρόπο (π.χ. η εντολή: ;ls ή &&ls).