ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ 2017 – 2018

Περιγραφή Προβλήματος

Η παρούσα εργασία αφορά την κατασκευή ενός προγράμματος τύπου terminal, το οποίο θα υποστηρίζει δύο διαφορετικές λειτουργίες: interactive & batch. Στην πρώτη ο χρήστης εισάγει τις εντολές που θέλει να εκτελέσει διαδοχικά σε περιβάλλον κονσόλας, ενώ στη δεύτερη τις έχει ήδη καταγράψει σε κάποιο script αρχείο (π.χ. sh file) και περνάει το path του σαν όρισμα στο πρόγραμμα. Κατά το interactive mode το πρόγραμμα τερματίζει με την εντολή quit, ενώ κατά την batch είτε με quit, είτε απλά με το τέλος του αρχείου.

Ανάλυση Προγράμματος

Με μια πρώτη σκέψη μπορούμε να πούμε ότι η batch λειτουργία είναι όμοια με την interactive με την διαφορά ότι στην πρώτη η εισαγωγή των εντολών γίνεται από αρχείο, ενώ στη δεύτερη από το terminal. Έτσι, ο κώδικας εκτέλεσης εντολών (ανά σειρά) είναι κοινός και κάθε φορά αλλάζουμε το input stream σύμφωνα με τα ορίσματα που πήρε το εκτελέσιμό μας.

Ξεκινώντας λοιπόν την εκτέλεση, το κέλυφός μας αρχικοποιεί το buffer εισόδου ανά γραμμή εντολών με μέγεθος BUF_SIZE = 512 χαρακτήρες. Αμέσως μετά ακολουθεί ο ορισμός όλων των μεταβλητών που θα χρειασθούν παρακάτω.

Πρώτα πρέπει να προσδιορισθεί ο τύπος λειτουργίας του κελύφους. Έτσι, αν έχουμε ακριβώς ένα όρισμα κατά την εκτέλεσή του (το όνομα του εκτελέσιμου του κελύφους δηλαδή), τότε σημαίνει πως ο χρήστης δεν έχει προσδιορίσει κάποιο batch file οπότε μπαίνουμε σε interactive mode και επιλέγουμε την stdin για ροή εισόδου. Αντιθέτως, εάν ο χρήστης χρησιμοποιήσει ένα αρχείο με εντολές, μπαίνουμε στην batch λειτουργία οπότε και ανοίγεται το αρχείο αυτό για ανάγνωση. Αν απ’ την άλλη ο χρήστης εισάγει

περισσότερα ορίσματα, αυτό είναι σφάλμα και εμφανίζεται αντίστοιχο μήνυμα error στην stderr.

Στη συνέχεια εκκινείται ο βασικός βρόχος του κελύφους κατά τον οποίο πραγματοποιείται όλη η λειτουργικότητά του. Πιο συγκεκριμένα, ο βρόχος αυτός εκτελείται για όσο η συνάρτηση fgets βρίσκει διαθέσιμους χαρακτήρες στην επιλεγμένη ροή για να διαβάσει. Αν είμαστε στην batch mode, τότε με το πέρας του αρχείου τερματίζεται η while loop, ενώ προφανώς δεν προκύπτει κάτι τέτοιο στην interactive mode, αφού η fgets πάντα περιμένει την εισαγωγή κειμένου από τον χρήστη.

Η πρώτη εργασία που λαμβάνει χώρα για κάθε σειρά εντολών είναι ο διαχωρισμός των εντολών σε περίπτωση που συνυπάρχουν περισσότερες της μίας σε μία σειρά. Η ιδέα είναι ότι αρχικά ξεχωρίζουμε τις εντολές στα semi colons (;) και έπειτα για κάθε ένα από τα υποσύνολα εντολών που προκύπτουν, διαχωρίζουμε πιθανά διπλά ampersands (&&). Για να επιτευχθούν τα παραπάνω χρειαζόμαστε ένα array of pointers to char, το οποίο θα περιλαμβάνει όλες τις διευθύνσεις έναρξης των tokens που προέκυψαν από τον διαχωρισμό semi colons. Το μέγεθος του πίνακα αυτού είναι αρχικά ίσο με 10, ενώ αν δεν είναι αρκετό εκχωρούμε δυναμικά επιπλέον 10 θέσεις τη φορά. Η δημιουργία του semicolonCommands κρίνεται απαραίτητη, καθώς η χρησιμοποιούμενη για το tokenization εντολή strtok λειτουργεί με static σε αυτή μεταβλητές κι έτσι δεν μπορεί να καλείται διαδοχικά για διαφορετικά strings.

Αφού ολοκληρωθεί ο διαχωρισμός των υποσυνόλων εντολών στα semi colons (πρώτος βρόχος do – while), τότε για κάθε ένα από αυτά διαχωρίζουμε τις εντολές του σε πιθανά διπλά ampersands (&&). Επομένως, και πάλι εντός ενός βρόχου do – while για κάθε εντολή πλέον, αρχικά κάνουμε trim τα starting και trailing spaces, ώστε η εντολή να συμφωνεί ακριβώς με το όνομα του αρχείου της στα linux. Στο σημείο αυτό γίνεται και ο έλεγχος για την εντολή “quit”, η οποία εάν βρεθεί πρέπει να διακοπεί η ροή του προγράμματος. Αν έχουμε μια οποιαδήποτε άλλη εντολή όμως και αυτή δεν είναι άδεια (ampersandCommand[0] == ‘\0’ : μη λανθασμένη περίπτωση όπου ο χρήστης εισάγει new line character), τότε ξεκινάει η διαδικασία εκτέλεσης της τρέχουσας εντολής.

Για την κάθε εντολή λοιπόν, πρώτα τη διαχωρίζουμε στα κενά, ώστε να περάσουμε το όνομά της και όλα της τα ορίσματα που την ακολουθούν σε ξεχωριστές παραμέτρους. Όλα αυτά αποθηκεύονται στον ίδιο πίνακα argv που χρησιμοποιεί και η συνάρτηση main χωρίς βλάβη της γενικότητας και περνούν στη συνάρτηση execvp μαζί με την προς εκτέλεση εντολή.

Αν η εκτέλεση της εντολής ήταν επιτυχής, η execvp θα “σκοτώσει” τη διεργασία που βρισκόμαστε (child) χωρίς να εκτελεστεί το υπόλοιπο block του if, ενώ αν κάτι πάει στραβά θα επιστρέψει την τιμή -1 στη μεταβλητή ret, θα εμφανιστεί το σφάλμα μέσω της συνάρτησης perror και έπειτα “σκοτώνεται” η child process. Σε κάθε περίπτωση ο πατέρας περιμένει το παιδί να τελειώσει για τη συνέχιση του προγράμματος.

Τέλος, υπολογίζεται η επόμενη εντολή και μόνο στην περίπτωση που η προηγούμενή της εκτελέστηκε επιτυχώς συνεχίζουμε με τον υπολογισμό των υπολοίπων μετά τα && για το τρέχον υποσύνολο semi colon. Μόλις δε εκτελεστούν όλες οι εντολές της σειράς, εκτυπώνεται το spyrou_8583> αναμένοντας νέα είσοδο από τον χρήστη.

Σημείωση

Η εργασία εκπονήθηκε σε συνεργασία με τον Γεώργιο Τσουμαλή με ΑΕΜ 8510.