ΓΕΩΡΓΙΟΣ ΚΥΡΙΑΚΟΠΟΥΛΟΣ / 1115201900092

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ ΕΡΓΑΣΙΑ 1 README

A)**Γενική Περιγραφή Κώδικα**
Έχω υλοποιήσει την εργασία σε C++ καθώς με διευκόλυνε αρκετά η χρήση vectors και κλάσεων. Η εργασία μου αποτελείται από τα αρχεία:

-main.cpp στο οποίο βρίσκεται η main συνάρτηση μου και ένα ατέρμονο loop που σπάει μόνο όταν ο χρήστης πατήσει exit. Το input κάθε command το παίρνω μέσω της συνάρτησης fgets() αποθηκεύοντάς το σε έναν char inputBuffer[50] τον οποίο έπειτα θα κάνω parse.
-shell.hpp, όπου βρίσκονται τα declarations των συναρτήσεων που υλοποιώ και χρησιμοποιώ
-shell.cpp, οπου βρίσκονται τα definitions των συναρτήσεων μου

B)**Αναλυτικές παραδοχές κάθε υπο-ερωτήματος**

-Αρχικά κάνω κάποιους ελέγχους στη main, όπως το αν ο χρήστης πληκτρολόγησε συγκεκριμένα commands(έλεγχος με strcmp()), όπως exit, myHistory, myHistory xx όπου xx ένας ακέραιος και τέλος cd. Αυτές είναι περιπτώσεις που διαχειρίζεται ξεχωριστά ο κώδικάς μου.

-Έπειτα αποθηκεύω την εντολή στο vector myHistory. Καλώ έπειτα τη συνάρτηση normalizeInput(), με την οποία μετατρέπω το inputBuffer σε string, σπάω τις λέξεις που βρίσκονται ανάμεσα σε κενά μέσω stream (istringstream), καθώς και αποθηκεύω κάθε εντολή ξεχωριστά σε ένα inputVector. Έπειτα αφού λάβω το inputVector με τα strings καλώ σε αυτό την:
-traverseCommand(inputVector), η οποία εξετάζει σειριακά κάθε command/string του inputVector και ελέγχει αν υπάρχουν ειδικά σύμβολα όπως >, <, >>, |, &, έτσι ώστε να καλέσει τις αντίστοιχες συναρτήσεις για κάθε ένα από αυτά. Π.χ. όταν βρίσκει το σύμβολο < καλείται η συνάρτηση redirectInput().
-Τέλος καλώ τη συνάρτηση executeCommand() στην οποία διαχειρίζομαι aliases, μετατρέπω το string που περιέχει όλο το command του χρήστη σε char * array για να μπορεί να κληθεί σε αυτό η execvp(). Κάνω fork(), και αν βρίσκομαι στο παιδί απλά καλώ την execvp() στην εντολή, αλλιώς αν βρίσκομαι στον πατέρα ελέγχω για signals και κάνω wait το παιδί, καθώς και redirect τα Input/Output πίσω στο terminal για να πάρει πάλι τον έλεγχο ο χρήστης μετά από την εκτέλεση μίας εντολής (εκτός και αν αυτή είναι στο background).

a.1) **IO Redirection**
Έχω υλοποιήσει τα functions redirectInput(), redirectOutput() και appendOutput() αντίστοιχα.

Όλα λειτουργούν με παρόμοιο τρόπο δηλαδή παίρνουν ένα string , π.χ. inputFileStr, και το μετατρέπουν σε char *. Καλούν σε αυτό την open() και αν πέτυχε κάνουν dup2() στον καινούριο fd που μόλις δημιουργήθηκε από την open, τα IO Streams π.χ. STDIN_FILENO. Μετά κάνω close() τον καινούριο fd μιας και δεν τον χρειαζόμαστε επιπλέον και επιστρέφω. Μία παραδοχή που έχω κάνει σε αυτές τις συναρτήσεις είναι να ελέγχω αν το αρχείο που δόθηκε για redirection σε αυτό είναι το terminal (δηλ. το "/dev/tty").

b) **Pipes**
Μέσω του function implementPipe(outputCommand, inputCommand) φτιάχνω ένα pipe() και κάνω dup2() και close() για το κάθε ένα αντίστοιχα ανάλογα με το αν θέλουμε το output η το input αυτού του command. Τέλος καλώ για κάθε ένα την executeCommand() μέσω της οποίας εκτελώ το κάθε command.

c) **Background Commands**
Αν βρω & στην εντολή τρέχοντας traverseCommand() τότε μέσω ενός Boolean που περνάω στο executeCommand(), το bool background_process, ελέγχω αν το process που εκτελώ στο executeCommand() πρέπει να εκτελεστεί στο bg και αν ναι, απλά δεν εκτελώ καθόλου τον κώδικα του πατέρα(signal, waitpid, redirect IO).

d) **Wild Characters**

Πριν εκτελέσω μία εντολή στην συνάρτηση executeCommand() ελέγχω αν καθένα από τα commands περιέχει τα σύμβολα *,?,[] και αν ναι καλώ την συνάρτηση glob() στην εντολή αυτή.

e) **Aliases**

Έχω μία κλάση Alias (στο shell.hpp), η οποία έχει ως πεδία της την originalCommand καθώς και την aliasedCommand. Κάθε φορά που ο χρήστης φτιάξει ένα alias, φτιάχνω καινούριο Alias αντικείμενο και το κάνω push back σε ένα vector vector<Alias *> aliasesVector. Έτσι για κάθε εντολή που πληκτρολογείτε εξετάζω αν έχουν γραφεί τα keywords createalias ή destroyalias και αν ναι καλώ τις συναρτήσεις createalias() η destroyalias(), η οποίες κάνουν push back/ remove στο aliasesVector. Επίσης κάθε εντολή που γράφεται εξετάζω αν μέσα της υπάρχει κάποιο alias με το να διατρέχω το aliasesVector και αν υπάρξει κάποιο aliasedCommand τότε απλά επιστρέφω το originalCommand για να εκτελεστεί.

f) **Signals**

Μέσω των συναρτήσεων signal(SIGINT, handleSigint) για control ^C, καθώς και της signal(SIGTSTP, handleSigtstp), ελέγχω κάθε φορά στην executeCommand() και συγκεκριμένα μετά το fork() στον κώδικα του πατέρα και πριν κάνει waitpid το παιδί που εκτέλεσε την εκάστοτε εντολή ελέγχω αν έχει σταλεί κάποιο από τα δύο αυτά signals κατά τη διάρκεια εκτέλεσης της εντολής.

g) **MyHistory**

Μέσω της κλάσης MyHistory η οποία έχει ως πεδία ένα vector commandsVector στο οποίο αποθηκεύω κάθε εντολή που εκτελείται από τον χρήστη (αφού πρώτα τη μετατρέψω σε string), καθώς και το πεδίο int currentFreeIndex μέσω του οποίου ξέρω πόσο γεμάτο είναι το vector. Αν πάω να προσθέσω εντολή μέσα στο vector και αυτό έχει ήδη 20 εντολές μέσα, τότε πηγαίνω όλες τις εντολές μία θέση προς τα πίσω, διαγράφοντας την παλαιότερη χρονικά εντολή, και προσθέτοντας στο commandsVector[19] την καινούρια εντολή.