ΣΤΑΥΡΙΝΑΚΗΣ ΠΑΝΑΓΙΩΤΗΣ
A.M.: 236217
email: stavrinakis@ceid.upatras.gr

Τα τρία πρώτα ερωτήματα είναι λυμένα, το 1 και 2 λειτουργούν σωστά ενώ το 3 όχι. Έπειτα από έρευνα για το πώς να δημιουργήσω τη προσωπική μου μινιμαλιστική C κατάλαβα ότι απαιτούνται τρία πράγματα: εισαγωγή εντολών, διαχωρισμός λέξεων και τέλος εκτέλεση εντολών. Χωρίς την χρήση συναρτήσεων ο κώδικας θα γινόταν πολύπλοκος μπορεί και ακατόρθωτος. Για την εισαγωγή εντολών χρησιμοποίησα το fgets μιας και δέχεται παραπάνω από έναν χαρακτήρες και στο τέλος του string τοποθετεί NULL οπότε δεν χρειαζόταν να προσθέσω '\n'. Το parsing προτίμησα να το κάνω με τη χρήση pointers χωρίς να χρειάζεται να βρίσκω κάθε φορά το μέγεθος του string και πέρασα τις εντολές στο argv τοποθετώντας στο τέλος NULL για να μην παρουσιάσει το execvp: bad address. Για την εκτέλεση των εντολών χρησιμοποίησα την execvp() μιας και δέχεται έναν πίνακα από pointers που τερματίζουν με null και παρουσιάζουν τη λίστα με τα arguments στο νέο πρόγραμμα. Προτίμησα για την wait() το WIFEXITED(status) και το WIFSIGNALED(status) σε περίπτωση που το child process τερμάτιζε κανονικά ή με την εισαγωγή ενός signal. Το πρόβλημα που αντιμετώπιζα ήταν στην περίπτωση που δεν έβαζα input επέστρεφε Segmentation fault οπότε πρόσθεσα έναν έλεγχο μόλις δεχόταν input αν ήταν κενό να επιστρέφει στην εισαγωγή. Αν και ο grader διαφωνούσε σε αυτό προτίμησα να το αφήσω έτσι(με το seg fault δεν έβγαζε κανένα πρόβλημα).
Επίσης, για να μην τερματίζει ο κώδικας πρόσθεσα ένα ασταμάτητο loop μέχρι να δοθεί συγκεκριμένη εντολή exit για να τερματίσει. Το execvp δεν θα αναγνώριζε τη λέξη exit σαν εντολή του INIT που τρέχει από την αρχή από ότι κατάλαβα αλλά σαν απλό string οπότε με compare συγκρίνω το πρώτο argument με το exit.
Για το δεύτερο ερώτημα δεν έχω να πω πολλά, αν το πρώτο argument είναι το cd τότε να εκτελείται το chdir() που δέχεται το επόμενο argument που αντιστοιχεί στο directory και να αλλάζει από το προηγούμενο. Επίσης, ήθελε δυναμική κατανομή για τα arguments οπότε χρησιμοποίησα malloc() για ανακατανομή και στο τέλος μια free() για απελευθέρωση μνήμης. Τώρα, όσον αφορά το τρίτο ερώτημα. Δεν κατάφερα να το κάνω να τρέξει ούτε γνωρίζω αν η λογική που χρησιμοποιώ είναι λάθος ή υπάρχει κάτι που έχω κατανοήσει λανθασμένα. Όμως, επειδή προσπάθησα υπέθεσα έπρεπε να το στείλω.. Λοιπόν, μιας και το pipe δέχεται το output μιας εντολής ως input της επόμενης σκέφτηκα είναι καλή ιδέα στο σημείο που υπάρχει το pipeline να χωρίσω το string σε δύο πίνακες, ο ένας θα περιέχει τις εντολές πριν το pipeline ενώ ο δεύτερος τις εντολές μετά. Δεν κατάφερα να κάνω το πρόγραμμα να διαβάζει έναν array of pointers για να τα διαχωρίσω, ή δεν επέστρεφε τίποτα ή τα επέστρεφε (null), οπότε έπρεπε να κάνω διαφορετικό parse άμα δεχόταν pipeline με strsep(). Επίσης, στο τέλος του πρώτου πίνακα έπρεπε πάλι να προσθέσω NULL για να μην γυρίσει bad address το execvp(). Θέμα είχα με το να τερματίζει ο κώδικας μόλις επέστρεφε το error, το πρόβλημα ήταν ότι έπρεπε να βάλω δεύτερη λειτουργία fork() για να εκτελείται σωστά επειδή αυτή τη φορά έχουμε δυο processes. Χρησιμοποιώ pipe() που περιέχει array με 2 integers για να δημιουργήσω

2 file descriptors ώστε να χρησιμοποιηθούν για το pipeline. Το στοιχείο 0 είναι για να δέχεται δεδομένα(read) και το 1 για να στέλνει δεδομένα(write), εάν ο γονιός δέχεται δεδομένα πρέπει να κλείσει το 1 και το παιδί το 0, αλλιώς το αντίθετο. Χρησιμοποιώ την dup2() για να διπλασιάσω τους descriptors και παράλληλα να τους κλείσω, ώστε να εκτελεστεί η execvp(). Με αυτόν τον τρόπο δεν θα τερματιστεί από κάποιο signal. Έχω αφήσει και μια εκτύπωση των πινάκων μέσα για να τσεκάρω αν δεν δέχτηκε κάποιο command ή λείπει το NULL στο τέλος.