Λειτουργικά Συστήματα

1η Εργασία

Μέλη ομάδας:

- Γεώργιος-Κρίτων Γεωργίου(p3150020)
- Αθανάσιος Μέμτσας(p3150103)

Mysh1

Σε έναν ατέρμονο βρόχο (while(1)) εκτελούμε το φλοιό μας, διαβάζοντας συνεχώς από το χρήστη για ποια εντολή να εκτελεστεί, χρησιμοποιώντας τη μέθοδο scanf. Επειδή η εντολή που θα εκτελούμε από το φλοιό mysh1 θα είναι μια μεμονωμένη χωρίς arguments, αρκεί να διαβάσουμε ένα απλό string. Εάν η scanf διαβάσει End of file από το χρήστη (συνδυασμός Ctrl-D), θα τερματίσει και όλος ο φλοιός εμφανίζοντας κατάλληλο μήνυμα στο χρήστη.

Αφού διαβάσουμε το όνομα της εντολής, χρησιμοποιούμε fork() για να γεννήσουμε μια νέα διεργασία. Στην διεργασία-παιδί θα εκτελέσουμε την εντολή με τη χρήση της execlp, ενώ στην διεργασία-πατέρα θα περιμένουμε το παιδί να τερματίσει, χρησιμοποιώντας την waitpid. Αυτό το κάνουμε για να μην αφήνουμε διεργασίες παιδιά ως ζόμπι.

Mysh2

Η διαφορά με το προηγούμενο φλοιό είναι πως θα έχουμε εντολές με ορίσματα, και πρέπει να διαβάζουμε από το χρήστη ολόκληρη γραμμή. Χρησιμοποιήσαμε τη getline() για να διαβάσουμε όλη την εντολή. Έπειτα, στη διεργασία – παιδί χρησιμοποιούμε την μέθοδο getArguments() που δημιουργήσαμε στο κοινό αρχείο. Η μέθοδος αυτή διαχωρίζει την εντολή σε κάθε whitespace με τη χρήση της strtok(), και επιστρέφει σε ένα πίνακα από strings τα επιμέρους τμήματα της εντολής. Ο πίνακας αυτός έχει NULL στην τελευταία του θέση. Τον πίνακα αυτό τον χρησιμοποιούμε στην μέθοδο execvp() για να εκτελέσουμε την εντολή στη νέα διεργασία. Όπως και προηγουμένως, η διεργασία-πατέρας θα περιμένει να τερματίσει το παιδί πριν προχωρήσει στην κανονική λειτουργία του φλοιού και να διαβάσει νέα εντολή από το χρήστη.

Mysh3

Για λόγους ευκολίας σε αυτό το φλοιό και τους επόμενους, δημιουργήσαμε μια κοινή μέθοδο που ονομάζεται spawn_process. Δέχεται ως ορίσματα 2 ακεραίους, που αντιστοιχούν στους file descriptors των αρχείων από το οποίο θα διαβάζει η εντολή και το πού θα εμφανίζει την έξοδο, και το string μιας εντολής που έδωσε ο χρήστης. Να σημειωθεί

πως στο mysh3 χρησιμοποιούμε μόνο τα default 0 και 1 αντίστοιχα για την κλήση της, (stdin και stdout), ενώ χρησιμοποιείται πλήρως στους επόμενους φλοιούς.

Στην spawn_process, αναλαμβάνουμε τη δημιουργία νέας διεργασίας με fork(), και την αναμονή από τον πατέρα. Στην διεργασία παιδί, ανακατευθύνουμε αν χρειάζεται την είσοδο και την έξοδο με τη χρήση της dup2(), και αναλύουμε με την getArguments() όπως πριν την εντολή μας. Για κάθε argument που μας επιστράφηκε, ελέγχουμε αν είναι ένα εκ των '>', '>>', '<'. Εάν ταυτίζεται με ένα από αυτά, ανοίγουμε το επόμενο argument (θα είναι αρχείο) με κατάλληλο τρόπο. Για '>', το ανοίγουμε για εγγραφή και καθαρίζουμε το περιεχόμενό του με το option O_TRUNC. Για '>>', το ανοίγουμε για εγγραφή αλλά ορίζουμε το option O_APPEND για να γράψουμε στο τέλος του, χωρίς να σβήσουμε το περιεχόμενό του. Για '<', το ανοίγουμε για εγγραφή. Στις δύο πρώτες περιπτώσεις, κάνουμε redirect το stdout στο αρχείο που ανοίξαμε, ενώ στην τελευταία το stdin. Τα υπόλοιπα ορίσματα τα περνάμε σε ένα νέο πίνακα από strings που θα είναι η εντολή που θα εκτελέσουμε ακριβώς, και με execvp την εκτελούμε.

Mysh4

Στο νέο φλοιό, αφού διαβάσουμε την εντολή, την χωρίζουμε με την κοινή μέθοδο splitPipes() στις επιμέρους εντολές. Η splitPipes είναι ανάλογη της getArguments, με τη διαφορά ότι χωρίζει το string στην εμφάνιση του '|' αντί για το whitespace. Έτσι παίρνουμε τις ξεχωριστές εντολές. Θα χρησιμοποιήσουμε ένα πίνακα ακεραίων 2 θέσεων για την διασωλήνωση, και με την pipe() θα τη δημιουργήσουμε, για να πάρουμε την έξοδο κάθε εντολής. Έτσι θα κάνουμε spawn_process(in, fd[1], commands[0]) για την πρώτη, και spawn_process(fd[0], out, commands[1]) για τη δεύτερη εντολή. Επαναφέρουμε τα file descriptors των stdin και stdout στα αρχικά τους, και συνεχίζουμε την επανάληψη.

Mysh5

Τέλος, ο φλοιός mysh5 αποτελεί άμεση επέκταση του mysh4, επιτρέποντας τη σωλήνωση περισσότερων εντολών, και αρκεί πάλι μια διασωλήνωση, όπου ενώνουμε την έξοδο της μιας εντολής με την είσοδο της επόμενης.