Σκοπός

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με την δημιουργία διεργασιών και τη διαχείριση αρχείων και σημάτων, γράφοντας ένα απλό κέλυφος mysh (my simple shell) σε περιβάλλουν Linux και στη γλώσσα προγραμματισμού C (ή C++).

Διατύπωση του προβλήματος

Το κέλυφος θα πρέπει να δίνει τη δυνατότητα στον χρήστη να δημιουργεί και να χειριζεται νέες διεργασίες και να δίνει την ευκαιρία στο χρήστη να εκτελεί οποιοδήποτε πρόγραμμα συστήματος είναι ήδη διαθέσιμο.

Η κλίση του κελύφους είναι απλή και γίνεται ως εξής από την γραμμή κλήσης εντολών (prompt):

linux01.di.uoa.gr:> ./mysh
in-mysh-now:>

Αναμένεται ότι το mysh θα χρησιμοποιήσει οποιαδήποτε πληροφορία είναι διαθέσιμη σε μεταβλητές περιβάλλοντος ώστε να επιτευχθεί σωστή διερμήνευση εντολών από το λειτουργικό σύστημα.

Οι εντολές που εισάγονται στο κέλυφος, θα πρέπει να γίνονται parse από το πρόγραμμα και να μην καλούνται ως έχουν. Δεν γίνονται αποδεκτές υλοποιήσεις όπου οι εντολές κελύφους εκτελούνται ώς έχουν (π.χ. χρησιμοποιώντας την system())

Η διεπαφή και τα χαρακτηριστικά του κελύφους

Το mysh θα πρέπει να παρέχει :

- Υποστήριξη Ανακατευθύνσεων
- Υποστήριξη σωληνώσεων (pipes)
- Εκτέλεση εντολών στο Background
- Διαχείριση Σημάτων (Signals)
- Υποστήριξη wild characters
- Διαχείριση aliases

a)Υποστήριξη Ανακατευθύνσεων

a.1)Ανακατεύθυνση εισόδου / εξόδου (I/O redirection) (< , >):

Στη συνέχεια ακολουθούν παραδείγματα χρήσης:

in-mysh-now:> myProgram > out.txt

Η εντολή myProgram τρέχει και το αποτέλεσμα αποθηκεύεται στο αρχείο out.txt.

in-mysh-now:> myProgram < input.txt

Η εντολή myProgram τρέχει και παίρνει input από το input.txt.

in-mysh-now:> sort < file1 > file2

Τα περιεχόμενα του file1 ταξινομούνται και το αποτέλεσμα αποθηκεύεται στο file2.

Σημείωση: Η εντολή sort χρησιμοποιείται ως παράδειγμα. Θα πρέπει οι ανακατευθύνσεις να λειτουργούν για όλες τις εντολές του υποκείμενου κελύφους. Το ίδιο ισχύει και για τα υπόλοιπα παραδείγματα που ακολουθούν.

a.2)Ανακατεύθυνση προσθήκης σε υπάρχον αρχείο (>>)

Το αποτέλεσμα εκτέλεσης του αριστερού μέρους θα πρέπει να προσαρτάται σε ένα υπάρχον αρχείο π.χ

in-mysh-now:> cat file1 >> file2

b)Υποστήριξη σωληνώσεων (pipes)

Παροχή σωλήνωσης (pipes) και συνδυασμός τους με ανακατευθύνσεις, δηλ:

in-mysh-now:> cat file1 file2 file3 | sort > file4
τα περιεχόμενα των αρχείων file1 file2 file3 χρησιμοποιούνται σαν είσοδος στο πρόγραμμα sort και το αποτέλεσμα γράφεται στο αρχείο file4.

c)Εκτέλεση εντολών στο Background (&)

Δυνατότητα για εκτέλεση εντολών στο background με χρήση του τελεστή &:

in-mysh-now:> sort file1 &; ls &;

Όταν μια εντολή τρέχει στο background μπορεί και πρέπει να τρέχει ταυτόχρονα με την επόμενη προς εκτέλεση εντολή.

d)Υποστήριξη wild characters

Δυνατότητα χρήσης wild characters. Οι wild characters θα μπορούν να χρησιμοποιούνται για να ορίσουν ένα υποσύνολο των αρχείων του τρέχοντας καταλόγου. Για παράδειγμα η εντολή:

in-mysh-now:> ls file.t?t*

θα πρέπει να παρουσιάζει το υποσύνολο των αρχείων του τρέχοντος καταλόγου που έχουν πρόθεμα file και επέκταση που αρχίζει και τελειώνει με το χαρακτήρα t και ενδιάμεσα έχει έναν οποιοδήποτε χαρακτήρα.

e)Διαχείριση aliases

Δυνατότητα για δημιουργία/ καταστροφή aliases. Για παράδειγμα:

in-mysh-now:> createalias myhome "cd /home/users/smith";

δημιουργεί ένα alias το οποίο ονομάζεται myhome κι αν κανείς το γράψει στην γραμμή εντολής, έχει

ισοδύναμο αποτέλεσμα με εκείνο της εκτέλεσης ολόκληρης της εντολής (δηλ. cd /home/users/smith). Παρομοίως υπάρχει και η εντολή:

in-mysh-now:> destroyalias myhome;

που απενεργοποιεί το alias.

f)Διαχείριση Σημάτων (Signals)

Το κέλυφος σας θα πρέπει να μπορεί να διαχειρίζεται απλά signals. Πιο συγκεκριμένα θα πρέπει ο χρήστης να μπορεί να

- στείλει ένα control-C στη διεργασία που είναι σε εξέλιξη μέσω του prompt του mysh. Το αποτέλεσμα θα πρέπει να είναι ο τερματισμός του προγράμματος που τρέχει αλλά **ΟΧΙ** του κελύφους mysh.
- στείλει ένα control-Z στη διεργασία που είναι σε εξέλιξη. Το κέλυφος θα πρέπει επίσης να αγνοεί το σήμα control-Z και να το προωθεί στο πρόγραμμα που τρέχει.

g)Αποθήκευση ιστορικού (myHistory)

Δυνατότητα ιστορίας εντολών. Το κέλυφος θα πρέπει να μπορεί να θυμάται και να ανακαλεί με απλό τρόπο τις τελευταίες 20 εντολές χρήστη. Ο χρήστης θα πρέπει να έχει τη δυνατότητα να εκτελέσει κάποια από αυτές τις εντολές χωρίς να χρειάζεται να την επαναπληκτρολόγησει..

Διαδικαστικά

● Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++) και σας θα πρέπει να τρέχει στα Linux workstations του Τμήματος. Κώδικας που δε μεταγλωττίζεται εκεί, θεωρείται ότι δεν μεταγλωττίζεται.

● Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com.

Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2023/k24/home. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.

● Ο κώδικάς σας θα πρέπει να αποτελείται από τουλάχιστον δύο (και κατά προτίμηση περισσότερα) διαφορετικά αρχεία. Η χρήση του separate compilation είναι επιτακτική και ο κώδικάς σας θα πρέπει να έχει ένα Makefile.

● Βεβαιωθείτε πως ακολουθείτε καλές πρακτικές software engineering κατά την υλοποίηση της άσκησης. Η οργάνωση, η αναγνωσιμότητα και η ύπαρξη σχολίων στον κώδικα αποτελούν κομμάτι της βαθμολογίας σας.

● Η υποβολή θα γίνει μέσω eclass.

● Ο κώδικάς σας θα πρέπει να κάνει compile στο εκτελέσιμο mysh όπως **ακριβώς** ορίζει η άσκηση.

Τι πρέπει να παραδοθεί

● Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία OnomaEponymoProject1.tar.gz. Προσοχή να υποβάλετε μόνο κώδικα, Makefile, README και όχι τα binaries. Η άσκησή σας θα γίνει compile από την αρχή πριν βαθμολογηθεί.

● Όποιες σχεδιαστικές επιλογές κάνετε, θα πρέπει να τις περιγράψετε σε ένα README (απλό text

αρχείο) που θα υποβάλετε μαζί με τον κώδικά σας. Το README χρειάζεται να περιέχει μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στον σχεδιασμό του προγράμματός σας. 1-2 σελίδες ASCII κειμένου είναι αρκετές. Συμπεριλάβετε την εξήγηση και τις οδηγίες για το compilation και την εκτέλεση του προγράμματός σας.

● Ο κώδικας που θα υποβάλετε θα πρέπει να είναι δικός σας. Απαγορεύεται η χρήση κώδικα που δεν έχει γραφεί από εσάς ή κώδικας που έχει γραφτεί με τη βοήθεια μηχανών τύπου chatGPT.

● Καλό θα είναι να έχετε ένα backup .tar της άσκησής σας όπως ακριβώς αυτή υποβλήθηκε σε κάποιο εύκολα προσπελάσιμο μηχάνημα (server του τμήματος, github, cloud).

● Η σωστή υποβολή ενός σωστού tar.gz που περιέχει τον κώδικα της άσκησής σας και ό,τι αρχεία χρειάζονται είναι αποκλειστικά ευθύνη σας. **Άδεια tar/tar.gz ή tar/tar.gz που έχουν λάθος και δε γίνονται extract δε βαθμολογούνται**.

Τι θα βαθμολογηθεί

● Η συμμόρφωση του κώδικά σας με τις προδιαγραφές της άσκησης.
● Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα.
● Η χρήση Makefile και το separate compilation.

Άλλες σημαντικές παρατηρήσεις

● Οι εργασίες είναι ατομικές.
● Όποιος υποβάλλει / δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο **μηδενίζεται** στο μάθημα.
● Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πώς θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιασδήποτε μορφής) είναι κάτι που δεν επιτρέπεται. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα απλά παίρνει μηδέν στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ. Τονίζουμε πως θα πρέπει να λάβετε τα κατάλληλα μέτρα ώστε να είναι προστατευμένος ο κώδικάς σας και να μην αποθηκεύεται κάπου που να έχει πρόσβαση άλλος χρήστης (π.χ., η δικαιολογία «Το είχα βάλει σε ένα github repo και μάλλον μου το πήρε από εκεί», δεν είναι δεκτή.)
● Οι ασκήσεις προγραμματισμού μπορούν να δοθούν με καθυστέρηση το πολύ 3 ημερών και με ποινή 5% για κάθε μέρα αργοπορίας. Πέραν των 3 αυτών ημερών, δεν μπορούν να κατατεθούν ασκήσεις.
● Το πρόγραμμά σας θα πρέπει να γραφτεί σε C ή C++. Μπορείτε να χρησιμοποιήσετε μόνο εντολές οι οποίες είναι διαθέσιμες στα μηχανήματα Linux του τμήματος. Πρόγραμμα που πιθανόν μεταγλωτίζεται ή εκτελείται στο προσωπικό σας υπολογιστή αλλά όχι στα μηχανήματα Linux του τμήματος, θεωρείται ότι δε μεταγλωτίζεται ή εκτελείται αντίστοιχα.