

**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΗΛΕΚΤΡΟΝΙΚΗΣ

**Σχεδιασμός Ενσωματωμένων Συστημάτων**

9ο εξάμηνο

2η Εργαστηριακή Άσκηση:

**Εργασία σε assembly του επεξεργαστή ARM**

ΑΜ: 03116112

Ονοματεπώνυμο: Γεζεκελιάν Βικέν

**Ερώτημα 1ο : Μετατροπή εισόδου από τερματικό**

Για το συγκεκριμένο ερώτημα, έχουμε δημιουργήσει 3 αρχεία. Τα 2 από αυτά είναι γραμμένα σε assembly ARM, ενώ το τρίτο είναι το Makefile για το link των αρχείων και τη δημιουργία του executable. Εκ των δύο assembly αρχείων, το ex\_3\_1.s περιέχει πρακτικά την main συνάρτηση του προγράμματος μας ενώ το ex\_3\_1\_func.s περιέχει την συνάρτηση μετατροπής χαρακτήρων που ζητείται. Το πρόγραμμα είναι συνεχούς λειτουργίας με έξοδο μόνο με εισαγωγή του q ή Q σε γραμμή.

Για το διάβασμα της εισόδου, τον χειρισμό των strings και την εμφάνιση της εξόδους έχουμε κάνει χρήση συναρτήσεων της C. Επίσης, εφόσον δοθεί είσοδος περισσότερων από 32 χαρακτήρες το πρόγραμμα τους αγνοεί. Εφόσον στην εκφώνηση αναφέρεται πως η συμβολοσειρά της εισόδου πρέπει να έχει μήκος 1 για να τερματίσει το πρόγραμμα, όσο δεν έχουμε συμβολοσειρά μήκους 1 καλούμε την συνάρτηση μετατροπής. Σε περίπτωση που το μήκος της συμβολοσειράς όντως ισούται με 1, ελέγχουμε πρώτα για ισότητα με q ή Q, και έπειτα καλούμε τη συνάρτηση μετατροπής. Όσον αφορά τη λειτουργία της συνάρτησης μετατροπής, έχουμε δημιουργήσει μια κύρια συνάρτηση, η οποία αναλόγως με την τιμή του χαρακτήρα σε ascii τον «εντάσσει» στη σωστή κατηγορία, ώστε να ξέρει πως να τον χειριστεί. Ο κώδικας λοιπόν, απλώς συγκρίνει την τιμή του χαρακτήρα της εισόδου με ορισμένες τιμές δοσμένες σε αύξουσα σειρά, ώστε να ξέρει που να τον εντάξει. Αφού γίνει η μετατροπή, η διαδικασία ξεκινάει εκ νέου έως ότου το length του εναπομένοντος string να είναι ίσο με 0, οπότε και το πρόγραμμα επιστρέφει στην κύρια συνάρτηση.

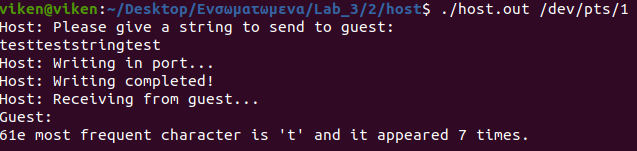
**Ερώτημα 2ο : Επικοινωνία των guest και host μηχανημάτων μέσω σειριακής θύρας**

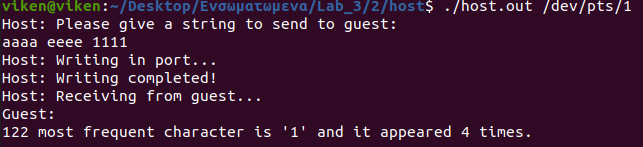
Για το συγκεκριμένο ερώτημα, χρειάστηκε να φτιάξουμε δύο αρχεία κώδικα(καθώς και τα αντίστοιχα Makefile). Ένα από τα δύο αρχεία, είναι υπεύθυνο για τη λειτουργία στο guest μηχάνημα, δηλαδή το vm που φτιάξαμε μέσω qemu, ενώ το άλλο είναι υπεύθυνο για τη λειτουργία στον host, δηλαδή τον υπολογιστή μας. Αρχικά, αναφέρουμε πως η επικοινωνία από τον host στον guest γίνεται μέσω ενός εκ των /dev/pts/x, όπου x το όνομα του, που δημιουργείται κατά το τρέξιμο του qemu με όρισμα -serial pty. Από την πλευρά του guest, η επικοινωνία γίνεται μέσω του /dev/ttyAMA0, στο οποίο έχουμε φροντίσει να δώσουμε πλήρη δικαιώματα.

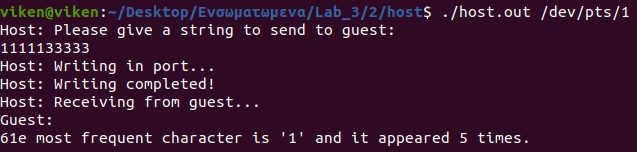
Όσον αφορά τον κώδικα host.c, αρχικά θέλουμε να τρέξει με argument το target αρχείο, καθώς αυτό αλλάζει κάθε φορά που ανοίγουμε το qemu. Έπειτα, θέλουμε μέσω chmod να δώσουμε πλήρη permissions για το device που θα χρησιμοποιηθεί, ώστε να μην έχουμε προβλήματα με τα read και write, πράγμα που κάνουμε μέσω της system. Ο υπόλοιπος κώδικας είναι αρκετά απλός, καθώς χρησιμοποιούμε τις εντολές read και write για να γράψουμε και να διαβάσουμε από το device που χρησιμοποιούμε.

Όσον αφορά τον κώδικα guest.s, αρχικά χρειάστηκε να βρούμε τις σωστές τιμές για να θέσουμε στο termios struct που χρησιμοποιούμε, ώστε guest και host να μοιράζονται ίδιο configuration της θύρας. Όσον αφορά λοιπόν τα flags, προσθέσαμε τις τιμές των c\_flags που δίνονται στο αρχείο termios.h, και έπειτα τις μετατρέψαμε σε δεκαεξαδική μορφή. Όσον αφορά την τιμή του πίνακα c\_cc, από το αρχείο termios.h είδαμε πως το VMIN καταλαμβάνει την 6η θέση, ενώ η τιμή που του δώσαμε είναι 1. Άρα αντικαταστήσαμε το 5ο bit με την τιμή 01. Όσον αφορά τον υπόλοιπο κώδικα, έχουμε χρησιμοποιήσει μια συνάρτηση που ονομάσαμε strcmp για να δούμε πόσες φορές εμφανίζεται κάποιος χαρακτήρας εντός του string, και έπειτα κάνουμε τις κατάλληλες συγκρίσεις για να κρατήσουμε τον χαρακτήρα που πρέπει. Για τη συσκευή, χρησιμοποιούμε το path /dev/ttyAMA0, και αφού την ανοίξουμε αποθηκεύουμε κατάλληλα τον file descriptor. Έπειτα, χρησιμοποιούμε τις snprintf και swi σε συνδυασμό ώστε να γράψουμε στο port, κλείνουμε τη συσκευή και το πρόγραμμα κάνει exit.

Όπως φαίνεται και στις παρακάτω φωτογραφίες, ενώ το πρόγραμμα φαίνεται να δουλεύει σωστά, υπάρχουν κάποια bugs τα οποία δεν έχω προλάβει να διορθώσω:







**Ερώτημα 3ο : Σύνδεση κώδικα C με κώδικα assembly του επεξεργαστή ARM**

Για το τελευταίο ερώτημα, χρειάστηκε να δημιουργήσουμε 4 καινούργια αρχεία σε κώδικα assembly ARM, τα strcpy.s, strcmp.s, strlen.s και strcat.s, κάθε ένα από τα οποία είναι υπεύθυνο για την υλοποίηση της αντίστοιχης συνάρτησης του ονόματός του. Επίσης, χρειάστηκε να κάνουμε μερικές μικροαλλαγές στον κώδικα που δίνεται, ώστε να χρησιμοποιηθούν οι δικές μας συναρτήσεις αντί για της έτοιμες της βιβλιοθήκης string.h. Για τον λόγο αυτό λοιπόν, βάλαμε σε comment τη γραμμή #inlclude <string.h>, ενώ προσθέσαμε στον κώδικα τις γραμμές

extern int strlen(char \*s);

extern char \*strcpy(char \*dest, char \*src);

extern int strcmp(char \*s1, char \*s2);

extern char \*strcat(char \*dest, char \*src);

Τέλος, χρειάστηκε να δημιουργήσουμε και ένα Makefile, υπεύθυνο για τη σύνδεση των 5 αρχείων κώδικα. Όσον αφορά τώρα συγκεκριμένα την υλοποίηση των συναρτήσεων, έχουμε:

Για την strcat: Η συγκεκριμένη συνάρτηση πρέπει να μπορεί να ενώνει δύο strings. Στην υλοποίηση μας λοιπόν διαβάζουμε το destination string(1o argument της συνάρτησης) από τον r0 έως ότου φτάσουμε στο τέλος του, οπότε και διαγράφουμε το /0 που υπήρχε και εισάγουμε το καινούργιο string από τον r1.

Για την strcmp: Για τη συγκεκριμένη συνάρτηση, διαβάζουμε έναν χαρακτήρα από το κάθε string έως ότου ένα εκ των δύο τελειώσει, δηλαδή εμφανιστεί το 0. Στην περίπτωση αυτή, πρέπει να κάνουμε μια σύγκριση, εφόσον γνωρίζουμε πως τουλάχιστον ένα εκ των δύο strings έχει τελειώσει:

Εάν και τα δύο string έχουν τιμή 0, τότε είναι ίσα σε μήκος και επιστρέφουμε 0. Εάν κάποιο από τα δύο έχει μεγαλύτερη τιμή από το άλλο, τότε επιστρέφουμε κατάλληλα -1 ή 1 για να δηλώσουμε ποιο από τα δύο είναι αυτό.

Σημείωση: Σε ορισμένες περιπτώσεις, βρήκα πως το return της cmp δεν είναι απλώς αρνητικό ή θετικό για να δηλώσει ποιο από τα δύο string είναι το μεγαλύτερο, αλλά επιστρέφει και ακριβώς τη διαφορά σε μέγεθος, όμως καθώς κάτι τέτοιο δεν είναι απαραίτητο στην υλοποίηση της string\_manipulation.c δεν το συμπεριέλαβα.

Για την strcpy: Η συγκεκριμένα συνάρτηση είναι πολύ απλή, καθώς χρειάζεται απλώς να διαβάσουμε το source έως ότου βρούμε /0, και έπειτα αντιγράφουμε στον προορισμό και επιστρέφουμε τον r0.

Για την strlen: Και η συγκεκριμένη συνάρτηση είναι πολύ απλή στην υλοποίησή της, καθώς χρειάζεται απλώς να διαβάσουμε το string έως ότου βρούμε /0, κρατώντας παράλληλα έναν counter.