ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2013

1η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

ΟΔΗΓΟΣ ΑΣΥΡΜΑΤΟΥ ΔΙΚΤΥΟΥ ΑΙΣΘΗΤΗΡΩΝ ΣΤΟ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ LINUX

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΟΜΑΔΑ B04

ΣΑΡΛΗΣ ΔΗΜΗΤΡΙΟΣ 03109078

ΤΖΑΝΝΕΤΟΣ ΔΗΜΗΤΡΙΟΣ 03109010

ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ

Στόχος της άσκησης ήταν η υλοποίηση του LUNIX:TNG, ενός απλού οδηγού συσκευής για ένα ασύρματο δίκτυο αισθητήρων κάτω από το λειτουργικό σύστημα Linux. Συγκεκριμένα, το δίκτυο αποτελούνταν από ένα αριθμό από αισθητήρες και ένα σταθμό βάσης, ο οποίος συνδέεται μέσω USB με υπολογιστικό σύστημα Linux στο οποίο θα εκτελείται και ο οδηγός. Ο σταθμός βάσης λαμβάνει πακέτα με δεδομένα μετρήσεων τα οποία προωθεί στη συνέχεια στο υπολογιστικό σύστημα. Ο ρόλος του οδηγού είναι να προσφέρει κατάλληλο μηχανισμό, αντιμετωπίζοντας ανεξάρτητα τους αισθητήρες και τα μετρούμενα μεγέθη, να επιτρέπει ταυτόχρονη πρόσβαση στα εισερχόμενα δεδομένα και να κάνει δυνατή την επιβολή διαφορετικής πολιτικής από το διαχειριστή του συστήματος όσον αφορά την πρόσβαση σε αυτά.

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

Το σύστημα οργανώνεται σε δύο κύρια μέρη: το πρώτο αναλαμβάνει τη συλλογή δεδομένων από το σταθμό βάσης (lunix line discipline) και την επεξεργασία τους ακολουθώντας συγκεκριμένο πρωτόκολλο (lunix protocol), έτσι ώστε να εξαχθούν οι τιμές των μετρούμενων μεγεθών και να αποθηκευθούν σε ενδιάμεσους κατάλληλους buffers (lunix sensor buffers). Το δεύτερο αναλαμβάνει να παραλάβει τα δεδομένα από τους χώρους ενδιάμεσης αποθήκευσης και να τα εξάγει στο χώρο χρήστη σε κατάλληλη μορφή, υλοποιώντας έτσι μια σειρά από συσκευές χαρακτήρων. Αυτό είναι και το κύριο κομμάτι στο οποίο πρέπει να επέμβουμε υλοποιώντας τις συναρτήσεις που θα ικανοποιούν τις κλήσεις συστήματος που θα εκτελεί πάνω στις συσκευές μια διεργασία του χώρου χρήστη.

Το τελικό σύστημα θα είναι σε θέση να λειτουργεί ως εξής: τα δεδομένα των μετρήσεων αφού παραληφθούν από το σταθμό βάσης προωθούνται μέσω USB στο υπολογιστικό σύστημα. Στη συνέχεια αντί να ακολουθήσουν τη συνήθη πορεία τους θα παραλαμβάνονται από ένα φίλτρο, τη γραμμή διάταξης του lunix, η οποία θα τα προωθεί σε κατάλληλο στρώμα πρωτοκόλλου. Αυτό θα είναι υπεύθυνο για την ερμηνεία των bytes των πακέτων που παραλήφθηκαν και στη συνέχεια θα τα αποθηκεύει σε διαφορετικούς buffers ανά αισθητήρα. Η συσκευή χαρακτήρων ανάλογα με το ποιο αρχείο χρησιμοποιεί η εφαρμογή χρήστη, θα επιτρέπει την ανάκτηση των δεδομένων από τους buffers και θα τα παρουσιάζει με συγκεκριμένη μορφή.

ΣΤΑΔΙΑ ΥΛΟΠΟΙΗΣΗΣ

Τα αρχεία που προσθέσαμε τον δικό μας κώδικα είναι τα `lunix-chrdev.c` και `lunix-chrdev.h`. Εκτός από τα βασικά ζητούμενα της άσκησης, υλοποιήσαμε και τη συνάρτηση `lunix_chrdev_ioctl` προκειμένου ο χρήστης να μπορεί με μια κλήση

συστήματος `ioctl` να τροποιήσει τον τρόπο που θα επιστρέφονται τα δεδομένα, δηλαδή αν θα είναι “raw” ή “cooked”. Για το σκοπό αυτό προσθέσαμε στο αρχείο `lunix_chrdev.h` το πεδίο `int data_type` στη δομή για την κατάσταση ενός ανοιχτού αρχείου και κάποιες σταθερές για τη χρήση εντολών `ioctl`. Οι υπόλοιπες προσθήκες κώδικα επικεντρώθηκαν στο αρχείο `lunix-chrdev.c`.

Αρχικά υλοποιήσαμε τη συνάρτηση `lunix_chrdev_state_needs_refresh` ,η οποία ελέγχει αν τα τελευταία δεδομένα που παραλάβαμε από τους buffers είναι επίσης τα τελευταία που έχουμε αποθηκευμένα στη συσκευή και αν όχι τότε σημαίνει ότι πρέπει να ανανεωθούν γιατί υπάρχουν νέα δεδομένα.

Στη συνέχεια συμπληρώσαμε τη συνάρτηση `lunix_chrdev_state_update`. Σκοπός αυτής της συνάρτησης είναι να πάρει τα δεδομένα από τον buffer και αν υπάρχουν νέα δεδομένα τότε χρησιμοποιεί τους πίνακες που υπάρχουν στο αρχείο `lunix-lookup.h` για να μετατρέψει τα δεδομένα στη μορφή που επιθυμούμε. Αυτή η διαδικασία παρακάμπτεται στην περίπτωση που ο χρήστης έχει αλλάξει τη μορφή των δεδομένων που επιστρέφονται μέσω μιας κλήσης `ioctl`. Το σημαντικό στη συγκεκριμένη συνάρτηση είναι ότι όταν προσπαθούμε να πάρουμε το κλείδωμα για να μπορέσουμε να έχουμε πρόσβαση στα δεδομένα των buffers χρησιμοποιούμε τη συνάρτηση `spin_lock_irqsave()` (η οποία φροντίζει να απενεργοποιήσει τις διακοπές) και αντίστοιχα την `spin_unlock_irqrestore()`για να αποδεσμεύσουμε το κλείδωμα. Ο λόγος έχει να κάνει με το γεγονός ότι ο κώδικας αυτός που τρέχει σε process context ανταγωνίζεται με κώδικα που τρέχει σε interrupt context (συγκεκριμένα ο κώδικας που όταν διαμορφώσει τα δεδομένα που παρελήφθησαν από το σταθμό βάσης, χρησιμοποιώντας το καθορισμένο πρωτόκολλο, τα αποθηκεύει μετά στους buffers). Με αυτό τον τρόπο αποφεύγουμε πιθανή κατάσταση αδιεξόδου που θα μπορούσε να παρουσιαστεί στην περίπτωση που συνέβαινε μια διακοπή στον επεξεργαστή που τρέχει η διεργασία, ακριβώς μετά που αυτή έχει πάρει το κλείδωμα για την πρόσβαση στους buffers.

Οι επόμενες δύο συναρτήσεις αφορούν τις ενέργειες που γίνονται όταν ο χρήστης ανοίγει ένα αρχείο και όταν στη συνέχεια το κλείσει. Η συνάρτηση `lunix_chrdev_open` φροντίζει να συσχετίσει το ανοιχτό αρχείο με το σωστό αισθητήρα ανάλογα με το minor number του inode που παίρνει σαν όρισμα και να αρχικοποιήσει μια καινούρια δομή state της συσκευής με τις κατάλληλες τιμές. Αντίστοιχα, η συνάρτηση `lunix_chrdev_release` είναι υπεύθυνη για την αποδέσμευση του χώρου μνήμης που αφορούσε τη δομή αυτή.

Η συνάρτηση που υλοποιεί τη λειτουργία της κλήσης συστήματος `ioctl` είναι η `lunix_chrdev_ioctl`. Σε αυτή αρχικά ελέγχεται αν ο κωδικός της εντολής που δόθηκε αφορά όντως τη συγκεκριμένη συσκευή χαρακτήρων και επίσης αν είναι μία από τις έγκυρες που επιτρέπονται. Στη συγκεκριμένη υλοποίηση η μοναδική εντολή που επιτρέπεται είναι αυτή που αλλάζει τον τρόπο που μεταφέρονται τα δεδομένα στο χώρο χρήστη. Πιο συγκεκριμένα, αυτό που κάνουμε είναι αφού πρώτα πάρουμε το σημαφόρο που μας επιτρέπει να έχουμε πρόσβαση στη δομή state μιας συσκευής αντιστρέφουμε την τιμή του πεδίου `data_type` (από 0 σε 1 και ανάποδα).

Η σημαντικότερη συνάρτηση για την υλοποίηση αυτού του οδηγού είναι αυτή που χειρίζεται την περίπτωση της ανάγνωσης από το αρχείο, η `lunix_chrdev_read`. Η συνάρτηση αυτή λειτουργεί ως εξής: Αν ο δείκτης `f_pos` είναι στην αρχή σημαίνει ότι ακόμα δεν έχουμε διαβάσει τίποτα. Στην περίπτωση αυτή αν δεν υπάρχουν φρέσκα δεδομένα, φροντίζουμε να στείλουμε τη διεργασία στην ουρά αναμονής του συγκεκριμένου αισθητήρα όπου κοιμάται και όταν έρθουν καινούρια δεδομένα θα ξυπνήσει μαζί με όσες άλλες ακόμα περιμένουν στην ίδια ουρά. Στην περίπτωση λοιπόν που υπάρχουν δεδομένα για ανάγνωση (ή που ο δείκτης `f_pos` δεν ήταν στην αρχή του αρχείου) φροντίζουμε να στείλουμε στο χρήστη τόσα δεδομένα όσα ζήτησε ή αν ζήτησε παραπάνω από όσα υπάρχουν στέλνουμε όσα είναι δυνατόν. Σε κάθε περίπτωση προσέχουμε να χρησιμοποιήσουμε τη συνάρτηση `copy_to_user()` για τη μεταφορά των δεδομένων στον buffer που ζήτησε ο χρήστης για ασφαλή αντιγραφή των δεδομένων. Επίσης, αν ο δείκτης `f_pos` φτάσει την οριακή τιμή `state->buf_lim` θεωρούμε τον μηδενίζουμε ξανά για να αρχίσει πάλι από την αρχή η ανάγνωση του αρχείου. Φυσικά, όλες αυτές οι λειτουργίες επιτελούνται αφού έχουμε εξασφαλίσει την αποκλειστική πρόσβαση στη δομή του αρχείου κλειδώνοντας το σηματοφόρο.

Οι τελευταίες δύο συναρτήσεις που υλοποιήσαμε αφορούν τις ενέργειες που πραγματοποιούνται όταν γίνεται η εγκατάσταση ή απεγκατάσταση του οδηγού στον πυρήνα του Linux. Η συνάρτηση που καλείται κατά την εγκατάσταση είναι η `lunix_chrdev_init`, η οποία φροντίζει να αρχικοποιήσει κατάλληλα τις συσκευές δηλώνοντας στον πυρήνα τις συναρτήσεις που θα είναι υπεύθυνες για την εξυπηρέτηση τους, ζητάει από τον πυρήνα μια περιοχή minor numbers και στη συνέχεια προσθέτει τις επιθυμητές συσκευές. Από την άλλη η συνάρτηση `lunix_chrdev_destroy` κάνει την αντίστροφη διαδικασία διαγράφοντας τις συσκευές, ενώ αποδεσμεύει την περιοχή των minor numbers που χρησιμοποιούσαν ώστε να είναι δυνατό να χρησιμοποιηθούν από άλλες πιθανόν συσκευές.