Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Λειτουργικά Συστήματα (ΗΥ321)

Ακαδημαϊκό Έτος 2014-2015

1η Εργαστηριακή Άσκηση

Τελευταία Ενημέρωση: 6 Μαρτίου 2015

Περιεχόμενα

1 Εισαγωγικά **3**

1.1 Προαπαιτούμενα 3

1.2 Προθεσμία και Τρόπος Παράδοσης 3

1.3 Κώδικας Ηθικής 4

2 Ζητούμενα **4**

2.1 Πρώτο Μέρος: Δημιουργία μίας Νέας Κλήσης Συστήματος 4

2.2 Δεύτερο Μέρος: Τροποποίηση ενός Kernel Module 6

2.2.1 Διαδικασία Τροποποίησης 6

2.2.2 Έλεγχος Λειτουργίας του Kernel Module 7

3 Πακετάρισμα των Αλλαγών – Δημιουργία του προς Υποβολή Συνημμένου Αρχείου **8**

3.1 Απομόνωση των Αλλαγών στον Κώδικα του Λειτουργικού 8

3.2 Πακετάρισμα των Προς Υποβολή Στοιχείων σε Αρχείο 9

1 Εισαγωγικά

1.1 Προαπαιτούμενα

Πριν ξεκινήσετε την υλοποίηση θα πρέπει να έχετε διαβάσει τις οδηγίες που σας έχουν δοθεί, όπως επίσης τα κεφάλαια 1 και 2 και 5 από το βιβλίο "Linux Kernel Development" και το κεφάλαιο 2 από το βιβλίο "Linux Device Drivers". Επίσης θα πρέπει προφανώς να έχετε καταφέρει να εγκαταστήσετε σωστά το λειτουργικό Ubuntu και να μπορείτε να μετακινήσετε αρχεία μεταξύ του λειτουργικού συστήματος ξενιστή (host) και του Ubuntu που τρέχει στο πλαίσιο της ιδεατής μηχανής.

1.2 Προθεσμία και Τρόπος Παράδοσης

Η άσκηση θα πρέπει να παραδοθεί έως τα **μεσάνυχτα της Δευτέρας 16/3/2015**. Η προθεσμία είναι τελική και δεν πρόκειται να δοθεί καμία παράταση – συνολικά ή ατομικά – για κανένα λόγο. Ο χρόνος που σας δίνεται υπερεπαρκεί. Χρησιμοποιήστε τον "σοφά".

Η παράδοση θα γίνει με e-mail στο ce321lab@gmail.com. Ο τίτλος (θέμα) του μηνύματος θα πρέπει να περιλαμβάνει τα εξής στοιχεία:

CE321 Project1 Group <ΑΕΜ ομάδας κατ' αύξουσα σειρά>

Για παράδειγμα, το μήνυμα με την υποβολή της εργασίας της ομάδας με ΑΕΜ 456 123 789 θα έχει τίτλο CE321 Project1 Group 123 456 789

Ακολουθήστε **αυστηρά** το παραπάνω format. Θα είστε αποκλειστικά υπεύθυνοι σε περίπτωση που δεν ακολουθήσετε το ζητούμενο format και το μήνυμά σας χαθεί.

Στο σώμα του μηνύματος θα πρέπει να υπάρχουν τα ονοματεπώνυμα, τα Α.Ε.Μ. και τα e-mail των μελών της ομάδας. Επίσης μπορείτε να συμπεριλάβετε και οτιδήποτε άλλο μήνυμα θέλετε να μας μεταφέρετε σχετικά με την άσκησή σας. Προσοχή στο encoding! Βεβαιωθείτε, αν γράφετε με ελληνικούς χαρακτήρες, ότι έχετε στείλει το μήνυμα με encoding είτε unicode (UTF-8) είτε ISO 8859-7. Σε αντίθετη περίπτωση το μήνυμα δεν είναι δυνατό να διαβαστεί.

Τέλος, η εργασία σας θα πρέπει να συμπεριληφθεί στο μήνυμα ως ένα συνημμένο αρχείο. Για τις λεπτομέρειες δημιουργίας του αρχείου δείτε την παράγραφο 3.

Το πολύ 24 ώρες μετά τη λήξη της προθεσμίας υποβολής θα ανακοινωθεί κατάλογος των ομάδων που έχουν υποβάλλει εργασία. Σε περίπτωση που έχετε υποβάλλει εργασία και δεν βρίσκεστε στον

κατάλογο, επικοινωνήστε άμεσα με τον διδάσκοντα (Χ. Αντωνόπουλο) ή τους μεταπτυχιακούς που επικουρούν το μάθημα.

1.3 Κώδικας Ηθικής

Κάθε ομάδα θα πρέπει να εργαστεί ανεξάρτητα. Είναι δεκτό και θεμιτό διαφορετικές ομάδες να ανταλλάξουν απόψεις σε επίπεδο γενικής ιδέας ή αλγόριθμου. Απαγορεύεται όμως κάθε συζήτηση / ανταλλαγή κώδικα ή ψευδοκώδικα. Σημειώστε ότι οι ασκήσεις θα ελεγχθούν τόσο από αυτόματο σύστημα όσο και από εμάς για ομοιότητες μεταξύ των ομάδων αλλά και για ομοιότητες με τυχόν λύσεις που μπορεί να βρεθούν στο διαδίκτυο ή λύσεις που έχουν δοθεί σε προηγούμενα έτη. Σε περίπτωση αντιγραφής οι ασκήσεις (όλων των φάσεων) όλων των εμπλεκόμενων φετινών ομάδων μηδενίζονται χωρίς καμία περαιτέρω συζήτηση.

Όλα τα μέλη στο εσωτερικό κάθε ομάδας θα πρέπει να έχουν ισότιμη συμμετοχή στην ανάπτυξη κάθε φάσης. Διατηρούμε το δικαίωμα να επιβεβαιώσουμε αν αυτό τηρείται με προσωπικές συνεντεύξεις / προφορική εξέταση.

2 Ζητούμενα

2.1 Πρώτο Μέρος: Δημιουργία μίας Νέας Κλήσης Συστήματος

Ως ένα πρώτο, εισαγωγικό βήμα στη διαδικασία τροποποίησης του πυρήνα του Linux, καλείστε να προσθέσετε μία νέα κλήση συστήματος. Η κλήση συστήματος θα ονομάζεται *find_roots* και θα έχει το ακόλουθο πρότυπο:

```
long find_roots(void)
```

Η νέα κλήση συστήματος θα εκτυπώνει το αναγνωριστικό (process id) και το όνομα της διεργασίας που την κάλεσε. Έπειτα, θα εκτυπώνει το όνομα και το αναγνωριστικό για κάθε διεργασία-πρόγονο της καλούσας διεργασίας. Ένα ενδεικτικό στιγμιότυπο από την κλήση της *find_roots()* παρουσιάζεται στην εικόνα 1.

Στο λειτουργικό σύστημα Linux, πρόγονος όλων των διεργασιών είναι η διεργασία *init*, η οποία έχει process id ίσο με 1. Συνεπώς, ο βρόγχος εκτύπωσης θα πρέπει να τερματίζεται όταν βρεθεί η *init*.

Υλοποιήστε όλες τις απαραίτητες αλλαγές στον πυρήνα προκειμένου να προσθέσετε τη νέα κλήση. Γι' αυτή τη διαδικασία, μπορείτε να ακολουθήσετε τις οδηγίες του αντίστοιχου οδηγού που

find_roots system call called by process 1869
id: 1869, name: find_roots_lib
id: 1830, name: bash
id: 1824, name: konsole
id: 1, name: init

Εικόνα 1: Ενδεικτική έξοδος από την εκτέλεση της κλήσης συστήματος.

είναι διαθέσιμος στη σελίδα του εργαστηρίου. Στη συνέχεια μεταγλωττίστε και εγκαταστήστε τον νέο πυρήνα και επανεκκινήστε το σύστημά σας.

Το τελευταίο βήμα, μετά την επανεκκίνηση του συστήματος, είναι η ενεργοποίηση της νέας κλήσης συστήματος. Αυτό θα πραγματοποιηθεί μέσω μίας βιβλιοθήκης επιπέδου χρήστη, όπως περιγράφεται στον αντίστοιχο οδηγό και εφαρμογής που θα τη χρησιμοποιεί. Δημιουργήστε την wrapper library με όνομα *libroots.a* (ονομάστε τα σχετικά αρχεία κώδικα *roots.c* και *roots.h*) και ένα αρχείο προγράμματος με όνομα *find_roots_lib.c*, το οποίο θα χρησιμοποιεί την wrapper library για να πραγματοποιήσει την κλήση συστήματος. **Για ακόμη μία φορά υπενθυμίζεται ότι δεν θα πρέπει να αποθηκεύσετε τα αρχεία του κώδικα επιπέδου χρήστη στο source tree του πυρήνα. Μπορείτε να τα αποθηκεύσετε στο home directory σας.**

Παρακάτω, μπορείτε να βρείτε κάποια βασικά στοιχεία που θα σας διευκολύνουν στην υλοποίηση της ζητούμενης λειτουργικότητας:

- Το PCB (process control block) της κάθε διεργασίας είναι μία δομή τύπου task_struct. Αυτή ορίζεται στο αρχείο *include/linux/sched.h*. Θα πρέπει να μελετήσετε προσεκτικά τη δομή της και να προσθέσετε τη γραμμή

  ```
  #include <linux/sched.h>
  ```

  στο αρχείο με την υλοποίηση της κλήσης συστήματος.

- Ο δείκτης προς τη δομή *task_struct* της τρέχουσας διεργασίας, δηλαδή της διεργασίας που ενεργοποίησε το system call αποθηκεύεται στην global μεταβλητή με όνομα *current*.

- Το id της διεργασίας, το όνομα του εκτελέσιμου που αυτή εκτελεί και ο δείκτης προς τον "πατέρα" της αποτελούν και αυτά πεδία της δομής *task_struct*.

2.2 Δεύτερο Μέρος: Τροποποίηση ενός Kernel Module

Το δεύτερο μέρος της παρούσας εργασίας έχει ως στόχο να σας εξοικειώσει με τον κώδικα ενός kernel module και με τις διαδικασίες μεταγλώττισης και φόρτωσης-αφαίρεσης από τον πυρήνα. Ως παράδειγμα, θα χρησιμοποιηθεί ένα απλό module που χρησιμοποιείται από τον πυρήνα για δρομολόγηση Εισόδου/Εξόδου. Το module είναι ο *noop (no fancy operations)* I/O scheduler, ένας απλός δρομολογητής που εξυπηρετεί τις διάφορες αιτήσεις για Είσοδο/Εξοδο σε συσκευές βάσει του αλγορίθμου *First Come - First Served*.

2.2.1 Διαδικασία Τροποποίησης

Δημιουργήστε στο home directory σας έναν φάκελο με όνομα project1_module. Αντιγράψτε σε αυτόν το αρχείο *block/noop-iosched.c* και μετονομάστε το σε *project1-iosched.c*. Στη συνέχεια θα προχωρήσετε σε μερικές απλές τροποποιήσεις του κώδικα, οι οποίες δεν αφορούν σε τροποποιήσεις στις δομές ή τους αλγορίθμους που χρησιμοποιούνται.

Ως πρώτο βήμα, προσθέστε το πρόθεμα *teamXX_*, όπου XX ο αριθμός της ομάδας σας, σε όλες τις συναρτήσεις του δρομολογητή *noop*. Επίσης, αλλάξτε το όνομα της δομής *elevator-type* και τροποποιήστε κατάλληλα τις εγγραφές της προκειμένου να αναγνωρίζονται τα ονόματα συναρτήσεων μετά τις τροποποιήσεις. Στη συνέχεια, αντικαταστήστε το περιεχόμενο του πεδίου *elevator-name* της δομής με το όνομα της ομάδας σας και τροποποιήστε κατάλληλα τα ορίσματα στα macros *MODULE_AUTHOR* και *MODULE_DESCRIPTION*. Τέλος, προσθέστε τη γραμμή

```
printk( "In teamXX_noop_dispatch function\n" );
```

ως δεύτερη εντολή στη συνάρτηση *team00_noop_dispatch*. Η εντολή αυτή θα εκτυπώνει το αντίστοιχο μήνυμα κάθε φορά που θα καλείται η συνάρτηση για την εξυπηρέτηση κάποιας αίτησης. Με αυτό τον τρόπο θα μπορούμε να διαπιστώσουμε κατά πόσο η δρομολόγηση πραγματοποιείται από το τροποποιημένο module.

Σε αυτό το σημείο μπορείτε να μεταγλωττίσετε τα αρχεία του module. Ακολουθήστε τις οδηγίες του φυλλαδίου "Εισαγωγικός Οδηγός στα Linux Kernel Modules" προκειμένου να δημιουργήσετε το κατάλληλο Makefile και να φορτώσετε το module στον πυρήνα. Πλέον, το module σας είναι έτοιμο προς χρήση.

2.2.2 Έλεγχος Λειτουργίας του Kernel Module

Προκειμένου να μπορεί να χρησιμοποιηθεί ένα kernel module που υλοποιεί I/O scheduling, θα πρέπει να ενημερώσουμε τον πυρήνα για τη συσκευή που θα αναλάβει να διαχειρίζεται το συγκεκριμένο module. Για λόγους επίδειξης, θα χρησιμοποιήσουμε τη συσκευή /dev/sda η οποία, με βάση την ονοματολογία που χρησιμοποιεί το Linux για τις συσκευές, αντιστοιχεί στον πρώτο σκληρό δίσκο του συστήματος. Σε περίπτωση που εργάζεστε σε native installation, αντί για sda στις παρακάτω οδηγίες χρησιμοποιήστε το όνομα του δίσκου που χρησιμοποιεί η εγκατάσταση linux σας (sdb, sdc ...). Μπορείτε να διαπιστώσετε ποιος είναι αυτός εκτελώντας την εντολή *mount*, η οποία σας αποκαλύπτει ποιες συσκευές αποθήκευσης χρησιμοποιείτε και από ποιο σημείο του συστήματος αρχείου είναι προσπελάσιμη η κάθε συσκευή. Παρακάτω βλέπετε ένα ενδεικτικό αποτέλεσμα εκτέλεσης της *mount*.

```
/dev/sda3 on / type ext4 (rw,errors=remount-ro,commit=0)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
...
```

Από το στιγμιότυπο προκύπτει ότι ο κατάλογος / (η ρίζα του συστήματος αρχείων) αντιστοιχεί στη συσκευή sda3 (το 3ο partition του 1ου δίσκου του συστήματος, δηλαδή του sda).

Εισάγετε το module στον πυρήνα χρησιμοποιώντας την εντολή *insmod*. Έπειτα, θα πρέπει να ελέγξετε τα I/O scheduling modules που είναι διαθέσιμα για τη συσκευή /dev/sda. Αυτό μπορεί να πραγματοποιηθεί με την εντολή *cat /sys/block/sda/queue/scheduler*. Μία ενδεικτική έξοδος από την εκτέλεση της εντολής είναι η παρακάτω:

```
noop deadline [cfq] team00_noop
```

Όπως μπορούμε να παρατηρήσουμε, υπάρχουν τέσσερις διαθέσιμοι schedulers. Το όνομα του scheduler που χρησιμοποιείται είναι αυτό που περικλείεται από τις αγκύλες. Για να ενεργοποιήσετε το δικό σας scheduler θα πρέπει να εκτελέσετε την εντολή

sudo bash -c 'echo team00_noop > /sys/block/sda/queue/scheduler'

Μετά από αυτή την εντολή, η έξοδος από την εκτέλεση της εντολής

cat /sys/block/sda/queue/scheduler

θα πρέπει να είναι η ακόλουθη:

```
noop deadline cfq [team00_noop]
```

Όπως παρατηρείτε, η διαχείριση του I/O scheduling θα πραγματοποιείται πλέον από τον τροποποιημένο noop driver. Για να επιβεβαιώσετε ότι αυτό συμβαίνει, δημιουργήστε ένα αρχείο στο home directory σας και προσθέστε κάποιο κείμενο. Έπειτα, εκτελέστε την εντολή *dmesg | tail*. Ανάμεσα στα μηνύματα που εμφανίζονται, θα πρέπει να είστε σε θέση να δείτε και το μήνυμα "In teamXX_noop_dispatch function".

Μετά την επιβεβαίωση της σωστής λειτουργίας, θα πρέπει *οπωσδήποτε* να επαναφέρετε τον αρχικό scheduler για τη συσκευή /dev/sda. Γι' αυτό το σκοπό, εκτελέστε την εντολή

sudo bash -c 'echo cfq > /sys/block/sda/queue/scheduler'

Τέλος, αφαιρέστε το module *project1-iosched* από τον πυρήνα χρησιμοποιώντας την εντολή *rmmod*.

3 Πακετάρισμα των Αλλαγών – Δημιουργία του προς Υποβολή Συνημμένου Αρχείου

3.1 Απομόνωση των Αλλαγών στον Κώδικα του Λειτουργικού

Προφανώς δεν είναι ιδιαίτερα πρακτικό να παραδώσετε τις αλλαγές στέλνοντας όλο το νέο κώδικα του λειτουργικού, δεδομένου και ότι τα αρχεία τα οποία στην πραγματικότητα θα έχετε πειράξει είναι ελάχιστα. Η εφαρμογή *diff* αναλαμβάνει να αναγνωρίσει τις αλλαγές που κάνατε στον πηγαίο κώδικα του λειτουργικού, συγκρίνοντάς τον με το αντίγραφο του αρχικού κώδικα που έχουμε κρατήσει στον κατάλογο /usr/src/linux-3.2.39-orig.

Κατ' αρχήν πηγαίνετε στον κατάλογο /usr/src/linux-3.2.39-dev (cd /usr/src/linux-3.2.39-dev) και δώστε την εντολή *make distclean*. Με τον τρόπο αυτό σβήνονται όλα τα αρχεία (.o, εκτελέσιμα κλπ) που δημιουργήθηκαν κατά τη μεταγλώττιση. Κάνετε το ίδιο -- για καλό και για κακό -- και στον κατάλογο /usr/src/linux-3.2.39-orig. **ΜΗΝ ΞΕΧΑΣΕΤΕ ΑΥΤΑ ΤΑ 2 ΒΗΜΑΤΑ!!!**

Πλέον είστε έτοιμοι να “συγκρίνετε” τους 2 καταλόγους. Πηγαίνετε στο /usr/src και από εκεί δώστε την εντολή

sudo bash -c 'diff -ruN linux-3.2.39-orig linux-3.2.39-dev > patch_1'

Το πρόγραμμα diff εντοπίζει αναδρομικά τις αλλαγές μεταξύ των καταλόγων linux-3.2.39-orig και linux-3.2.39-dev. Το τελικό αποτέλεσμα αποθηκεύεται στο αρχείο patch_1 το οποίο δημιουργείται μέσα στον κατάλογο από τον οποίο καλέσατε την *diff* (δηλαδή τον κατάλογο /usr/src). Αν θέλετε, διαβάστε το αρχείο patch_1 (είναι ένα αρχείο κειμένου) και προσπαθήστε να καταλάβετε τη δομή του.

Αν θέλετε να επιβεβαιώσετε ότι το patch φτιάχτηκε σωστά, μπορείτε να κάνετε τα ακόλουθα: Η εφαρμογή *patch* μπορεί να πάρει ένα patch και να εφαρμόσει τις αλλαγές που περιγράφει το patch σε έναν κατάλογο πηγαίου κώδικα. Κατόπιν, μεταβείτε στον κατάλογο /usr/src/linux-3.2.39-orig και δώστε την εντολή

patch -p1 --dry-run < ../patch_1

Με την εντολή αυτή θα γίνει προσομοίωση εφαρμογής στον κατάλογο που βρισκόμαστε του patch που περιέχεται στο αρχείο patch_1 (το οποίο patch_1 είναι τοποθετημένο στον στον αμέσως προηγούμενο κατάλογο, γι' αυτό και το ../). Η παράμετρος *--dry-run* λέει στην εφαρμογή patch να προσποιηθεί ότι εφαρμόζει το patch χωρίς να κάνει στην πραγματικότητα οποιεσδήποτε αλλαγές στα αρχεία. Το patch θα πρέπει να εφαρμοστεί "καθαρά", δε θα πρέπει δηλαδή να σας εμφανιστούν μηνύματα λάθους. Δώστε προσοχή σε αυτό το βήμα, γιατί προϋπόθεση για τη βαθμολόγηση κάθε άσκησης είναι το patch που θα μας στείλετε να μπορεί να εφαρμοστεί "καθαρά".

3.2 Πακετάρισμα των Προς Υποβολή Στοιχείων σε Αρχείο

Το τελευταίο βήμα είναι να πακετάρετε όλα τα αρχεία που πρέπει να μας στείλετε σε ένα αρχείο. Πηγαίνετε στον home directory του λογαριασμού σας. Φτιάξτε εκεί με την εντολή *mkdir* έναν κατάλογο με όνομα project_1_omada_<AEM_omadas>. Για παράδειγμα, η ομάδα με ΑΕΜ 456 123 789 θα πρέπει να δώσει την εντολή *mkdir project_1_omada_123_456_789*. Αντιγράψτε (με την εντολή *cp*) το patch που φτιάξατε σε αυτό τον κατάλογο με την εντολή. Π.χ. η ομάδα 123_456_789 θα δώσει την εντολή: *cp /usr/src/patch_1 /project_1_omada_123_456_789*.

Κάντε το ίδιο και με τα αρχεία roots.c, roots.h, find_roots_lib.c καθώς επίσης και με τον φάκελο project1_module. Μπείτε στον φάκελο project1_module και δώστε την εντολή *make clean*, ώστε να σβηστούν τα αρχεία που παράχθηκαν κατά τη μεταγλώττιση του module και να μείνει μόνο ο κώδικας και το Makefile. **ΜΗΝ ΞΕΧΑΣΕΤΕ ΑΥΤΟ ΤΟ ΒΗΜΑ!!!**

Επίσης, δημιουργήστε μέσα στον κατάλογο ένα αρχείο με όνομα README.txt στον οποίο θα γράψετε τα ονόματα, ΑΕΜ και e-mail των μελών της ομάδας, καθώς και αναλυτικές οδηγίες για

τη μεταγλώττιση της εφαρμογής find_roots_lib και του module. Μπορείτε να συμπεριλάβετε στο αρχείο και οτιδήποτε άλλο θέλετε να έχουμε υπόψη μας κατά τη διόρθωση.

Ακολούθως, πηγαίνετε στον πηγαίο κατάλογο του λογαριασμού σας και από εκεί δώστε την εντολή

tar -cvf project_1_omada_<AEM_omadas>.tar project_1_omada_<AEM_omadas>

Για παράδειγμα, η ομάδα με μέλη με ΑΕΜ 456 123 789 θα δώσει την εντολή

tar -cvf project_1_omada_123_456_789.tar project_1_omada_123_456_789

Με την εντολή tar θα πακεταριστούν τα περιεχόμενα του καταλόγου σε ένα αρχείο με κατάληξη .tar.

Τέλος, δώστε την εντολή *bzip2 project_1_omada_<AEM_omadas>.tar*. Θα δημιουργηθεί ένα αρχείο με κατάληξη .bz2, το οποίο περιέχει συμπιεσμένο το αρχείο με κατάληξη .tar. Το αρχείο με κατάληξη .bz2 είναι αυτό που θα πρέπει να επισυνάψετε στο mail παράδοσης της άσκησης. Για παράδειγμα, η ομάδα με μέλη με ΑΕΜ 456 123 789 θα δώσει την εντολή

bzip2 project_1_omada_123_456_789.tar

Αφού φτιαχτεί το αρχείο, ελέγξτε το μέγεθός του. Αν το μέγεθος είναι αδικαιολόγητα μεγάλο (θυμηθείτε ότι το αρχείο στην ουσία περιέχει μερικά πολύ μικρά αρχεία κειμένου) έχετε κάνει κάτι λάθος (το πιθανότερο είναι ότι έχετε ξεχάσει το βήμα make distclean ή έχετε συμπεριλάβει και εκτελέσιμα αρχεία).