ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Εργαστήριο Λειτουργικών Συστημάτων

8ο εξάμηνο, Ακαδημαϊκή περίοδος 2010–2011

Οδηγός προγραμματισμού σε περιβάλλον User-Mode Linux

Εργαστήριο Υπολογιστικών Συστημάτων Ε.Μ.Π.
cslab-assist@cslab.ece.ntua.gr

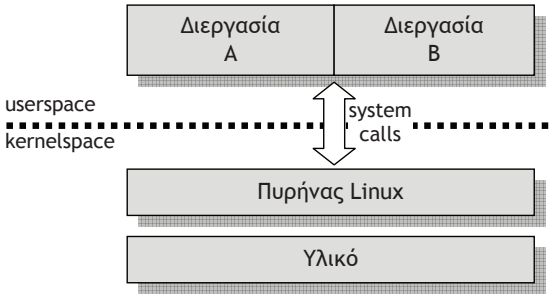
Μάρτιος 2012

Περιεχόμενα

Επιμέλεια: Β. Καρακάσης, Γ. Κουβέλη, Ευ. Κούκης

1 Εισαγωγή – Χώρος πυρήνα και χώρος χρήστη

Το Linux είναι ένα σύγχρονο λειτουργικό σύστημα που ακολουθεί την παράδοση του Unix. Υποστηρίζει πολλούς ταυτόχρονους χρήστες (είναι δηλαδή multi-user) και πολλές ταυτόχρονες διεργασίες (είναι multi-tasking), οι οποίες μοιράζονται το χρόνο της CPU. Η οργάνωση του λογισμικού ενός υπολογιστικού συστήματος όταν χρησιμοποιείται το Linux παρουσιάζεται στο παρακάτω σχήμα.



Σχήμα 1: Οργάνωση σε χώρους πυρήνα και χρήστη

Εφόσον πολλές διαφορετικές διεργασίες πολλών διαφορετικών χρηστών εκτελούνται στο ίδιο υπολογιστικό σύστημα, είναι ανάγκη να υπάρχουν συγκεκριμένοι μηχανισμοί απομόνωσης και ελέγχου πρόσβασης, ώστε διεργασίες που ανήκουν σε κάποιον χρήστη να μην έχουν πρόσβαση σε δεδομένα άλλων χρηστών, μία διεργασία που δυσλειτουργεί να μην επηρεάζει τη λειτουργία των υπολοίπων αλλά και να εξασφαλίζεται η δίκαιη κατανομή των πόρων του συστήματος ανάμεσα στις διεργασίες (χρόνος CPU, μνήμη, πρόσβαση σε συσκευές I/O).

Για να ικανοποιηθούν οι παραπάνω περιορισμοί, κάθε διεργασία εκτελείται απομονωμένη από τις υπόλοιπες, στον δικό της εικονικό χώρο διευθύνσεων αλλά και απομονωμένη από το hardware (δεν επιτρέπεται η εκτέλεση εντολών I/O). Για να μπορέσει να χρησιμοποιήσει τους πόρους του υπολογιστικού συστήματος, χρησιμοποιεί τις υπηρεσίες του πυρήνα του Linux μέσα από τον μηχανισμό των κλήσεων συστήματος. Ο πυρήνας είναι το μόνο πρόγραμμα που έχει τη δυνατότητα για απευθείας προσπέλαση στο υλικό. Ελέγχει κάθε κλήση συστήματος και αν επιτρέπεται η πρόσβαση στη συγκεκριμένη διεργασία, ικανοποιεί το αίτημά της και επιστρέφει τα αποτελέσματα.

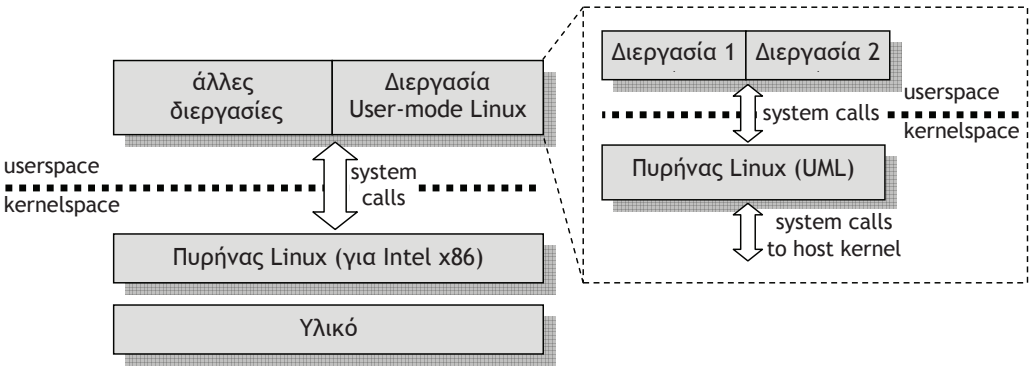
Είναι φανερή λοιπόν η διάκριση ανάμεσα στο χώρο πυρήνα (kernelspace) και το χώρο χρήση (userspace). Ο κώδικας των διεργασιών, που εκτελείται στο userspace, δεν επιτρέπεται να προσπελάσει απευθείας το υλικό και δεν έχει πρόσβαση στο χώρο μνήμης άλλων διεργασιών. Αντίθετα, ο κώδικας του πυρήνα, που εκτελείται στο kernelspace έχει ελεύθερη πρόσβαση στο υλικό και στη μνήμη του συστήματος. Επιπλέον, πιθανή δυσλειτουργία του μπορεί να αποβεί μοιραία για τη

λειτουργία του υπολογιστικού συστήματος.

2 User-Mode Linux

Στις ασκήσεις του Εργαστηρίου Λειτουργικών Συστημάτων σας ζητείται η τροποποίηση του πυρήνα του Linux, ώστε να προστεθούν νέες λειτουργίες (κλήση συστήματος, υποστήριξη συσκευών). Παρόλο που για την διαδικασία ανάπτυξης (τροποποίηση κώδικα, μεταγλώττιση) δεν χρειάζονται αυξημένα δικαιώματα, μόνο ο διαχειριστής του συστήματος (χρήστης root) έχει το δικαίωμα τόσο να εγκαταστήσει ένα νέο πυρήνα όσο και να εισαγάγει κάποιο kernel module στον πυρήνα που ήδη τρέχει. Ο λόγος πίσω από αυτό τον περιορισμό είναι λίγο-πολύ προφανής: ο,τιδήποτε τρέχει στον χώρο πυρήνα, μπορεί να κάνει *τα πάντα* σε ένα σύστημα. Επομένως, εάν όλοι οι χρήστες είχαν αυτό το δικαίωμα, τότε η ασφάλεια και η σταθερότητα του συστήματος θα είχαν καταστρατηγηθεί. Ακόμα, όμως, και αν κανείς είναι ο διαχειριστής ενός συστήματος, δεν είναι σοφή ιδέα να «τεστάρει» τις αλλαγές που έκανε στον πυρήνα απευθείας σε ένα μηχάνημα, ειδικά εάν αυτό το χρησιμοποιούν και άλλοι χρήστες, γιατί και με αυτό τον τρόπο διακυβεύεται η ασφάλεια και η σταθερότητα του συστήματος. Η λύση στα παραπάνω ζητήματα, που αφορούν στον προγραμματισμό του πυρήνα του Linux, είναι η χρήση του user-mode Linux, που αναλύεται στην συνέχεια.

Ο πυρήνας του Linux έχει μεταφερθεί ώστε να τρέχει σε πολλές διαφορετικές αρχιτεκτονικές. Έτσι, εκτός από τους επεξεργαστές x86 της Intel, υπάρχουν εκδόσεις για επεξεργαστές Itanium, PowerPC, SPARC κ.ά. Μία ακόμη αρχιτεκτονική στην οποία έχει μεταφερθεί είναι το userspace του Linux. Υπάρχει δηλαδή μια έκδοση του πυρήνα που δεν αλληλεπιδρά απευθείας με πραγματικό hardware, αλλά εκτελείται επάνω σε έναν άλλο πυρήνα Linux. Έτσι, έχουμε την εξής οργάνωση:



Σχήμα 2: Η εικονική μηχανή του πυρήνα UML.

Ο πυρήνας user-mode Linux (UML) είναι ένας πλήρης πυρήνας Linux, εφαρ-

μόζει ακριβώς τους ίδιους μηχανισμούς χρονοδρομολόγησης στη CPU, διαχείρισης μνήμης, ελέγχου πρόσβασης, υποστηρίζει πολλά διαφορετικά συστήματα αρχείων και προσφέρει σε διεργασίες ακριβώς τον ίδιο μηχανισμό κλήσεων συστήματος. Ωστόσο, για την εξυπηρέτησή τους δεν αλληλεπιδρά απευθείας με το υλικό, όπως θα έκανε αν έτρεχε επάνω σε ένα πραγματικό υπολογιστικό σύστημα, αλλά εκτελείται ως ένα σύνολο διεργασιών επάνω σε έναν άλλο πυρήνα και χρησιμοποιεί με τη σειρά του κλήσεις συστήματος. Οι διεργασίες που εκτελούνται κάτω από το UML, εκτελούνται στην ουσία μέσα σε μία εικονική μηχανή (virtual machine).

Έτσι, τρέχοντας έναν πυρήνα UML στο χώρο χρήστη του πραγματικού συστήματος, θα μπορείτε να έχετε τη δική σας εικονική μηχανή, στην οποία θα έχετε δικαιώματα διαχειριστή, δυνατότητα για οποιαδήποτε μεταβολή στο σύστημα, όντας όμως ταυτόχρονα περιορισμένοι στο χώρο χρήστη του πραγματικού συστήματος. Το πώς ακριβώς γίνεται αυτό, αναλύεται στη συνέχεια.

3 Χρήση του User-Mode Linux

Για να ξεκινήσετε την λειτουργία της εικονικής μηχανής, σας παρέχονται δύο αρχεία για δική σας ευκολία: (i) `utopia.sh`, το οποίο εκκινεί την εικονική μηχανή, και (ii) `utopia.config`, το οποία περιέχει κάποιες απαραίτητες ρυθμίσεις, ώστε να μπορέσει να εκκινήσει σωστά η εικονική μηχανή. Το script `utopia.sh` διαβάζει τις παραμέτρους στο `utopia.config` και εκκινεί την εικονική μηχανή. Εάν συμβεί κάποιο λάθος, θα το επισημάνει και θα προτείνει κάποια πιθανή λύση. Αρχικά, όλες οι παράμετροι στο `utopia.config` είναι σε σχόλια, οπότε θα πρέπει να τις αποσχολιάσετε, ώστε να τρέξει επιτυχώς το `utopia.sh`. Το `utopia.config` περιέχει στην ουσία μία σειρά από μεταβλητές περιβάλλοντος που τις χρησιμοποιεί το `utopia.sh`, ώστε να εκκινήσει την εικονική μηχανή. Οι μεταβλητές αυτές εξηγούνται στην συνέχεια:

`KERNEL_SRCDIR` Αυτός είναι ο φάκελος στον οποίο βρίσκεται ο κώδικας του πυρήνα. Το `utopia.sh`, εάν υπάρχει αυτός ο φάκελος, θα ψάξει για το εκτελέσιμο `linux`, το οποίο και θα εκτελέσει θεωρώντας ότι είναι ο πυρήνας UML.

`UML_ROOT_FS_IMAGE` Αυτό είναι το root filesystem που θα χρησιμοποιήσει η εικονική μηχανή για να τρέξει ένα πλήρες σύστημα Linux. Στο filesystem που σας παρέχεται στο εργαστήριο, υπάρχει εγκατεστημένο Debian 5.0 (lenny). Το filesystem αυτό είναι read-only, αλλά μόλις εκκινήσετε την εικονική μηχανή, θα μπορείτε να δουλεύετε όπως ακριβώς σε ένα native Debian σύστημα. Οι τοπικές αλλαγές, όμως, αποθηκεύονται σε ένα Copy-On-Write (COW) αρχείο, το οποίο εξηγείται στην συνέχεια.

`PRIVATE_COW` Αυτό είναι το τοπικό σας COW αρχείο, στο οποίο αποθηκεύονται όλες οι αλλαγές που κάνετε στο root filesystem. Η κάθε ομάδα έχει το δικό της COW αρχείο και όλες μοιράζονται το αρχικό root filesystem που δίνεται από το εργαστήριο. Ανά πάσα στιγμή μπορείτε να σβήσετε αυτό το αρχείο και να επαναφέρετε την εικονική μηχανή σας στην αρχική της κατάσταση.

ΠΡΟΣΟΧΗ: Σβήνοντας το τοπικό COW αρχείο, ισοδυναμεί με το να κάνετε format στην εικονική μηχανή και να εγκαθιστάτε πάλι το λειτουργικό σύστημα. Όλες οι αλλαγές που τυχόν έχετε κάνει στο root filesystem θα χαθούν!

`OVER_SSH` Αυτή είναι μία δυαδική μεταβλητή με τιμές 'yes' ή 'no', με την οποία δηλώνετε την πρόθεσή σας να τρέξετε την εικονική μηχανή πάνω από SSH. Για περισσότερες πληροφορίες σχετικά με την απομακρυσμένη πρόσβαση, ανατρέξτε στην ενότητα 5.

Αφού πλέον έχετε ρυθμίσει σωστά την εικονική σας μηχανή, μπορείτε να την εκκινήσετε εκτελώντας το `utopia.sh`[1].

```
$ ./utopia.sh
```

Η εικονική μηχανή θα ξεκινήσει και θα εμφανιστεί ένας αριθμός από παράθυρα, τα οποία αντιστοιχούν σε εικονικά τερματικά. Στην εικονική μηχανή μπορείτε να κάνετε login είτε ως διαχειριστής (όνομα: `root`, κωδικός: `root`) είτε ως χρήστης (όνομα: `cslabXXX`, χωρίς κωδικό).

Ακριβώς όπως και για ένα πραγματικό μηχάνημα, η λειτουργία της εικονικής μηχανής πρέπει να τερματίζεται σωστά. Ως root, χρησιμοποιείστε την εντολή `halt` ή `poweroff` για να «σβήσετε» την εικονική μηχανή, ή την εντολή `reboot` για να κάνετε επανεκκίνηση. Εάν για κάποιο λόγο η εικονική μηχανή δεν αποκρίνεται, π.χ., ο τροποποιημένος πυρήνας σας εκτελέσει κάποια εσφαλμένη λειτουργία, μπορείτε να τραβήξετε στην ουσία την εικονική μηχανή από την πρίζα, εκτελώντας στο πραγματικό μηχάνημα `killall linux` και, αν δεν υπάρξει αποτέλεσμα, `killall -9 linux`.

Τα αρχεία που βρίσκονται στο home directory του χρήστη `cslabXXX` μέσα στην εικονική μηχανή, αποθηκεύονται στον φάκελο `uml-home` του πραγματικού μηχανήματος. Έτσι διευκολύνεται η μεταφορά αρχείων από και προς την εικονική μηχανή. Επιπλέον, τα αρχεία αυτά θα είναι ασφαλή ακόμη και σε περίπτωση που το σύστημα αρχείων της εικονικής μηχανής πάθει κάποια βλάβη (π.χ., λόγω προγραμματιστικού σφάλματος κατά την εκτέλεση του πυρήνα σας ή «βίαιου» shutdown της εικονικής μηχανής) και αναγκαστείτε να επαναφέρετε την εικονική μηχανή

[1] Στα επόμενα παραδείγματα υποθέτουμε ότι "$" είναι το πρωτεύον prompt (`PS1`) και ">" το δευτερεύων (`PS2`).

στην αρχική της κατάσταση διαγράφοντας το τοπικό COW αρχείο σας. Για τον λόγο αυτό, σας προτείνουμε να κρατάτε ολόκληρο τον πηγαίο κώδικα της άσκησης στο home directory του χρήστη `cslabXXX` μέσα στην εικονική μηχανή (άρα στο directory `~/uml-home` του πραγματικού συστήματος). Εκεί μπορείτε να κάνετε αλλαγές από το πραγματικό μηχάνημα, τρέχοντας, π.χ., κάποιον editor όπως το gedit σε γραφικό περιβάλλον.

4 Μεταγλώττιση πυρήνα για το UML

Για τις ανάγκες τις πρώτης εργαστηριακής άσκησης θα χρειαστεί να μεταγλωττίσετε τον πυρήνα του Linux για την εικονική μηχανή του UML. Συγκεκριμένα, η μεταγλώττιση του πυρήνα του Linux είναι μια αρκετά απλή διαδικασία, η οποία συνίσταται στα επόμενα βήματα.

Λήψη του κώδικα Αρχικά, θα πρέπει να ανακτήσετε τον κώδικα του πυρήνα είτε από την επίσημη ιστοσελίδα του πυρήνα του Linux (`www.kernel.org`) είτε από την τοποθεσία `/home/cslab/common` στα μηχανήματα του εργαστηρίου. Για τις ανάγκες της άσκησης χρησιμοποιήσαμε την έκδοση 2.6.37.4 του πυρήνα, την οποία και προτείνουμε να χρησιμοποιήσετε. Για να ανακτήσετε την συγκεκριμένη έκδοση από την επίσημη ιστοσελίδα, θα πρέπει να πληκτρολογήσετε

```
$ wget http://www.kernel.org/pub/linux/kernel/\
> v2.6/linux-2.6.37.4.tar.bz2
```

Εναλλακτικά, μπορείτε να τον αντιγράψετε τοπικά από τον σέρβερ του εργαστηρίου στο home directory σας:

```
$ cp /home/cslab/common/linux-2.6.37.4.tar.bz2 .
```

Αποσυμπίεση του κώδικα Ο κώδικας που μόλις ανακτήσατε είναι σε συμπιεσμένη μορφή, οπότε θα πρέπει να τον αποσυμπιέσετε, ώστε να μπορέσετε να εργαστείτε σε αυτόν.

```
$ tar xjf linux-2.6.37.4.tar.bz2
```

Ολόκληρος ο κώδικας του πυρήνα έχει αποσυμπιεστεί στον κατάλογο `linux-2.6.37.4`, στον οποίο μπορείτε πλέον να μεταβείτε και να περιηγηθείτε ή κάνετε τις αλλάγες σας στον πυρήνα.

Ρυθμίσεις του πυρήνα Προτού ξεκινήσετε την μεταγλώττιση του πυρήνα, θα πρέπει να πρώτα να κάνετε κάποιες ρυθμίσεις. Ο πιο απλός και φιλικός προς τον χρήστη τρόπος γι' αυτό είναι μέσω της εντολής

```
$ make menuconfig ARCH=um SUBARCH=i386
```

Η εντολή αυτή θα ανοίξει ένα μενού μέσα στο τερματικό που βρίσκεστε, απ' όπου θα μπορέσετε να κάνετε μία πληθώρα ρυθμίσεων για τον πυρήνα που πρόκειται να μεταγλωττίσετε. Μιας και για τον σκοπό της άσκησης δεν χρειαζόμαστε κάτι εξεζητημένο, οι τυπικές ρυθμίσεις αρκούν, οπότε μπορείτε απλά να πατήσετε στο "Exit" και στην επόμενη ερώτηση για την αποθήκευση των ρυθμίσεων απαντήστε καταφατικά. Εναλλακτικά, μπορείτε να χρησιμοποιήσετε τις ρυθμίσεις του πυρήνα που υπάρχουν στο εργαστήριο.

```
$ cd linux-2.6.37.4
$ cp /home/cslab/common/utopia/linux/.config .
```

Μπορείτε ακόμα να ενεργοποιήσετε με απλό τρόπο τις default ρυθμίσεις για την μεταγλώττιση του πυρήνα, εκτελώντας απλά

```
$ make defconfig ARCH=um SUBARCH=i386
```

Μεταγλώττιση του πυρήνα Τώρα είστε έτοιμοι να μεταγλωττίσετε τον πυρήνα!

```
$ make ARCH=um SUBARCH=i386
```

Το βήμα αυτό μπορεί να πάρει αρκετή ώρα. Στο τέλος, θα έχει δημιουργηθεί ένα εκτελέσιμο αρχείο `linux`, το οποίο αποτελεί και τον πυρήνα UML. Ένας πυρήνας για μία πραγματική μηχανή, ωστόσο, δεν έχει την μορφή ενός εκτελέσιμου αρχείου για κάποια συγκεκριμένη αρχιτεκτονική, όπως συμβαίνει με τον πυρήνα UML, αλλά είναι ένα συμπιεσμένο binary image που φορτώνεται στην μνήμη από τον bootloader κατά την διαδικασία εκκίνησης του συστήματος, αποσυμπιέζεται και, τελικά, αναλαμβάνει τον πλήρη έλεγχο του συστήματος.

Εκκίνηση εικονικής μηχανής με τον νέο πυρήνα Για να δοκιμάσετε τον πυρήνα που μόλις μεταγλωττίσατε στην εικονική μηχανή του UML, θα πρέπει να αλλάξετε την μεταβλητή `KERNEL_SRCDIR` στο αρχείο `utopia.config`, ώστε να δείχνει στον φάκελο που έχετε τον τροποποιημένο κώδικα του δικού σας πυρήνα. Στην συνέχεια, τρέχετε το `utopia.sh`, όπως περιγράφηκε στην ενότητα 3.

Όπως θα παρατηρήσατε, για την μεταγλώττιση «περάσαμε» στο `make` δύο επιπλέον μεταβλητές: την `ARCH` και την `SUBARCH`. Με την μεταβλητή `ARCH` δηλώνουμε την πρόθεσή μας να μεταγλωττίσουμε τον πυρήνα για την αρχιτεκτονική `um`, που αντιστοιχεί στην εικονική μηχανή του UML. Με την μεταβλητή `SUBARCH` δηλώνουμε την αρχιτεκτονιή του host μηχανήματος, στο οποίο πρόκειται να τρέξει το

UML. Στην περίπτωσή μας αυτή είναι x86 32-bit. Θα μπορούσαμε να παραλείψουμε την μεταβλητή αυτή, εάν το μηχάνημα στο οποίο μεταγλωττίζουμε τον πυρήνα για το UML είναι το ίδιο με αυτό στο οποίο πρόκειται να τρέξουμε το UML, όπως συμβαίνει στην περίπτωση του εργαστηρίου. Για δική σας διευκόλυνση δίνεται, επίσης, ένα script, `/home/cslab/common/utopia/build-uml.sh` το οποίο αυτοματοποιεί την διαδικασία μεταγλώττισης του πυρήνα του Linux για την εικονική μηχανή του UML και για host αρχιτεκτονική x86 32-bit.

5 Συχνές ερωτήσεις

Κατά την διαδικασία υλοποίησης της άσκησης μπορούν προκύψουν διάφορα τεχνικά προβλήματα, τα πιο συχνά από τα οποία αναφέρονται στην συνέχεια.

Ερώτηση:
Κατά την εκκίνηση του UML εμφανίζεται κάποιο μήνυμα λάθους για το root fs και ο πυρήνας πανικοβάλλεται.

Απάντηση:
Η πιο συνηθισμένη αιτία για αυτό το μήνυμα λάθους είναι ότι ο πυρήνας δεν μπορεί να κλειδώσει το τοπικό COW αρχείο, που κρατάει τις αλλαγές κάθε ομάδας σε σχέση με το κοινό filesystem. Αυτό φαίνεται από ανάλογο μήνυμα λάθους ανάμεσα στα διαγνωστικά του πυρήνα. Είναι πάρα πολύ σημαντικό μόνο μία εικονική μηχανή να εκτελείται από κάθε ομάδα κάθε φορά, αφού ταυτόχρονη χρήση του COW αρχείου από πολλές εικονικές μηχανές θα μπορούσε να προκαλέσει βλάβες στο εικονικό σύστημα αρχείων. Το πιο πιθανό είναι ότι κάπου έχετε αφήσει να τρέχει ένας πυρήνας UML ο οποίος κρατά κλειδωμένο το αρχείο. Δοκιμάστε να τον τερματίσετε (`killall -9 linux`). Αν αυτό δεν δουλέψει, μπορείτε να σβήσετε το COW αρχείο, οπότε χάνετε όλες τις αλλαγές που έχετε κάνει στην εικονική μηχανή (όχι όμως και τον κώδικά σας, αν τον κρατάγατε στο `/home/cslab/cslabXXX`), και επανέρχεστε στην αρχική κατάσταση.

ΠΡΟΣΟΧΗ: Θα χάσετε ο,τιδήποτε έχετε αποθηκεύσει στην εικονική μηχανή και βρίσκεται έξω από αυτό το directory.

Ερώτηση:
Το UML δεν εκκινεί (boot) και «κολλάει» με μία σειρά από μηνύματα της μορφής ***"request_module: runaway loop modprobe binfmt-464c"****.*

Απάντηση:
Το πιθανότερο πρόβλημα σε αυτή την περίπτωση είναι ότι ο πυρήνας UML που προσπαθείτε να εκκινήσετε είναι μεταγλωττισμένος για 64-bit host και το root

filesystem που χρησιμοποιείτε είναι για αρχιτεκτονική 32-bit. Συγκεκριμένα, ο πυρήνας UML προσπαθεί να φορτώσει κάποια modules που περιμένει να είναι για αρχιτεκτονική 64-bit και, τελικά, βρίσκει modules για αρχιτεκτονική 32-bit. Γι' αυτό αρνείται να προχωρήσει με την εκκίνηση του συστήματος. Το πρόβλημα αυτό μπορεί να ανακύψει εάν μεταγλωττίσετε τον πυρήνα UML σε κάποιο δικό σας μηχάνημα 64-bit και προσπαθήσετε να χρησιμοποιήσετε το root filesystem του εργαστηρίου που παρέχεται για την άσκηση. Η λύση σ' αυτό το πρόβλημα είναι να μεταγλωττίστε ξανά τον πυρήνα όπως αναφέρεται στην ενότητα 4 προσέχοντας να χρησιμοποιείτε την μεταβλητή `SUBARCH=i386` σε κάθε βήμα της μεταγλώττισης. Με αυτό τον τρόπο δηλώνετε ότι ο πυρήνας UML που θα φτιάξετε θέλετε να είναι για αρχιτεκτονική host 32-bit[2], όπως είναι τα μηχανήματα του εργαστηρίου.

Ερώτηση:
Μπορώ να τρέξω το UML πάνω από SSH, ώστε να δουλέψω από απόσταση;

Απάντηση:
Ναι, μπορείτε. Κάνετε login κανονικά σε κάποιο μηχάνημα του εργαστηρίου, θέτετε `OVER_SSH=yes` στο αρχείο `utopia.config` και εκκινείτε το UML κατά τα γνωστά. Μόλις το UML θα εκκινήσει, ανάμεσα στ' άλλα, θα τυπώσει και ένα μήνυμα της μορφής **"Virtual console 1 assigned device '/dev/pts/X'"**. Θα παρατηρήσατε ότι αυτή την στιγμή δεν έχετε πρόσβαση στην εικονική μηχανή UML. Για να αποκτήσετε, κάνετε πάλι login στο ίδιο μηχάνημα, και αφότου εισέλθετε στον λογαριασμό σας τρέχετε την εντολη

```
$ screen /dev/pts/X
```

και αμέσως μετά πατήστε "Enter": θα πρέπει να εμφανιστεί το login prompt της εικονικής μηχανής και από εδώ και πέρα θα μπορέσετε να δουλέψετε κανονικά στην εικονική μηχανή. Μόλις τελειώσετε, εκτελέστε κατά τα γνωστά την εντολή `halt`, οπότε θα τερματιστεί και το πρόγραμμα `screen`. Το πρόγραμμα `screen` είναι ένα πρόγραμμα χειρισμού τερματικών με πολλές δυνατότητες, στο οποίο λέτε να «προσκολληθεί» στο ψευδο-τερματικό `/dev/pts/X`. Περισσότερα για τις δυνατότητες του `screen` μπορείτε να μάθετε στο αντίστοιχο manpage.

Ερώτηση:
Πώς μπορώ να απομονώσω τις αλλαγές που έκανα στον κώδικα του πυρήνα και να τις εφαρμόσω σε έναν άλλο πυρήνα, π.χ., στο μηχάνημά μου ή αντίστροφα;

Απάντηση:
Για να το πετύχετε αυτό θα πρέπει να φτιάξετε ένα *patch* για τον αρχικό αυθε-

[2] Γι' αυτού του είδους την μεταγλώττιση θα χρειαστείτε και τις βιβλιοθήκες 32-bit για το σύστημά σας.

ντικό πυρήνα (όπως τον κατεβάσατε από το διαδίκτυο ή από το εργαστήριο), το οποίο θα περιέχει όλες τις αλλαγές που κάνατε σε οποιαδήποτε αρχεία. Για να δημιουργήσετε το patch, θα πρέπει να έχετε *και* τις δύο εκδόσεις του πυρήνα στο μηχάνημά σας: την αυθεντική και την τροποποιημένη. Στο εργαστήριο η αυθεντική έκδοση του πυρήνα που χρησιμοποιούμε για την άσκηση βρίσκεται στο `/home/cslab/common/utopia/linux`, οπότε αρκεί να έχετε στον χώρο σας την δική σας τροποποιημένη έκδοση. Στην ουσία, ένα patch για τον πυρήνα του Linux δεν είναι τίποτα άλλο από ένα `diff` των source αρχείων του. Για την διευκόλυνσή σας υπάρχει το script `mkpatch.sh` που κάνει όλες τις απαραίτητες ενέργειες, ώστε να δημιουργηθεί σωστά και αποδοτικά το patch σας. Το script αυτό υποθέτει ότι ο τροποποιημένος πυρήνας βρίσκεται σε κάποιο φάκελο `linux`, ενώ ο αυθεντικός σε κάποιο φάκελο `linux-orig`. Επομένως, θα χρειαστεί να δημιουργήσετε συμβολικές συνδέσεις (symbolic links) στους αντίστοιχους πραγματικούς φακέλους. Αφότου τρέξετε επιτυχώς το `mkpatch.sh`, το patch με τις αλλαγές σας θα βρίσκεται στο αρχείο `curse.patch`.

Για να εφαρμόσετε το patch που μόλις δημιουργήσατε σε ένα «καθαρό» πυρήνα, θα πρέπει να εκτελέσετε την `patch` στο top-level source directory του πυρήνα, έστω `linux-vanilla`, το εξής:

```
$ cd linux-vanilla
$ patch -p1 < ../curse.patch
```

Πλέον οι αλλαγές σας έχουν εφαρμοστεί στον πυρήνα που βρίσκεται στον φάκελο `linux-vanilla`. Όταν εφαρμόζετε το patch θα πρέπει να προσέξετε ο πυρήνας στον οποίο το εφαρμόζετε αφενός να είναι ίδιας έκδοσης με τον αρχικό βάσει του οποίου δημιουργήθηκε το patch και αφετέρου να είναι «καθαρός», δηλ., μην έχετε εφαρμόσει άλλο patch προηγουμένως. Σε αντίθετη περίπτωση, υπάρχει πιθανότητα να δημιουργηθεί κάποιο conflict και να αποτύχει η εφαρμογή του patch.

Ερώτηση:
Μπορώ να τρέξω το UML στο δικό μου μηχάνημα;

Απάντηση:
Ναι, μπορείτε. Για να το επιτύχετε αυτό, θα χρειαστείτε τα εξής: (α′) ένα πυρήνα μεταγλωττισμένο για το UML, (β′) ένα root filesystem, και (γ′) υποστηρικτικά εργαλεία για το UML και βιβλιοθήκες για την μεταγλώττιση εφαρμογών 32-bit. Για το πρώτο έχουμε ήδη αναφερθεί εκτενώς στην ενότητα 4. Όσον αφορά το root filesystem, μπορείτε να φτιάξετε το δικό σας, αλλά η πιο απλή λύση είναι να χρησιμοποιήσετε το root filesystem που σας παρέχεται στο εργαστήριο[3]. Στην συνέχεια,

[3]Θα πρέπει να έχετε υπόψη σας ότι το root filesystem του εργαστηρίου είναι για 32-bit αρχιτεκτονική, οπότε κατά την διαδικασία μεταγλώττισης του UML πυρήνα σας δεν θα πρέπει να ξεχάσετε την μεταβλητή `SUBARCH`, εάν έχετε μηχάνημα 64-bit.

θα πρέπει να τροποποιήσετε κατάλληλα το `utopia.config`, ώστε να χρησιμοποιήσει το δικό σας (πλέον) root filesystem. Για να δουλέψετε τώρα στην εικονική μηχανή, θα πρέπει να κάνετε login ως χρήστης root. Παρόλο που ο λογαριασμός σας, π.χ., `cslabb23`, υπάρχει, το home directory που είναι συσχετισμένο με αυτόν δεν υπάρχει. Αντίθετα, το `/home` της εικονικής μηχανής αντιστοιχεί στο `/home` του πραγματικού μηχανήματός σας και έτσι μπορείτε να έχετε κανονικά πρόσβαση στα τοπικά σας αρχεία μέσω της εικονικής μηχανής. Τέλος, εάν προτίθεστε να χρησιμοποιήσετε το root filesystem του εργαστηρίου και έχετε μηχάνημα 64-bit, θα πρέπει να εγκαταστήσετε τόσο την libc για 32-bit αρχιτεκτονικές όσο και τον GCC για 32-bit. Συγκεκριμένα, για διανομές βασισμένες στο Debian, θα πρέπει να έχετε εγκατεστημένα τα εξής πακέτα:

libc6-i386 Η βιβλιοθήκη libc για 32-bit αρχιτεκτονικές.

libc6-i386-dev Απαραίτητα αρχεία για την ανάπτυξη εφαρμογών 32-bit.

gcc-multilib Αρχεία του GCC για ανάπτυξη εφαρμογών σε συστήματα 64-bit που υποστηρίζουν και εκτέλεση εφαρμογών 32-bit.

uml-utilities Βοηθητικά εργαλεία για το UML.

Ερώτηση:
Ακολουθώντας τις οδηγίες καταφέρνω να εκκινήσω την εικονική μηχανή, αλλά στο τερματικό που ανοίγει δείχνει να έχει κάνει login με τον δικό μου λογαριασμό στο πραγματικό μου μηχάνημα και ελέγχει το δικό μου filesystem.

Απάντηση:
Αυτό είναι ένα σύνηθες πρόβλημα που παρουσιάζεται όταν δεν έχουν εγκατασταθεί τα βοηθητικά εργαλεία για το UML (βλ. 'uml-utilities'). Ένα από αυτά τα εργαλεία είναι υπεύθυνο για την «σύνδεση» του εικονικού τερματικού που ανοίγει κατά την έναρξη της εικονικής μηχανής με αυτή. Η λύση σ' αυτή την περίπτωση είναι να εγκαταστήσετε το πακέτο 'uml-utilities' και να δοκιμάσετε ξανά.

Ερώτηση:
Υπάρχει περίπτωση να αλλάξει το root filesystem ή να γίνουν αλλαγές στα αρχεία της εικονικής μηχανής κατά την διάρκεια του εξαμήνου; Πώς ξεχωρίζω τις διαφορετικές εκδόσεις και πώς «εγκαθιστώ» ένα καινούργιο root filesystem?

Απάντηση:
Τόσο το root filesystem που σας παρέχεται στο εργαστήριο όσο και τα αρχεία για την εκκίνηση της εικονικής μηχανής βρίσκονται σε σταθερή (stable) μορφή. Παρόλα αυτα, εάν προκύψει κάποιο πρόβλημα ή σφάλμα σε αυτά ή αλλάξουν κάποιες

από τις απαιτήσεις/ανάγκες του μαθήματος, ενδέχεται τα αρχεία αυτά να πρέπει να αλλάξουν. Για να μπορούν να αναγνωρίζονται οι διαφορετικές εκδόσεις, τόσο το αρχείο του root filesystem όσο και το συμπιεσμένο αρχείο με τα απαραίτητα για την εκκίνηση της εικονικής μηχανής αρχεία έχουν ως επίθεμα την ημερομηνία της τελευταίας τροποποίησης. Για παράδειγμα, το root filesystem την στιγμή που γράφεται αυτός ο οδηγός είναι το `cslab_common_rootfs_2012-03-21`, που σημαίνει ότι τροποποιήθηκε στις 21/03/2012. Τυχόν αλλαγές και νέες «εκδόσεις» στα σχετικά με το UML αρχεία θα κοινοποιούνται στην λίστα του μαθήματος και θα ενημερώνεται η σελίδα του.

Εάν αλλάξει το `utopia.sh` που εκκινεί την εικονική μηχανή, τότε απλά κατεβάζετε την τελευταία έκδοση από την σελίδα του εργαστηρίου και την χρησιμοποιείτε χωρίς καμία αλλαγή.

Στην περίπτωση που αλλάξει το root filesystem, αφότου το κατεβάσετε σε συμπιεσμένη μορφή (περίπου 200MB) από την σελίδα του εργαστήριου, θα πρέπει να το αποσυμπιέσετε, να ανανεώσετε την αντίστοιχη μεταβλητή στο `utopia.config`, να σβήσετε το COW αρχείου του παλιού root filesystem και, τέλος, να εκκινήσετε την εικονική μηχανή.

ΠΡΟΣΟΧΗ: Σβήνοντας το COW αρχείου παλιού root filesystem, χάνετε όλες τις αλλαγές που είχατε κάνει σε αυτό.

6 Μία "Hello, World!" κλήση συστήματος στο Linux

Οι διεργασίες χρήστη δεν είναι δυνατόν να εκτελέσουν κώδικα του πυρήνα απευθείας με κλήση συναρτήσεων του πυρήνα, γιατί ο κώδικας και τα δεδομένα του πυρήνα βρίσκονται σε προστατευμένες θέσεις μνήμης. Όταν λοιπόν κάποιο πρόγραμμα χρήστη πρέπει να ζητήσει την εκτέλεση μιας λειτουργίας από τον πυρήνα, για παράδειγμα Ε/Ε από κάποιο αρχείο, πρέπει να εκτελέσει μια κλήση συστήματος. Ο μηχανισμός με τον οποίο το πρόγραμμα ενημερώνει τον πυρήνα ότι θέλει να εκτελέσει μια κλήση συστήματος, είναι μια διακοπή λογισμικού (software interrupt). Ο χειριστής της διακοπής αυτής είναι ο χειριστής κλήσεων συστήματος (συνάρτηση `system_call:arch/x86/kernel/entry_32.S`), η υλοποίηση του οποίου εξαρτάται από την αρχιτεκτονική του συστήματος. Η διακοπή αυτή στην αρχιτεκτονική x86 είναι η `$0x80`. Με την εκτέλεση αυτής της διακοπής πραγματοποιείται μετάβαση από το χώρο χρήστη στο χώρο πυρήνα και εξυπηρείται η αίτηση του προγράμματος χρήστη.

Μιας και ο μηχανισμός μετάβασης σε χώρο πυρήνα είναι κοινός για όλες τις κλήσεις συστήματος, πρέπει ο πυρήνας να έχει τρόπο να διαχωρίζει τις κλήσεις συστήματος μεταξύ τους. Για το λόγο αυτό, στο Linux σε κάθε κλήση συστήματος αντιστοιχείζεται ένας αριθμό (syscall number). Όταν ένα πρόγραμμα χρήστη

εκτελεί μια κλήση συστήματος, αναφέρεται σε αυτή με βάση τον αριθμό αυτό (περνώντας τον ως "όρισμα", συνήθως μέσω κάποιου καταχωρητή) και όχι με βάση το όνομα της κλήσης συστήματος. Ο πυρήνας του Linux διατηρεί μια λίστα με όλες τις καταχωρημένες κλήσεις συστήματος στον πίνακα κλήσεων συστήματος (`sys_call_table`). Ο πίνακας αυτός εξαρτάται από την αρχιτεκτονική του συστήματος και για την αρχιτεκτονική x86 (32-bit) ορίζεται στο αρχείο `arch/x86/kernel/syscall_table_32.S`.

Η συνάρτηση `system_call` (system call handler) ελέγχει ότι ο αριθμός της κλήσης συστήματος είναι έγκυρος συγκρίνοντάς τον με μια σταθερά που δηλώνει τον συνολικό αριθμό των registered κλήσεων συστήματος `NR_syscalls`. Αν ο αριθμός είναι έγκυρος, καλείται η αντίστοιχη συνάρτηση από τον πίνακα κλήσεων συστήματος (`sys_call_table`).

Με βάση τα παραπάνω, για την προσθήκη μιας κλήσης συστήματος σε ένα σύστημα Linux x86 32-bit πρέπει να ακολουθηθούν τα εξής βήματα:

(1) Προσθέτουμε στο αρχείο `arch/x86/include/asm/unistd_32.h` τον ορισμό του αριθμού της κλήσης συστήματος που θέλουμε να υλοποιήσουμε, αυξάνοντας κατά ένα τον τελευταίο ήδη ορισμένο αριθμό. Για παράδειγμα, προσθέτουμε τη γραμμή:

```
#define __NR_foo    341
```

Στο ίδιο αρχείο, αυξάνουμε κατά ένα την τιμή της σταθεράς NR_syscalls.

(2) Προσθέτουμε στο αρχείο arch/x86/kernel/syscall_table_32.S για κάθε κλήση συστήματος που υλοποιούμε μια γραμμή σαν την εξής:

```
.long sys_foo
```

Με τη γραμμή αυτή προσθέτουμε στην ουσία στον πίνακα κλήσεων συστήματος (`sys_call_table`) την συνάρτηση που θα υλοποιεί την κλήση συστήματος που προσθέτουμε.

(3) Προσθέτουμε στο αρχείο include/linux/syscalls.h για κάθε κλήση συστήματος που υλοποιούμε τη δήλωση της συνάρτησης που θα την υλοποιεί:

```
asmlinkage long sys_foo(int flag);
```

(4) Προσθέτουμε στον φάκελο `kernel/` (ή οπουδήποτε αλλού κρίνουμε κατάλληλο) ένα αρχείο με την υλοποίηση της συνάρτησης που υλοποιεί την κλήση συστήματος. Μπορούμε επίσης να την προσθέσουμε σε ένα ήδη υπάρχον αρχείο. Ένα παράδειγμα είναι το ακόλουθο (προστίθεται στο νέο αρχείο `kernel/foo.c`):

```
#include "linux/kernel.h"

asmlinkage long sys_foo(int flag)
{
   printk(KERN_DEBUG
          "Hello␣from␣foo!␣flag=%d.\n", flag);
   return 0;
}
```

Η κλήση συστήματος `foo()` απλώς τυπώνει στο log αρχείο του πυρήνα (`/var/log/kern.log`) ένα string και το όρισμά της, μέσω της συνάρτησης `printk()`.

(5) Αν κατά το προηγούμενο βήμα δημιουργήσαμε νέο αρχείο, προσθέτουμε μια γραμμή σαν την ακόλουθη στο Makefile του καταλόγου όπου το προσθέσαμε (στην συγκεκριμένη περίπτωση, στο kernel/Makefile):

```
obj-y += foo.o
```

(6) Μεταγλωττίζουμε ξανά τον πυρήνα.

```
$ make ARCH=um SUBARCH=i386
```

(7) Ένα απλό πρόγραμμα (στον χώρο χρήστη) που καλεί την κλήση συστήματος που μόλις φτιάξαμε είναι το ακόλουθο:

```
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
#define __NR_foo    341

int main()
{
   long ret;
   ret = syscall(__NR_foo, 100);
   printf("ret=%lu\n", ret);
   return 0;
}
```

Εκτελώντας το συγκεκριμένο παράδειγμα, καλούμε την νέα κλήση συστήματος `foo()`. Μπορούμε να δούμε το απλό μήνυμα που τυπώνει στα logs του πυρήνα δίνοντας την εντολή `dmesg` στον φλοιό.

7 Περισσότερες πληροφορίες

Περισσότερα για το User-Mode Linux, το σχεδιασμό, την εγκατάσταση και τη χρήση του μπορείτε να μάθετε στη σελίδα http://user-mode-linux.sourceforge.net.