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

7ο εξάμηνο, Ακαδημαϊκή περίοδος 2021–2022

Οδηγός προγραμματισμού σε περιβάλλον QEMU-KVM

Εργαστήριο Υπολογιστικών Συστημάτων Ε.Μ.Π.
os-lab@lists.cslab.ece.ntua.gr

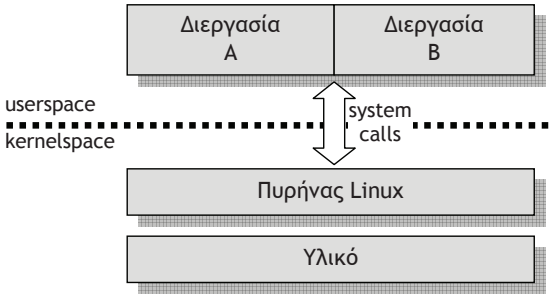
Οκτώβριος 2021

Περιεχόμενα

Επιμέλεια: Δ. Σιακαβάρας, 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 Virtualization (Εικονικοποίηση)

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. Ακόμη όμως και αν κανείς είναι ο διαχειριστής ενός συστήματος, δεν είναι σοφή ιδέα να δοκιμάζει 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 εικονικοποίησης, η οποία αναλύεται στην συνέχεια.

Με τον όρο *εικονικοποίηση* (virtualization), αναφερόμαστε συνήθως σε με μεθόδους και τεχνικές για τη δημιουργία «εικονικών» αντικειμένων, αντίστοιχων των «φυσικών», τα οποία μπορούν να χρησιμοποιηθούν με ισοδύναμο τρόπο. Στην περίπτωσή μας αναφερόμαστε σε εικονικοποίηση υλικού (full virtualization), όπου ο τελικός στόχος είναι η κατασκευή ολόκληρων *εικονικών μηχανών* (virtual machines): προσομοιώνουμε τη λειτουργία ενός ολόκληρου υπολογιστικού συστήματος σε λογισμικό. Μέσα στην εικονική μηχανή που προκύπτει, ο χρήστης μπορεί να τρέχει το Λειτουργικό Σύστημα και εφαρμογές με τον ίδιο ακριβώς τρόπο όπως και στο πραγματικό μηχάνημα. Τα πλεονεκτήματα είναι πολλά, ανάμεσα στα άλλα: (i) ευελιξία – ταυτόχρονη εκτέλεση πολλών διαφορετικών ΛΣ στο ίδιο φυσικό σύστημα, (ii) φορητότητα – προγράμματα εκτελούνται σε σε διαφορετικό υλικό από αυτό για το ποίο έχουν γραφτεί/μεταγλωττιστεί, (iii) ασφάλεια – κάθε εικονική μηχανή εκτελείται απομονωμένα από όλες τις υπόλοιπες και από το φυσικό υλικό, (iv) αποδοτικότερη χρήση των φυσικών πόρων – μεγάλο πλήθος εικονικών μηχανών μπορεί να μοιράζεται περιορισμένους φυσικούς πόρους, (v) βελτιωμένη αξιοπιστία – η εικονική μηχανή μπορεί να μεταφερθεί σε διαφορετικό φυσικό σύστημα, ώστε να συνεχίσει τη λειτουργία χωρίς διακοπή, ξεπερνώντας προβλήματα του υλικού, π.χ. μια επερχόμενη διακοπή ρεύματος.

Η προσομοίωση της λειτουργίας ενός υπολογιστικού συστήματος εξ ολοκλήρου σε λογισμικό είναι μια υπολογιστικά απαιτητική διαδικασία. Η ανάγκη για μετάφραση από το σύνολο εντολών της εικονικής μηχανής ("*guest*") στο σύνολο εντολών του φυσικούς μηχανήματος ("*host*") εισάγει σημαντική επιβάρυνση στην ταχύτητα εκτέλεσης της εικονικής μηχανής.

Για το λόγο αυτό, συνήθως δεν γίνεται μετάφραση κάθε εντολής, αλλά η εικονική μηχανή εκτελείται σε φυσικό σύστημα με το *ίδιο* σύνολο εντολών. Έτσι, τα πράγματα απλοποιούνται σημαντικά: το λογισμικό δεν χρειάζεται να μεταφράζει κάθε εντολή, αλλά χρειάζεται να παρεμβαίνει για να συντηρήσει την ψευδαίσθηση μόνο όταν η εικονική μηχανή χρειάζεται να εκτελέσει μια *προνομιούχο* εντολή.

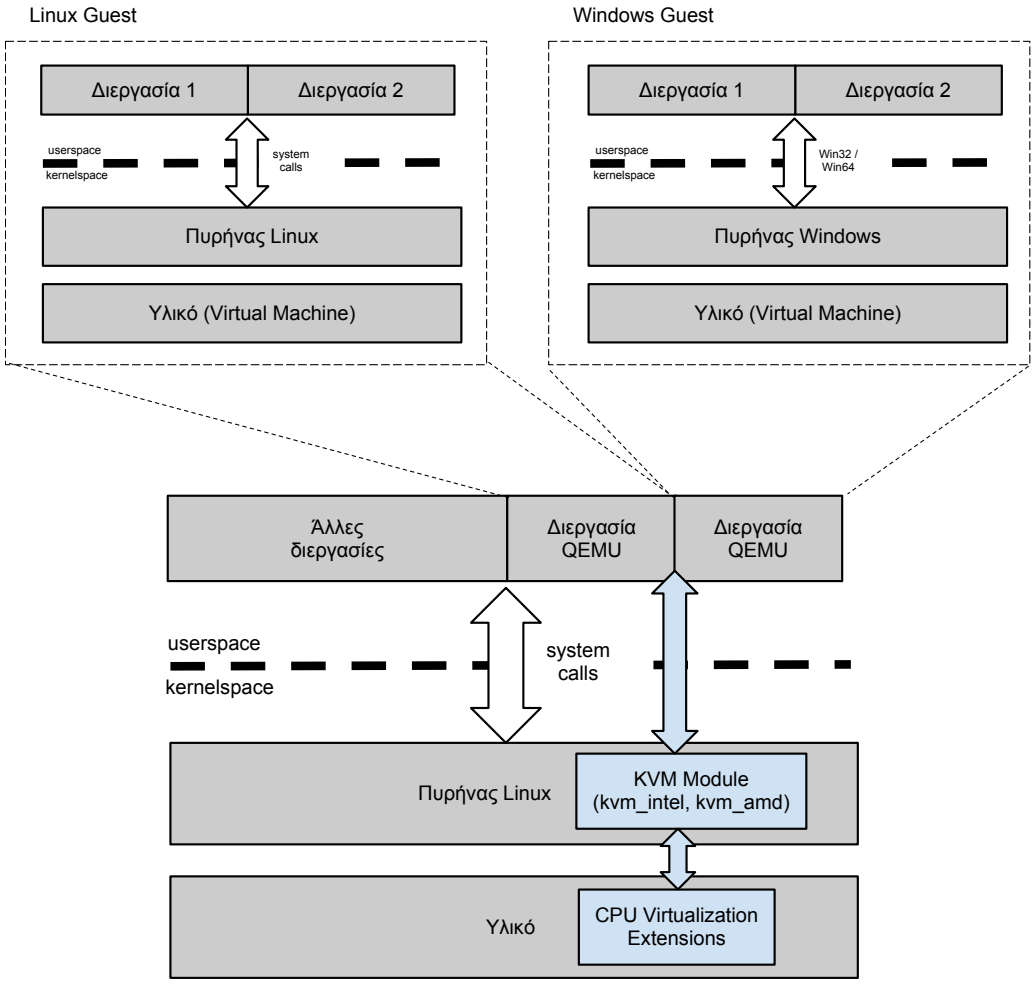
Επιπλέον, η αυξημένη ζήτηση για αποδοτική εκτέλεση εικονικών μηχανών, οδήγησαν στην ανάπτυξη ειδικών επεκτάσεων υλικού (π.χ. ειδικές εντολές στο σύνολο εντολών των σύγχρονων επεξεργαστών), έτσι ώστε το *υλικό* να υποστηρίζει αποδοτικότερα λειτουργίες εικονικοποίησης. Οι νεότεροι επεξεργαστές x86, τόσο από την AMD όσο και από την Intel, διαθέτουν τέτοιες επεκτάσεις, επιτρέποντας την εκτέλεση εικονικών μηχανών με απόδοση που προσεγγίζει αυτή του πραγματικού μηχανήματος.

Παραδείγματα λογισμικού virtualization αποτελούν τα VMWare ESXi, Microsoft Hyper-V, Oracle VM VirtualBox, Xen και QEMU. Στο εργαστήριο θα χρησιμοποιήσουμε για τη δημιουργία εικονικών μηχανών το QEMU-KVM, δηλαδή το λογισμικό QEMU σε συνδυασμό με το υποσύστημα KVM του πυρήνα του Linux.

Το KVM (Kernel-based Virtual Machine) είναι ένα υποσύστημα εικονικοποίησης που έχει ενσωματωθεί στον πηγαίο κώδικα του πυρήνα του Linux από την έκδοση 2.6.20, το οποίο προσφέρει έναν ενιαίο τρόπο για χρήση των επεκτάσεων εικονικοποίησης του επεξεργαστή. Αποτελείται από δύο modules, το `kvm.ko` και το `kvm_intel.ko` ή το `kvm_amd.ko` ανάλογα με το μοντέλο του επεξεργαστή του φυσικού μηχανήματος.

Το QEMU (Quick EMUlator) είναι μία πλατφόρμα για τη δημιουργία και διαχείριση εικονικών μηχανών. Το QEMU εκτελείται σε επίπεδο χρήστη και υποστηρίζει εικονικοποίηση είτε αμιγώς μέσω λογισμικού, είτε εικονικοποίηση μέσω υλικού. Στην πρώτη περίπτωση δεν απαιτεί ειδική υποστήριξη από το υλικό αλλά είναι πολύ αργό, ειδικά αν το σύνολο εντολών της εικονικής μηχανής διαφέρει από το σύνολο εντολών του φυσικού μηχανήματος. Στη δεύτερη περίπτωση, χρησιμοποιεί το KVM που είδαμε προηγουμένως ώστε να έχει πρόσβαση στις επεκτάσεις εικονικοποίησης του επεξεργαστή. Τότε, η εικονική μηχανή είναι απαραίτητο να είναι της ίδιας αρχιτεκτονικής με το μηχάνημα στο οποίο εκτελείται.

Εκτελώντας μία διεργασία `qemu` στο χώρο χρήστη του πραγματικού συστήματος, θα μπορείτε να έχετε τη δική σας εικονική μηχανή, στην οποία θα έχετε πλήρες δι-



Σχήμα 2: Δημιουργία εικονικών μηχανών με χρήση των εργαλείων QEMU και KVM.

καίωμα διαχειριστή και δυνατότητα για οποιαδήποτε μεταβολή στο σύστημα. Ταυτόχρονα όμως θα είστε περιορισμένοι στο χώρο χρήστη του πραγματικού συστήματος. Το πώς ακριβώς γίνεται αυτό, αναλύεται στη συνέχεια. Στις επόμενες ενότητες θα αναφερόμαστε με τον όρο host στο πραγματικό μας μηχάνημα στο οποίο εκτελείται το QEMU, ενώ με τον όρο guest θα εννοούμε την εικονική μηχανή.

3 Προετοιμασία host μηχανήματος

Για την εκτέλεση του QEMU-KVM όπως περιγράφεται στον παρόντα οδηγό, θα

χρειαστείτε πρόσβαση σε κάποιο μηχάνημα το οποίο διαθέτει επεξεργαστή με δυνατότητες εικονικοποίησης και τρέχει κάποια διανομή Linux. Μπορείτε να χρησιμοποιήσετε ένα δικό σας μηχάνημα, ή, αν θέλετε να ακολουθήσετε ακριβώς τις οδηγίες του παρόντος οδηγού, να δημιουργήσετε μία εικονική μηχανή στην υπηρεσία ~okeanos (http://okeanos.grnet.gr, βλ. ενότητα 6).

Στο σημείο αυτό μπορείτε να αποκτήσετε πλήρη γραφική πρόσβαση στο host μηχάνημά σας, αλλά και στο guest VM αργότερα, χρησιμοποιώντας το σύστημα X2Go. Ο server του X2Go είναι ήδη εγκατεστημένος στις υποστηριζόμενες εικόνες μηχανών του ~okeanos, αλλά και στο root filesystem που σας δίνεται για την εικονική μηχανή ανάπτυξης. Η σύνδεση X2Go πραγματοποιείται πάνω από κρυπτογραφημένο κανάλι ssh. Ακολουθήστε τις οδηγίες σύνδεσης όπως περιγράφονται στην ενότητα 6.

ΠΡΟΣΟΧΗ: Στα ακόλουθα υποθέτουμε μια διανομή Debian ή βασισμένη στο Debian. Οι οδηγίες έχουν δοκιμαστεί πλήρως σε διανομή Ubuntu LTS 14.04 64bit. Θεωρούμε ότι υπάρχει λογαριασμός απλού χρήστη `user` κι εκτελείτε τις εντολές ως `user`. Η εικόνα της εικονικής μηχανής που σας δίνεται (`cslab_rootfs`) είναι μηχάνημα Debian Buster x86-64, οπότε απαιτείται κι ο host να είναι εγκατάσταση x86-64 για να εκτελέσει εικονική μηχανή αυτού του τύπου με KVM.

Πριν ξεκινήσετε, βεβαιωθείτε ότι έχετε θέσει το password του `root`, και μπορείτε να αποκτήσετε root shell, δίνοντας `su -`. Για το σκοπό αυτό:

```
$ sudo su -
... enter user's password here, if requested ...
# passwd
... change password for root, pick a STRONG password ...
# exit
$ su -
... enter password for root again ...
# ... you are now the superuser, note the different prompt.
# exit
$ ... back to unprivileged user shell.
```

Το πρώτο πράγμα που πρέπει να κάνουμε στο μηχάνημα αυτό που θα χρησιμοποιηθεί ως host για τις εικονικές μηχανές μας, είναι να εγκαταστήσουμε κάποια πακέτα από τα repositories της διανομής μας τα οποία απαιτούνται. Για αυτό το σκοπό δίνουμε ως root την επόμενη εντολή[1]

```
# apt-get install build-essential vim bzip2 gzip xz-utils
# apt-get install socat git libglib2.0-dev
```

[1]Στα επόμενα παραδείγματα υποθέτουμε ότι "`$`" είναι το πρωτεύον prompt (`PS1`) του χρήστη, "`#`" του `root`, "`>`" το δευτερεύον (`PS2`).

```
# apt-get install libfdt-dev libpixman-1-dev zlib1g-dev
```

3.1 Έλεγχος για CPU virtualization extensions

Για να ελέγξετε αν ο υπολογιστής σας παρέχει επεκτάσεις εικονικοποίησης ανοίξτε ένα τερματικό και δώστε την παρακάτω εντολή:

```
$ egrep '(vmx|svm)' /proc/cpuinfo
```

Σε περίπτωση που η εντολή εκτυπώσει έξοδο που περιλαμβάνει τις λέξεις `vmx` (Intel) ή `svm` (AMD) σημαίνει ότι ο επεξεργαστής σας παρέχει δυνατότητες εικονικοποίησης. Για παράδειγμα, η έξοδος της παραπάνω εντολής για επεξεργαστή Intel δίνει έξοδο παρόμοια με την παρακάτω:

```
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep
   mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
   sse2 ss ht tm pbe nx lm constant_tsc arch_perfmon pebs bts
    aperfmperf pni dtes64 monitor ds_cpl vmx smx est tm2
   ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dtherm
   tpr_shadow vnmi flexpriority
```

3.2 Φόρτωση KVM modules

Αφού έχουμε βεβαιωθεί ότι ο επεξεργαστής μας παρέχει τις επεκτάσεις εικονικοποίησης, πρέπει να ελέγξουμε αν ο πυρήνας Linux που τρέχουμε περιλαμβάνει τα modules του KVM. Αν η εντολή

```
$ lsmod | grep kvm
```

δε δώσει κάποια έξοδο σημαίνει ότι τα modules είτε δεν περιλαμβάνονται στον πυρήνα μας είτε ότι απλά δεν έχουν φορτωθεί. Στην πρώτη περίπτωση θα πρέπει να κάνουμε χειροκίνητη εγκατάσταση του KVM αλλά κάτι τέτοιο δεν είναι στα πλαίσια αυτού του μαθήματος και παραλείπεται. Στην δεύτερη περίπτωση αρκεί να δώσουμε ώς root (εντολή su)

```
# modprobe kvm_amd
```

(ή `kvm_intel`, ανάλογα με την αρχιτεκτονική) η οποία θα φορτώσει αυτόματα και το module `kvm`, απ' όπου εξαρτώνται τα `kvm_{amd, intel}`.

Τέλος, με

```
$ lsmod | grep kvm
```

πρέπει να βλέπουμε τα δύο φορτωμένα modules του πυρήνα.

Σε περίπτωση που χρειάστηκε να φορτώσουμε τα modules με το χέρι μπορούμε να ρυθμίσουμε τον πυρήνα να τα φορτώνει κατά την εκκίνηση του υπολογιστή. Για να γίνει αυτό χρειάζεται να προσθέσουμε στο αρχείο `/etc/modules` τις γραμμές `kvm` και `kvm_{intel`, ή `kvm_amd}`. Αυτό μπορεί να γίνει με την εντολή

```
# echo -e "kvm\nkvm_amd" >> /etc/modules
```

Σε αυτό το σημείο αφού έχουμε φορτώσει τα modules του KVM στον πυρήνα μας, πρέπει να βλέπουμε το ειδικό αρχείο `/dev/kvm` το οποίο χρησιμοποιεί όποιο πρόγραμμα θέλει να χρησιμοποιήσει τις επεκτάσεις εικονικοποίησης ώστε να καλέσει τις υπηρεσίες του KVM. Για να σηκώσουμε μία εικονική μηχανή με χρήση QEMU-KVM πρέπει να έχουμε δικαιώματα εγγραφής-ανάγνωσης σε αυτό το αρχείο. Αρχικά, το αρχείο αυτό δημιουργείται με δικαιώματα ανάγνωσης-εγγραφής (`rw`) για το χρήστη `root` και το group `root`. Επειδή θέλουμε για λόγους ασφάλειας κι απομόνωσης οι εικονικές μας μηχανές να εκτελούνται με δικαιώματα χρηστών εκτός του `root`, θα δημιουργήσουμε ένα group `kvm`, αν δεν υπάρχει, στο οποίο θα προσθέσουμε τον χρήστη με τον οποίο εμείς χρησιμοποιούμε το σύστημα (π.χ. `user`), και θα ρυθμίσουμε το αρχείο `/dev/kvm` να δημιουργείται με δικαιώματα στο group αυτό. Αυτό γίνεται ως εξής:

```
$ su
# groupadd kvm
# usermod -aG kvm user
# echo KERNEL==\"kvm\", GROUP=\"kvm\" >> \
> /etc/udev/rules.d/40-permissions.rules
# modprobe -r kvm_amd
# modprobe -r kvm
# modprobe kvm_amd
# exit
$ groups
user adm cdrom sudo dip plugdev lpadmin sambashare
```

Επειδή μόλις προσθέσαμε τον χρήστη `user` στο group `kvm`, ο τρέχων φλοιός του χρήστη εκτελείται χωρίς δικαίωμα στο group `kvm`. Για να αποκτήσει δικαίωμα στο group kvm, ο απλούστερος τρόπος είναι ο χρήστης να αποσυνδεθεί (logout) και να συνδεθεί εκ νέου. Οπότε ανήκει στο group, κι έχει δικαίωμα εγγραφής στο `/dev/kvm`:

```
$ groups
user adm cdrom sudo dip plugdev lpadmin sambashare kvm
$ ls -l /dev/kvm
crw-rw----+ 1 root kvm 10, 232 Mar  9 19:53 /dev/kvm
```

3.3 Εγκατάσταση του QEMU

Το QEMU είναι διαθέσιμο από τα repositories των περισσότερων διανομών αλλά σε αυτήν την ενότητα θα δείξουμε πως μπορούμε να κάνουμε τη μεταγλώττιση του κατευθείαν από τον πηγαίο κώδικά του, καθώς θα μας φανεί χρήσιμο και για επόμενες εργαστηριακές ασκήσεις. Για το σκοπό αυτό, κατεβάζουμε τον πηγαίο κώδικα (www.qemu.org), τον αποσυμπιέζουμε και τον μεταγλωττίζουμε. Στα επόμενα, υποθέτουμε ότι χτίζουμε την έκδοση QEMU 3.0.0:

```
$ wget https://download.qemu.org/qemu-3.0.0.tar.xz
$ tar xvJf qemu-3.0.0.tar.xz
$ cd qemu-3.0.0
$ ./configure --prefix=/path/to/qemu/build/dir --enable-kvm \
 > --target-list=x86_64-softmmu
$ make
$ make install
```

Ως μονοπάτι εγκατάστασης του QEMU επιλέγουμε ένα μονοπάτι στο οποίο ο χρήστης μας έχει δικαίωμα εγγραφής, π.χ. το `/home/user/opt/qemu`, δημιουργώντας το αν χρειαστεί εκ των προτέρων.

Το χτίσιμο του QEMU θα πάρει λίγη ώρα. Αν όλα πάνε καλά, με το τέλος διαδικασίας στο `/path/to/qemu/build/dir/` θα έχουμε το build tree του QEMU. Σε αυτό το σημείο το μηχάνημα το οποίο θα χρησιμοποιηθεί ως host για τις εικονικές μηχανές μας είναι έτοιμο.

4 Χρήση του QEMU - εικονική μηχανή ανάπτυξης

Για τη δική σας ευκολία, σας παρέχονται δύο αρχεία για να να ξεκινήσετε τη λειτουργία της εικονικής σας μηχανής για *ανάπτυξη* του κώδικά σας: (i) `utopia.sh`, το οποίο εκκινεί την εικονική μηχανή, και (ii) `utopia.config`, το οποία περιέχει τις απαραίτητες ρυθμίσεις ώστε να μπορέσει να εκκινήσει σωστά η εικονική μηχανή. Το script `utopia.sh` διαβάζει τις παραμέτρους στο `utopia.config` και ξεκινά 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 ενεργοποιήσετε για να τρέξει επιτυχώς το `utopia.sh`. Το `utopia.config` περιέχει στην ουσία μία σειρά από μεταβλητές περιβάλλοντος που χρησιμοποιεί το `utopia.sh` κατά τη λειτουργία του. Οι μεταβλητές αυτές είναι:

`QEMU_BUILD_DIR` Αυτός είναι ο κατάλογος στον οποίο έχει γίνει η εγκατάσταση του QEMU. Αν ο κατάλογος υπάρχει, το `utopia.sh` θα ψάξει εκεί για το εκτε-

λέσιμο `bin/qemu-system-x86_64`, το οποίο και θα χρησιμοποιήσει για να εκκινήσει την εικονική μηχανή.

`QCOW2_BACKING_FILE` Αυτό είναι το αρχικό Image που θα χρησιμοποιήσει η εικονική μηχανή για να τρέξει ένα πλήρες σύστημα Linux. Στο filesystem που σας παρέχεται στο εργαστήριο, υπάρχει εγκατεστημένο Debian 10 (Buster). Τίποτε δεν γράφεται σε αυτό το αρχείο. Όλες οι αλλαγές σας, καταλήγουν στο αρχείο `$QCOW2_PRIVATE_FILE`, δηλ. στο `private.qcow2` Σβήνοντάς το, η μηχανή επαναφέρεται στην αρχική της κατάσταση.

Για να ξεκινήσετε, κατεβάστε το root filesystem της εικονικής σας μηχανής από το site του μαθήματος, έστω `cslab_rootfs_20211022.raw.gz`, αποσυμπιέστε το, με χρήση του εργαλείου `gunzip`, και μετακινήστε το σε φάκελο της επιλογής σας, έστω `~/utopia`:

```
$ mkdir ~/utopia
$ wget http://www.cslab.ece.ntua.gr/courses/compsyslab/files/\
 > 2021-22/cslab_rootfs_20211022.raw.gz
$ gunzip -k cslab_rootfs_20211022.raw.gz
$ mv cslab_rootfs_20211022.raw ~/utopia
```

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

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

Η εικονική μηχανή θα ξεκινήσει και θα τυπωθούν διαγνωστικά μηνύματα σχετικά με το πώς μπορείτε να συνδεθείτε σε αυτήν. Η εικονική μηχανή που σας δίνεται έχει ρυθμισμένους δύο χρήστες, τους `user` και `root`, με passwords `user` και `root` αντίστοιχα.

Υπάρχουν οι εξής τρόποι σύνδεσης στην εικονική σας μηχανή:

- Με VNC viewer, από το φυσικό μηχάνημα, ή το εξωτερικό VM:
  `vncviewer localhost:0`.
  Είναι αργός τρόπος, αλλά διευκολύνει πολύ το debugging, γιατί μπορείτε να δείτε την εκκίνηση της μηχανής από την πρώτη στιγμή. *Υπόδειξη:* Πατήστε F8 στο παράθυρο του VNC viewer για να μπορέσετε να στείλετε τον συνδυασμό πλήκτρων Ctrl-Alt-Del στην εικονική μηχανή.

- Με ssh από τον host / εξωτερική εικονική μηχανή, στο port 22223, με χρήση της εντολής `ssh -p 22223 root@localhost`. Η εσωτερική εικονική μηχανή ακούει στο TCP port 22223 του localhost (εξωτερικού VM).

- Με χρήση του `x2goclient` από την εξωτερική μηχανή στην εσωτερική μηχανή, πάλι με σύνδεση ssh στο port 22223 του localhost.

Μπορείτε να αλλάξετε τη μεταβλητή `UTOPIA_SSH_INTERFACE` στο `utopia.config`, ώστε η εικονική μηχανή ανάπτυξης να επιτρέπει συνδέσεις από οπουδήποτε. Με το τρόπο αυτό θα μπορείτε να συνδέεστε απευθείας προς αυτή, από οποιοδήποτε εξωτερικό μηχάνημα, στη θύρα 22223 του εξωτερικού μηχανήματος, π.χ. `snf-xxxx.vm.okeanos.grnet.gr`.

ΠΡΟΣΟΧΗ: Βεβαιωθείτε ότι έχετε θέσει *ισχυρά* passwords και για τους *δύο* χρήστες, `user` και `root`, πριν ενεργοποιήσετε μια τέτοια επιλογή.

Ακριβώς όπως και για ένα πραγματικό μηχάνημα, η λειτουργία της εικονικής μηχανής πρέπει να τερματίζεται σωστά. Ως `root`, χρησιμοποιήστε 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, δίνοντας Ctrl-C στο τερματικό που εκτελείται το `utopia.sh`. Αν δεν υπάρξει αποτέλεσμα, εκτελέστε στο host `killall qemu-system-x86_64` ή `killall -9 qemu-system-x86_64`.

Για να κρατάτε τα αρχεία σας έξω από την εικονική μηχανή και να 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), συνιστούμε να χρησιμοποιήσετε το σύστημα αρχείων sshfs. Το sshfs επιτρέπει πρόσβαση σε κατάλογο απομακρυσμένου μηχανήματος, προσαρτώντας τον στο τοπικό μηχάνημα με χρήση του πρωτοκόλλου ssh. Ουσιαστικά, κάθε πρόσβαση στο τοπικό σημείο προσάρτησης μεταφράζεται σε λειτουργίες Εισόδου/Εξόδου πάνω από κρυπτογραφημένο κανάλι ssh.

Θα το χρησιμοποιήσουμε ως εξής: Στην εικονική μηχανή ανάπτυξης, θα προσαρτήσουμε τον κατάλογο `/home/<host_user>/` του host, στο σημείο `/home/user/host` της εσωτερικής εικονικής μηχανής. Έτσι, μπορείτε να κρατάτε τα αρχεία σας στο `/home/user/host/ask1` της εσωτερικής εικονικής μηχανής, που θα είναι το `/home/<host_user>/ask1` στον host.

Για την προσάρτηση καταλόγου του host, μπορείτε στην εσωτερική μηχανή να εκτελέσετε (αντικαθιστώντας την `<host_user>` με έναν υπαρκτό χρήστη του host):

```
# echo "user_allow_other" >> /etc/fuse.conf
$ mkdir /home/user/host
$ sshfs -o allow_other <host_user>@10.0.2.2:/home/<host_user> \
 > /home/user/host
```

Η παραπάνω διαδικασία υποθέτει ότι στον host είναι ενεργοποιημένη η πρόσβαση μέσω SSH. Αν όχι, χρειάζεται να εγκαταστήσετε (`apt-get install`) το πακέτο `openssh-server`.

Έτσι διευκολύνεται και η μεταφορά αρχείων από και προς την εικονική μηχανή. Σας προτείνουμε να κρατάτε ολόκληρο τον πηγαίο κώδικα της άσκησης σε κάποιο directory το οποίο είναι mounted μέσω sshfs στον host έτσι ώστε αν αναγκαστείτε να επαναφέρετε την εικονική μηχανή στην αρχική της κατάσταση διαγράφοντας το τοπικό COW αρχείο (`private.qcow2`), τα αρχεία αυτά να μην επηρεαστούν.

ΠΡΟΣΟΧΗ: Για τις ανάγκες της 2ης άσκησης το επόμενο βήμα της μεταγλώττισης νέου πυρήνα είναι προαιρετικό. Αντί να μεταγλωττίσετε το δικό σας πυρήνα, μπορείτε να εγκαταστήσετε, με `apt-get` το πακέτο `linux-headers-$(uname -r)` που αντιστοιχεί στον πυρήνα που τρέχετε μέσα στην εικονική μηχανή. Για παράδειγμα:

```
# uname -r
...
# apt-cache show linux-headers-$(uname -r)
...
# apt-get install linux-headers-$(uname -r)
```

Το πακέτο αυτό εγκαθιστά τα απολύτως απαραίτητα κάτω από τον κατάλογο `/lib/modules/` ώστε να έχετε κατάλληλο `$KERNELDIR` για τη μεταγλώττιση δικών σας kernel modules.

5 (Προαιρετικά) Μεταγλώττιση πυρήνα για το VM

Προαιρετικά, μπορείτε να μεταγλωττίσετε το δικό σας πυρήνα Linux για την εικονική μηχανή του QEMU. Η διαδικασία που θα ακολουθηθεί είναι ακριβώς η ίδια διαδικασία που θα ακολουθούσαμε για τη μεταγλώττιση του πυρήνα για ένα πραγματικό μηχάνημα.

ΠΡΟΣΟΧΗ: Ο πυρήνας αποτελείται από πάρα πολλά αρχεία και η διαδικασία μεταγλώττισης του είναι πολύ επίπονη και χρονοβόρα διαδικασία. Για αυτό το λόγο θα μεταγλωττίσετε τον πυρήνα στο host μηχάνημα και όχι στο utopia VM. Έτσι τα βήματα της ρύθμισης και της μεταγλώττισης του πυρήνα θα γίνουν στο host και στη συνέχεια αφού έχετε συνδεθεί στο VM σας και έχετε κάνει mount τον φάκελο του πυρήνα με sshfs θα εκτελέσετε τις εντολές που αφορούν την εγκατάσταση του νέου πυρήνα.

Συγκεκριμένα, για τη μεταγλώττιση και εγκατάσταση του πυρήνα απαιτούνται τα

επόμενα βήματα:

Λήψη του κώδικα Αρχικά, θα πρέπει να ανακτήσετε τον κώδικα του πυρήνα από την επίσημη ιστοσελίδα του πυρήνα του Linux (www.kernel.org). Ανακτήστε την τελευταία έκδοση. Οι ακόλουθες οδηγίες έχουν δοκιμαστεί με την έκδοση Linux 3.14.3.

```
$ wget http://www.kernel.org/pub/linux/kernel/\
> v3.x/linux-3.14.3.tar.xz
```

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 xf linux-3.14.3.tar.xz
```

Ολόκληρος ο κώδικας του πυρήνα έχει αποσυμπιεστεί στον κατάλογο `linux-3.14.3`, στον οποίο μπορείτε πλέον να μεταβείτε και να περιηγηθείτε ή κάνετε τις αλλάγες σας στον πυρήνα.

Ρύθμιση του πυρήνα Προτού ξεκινήσετε την μεταγλώττιση του πυρήνα, θα πρέπει να πρώτα να κάνετε κάποιες ρυθμίσεις. Ο πιο γρήγορος τρόπος είναι να χρησιμοποιήσετε τις προκαθορισμένες ρυθμίσεις οι οποίες ενεργοποιούνται μέσω της εντολής

```
$ make defconfig
```

Στις περισσότερες περιπτώσεις αυτές οι ρυθμίσεις είναι αρκετές ώστε να έχετε έναν λειτουργικό πυρήνα. Στην περίπτωσή μας, χρειάζεται μικρή τροποποίηση στις προκαθορισμένες ρυθμίσεις. Ο πιο απλός και φιλικός προς το χρήστη τρόπος για να το κάνετε αυτό είναι μέσω της εντολής

```
$ make menuconfig
```

Η εντολή αυτή θα ανοίξει ένα μενού στο τερματικό σας, απ' όπου θα μπορέσετε να κάνετε μία πληθώρα ρυθμίσεων για τον πυρήνα που πρόκειται να μεταγλωττίσετε. Για την εκτέλεση στο utopia, θα χρειαστεί να ενεργοποιήσετε (i) τα modules για την αναγνώριση των δίσκων `virtio` (`virtual IO`), που χρησιμοποιεί το VM σας, (ii) το module που θα σας επιτρέψει να χρησιμοποιήσετε το `sshfs` (FUSE), (iii) το filesystem `devtmpfs`. (iv) τις βελτιστοποίησεις του πυρήνα για καλύτερες επιδόσεις όταν τρέχει ως guest.

Πιο συγκεκριμένα, περιηγηθείτε στο μενού και ενεργοποιήστε τις παρακάτω επιλογές. Έχετε υπόψη πως ορισμένες από αυτές εξαρτώνται από άλλες και δεν εμφανίζονται μέχρι να ενεργοποιηθούν οι εξαρτήσεις τους. Συνεπώς, αν

κάποια επιλογή δεν είναι διαθέσιμη άμεσα, προχωρήστε στις επόμενες και επιστρέψτε για να την ενεργοποιήσετε αργότερα.

```
Virtualization->Host kernel accelerator for virtio net <M>
Virtualization->Virtio balloon driver <M>
Device Drivers->Virtio drivers->PCI driver for virtio devices <M>
Device Drivers->Block Devices->Virtio block driver <M>

File systems->FUSE (Filesystem in Userspace) support <M>

Device Drivers->Generic Driver Options->
  Maintain a devtmpfs filesystem to mount at /dev [*]

Processor type and features->Linux guest support [*]
Processor type and features->Linux guest support->
  Enable paravirtualization code [*]
Processor type and features->Linux guest support->
  Enable paravirtualization code->
    Paravirtualization layer for spinlocks [*]
Processor type and features->Linux guest support->
  KVM Guest support (including kvmclock) [*]
```

Τέλος, πατήστε στο "Exit" και στην επόμενη ερώτηση για την αποθήκευση των ρυθμίσεων απαντήστε καταφατικά.

Μεταγλώττιση του πυρήνα Τώρα είστε έτοιμοι να μεταγλωττίσετε τον πυρήνα!

```
$ make
```

Το βήμα αυτό μπορεί να πάρει αρκετή ώρα. Στο τέλος, θα έχει δημιουργηθεί 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. Εκτός από τον πυρήνα, χρειάζεται μεταγλωττίσουμε τα συνοδευτικά modules του, με την εντολή

```
$ make modules
```

Εγκατάσταση νέου πυρήνα και επανεκκίνηση εικονικής μηχανής Για να ξεκινήσει το μηχάνημα με το νέο πυρήνα που μόλις μεταγλωττίσαμε, χρειάζεται να αντιγράψουμε το συμπιεσμένο binary image του στον κατάλογο `/boot`. Κατά σύμβαση, ο πυρήνας ακολουθείται κι από το αντίστοιχο αρχείο ρυθμίσεών του. Αυτό το βήμα το εκτελείται *μέσα* στην εικονική μηχανή, έχοντας πρόσβαση μέσω sshfs στον κατάλογο στον οποίο έχετε μεταγλωττίσει τον πυρήνα.

```
$ su -
# cd /host/directory/where/you/compiled/the/kernel
# make modules_install
# make install
```

Σε αυτό το σημείο έχουμε αντιγράψει τον πυρήνα στην τελική του θέση. Χρειάζεται επιπλέον να ενημερώσουμε το περιεχόμενου του αρχείου `initrd`, το οποίο αναλαμβάνει την προσάρτηση του ριζικού καταλόγου κατά την εκκίνηση του συστήματος και να ανανεώσουμε τη λίστα επιλογών του bootloader(`grub`), του προγράμματος που αναλαμβάνει την φόρτωση του πυρήνα στη μνήμη κατά την εκκίνηση του συστήματος:

```
$ su -
# mkinitramfs -o /boot/initrd.img-3.14.3 -v 3.14.3
# update-grub
```

Σε αυτό το σημείο και αν όλα έχουν πάει καλά με τη μεταγλώττιση του πυρήνα είμαστε έτοιμοι να εκκινήσουμε το μηχάνημα με το νέο πυρήνα. Κατά την εκκίνηση, ο grub μας ζητάει να επιλέξουμε τον πυρήνα που θα χρησιμοποιηθεί. Αν ο νέος μας πυρήνας αποτυγχάνει, μπορούμε να επαναφέρουμε τον παλιό απλά επιλέγοντας την αντίστοιχη εγγραφή στο μενού του grub.

6 Συχνές ερωτήσεις

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

Ερώτηση:
Τι είναι ο ~okeanos και πως μπορώ να δημιουργήσω μία εικονική μηχανή εκεί;

Απάντηση:
Πληροφορίες σχετικά με την υπηρεσία ~okeanos και τη δημιουργία και χρήση VM εκεί μπορείτε να βρείτε στο site της υπηρεσίας `https://okeanos.grnet.gr/home`. Μπορείτε να βρείτε απαντήσεις σε συχνές ερωτήσεις για την υπηρεσία στο `https://okeanos.grnet.gr/support/faq`, καθώς και οδηγό για τη δημιουργία εικονικής μηχανής:
`https://okeanos.grnet.gr/support/user-guide/cyclades-how-to-create-a-vm`

Ερώτηση:
Πώς μπορώ να συνδεθώ με γραφικό περιβάλλον στην εικονική μου μηχανή;

Απάντηση:
Ακολουθήστε τις οδηγίες σύνδεσης με χρήση του συστήματος X2Go, τις οποίες μπορείτε να βρείτε στον οδηγό χρήσης του ~okeanos:
https://okeanos.grnet.gr/support/user-guide/cyclades-how-do-i-connect-to-a-vm

Ερώτηση:
Πώς μπορώ να δημιουργήσω το δικό μου root filesystem;

Απάντηση:
Αυτή είναι μια αρκετά επίπονη διαδικασία. Αν θέλετε να πειραματιστείτε, μπορείτε να βρείτε οδηγίες στα http://wiki.colar.net/creating_a_qemu_image_and_installing_debian_in_it, http://wiki.debian.org/QEMU.

7 (Προαιρετικά) Μία κλήση συστήματος "Hello, World!" στο Linux

ΠΡΟΣΟΧΗ: Για τις ανάγκες της 2ης άσκησης, το παρόν βήμα της προσθήκης δικής σας κλήσης συστήματος στο Linux είναι προαιρετικό. Ένα σύγχρονο ΛΣ όπως το Linux είναι εύκολα επεκτάσιμο μέσω οδηγών συσκευής που υλοποιούν συγκεκριμένες διεπαφές και σπάνια χρειάζεται η υλοποίηση νέων κλήσεων συστήματος για την προσθήκη νέας λειτουργικότητας.

ΠΡΟΣΟΧΗ: Το παρόν βήμα έχει δοκιμαστεί πλήρως για την έκδοση 2.6.37.4 του πυρήνα του Linux και μπορεί να απαιτεί μικρές αλλαγές για την εκτέλεσή του με νεότερες εκδόσεις του πυρήνα.

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 αντιστοιχίζεται 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 κατάλογο `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
```

(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.

8 Περισσότερες πληροφορίες

Περισσότερα για το QEMU, το KVM, το σχεδιασμό, την εγκατάσταση και τη χρήση τους μπορείτε να μάθετε στις σελίδες

- http://www.linux-kvm.org
- http://www.qemu.org

Περισσότερα στοιχεία για την υπηρεσία ~okeanos και το open source λογισμικό Synnefo που την υποστηρίζει μπορείτε να βρείτε στις σελίδες

- http://okeanos.grnet.gr
- http://www.synnefo.org