Εισαγωγή στο xv6 Kernel & Προσθήκη System Call

Χρήστος Φείδας Ευάγγελος Λάμπρου fidas@upatras.gr e.lamprou@upnet.gr

University of Patras — 2023

1 Εισαγωγή

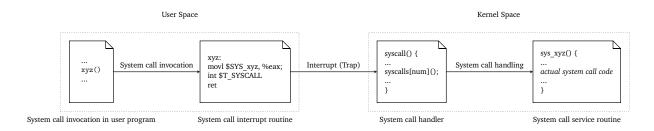
Ο πυρήνας xv6 (xv6 kernel) [2] είναι μια μικρή υλοποίηση ενός πυρήνα λειτουργικού συστήματος βασισμένο στο λειτουργικό σύστημα Unix v6. Είναι ένας πολύ καλός τρόπος αρχάριοι μηχανικοί συστημάτων να εξοικιωθούν με την ανάγνωση και συγγραφή κώδικα λειτουργικών συστημάτων. Ολόκληρο το πρότζεκτ έχει έκταση ~ 6000 γραμμών C, με μέρος του κώδικα γραμμένο σε assembly. Υπάρχει πλούσιο περιεχόμενο online για να μελετήσετε εις βάθος τις λεπτομέρειες των λειτουργιών του πυρήνα xv6 [3, 1].

Στόχος των εργασιών αυτών είναι η εξοικοίωση με μέρη της υλοποίησης ενός πύρηνα και η πλοήγηση μέσα σε μία μεγάλη βάση κώδικα.

2 User Space & Kernel Space

Ο διαχωρισμός μεταξύ χώρου χρήστη (user space) και χώρου πυρήνα (kernel space) είναι ένας από τους σημαντικότερους μηχανισμούς που παρέχει ένα λειτουργικό σύστημα. Ο χώρος χρήστη είναι ο χώρος στον οποίο όλες οι διεργασίες του χρήστη εκτελούνται. Ο χώρος πυρήνα είναι όπου γίνονται όλες οι ευαίσθητες λειτουργίες του συστήματος, όπως η διαχείριση της μνήμης, η διαχείριση των διεργασιών, η διαχείριση των συσκευών κ.α. Ο χώρος πυρήνα είναι προστατευμένος από το χώρο χρήστη και η επικοινωνία μεταξύ των δύο γίνεται από καθορισμένα σημεία επαφής (interfaces). Ένα από αυτά τα σημεία επαφής είναι οι κλήσεις συστήματος (system calls).

Όταν μία διεργασία όπως ένας task manager τρέχει στον υπολογιστή μας, γίνεται μία «συζήτηση» μεταξύ αυτής και του πυρήνα. Η διεργασία ζητάει από τον πυρήνα πληροφορίες για τις διεργασίες και τις συσκευές που υπάρχουν στο σύστημα. Ο πυρήνας απαντάει στην διεργασία με τις πληροφορίες που ζήτησε. Αυτή η συζήτηση γίνεται μέσω των κλήσεων συστήματος. Συνύθως, οι κλήσεις συστήματος είναι «κρυμμένες» πίσω από τη βασική βιβλιοθήκη της εκάστοτε γλώσσας που χρησιμοποιούμε. (π.χ. libc στην C, όπου η συνάρτηση printf καλεί την κλήση συστήματος write).



Σχήμα 1: Η επικοινωνία εφαρμογών χρήστη με το λειτουργικό σύστημα μέσω κλήσεων συστήματος (system calls).

3 System Calls στο xv6

Η επικάλεση ενός system call από μία διεργασία στο xv6 ακολουθεί την εξής ακολουθία:

- 1. Ένα πρόγραμμα χρήστη καλεί μία system call (π.χ read)
- 2. Ο αριθμός που αντιστοιχεί στη συγκεκριμένη system call αποθηκεύεται στον καταχωρητή ΕΑΧ και καλείται στον επεξεργαστή interrupt (trap)
- 3. Το λειτουργικό σύστημα αντιπετωπίζει αυτό το trap διαβάζωντας τον αριθμό του trap το οποίο καλέστηκε. Έπειτα, καλεί την αντίστοιχη system call ανάλογα με τον αριθμό που βλέπει αποθηκευμένο στον καταχωρητή ΕΑΧ.

Παρακάτω φαίνεται ο assembly κώδικας ο οποίος εκτελείται όταν καλείται μία system call από μία διεργασία χρήστη. Ο ορισμός του κώδικα για την κάθε διαφορετική system call γίνεται με τη χρήση της μακροεντολής (macro) SYSCALL.

0

Σημείωση: Ο κώδικας του χν6 που αφορά τα system calls βρίσκεται:

- Ο user-space κώδικας στα αρχεία include/user.h και ulib/usys.S
- O kernel-space κώδικας στα αρχεία include/syscall.h και kernel/syscall.c

4 Προετοιμασία

Για την εκπόνηση των εργασιών προτείνεται να δουλέψετε σε μηχάνημα που τρέχει Linux. Συγκεκριμένα, για την ελαχιστοποίηση των τεχνικών προβλημάτων προτείνεται να χρησιμοποιήσετε το Debian Linux distribution.

4.1 Προτεινόμενο Διάβασμα

- Διαβάστε από το βιβλίο του xv6 [1] το κεφάλαιο 1 (Operating system organisation) και το κεφάλαιο
 3 (Traps, interrupts, and drivers) μέχρι την ενότητα Drivers.
- Δείτε το επεξηγηματικό βίντεο το οποίο εξηγεί πώς καλείται μία system call στο χν6 μέσα από μία εφαρμογή χρήστη.
- Δείτε ένα επεξηγηματικό βίντεο για το πώς λειτουργούν οι system calls στο xv6.

0

Σημείωση Στα δοθέντα βίντεο, ο κώδικας που παρουσιάζεται από τη RISC-V έκδοση του χν6. Η δικιά μας έκδοση είναι για την αρχιτεκτονική χ86. Οι διαφορές μεταξύ των δύο εκδόσεων είναι πρακτικά αμελητέες, με μόνες εξαιρέσεις τα σημεία του πηγαίου κώδικα γραμμένα σε assembly όπως οι ονοματολογίες των καταχωρητών (registers).

4.2 Προετοιμασία Συστήματος

Για να σετάρετε μία εικονική μηχανή που τρέχει Debian Linux, μπορείτε να ακολουθήσετε αυτό τον οδηγό. Αφού είστε μέσα στο Linux σύστημά σας, πρέπει να κατεβάσετε ορισμένα πακέτα που θα χρειαστείτε για την εκπόνηση των εργασιών.

Command Line

\$ sudo apt-get install build-essential gdb git qemu qemu-utils qemu-system-x86

Κατά τη διάρκεια εκπόνησης των εργασιών συνίσταται να κρατάτε εκδόσεις της δουλειάς σας χρησιμοποιώντας το git ώστε να γίνεται πιο εύκολα ο εντοπισμός λαθών μεταξύ των διαφόρων εκδόσεων του κώδικά σας. Μία σύντομη εισαγωγή για το εγαλείο μπορείτε να βρείτε εδώ.

Μέσα στον φάκελο xv6 θα βρείτε τον πηγαίο κώδικα του xv6. Ο κώδικας είναι οργανωμένος σε φακέλους με τον εξής τρόπο:

- kernel: Κώδικας του πυρήνα.
- include: Αρχεία κεφαλίδων που χρησιμοποιούνται από τον πυρήνα και τις εφαρμογές.
- ulib: Βασικές βιβλιοθήκες που γρησιμοποιούνται από τις εφαρμογές γρήστη.
- user: Κώδικας εφαρμογών χρήστη. Αυτές είναι εφαρμογές όπως αυτές που τρέχουν σε user space και αλληλεπιδρούν με τον πυρήνα μέσω των κλήσεων συστήματος.
- tools: Εργαλεία που χρησιμοποιούνται για την εκτέλεση του xv6 (π.χ. mkfs το οποίο χρησιμοποιείται για τη δημιουργία του image από το οποίο θα κάνει boot το λειτουργικό σύστημα.).

5 Ασκήσεις

Άσκηση 1

Βρείτε το αρχείο στον πηγαίο κώδικα του χνό όπου ορίζονται οι κλήσεις συστήματος οι οποίες θα είναι διαθέσιμες σε προγράμματα χρήση. Έπειτα, διαλέξτε ένα από τα προγράμματα χρήστη στον φάκελο user. Εξηγήστε τη λειτουργία του κάνοντας συγκεκριμένες αναφορές στα system calls που καλεί.

Άσκηση 2

Δημιουργήστε ένα πρόγραμμα χρήστη για το xv6.

Μέσα στο φάκελο user, δημιουργήστε ένα αρχείο με ό'τι όνομα θέλετε. Εκεί, γράψτε ένα πρόγραμμα σε γλώσσα C το οποίο θα κάνει κάτι χρήσιμο. Δώστε προσοχή σε ποιες συναρτήσεις έχετε πρόσβαση. Εδώ, δεν μπορείτε να χρησιμοποήσετε τη βασική βιβλιοθήκη της C που ξέρετε αλλά τη βασική βιβλιοθήκη που έχει δημιουργηθεί για το xv6 (ulib.c).

Έπειτα, επεξεργαστείτε κατάλληλα το αρχείο Makefile ώστε να προστεθεί το πρόγραμμα στο image του χν6 κατά την κατασκευή του (πρέπει να προσθέσετε το όνομα του προγράμματός σας στη λίστα

USER_PROGS).

Εκτλέστε make qemu και χρησιμοποιήστε το πρόγραμμα που δημιουργήσατε μέσα στο xv6.



Σημείωση: Μπορείτε να βρείτε τις συναρτήσεις στις οποίες έχετε πρόσβαση σε περιβάλλον χρήστη στο αρχείο include/user.h.

Άσκηση 3

Αναζητώντας μέσα στον πηγαίο κώδικα του χν6, βρείτε την κλήση συστήματος uptime.

- 1. Βρείτε όλα τα σημεία του κώδικα του πυρήνα στα οποία γίνεται αναφορά σε αυτή.
- 2. Εξηγήστε την υλοποίησή της.

Άσκηση 4

Βρείτε τη συνάρτηση μέσα στον kernel κώδικα όπου τελικά εκτελούνται οι system calls. Καταγράψτε τη σειρά των ενεργειών που γίνονται όταν καλείται ένα system call.

Άσκηση 5

Προσθέστε μία κλήση συστήματος η οποία θα επιστρέφει τον αγαπημένο σας αριθμό. Η κλήση που θα φτιάξετε πρέπει να έχει το εξής πρωτότυπο: int getfavnum(void). Μία τέτοια κλήση δεν έχει κανέναν πρακτικό σκοπό αφού δεν εκμεταλεύεται κάποια πληροφορία ή πόρο που κατέχει το λειτουργικό σύστημα. Στόχος εδώ είναι να κατανοήσουμε τον τρόπο με τον οποίο προστίθενται κλήσεις συστήματος στο χν6.

Άσκηση 6

Προσθέστε μία κλήση συστήματος η οποία θα εκτελεί λειτουργία «shutdown». Η κλήση που θα φτιάξετε πρέπει να έχει το εξής πρωτότυπο: void halt(void). Έπειτα, φτιάξτε ένα πρόγραμμα χρήστη με όνομα shutdown το οποίο θα καλεί την κλήση συστήματος που φτιάξατε, ώστε χρήστες του χνό να μπορούν πλέον να κάνουν shutdown το σύστημα χωρίς να χρειάζεται να το κάνουν από το host μηγάνημα.

Σε αυτή τη κλήση συστήματος θα πρέπει να επικοινωνήσετε άμεσα με το hardware της εικονικής μηχανής (στην περίπτωσή μας το QEMU). Για την απενεργοποίηση του συστήματος το hardware περιμένει σε κάποια συγκεκριμένγ διεύθυνση (port) να υπάρξει μία συγκεκριμένη τιμή. Οι αριθμοί για αυτά τα δύο ορίζονται από τον κατασκευαστή. Οδηγίες για την τιμή που πρέπει να στείλετε σε ποια πόρτα έχουν καταγραφεί για διάφορους emulators εδώ. Εξηγήστε τη λειτουργία της συνάρτησης ουτων την οποία θα χρησιμοποιήσετε.



Σημείωση: Η συνάρτηση outw είναι ήδη υλοποιημένη στο χν6 και βρίσκεται στο αρχείο include/x86.h.

Άσκηση 7

Προσθέστε μία κλήση συστήματος η οποία θα επιστρέφει το πόσες φορές έχει εκτελεστεί μία συγκεκριμένη κλήση συστήματος η οποία θα περνιέται σαν όρισμα. Η κλήση που θα φτιάξετε πρέπει να έχει το εξής πρωτότυπο: int getcount(int syscall).

Άσκηση 8

Προσθέστε μία κλήση συστήματος η οποία θα τερματίζει μια τυχαία διεργασία με πρότυπο int killrandom(void).

Θα χρειαστεί να προσθέσετε μία γεννήτρια τυχαίων αριθμών στο kernel. Πώς μπορείτε να βρείτε τα PID όλων των τρέχοντων διεργασιών στο σύστημα;



Σημείωση: Για να σκοτώσετε μία διεργασία δωθέντος ενός συγκεκριμένου PID μπορείτε να χρησιμοποιήσετε την κλήση συστήματος kill.

Αναφορές

- [1] Russ Cox, Frans Kaashoek και Robert Morris. *The Design of the xv6 x86 Operating System*. 2019. URL: https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev10.pdf.
- [2] Russ Cox, Frans Kaashoek και Robert Morris. xv6, a simple Unix-like teaching operating system. MIT, 2012. URL: https://pdos.csail.mit.edu/6.828/2022/xv6.html.
- [3] Harry H. Porter III. xv6 video series. 2021. URL: https://www.youtube.com/playlist?list=PLbtzT1TYeoMhTPzyTZboW_j7TPAnjv9XB.