Χρόνο-προγραμματισμός Διεργασιών στο xv6 Kernel Λειτουργικά Συστήματα (ECE ΓΚ702)

Χρήστος Φείδας
fidas@upatras.gr

Ευάγγελος Λάμπρου
e.lamprou@upnet.gr

University of Patras — 2023

Περιεχόμενα

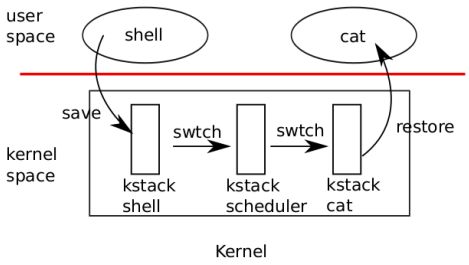
1 Εισαγωγή

Ο χρονοπρογραμματισμός διεργασιών είναι μία από τις σημαντικότερες αρμοδιότητες ενός λειτουργικού συστήματος. Στόχος είναι η κάθε διεργασία να "νομίζει" πως έχει ένα δικό της CPU στο οποίο εκτελούνται σειριακά οι εντολές της. Αν αυτό ήταν αλήθεια βέβαια, θα χρειαζόμασταν n επεξεργαστικές μονάδες για να εκτελέσουμε n διεργασίες ταυτόχρονα στον υπολογιστή μας. Μέσω της πολύπλεξης διεργασιών γίνεται να φαίνεται πως τρέχουν ταυτόχρονα στον υπολογιστή μας πολύ περισσότερες διεργασίες απ'ότι οι πυρήνες που κατέχει το σύστημά μας.

Στο xv6 η πολύπλεξη αυτή γίνεται με την CPU να εναλλάσεται από τη μία διεργασία στην άλλη σε περιπτώσεις 1. Μηχανισμός sleep και wake [3]. 2. CPU time interrupt σε περίπτωση που μία διεργασία εκτελείται για μεγάλο χρονικό διάστημα.

Εδώ φαίνεται πως το λειτουργικό σύστημα δεν είναι απλά ένα πρόγραμμα το οποίο εκτελείται σειριακά στον επεξεργαστή, αλλά ένα σύστημα το οποίο αξιοποιεί λειτουργίες του υλικού (interrupts) για να επανέρχεται σε εκτέλεση, διακόπτωντας την τώρα εκτελούμενη διεργασία και επιστρέφοντας σε αυτήν όταν πρέπει.

2 Context Switching



Σχήμα 1: Context switch από μία διεργασία χρήστη (`shell`) προς μία ρουτίνα πυρήνα (`scheduler`) και τελικά σε μία άλλη διεργασία χρήστη (`cat`).

Μία προσιτή εισαγωγή στο μηχανισμό context switching του επεξεργαστή μπορείτε να βρείτε εδώ.

3 Scheduler

Ο χρονοπρογραμματιστής (scheduler) είναι το κομμάτι του κώδικα του πυρήνα το οποίο είναι υπεύθυνο στο διαμερισμό του χρόνου του επεξεργαστή ανάμεσα στις ενεργές διεργασίες του συστήματος. Ο σχεδιασμός ενός scheduler πρέπει να λάβει πολλά πράγματα υπόψιν όπως η ταχύτητα, το είδος του φόρτου εργασίας που θα δέχεται το σύστημα στο οποίο τρέχει (π.χ. ένας υπολογιστής οικιακής χρήσης πιθανότατα θα έχει διαφορετικές ανάγκες από έναν server), ακόμα και κατανάλωση ενέργειας. Μία σύντομη αναδρομή της ιστορίας του scheduler στο Linux Kernel μπορείτε να βρείτε εδώ και εδώ (βίντεο).

4 Προετοιμασία

Θα συνεχίσετε πάνω στον πηγαίο κώδικα της προηγούμενης εργασίας. Για αρχή, ενώ είστε μέσα στο φάκελο `xv6`, εκτελέστε τις παρακάτω `git` εντολές ώστε να φτιάξετε ένα νέο branch πάνω στο οποίο θα κάνετε τις νέες αλλαγές. Αυτό θα το κάνει εύκολο να πάτε πίσω στο αρχικό branch σε περίπτωση λαθών.

Command Line

```
$ git chechout -b scheduling
```

4.1 Προτεινόμενο Διάβασμα

- Διαβάστε από το βιβλίο του xv6 [2] το κεφάλαιο 5 (*Scheduling*)
- Δείτε το επεξηγηματικό βίντεο για μία λεπτομερή παρουσίαση της διαδικασίας scheduling στο xv6.

5 Ασκήσεις

Άσκηση 1

Πώς αναπαρίσταται μία διεργασία στο xv6 kernel; Εξηγήστε τη σημασία καθενός από τα members του struct το οποίο αντιπροσωπεύει μία διεργασία στο σύστημα.

Σημείωση: Κοιτάξτε στο αρχείο `kernel/proc.h`.

Άσκηση 2

Ποια είναι η συνάρτηση `scheduler`, ποια είναι η συνάρτηση `sched`. Εξηγήστε τη λειτουργία της καθεμίας.

Άσκηση 3

Τροποποιήστε το xv6 kernel ώστε να υποστηρίζει την πολιτική προγραμματισμού Lottery Scheduling [1, 4]

Το Lottery Scheduling θα λειτουργεί ως εξής:

Μία διεργασία θα ξεκινά την εκτέλεσή της με ένα συγκεκριμένο αριθμό εισητηρίων (`tickets`), αυτό μπορεί να είναι ένας οποιοδήποτε αριθμός θέλετε. Ο scheduler κάνει μια κλήρωση για την κάθε διεργασία. Αν κάποια από αυτές "κερδίσει", θα ξεκινήσει η εκτέλεσή της. Αυτή η διαδικασία θα επαναλαμβάνεται για κάθε κλήση στον scheduler.

Ο αριθμός εισητηρίων μιας διεργασίας μπορεί να αλλάξει με τη χρήση της system call `settickets(int n)` που θα πρέπει να υλοποιήσετε. Όταν ο χρήστης εκτελεί αυτή τη system call, η τρέχουσα διεργασία θα αλλάζει αριθμό εισητηρίων με βάση το όρισμα. Η τρέχουσα διεργασία που εκτελείται στον τρέχοντα πυρήνα (core) βρίσκεται στην global μεταβλητή `proc` μέσα στο kernel.

Τη διαδικασία της κλήρωσης μπορείτε να την υλοποιήσετε με διάφορους τρόπους. Μια απλή προσέγγιση είναι ο scheduler να υπολογίζει έναν τυχαίο αριθμό μεταξύ 0 και `total_tickets`. Έπειτα, για την κάθε διεργασία να γίνεται έλεγχος αν ο αριθμός των εισητηρίων της είναι μεγαλύτερος από αυτό το νούμερο. Αν ναι, κερδίζει την κλήρωση.

Θα χρειαστεί επίσης να προσθέσετε μία γεννήτρια τυχαίων αριθμών στο kernel.

Για να καταλάβετε αν λειτουργεί σωστά ο scheduler σας μπορείτε να χρησιμοποιήσετε το έτοιμο πρόγραμμα `sched-test`. Αυτό θα σας δώσει στατιστικά σχετικά με το χρόνο εκτέλεσης της κάθε διεργασίας. Εξηγήστε γιατί η υλοποίησή σας είναι σωστή με βάση τα αποτελέσματα. Επεξεργαστείτε τον κώδικα στο αρχείο `user/sched-test.c` αλλάζωντας τον αριθμό των εισητηρίων που δίνονται στην κάθε διεργασία. Εξηγήστε τα αποτελέσματα της εκτέλεσης.

Βοήθεια: Τα σημεία του πηγαίου κώδικα που θα πρέπει να επεξεργαστείτε είναι τα εξής:

- Συνάρτηση `scheduler` στο αρχείο `kernel/proc.c` ώστε να γίνεται η επιλογή της διεργασίας που θα εκτελεστεί σε κάθε κβάντο με βάση τον νικητή της κλήρωσης.
- Συνάρτηση `fork` στο αρχείο `kernel/proc.` μπορείτε να έχετε μία διεργασία να κληρονομεί τα εισητήρια από το γονέα της.
- Τα αρχεία `kernel/sysproc.c`, `kernel/syscall.c`, `include/sysproc.h`, `include/user.h`, `ulib/usys.S` για την προσθήκη της απαραίτητης κλήσης συστήματος ώστε να ελέγχεται ο αριθμός εισητηρίων μιας διεργασίας από κώδικα χρήστη.
- Συμπληρώστε τον κατάλληλο κώδικα στο αρχείο `include/random.h` για την υλοποίηση της γεννήτριας τυχαίων αριθμών.

Φροντίστε να δώσετε στην διεργασία `init` (PID 1) ένα τουλάχιστον εισητήριο.

Αναφορές

[1] Remzi H. Arpaci-Dusseau και Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. 2023. Κεφ. Scheduling: Proportional Share. URL: http://www.cs.wisc.edu/~remzi/OSTEP/cpu-sched-lottery.pdf.

[2] 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.

[3] Denning Institute. *Sleep and Wakeup*. URL: http://denninginstitute.com/modules/ipc/red/sleep.html.

[4] Carl A. Waldspurger και William E. Weihl. «Lottery Scheduling: Flexible Proportional-Share Resource Management.» Στο: *OSDI*. USENIX Association, 1994, σσ. 1–11. URL: http://dblp.uni-trier.de/db/conf/osdi/osdi94.html#WaldspurgerW94.