Φροντιστήριο: 16/11/2016
Παράδοση: 2/12/2016

Implementation of "Least Remaining Time First" scheduling policy in the Linux Operating System

Σε αυτή την άσκηση σας ζητείτε να υλοποιήσετε τον αλγόριθμο "least-remaining-time-first" στον scheduler του λειτουργικού συστήματος Linux. Σύμφωνα με τον αλγόριθμο αυτό, σε κάθε context switch ο scheduler θα επιλέγει διεργασίες σύμφωνα με το Least Remaining Time. Για την υλοποίηση του αλγορίθμου, θα χρησιμοποιήσετε το system call **"set_total_computation_time"** που υλοποιήσατε στην προηγούμενη άσκηση. Τέλος, θα πρέπει να δοκιμάσετε τις αλλαγές σας στον scheduler με ένα demo πρόγραμμα.

1. Τροποποιήσεις στον Linux Kernel

Σε αυτήν την άσκηση, θα χρησιμοποιήσετε τον qemu emulator και το disk image της προηγούμενης άσκησης για να ξεκινήσετε το λειτουργικό σύστημα. Τις οδηγίες για το πώς θα κάνετε compile τον Linux kernel και πως μπορείτε να κάνετε boot το image, μπορείτε να τις ξαναδείτε στην περιγραφή της προηγούμενης άσκησης. Στην υλοποίηση της άσκησης αυτής θα ξεκινήσετε χρησιμοποιώντας τον κώδικα σας από την άσκηση 3.

Ο κώδικας του Linux scheduler είναι στο αρχείο linux-source-2.6.38.1/kernel/sched.c. Η κυριότερη συνάρτηση του Linux scheduler είναι η συνάρτηση *asmlinkage void __sched* ***schedule****(void)*. Οι τροποποιήσεις που πρέπει να κάνετε στον scheduler είναι οι εξής:

1. Κάθε φορά που ο scheduler κάνει context switch θα κάνει update το *remaining_time* της διεργασίας που έτρεχε
2. Αν το *remaining_time* πάρει αρνητική τιμή, σημαίνει ότι η διεργασία, δήλωσε λανθασμένο **total_computation_time** και έτσι το *infite* flag θα "σηκωθεί" για τη διεργασία
3. Τέλος, η επόμενη διεργασία που θα τρέξει θα πρέπει να είναι αυτή με το μικρότερο *remaining_time.* Αν η διεργασία που έτρεχε έχει και πάλι το μικρότερο remaining time, θα συνεχίσει να τρέχει
4. Αν όλες οι διεργασίες έχουν infinite, τότε ο scheduler μπορεί να πάρει την διεργασία που θα έτρεχε κανονικά (πριν τις αλλαγές σας)

2. Δοκιμή

Για την δοκιμή του scheduler σας μπορείτε να υλοποιήσετε τα εξής πρόγραμμα. Τα προγράμματα αρχικά θα πρέπει να κάνουν set το **total_computation_time** χρησιμοποιώντας το system call που υλοποιήσατε στην προηγούμενη άσκηση. Ο χρόνος που θα θέτει το κάθε πρόγραμμα θα πρέπει να είναι διαφορετικός (καλό θα είναι η διαφορά του χρόνου να είναι αρκετά μεγάλη, το πρώτο 10, το δεύτερο 20…). Έπειτα, κάθε

διεργασία θα πρέπει να κάνει busy waiting (π.χ. με ένα μεγάλο loop, όχι με sleep καθώς έτσι θα γίνει reschedule). Η επαναλήψεις του loop καλό θα είναι να είναι ίδιες για κάθε πρόγραμμα. Μετά το loop κάθε διεργασία θα πρέπει να τυπώνει ένα μοναδικό αναγνωριστικό (pid, "programA"… "programB",…). Επειδή οι διεργασίες ξεκινάνε με infinite flag μέχρι να κάνουν set το total_computation_time, η διεργασία που θα τερματίσει πρώτη θα είναι αυτή που θα προλάβει να κάνει πρώτη το set_total_computation_time. Μπορείτε να ξεκινάτε τις διεργασίες με αρχικό total_computation_time (στο task_struct π.χ. 2) έτσι ώστε να προλαβαίνουν να κάνουν set ένα μεγαλύτερο.

Μπορείτε να φτιάξετε και δικά σας προγράμματα αρκεί να δικαιολογήσετε την ορθότητα τους στο να αποδεικνύουν τη σωστή λειτουργεία του scheduler σας. Αναφέρετε και δικαιολογείστε τι παρατηρείτε, στην αναφορά σας.

Links:

- Το Linux Cross Reference θα σας βοηθήσει να περιηγηθείτε στον source code του Linux Kernel.
  http://lxr.free-electrons.com/source/kernel/?v=2.6.38
- Μερικές λεπτομέρειες για τον scheduler της έκδοσης Linux που θα δουλέψετε
  http://www.ibm.com/developerworks/library/l-completely-fair-scheduler/

Τι πρέπει να παραδώσετε:

1. Το καινούργιο kernel image, δηλαδή το αρχείο linux-2.6.38.1/arch/x86/boot/bzImage.

2. Όλα τα αρχεία που τροποποιήσατε ή δημιουργήσατε στον source code του Linux kernel 2.6.38.1 για να υλοποιήσετε το scheduling policy. Δηλαδή όλα τα αρχεία .c, .h, Makefile κτλ που κάνατε κάποια αλλαγή, ή δημιουργήσατε εσείς. Μην παραδώσετε αρχεία που δεν χρειάστηκε να τα τροποποιήσετε για την υλοποίησή σας.

3. Τον source code από όλα τα test προγράμματα που γράψατε και τρέξατε μέσα στο guest Linux OS για να δοκιμάσετε το scheduling policy υλοποιήσατε. Και επίσης ότι header files χρησιμοποιήσατε για type και function definitions (π.χ. το unist.h). Δηλαδή τα αρχεία .c, .h και Makefile και ότι άλλο αρχείο δημιουργήσατε στο guest OS για να δοκιμάσετε τις αλλαγές σας.

4. Ένα README file στο οποίο να περιγράφετε συνοπτικά (αλλά περιεκτικά και ξεκάθαρα) όλα τα βήματα που ακολουθήσατε για την δημιουργία του scheduling policy. Επίσης πρέπει να σχολιάσετε τι παρατηρήσατε από τα test προγράμματα που τρέξατε. Αν έχετε κάνει κάτι διαφορετικό ή παραπάνω από όσα αναφέρουμε στην εκφώνηση της άσκησης σε οποιοδήποτε βήμα μπορείτε επίσης να το αναφέρετε στο README. Καλό θα ήταν το README να είναι από 20 μέχρι 30 γραμμές.