Φροντιστήριο: 25/04/2018 Παράδοση: 16/05/2018

Implementation of "Lowest Demand First" scheduling policy in the Linux Operating System

Σε αυτή την άσκηση σας ζητείτε να υλοποιήσετε τον αλγόριθμο "lowest-demand-first" στον scheduler του λειτουργικού συστήματος Linux. Θεωρήστε ότι ο χρόνος είναι χωρισμένος σε δευτερόλεπτα Στην αρχή κάθε δευτερολέπτου ο πυρήνας διαλέγει να εκτελέσει την διεργασία με την μικρότερη απαίτηση και την εκτελεί μέχρι να εξαντλήσει την απαίτηση της για το τρέχον δευτερόλεπτο Αν η απαίτηση μιας διεργασίας είναι 10 τότε η διεργασία ζητά το 10% του χρόνου ενός πυρήνα. Αν η απαίτηση μιας διεργασίας είναι 1 τότε ζητά το 1% του χρόνου ενός πυρήνα. Μετά την εκτέλεση της η διεργασία δεν θα ξαναπάρει χρόνο στο CPU μέχρι το επόμενο δευτερόλεπτο 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 επιλέγει τη διεργασία με το μικρότερο demand_time που δεν έχει εκτελεστεί κατά το τρέχον δευτερόλεπτο.

2. Επίσης θα πρέπει να οριστεί ο χρόνος που θα γίνει το επόμενο context switch.

3. Τέλος, στο επόμενο context switch θα πρέπει η διεργασία που έτρεχε να σημειωθεί ως ανενεργή για το τρέχον δευτερόλεπτο

4. Μετά το πέρας του δευτερολέπτου, όλες οι ανενεργές διεργασίες σημειώνονται ξανά ως ενεργές.

Σημείωση: Αν δεν υπάρχουν διεργασίες με καθορισμένο demand_time ή όλες έχουν ολοκληρώσει την εκτέλεση τους για το τρέχον δευτερόλεπτο, τότε οι διεργασίες που θα εκτελεστούν είναι αυτές που θα αποφάσιζε ο linux scheduler.

2. Δοκιμή

Για την δοκιμή του scheduler σας μπορείτε να υλοποιήσετε τα εξής πρόγραμμα. Τα προγράμματα αρχικά θα πρέπει να κάνουν set το demand_time χρησιμοποιώντας το system call που υλοποιήσατε στην προηγούμενη άσκηση (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). Οι επαναλήψεις του loop καλό θα είναι να είναι ίδιες για κάθε πρόγραμμα και να διαρκούν αρκετά δευτερόλεπτα Τέλος ο scheduler θα εκτυπώνει (με χρήση printk) για τις διεργασίες που έχει οριστεί το demand_time της, το pid και το demand_time της σε κάθε context switch, καθώς και όποτε ενεργοποιεί ξανά τις διεργασίες που έχουν τρέξει μόλις περάσει το κάθε δευτερόλεπτο.

Μπορείτε να φτιάξετε και δικά σας προγράμματα αρκεί να δικαιολογήσετε την ορθότητα τους στο να αποδεικνύουν τη σωστή λειτουργία του scheduler σας. Αναφέρετε και δικαιολογήστε 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.