HY345 – Άσκηση 4

Παράδοση Άσκησης: έως Παρασκευή, 11/12/2020 (23:59)
Φροντιστήριο Άσκησης: Παρασκευή, 20/11/2020 (10:00)

Implementation of the "Least Slack Time (LST)" scheduling policy in the Linux Operating System

Σε αυτή την άσκηση σας ζητείται να υλοποιήσετε τον αλγόριθμο "Least Slack Time" στον scheduler του kernel του λειτουργικού συστήματος Linux. ***Για την υλοποίηση της άσκησης αυτής θα ξεκινήσετε χρησιμοποιώντας τον κώδικα σας από την άσκηση 3.***

Θεωρήστε ότι ο χρόνος είναι χωρισμένος σε κβάντα. Στην αρχή κάθε κβάντου ο scheduler σκοτώνει όλες τις διεργασίες που έχουν ξεπεράσει το deadline τους. Για όλες τις υπόλοιπες διεργασίες, ο scheduler διαλέγει πρώτη εκείνη που έχει τη μεγαλύτερη προτεραιότητα. Η προτεραιότητα της κάθε διεργασίας χαρακτηρίζεται από τον χρόνο τον οποίο μπορεί να καθυστερήσει ακόμα, πριν χάσει τη διορία για την εκτέλεση της. Αυτός ο χρόνος ονομάζεται *slack time*. Όσο μικρότερο είναι το *slack time* μιας διεργασίας, τόσο μεγαλύτερη είναι η προτεραιότητα της να εκτελεστεί και τόσο μεγαλύτερη η πιθανότητα της να γίνει schedule.

Συγκεκριμένα, για κάθε διεργασία, ο kernel θα κρατάει ένα structure με τις εξής πληροφορίες, το οποίο πρέπει να ανανεώνεται ανά κβάντο:

- Το deadline της διεργασίας (Deadline)
- Το συνολικό χρόνο για την εκτέλεση της διεργασίας (Computation time)
- Το συνολικό χρόνο που έχει εκτελεστεί η διεργασία (Elapsed runtime)
- Τον εναπομείναντα χρόνο για την εκτέλεση της διεργασίας (Remaining time, R = C - E)
- Τον χρόνο που μπορεί να καθυστερήσει η εκτέλεση της διεργασίας πριν να χάσει τη διορία/deadline (Slack time, S = D – R – current_time)

Δεδομένων N διεργασιών με slack times S_1, S_2, S_3, …, S_N, η διεργασία «i» θα εκτελεστεί με πιθανότητα $(1/S_i)/(1/S_1 + 1/S_2 + 1/S_3 + … + 1/S_N)$.

Τέλος, θα πρέπει να δοκιμάσετε τις αλλαγές που κάνατε στον scheduler με μερικά demo προγράμματα, τα οποία χρειάζεται να κάνετε submit μαζί με τον κώδικα σας.

Τροποποιήσεις στον Linux Kernel

Σε αυτή 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 βρείτε ξανά στην εκφώνηση της προηγούμενης άσκησης. Για την υλοποίηση της άσκησης αυτής θα ξεκινήσετε χρησιμοποιώντας τον κώδικα σας από την άσκηση 3.

Ο κώδικας του Linux scheduler βρίσκεται στο αρχείο linux-source-2.6.38.1/kernel/sched.c και η κυριότερη συνάρτηση του Linux scheduler, δηλαδή αυτή που θα τροποποιήσετε κυρίως, είναι η: **asmlinkage void __sched schedule(void)**.

Τα βήματα που πρέπει να ακολουθήσετε τροποποιόντας τον kernel περιλαμβάνουν τα εξής:

1. Ο context switching μηχανισμός εκτελείται σε συγκεκριμένα κβάντα χρόνου.
2. Ο μηχανισμός αυτός θα ενεργοποιείται στην αρχή κάθε κβάντου.
3. Σε κάθε κβάντο, ο scheduler θα αναζητά όλες τις διεργασίες που έχει περάσει το deadline τους και θα τις σκοτώνει.
4. Ο context switch μηχανισμός θα υπολογίζει σε κάθε κβάντο το slack time της κάθε διεργασίας και θα ανανεώνει τις υπόλοιπες πληροφορίες κάθε διεργασίας. Για κάθε διεργασία, το slack time της υπολογίζεται ως εξής: ***slack_time = deadline – (computation_time – elapsed_runtime) – current_time***.
5. Στη συνέχεια θα επιλέγει τη διεργασία εκείνη με το μικρότερο slack time (Least Slack Time scheduling policy).

Σημείωση: Αν το slack time είναι μικρότερο ή ίσο του μηδέν ή το deadline έχει λήξει τότε η διεργασία αυτή δεν θα θεωρείτε υποψήφια για να επιλεγεί για εκτέλεση στην συνάρτηση schedule() και θα επιλεγεί μία άλλη διεργασία. Αν δεν υπάρχει καμία άλλη διεργασία έτοιμη για εκτέλεση τότε καμία διεργασία δεν θα πρέπει να εκτελεστεί στον επεξεργαστή. Έτσι, οι διεργασίες που έχουν slack time μικρότερο ή ίσο του μηδέν ή έχει λήξει το deadline τους δεν θα πρέπει να εκτελούνται ούτε όταν δεν υπάρχουν άλλες διεργασίες έτοιμες για εκτέλεση.

Δοκιμή του scheduler

Θα πρέπει να φτιάξετε (τουλάχιστον) ένα demo πρόγραμμα που να δείχνει ότι οι αλλαγές που κάνατε στον Linux scheduler δουλεύουν σωστά. Συγκεκριμένα, το demo πρόγραμμα θα έχει την δυνατότητα να δέχεται ένα όρισμα *X* από command line, το οποίο *X* θα πρέπει να είναι ακέραιος αριθμός με τιμές από 2 μέχρι 10. Σε περίπτωση που δεν δοθεί όρισμα, η default τιμή για το *X* θα είναι το 2. Στη συνέχεια, θα δημιουργεί *X* θυγατρικές διεργασίες και για κάθε μία από αυτές, η πατρική διεργασία θα θέτει με το system call *set_deadlines* τον υπολειπόμενο χρόνο (computation_time) και το deadline του ίσο με gettimeofday()+100 seconds. Επιπρόσθετα, κάθε θυγατρική διεργασία θα πρέπει να κάνει spin για κάποιο χρόνο (μπορείτε να χρησιμοποιήσετε κάποια while ή for). Συζητήστε στο README σας το πρόγραμμά σας. Εκτός από αυτό το demo πρόγραμμα, μπορείτε να φτιάξετε και να παραδώσετε όσα περισσότερα θέλετε, έτσι ώστε να βεβαιωθείτε ότι οι αλλαγές σας στον Linux scheduler δουλεύουν σωστά.

Παραδοτέα αρχεία

1. Το καινούριο kernel image, δηλαδή το αρχείο linux-2.6.38.1/arch/ x86/boot/bzImage.
2. ΄Ολα 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. Το source code από όλα τα test προγράμματα που γράψατε και τρέξατε μέσα στο guest Linux OS για να δοκιμάσετε το scheduling policy που υλοποιήσατε. Επιπλέον ό,τι header files χρησιμοποιήσατε για type και function definitions (π.χ., το unistd.h). Δηλαδή τα αρχεία .c, .h και Makefile καθώς και ότι άλλο αρχείο δημιουργήσατε στο guest OS για να δοκιμάσετε τις αλλαγές σας.
4. ΄Ενα README στο οποίο να περιγράφερε συνοπτικά (αλλά περιεκτικά και ξεκάθαρα) όλα τα βήματα που ακολουθήσατε για την δημιουργια του scheduling policy. Επίσης πρέπει να σχολιάσετε τι παρατηρήσατε απο τα test προγράμματα που τρέξατε. Αν έχετε κάνει κάτι διαφορετικό ή παραπάνω απο 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.

Βοηθητικό υλικό

- Μην ξεχνάτε να συμβουλεύεστε το φροντιστήριο της Άσκησης 4.
- Μπορείτε να διαβάσετε περισσότερα για τον Linux scheduler εδώ: http://www.ibm.com/developerworks/library/l-completely-fair-scheduler/
- Περιηγηθείτε στον κώδικα του Linux Kernel με μερικά clicks από τον browser σας: https://elixir.bootlin.com/linux/v2.6.38.1/source
- Ctags: http://www.tutorialspoint.com/unix_commands/ctags.htm
- Χρησιμοποιήστε τη συνάρτηση printk(). Τα μηνύματα που τυπώνετε στον kernel με την συνάρτηση printk μπορείτε να τα βλέπετε όταν έχετε φορτώσει το Linux με το συγκεκριμένο kernel τρέχοντας το dmesg ή κάνοντας cat /var/log/messages.