ΗΥ-345 Λειτουργικά Συστήματα
Χειμερινό Εξάμηνο 2023
Άσκηση 4

Implementation of the “Group Fairness” Scheduling Policy in the linux Operating System

Φροντιστήριο: 05/12/2023
Παράδοση: 22/12/2023

Εισαγωγή

Ο πυρήνας του λειτουργικού συστήματος Linux περιέχει έναν χρονοπρογραμματιστή (Scheduler) ο οποίος αποφασίζει ποια θα είναι η επόμενη διεργασία η οποία θα εκτελεστεί στον επεξεργαστή του υπολογιστή (CPU). Ο χρονοπρογραμματιστής παίρνει αποφάσεις σύμφωνα με την πολιτική χρονοπρογραμματισμού (Scheduling Policy) η οποία βοηθάει στην αποδοτική χρήση του επεξεργαστή. Στην άσκηση αυτή θα υλοποιήσετε μια νέα πολιτική χρονοπρογραμματισμού ως μέρος του λειτουργικού συστήματος Linux.

Group Fairness

Στην άσκηση αυτή σας ζητείται να υλοποιήσετε τον αλγόριθμο χρονοπρογραμματισμού “Group Fairness”. Σύμφωνα με τον αλγόριθμο αυτό, κάθε διεργασία ανήκει σε ένα group και εκτελείται για ένα ποσοστό που της αναλογεί το οποίο καθορίζεται από τον αριθμό των groups αλλά και των αριθμό των διεργασιών στο εκάστοτε group.
Σύμφωνα με τον αλγόριθμο “Group Fairness” όλες οι διεργασίες οφείλουν να δηλώνουν το group στο οποίο ανήκουν. Κάθε group παίρνει ίσο ποσοστό του συνολικού χρόνου. Στη συνέχεια ο χρόνος εκτέλεσης της κάθε διεργασίας υπολογίζεται με βάση τον αριθμό των διεργασιών που βρίσκονται στο group. Συγκεκριμένα, αν έχουμε 2 groups, A και B, με τις διεργασίες A1, A2 και A3 στο group A και τις διεργασίες B1 και B2 στο group B, ο scheduler πρέπει να βεβαιωθεί ότι η διεργασία A1 θα πάρει 100/2/3 = 16.6% του χρόνου, η A2 θα πάρει 100/2/3 = 16.6% και η A3 θα πάρει 100/2/3=16.6% του συνολικού χρόνου του επεξεργαστή. Από την άλλη η B1 και B2 θα πάρουν η κάθε μια 25% του συνολικού χρόνου που ισοδυναμεί με 100/2/2.
Η εξίσωση που υπολογίζει το χρόνο εκτέλεσης της κάθε διεργασίας είναι:

T(process_params, number_of_groups) =
100/number_of_groups/number_of_processes_in_group(process_params.group_name)

Όπου:

T(): χρόνος εκτέλεσης διεργασιας
process_params: παράμετροι διεργασίας

number_of_groups: συνολικός αριθμός των groups
number_of_processes_in_group: επιστρέφει τον αριθμό των διεργασιών στο group που μας ενδιαφέρει την εκάστοτε στιγμή

Παράδειγμα Εκτέλεσης

1. Η διεργασία A1 ξεκινάει να εκτελείται και εισάγεται στο Group A. Καθώς το group A είναι το μόνο group και η διεργασία A1 η μόνη διεργασία θα πάρει το 100% του επεξεργαστή.
2. Η διεργασία A2 ξεκινάει να εκτελείται και εισάγεται στο group A. Καθώς τώρα στο group A υπάρχουν 2 διεργασίες η καθεμία θα πάρει 50% του επεξεργαστή
3. Η διεργασία B1 ξεκινάει να εκτελείται και εισάγεται στο group B. Καθώς τώρα έχουμε 2 groups κάθε group θα πάρει 50% του επεξεργαστή. Αρα η A1 και A2 εκτελούνται για 25% και η B1 για 50% από το χρόνο του επεξεργαστή.

Τροποποιήσεις στον πυρήνα του Linux

Για την άσκηση αυτή, θα χρησιμοποιήσετε τον κώδικα της άσκησης 3 ως βάση. Επιπλέον, θα χρησιμοποιήσετε τον emulator QEMU καθώς και το virtual disk image που χρησιμοποιήσατε στην άσκηση 3. Οδηγίες σχετικά με την μεταγλώττιση του Linux Kernel και την χρήση του virtual disk image μπορείτε να βρείτε στην εκφώνηση της προηγούμενης άσκησης.
Για την άσκηση αυτή χρειάζεται να τροποποιήσετε τον πυρήνα του λειτουργικού συστήματος και να υλοποιήσετε την νέα πολιτική χρονοπρογραμματισμού. Για την σωστή λειτουργία της πολιτικής αυτής θα γίνει χρήση των system calls που υλοποιήσατε στην προηγούμενη άσκηση. Η κυριότερη συνάρτηση του Linux Scheduler καθώς και το σημείο εισόδου είναι η συνάρτηση void __sched schedule(void) στο αρχείο kernel/sched.c.

Υλοποιηση

Στόχος της άσκησης είναι υλοποιήσετε τον ζητούμενο αλγόριθμο χρονοπρογραμματισμού. Η υλοποίηση που θα κάνετε βρίσκεται στην κρίση σας και δεν υπάρχει μια σωστή υλοποίηση.
Ακολουθούν κάποιες χρήσιμες δομές και συναρτήσεις που μπορεί να σας βοηθήσουν.

File	Entity	Description
Include/linux/sched	struct task_struct	Process descriptor. Each process is represented as such a struct. It offers all the information about one particular task (i.e. process) such as pid, state, parent process, children, opened files, etc
	struct rq	Runqueue. It is the main data structure in process scheduling. It manages active processes by holding

		the tasks that are in a runnable state at any given moment of time.
	struct sched_entity	CFS works with more general entities than tasks. This struct contains attributes for accounting run time of processes.
	struct sched_class	The current Linux scheduler has been designed with an extensible hierarchy of modules in mind. These modules encapsulate scheduling policy details. Scheduling classes are implemented through the sched_class structure, which contains hooks to the functions that implement the policy.
Kernel/sched.c	schedule(void)	Main function of the Linux scheduler. Responsible for implementing the process scheduling functionality.
	void context_switch(...)	Performs the actual context switch operation by switching from the old task_struct to the new one.
	Pick_next_task(…)	Selects task_struct of the next process that will run on the processor. Iterates over the list of processes in the runnable state.

Επιπλέον, μπορείτε να εξετάσετε το αρχείο kernel/sched_rt.c αλλά και το kernel/sched_fair.c που υλοποιούν το Real-Time Scheduling Class και τον Completely Fair Scheduler αντιστοίχως. Οι υλοποιήσεις τους μπορεί να σας βοηθήσουν να καταλάβετε πως λειτουργεί ο Linux scheduler αλλά και πως γίνεται το process management. Τέλος, στο αρχείο include/linux/time.h υπάρχουν διάφορα structs για μέτρηση χρόνου καθώς και διάφορες συναρτήσεις για μετατροπές μεταξύ μονάδων μέτρησης.

Παρατηρησεις

- Φροντίστε στην υλοποίηση σας να μην γίνονται starve οι υπόλοιπες διεργασίες του συστήματος. Αυτό μπορείτε να το επιτύχετε είτε θέτοντας κατάλληλες παραμέτρους

μέσω των system calls είτε εναλλάσσοντας μεταξύ policies που επιλέγονται στον scheduler

- Για την άσκηση αυτή θα χρησιμοποιήσετε τον Linux kernel 2.6.38.1 Μπορείτε να χρησιμοποιήσετε το Elixir platform για να περιηγηθείτε στον κώδικα του Linux Kernel.
- Για να πάρετε το task_struct της τρέχουσας διεργασίας που έκανε το system call μπορείτε κοιτάξτε στο αρχείο arch/x86/include/asm/current.h
- Ο Linux kernel αποθηκεύει τις αναλυτικές πληροφορίες για όλες τις τρέχουσες διεργασίες σε μία λίστα από task_struct objects. Για να προσπελάσετε όλες τις διεργασίες του συστήματος στη λίστα αυτή, μπορείτε να χρησιμοποιήσετε το macro for_each_process.
- Χρησιμοποιήστε την συνάρτηση printk για να ελέγξετε τη σωστή λειτουργία της υλοποίησης σας. Για να δείτε τα μηνύματα αυτά μπορείτε να χρησιμοποιήσετε το dmesg ή να εκτελέσετε την εντολή “cat /var/log/messages”.

Demo Programs

Πρέπει να φτιάξετε και να παραδώσετε τουλάχιστον ένα δοκιμαστικό πρόγραμμα το οποίο θα κάνει χρήση των system calls της προηγούμενης άσκησης και θα επιδεικνύει την σωστή υλοποίηση του νέου αλγόριθμου χρονοπρογραμματισμού. Ενδεικτικά, αναφέρονται κάποιες περιπτώσεις που μπορείτε να ελέγξετε.

1. Δημιουργήστε ένα απλό πρόγραμμα το οποίο καλεί το system call set_task_params (από την προηγούμενη άσκηση) και ορίζει τις παραμέτρους χρονοπρογραμματισμού.
2. Δημιουργήστε ένα απλό πρόγραμμα το οποίο ορίζεις τις παραμέτρους του και βεβαιωθείτε ότι η δική σας υλοποίηση καλείται σωστά όταν εντοπίζει μια τέτοια διεργασία.
3. Δημιουργήστε πολλαπλά processes που ορίζουν διαφορετικές παραμέτρους η κάθε μια και βεβαιωθείτε ότι η υλοποίηση σας δουλεύει σωστά και επιλέγει σωστά ποια διεργασία θα εκτελεστεί πρώτα ανάλογα με την προτεραιότητα τους.
    a. Οι διεργασίες σας πρέπει να απαιτούν την χρήση του επεξεργαστή (spinning) και να μην βρίσκονται σε sleep state. Αυτό μπορείτε να το πετύχετε με την χρήση ενός loop και αριθμητικών πράξεων.
4. Δημιουργείστε πολλαπλά processes και ορίσετε έτσι τις παραμέτρους ώστε μια διεργασία να μην προλάβει να εκτελεστεί μέσα στην προθεσμία της. Ελέγξτε ότι όταν περάσει η προθεσμία της ο χρονοπρογραμματιστής την τερματίζει.
5. Για κάθε δοκιμαστικό πρόγραμμα που θα δημιουργήσετε, χρησιμοποιήσετε τα κατάλληλα prints τα οποία θα δείχνουν τις παραμέτρους που έχει ορίσει η κάθε διεργασία καθώς και πότε ξεκινάει να κάνει spin. Τα prints αυτά θα σας βοηθήσουν να καταλάβετε εάν η υλοποίηση σας παίρνει τις σωστές αποφάσεις.

Παράδοση

Αφού κάνετε την άσκηση θα πρέπει να παραδώσετε τα παρακάτω:

1. Το καινούργιο kernel image που προέκυψε από τη μεταγλώττιση, δηλαδή το αρχείο linux-2.6.38.1/arch/x86/boot/bzImage.
2. Όλα τα αρχεία που χρειάστηκε να τροποποιήσετε ή να δημιουργήσετε στον source code του Linux kernel για να υλοποιήσετε τα system calls. Αυτό σημαίνει ότι θα παραδώσετε όλα τα αρχεία .c, .h, Makefile, κλπ στα οποία κάνατε οποιαδήποτε αλλαγή ή δημιουργήσατε εσείς. Προσοχή, μην παραδώσετε αρχεία που δεν χρειάστηκε να τα τροποποιήσετε για την υλοποίησή σας (π.χ. όλο το υπόλοιπο source tree του kernel).
3. Τον κώδικα από όλα τα test προγράμματα που γράψατε και τρέξατε μέσα στο guest Linux OS για να δοκιμάσετε τα system calls που υλοποιήσατε. Επίσης, ότι header files χρησιμοποιήσατε για type και function definitions αλλά και ότι Mekefiles χρειάζονται για την μεταγλώττιση των προγραμμάτων αυτών. Δεν χρειάζεται να παραδώσετε τα executable αρχεία.
4. Ένα README file στο οποίο να περιγράφετε συνοπτικά (αλλά περιεκτικά και ξεκάθαρα) όλα τα βήματα που ακολουθήσατε για την προσθήκη και υλοποίηση των νεων system calls. Επίσης, πρέπει να σχολιάσετε τι παρατηρήσατε από τα test προγράμματα που τρέξατε. Αν έχετε κάνει κάτι διαφορετικό ή παραπάνω από όσα αναφέρονται στην εκφώνηση της άσκησης σε οποιοδήποτε βήμα μπορείτε επίσης να το αναφέρετε στο README. Λόγω της πολυπλοκότητας της άσκησης αυτής, προτείνεται να αναφέρετε στο README και όποιες προσπάθειες υλοποίησης κάνατε ακόμα κι αν αυτές δεν σας οδήγησαν κάπου ή δεν δούλευαν σωστά.
5. Μπορείτε να φτιάξετε έναν κατάλογο με τα τροποποιημένα αρχεία του kernel (αν θέλετε θα είναι καλό να κρατήσετε και την δομή των αρχείων μέσα στον πυρήνα) καθώς και έναν κατάλογο με τα test προγράμματα και header files από το guest OS.

Προσοχή:

1. **ΔΕΝ** χρειάζεται να παραδώσετε το disk image (hy345-linux.img) ακόμα και αν αυτό έχει τροποποιηθεί. Όντως, το disk image μπορεί να αλλάξει όσο χρησιμοποιείτε το guest OS αλλά δεν χρειάζεται να το παραδώσετε.
2. **ΔΕΝ** χρειάζεται να παραδώσετε κάποιο αρχείο με ολόκληρο τον source code του Linux kernel. Πρέπει να σημειώσετε και να παραδώσετε μόνο τα αρχεία που τροποποιήσατε ή δημιουργήσατε. Το kernel image (bzImage), τα source και header files καθώς και τα test προγράμματα που θα παραδώσετε θα πρέπει να είναι αρκετά ώστε η άσκησή σας να μπορεί να τρέξει με το αρχικό disk image και το QEMU έτσι ώστε να φαίνεται η σωστή υλοποίηση της άσκησης.