# סיכום החלק הרטוב

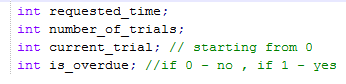
## טיפול במדיניות SCHED\_SHORT:

נחלק את הטיפול לפי קבצים:

### \usr\src\linux-2.4.18-14custom\include\linux\sched.h

קובץ זה אחראי להגדרות הקשורות לניהול משימות בגרעין.

* בtask\_sturct הוספנו שדות המשמרים את תכונות התהליך כתהליך ממדיניות short:

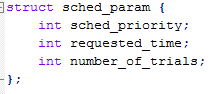
  
    


בנוסף אפסנו את כל השדות הללו במאקרו INIT\_TASK.

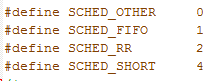
* ייצוא פונקציות ההכנסה וההוצאה של תהליכים מהתורים בהם מורצים תהליכים במערכת. ייצוא זה נעשה על מנת לעמוד בדרישה כי בעת ביצוע fork, יוותר אב ממדיניות sohrt מיידית על המעבד.



* עדכון הstruct- : sched\_param, זאת על מנת לאפשר למשתמש להגדיר תהליך קיים כshort תוך שימוש בפונקציה sched\_setscheduler:



* הגדרה של מדיניות חדשה ברשימת הקבועים המגדירים מדיניות:



### \usr\include\bits\sched.h

שינוי קובץ זה נועד על מנת לאפשר למשתמש ליצור בעצמו מבנה מסוג sched\_param וכן על מנת לזהות את קבוע המדיניות החדשה.

### \usr\src\linux-2.4.18-14custom\kernel\sched.c

בקובץ זה נעשו רב השינויים – הוא האחראי בפועל לבחירת התהליכים הרצים וסדר ריצתם.  
נחלק את ההסבר לקובץ זה לפי הפונקציות והמבנים אותם שינינו בקובץ:

* המאקרו SHORT\_TIMESLICE – מחשב את משך הקוואנטום שיש לתת לתהליך short בעת היווצרו או לאחר סיום ריצתו בepoch הנוכחי.
* שינוי מבני בrunqueue: הוספת תורים שיאכלסו את תהליכי הshort- וה-overdue\_short שרצים במערכת











בנוסף, נרחיב את אתחול הarrays וכן נאתחל את התורים short\_q ו overdue בפונקציה sched\_init.

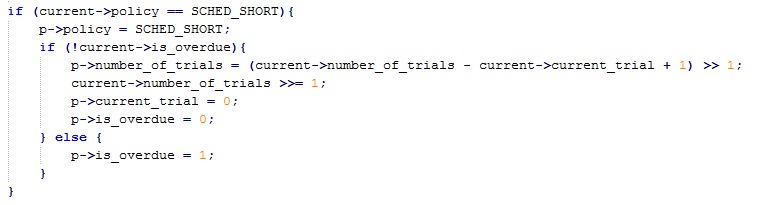
* **effective\_prio** : הוספנו טיפול במקרה בו התהליך מסוג short כך שהdynamic priority של SHORT היא הstatic priority שלו, ו שהdynamic priority של SHORT overdue היא 0. השגנו זאת ע"י קביעת ערכים מתאימים בbonus.
* **: activate\_task**הוספנו עדכון של הprio\_array בrq המתאים לSHORT (short\_q) ול SHORT overdue (overdue) כך שהפונקציה תבצע הכנסה לתור המתאים של התהליכים.
* **: deactivate\_task**הוצאנו מהprio\_array בrq המתאים לSHORT (short\_q) ול SHORT overdue (overdue) את התהליך הנתון.
* **scheduler\_tick** : הוספנו טיפול מתאים במעבר של SHORT לSHORT overdue – כאשר הtime\_slice הוא 0 או כאשר נגמרו לתהליך SHORT ה number\_of\_trials- הוצאה מהתור של SHORT הרגיל לתור של הoverdue. מתעדכנים בנוסף השדות time\_slice וcurrent\_trail בכל כניסה לפונקציה, לשם מעקב אם צריך לעבור לoverdue.
* **schedule** : נוסף חילוק למקרים לפי הpolicy ולפי בחירת עדיפות התהליך (RT,OTHER,SHORT,SHORT overdue) לשם בחירת המקום בqueue המתאים ובחירת תהליך הnext שירוץ.
* **set\_user\_nice** : ביטלנו את האפשרות לשנות את הpriority של תהליך שהוא SHORT, כפי שהוגדר בתרגיל.
* **setscheduler** : הוספנו בדיקה עבור טווח הפרמטרים של requested\_time ו number\_of\_trials לפי ההגדרה בתרגיל. בנוסף הוספנו טיפול עבור תהליך other שרוצה להתחלף עם short - עדכון השדות שלו לפני מעבר לshort תקין.
* **sys\_sched\_getparam** : הוספנו טיפול בשדות של number\_of\_trials ו requested\_time.
* **sys\_sched\_get\_priority\_max** : הוסף טיפול במקרה בו התהליך הוא short שיחזיר 0.
* **sys\_sched\_get\_priority\_min** : הוסף טיפול במקרה בו התהליך הוא short שיחזיר 0.

### \usr\src\linux-2.4.18-14custom\kernel\fork.c

בקובץ זה בפונקציה do\_fork טיפלנו במקרה בו תהליך short מבצע fork. היה עלינו לחשב מחדש את number\_of\_trials הן של הבן והן של האב לפי הנוסחה שניתנה:





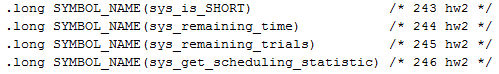


## הגדרת קריאות מערכת

ההגדרה מצויה בשלושה קבצים:

### \usr\src\linux-2.4.18-14custom\arch\i386\kernel\entry.S

בקובץ זה היה עלינו להקצות שלוש קריאות מערכת נוספות:



### \usr\include\hw2\_syscalls.h

בקובץ זה הגדרנו את פונקציות המעטפת האחראיות למעבר שבין user mode לkernel mode.

### \usr\src\linux-2.4.18-14custom\kernel\hw2\_syscalls.c

בקובץ זה הוגדרו ממש פונקציות המערכת, בהן השתמשנו בשדות אותם שמרנו task\_struct ובrunqueue:









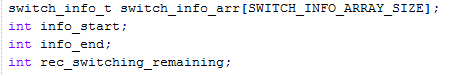
## ניטור המידע אודות context\_switch

מבנה הנתונים:

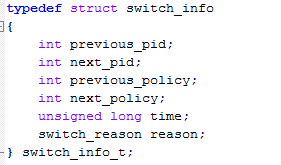
בקובץ usr\src\linux-2.4.18-14custom\kernel\sched.c\ הגדרנו מערך השייך לrunqueue:







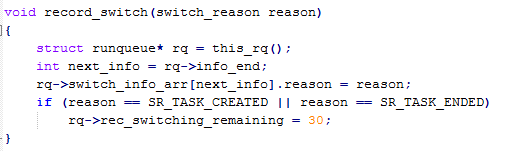
כשswitch\_info\_arr הוא מערך בגודל 150 שיכיל רשומות לגבי החלפות הקשר שנעשות במערכת. הטיפוס switch\_info\_t מוגדר כך:



כלומר, לכל החלפת הקשר נשמרים הpid והpolicy של התהליך הקודם והתהליך הבא, כמו כן את זמן ביצוע החלפת ההקשר והסיבה להחלפת ההקשר.

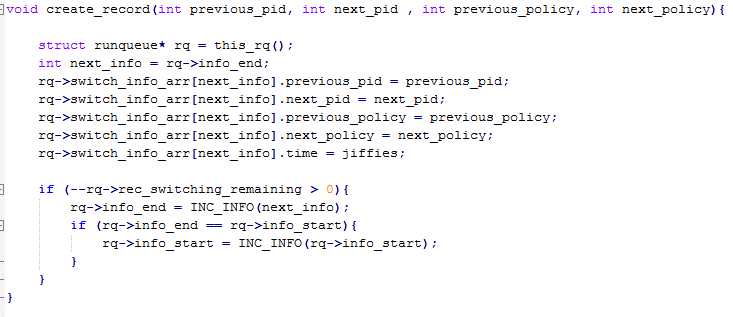
עדכון מבנה הנתונים:

הגדרנו פונקציה אליה ניתן לקרוא מכל קובץ מערכת שעושה include לsched.h:



כך, בכל מקום שלאחריו יש קריאה לschedule, נעדכן את הסיבה הבאה בגינה עתידה אולי להתרחש החלפת הקשר – וכמו כן, נעדכן את מונה החלפות ההקשר אם יצרנו תהליך או אם תהליך מת.

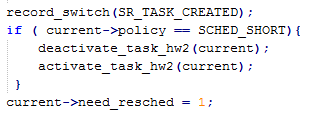
בנוסף הגדרנו פונקציה פנימית בsched.c שנקראת רגע לפני ביצוע החלפת ההקשר ממש:



בזמן הקריאה לפונקציה, קיים כבר מידע לגבי זהות התהליך המוחלף וכן זהותה התהליך המחליף. כמו כן נעדכן את שדה הזמן. רק אם מונה ההחלפות rec\_switching\_remaining לא התאפס נעדכן את האינדקס end כדי שהרשומה שעודכנה כעת לא תידרס בעת החלפת ההקשר הבאה. נבצע את הקידום של info\_end וכן של info\_start באופן ציקלי באמצעות המאקרו INC\_INFO.

ישנם מקומות שונים בקוד מהם עשויה להתרחש קריאה לschedule, בכל מקום מעין זה נעדכן את הסיבה בגינה תתרחש החלפת ההקשר. נביא דוגמה:

בפונקציה do\_fork:



לפני ביצוע פעולות שמהן עשויה להיווצר החלפת הקשר (הדלקת דגל need\_resched או deactivate) נעדכן תחילה את switch\_info\_arr כך שהחלפת ההקשר הבאה תדע שבוצעה בעקבות יצירת תהליך בפונקציה .  
  
  
עמידה בדרישות הסיבוכיות:

**סיבוכיות מקום** : הוספנו 2 prio\_array\_t חדשים, אחד עבור תהליכי short והשני עבור תהליכי short overdue.  
במקרה זה הסיבוכיות נשארת O(n) , מכיוון שהסיבוכיות מקום עבור כל אחד מהתורים שהיו קיימים לפני היא O(n) וגם של 2 התורים שנוספו היא O(n) –ולכן בסהכ הסיבוכיות היא O(n).  
  
switch\_info\_arr, המערך ששומר recordes של context switches, הוא בגודל מקסימלי של 150 ולכן סיבוכיות המקום שלו קבועה ב (1)O.  
  
**סיבוכיות זמן** : עבור הפונקציה **schedule**, בכל פעם שרוצים לבחור את התהליך הבא שירוץ מבצעים בדיקה בתורים של הactive הshort והoverdue אם הם לא ריקים. בדיקה זו נעשית ב בזכות שמירת הbitmap בכל אחד מהprio\_array\_t ובאמצעות הפונקציה sched\_find\_first\_bit. כך נוכל לחלץ את התהליך next הבא שירוץ בO(1) .  
עבור תהליך שמבצע **exit** או **fork** הכנסות והוצאות לrunqueue מתבצעות ב - הוצאות מתחילת התור לפי סדר הגעה, והכנסות אל סוף התור.

עבור תהליך ש**עובר מshort ל short overdue** מתבצעת בחירה של התור החדש שהתהליך ירוץ בו ב-O(1),  
הדלקת דגל ופעולות נוספות בO(1) – סה"כ המעבר נעשה בO(1) .

עבור תהליך **שעובר מother לshort** מתבצע תחילה שינוי השדות (כולל השדה array המתאר את התור אליו שייך) בO(1) , ולאחר מכן מתבצע schedule שמבצע החלפת הקשר בין התהליכים בO(1) כפי שצוין קודם.