TECHNION
Israel Institute of Technology

Computer Science Department

Operating Systems – 234123

Homework Exercise 2 – Dry

Omer Stoler : 318471356

stoler.omer@campus.technion.ac.il

Adi Arbel : 207919614

adi.arbel@campus.technion.ac.il

Teaching Assistant in charge:

Mohammed Dabbah

Assignment Subjects & Relevant Course material

Processes, Scheduler.

Recitations 1-4, Lectures 1-3

שאלה 1

שאלה זו עוסקת במדיניות זימון התהליכים של לינוקס כפי שנלמדה בתרגולים. לנוחיותכם מצורפים מספר macros המשמשים את זמן התהליכים

```
#define MAX_PRIO 140
#define MIN_TIMESLICE (10 * HZ / 1000)
#define MAX_TIMESLICE (300 * HZ / 1000)
#define TASK_TIMESLICE(p)
MIN_TIMESLICE + (MAX_TIMESLICE – MIN_TIMESLICE) * \
(MAX_PRIO – 1 – (p)>static_prio)/39
#define TASK_INTERACTIVE(p) \
((p)>prio <= (p)>static_prio – DELTA(p))
prio = static_prio – bonus
#define EXPIRED_STARVING(rq) \
((rq)>expired_timestamp && \ ((jiffies – (rq)>expired_timestamp)
>=STARVATION_LIMIT * ((rq)>nr_running + 1))
BONUS(p) = 25% × 40 × ( SleepAvg/MaxSleepAvg – 1/2)
DELTA(p) = 5 × TaskNice(p) / 20 + 2
```

א. נניח כי תהליך A מסווג כחישובי על ידי אלגוריתם הזימון של לינוקס ובעל עדיפות סטטית x ותהליך B מסווג כאינטראקטיבי על ידי אלגוריתם הזימון של לינוקס ובעל אותה עדיפות סטטית x. האם יתכן כי העדיפות הדינמית של A טובה יותר משל B?

תהליך נקבע כאינטראקטיבי אמ"ם הביטוי הלוגי $(\Delta \leq \text{bonus})$ מתקיים.

מכיוון ש-$\Delta = 5 \times \frac{nice}{20} + 2$ וכן ש-$nice = static\_prio - 120$, הרי שעקב עדיפות סטטית זהה בין שני התהליכים, Δ זהה עבור תהליכים A ו-B. מכאן שהיות A מסווג כחישובי ו-B מסווג כאינטראקטיבי אומרת ש-$bonus_A < \Delta \leq bonus_B$.

ביוון ש-$prio = static\_prio - bonus$, ומראש נתון ישירות ש-$static\_prio$ שלהם זהה, הרי ש-$prio_B < prio_A$, ומכיוון שכבל ש-$prio$ גבוה יותר, כך העדיפות לריצה יותר נמוכה, הראינו שלא יקרה מצב שבו העדיפות הדינמית של A טובה משל B.

ב. נניח כי תהליכים A ו B מסווגים כחישוביים על ידי אלגוריתם הזימון של לינוקס ובעלי עדיפות דינמית שווה, אבל עדיפות הסטטית של A טובה יותר (נמוכה יותר מספרית). איזה יתרון מקבל A על B?

A ו-B מסווגים כחישוביים אומר ש-$\text{bonus} < \Delta$ בשניהם.

כמו כן העדיפות הדינמית שלהם זהה מה שאומר ש-$static\_prio - bonus$ שווה בין שניהם. אבל העדיפות הסטטית של A טובה יותר – נמוכה יותר מספרית ולכן בהכרח $bonus_A < bonus_B$. זה אומר שהוא ישן פחות מ-B ע"מ להגיע לאותה עדיפות ולכן מרגע שנכנס להמתנה, חוזר לרוץ מהר יותר מאשר אם B היה מבצע את אותה הפעולה.

כמו כן ה-$time\_slice$ של תהליך הוא טרנספורמציה לינארית של ה-$static\_prio$ ולכן ככל שזו יותר נמוכה, הוא יותר גדול, כאשר שני התהליכים באותה העדיפות. זהו יתרון של A ביחס ל-B – קבלת זמן מעבד גדול יותר ב-epoch.

ג. נניח כי תהליכים A ו B מסווגים כאינטראקטיביים על ידי אלגוריתם הזימון של לינוקס ובעלי עדיפות דינמית שווה. אבל העדיפות הסטטית של A טובה יותר (נמוכה יותר מספרית). איזה יתרון מקבל A על B?

תהליכים אינטרקטיביים חוזרים בסיום ה-quantum לסוף הרשימה בתור ה-active שמתאימה לעדיפות הדינמית שלהם. לפי המשוואות למעלה, ניתן להגיע עבור תהליך אינטרקטיבי לאי-השוויון הבא:

$$\frac{1}{4}\text{static prio} - 32 = \Delta \leq bonus$$

ועל כן עבור A שהעדיפות הסטטית שלו נמוכה יותר, קל יותר לעמוד בדרישות הסף ולקבל עוד quantum בסוף ה-epoch.

שימו לב: בסעיף ד, שלושה תת סעיפים (1,2,3)

ד. לפניך קטע מתוך הקוד של פונקציית הגרעין yield_sched_sys אשר מממש את הטיפול בתהליכים עם מדיניות זימון OTHER

```
1.list_del(&current¬>run_list);
2. if(!list_empty(array¬>queue +current¬>prio)){
3.  list_add(&current¬>run_list,array¬>queue[current¬>prio].next);
4.  goto out_unlock;
5. }
6. __clear_bit(current¬>prio, array¬>bitmap);
7. i =sched_find_first_bit(array¬>bitmap); // this would return MAX_PRIO on
fail (in case no set bits found)
8. if(i==MAX_PRIO ||i<=current¬>prio)
9.  i =current¬>prio;
10. else
11.     current¬>prio =i;
12. list_add(&current¬>run_list,array¬>queue[i].next);
13. __set_bit(i,array¬>bitmap);
14. out_unlock:
15. // release locks & call schedule
```

1. בהנחה שקיימים תהליכים נוספים שאינם expired ב runqueue האם יתכן כי ביצוע yield_sched על ידי תהליך עם מדיניות זימון OTHER לא יגרום להחלפת הקשר?

לא קיים מצב כזה כאשר ישנם תהליכים נוספים בתוך תור הריצה. תסריט אחד הוא שקיים תהליך שהגיע והוא בעדיפות גבוהה יותר מן התהליך הקיים. במצב זה הקוד יוציא ויחזיר את התהליך שכעת רץ לסוף התור שלו, אבל schedule תגרום להחלפת הקשר כיוון שהיא תמצא תהליך בעדיפות גבוהה יותר. אם קיימים תהליכים נוספים באותה עדיפות כמו התהליך הקיים, הוא פשוט יחזור לסוף התור שלו והאחרים באותה עדיפות יוחלפו במקומו. אם קיימים רק תהליכים בעדיפות פחותה ממנו והוא היחיד בתור העדיפות שלו, הקוד יוציא אותו ויכניס אותו לסוף התור עם העדיפות הגדולה ביותר שנותר לא ריק לאחר הוצאתו. על כן בכל תסריט בו קיים תהליך נוסף בתור תוודא הפונקציה שהתהליך מוותר על המעבד לטובת אחר.

2. איזו בעיה הייתה נוצרת אם היינו מחליפים את שורה 8 בשורה:

```
if(i<=current>prio)
```

עבור תור ריצה בו יש תהליך אחד, בעת הוצאתו בשימוש בפונקציה sched_yield נקבל שi=140 ולכן לא יקיים את התנאי שהוחלף, לכן יגיע לשורת הקוד בה current->prio=i והיא אינה עדיפות ריצה חוקית בתור הריצה שלנו.

3. איזו בעיה הייתה נוצרת אם היינו מחליפים את שורה 8 בשורה:

```
if(i==MAX_PRIO)
```

עבור תור ריצה ריק אין בעיה עם תנאי זה. עם זאת ברגע שיש תהליך עם עדיפות גבוהה יותר מהתהליך הנוכחי שקורא לפונקציה, אי הוספת החלק השני של התנאי תגרור שינוי של העדיפות הדינמית לעדיפות שנמצאת בתוך i שהיא העדיפות הגבוהה ביותר שנמצאת כרגע בתור הריצה. על כן שימוש בsched_yield יבצע העלאה לא הוגנת של התהליך הנוכחי לרמת ההרשאה הגבוהה ביותר – זה עשוי לגרום לעומס בעדיפות הגבוהה ביותר, מה שירעיב עדיפויות אחרות שלא ביצעו sched_yield. ניתן לעשות שימוש בכך כאסטרטגיה לגזילת זמני ריצה – בכל פעם שמגיע זמן ריצתו של התהליך הוא מבצע sched_yield בסמוך לסוף פרק הזמן שלו וכך מבטיח לעצמו עדיפות מירבית בepoch הבא (ואם אינטרקטיבי עוד במחזור הנוכחי).

שאלה 2

במסגרת הבחינה בקורס, תתבקשו לענות על שאלות הן על החומר הנלמד (בתרגולים ובהרצאות) והן על מערכות שונות ומגוונות אשר לא נלמדו בקורס, דבר הדורש הכללה של החומר ועקרונותיו.

בשאלה זו ננתח מערכת זימון הדומה במהותה ללינוקס, אך במקביל, גם קצת שונה. נתון אלגוריתם SCHED_OTHER במערכת זו:

- עדיפות התהליכים נקבעת על פי השדה **static_prio** אשר טווח ערכיו בין 1 ל 5 וערכו נקבע על ידי המשתמש (1 הוא העדיף ביותר ו 5 הכי פחות עדיף).
- בכל רגע נתון מוגדר **זמן ריצה מינימלי** למשימות בעלות עדיפות i אשר נסמנו q_i או q_i , על פי הנוסחה:

$$q_i = \max\left\{\frac{target\_latency}{N_i}, min\_granularity\right\}$$

כאשר target_latency, min_granulartiy קבועים המוגדרים במערכת, ו- N_i מספר המשימות עם עדיפות i.

- לכל משימה יש שדה vruntime המאותחל ל 0 בעת יצירתה. בכל פסיקת שעון מתעדכן שדה באופן הבא:

$$current \rightarrow vruntime += current \rightarrow static\_prio$$

- בכל פעם שמשימה נבחרת לרוץ נשמר ערכו של השדה vruntime כך:

$$current \rightarrow start\_vruntime = current \rightarrow vruntime$$

- בכל פסיקת שעון נבדקים שני התנאים הבאים:
  - קיימת משימה בעלת vruntime קטן יותר משל המשימה הנוכחית
  - q_{current->static_prio} <= current->vruntime - current->start_vruntime
- אם שני התנאים הללו מתקיימים, מתבצעת החלפת הקשר (בעזרת הדגל need_resched)
- בהחלפת הקשר המשימה הבאה שנבחרת לרוץ היא זאת בעלת ה vruntime המינימלי.

א. בהנחה שהאלגוריתם הנ"ל עושה שימוש בעץ חיפוש מאוזן הממוין לפי vruntime.

1. מהי סיבוכיות הזמן של **בחירת** המשימה הבאה? הסבר.

כיוון שזהו עץ חיפוש בינארי מאוזן, אנו יודעים כי הערך המינימלי יוחזק בעלה השמאלי ביותר בעץ. ניתן להחזיק מצביע לבן השמאלי ביותר בעץ ועל כן הגישה אליו תוכל להתבצע במהירות. בכל שינוי נעדכן את המצביע כך ששליפה של התהליך הבא לריצה תתבצע ב(O(1

2. מהי סיבוכיות הזמן של **הוספת** משימה חדשה? הסבר.

כיוון שהוספה לעץ מאוזן דורשת תיקונים בעץ עשויה להתארך הסיבוכיות. אבל אנו יודעים שעץ מאוזן מבטיח כל תיקון ב (O(1 פעולות ועל כן במעבר במסלול בעץ לנקודה המתאימה שמצריכה הוספת איבר אליה, התיקונים יהיו לכל היותר (O(log(n

3. מהי סיבוכיות הזמן של **הסרת** משימה? הסבר.

כיוון שהסרה מעץ מאוזן דורשת תיקונים בעץ עשויה להתארך הסיבוכיות. אבל אנו יודעים שעץ מאוזן מבטיח כל תיקון ב (O(1 פעולות ועל כן במעבר במסלול בעץ לנקודה המתאימה שמצריכה הסרת איבר ממנה, התיקונים יהיו לכל היותר (O(log(n

ב. נניח שבמערכת יש שני תהליכים CPU-Bound בעלי עדיפות זהה ושהשניים הללו הינם התהליכים היחידים במערכת. איזה אחוז מהזמן ירוץ כל אחד מהתהליכים?

לאור הסימטריה בין התהליכים נצפה שלא יהיה תהליך שיהיה עדיף אחד על פני האחר. נצפה לזמן ריצה של 50% על ידי תהליך אחד וגם על ידי האחר. עם זאת, הזמן בין כל החלפה בין התהליכים תלויה בתנאי השני, מכיוון שעמידה בסף זה תתקיים רק לאחר שעבר פרק זמן מספיק גדול בין הruntime ההתחלתי לנוכחי. בכל מקרה רק כאשר qi יהיה קטן מסף זה, נוכל להגיד שנסתיים זמן הריצה שהוקצה, ומכאן שרק לאחר qi (מעוגל) פסיקות שעון תתבצע כל החלפת הקשר.

ג. נניח שבמערכת שני תהליכים CPU-Bound, אחד בעל עדיפות 1, והשני בעל עדיפות 3. נניח שהשניים הללו הינם התהליכים היחידים במערכת. איזה אחוז מהזמן ירוץ כל אחד מהתהליכים?

ניתן לראות שגודלו של qi עשוי להשתנות כתוצאה מן השינוי בכמות התהליכים בכל עדיפות, אך כיוון שיש רק שני תהליכים הם בעלי ערך זהה ועל כן זה לא יביא להבדלים. ההבדל העיקרי יבוא מקצב הגדילה בכל פסיקת שעון של השדה vruntime , לאור ההבדל בstatic_prio . בכל מצב בו יינתן לתהליך בעל עדיפות 1 לרוץ, הוא יגדיל את זמן הריצה עד לqi , בקפיצות של 1. בסיום הזמן שלו יעבור המעבד לרשות התהליך בעל עדיפות 3 שבכל פסיקת שעון יגדיל את הזמן שחלף בקצב גדול פי שלושה, ועל כן יגיע בשליש זמן של פסיקות שעון לערך qi ותתבצע החלפת הקשר לאחריה. מכאן שעל כל פסיקת שעון של תהליך 3 יתבצעו 3 פסיקות שעון עבור תהליך 1. היחס הוא 1:3 כלומר 75 אחוז עבור עדיפות 1 ו25 אחוז עבור תהליך 3, כאשר החלפת הקשר תתבצע עבור תהליך לאחר qi (מעוגל) פסיקות שעון, בעוד עבור תהליך 3 יתבצעו לאחר qi/3 (מעוגל) פסיקות שעון.

ד. באלגוריתמי הזימון SCHED_OTHER שלמדנו (בתרגול ובהרצאה), המערכת מחשבת לכל תהליך עדיפות דינמית כדי להבדיל בין תהליכים שהם IO-Bound לתהליכים שהם CPU-Bound. באלגוריתם הנ"ל, אין הפרדה כזו. כיצד בכל זאת מתעדפת המערכת תהליכי IO-Bound כראוי?

השימוש בשדה vruntime הוא הדרך בה מגולם התעדוף של תהליכים אינטראקטיביים IO-bound לעומת תהליכים חישוביים. לאור העובדה שהתנאי הבא שנבחר לרוץ הוא בעל vruntime מינימלי, יהיה תעדוף לבחור את התהליך שרץ הכי מעט זמן – תכונה המאפיינת תהליכים אינטרקטיביים שמוותרים מהר על המעבד בדרך כלל לפני הצורך בהחלפת הקשר.

ה. באיזו בעיה היינו עלולים להיתקל אם לא היה נעשה שימוש בקבוע `min_granulartiy`?

במידה ולא היינו עושים שימוש בקבוע, הזמן שהיה מוקצה לריצת כל תהליך לא היה חסום מלרע. על כן בעת עומס רב על המערכת הקוואנטום שהיה ניתן לכל תהליך היה מתחלק לשברים כה קטנים, עד שתהליכים לא היו מצליחים לבצע ולו מספר פקודות בודדות והיו מאולצים לבצע החלפת הקשר. מצב זה הופך בעייתי כאשר הזמן המוקצה לריצת תהליך הוא כה קטן, כך שהנצילות של המעבד פוחתת לאור העובדה שנתח זמן גדול יותר מוקצה לפקודות שאינן מקדמות את סיום ריצת התהליכים.

שימו לב: הסעיף הבא מתייחס למערכת כפי שנלמדה בתרגולים. בפרט, אלגוריתם זימון המשימות הוא זה שנלמד בביתה ולא האלגוריתם שהוזכר לעיל.

ו. נניח כי תהליך מסויים מודע לזמן בו מתרחשת פסיקת שעון והוא מסוגל לבחור להריץ קוד כרצונו בדיוק לפני/אחרי שמתקבלת פסיקת שעון. כיצד יכול התהליך לנצל זאת כדי "לרמות" את אלגוריתם הזימון?

אלגוריתם הזימון מבוסס על הביקורת על התהליכים שרצים ברגע שמגיעה פסיקת שעון. ברגע שמגיעה פסיקת שעון תהליך שמצוי במעבד מאבד מהזמן שהוקצה לו פסיקת שעון אחת. ידע אודות הזמן בו מגיעה פסיקת שעון עשוי לאפשר התחמקות מביקורת זו, על ידי ויתור על המעבד בדיוק בזמן כדי שלא יילקח נתח זמן מאותו תהליך. בנוסף לעובדה שלא יופחת לו זמן, גם יישמר לו הsleep_average, מדד שיקנה לו עדיפות דינמית גדולה יותר במחזור הבא, ועל כן הרווח הוא גם לטווח הארוך. במידה והעומס רב במערכת, בייחוד אם אלו תהליכים בעדיפות כמו שלו הויתור על המעבד יגרום לתהליך לחזור לסוף התור ולהמתין זמן שעשוי להיות רב, אבל במידה ותור התהליכים פנוי, יוכל לנצל זאת באופן מיטבי על ידי ריצה רק בין פסיקות שעון – אנלוגיה למשחק הילדות האהוב 1 2 3 דג מלוח.

כשאתה תהליך שלא קיבל מספיק time slice ועכשיו
אתה חוזר לנקום

Credit to: **Yoray Hammer**

Submission Format

1. Only **typed** submissions in **PDF** format will be accepted. Scanned handwritten submissions will not be graded.
2. The dry part submission must contain a single PDF file named with your student IDs –

   DHW2_123456789_300200100.pdf
3. The submission should contain the following:
   a. The first page should contain the details about the submitters - Name, ID number and email address.
   b. Your answers to the dry part questions.
4. Submission is done electronically via the course website, in the **HW2 – Dry** submission box.

Grading

1. **All** question answers must be supplied with a **full explanation**. Most of the weight of your grade sits on your **explanation** and **evident effort**, and not on the absolute correctness of your answer.
2. Remember – your goal is to communicate. Full credit will be given only to correct solutions which are **clearly** described. Convoluted and obtuse descriptions will receive low marks.

Questions & Answers

- The Q&A for the exercise will take place at a public forum Piazza **only**. Please **DO NOT** send questions to the private email addresses of the TAs.
- Critical updates about the HW will be published in **pinned** notes in the piazza forum. These notes are mandatory and it is your responsibility to be updated.

A number of guidelines to use the forum:

- Read previous Q&A carefully before asking the question; repeated questions will probably go without answers
- Be polite, remember that course staff does this as a service for the students
- You're not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you have to discuss such a matter, please come to the reception hour
- When posting questions regarding **hw2**, put them in the **hw2** folder

Late Days

Please **DO NOT** send postponement requests to the TA responsible for this assignment. Only the **TA in charge** can authorize postponements. In case you need a postponement, please fill out the attached form : https://goo.gl/forms/HDFZz3MMtmZxvgXg2