Operating Systems – 234123

**Homework Exercise 3 – Dry**

Teaching Assistant in charge:

Michael Ezra

Assignment Subjects & Relevant Course material

**Modules, Scheduling, Context Switch**

**Recitations 6-8, Lectures 6-8**

# 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 –

**DHW3\_123456789\_300200100.pdf**

1. The submission should contain the following:
   1. The first page should contain the details about the submitters - Name, ID number and email address.
   2. Your answers to the dry part questions.
2. Submission is done electronically via the course website, in the **HW3 – 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 **hw3**, put them in the **hw3** 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/eW76r9cRNPTw9vAW2>

# חלק 1 - החלפת הקשר (50 נק')

חלק מהשאלות הן שאלות רב-ברירה (שאלות אמריקאיות) כאשר לכל שאלה יש **בדיוק תשובה אחת נכונה**. בשאלות אלו יש להקיף את התשובה הנכונה ביותר לדעתכם ולנמק. **תשובות לא מנומקות לא יתקבלו**.

תזכורת: בלינוקס יש שני סוגים של החלפות הקשר:

(א) החלפת הקשר יזומה - כאשר התהליך מוותר על המעבד.

(ב) החלפת הקשר כפויה - כאשר מערכת ההפעלה מפקיעה את המעבד מהתהליך.

1. (8 נק') איזה אירוע מבין הבאים יוביל בהכרח להחלפת הקשר כפויה (כלומר הפקעה)?
   1. פסיקת חומרה אשר מעירה תהליך עדיף יותר מהתהליך הנוכחי.
   2. חריגת דף בעקבות גישה לא חוקית של התהליך לזיכרון.
   3. קריאת מערכת ()wait.
   4. קריאת מערכת ()read.
   5. קריאת מערכת ()exit.

**נימוק:**

c,d,e הן קריאות מערכת שהתהליך קורא להן באופן **יזום** בקוד.

b זו החלפת הקשר יזומה מאחר והתהליך יכל **להמנע** מהגישה הלא חוקית לזכרון ולכן מאחר ו**לא נמנע** הוא **יזם** את החלפת ההקשר.

a הוא אירוע אשר התהליך **לא יכל למנוע** והוא קורה באופן **שלא תלוי** בתהליך. ולכן החלפת ההקשר שתגיע בעקבותיו תהיה כפויה.

תזכורת: גרעין לינוקס עד גרסה 2.6 אינו ניתן להפקעה --- non-preemptible kernel.

1. (8 נק') מה משמעות המושג "גרעין שאינו ניתן להפקעה"?

ישנן מערכות הפעלה בהן הגרעין לא מתיר להפקיע את המעבד מתהליכים שנמצאים במצב גרעין. לגרסאות אלו חסרונות מרכזיים:

* קוד הקרנל יכול להכנס ללופ אינסופי, והמערכת תקרוס מאחר ותהליך הקרנל הספציפי לעולם לא יסתיים.
* קוד קרנל מסויים עלול להיות מאוד איטי לביצוע, ויתפוס את המעבד להרבה זמן עד סיום ביצועו ובכך גורם לפגיעה בביצועים.

לגרעין שאינו ניתן להפקעה יש מספר חסרונות, לדוגמה פגיעה בביצועים של תהליכים מסוימים. פרדי, חובב מוזיקה ומשתמש לינוקס גרסה 2.4, סובל מבעיה כזו כאשר הוא מאזין למוזיקה בזמן שהוא מקמפל קוד של תרגילי בית אבל שומע "קפיצות" במוזיקה. לצורך המשך השאלה, נסביר בקצרה איך עובד נגן מוזיקה:

קובץ אודיו ("שיר") מורכב מהרבה דגימות (samples) שנשלחות להתקן חומרה מיוחד -- כרטיס קול -- אשר ממיר את המידע הדיגיטלי (הדגימות) לאות אנלוגי ומשמיע אותו ברמקולים. כרטיס הקול מכיל חוצץ (buffer) קטן יחסית, למשל בגודל 64KB, ולכן נגן המוזיקה לא יכול להעביר את הקובץ כולו בבת אחת. נגן המוזיקה ממלא את החוצץ (במשך X מילי-שניות) ואז מוותר על המעבד ועובר להמתין בזמן שכרטיס הקול קורא את המידע מהחוצץ (במשך Y מילי-שניות). מעט לפני שהחוצץ מתרוקן, כרטיס הקול שולח פסיקה למעבד, אשר מעירה את נגן המוזיקה כדי שימלא שוב את החוצץ, וחוזר חלילה.

נתון כי:

* החוצץ של כרטיס הקול הוא בגודל 64KB.
* תדירות הדגימות ברצועת השמע היא 44.1KHz.
* כל דגימה מכילה מידע על שני ערוצים (stereo) ברוחב 16 ביט כל אחד.
* זמן הקריאה מהדיסק הוא הדומיננטי ביותר בפעולת נגן המוזיקה.
* רוחב פס של דיסק הוא 10MB/s.

1. (6 נק') מהו משך הזמן X?

X= זמן מילוי החוצץ

1. (6 נק') מה משך הזמן Y? כלומר, מהו משך הזמן שבו יכולה מערכת ההפעלה להריץ תהליכים אחרים לפני שיהיה עליה לחזור ולמלא את החוצץ כך שהמשתמש לא ישמע "קפיצות"?

Y= זמן של המעבד עד שיצטרך למלא מחדש את החוצץ.

*כמו שניתן לראות ולכן ניתן להגדיר זמן*

*בו המעבד יצטרך למלא שוב את החוצץ. כלומר:*

**בשאלות הבאות הניחו כי:**

* במחשב של פרדי רצים רק שני תהליכים: נגן המוזיקה + קומפילציה של קוד.
* הגרעין אינו ניתן להפקעה.
* תהליך הקומפיילר משתמש הרבה בשירותי מערכת ההפעלה.
* במחשב של פרדי יש מעבד יחיד.

1. )6 נק') מה אופיו של תהליך נגן המוזיקה, חישובי או אינטראקטיבי? מה לגבי תהליך הקומפיילר האם הוא חישובי או אינטראקטיבי?

נתון כי:

* שני התהליכים הם בעלי ערך (ערך ברירת המחדל).
* תהליך הקומפילציה רץ הרבה על המעבד ויוצא מעט להמתנה.
* *תהליך נגן המוזיקה הוא אינטראקטיבי מאחר והמעבד עובד על תהליך המוזיקה במשך 6.25ms ולאחר מכן כרטיס הקול עובד במשך 371.5ms עד שתהליך המוזיקה צריך שוב למלא את החוצץ. כלומר, התהליך דורש פעילות באופן תכוף כדי להקטין את זמן התגובה ויוצא הרבה להמתנה.*
* *תהליך הקומפיילר הוא תהליך חישובי מאחר והוא רץ הרבה על המעבד ויוצא מעט להמתנה. בנוסף, הוא משתמש הרבה בשירותי מערכת ההפעלה ולכן מאחר והגרעין לא ניתן להפקעה הוא עלול להיות זמן רב במצב גרעין ובכך אף להגדיל את שימושו במעבד.*

1. (8 נק') מדוע נגן המוזיקה סובל מגרעין שאינו ניתן להפקעה?

*כי הקומפיילר משתמש הרבה בשירותי מערכת ההפעלה, ולכן הוא עלול להיות בתוך קריאת מערכת מאוד ארוכה שפועלת במצב הגרעין ואינה ניתנת להפקעה. בכך הקומפיילר יתפוס את המעבד להרבה זמן, שעלול לגלוש מעבר לזמן Y שיש למעבד להריץ את תהליך הקומפיילר לפני שיצטרך לטעון את הדגימות הבאות של קובץ הסאונד לחוצץ, בכך יווצר מצב שכרטיס הקול יסיים לרוקן את החוצץ לפני שהמעבד יתחיל למלא אותו, ולא יהיה לו מה לשדר- בכך תווצר "קפיצה" בסאונד.*

החברים של פרדי ניסו לעזור לו לפתור את הבעיה.

בריאן הציע לשפר את העדיפות של נגן המוזיקה ע"י הקטנת הערך nice.

רוג'ר הציע להפוך את נגן המוזיקה לתהליך זמן-אמת במדיניות SCHED\_RR.

ג'ון הציע להפוך את נגן המוזיקה לתהליך זמן-אמת במדיניות SCHED\_FIFO.

1. (8 נק') מי מהחברים של פרדי הציע דרך שתצמצם את הבעיה?
   1. אף אחד מבין החברים.
   2. רק בריאן.
   3. רק רוג'ר.
   4. רק ג'ון.
   5. רק רוג'ר וג'ון.
   6. כל שלושת החברים.

*בריאן- לא יעזור מאחר והגרעין אינו ניתן להפקעה ולכן לא נפתרה בעיית ה"התעכבות" של תהליך הקומפיילר במצב הגרעין.*

*רוג'ר וג'ון גם כן לא יעזרו.*

*נסמן קומפילציה ב-C ומוזיקה בM.*

*C יכול להכנס לפעילות ארוכה במצב גרעין בכל מקרה ולעכב את M בלי קשר להיותו של M תהליך real-time מאחר והגרעין לא ניתן להפקעה.(אם C נכנס קודם אז הוא יעכב את M כאמור, ואם M נכנס קודם למעבד, אז בעת שM יצא להמתנה C ייכנס ויעכב את M כאמור).*

# חלק 2 - זימון תהליכים (50 נק')

בתרגיל זה ננסה לבחון לעומק את אלגוריתם התזמון של לינוקס לתהליכים רגילים שלמדתם בתרגול - CFS .

1. (7 נק') נתונה מערכת בעלת n תהליכים עם משקלים בהתאמה.   
   נניח כי בתחילת ה-Epoch לכל התהליכים אותו זמן וירטואלי ((.

תחת ההנחה שכל תהליך מנצל את כל הקוונטום שלו, מהו ההפרש הגדול ביותר האפשרי בין הזמן הווירטואלי המקסימלי והזמן הווירטואלי המינימלי של התהליכים בסוף ה-Epoch?

מכיוון שכל תהליך מנצל את כל הקוואטום שלו מתקיים ולכן לכל i:

ולכן ההפרש האפשרי הגדול ביותר הוא 0.

1. (7 נק') כזכור, בתרגול ראיתם נוסחה לקצב התקדמות הזמן הווירטואלי ונוסחה לחישוב הקוונטום של כל תהליך. ננסה לבחון את הקשר בין 2 הנוסחאות:  
     
   אלגוריתם CFS שואף שהזמן הווירטואלי בין כל התהליכים יהיה זהה, ובפרט שבסוף כל Epoch לכל התהליכים יהיה את אותו זמן וירטואלי.   
     
   הראו, שאם הזמן הווירטואלי מתקדם ביחס הפוך לעדיפות התהליך:   
   (עבור כלשהו), וגודל ה-Epoch הוא קבוע , **אזי** הדרישה שהזמן הוירטואלי של כל התהליכים בסוף ה-Epoch זהה, גוררת את הנוסחה שראינו באלגוריתם CFS לחישוב הקוונטום של כל תהליך ().

מכיוון שהזמן הוירטואלי של כל התהליכים בסוף ה-Epoch זהה מתקיים לכל i:

בהנחה שכל תהליך מנצל את כל הקוואטום שלו (אחרת לא יתקיים שיוויון של הזמן הווירטואלי בסוף הEpoch) מתקיים ולכן:

1. (24 נק') אמיר, מומחה עולמי לאלגוריתמי תזמון החליט לנסות לשפר את אלגוריתם CFS המוכר. אמיר, הציע שיפורים שונים לאלגוריתם. בכל אחד מהסעיפים הבאים, מוצע שינוי/שיפור לאלגוריתם CFS. כתוב חיסרון שנקבל כתוצאה מאותו שינוי.  
   1. (4 נק') תהליך שחזר מהמתנה, ישאר עם אותו זמן וירטואלי שהיה לו כשהוא יצא להמתנה.

במקרה בו רצים תהליכים A ו-B במקביל ואחד מהם יצא להמתנה ארוכה, אז שהוא יחזור תתבצע הרעבה של התהליך שני.

* 1. (4 נק') שימוש במבנה נתונים של רשימה במקום עץ אדום-שחור.

הוצאה של איבר לא תשתנה מכיוון שהאיבר בעל הזמן הוירטואלי הקטן ביותר יהיה בראש הרשימה, אולם הכנסה של איבר לרשימה תיקח O(n) זמן במקום O(log(n)) זמן מכיוון שהזמן הוירטואלי שיהיה לו אינו בהכרח הגדול ביותר ( בהנחה שהרשימה ממויינת אם היא לא ממוינת הוצאה של איבר מהרשימה תיקח O(n) זמן במקום O(log(n)))

* 1. (4 נק') הסרת המינימום על גודל הקוונטוםשל תהליך (min\_granularity)

אם גודל הקוואנטום יהיה בסדר גודל של זמן החלפת הקשר(או קטן ממנו) אז יהיו הרבה יותר החלפות הקשר עד רמה של פגיעה בביצועי המחשב

* 1. (4 נק') הגדרת הקוונטום של כל תהליך, להיות בלתי תלוי במספר התהליכים, ובפרט (עבור קבוע חיובי כלשהו)

מערכת ההפעלה לא תוכל לשלוט בגודל הEpoch וכך יווצר מצב שבריבוי תהליכים יעבור הרבה זמן בין קוואנטום לקוואנטום של אותו תהליך

* 1. (4 נק') שינוי קצב התקדמות הזמן הוירטואלי להיות מהצורה

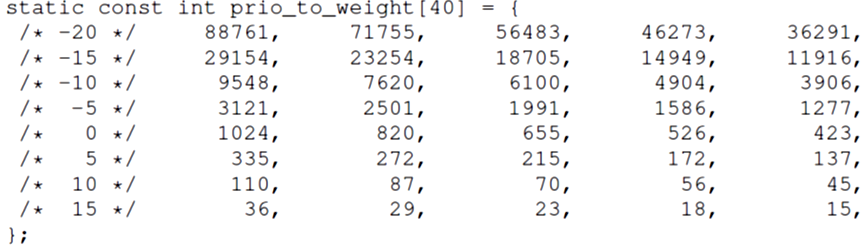
הגודל היחסי בין הזמן הוירטואלי בין שני תהליכים לא יהיה פרופורציונאלי הפוך למשקלים ביניהם ולא יישמר היחס בין זמני הריצה האמיתיים שלהם

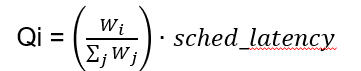
* 1. (4 נק') תהליכים חדשים מגיעים למערכת עם זמן וירטואלי השווה לזמן הווירטואלי הממוצע של כל התהליכים במערכת.

אם קיימים תהליכים A ו-B שזמן הריצה הוירטואלי שלהם שווה אז אם אחרי שB רץ קוואנטום יתחילו להתווסף הרבה תהליכים אז תווצר הרעבה של תהליך B מכיוון שהוא בעל הזמן הוירטואלי הגדול ביותר וכל תהליך שייכנס יהיה בעל זמן וירטואלי קטן ממנו

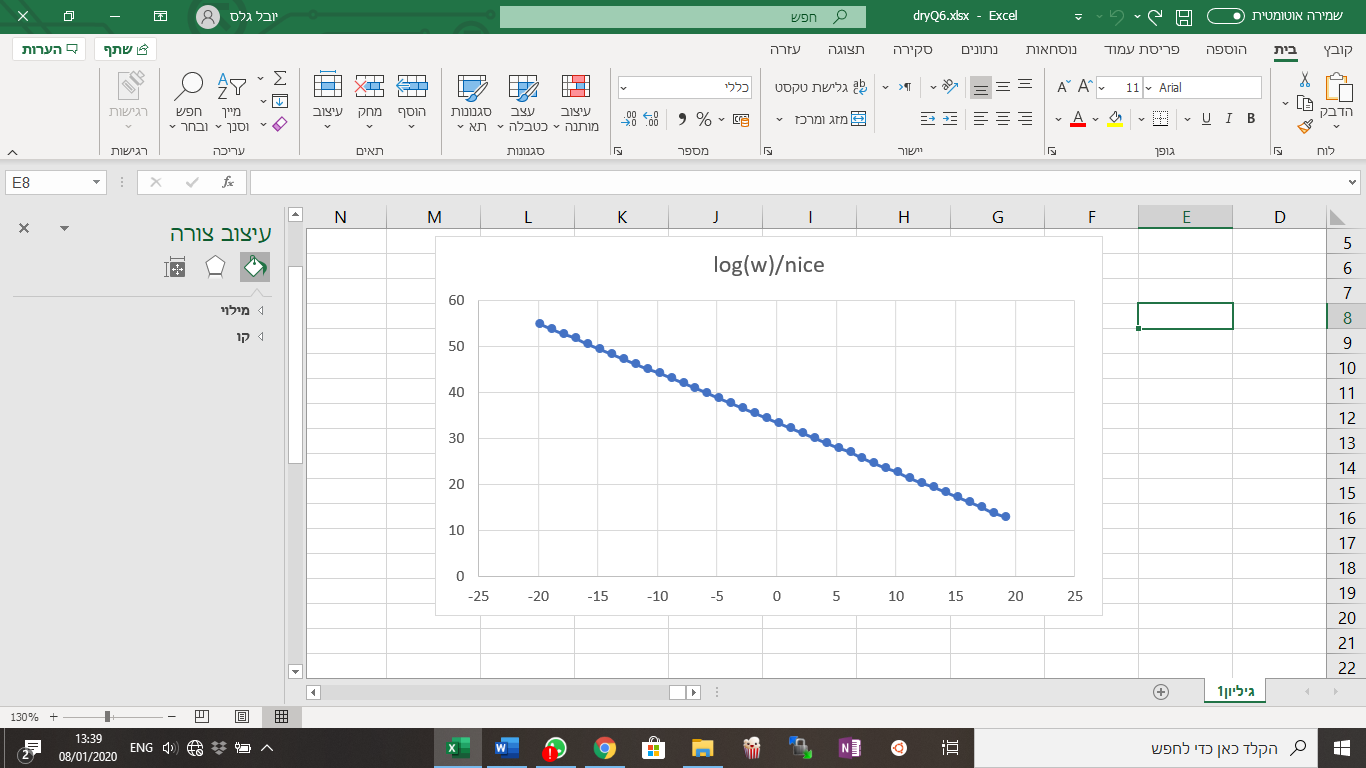
1. (6 נק') על מנת להפחית את התקורה בעת החלפת הקשר כתוצאה מהחלפת אזורי הזיכרון וניקוי ה-TLB, אמיר (אותו אחד משאלה קודמת - הוא אוהב להציע הצעות), הציע שתהליכים שמוכנים לריצה, וחולקים את אותו מרחב זיכרון עם התהליך שרץ אחרון על המעבד, יקבלו עדיפות על המעבד וירוצו לפני כל התהליכים האחרים. אילן, סטודנט למערכות הפעלה, טוען שהפתרון לא מדהים במיוחד, ושבצורה כזאת תהליך יכול לגנוב זמן מעבד. **הסבירו** כיצד.

תהליך ייצר לעצמו חוט לקראת סוף הקוואנטום שלו וכך החוט בעל אזורי זכרון משותפים עם התהליך הקודם ירוץ ראשון והחוט ייצר לעצמו עוד חוט וכו' וכך התהליך יוכל לגנוב לעצמו זמן מעבד

1. (6 נק') נזכיר, שבאלגוריתם CFS, לכל תהליך יש משקל שנקבע לו בהתאם לערך ה-nice שלו ():  
     
     
   כמו כן, ראינו כי גודל הקוונטום של תהליך, נמצא ביחס ישר למשקל התהליך:



לכן, לתהליך בעל ערך nice **נמוך**, יש משקל **גבוה**, וכתוצאה מכך קוונטום **ארוך** יותר. בסעיף זה, נרצה לבחון את הרציונל מאחורי בחירת ערכי המשקלים.

* 1. (3 נק') ציירו (בעזרת אקסל או כל תוכנה אהובה לבחירתך) גרף של לוג משקל התהליך (בבסיס 1.23), כפונקציה של ערך הnice המתאים לה. **צרף תמונה של הגרף לפתרון.**
  2. (3 נק') על סמך הגרף, מהו הקשר המתמטי בין ערך הnice למשקל התהליך?